N O R M A I T A L I A N A CEI

Norma Italiana

CEI EN 50128

Data Pubblicazione	*Edizione*
2002-04	**Prima**
Classificazione	*Fascicolo*
9-72	**6421**

Titolo

Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane
Sistemi di telecomunicazione, segnalamento ed elaborazione - Software per sistemi ferroviari di comando e di protezione

Title

Railway applications

Communications, signalling and processing systems - Software for railway control and protection systems

APPARECCHIATURE ELETTRICHE PER SISTEMI DI ENERGIA E PER TRAZIONE

COMITATO ELETTROTECNICO ITALIANO

CNR CONSIGLIO NAZIONALE DELLE RICERCHE • **AEI** ASSOCIAZIONE ELETTROTECNICA ED ELETTRONICA ITALIANA

SOMMARIO

La presente Norma specifica le procedure ed i requisiti tecnici per lo sviluppo del software, per impiego nelle applicazioni ferroviarie con livello di sicurezza superiore allo 0.

DESCRITTORI • *DESCRIPTORS*

Ferrovie • *Railways;* Tranvie • *Tramways;* Filovie • *Trolley buses;* Metropolitane • *Underground;* Software • *Software;*

COLLEGAMENTI/RELAZIONI TRA DOCUMENTI

Nazionali	(UTE) CEI EN 50126:2000-03;
Europei	(IDT) EN 50128:2001-03;
Internazionali	
Legislativi	

INFORMAZIONI EDITORIALI

Norma Italiana	CEI EN 50128	*Pubblicazione*	Norma Tecnica	*Carattere Doc.*	
Stato Edizione	In vigore	*Data validità*	2002-6-1	*Ambito validità*	Europeo
Varianti	Nessuna				
Ed. Prec. Fasc.	Nessuna				
Comitato Tecnico	9-Trazione				
Approvata dal	Presidente del CEI	*in Data*	2002-3-13		
	CENELEC	*in Data*	2000-11-1		
Sottoposta a	inchiesta pubblica come Documento originale			*Chiusa in data*	2000-7-31
Gruppo Abb.	3	*Sezioni Abb.*	B		
ICS	29.280; 45.060.10;				
CDU					

LEGENDA

(UTE) La Norma in oggetto deve essere utilizzata congiuntamente alle Norme indicate dopo il riferimento (UTE)
(IDT) La Norma in oggetto è identica alle Norme indicate dopo il riferimento (IDT)

Copia concessa a ANSALDO S.F. SPA e ANSALDO BREDA in data 02/10/2007 da CEI-Comitato Elettrotecnico Italiano

Europäische Norm • Norme Européenne • European Standard • Norma Europea
EN 50128

Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane
Sistemi di telecomunicazione, segnalamento ed elaborazione - Software per sistemi ferroviari di comando e di protezione

Railway applications
Communications, signalling and processing systems - Software for railway control and protection systems

Applications ferroviaires
Système de signalisation, de télécommunication et de traitement - Logiciels pour systèmes de commande et de protection ferroviaire

Bahnanwendungen -
Telekommunikationstechnik - Signal- technik und Datenverarbeitungssysteme - Software für Eisenbahnsteuerungs- und Überwachungssysteme

C E N E L E C

Comitato Europeo di Normalizzazione Elettrotecnica
European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Secrétariat Central: *rue de Stassart 35, B - 1050 Bruxelles*

Copia concessa a ANSALDO S.F. SPA e ANSALDO BREDA in data 02/10/2007 da CEI-Comitato Elettrotecnico Italiano

CONTENTS

INDICE

FOREWORD

This European Standard was prepared by SC 9XA, Communication, signalling and processing systems, of Technical Committee CENELEC TC 9X, Electrical and electronic applications for railways.

The text of the draft was submitted to the formal vote and was approved by CENELEC as EN 50128 on 2000/11/01.

The following dates were fixed:

- latest date by which the EN has to be implemented at national level by publication of an identical national standard or by endorsement
  (dop) 2001/11/01
- latest date by which the national standards conflicting with the EN have to be withdrawn
  (dow) 2003/11/01

This European Standard should be read in conjunction with EN 50126: "Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)" and EN 50129: "Railway applications - Safety related electronic systems for signalling".

Annexes designated "normative" are part of the body of the standard.

Annexes designated "informative" are given for information only.

In this standard, annex A is normative and annex B is informative.

PREFAZIONE

La Presente Norma Europea è stata preparata dal SC 9XA, Communication, signalling and processing systems, del Comitato Tecnico CENELEC 9X, Electrical and electronic applications for railways.

Il testo del progetto è stato sottoposto al voto formale ed è stato approvato dal CENELEC come Norma Europea EN 50128 in data 01/11/2000.

Sono state fissate le date seguenti:

- data ultima entro la quale la EN deve essere recepita a livello nazionale mediante pubblicazione di una Norma nazionale identica o mediante adozione
  (dop) 01/11/2001
- data ultima entro la quale le Norme nazionali contrastanti con la EN devono essere ritirate
  (dow) 01/11/2003

La presente Norma Europea dovrebbe essere letta congiuntamente alla EN 50126: "Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)" e alla EN 50129: "Railway applications - Safety related electronic systems for signalling".

Gli Allegati indicati come "normativi" sono parte integrante della Norma.

Gli Allegati indicati come "informativi" sono dati solo per informazione.

Nella presente Norma, l'Allegato A è normativo e l'Allegato B è informativo.

INTRODUCTION

This Standard is part of a group of related Standards. The others are EN 50126 “Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)” and EN 50129 “Railway applications - Safety related electronic systems for signalling”. EN 50126 addresses system issues on the widest scale, while EN 50129 addresses the approval process for individual systems which may exist within the overall railway control and protection system. This Standard concentrates on the methods which need to be used in order to provide software which meets the demands for safety integrity which are placed upon it by these wider considerations.

This Standard owes much of its direction to earlier work done by Working Group 9 of IEC/TC 65. The work of WG 9 resulted in a generic standard for software for safety systems which is now part of IEC 61508. A particular aspect of the work by WG 9 is its inclusion of Software Integrity Level 0, which covers non-safety software, as well as Software Integrity Levels 1 to 4, which cover safety-related and safety-critical software. This Standard also covers all five Software Integrity Levels.

Account has also been taken of the work of the Institution of Railway Signal Engineers (the IRSE), in particular its Technical Report Number 1, which addressed the same topic.

The key concept of this European Norm is that of levels of software safety integrity. The more dangerous the consequences of a software failure, the higher the software safety integrity level will be.

This European Standard has identified techniques and measures for 5 levels of software safety integrity where 0 is the minimum level and 4 the highest level. Four of these levels, 1 to 4, refer to safety-related software, whilst level 0 refers to non safety-related software. This level has been included as normative in order to allow a smooth transition between software developments for non-safety related systems and those for safety-related systems. The required techniques and measures for each software safety integrity level and for the non safety-related level are shown in the tables. In this version, the required techniques for level 1 are the same as for level 2, and the required techniques for level 3 are the same as for level 4. This European Standard does not give guidance on which level of software integrity is appropriate for a given risk. This decision will depend upon the many factors including the nature of the application, the extent to which other systems car-

INTRODUZIONE

La presente Norma fa parte di un gruppo di Norme correlate. Le altre sono la EN 50126 “Applicazioni ferroviarie - La specificazione e la dimostrazione di Affidabilità, Disponibilità, Manutenibilità e Sicurezza (RAMS)” e la EN 50129: “Applicazioni ferroviarie - Sistemi elettronici in sicurezza per il segnalamento”. La EN 50126 tratta i problemi di sistema su più ampia scala, mentre la EN 50129 tratta i processi di approvazione di singoli sistemi che possono essere presenti nell’intero sistema ferroviario di controllo e protezione. La presente Norma si concentra sui metodi che è necessario usare al fine di fornire software che soddisfi ai requisiti di integrità della sicurezza che sono determinati da queste più ampie considerazioni.

La presente Norma deve molti dei suoi orientamenti al lavoro iniziale compiuto dal WG 9 dell’IEC/TC 65. Dal lavoro del WG 9 è risultata una norma generica per il software dei sistemi in sicurezza che ora fa parte della IEC 61508. Un aspetto particolare del lavoro del WG 9 è l’introduzione del Livello 0 di Integrità del Software, che riguarda il software non in sicurezza, e dei Livelli di Integrità del Software da 1 a 4, che riguardano il software con caratteristiche di sicurezza e quello critico per la sicurezza. Anche la presente Norma tratta tutti i cinque Livelli di Integrità del Software.

Si è tenuto anche conto del lavoro dell’Institution of Railway Signal Engineers (l’IRSE), in particolare del suo Technical Report Number 1, che trattava lo stesso argomento.

Il concetto chiave della presente Norma Europea è quello dei livelli di integrità della sicurezza del software. Più pericolose saranno le conseguenze di un malfunzionamento del software, più elevato sarà il livello di integrità del software.

La presente Norma Europea ha identificato tecniche e provvedimenti per i 5 livelli di integrità della sicurezza del software, dove 0 è il livello minimo e 4 il livello più elevato. Quattro di questi livelli, da 1 a 4, fanno riferimento al software in sicurezza, mentre il livello 0 fa riferimento al software non in sicurezza. Questo livello è stato introdotto come normativo al fine di consentire una transizione graduale tra gli sviluppi del software per sistemi non in sicurezza e quelli per i sistemi in sicurezza. Le tecniche e i provvedimenti per ciascun livello di integrità della sicurezza del software e per il livello non di sicurezza sono mostrati nelle tabelle. In questa versione, le tecniche richieste per il livello 1 sono le stesse del livello 2, e le tecniche richieste per il livello 3 sono le stesse del livello 4. La presente Norma Europea non fornisce una guida circa il livello di integrità del software appropriato per un dato rischio. Questa decisione dipenderà da molti fattori fra cui la natura dell’applicazione, la misura con la quale altri

ry out safety functions and social and economic factors.

It is the function of EN 50126 and EN 50129 to specify the safety functions allocated to software.

This European Standard specifies those measures necessary to achieve these requirements. The process is illustrated in Figure 1.

EN 50126 and EN 50129 require that a systematic approach be taken to:

i) identifying hazards, risks and risk criteria;
ii) identifying the necessary risk reduction to meet the risk criteria;
iii) defining an overall System Safety Requirements Specification for the safeguards necessary to achieve the required risk reduction;
iv) selecting a suitable system architecture;
v) planning, monitoring and controlling the technical and managerial activities necessary to translate the System Safety Requirements Specification into a Safety-Related System of a validated safety performance (or safety integrity).

As decomposition of the specification into a design comprising safety-related systems and components takes place, further allocation of safety integrity levels is performed. Ultimately this leads to the required software safety integrity levels.

The current state-of-the-art is such that neither the application of quality assurance methods (so-called fault avoiding measures) nor the application of software fault tolerant approaches can guarantee the absolute safety of the system. There is no known way to prove the absence of faults in reasonably complex safety-related software, especially the absence of specification and design faults.

The principles applied in developing high integrity software include, but are not restricted to:

- top-down design methods;
- modularity;
- verification of each phase of the development lifecycle;
- verified modules and module libraries;
- clear documentation;
- auditable documents; and
- validation testing.

These and related principles must be correctly applied. This standard specifies the level of assurance required to demonstrate this at each software safety integrity level.

After the System Safety Requirements Specification, which identifies all safety functions allocated to software and determines the system safety integrity level, has been obtained or produced,

sistemi adempiono funzioni di sicurezza e fattori sociali ed economici.

È funzione della EN 50126 e della EN 50129 specificare le funzioni di sicurezza assegnate al software.

La presente Norma Europea specifica quei provvedimenti necessari a conseguire questi requisiti. Il processo è illustrato in Figura 1.

La EN 50126 e la EN 50129 richiedono che sia adottato un approccio sistematico per:

i) identificare pericoli, rischi e criteri di rischio;
ii) identificare la riduzione di rischio necessaria a soddisfare i criteri di rischio;
iii) definire una Specifica dei Requisiti di Sicurezza del Sistema per le contromisure necessarie a conseguire la riduzione di rischio richiesta;
iv) scegliere un'adatta architettura di sistema;
v) pianificare, sorvegliare e controllare le attività tecniche e di gestione necessarie a trasformare la Specifica dei Requisiti di Sicurezza del Sistema in un Sistema in Sicurezza che abbia una prestazione di sicurezza (o integrità della sicurezza) validata.

Quando ha luogo una scomposizione della specificazione in un progetto che comprende sistemi e componenti in sicurezza, viene eseguita un'ulteriore assegnazione dei livelli di integrità della sicurezza. Alla fine ciò porta ai richiesti livelli di integrità della sicurezza del software.

L'attuale stato dell'arte è tale che né l'applicazione dei metodi di assicurazione della Qualità (i cosiddetti provvedimenti per evitare difetti [fault avoiding measures]) né l'applicazione di software con approcci tolleranti il difetto [fault tolerant] possono garantire l'assoluta sicurezza del sistema. Non vi è alcun modo noto di comprovare l'assenza di difetti in un software in sicurezza ragionevolmente complesso, specialmente l'assenza di difetti di specificazione e di progettazione.

I principi applicati nello sviluppo di software ad elevata integrità comprendono, ma non sono limitati a:

- metodi di progettazione top-down;
- modularità;
- verifica di ogni fase del ciclo di vita di sviluppo;
- moduli verificati e librerie di moduli;
- documentazione chiara;
- documenti revisionabili; e
- prove di validazione.

Questi principi e quelli collegati devono essere applicati correttamente. La presente norma specifica il livello di confidenza richiesto per dimostrarlo per ogni livello di integrità della sicurezza del software.

Dopo che è stata ottenuta o prodotta la Specifica dei Requisiti di Sicurezza del Sistema, che identifica tutte le funzioni di sicurezza attribuite al software e che determina il livello di integrità della

the functional steps in the application of this European Standard are shown in Figure 2 and are as follows:

i) define the Software Requirements Specification and in parallel consider the software architecture. the software architecture is where the basic safety strategy is developed for the software and the software safety integrity level (clauses 5, 8 and 9);
ii) design, develop and test the software according to the Software Quality Assurance Plan, software safety integrity level and the software lifecycle (clause 10);
iii) integrate the software on the target hardware (clause 12);
iv) validate the software (clause 13);
v) if software maintenance is required during operational life then re-activate this European Standard as appropriate (clause 16).

A number of activities run across the software development. These include verification (clause 11), assessment (clause 14) and quality assurance (clause 15).

Requirements are given for systems which are configured by application data (clause 17).

Requirements are also given for the competency of staff involved in software development (clause 6).

The standard does not mandate the use of a particular software development lifecycle. However a recommended lifecycle and documentation set are given (clause 7 and Figures 3 and 4).

Tables have been formulated ranking various techniques/measures against the 5 software safety integrity levels. The tables are in annex A. Cross-referenced to the tables is a bibliography giving a brief description of each technique/measure with references to further sources of information. The bibliography is in annex B.

sicurezza del sistema, i passi funzionali nell'applicazione della presente Norma Europea sono mostrati nella Figura 2 e sono i seguenti:

i) definire la Specifica dei Requisiti del Software e in parallelo considerare l'architettura software. L'architettura software è dove viene sviluppata la strategia di sicurezza di base per il software ed il livello di integrità della sicurezza del software (art. 5, 8 e 9);
ii) progettare, sviluppare e provare il software secondo il Piano di Assicurazione della Qualità del Software, il livello di integrità della sicurezza del software e il ciclo di vita del software (art. 10);
iii) integrare il software nell'hardware cui è destinato (art. 12);
iv) validare il software (art. 13);
v) se è richiesta la manutenzione del software durante la vita operativa va applicata nuovamente la presente Norma Europea in modo appropriato (art. 16).

Numerose attività sono attivate durante lo sviluppo del software. Esse comprendono la verifica (art. 11), la valutazione (art. 14) e l'assicurazione qualità (art. 15).

Vengono stabiliti requisiti per sistemi che sono configurati dai dati dell'applicazione (art. 17).

Vengono stabiliti anche requisiti per la competenza del gruppo di lavoro coinvolto nello sviluppo del software (art. 6).

La norma non impone l'uso di un particolare ciclo di vita di sviluppo del software. Tuttavia vengono dati un ciclo di vita raccomandato e un insieme di documentazioni (art. 7 e Figure 3 e 4).

Sono state elaborate tabelle che classificano varie tecniche/provvedimenti per i 5 livelli di integrità della sicurezza del software. Le tabelle sono nell'Allegato A. Vi è una bibliografia correlata con le tabelle che da una breve descrizione di ogni tecnica/provvedimento con riferimento ad ulteriori fonti di informazione. La bibliografia è nell'Allegato B.

1 SCOPE

CAMPO D'APPLICAZIONE

1.1 This European Standard specifies procedures and technical requirements for the development of programmable electronic systems for use in railway control and protection applications. It is aimed at use in any area where there are safety implications. These may range from the very critical, such as safety signalling to the non-critical, such as management information systems. These systems may be implemented using dedicated microprocessors, programmable logic controllers, multiprocessor distributed systems, larger scale central processor systems or other architectures.

La presente Norma Europea specifica le procedure e i requisiti tecnici per lo sviluppo di sistemi elettronici a logica programmabile, utilizzati nelle applicazioni ferroviarie di controllo e protezione. Essa è utilizzata in qualsiasi settore dove vi sono implicazioni riguardanti la sicurezza. Queste possono variare da quelle molto critiche, come la sicurezza per il segnalamento, a quelle non critiche, come i sistemi per la gestione di informazioni. Questi sistemi possono essere implementati usando microprocessori dedicati, controllori a logica programmata, sistemi distribuiti a multiprocessore, sistemi a processore centrale di dimensioni più ampie o altre architetture.

1.2 This European Standard is applicable exclusively to software and the interaction between software and the system of which it is part.

La presente Norma Europea è applicabile esclusivamente al software e all'interazione tra il software e il sistema del quale esso fa parte.

1.3 Software safety integrity levels above zero are for use in systems in which the consequences of failure could include loss of life. Economic or environmental considerations, however, may also justify the use of higher software safety integrity levels.

Livelli di integrità della sicurezza del software superiori allo 0 sono utilizzati in sistemi nei quali le conseguenze di un mancato funzionamento potrebbero comportare la perdita di vite umane. Considerazioni economiche o ambientali possono, tuttavia, giustificare anche l'uso di livelli più alti di integrità della sicurezza del software.

1.4 This European Standard applies to all software used in development and implementation of railway control and protection systems including:

application programming;
operating systems;
support tools;
firmware.

La presente Norma Europea si applica a tutto il software usato nello sviluppo ed nell'implementazione di sistemi ferroviari di controllo e protezione, compreso:

programmazione applicativa;
sistemi operativi;
strumenti (tools) di supporto;
firmware.

Application programming comprises high level programming, low level programming and special purpose programming (for example: Programmable Logic Controller ladder logic).

La programmazione applicativa comprende, programmazione ad alto livello, programmazione a basso livello e programmazione per usi speciali (per esempio: Controllori a Logica programmabile a logica ladder.

1.5 The use of standard, commercially available software and tools is also addressed in this European Standard.

Nella presente Norma Europea viene anche definito l'uso di software e di strumenti standard disponibili in commercio.

1.6 This European Standard also addresses the requirements for systems configured by application data.

La presente Norma Europea definisce anche i requisiti per i sistemi configurati mediante dati applicativi.

1.7 This European Standard is not intended to address commercial issues. These should be addressed as an essential part of any contractual agreement. All the clauses of this European Standard will need careful consideration in any commercial situation.

La presente Norma Europea non intende definire argomenti commerciali. Questi dovrebbero essere trattati come una parte indispensabile di ogni accordo contrattuale. Tutti gli articoli della presente Norma Europea necessiteranno di attenta considerazione in ogni situazione commerciale.

1.8 This European Standard is not intended to be retrospective. It therefore applies primarily to new developments and only applies in its entirety to existing systems if these are subjected to major modifications. For minor changes, only clause 16 applies.

Non è previsto che la presente Norma Europea sia retroattiva. Essa si applica quindi principalmente a nuovi sviluppi e si applica nella sua interezza ai sistemi esistenti solo se questi sono soggetti a modifiche importanti. Per modifiche minori, si applica solo l'art.16.

2 NORMATIVE REFERENCES

RIFERIMENTI NORMATIVI

This European Standard incorporates by dated or undated reference, provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this European Standard only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies (including amendments).

La presente Norma Europea incorpora, tramite riferimenti datati e non datati, indicazioni provenienti da altre pubblicazioni. Questi riferimenti normativi sono citati, dove appropriato, nel testo, e le Pubblicazioni sono elencate di seguito. Per i riferimenti datati, le successive modifiche o revisioni di ognuna di tali pubblicazioni si applicano alla presente Norma Europea solo quando incluse in essa da una modifica o revisione. In caso di riferimenti non datati, si applica l'ultima edizione della Pubblicazione indicata (comprese le modifiche).

Pubblicazione *Publication*	**Titolo** *Title*	**Norma CEI** *CEI Standard*
EN 50126	Applicazioni ferroviarie, tranviarie, filotranviarie, metropolitane - La specificazione e la dimostrazione di Affidabilità, Disponibilità, Manutenibilità e Sicurezza (RAMS) *Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)*	9-58
EN 50129(1)	Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane - Sistemi elettronici di sicurezza per il segnalamento *Railway applications - Safety related electronic systems for signalling*	9-55
EN 50159-1	Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane - Sistemi di telecomunicazione, segnalamento ed elaborazione Parte 1: Comunicazioni di sicurezza in sistemi di trasmissione di tipo chiuso *Railway applications - Communication, signalling and processing systems* *Part 1: Safety-related communication in closed transmission systems*	9-66
EN 50159-2	Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane - Sistemi di telecomunicazione, segnalamento ed elaborazione Parte 2: Comunicazioni di sicurezza in sistemi di trasmissione di tipo aperto *Railway applications - Communication, signalling and processing systems* *Part 2: Safety-related communication in open transmission systems*	9-66/2
EN ISO 9001	Sistemi qualità – Modello di assicurazione qualità nel progetto/sviluppo, produzione, installazione ed esercizio *Quality systems - Model for quality assurance in design/development, production, installation and servicing*	—
EN ISO 9000-3	Gestione della qualità e norme di assicurazione qualità – Parte 3: Linee guida per l'applicazione della ISO 9001:1994 allo sviluppo, fornitura, installazione e manutenzione del software dei calcolatori *Quality management and quality assurance standards – Part 3: Guidelines for the application of ISO 9001:1994 to the development, supply, installation and maintenance of computer software*	—

(1) In fase di progetto.
At draft stage

3 DEFINITIONS

DEFINIZIONI

For the purposes of this European Standard, the following definitions apply. For terms not defined here, the following references should be consulted in order of priority:

Per gli scopi della presente Norma Europea, si applicano le seguenti definizioni. Per i termini che qui non sono definiti, dovrebbero essere consultati, in ordine di priorità, i seguenti riferimenti:

EN ISO 8402	Gestione della qualità e assicurazione qualità – Vocabolario *Quality management and quality assurance – Vocabulary*
IEC 60050-191	Capitolo 191 del Vocabolario Elettrotecnico Internazionale: Fidatezza e qualità del servizio *International Electrotechnical Vocabulary of Chapter 191: Dependability and quality of service*
IEEE 610.12	IEEE glossario standard della terminologia dell'ingegneria del software *IEEE standard glossary of software engineering terminology*
ISO/IEC 2382	Vocabolario della Tecnologia dell'Informazione *Information Technology Vocabulary*
ISO/IEC 9126	Tecnologia dell'Informazione – Valutazione di Prodotto del Software – Caratteristiche di qualità e linee guida per il loro uso *Information Technology – Software Product Evaluation – Quality characteristics and guidelines for their use*

3.1 Assessment

Valutazione

Process of analysis to determine whether the Design Authority and the Validator have achieved a product that meets the specified requirements and to form a judgement as to whether the product is fit for its intended purpose.

Processo di analisi per determinare se l'Autorità per il Progetto e il Validatore hanno ottenuto un prodotto che rispetta i requisiti specificati e formulare un giudizio se il prodotto è adatto allo scopo stabilito.

3.2 Assessor

Valutatore

Person or agent appointed to carry out the assessment.

Persona o agente designato per eseguire la valutazione.

3.3 Availability

Disponibilità

Ability of a product to be in a state to perform a required function under given conditions at a given instant of time or over a given time interval, assuming the required external resources are provided.

La capacità di un prodotto di essere nella condizione di eseguire una funzione richiesta sotto determinate condizioni in un dato istante o in un dato intervallo di tempo, assumendo che siano fornite le risorse esterne richieste.

3.4 Commercial off-the-shelf (COTS) software

Software disponibile commercialmente (COTS)

Software defined by market-driven need, commercially available and whose fitness for purpose has been demonstrated by a broad spectrum of commercial users.

Software definito da necessità imposte dal mercato, disponibile commercialmente e la cui idoneità allo scopo è stata dimostrata da un ampio spettro di utilizzatori commerciali.

3.5 Design authority

Autorità per il progetto

Body responsible for the formulation of a design solution to fulfil the specified requirements and for overseeing the subsequent development and setting to work of a system in its intended environment.

Ente responsabile per la formulazione di una soluzione progettuale che soddisfi i requisiti specificati e per sovrintendere i conseguenti sviluppi e messa in opera di un sistema nel suo ambiente previsto.

3.6 Designer

Progettista

One or more persons assigned by the Design Authority to analyse and transform specified requirements into acceptable design solutions which have the required safety integrity.

Una o più persone designate dall'Autorità per il Progetto ad analizzare e trasformare i requisiti specificati in soluzioni progettuali accettabili che abbiano l'integrità della sicurezza richiesta.

3.7 Element

Part of a product that has been determined to be a basic unit or building block. An element may be simple or complex.

Elemento

Parte di un prodotto che è stato stabilito essere un'unità di base o un blocco costruttivo. Un elemento può essere semplice o complesso.

3.8 Error

Deviation from the intended design which could result in unintended system behaviour or failure.

Errore

Deviazione dal progetto previsto che potrebbe determinare un comportamento non previsto del sistema od un mancato funzionamento.

3.9 Failure

Deviation from the specified performance of a system. A failure is the consequence of a fault or error in a system.

Mancato funzionamento

Deviazione di un sistema dalla prestazione specificata. Un mancato funzionamento è la conseguenza di un guasto o di un errore in un sistema.

3.10 Fault

Abnormal condition that could lead to an error or a failure in a system. A fault can be random or systematic.

Guasto

condizione anormale che potrebbe portare ad un errore o ad un mancato funzionamento in un sistema. Un guasto può essere casuale o sistematico.

3.11 Fault avoidance

Use of design techniques which aim to avoid the introduction of faults during the design and construction of the system.

Tecnica per evitare i guasti

Utilizzo di tecniche di progetto che mirano ad evitare l'introduzione di guasti durante il progetto e la costruzione del sistema.

3.12 Fault tolerance

Built-in capability of a system to provide continued correct provision of service as specified, in the presence of a limited number of hardware or software faults.

Tolleranza ai guasti

Capacità intrinseca di un sistema di fornire prestazione continua e corretta di servizio come specificato, in presenza di un limitato numero di guasti hardware o software.

3.13 Firmware

Ordered set of instructions and associated data stored in a way that is functionally independent of main storage, usually in a ROM.

Firmware

Serie ordinata di istruzioni e di dati associati memorizzati in modo funzionalmente indipendente dalla memoria principale, comunemente in una ROM.

3.14 Generic software

Generic software is software which can be used for a variety of installations purely by the provision of application-specific data.

Software generico

Il software generico è un software che può essere usato per varie installazioni, fornendo unicamente i dati per la specifica applicazione.

3.15 Implementer

One or more persons assigned by the Design Authority to transform specified designs into their physical realisation.

Implementatore

Una o più persone designate dall'Autorità per il Progetto a trasformare progetti specifici nella loro realizzazione fisica.

3.16 Product

Collection of elements, interconnected to form a system, sub-system or item of equipment, in a manner which meets the specified requirements. In this European Standard, a product may be considered to consist entirely of elements of software or documentation.

Prodotto

Insieme di elementi, interconnessi per formare un sistema, sottosistema o elemento di una apparecchiatura, tale da soddisfare i requisiti specificati. Nella presente Norma Europea, un prodotto può consistere interamente di elementi di software o di documentazione.

3.17 Programmable logic controller (PLC)

Solid-state control system which has a user programmable memory for storage of instructions to implement specific functions.

Controllore a logica programmata (PLC)

Sistema di comando a stato solido che ha una memoria programmabile dall'utente per la memorizzazione di istruzioni che implementano funzioni specifiche.

3.18 Reliability

Ability of an item to perform a required function under given conditions for a given period of time.

Affidabilità

Capacità di un'elemento di eseguire una funzione richiesta in condizioni stabilite per un dato periodo di tempo.

3.19 Requirements traceability

Objective of requirements traceability is to ensure that all requirements can be shown to have been properly met.

Tracciabilità dei requisiti

L'obiettivo della tracciabilità dei requisiti è quello di assicurare che si possa dimostrare che sono stati propriamente soddisfatti tutti i requisiti.

3.20 Risk

Combination of the frequency, or probability, and the consequence of a specified hazardous event.

Rischio

Combinazione della frequenza, o probabilità, e della conseguenza di un evento pericoloso specificato.

3.21 Safety

Freedom from unacceptable levels of risk.

Sicurezza

Libertà da un inaccettabile livello di rischio.

3.22 Safety authority

Body responsible for certifying that the safety-related system is fit for service and complies with relevant statutory and regulatory safety requirements.

Autorità per la sicurezza

Ente responsabile a certificare che il sistema di sicurezza è adatto per il servizio e si conforma con i pertinenti requisiti di sicurezza statuti e regolamentati.

3.23 Safety-related software

Software which carries responsibility for safety.

Software in sicurezza

Software che comporta responsabilità per la sicurezza.

3.24 Software

Intellectual creation comprising the programs, procedures, rules and any associated documentation pertaining to the operation of a system.

Note/Nota *Software is independent of the media used for transport.*

Software

Creazione intellettuale comprendente programmi, procedure, regole ed ogni documentazione associata pertinente al funzionamento di un sistema.

Il software è indipendente dal mezzo di comunicazione usato per la trasmissione.

3.25 Software life-cycle

Activities occurring during a period of time that starts when software is conceived and ends when the software is no longer available for use. The software lifecycle typically includes a requirements phase, development phase, test phase, integration phase, installation phase and a maintenance phase.

Ciclo di vita del software

Attività che si verificano durante il periodo di tempo che inizia quando il software è concepito e finisce quando il software non è più disponibile per l'uso. Il ciclo di vita del software comprende tipicamente una fase dei requisiti, una fase di sviluppo, una fase di prova, una fase di integrazione, una fase di installazione e una fase di manutenzione.

3.26 Software maintainability

Capability of a system to be modified to correct faults, improve performance or other attributes, or adapt it to a different environment.

Manutenibilità del software

Attitudine di un sistema di essere modificato per correggere guasti, migliorare prestazioni o altri attributi, o adattarlo ad un diverso ambiente.

3.27 Software maintenance

Action, or set of actions, carried out on software after its acceptance by the final user. The aim is to improve, increase and/or correct its functionality.

Manutenzione del software

Azione, o serie di azioni, svolte sul software dopo la sua accettazione da parte dell'utente finale. Lo scopo è quello di migliorare, aumentare, e/o correggere la sua funzionalità.

3.28 Software safety integrity level

Classification number which determines the techniques and measures that have to be applied in order to reduce residual software faults to an appropriate level.

Livello di integrità della sicurezza del software

Numero di classificazione che determina le tecniche e i provvedimenti che devono essere applicati al fine di ridurre i guasti residui del software ad un livello opportuno.

3.29 System safety integrity level

Number which indicates the required degree of confidence that a system will meet its specified safety features.

Livello di integrità della sicurezza del sistema

Numero che indica il grado di confidenza richiesto che un sistema soddisferà le sue specificate caratteristiche di sicurezza.

3.30 Traceability

Degree to which a relationship can be established between two or more products of a development process, especially those having a predecessor/successor or master/subordinate relationship to one another.

Tracciabilità

Parametro con il quale può essere stabilita una relazione tra due o più prodotti di un processo di sviluppo, specialmente quelli che hanno tra loro un rapporto precedente/successivo o principale/subordinato.

3.31 Validation

Activity of demonstration, by analysis and test, that the product meets, in all respects, its specified requirements.

Validazione

Attività di dimostrazione, attraverso analisi e prove, che il prodotto soddisfa, in tutti gli aspetti, i suoi requisiti specificati.

3.32 Validator

Person or agent appointed to carry out validation.

Validatore

Persona o agente incaricato di svolgere la validazione.

3.33 Verification

Activity of determination, by analysis and test, that the output of each phase of the life-cycle fulfils the requirements of the previous phase.

Verifica

Attività che determina, attraverso analisi e prove, che l'uscita di ogni fase del ciclo di vita soddisfa i requisiti della fase precedente.

3.34 Verifier

Person or agent appointed to carry out verification.

Verificatore

Persona o agente incaricato di svolgere la verifica.

4 OBJECTIVES AND CONFORMANCE

OBIETTIVI E CONFORMITÀ

4.1 In each of the following clauses, the objectives and requirements of the clause are detailed.

In ognuno dei seguenti articoli sono dettagliati gli obiettivi e i requisiti del articolo.

4.2 To conform to this European Standard it shall be shown that each of the requirements have been satisfied to the software safety integrity level defined and therefore the clause objective has been met.

Per essere conformi alla presente Norma Europea deve essere dimostrato che è stato soddisfatto ogni requisito relativo al livello definito per l'integrità della sicurezza del software e che pertanto l'obiettivo del articolo è stato soddisfatto.

4.3 Where a requirement is qualified by the words "To the extent required by the software safety integrity level", this indicates that a range of techniques and measures can be used to satisfy that requirement.

Ove un requisito è qualificato dalle parole "Nella misura in cui è richiesta dal livello di integrità della sicurezza del software", questo indica che può essere usata una gamma di tecniche e provvedimenti per soddisfare tale requisito.

4.4 Where 4.3 applies the tables detailed in this European Standard should be used to assist in the selection of techniques and measures appropriate to the software safety integrity level.

Quando si applica l'art. 4.3 le tabelle dettagliate nella presente Norma Europea dovrebbero essere usate come ausilio nella scelta di tecniche e provvedimenti adatti al livello di integrità della sicurezza del software.

4.5 If a technique or measure is ranked as highly recommended (HR) in the tables then the rationale for not using that technique should be detailed and recorded either in the Software Quality Assurance Plan or in another document referenced by the Software Quality Assurance Plan. This is not necessary if an approved combination of techniques given in the corresponding table is used.

Se una tecnica o provvedimento è classificato come altamente raccomandato (HR) nelle tabelle allora il motivo per cui non è stata usata tale tecnica dovrebbe essere dettagliato e riportato nel Piano di Assicurazione della Qualità del Software o in altro documento al quale si fa riferimento nel Piano di Assicurazione della Qualità del Software. Questo non è necessario se è usata una combinazione approvata di tecniche definita nella tabella corrispondente.

4.6 If a technique or measure is proposed to be used that is not contained in the tables then its effectiveness and suitability in meeting the particular requirement and overall objective of the clause shall be justified and recorded in either the Software Quality Assurance Plan or in another document referenced by the Software Quality Assurance Plan.

Se viene proposto di usare una tecnica o provvedimento che non è contenuto nelle tabelle allora la sua efficacia e idoneità a soddisfare il particolare requisito e l'obiettivo generale del articolo deve essere giustificata e registrata nel Piano di Assicurazione della Qualità del Software o in altro documento richiamato dal Piano di Assicurazione della Qualità del Software.

4.7 Compliance with the requirements of a particular clause and their respective techniques and measures detailed in the tables shall be assessed by the inspection of documents required by this standard, other objective evidence, auditing and the witnessing of tests.

La conformità con i requisiti di una particolare articolo e con le corrispondenti tecniche e provvedimenti definite nelle tabelle devono essere valutate tramite l'ispezione dei documenti richiesti dalla presente norma, altre evidenze oggettive, eseguendo verifiche ispettive e assistendo alle prove.

4.8 This European Standard requires the use of a package of techniques and their correct application. These techniques are required from the tables and detailed in the bibliography.

La presente Norma Europea richiede l'uso di un insieme di tecniche e la loro corretta applicazione. Queste tecniche sono richieste dalle tabelle e dettagliate nella bibliografia.

5 SOFTWARE SAFETY INTEGRITY LEVELS

LIVELLI DI INTEGRITÀ DELLA SICUREZZA DEL SOFTWARE

5.1 Objective

To describe the assignment of software safety integrity levels to the software.

Obiettivi

Descrivere l'assegnazione dei livelli di integrità della sicurezza del software.

5.2 Requirements

5.2.1 There shall be produced, in accordance with EN 50126 and EN 50129:

- System Requirements Specification;
- System Safety Requirements Specification;
- System Architecture Description;
- System Safety Plan;

which include:

- safety functions;
- configuration or architecture of the system;
- hardware reliability requirements;
- safety integrity requirements.

The software safety integrity level shall be specified through following the general process for obtaining a safety integrity level identified in EN 50126.

5.2.2 The required software safety integrity level shall be decided on the basis of the level of risk associated with the use of the software in the system and the system safety integrity level.

5.2.3 Without further precautions, the software safety integrity level shall be, as a minimum, identical to the system safety integrity level. However, if mechanisms exist to prevent the failure of a software module from causing the system to go to an unsafe state, the software safety integrity level of the module may be reduced.

5.2.4 Risks which shall be taken into account are those associated with the following hazard consequences:

- loss of human life or lives;
- injuries to or illness of persons;
- environmental pollution; and
- loss of or damage to property.

5.2.5 Risk may be quantified but it is not possible to specify the software safety integrity in the same manner. Therefore for this European Standard the software safety integrity shall be specified as one of the following five levels:

Requisiti

Devono essere prodotti, in accordo con la EN 50126 e la EN 50129:

- Specifica dei Requisiti di Sistema;
- Specifica dei Requisiti di Sicurezza del Sistema;
- Descrizione dell'Architettura del Sistema;
- Piano di Sicurezza del Sistema;

che comprendono:

- funzioni di sicurezza;
- configurazione o architettura del sistema;
- requisiti di affidabilità dell'hardware;
- requisiti di integrità della sicurezza.

Il livello di integrità della sicurezza del software deve essere specificato seguendo completamente il processo generale identificato nella in EN 50126, per ottenere un livello di integrità della sicurezza.

Il livello di integrità della sicurezza del software richiesto deve essere deciso sulla base del livello di rischio associato all'uso del software nel sistema e del livello di integrità della sicurezza del sistema.

Senza ulteriori precauzioni, il livello di integrità della sicurezza del software deve essere, come minimo, identico al livello di integrità della sicurezza del sistema. Tuttavia, se esiste un meccanismo che per impedire il mancato funzionamento di un modulo software provochi il passaggio ad uno stato non sicuro del sistema, il livello di integrità della sicurezza del software del modulo può essere ridotto.

I rischi di cui si deve tenere conto sono quelli associati alle seguenti conseguenze pericolose:

- perdita della vita o di vite umane;
- ferite o malattie per le persone;
- inquinamento ambientale; e
- perdita o danneggiamento di beni.

Il rischio può essere quantificato, ma non è possibile specificare l'integrità della sicurezza del software allo stesso modo. Pertanto per la presente Norma Europea l'integrità della sicurezza del software deve essere specificata con uno dei seguenti 5 livelli:

Livello di integrità della sicurezza del software *Software safety integrity level*	Descrizione dell'integrità della sicurezza del software *Description of software safety integrity*
4	Molto Alta *Very High*
3	Alta *High*
2	Media *Medium*
1	Bassa *Low*
0	Non in sicurezza *Non safety-related*

5.2.6 The software safety integrity level shall be specified in the Software Requirements Specification (clause 8). If different software components have different software safety integrity levels, these shall be specified in the Software Architecture Specification (clause 9).

Il livello di integrità della sicurezza del software deve essere specificato nella Specifica dei Requisiti del Software (art. 8). Se diversi componenti del software hanno diversi livelli di integrità della sicurezza, questi devono essere specificati nella Specifica dell'Architettura del Software (art. 9).

6 PERSONNEL AND RESPONSIBILITIES

PERSONALE E RESPONSABILITÀ

6.1 Objective

Obiettivi

To ensure that all personnel who have responsibilities for the software are competent to discharge those responsibilities.

Assicurare che tutto il personale responsabile del software abbia la competenza per assumere queste responsabilità.

6.2 Requirements

Requisiti

6.2.1 As a minimum, the supplier and/or developer and the customer shall implement the relevant parts of EN ISO 9001, in accordance with the guidelines contained in EN ISO 9000-3.

Il fornitore e/o lo sviluppatore ed il cliente, devono come minimo implementare le parti della EN ISO 9001 pertinenti, in accordo con le linee guida contenute nella EN ISO 9000-3.

6.2.2 Except at software safety integrity level zero, the safety process shall be implemented under the control of an appropriate safety organisation which is compliant with the "Safety Organisation" sub-clause in the "Evidence of Safety Management" clause of EN 50129.

Salvo che per il livello zero di integrità della sicurezza del software, il processo di sicurezza deve essere implementato sotto il controllo di una idonea organizzazione per la sicurezza che sia conforme al paragrafo "Organizzazione per la Sicurezza" nel articolo "Evidenza della Gestione della Sicurezza" della EN 50129.

6.2.3 All personnel involved in all the phases of the Software Lifecycle, including management activities, shall have the appropriate training, experience and qualifications.

Tutto il personale coinvolto in tutte le fasi del Ciclo di Vita del Software, comprese le attività di gestione, deve avere l'adeguata istruzione, competenza e qualifica.

6.2.4 It is highly recommended that the training, experience and qualifications of all personnel involved in all the phases of the Software Lifecycle, including management activities, be justified with respect to the particular application, except at software safety integrity level zero.

È altamente raccomandato che l'addestramento, l'esperienza e le qualificazioni di tutto il personale coinvolto in tutte le fasi del Ciclo di Vita del Software, comprese le attività di gestione, siano motivate con riferimento alla specifica applicazione, fatta eccezione per il livello zero di integrità della sicurezza del software.

6.2.5 The justification contained in 6.2.4 shall be recorded in the Software Quality Assurance Plan, and shall include evidence of competency in the following areas, as appropriate:

i) engineering appropriate to the application area;
ii) software engineering;
iii) computer-systems engineering;
iv) safety engineering;
v) legal and regulatory framework.

La motivazione prevista in 6.2.4 deve essere riportata nel Piano di Assicurazione della Qualità del Software, e deve contenere, in modo adeguato, l'evidenza della competenza nelle seguenti aree:

i) ingegneria per il settore di applicazione;
ii) ingegneria del software;
iii) ingegneria dei sistemi computerizzati;
iv) ingegneria della sicurezza;
v) contesto legislativo e regolamentare.

6.2.6 An independent assessor for the software shall be appointed. See also 6.2.10 and 14.4.1.

Deve essere nominato un valutatore indipendente per il software. Vedere anche 6.2.10 e 14.4.1.

6.2.7 The assessor shall be given authority to perform the assessment of the software.

Al valutatore deve essere data l'autorità per eseguire la valutazione del software.

6.2.8

Throughout the Software Lifecycle, the parties involved shall be independent, to the extent required by the software safety integrity level, in accordance with Figure 5, which shall be interpreted as follows.

At all software safety integrity levels, the Assessor shall be approved by the Safety Authority and independent from the supplier except in the circumstances defined in 6.2.10.

The Designer/Implementer, Verifier and Validator can all belong to the same company but the following rules for minimum independence shall be complied with:

At software safety integrity level 0:

There are no constraints; the Designer/Implementer, Verifier and Validator can all be the same person.

At software safety integrity level 1 & 2:

The Verifier and Validator can be the same person but they shall not be the Designer/Implementer. However, the Designer/Implementer, Verifier and Validator can all report through the Project Manager.

At software safety integrity level 3 & 4 there are two permissible arrangements:

a) The Verifier and Validator can be the same person but they shall not also be the Designer/Implementer. In addition, the Verifier and Validator shall not all report through the Project Manager as the Designer/Implementer does and they shall have the authority to prevent the release of the product.

b) The Designer/Implementer, Verifier and Validator must all be separate persons. The Designer/Implementer and Verifier can report through the Project Manager, whereas the Validator shall not and the Validator shall have the authority to prevent the release of the product.

Per tutto il Ciclo di Vita del Software, le parti coinvolte devono essere indipendenti nella misura richiesta per il livello di integrità della sicurezza del software, in accordo con la Fig. 5, che deve essere interpretata come segue.

Per tutti i livelli di integrità della sicurezza del software, il Valutatore deve essere riconosciuto dalla Autorità per la Sicurezza ed essere indipendente dal fornitore salvo che nelle circostanze definite in 6.2.10.

Il Progettista/implementatore, il Verificatore e il Validatore possono appartenere tutti alla medesima società, ma per ottenere un minimo di indipendenza devono essere rispettate le seguenti regole:

Per il livello 0 di integrità della sicurezza del software:

Non vi sono restrizioni; il Progettista/Implementatore, il Verificatore e il Validatore possono essere tutti la medesima persona.

Per il livello 1 e 2 di integrità della sicurezza del software:

Il Verificatore e il Validatore possono essere la medesima persona, ma non devono essere il Progettista/Implementatore. Tuttavia, il Progettista/Implementatore, il Verificatore e il Validatore possono tutti relazionare attraverso il Project Manager.

Per il livello 3 e 4 di integrità della sicurezza del software vi sono due soluzioni ammesse:

a) Il Verificatore ed il Validatore possono essere la medesima persona, ma non devono anche essere il Progettista/Implementatore. Inoltre, il Verificatore e il Validatore non devono relazionare attraverso il Project Manager, come deve fare il Progettista/Implementatore, e devono avere l'autorità per impedire il rilascio del prodotto.

b) Il Progettista/Implementatore, il Verificatore e il Validatore devono essere persone distinte. Il Progettista/Implementatore e il Verificatore possono relazionare attraverso il Project Manager, mentre il Validatore non deve ed il Validatore deve avere l'autorità per impedire il rilascio del prodotto.

6.2.9

The parties responsible for the various clauses are as follows:

Le parti responsabili per le varie articoli sono le seguenti:

Specifica del Software (art. 8) *Software Requirements Specification (clause 8)*	Progettista *Designer*
Specifica di Prova dei Requisiti del Software (art. 8) *Software Requirements Test Specification (clause 8)*	Validatore *Validator*
Architettura del Software (art. 9) *Software Architecture (clause 9)*	Progettista *Designer*
Progetto e Sviluppo del Software (art. 10) *Software Design and Development (clause 10)*	Progettista *Designer*
Verifica e Prova del Software (art. 11) *Software Verification and Testing (clause 11)*	Verificatore *Verifier*
Integrazione Software/Hardware (art. 12) *Software/Hardware Integration (clause 12)*	Progettista *Designer*
Validazione del Software (art. 13) *Software Validation (clause 13)*	Validatore *Validator*
Valutazione del Software (art. 14) *Software Assessment (clause 14)*	Valutatore *Assessor*

6.2.10 At the discretion of the Safety Authority, the Assessor may be part of the supplier's organisation or of the customer's organisation but, in such cases, the Assessor shall

- be authorised by the Safety Authority;
- be totally independent from the project team;
- report directly to the Safety Authority.

A discrezione dell'Autorità per la Sicurezza, il Valutatore può far parte dell'organizzazione del fornitore, o dell'organizzazione del cliente, ma in tale caso, il Valutatore deve

- essere autorizzato dall'Autorità per la Sicurezza;
- essere totalmente indipendente dal gruppo di progetto;
- relazionare direttamente alla Autorità per la Sicurezza.

7 LIFECYCLE ISSUES AND DOCUMENTATION

RISULTATI E DOCUMENTAZIONE DEL CICLO DI VITA

7.1 Objectives

Obiettivi

7.1.1 To structure the development of the software into defined phases and activities.

Strutturare lo sviluppo del software in fasi ed attività definite.

7.1.2 To record all information pertinent to the software throughout the lifecycle of the software.

Registrare tutte le informazioni riguardanti il software nel corso del ciclo di vita del software.

7.2 Requirements

Requisiti

7.2.1 A lifecycle model for the development of software shall be selected. It shall be detailed in the Software Quality Assurance Plan in accordance with clause 15 of this European Standard. For example, two lifecycle models are shown in Figures 3 and 4.

Per lo sviluppo del software deve essere scelto un modello di ciclo di vita. Esso deve essere definito nel Piano di Assicurazione della Qualità del Software in accordo con l'art. 15 della presente Norma Europea. Due modelli di ciclo di vita sono mostrati come esempio, nelle Figure 3 e 4.

7.2.2 Quality Assurance procedures shall run in parallel with lifecycle activities and use the same terminology.

Le procedure di Assicurazione Qualità devono svolgersi in parallelo con le attività del ciclo di vita ed usare la medesima terminologia.

7.2.3 All activities to be performed during a phase shall be defined prior to the phase commencing. Each phase of the software lifecycle shall be divided into elementary tasks with a well defined input, output and activity for each of them.

Tutte le attività che devono essere eseguite durante una fase devono essere definite prima dell'inizio della fase. Ogni fase del ciclo di vita del software deve essere divisa in compiti elementari con ingressi, uscite ed attività ben definite per ognuna di esse.

7.2.4 The Software Quality Assurance Plan shall describe which verification steps and reports are required.

Il Piano di Assicurazione della Qualità del Software deve descrivere quali passi di verifica e quali rapporti sono richiesti.

7.2.5 All documents shall be structured to allow continued expansion in parallel with the design process.

Tutti i documenti devono essere strutturati per consentire uno sviluppo continuo in parallelo con il processo di progettazione.

7.2.6 Traceability of documents shall be provided for by each document having a unique reference number and a defined and documented relationship with other documents. Each term, acronym or abbreviation shall have the same meaning in every document. If, for historical reasons, this is not possible, the different meanings shall be listed and the references given.

In addition, each document except documents for COTS software (see 9.4.5) or previously de-

La tracciabilità dei documenti deve essere prevista in modo che ogni documento abbia un unico numero di riferimento ed una relazione definita e documentata con gli altri documenti. Ogni termine, acronimo o abbreviazione deve avere lo stesso significato in ogni documento. Se, per motivi storici, questo non è possibile, devono essere elencati i diversi significati e devono essere dati i riferimenti.

Inoltre, ogni documento, eccetto i documenti per il software COTS (vedere 9.4.5) o per il software

veloped software (see 9.4.6) shall be written according to the following rules:

a) it shall contain or implement all applicable conditions and requirements of the predecessor document with which it has a hierarchical relationship;
b) it shall not contradict the predecessor document;
c) each term, acronym or abbreviation shall have the same meaning in every document;
d) each item or concept shall be referred to by the same name or description in every document.

sviluppato in precedenza (vedere 9.4.6) deve essere scritto secondo le seguenti regole:

a) deve contenere o implementare tutte le condizioni ed i requisiti applicabili del documento predecessore con il quale ha una relazione gerarchica;
b) non deve contraddire il documento predecessore;
c) ogni termine, acronimo o abbreviazione deve avere lo stesso significato in ogni documento;
d) ogni elemento (item) o concetto deve essere referenziato con lo stesso nome o descrizione in ogni documento.

7.2.7 The contents of all documents shall be recorded in a form appropriate for manipulation, processing and storage.

I contenuti di tutti i documenti devono essere riportati in forma adatta per manipolazioni, elaborazioni e memorizzazzioni.

7.2.8 To the extent required by the software safety integrity level, the documents listed in the Documents Cross Reference Table (see below) shall be produced.

Nella misura in cui è richiesto dal livello di integrità della sicurezza del software, devono essere prodotti i documenti elencati nella Tabella dei Riferimenti Incrociati dei Documenti (vedere sotto).

7.2.9 Depending upon the size, complexity and lifecycle of the software being developed, the number of separate documents required to be produced will vary. Some documents may be combined (providing there is no loss of the required detail in the process). For large projects it may be necessary to sub-divide the documentation listed (in a hierarchical manner) into a number of more manageable child documents. Documents which have been produced by independent teams or entities shall not be combined into a single document.

In relazione alla dimensione, complessità e ciclo di vita del software che deve essere sviluppato, varierà il numero di documenti diversi che è richiesto siano prodotti. Alcuni documenti possono essere uniti (purché non ci sia perdita dei richiesti dettagli nel processo). Per grandi progetti può essere necessario suddividere la documentazione elencata (in modo gerarchico) in un certo numero di documenti discendenti più maneggevoli. Documenti che sono stati prodotti da entità o gruppi di lavoro indipendenti non devono essere combinati in un unico documento.

7.2.10 The relationship between the various documents identified in clause 7 can also be defined by using a DCRT (a Document Cross Reference Table). For each document listed in the "Documents" column, the phase and clause associated with its creation can be found by reading horizontally and vertically from the cell containing the symbol "■". The phases in which it is used can be found by reading vertically from the cells marked with the symbol "◆". The clause or other reference to the definition of the document can be found in the "Where defined" column. Where a clause is given, the following clauses should also be checked as they may contain further information. It should be noted that the reference for the Software Configuration Management Plan is shown in brackets because that clause simply references EN ISO 9001.

La relazione tra i vari documenti identificati nell'art. 7 può essere definita anche usando una DCRT (una Tabella dei Riferimenti Incrociati dei Documenti). Per ogni documento elencato nella colonna "Documenti", la fase e l'articolo associati con la sua creazione possono essere trovati leggendo orizzontalmente e verticalmente dalla cella contenente il simbolo "■". Le fasi nelle quali esso viene usato possono essere trovate leggendo verticalmente dalle celle segnate con il simbolo "◆". L'articolo o altro riferimento riguardante la definizione del documento possono essere trovati nella colonna "Dove definito". Dove viene indicato un articolo, devono essere verificati anche gli articoli che seguono poiché essi possono contenere ulteriori informazioni. Da notare che il riferimento per il Piano di Gestione della Configurazione del Software è riportato tra parentesi perché questo articolo cita solo la EN ISO 9001.

DOCUMENTS CROSS-REFERENCE TABLE

clause	8	9	10	11	12	13	14	15	16		
title	SRS	SA	SDD	SVer	S/H I	SVal	Ass	Q	Ma		
PHASES (*) = in parallel withother phases										**DOCUMENTS**	where defined
SYSTEM INPUTS	♦			♦	♦					System Requirements Specification	EN 50129 annex B.2.3
	♦	♦		♦	♦	♦	♦			System Safety Requirements Specification	EN 50129 annex B.2.4
	♦				♦					System Architecture Description	EN 50129 annex B.2.1
										System Safety Plan	EN 50129 EN 50126
SW PLANNING (*)	♦	♦	♦	♦		♦	♦	▮		Sw Quality Assurance Plan	15.4.3
						♦	♦	▮		Sw Configuration Management Plan	(15.4.2)
				▮		♦	♦			Sw Verification Plan	11.4.1
				▮		♦	♦			Sw Integration Test Plan	11.4.5
					▮	♦	♦			Sw/Hw Integration Test Plan	12.4.1
						▮	♦			Sw Validation Plan	13.4.3
							♦		▮	Sw Maintenance Plan	16.4.3
										Data Preparation Plan	17.4.2.1
										Data Test Plan	17.4.2.4
SW REQUIREMENTS	▮	♦	♦	♦	♦	♦	♦			Sw Requirements Specification	8.4.1
										Application Requirements Specification	17.4.1.1
	▮			♦	♦	♦	♦			Sw Requirements Test Specification	8.4.13
				▮						Sw Requirements Verification Report	11.4.11
SW DESIGN		▮	♦	♦	♦	♦	♦			Sw Architecture Specification	9.4.1
			▮	♦	♦	♦	♦			Sw Design Specification	10.4.3
				▮						Sw Arch. and Design Verification Report	11.4.12
SW MODULE DESIGN			▮	♦	♦	♦	♦			Sw Module Design Specification	10.4.3
			▮	♦	♦	♦	♦			Sw Module Test Specification	10.4.14
				▮						Sw Module Verification Report	11.4.13
CODE			▮	♦	♦	♦	♦			Sw Source Code	
				▮		♦	♦			Sw Source Code Verification Report	11.4.14
MODULE TESTING			▮	♦						Sw Module Test Report	10.4.14
SW INTEGRATION				▮						Sw Integration Test Report	11.4.15
										Data Test Report	17.4.2.4
SW/HW INTEGRATION					▮					Sw/Hw Integration Test Report	12.4.8
VALIDATION(*)						▮				Sw Validation Report	13.4.10
ASSESSMENT (*)							▮			Sw Assesment Report	14.4.9
MAINTENANCE									▮	Sw Change Records	16.4.9
									▮	Sw Maintenance Records	16.4.8

Articolo titolo	8	9	10	11	12	13	14	15	16	**DOCUMENTI**	Dove definiti
FASI *(*) = in parallelo con altre fasi*	**SRS**	**SA**	**SDD**	**SVer**	**S/HI**	**SVal**	**Ass**	**Q**	**Ma**		
INGRESSI DEL SISTEMA	♦			♦	♦					Specifica dei Requisiti di Sistema	EN 50129 Allegato B.2.3
	♦	♦		♦	♦	♦	♦			Specifica dei Requisiti di Sicurezza del Sistema	EN 50129 Allegato B.2.4
	♦				♦					Descrizione dell'Architettura del Sistema	EN 50129 Allegato B.2.1
										Piano di Sicurezza del Sistema	EN 50129 EN 50126
PIANIFICAZIONE SW(*)	♦	♦	♦	♦		♦	♦	▮		Piano di Assicurazione della Qualità del Sw	15.4.3
						♦	♦	▮		Piano di Gestione della Configurazione del Sw	(15.4.2)
				▮		♦	♦			Piano di Verifica del Sw	11.4.1
				▮		♦	♦			Piano di Prova di Integrazione del Sw	11.4.5
					▮	♦	♦			Piano di Prova di Integrazione Sw/Hw	12.4.1
						▮	♦			Piano di Validazione del Sw	13.4.3
							♦		▮	Piano di Manutenzione del Sw	16.4.3
										Piano di Preparazione dei Dati	17.4.2.1
										Piano di Prova dei Dati	17.4.2.4
REQUISITI SW	▮	♦	♦	♦	♦	♦	♦			Specifica dei Requisiti del Sw	8.4.1
										Specifica dei Requisiti dell'Applicazione	17.4.1.1
	▮			♦	♦	♦	♦			Specifica di Prova dei Requisiti del Sw	8.4.13
				▮						Rapporto di Verifica dei Requisiti Sw	11.4.11
PROGETTAZIONE SW		▮	♦	♦	♦	♦	♦			Specifica dell'Architettura del Sw	9.4.1
			▮	♦	♦	♦	♦			Specifica della Progettazione del Sw	10.4.3
				▮						Rapporto di Verifica dell'Architettura e della Progettazione del Sw	11.4.12
PROGETTAZIONE MODULO SW			▮	♦	♦	♦	♦			Specifica della Progettazine del Modulo	10.4.3
			▮	♦	♦	♦	♦			Specifica di Prova del Modulo Sw	10.4.14
				▮						Rapporto di Verifica del Modulo Sw	11.4.13
CODICE			▮	♦	♦	♦	♦			Codice Sorgente del Sw	
				▮		♦	♦			Rapporto di Verifica del Codice Sorgente del Sw	11.4.14
PROVA MODULO			▮	♦						Rapporto di Prova del Modulo Sw	10.4.14
INTEGRAZIONE SW				▮						Rapporto di Prova di Integrazione del Sw	11.4.15
										Rapporto di Prova dei Dati	17.4.2.4
INTEGRAZIONE SW/HW					▮					Rapporto di Prova di Integrazione Sw/Hw	12.4.8
VALUDAZIONE (*)						▮				Rapporto di Validazione del Sw	13.4.10
VALUTAZIONE (*)							▮			Rapporto di Valutazione del Sw	14.4.9
MANUTENZIONE									▮	Registrazioni delle Variazioni del Sw	16.4.9
									▮	Registrazioni della Manutenzione del Sw	16.4.8

8 SOFTWARE REQUIREMENTS SPECIFICATION

8 SPECIFICA DEI REQUISITI DEL SOFTWARE

8.1 Objectives

8.1.1 To describe a document which defines a complete set of requirements for the software meeting all System Requirements to the extent required by the Software safety Integrity level. It serves the purpose of a comprehensive document for each software engineer and makes it unnecessary for him to screen for requirements in any other document.

8.1.2 To describe the Software Requirements Test Specification.

8.1 Obiettivi

Elaborare un documento che definisca un insieme completo di requisiti per il software soddisfacendo tutti i Requisiti del Sistema nella misura in cui è richiesto dal livello di integrità della sicurezza del Software. Ciò ha lo scopo di disporre di un documento esauriente per ogni ingegnere del software e non rendere necessario che egli selezioni i requisiti in qualche altro documento.

Descrivere la Specifica di Prova dei Requisiti del Software.

8.2 Input documents

1) System Requirements Specification
2) System Safety Requirements Specification
3) System Architecture Description
4) Software Quality Assurance Plan

8.2 Documenti d'ingresso

1) Specifica dei Requisiti del Sistema
2) Specifica dei Requisiti di Sicurezza del Sistema
3) Descrizione dell'Architettura del Sistema
4) Piano di Assicurazione della Qualità del Software

8.3 Output documents

1) Software Requirements Specification
2) Software Requirements Test Specification

8.3 Documenti di uscita

1) Specifica dei Requisiti del Software
2) Specifica di Prova dei Requisiti del Software

8.4 Requirements

8.4.1 The Software Requirements Specification shall express the required properties of the software being developed, not the procedures to develop them. These properties, which are all (except safety) defined in ISO/IEC 9126, shall include:

- functionality (including capacity and response time performance);
- reliability and maintainability;
- safety (including safety functions and their associated software safety integrity levels);
- efficiency;
- usability;
- portability.

The software safety integrity level shall be derived as defined in clause 5 and recorded in the Software Requirements Specification.

8.4.2 To the extent required by the software safety integrity level the Software Requirements Specification shall be expressed and structured in such a way that it is

i) complete, clear precise, unequivocal, verifiable, testable, maintainable and feasible,

ii) traceable back to all documents mentioned under 8.2.

8.4 Requisiti

La Specifica dei Requisiti del Software deve esprimere le proprietà richieste al software che deve essere sviluppato, non le procedure per svilupparlo. Queste proprietà che sono tutte definite (ad eccezione della sicurezza) nella ISO/IEC 9126, devono comprendere:

- funzionalità (compresa la capacità e la prestazione di risposta al tempo);
- affidabilità e manutenibilità;
- sicurezza (comprese le funzioni di sicurezza e i loro associati livelli di integrità della sicurezza del software);
- efficienza;
- usabilità;
- portabilità.

Il livello di integrità della sicurezza del software deve essere ricavato come definito nell'art. 5 e registrato nella Specifica dei Requisiti del Software.

Nella misura in cui è richiesto dal livello di integrità della sicurezza del software la Specifica dei Requisiti del Software deve essere espressa e strutturata in modo tale che sia

i) completa, chiara precisa, inequivocabile, verificabile, che si possa provare, manutenibile e fattibile,

ii) tracciabile a ritroso verso tutti i documenti menzionati in 8.2.

8.4.3 The Software Requirements Specification shall include modes of expression and descriptions which are understandable to the responsible personnel involved in the whole life cycle of the system.

La Specifica dei Requisiti del Software deve comprendere modi di esprimersi e descrizioni che siano comprensibili al personale responsabile coinvolto nell'intero ciclo di vita del sistema.

8.4.4 The Software Requirements Specification shall identify and document all interfaces with any other systems, either within or outside the equipment under control, including operators, wherever a direct connection exists or is planned.

La Specifica dei Requisiti del Software deve identificare e documentare tutte le interfacce con ogni altro sistema, sia all'interno che all'esterno dell'apparecchiatura sotto controllo, compresi gli operatori, ovunque esista o sia pianificato un collegamento diretto.

8.4.5 All relevant modes of operation shall be detailed in the Software Requirements Specification.

Tutti i modi associati al funzionamento devono essere dettagliati nella Specifica dei Requisiti del Software.

8.4.6 All relevant modes of behaviour of the programmable electronics, in particular failure behaviour, shall be detailed in the Software Requirements Specification.

Tutti i modi associati al comportamento dell'elettronica programmabile, in particolare comportamenti di mancato funzionamento, devono essere dettagliati nella Specifica dei Requisiti del Software.

8.4.7 Any constraints between the hardware and the software shall be identified and documented in the Software Requirements Specification.

Ogni vincolo tra hardware e software deve essere identificato e documentato nella Specifica dei Requisiti del Software.

8.4.8 The Software Requirements Specification shall indicate the degree of software self-checking and the specified degree of hardware checking by the software. Software self-checking consists of the detection and reporting by the software of its own failures and errors.

La Specifica dei Requisiti del Software deve indicare il grado di autodiagnosi del software e il grado stabilito di controllo dell'hardware da parte del software. L'autodiagnosi del Software consiste nella rilevazione e nella presentazione da parte del software dei propri mancati funzionamenti ed errori.

8.4.9 The Software Requirements Specification shall include requirements for the periodic testing of functions to the extent required by the System Safety Requirements Specification.

La Specifica dei Requisiti del Software deve comprendere i requisiti delle prove periodiche delle funzioni nella misura in cui è richiesto dalla Specifica dei Requisiti di Sicurezza del Sistema.

8.4.10 The Software Requirements Specification shall include requirements to enable all safety functions to be testable during overall system operation to the extent required by the System Safety Requirements Specification.

La Specifica dei Requisiti del Software deve comprendere i requisiti per consentire che tutte le funzioni di sicurezza possano essere provate durante il funzionamento generale del sistema nella misura in cui è richiesto dalla Specifica dei Requisiti di Sicurezza del Sistema.

8.4.11 When the software is required to perform functions especially those related to achieving the required system safety integrity level, then these shall be clearly identified in the Software Requirements Specification.

Quando è richiesto che il software esegua funzioni, in special modo quelle connesse con il raggiungimento del livello richiesto di integrità della sicurezza del sistema, queste devono essere chiaramente identificate nella Specifica dei Requisiti del Software.

8.4.12 When the software is required to perform non-safety functions then these shall be clearly identified in the Software Requirements Specification.

Quando è richiesto che il software esegua funzioni non in sicurezza queste devono essere chiaramente identificate nella Specifica dei Requisiti del Software.

8.4.13 A Software Requirements Test Specification shall be developed from the Software Requirements Specification. This test specification shall be used for verification of all the requirements as described in the Software Requirements Specifica-

Una Specifica di Prova dei Requisiti del Software deve essere sviluppata dalla Specifica dei Requisiti del Software. Questa specifica di prova deve essere usata per la verifica di tutti i requisiti descritti nella Specifica dei Requisiti del Software ed anche

tion and also as a description of the tests to be performed on the completed software.

come una descrizione delle prove che devono essere eseguite sul software ultimato.

8.4.14 The Software Requirements Test Specification shall identify for each required function the test cases including:

i) the required input signals with their sequences and their values;
ii) the anticipated output signals with their sequences and their values;
iii) the acceptance criteria, including performance and quality aspects.

La Specifica di prova dei Requisiti del Software deve identificare, per ogni funzione richiesta, i casi di prova, comprendendo:

i) i segnali d'ingresso richiesti con le loro sequenze ed i loro valori;
ii) i segnali di uscita previsti con le loro sequenze e i loro valori;
iii) i criteri di accettazione, compresi gli aspetti di prestazione e qualità.

8.4.15 Traceability to requirements shall be an important consideration in the validation of a safety-related system and means shall be provided to allow this to be demonstrated throughout all phases of the lifecycle.

La tracciabilità dei requisiti deve essere una importante considerazione nella validazione di un sistema in sicurezza e devono essere previsti mezzi per permettere che essa sia dimostrata attraverso tutte le fasi del ciclo di vita.

8.4.16 Any untraceable material shall be shown to have no bearing upon the safety or integrity of the system.

Si deve dimostrare che qualsiasi documentazione non tracciabile non deve avere relazione con la sicurezza o con l'integrità del sistema.

9 SOFTWARE ARCHITECTURE

ARCHITETTURA DEL SOFTWARE

9.1 Objectives

Obiettivi

9.1.1 To develop a software architecture that achieves the requirements of the Software Requirements Specification to the extent required by the software safety integrity level.

Sviluppare un'architettura del software che realizzi i requisiti della Specifica dei Requisiti del Software nella misura in cui è richiesto dal livello di integrità della sicurezza del software.

9.1.2 To review the requirements placed on the software by the system architecture.

Prendere in esame i requisiti assegnati al software dall'architettura di sistema.

9.1.3 To identify and evaluate the significance of Hardware/Software interactions for safety.

Identificare e valutare l'importanza dell'interazione Hardware/Software per la sicurezza.

9.1.4 To choose a design method if one has not been previously defined.

Scegliere un metodo progettuale se non ne è stato definito uno in precedenza.

9.2 Input documents

Documenti d'ingresso

1) Software Requirements Specification
2) System Safety Requirements Specification
3) System Architecture Description
4) Software Quality Assurance Plan

1) Specifica dei Requisiti del Software
2) Specifica dei Requisiti di Sicurezza del Sistema
3) Descrizione dell'Architettura del Sistema
4) Piano di Assicurazione della Qualità del Software

9.3 Output documents

Documenti di uscita

Software Architecture Specification

Specifica dell'Architettura del Software

9.4 Requirements

Requisiti

9.4.1 The proposed software architecture shall be established by the software supplier and/or developer and detailed in the Software Architecture Specification.

L'architettura software proposta deve essere stabilita dal fornitore del software e/o da chi lo sviluppa e dettagliata nella Specifica dell'Architettura del Software.

9.4.2 The Software Architecture Specification shall consider the feasibility of achieving the Software Requirements Specification at the required software safety integrity level.

La Specifica dell'Architettura del Software deve considerare la fattibilità nel realizzare la Specifica dei Requisiti del Software per il livello di integrità della sicurezza del software richiesto.

9.4.3 The Software Architecture Specification shall identify, evaluate and detail the significance of all hardware/software interactions. As required by EN 50126 and EN 50129, the preliminary studies concerning the interactions between hardware and software shall have been recorded in the System Safety Requirements Specification.

La Specifica dell'Architettura del Software deve identificare, valutare e dettagliare l'importanza di tutte le interazioni hardware/software. Come richiesto dalla EN 50126 e dalla EN 50129, gli studi preliminari concernenti l'interazione tra hardware e software devono essere registrati nella Specifica dei Requisiti di Sicurezza del Sistema.

9.4.4 The Software Architecture Specification shall identify all software components and for these components identify:

i) whether these components are new, existing or proprietary;
ii) whether these components have been previously validated and if so their validation conditions;
iii) the software safety integrity level of the component.

La Specifica dell'Architettura del Software deve identificare tutti i componenti software e per questi componenti identificare:

i) se questi componenti sono nuovi, esistenti o proprietari;
ii) se questi componenti sono stati precedentemente validati ed in caso affermativo le loro condizioni di validazione;
iii) il livello di integrità della sicurezza del software del componente.

9.4.5 The use of COTS software shall be subject to the following restrictions:

i) for software safety integrity level 0, the use of the COTS software shall be accepted with no further precautions;
ii) if COTS software is to be used at software safety integrity levels 1 or 2, it shall be included in the software validation process;
iii) if COTS software is to be used at software safety integrity levels 3 or 4, the following precautions shall also be taken:
   a) the COTS software shall be included in the validation testing;
   b) an analysis of possible failures shall be carried out;
   c) a strategy shall be defined to detect failures of the COTS software and to protect the system from these failures;
   d) the protection strategy shall be the subject of validation testing;
   e) error logs shall exist and shall be evaluated;
   f) as far as practicable, only the simplest functions of the COTS software shall be used.

L'utilizzo di software COTS deve essere soggetto alle seguenti restrizioni:

i) Per livello di integrità della sicurezza del software 0, l'utilizzo del software COTS deve essere accettato senza ulteriori precauzioni;
ii) Se il software COTS viene usato per i livelli di integrità della sicurezza del software 1 o 2, esso deve essere inserito nel processo di validazione del software;
iii) Se il software COTS viene usato per i livelli di integrità della sicurezza del software 3 o 4, devono essere prese anche le seguenti precauzioni:
   a) il software COTS deve essere incluso nelle prove di validazione;
   b) deve essere eseguita un'analisi dei possibili mancati funzionamenti;
   c) deve essere definita una strategia per rilevare i mancati funzionamenti del software COTS e per proteggere il sistema da questi mancati funzionamenti;
   d) la strategia di protezione deve essere soggetta a prove di validazione;
   e) devono esistere registrazioni degli errori (error logs) ed esse devono essere valutate;
   f) per quanto praticabile, devono essere usate solo le funzioni più semplici del software COTS.

9.4.6 If previously developed software is to be used as part of the design then it shall be clearly identified and documented. The Software Architecture Specification shall justify the software's suitability in satisfying the Software Requirements Specification and the software safety integrity level. The effects of any changes to the software on the rest of the system must be carefully considered in order to decide whether

Se il software sviluppato in precedenza viene utilizzato come parte del progetto allora esso deve essere chiaramente identificato e documentato. La Specifica dell'Architettura del Software deve giustificare l'idoneità del software a soddisfare la Specifica dei Requisiti del Software ed il livello di integrità della sicurezza del software. Gli effetti di ogni variazione del software sul resto del sistema devono essere attentamente considerati al fine di

they require a re-inspection and re-assessment. There shall be evidence that interface specifications to other modules which are not being re-verified, re-validated and re-assessed are being followed.

decidere se essi richiedono una nuova ispezione ed una nuova valutazione. Ci deve essere evidenza che le specifiche di interfaccia con altri moduli che non sono stati nuovamente verificati, validati e valutati siano state seguite.

9.4.7 Whenever possible existing verified software modules developed according to this standard shall be used in the design.

Ogni volta che è possibile devono essere utilizzati nel progetto i moduli software esistenti, verificati, sviluppati secondo la presente norma.

9.4.8 The Software Architecture shall minimise the safety part of the application.

L'Architettura Software deve minimizzare la parte in sicurezza dell'applicazione.

9.4.9 Where the software consists of components of different software safety integrity levels then all of the software components shall be treated as belonging to the highest software safety integrity level unless there is evidence of independence between the higher software safety integrity level components and the lower software safety integrity level components. This evidence shall be recorded in the Software Architecture Specification.

Ove il software sia composto da componenti aventi diversi livelli di integrità della sicurezza del software, tutti i componenti software devono essere trattati come se appartenessero al più alto livello di integrità della sicurezza del software, salvo che vi sia evidenza dell'indipendenza tra i componenti di più alto livello di integrità della sicurezza del software e i componenti di livello più basso. Questa evidenza deve essere registrata nella Specifica dell'Architettura del Software.

9.4.10 The Software Architecture Specification shall identify the strategy for the software development to the extent required by the software safety integrity level. The Software Architecture Specification shall be expressed and structured in such a way that it is:

i) complete, clear, precise, unequivocal, verifiable, testable, maintainable and feasible,

ii) traceable back to the Software Requirements Specification.

La Specifica dell'Architettura del Software deve identificare la strategia di sviluppo del software nella misura in cui è richiesto dal livello di integrità della sicurezza del software. La Specifica dell'Architettura del Software deve essere scritta e strutturata in modo tale che sia:

i) completa, chiara, precisa, inequivocabile, verificabile, che si possa provare, manutenibile e realizzabile.

ii) tracciabilerispetto allaSpecifica dei Requisiti del Software.

9.4.11 The Software Architecture Specification shall justify the balance taken between the strategies of avoiding faults and handling faults.

La Specifica dell'Architettura del Software deve giustificare il bilanciamento assunto tra le strategie per evitare i guasti e per gestire i guasti.

9.4.12 The Software Architecture Specification shall justify that the techniques and measures chosen form a set which satisfies the Software Requirements Specification at the required software safety integrity level.

La Specifica dell'Architettura del Software deve giustificare che le tecniche e i provvedimenti scelti costituiscano un'insieme che soddisfi la Specifica dei Requisiti del Software per il livello di integrità della sicurezza del software richiesto.

10 SOFTWARE DESIGN AND IMPLEMENTATION

PROGETTAZIONE ED IMPLEMENTAZIONE DEL SOFTWARE

10.1 Objectives

Obiettivi

10.1.1 To design and implement software of a defined software safety integrity level from the Software Requirements Specification and the Software Architecture Specification.

Progettare ed implementare il software di livello di integrità della sicurezza definito dalla Specifica dei Requisiti del Software e dalla Specifica dell'Architettura del Software.

10.1.2 To achieve software which is analysable, testable, verifiable and maintainable. Module testing is also included in this phase. As verification

Realizzare un software che sia analizzabile, sottoponobile a prova, verificabile e manutenibile. Anche le prove del modulo sono comprese in questa

and test will be a critical element in the validation, particular consideration shall be given to verification and test needs throughout the design and development, in order to ensure the resultant system and its software will be readily testable from the outset.

fase. Poiché la verifica e le prove saranno elementi critici per la validazione, si deve prestare particolare attenzione alle verifiche ed alle prove necessarie in ogni momento del progetto e dello sviluppo, al fine di assicurare che il sistema risultante ed il suo software sia prontamente sottoponibile a prova fin dall'inizio.

10.1.3 To select a suitable set of tools, including languages and compilers, for the required software safety integrity level, over the whole lifecycle of the software which assists verification, validation, assessment and maintenance.

Selezionare un adeguato insieme di strumenti (tools), compresi linguaggi e compilatori, per il livello di integrità della sicurezza del software richiesto, per l'intero ciclo di vita del software che aiuta la verifica, validazione, valutazione e manutenzione.

10.1.4 To carry out software integration.

Attuare l'integrazione del software.

10.2 Input documents

Documenti d'ingresso

1) Software Requirements Specification
2) Software Architecture Specification
3) Software Quality Assurance Plan

1) Specifica dei Requisiti del Software
2) Specifica dell'Architettura del Software
3) Piano di Assicurazione della Qualità del Software

10.3 Output documents

Documenti di uscita

1) Software Design Specification
2) Software Module Design Specification
3) Software Module Test Specification
4) Software Source code and supporting documentation
5) Software Module Test Report

1) Specifica della Progettazione del Software
2) Specifica della Progettazione del Modulo Software
3) Specifica di Prova del Modulo Software
4) Codice sorgente del Software e documentazione di supporto
5) Rapporto di Prova del Modulo Software

10.4 Requirements

Requisiti

10.4.1 The Software Requirements Specification and the Software Architecture Specification shall be available, although not necessarily finalised, prior to the start of the design process.

La Specifica dei Requisiti del Software e la Specifica dell'Architettura del Software devono essere disponibili, anche se non necessariamente completamente definiti, prima dell'inizio del processo di progettazione.

10.4.2 The size and complexity of the software developed shall be kept to a minimum.

La dimensione e la complessità del software sviluppato devono essere mantenute al minimo.

10.4.3 The Software Design Specification shall describe the software design based on a decomposition into modules with each module having a Software Module Design Specification and a Software Module Test Specification.

La Specifica di progettazione del Software deve descrivere il progetto del software sulla base di una sua decomposizione in moduli, con ogni modulo dotato di Specifica della progettazione del Modulo Software e Specifica di Prova del Modulo Software.

10.4.4 The Software Design Specification shall address:

i) software components traced back to software architecture and their safety integrity level;
ii) interfaces of software components with the environment;
iii) interfaces between the software components;
iv) data structure;
v) partitioning of requirements on components;
vi) main algorithms and sequencing;
vii) diagrams.

La Specifica della progettazione del Software deve trattare:

i) componenti software tracciati a ritroso verso l'architettura software e loro livello di integrità della sicurezza;
ii) interfacce dei componenti software con l'ambiente;
iii) interfacce tra i componenti software;
iv) struttura dei dati;
v) suddivisione dei requisiti nei componenti;
vi) algoritmi principali e sequenze;
vii) diagrammi.

10.4.5 The Software Module Design Specifications shall address (one Software module Design Specification per module):

i) identification of all lowest software components (called modules in the present standard) traced back to the upper level;
ii) their detailed interfaces with environment and other modules with detailed inputs and outputs;
iii) their safety integrity level;
iv) detailed algorithms and data structures.

Each Software Module design Specification should be self consistent and allow coding of the corresponding module.

Le Specifiche di Progetto del Modulo Software devono trattare (una Specifica della progettazione del modulo Software per ogni modulo):

i) identificazione di tutti i componenti software di livello più basso (chiamati moduli nella presente norma) tracciati verso il livello superiore;
ii) le loro interfacce dettagliate con l'ambiente e gli altri moduli con ingressi e uscite dettagliati;
iii) il loro livello di integrità della sicurezza;
iv) gli algoritmi e le strutture dei dati dettagliati.

Ogni Specifica di progettazione del Modulo Software dovrebbe essere autoconsistente e consentire la codifica del modulo corrispondente.

10.4.6 Each software module shall be readable, understandable and testable.

Ogni modulo software deve essere leggibile, comprensibile e sottoponibile a prova.

10.4.7 A suitable set of tools, including design methods, languages and compilers shall be selected for the required software safety integrity level over the whole lifecycle of the software.

Devono essere scelti, per il livello di integrità della sicurezza del software richiesto, per l'intero ciclo di vita del software, un adeguato insieme di strumenti, compresi metodi di progettazione, linguaggi e compilatori.

10.4.8 When applicable, automatic testing tools and integrated development tools shall be used. This shall take account of the needs of the Verifier and Validator.

Quando applicabili, devono essere usati strumenti (tools) di prova automatici e strumenti (tools) di sviluppo integrato. Questi devono tenere in considerazione le necessità del Verificatore e del Validatore.

10.4.9 To the extent required by the software safety integrity level, the programming language selected shall have a translator/compiler which has one of the following:

i) a "Certificate of Validation" to a recognised National/International standard;
ii) an assessment report which details its fitness for purpose;
iii) a redundant signature control based process that provides detection of the translation errors.

Nella misura in cui è richiesto dal livello di integrità della sicurezza del software, il linguaggio di programmazione scelto deve avere un traduttore/compilatore che possiede una tra le cose seguenti:

i) un "Certificato di Validazione" basato su una norma Nazionale/Internazionale riconosciuta;
ii) un rapporto di valutazione che specifichi che è adatto allo scopo;
iii) un processo basato sul controllo di firma ridondante che fornisce l'individuazione degli errori di traduzione.

10.4.10 The language chosen shall meet the following requirements:

i) the language chosen shall contain features that facilitate the identification of programming errors;
ii) the language chosen shall support features that match the design method.

Il linguaggio scelto deve soddisfare i seguenti requisiti:

i) il linguaggio scelto deve contenere caratteristiche che facilitino l'identificazione degli errori di programmazione;
ii) il linguaggio scelto deve supportare caratteristiche in armonia con il metodo di progettazione.

10.4.11 When 10.4.10 cannot be satisfied then a justification for any alternative language detailing its fitness for purpose shall be recorded in the Software Architecture Specification or Software Quality Assurance Plan.

Quando la 10.4.10 non può essere soddisfatta, deve essere registrata nella Specifica dell'Architettura del Software o nel Piano di Assicurazione della Qualità del Software una giustificazione per ogni linguaggio alternativo dettagliando la sua idoneità all'uso.

10.4.12 Coding standards shall be developed and used for the development of all software. These shall be referenced in the Software Quality Assurance Plan (see 15.4.5).

Per lo sviluppo di tutto il software, devono essere sviluppati ed usati standard di codifica. Questi devono trovare riferimento nel Piano di Assicurazione della Qualità del Software (vedere 15.4.5).

10.4.13 The coding standards shall specify good programming practice, proscribe unsafe language features and describe procedures for source code documentation. As a minimum, each software module shall contain in the source code the information defined in the following (non-exhaustive) list:

- author;
- configuration history;
- short description.

The use of a standard form for this information is recommended. It should be the same for all the modules.

Gli standard di codifica (programmazione) devono specificare una buona pratica di programmazione, proibire caratteristiche del linguaggio non sicure e descrivere procedure per la documentazione del codice sorgente. Come minimo, ogni modulo software deve contenere nel codice sorgente le informazioni definite nel seguente elenco (non esaustivo):

- autore;
- storia della configurazione;
- breve descrizione.

Si raccomanda l'uso di un formato standard per queste informazioni. Dovrebbe essere lo stesso per tutti i moduli.

10.4.14 Software Module Testing: Each module shall have a Software Module Test Specification which the module shall be tested against. These tests shall show that each module performs its intended function. The Software Module Test Specification shall define the required degree of test coverage.

A Software Module Test Report shall be produced and shall include the following features:

i) a statement of the test results and whether each module has met the requirements of its Software Module Design Specification;
ii) a statement of test coverage shall be provided for each module, showing that all source code instructions have been executed at least once;
iii) it shall be in a form that is auditable;
iv) test cases and their results shall be recorded in a machine readable form for subsequent analysis; Tests should be repeatable and be performed by automatic means, if practicable.

Checking that the module has correctly satisfied its test specification is a verification activity, see clause 11.

Prove del Modulo Software: Ogni modulo deve avere una Specifica di Prova del Modulo Software con la quale deve essere provato il modulo. Queste prove devono dimostrare che ogni modulo esegue le funzioni per esso volute. La Specifica di Prova del Modulo Software deve definire il grado richiesto di copertura della prova.

Deve essere prodotto un Rapporto di Prova del Modulo Software e deve comprendere i seguenti aspetti:

i) una dichiarazione dei risultati della prova e se ogni modulo ha soddisfatto i requisiti della sua Specifica di Progettazione del Modulo Software;
ii) per ogni modulo deve essere fornita una dichiarazione di copertura della prova, dimostrando che tutte le istruzioni di codice sorgente sono state eseguite almeno una volta;
iii) deve essere in un formato tale da consentire la verifica ispettiva;
iv) i casi di prova e i loro risultati devono essere registrati in una forma leggibile dalla macchina per successive analisi; Le prove dovrebbero essere ripetibili e essere eseguite con mezzi automatici, se fattibile.

Il controllo che il modulo ha correttamente soddisfatto la sua specifica di prova è un'attività di verifica, vedere l'art. 11.

10.4.15 In accordance with the required software safety integrity level the design method chosen shall possess features that facilitate:

i) abstraction, modularity and other features which control complexity;
ii) the clear and precise expression of
   - functionality;
   - information flow between components,
   - sequencing and time related information;
   - concurrency;
   - data structure and properties;
iii) human comprehension;
iv) verification and validation.

In accordo con il livello di integrità della sicurezza del software richiesto il metodo di progettazione scelto deve possedere caratteristiche che facilitano:

i) astrazione, modularità e altre caratteristiche che controllano la complessità;
ii) la chiara e precisa espressione di
   - funzionalità;
   - flusso di informazioni tra componenti;
   - sequenza ed informazioni relazionate con il tempo;
   - concorrenzialità;
   - struttura dei dati e proprietà;
iii) comprensione per l'uomo;
iv) verifica e validazione.

10.4.16 The design method chosen shall possess features that facilitate software maintenance. Such features shall include modularity, information hiding and encapsulation.

Il metodo di progettazione scelto deve possedere caratteristiche che facilitino la manutenzione del software. Tali caratteristiche devono comprendere modularità, information hiding ed encapsulation.

10.4.17 The integration of software modules shall be the process of progressively combining individual and previously tested modules of software into a composite whole (or into a number of composite sub-systems) in order that the module interfaces and the assembled software may be adequately proven prior to system integration and test.

L'integrazione dei moduli software deve essere il processo di combinare progressivamente moduli software singoli e provati precedentemente in un unico insieme (o in una serie di sottosistemi) in modo che le interfacce del modulo ed il software assemblato possano essere adeguatamente provati prima della integrazione di sistema e della relativa prova.

10.4.18 Within the context of this standard, and to a degree appropriate to the specified software safety integrity level, traceability shall particularly address:

i) traceability of requirements to the design or other objects which fulfil them;
ii) traceability of design objects to the implementation objects which instantiate them.

Nel contesto della presente norma, e per un grado adeguato al livello di integrità della sicurezza del software specificato, la tracciabilità deve riguardare particolarmente:

i) tracciabilità dei requisiti verso la progettazione o altri oggetti che li soddisfano;
ii) tracciabilità degli oggetti della progettazione verso gli oggetti dell'implementazione che li realizzano.

The output of the traceability process shall be the subject of formal configuration management.

Il risultato del processo di tracciabilità deve essere oggetto di gestione della configurazione formale.

11 SOFTWARE VERIFICATION AND TESTING — VERIFICA E PROVA DEL SOFTWARE

11.1 Objective — Obiettivi

To the extent required by the software safety integrity level, to test and evaluate the products of a given phase to ensure correctness and consistency with respect to the products and standards provided as input to that phase.

Nella misura in cui è richiesta dal livello di integrità della sicurezza del software, provare e valutare i prodotti di una data fase per assicurare correttezza e consistenza nei confronti dei prodotti e delle norme stabilite come dati di ingresso per quella fase.

11.2 Input documents — Documenti d'ingresso

1) System Requirements Specification
2) System Safety Requirements Specification
3) Software Requirements Specification
4) Software Requirements Test Specification
5) Software Architecture Specification
6) Software Design Specification
7) Software Module Design Specification
8) Software Module Test Specification
9) Software Source code and supporting documentation
10) Software Quality Assurance Plan
11) Software Module Test Report

1) Specifica dei Requisiti del Sistema
2) Specifica dei Requisiti di Sicurezza del Sistema
3) Specifica dei Requisiti del Software
4) Specifica di Prova dei Requisiti del Software
5) Specifica dell'Architettura del Software
6) Specifica della progettazione del Software
7) Specifica della progettazione del Modulo Software
8) Specifica di Prova del Modulo Software
9) Codice Sorgente del Software e documentazione di supporto
10) Piano di Assicurazione della Qualità del Software
11) Rapporto di Prova del Modulo Software

11.3 Output documents — Documenti di uscita

1) Software Verification Plan
2) Software Requirements Verification Report
3) Software Architecture and Design Verification Report
4) Software Module Verification Report

1) Piano di Verifica del Software
2) Rapporto di Verifica dei Requisiti del Software
3) Rapporto di Verifica dell'Architettura e della progettazione del Software
4) Rapporto di Verifica del Modulo Software

5) Software Source Code Verification Report

6) Software Integration Test Plan
7) Software Integration Test Report

5) Rapporto di Verifica del Codice Sorgente del Software
6) Piano di Prova di Integrazione del Software
7) Rapporto di Prova di Integrazione del Software

11.4 Requirements

Requisiti

11.4.1 A Software Verification Plan shall be created in order that verification activities may be properly directed and that particular design or other verification needs may be suitably provided for. During development (and depending upon the size of the system) the plan may be sub-divided into a number of child documents and be naturally added to, as the detailed needs of verification become clearer.

Deve essere creato un Piano di Verifica del Software affinché le attività di verifica possano essere opportunamente orientate e affinché possa essere adeguatamente provveduto a progetti particolari o altre necessità di verifica. Durante lo sviluppo (e in relazione alla dimensione del sistema) il piano può essere suddiviso in una serie di documenti figlio che naturalmente possono essere aggiunti, mentre divengono più chiare le dettagliate necessità di verifica.

11.4.2 The Software Verification Plan shall document all the criteria, techniques and tools to be utilised in the verification process for that phase.

Il Piano di Verifica del Software deve documentare tutti i criteri, le tecniche e gli strumenti (tools) da utilizzare nel processo di verifica per quella fase.

11.4.3 The Software Verification Plan shall describe the activities to be performed to ensure correctness and consistency with respect to the products and standards provided as input to that phase.

Il Piano di Verifica del Software deve descrivere le attività da eseguire per assicurare correttezza e consistenza nei confronti dei prodotti e delle norme fornite come dati d'ingresso per quella fase.

11.4.4 The Software Verification Plan shall address the following:

i) the selection of verification strategies and techniques. To avoid undue complexity in the assessment of the verification and test activity, preference should be given to the selection of test cases and methods etc., which are in themselves readily analysable;
ii) the selection and utilisation of the software test equipment;
iii) the selection and documentation of verification activities;
iv) the evaluation of verification results gained;
v) the evaluation of the reliability requirements;
vi) the roles and responsibilities of those involved in the test process;
vii) the degree of test coverage required to be achieved.

Il Piano di Verifica del Software deve trattare quanto segue:

i) la scelta delle strategie e delle tecniche di verifica. Per evitare eccessive complessità nella valutazione della verifica e dell'attività di prova, dovrebbe essere data preferenza alla scelta di casi di prova e metodi, ecc., che siano di per se prontamente analizzabili;
ii) la scelta e l'utilizzo di dispositivi di prova software;
iii) la scelta e la documentazione delle attività di verifica;
iv) la valutazione dei risultati di verifica ottenuti;
v) la valutazione dei requisiti di affidabilità;
vi) i ruoli e le responsabilità di quanti coinvolti nel processo di prova;
vii) il grado di copertura della prova che si richiede sia raggiunto.

11.4.5 The Software Integration Test Plan shall document the following:

i) test cases and test data;
ii) types of tests to be performed;
iii) test environment, tools, configuration and programs;
iv) test criteria on which the completion of the test will be judged.

Il Piano di Prova di Integrazione del Software deve documentare quanto segue:

i) casi di prova e dati di prova;
ii) tipi di prove da eseguire;
iii) ambiente di prova, strumenti (tools), configurazione e programmi;
iv) criteri di prova in base ai quali sarà giudicato il completamento delle prove.

11.4.6 In each development phase it shall be shown that the functional, reliability, performance and safety requirements are met.

In ogni fase di sviluppo deve essere mostrato che sono soddisfatti i requisiti di funzionalità, affidabilità, prestazione e sicurezza.

11.4.7 Verification shall be carried out by an independent party to the extent required by the software safety integrity level as defined in Figure 5.

La verifica deve essere svolta da una parte indipendente nella misura in cui è richiesto dal livello di integrità della sicurezza del software come definito in Fig. 5.

11.4.8 Testing which is not fully documented and is performed by the designer prior to verification shall not be regarded as part of the verification.

La prova che non è completamente documentata ed è eseguita dal progettista prima della verifica non deve essere considerata una parte della verifica.

11.4.9 The results of each verification shall be retained in a form defined or referenced in the Software Verification Plan such that it is auditable.

I risultati di ogni verifica devono essere conservati in un formato definito o referenziati nel Piano di Verifica del Software in modo da poter essere sottoposti a verifica ispettiva.

11.4.10 After each verification activity a verification report shall be produced stating either that the software has passed the verification or the reasons for the failures. The verification reports shall address the following:

i) items which do not conform to the Software Requirements Specification, Software Design Specification or Software Module Design Specifications;
ii) items which do not conform to the Software Quality Assurance Plan;
iii) modules, data, structures and algorithms poorly adapted to the problem;
iv) detected errors or deficiencies;
v) the identity and configuration of the items verified.

Dopo ogni attività di verifica deve essere prodotto un rapporto di verifica che dichiari se il software ha superato la verifica o i motivi del fallimento. I rapporti di verifica devono trattare quanto segue:

i) elementi (items) che non sono conformi alla Specifica dei Requisiti del Software, alla Specifica della progettazione del Software o alle Specifiche di Progetto del Modulo Software;
ii) elementi (items) che non sono conformi al Piano di Assicurazione della Qualità del Software;
iii) moduli, dati, strutture e algoritmi che si adattano poco al problema;
iv) errori rilevati o difetti;
v) l'identità e la configurazione degli elementi (items) verificati.

11.4.11 Software Requirements Verification**:** Once the Software Requirements Specification has been established, verification shall address:

i) the adequacy of the Software Requirements Specification in fulfilling the requirements set out in the System Requirements Specification, the System Safety Requirements Specification and the Software Quality Assurance Plan;
ii) the adequacy of the Software Requirements Test Specification as a test of the Software Requirements Specification;
iii) the internal consistency of the Software Requirements Specification.

The results shall be recorded in a Software Requirements Verification Report.

Verifica dei Requisiti del Software: Una volta che è stata stabilita la Specifica dei Requisiti del Software la verifica deve trattare:

i) l'adeguatezza della Specifica dei Requisiti del Software nel soddisfare i requisiti definiti nella Specifica dei Requisiti del Sistema, nella Specifica dei Requisiti di Sicurezza del Sistema e nel Piano di Assicurazione della Qualità del Software;
ii) l'adeguatezza della Specifica di Prova dei Requisiti del Software come prova della Specifica dei Requisiti del Software;
iii) la consistenza interna della Specifica dei Requisiti del Software.

I risultati devono essere registrati nel Rapporto di Verifica dei Requisiti del Software.

11.4.12 Software Architecture and Design Verification: After the Software Architecture Specification and the Software Design Specification have been established, verification shall address:

i) the adequacy of the Software Architecture Specification and the Software Design Specification in fulfilling the Software Requirements Specification;
ii) the adequacy of the Software Design Specification for the Software Requirements Specification with respect to consistency and completeness;

L'Architettura del Software e la Verifica della progettazione: dopo che sono state definite la Specifica dell'Architettura del Software e la Specifica della Progettazione del Software la verifica deve trattare:

i) l'adeguatezza della Specifica dell'Architettura del Software e della Specifica della Progettazione del Software nel soddisfare la Specifica dei Requisiti del Software;
ii) l'adeguatezza della Specifica della Progettazione del Software alla Specifica dei Requisiti del Software riguardo a consistenza e completezza;

iii) the adequacy of the Software Integration Test Plan as a set of test cases for the Software Architecture Specification and the Software Design Specification;

iv) the internal consistency of the Software Architecture and Design Specifications.

The results shall be recorded in a Software Architecture and Design Verification Report.

iii) l'adeguatezza del Piano di Prova di Integrazione del Software come un insieme di casi di prova per la Specifica dell'Architettura del Software e la Specifica della Progettazione del Software;

iv) la consistenza interna dell'Architettura e delle Specifiche di Progettazione del Software.

I risultati devono essere registrati in un Rapporto dell'Architettura del Software e di Verifica della Progettazione.

11.4.13 Software Module Verification: After each Software Module Design Specification has been established, verification shall address:

i) the adequacy of the Software Module Design Specification in fulfilling the Software Design Specification;

ii) the adequacy of the Software Module Test Specification as a set of test cases for the Software Module Design Specification;

iii) the decomposition of the Software Design Specification into software modules and the Software Module Design Specifications with reference to
- feasibility of the performance required,
- testability for further verification,
- readability by the development and verification team, and
- maintainability to permit further evolution;

iv) the adequacy of the Software Module Test Reports as a record of the tests carried out in accordance with the Software Module Test Specification.

The results shall be recorded in a Software Module Verification Report.

Verifica del Modulo Software: dopo che ogni Specifica della Progettazione del Modulo Software è stata definita, la verifica deve trattare:

i) l'adeguatezza della Specifica della Progettazione del Modulo Software nel soddisfare la Specifica della Progettazione del Software;

ii) l'adeguatezza della Specifica di Prova del Modulo Software come insieme di casi di prova per la Specifica della Progettazione del Modulo Software;

iii) la scomposizione della Specifica della Progettazione del Software in moduli software e le Specifiche di Progettazione del Modulo Software con riferimento a
- fattibilità delle prestazioni richieste,
- possibilità di prova per ulteriori verifiche,
- leggibilità da parte del gruppo di sviluppo e verifica, e
- manutenibilità per ottenere ulteriori sviluppi;

iv) l'adeguatezza dei Rapporti di Prova dei Moduli Software come registrazione delle prove eseguite in accordo con la Specifica di Prova del Modulo Software.

I risultati devono essere registrati in un Rapporto di Verifica del Modulo Software.

11.4.14 Software Source Code Verification: To the extent demanded by the software safety integrity level the Software Source Code shall be verified to ensure conformance to the Software Module Design Specification and the Software Quality Assurance Plan. This shall include a check to determine whether the coding standards have been applied correctly.

The results shall be recorded in a Software Source Code Verification Report.

Verifica del Codice Sorgente del Software: nella misura in cui è richiesto dal livello di integrità della sicurezza del software il Codice Sorgente del Software deve essere verificato per assicurare conformità alla Specifica di Progettazione del Modulo Software e al Piano di Assicurazione della Qualità del Software. Questo deve comprendere una verifica per stabilire se gli standard di codifica (programmazione) sono stati applicati correttamente.

I risultati devono essere registrati in un Rapporto di Verifica del Codice Sorgente del Software.

11.4.15 A Software Integration Test Report shall be produced as follows:

i) a Software Integration Test Report shall be produced stating the test results and whether the objectives and criteria of the Software Integration Test Plan have been met. If there is a failure, the reasons for the failure shall be recorded;

Un Rapporto di Prova di Integrazione del Software deve essere prodotto come segue:

i) deve essere prodotto un Rapporto di Prova di Integrazione del Software che dichiari i risultati delle prove e dove sono stati soddisfatti gli obiettivi e i criteri del Piano di Prova di Integrazione del Software. Se vi è un mancato funzionamento, devono essere registrati i motivi del mancato funzionamento;

ii) the Software Integration Test Report shall be in a form that is auditable;

iii) test cases and their results shall be recorded, preferably in machine readable form for subsequent analysis;

iv) tests should be repeatable and be performed by automatic means,if practicable;

v) the identity and configuration of the items verified.

ii) il Rapporto di Prova di Integrazione del Software deve essere in un formato che ne consenta la verifica ispettiva;

iii) i casi di prova e i loro risultati devono essere registrati, preferibilmente in forma leggibile dalla macchina per analisi successive;

iv) le prove dovrebbero essere ripetibili ed essere eseguite con mezzi automatici, se possibile;

v) l'identità e la configurazione degli elementi (items) verificati.

11.4.16 For software/hardware integration, see 12.4.8.

Per l'integrazione software/hardware, vedere 12.4.8.

12 SOFTWARE/HARDWARE INTEGRATION

INTEGRAZIONE SOFTWARE/HARDWARE

12.1 Objectives

Obiettivi

12.1.1 To demonstrate that the software and the hardware interact correctly to perform their intended functions.

Dimostrare che il software e l'hardware interagiscono correttamente per eseguire le loro previste funzioni.

12.1.2 To combine the software and hardware, ensuring their compatibility, to meet the System Safety Requirements Specification and the requirements of the intended software safety integrity level.

Mettere insieme il software e l'hardware, assicurando la loro compatibilità, per soddisfare la Specifica dei Requisiti di Sicurezza del Sistema e i Requisiti del livello previsto di integrità della sicurezza del software.

12.2 Input documents

Documenti d'ingresso

1) System Requirements Specification
2) System Safety Requirements Specification
3) System Architecture Description
4) Software Requirements Specification
5) Software Requirements Test Specification
6) Software Architecture Specification
7) Software Design Specification
8) Software Module Design Specification
9) Software Module Test Specification
10) Software Source code and supporting documentation
11) Hardware documentation

1) Specifica dei Requisiti del Sistema
2) Specifica dei Requisiti di Sicurezza del Sistema
3) Descrizione dell'Architettura del Sistema
4) Specifica dei Requisiti del Software
5) Specifica di prova dei Requisiti del Software
6) Specifica dell'Architettura del Software
7) Specifica della Progettazione del Software
8) Specifica della progettazione del Modulo Software
9) Specifica di Prova del Modulo Software
10) Codice Sorgente del Software e documentazione di supporto
11) Documentazione Hardware

12.3 Output documents

Documenti di uscita

1) Software/Hardware Integration Test Plan
2) Software/Hardware Integration Test Report

1) Piano di Prova di Integrazione Software/Hardware
2) Rapporto di Prova dell'Integrazione Software/Hardware

12.4 Requirements

Requisiti

12.4.1 For software safety integrity levels greater than zero, a Software/Hardware Integration Test Plan will be created early in the development lifecycle, in order that integration activities may be properly directed and that particular design or other integration needs may be suitably provided for. During development (and depending

Per livelli di integrità della sicurezza del software maggiori di zero, sarà creato, all'inizio del ciclo di vita dello sviluppo, un Piano di Prova di Integrazione Software/Hardware, in modo che le attività di integrazione possano essere orientate in modo appropriato e che il progetto specifico o altre necessità di integrazione possano essere previste in

upon the size of the system) the plan may be subdivided into a number of child documents and be naturally added to, as the hardware and software designs evolve and the detailed needs of integration become clearer.

modo idoneo. Durante lo sviluppo (e in relazione alla dimensione del sistema) il piano può essere suddiviso in più documenti figlio ed essere naturalmente aggiuntimentre i progetti hardware e software evolvono e divengono più chiare le necessità dettagliate di integrazione.

12.4.2 The Software/Hardware Integration Test Plan shall document the following:

i) test cases and test data;
ii) types of tests to be performed;
iii) test environment including tools, support software and configuration description; and
iv) test criteria on which the completion of the test will be judged.

Il Piano di Prova di Integrazione Software/Hardware deve documentare quanto segue:

i) casi di prova e dati di prova;
ii) tipi di prove da eseguire;
iii) ambiente di prova, comprendente strumenti, software di supporto e descrizione della configurazione; e
iv) criteri di prova per mezzo dei quali sarà stimata la completezza della prova.

12.4.3 The Software/Hardware Integration Test Plan shall distinguish between those activities which can be carried out by the developer on his premises and those that require access to the user's site.

Il Piano di Prova di Integrazione Software/Hardware deve distinguere tra quelle attività che possono essere eseguite da chi sviluppa nel suo ambiente e quelle che richiedono di accedere all'ambiente dell'utilizzatore.

12.4.4 The Software/Hardware Integration Test Plan shall distinguish between the following activities:

i) merging of software onto the target hardware;
ii) system integration;

Il Piano di Prova di Integrazione Software/Hardware deve fare distinzione tra le seguenti attività:

i) allocazione del software nell'hardware a cui è destinato;
ii) integrazione del sistema;

12.4.5 Tools and facilities identified in the Software/Hardware Integration Test Plan should be available at the earliest practicable time.

Strumenti e supporti identificati nel Piano di Prova di Integrazione Software/Hardware dovrebbero essere disponibili per quanto possibile fin dall'inizio.

12.4.6 During Software/Hardware Integration any modification or change to the integrated system shall be subject to an impact study which shall identify all modules impacted and the necessary reverification activities.

Durante l'Integrazione Software/Hardware ogni modifica o cambiamento al sistema integrato deve essere soggetto ad uno studio dell'impatto che deve identificare tutti i moduli coinvolti e le necessarie attività di riverifica

12.4.7 Test cases and their results shall be recorded, preferably in machine readable form for subsequent analysis.

I casi di prova e i loro risultati devono essere registrati, preferibilmente in forma leggibile da macchine per successive analisi.

12.4.8 A Software/Hardware Integration Test Report shall be produced as follows:

i) Software/Hardware Integration Test Report shall state the test results and whether the objectives and criteria of the Software/Hardware Integration Test Plan have been met. If there is a failure, it shall be recorded;
ii) the Software/Hardware Integration Test Report shall be in a form that is auditable;
iii) test cases and their results shall be recorded, preferably in amachine-readable form for subsequent analysis;
iv) the Software/Hardware Integration Test Report shall also contain evidence that the Verifier is satisfied with the adequacy of the Software/Hardware Integration Test Report as a record of the tests carried out in ac-

Un Rapporto di Prova di Integrazione Software/Hardware deve essere prodotto come segue:

i) Il Rapporto di Prova di Integrazione Software/Hardware deve riportare i risultati di prova e se gli obiettivi e i criteri del Piano di Prova di Integrazione Software/Hardware sono stati soddisfatti. Se vi è un mancato funzionamento, questo deve essere registrato;
ii) Il Rapporto di Prova dell'Integrazione Software/Hardware deve essere in un formato che ne consenta la verifica ispettiva;
iii) I casi di prova e i loro risultati devono essere registrati, preferibilmente in forma leggibile da macchine per successive analisi;
iv) Il Rapporto di Prova dell'Integrazione Software/Hardware deve anche contenere l'attestazione che il Verificatore è soddisfatto per l'adeguatezza del Rapporto di Prova dell'Integrazione Software/Hardware inteso come raccolta delle

cordance with the Software/Hardware Integration Test Plan;

v) the Software/Hardware Integration Test Report shall document the identity and configuration of all items involved.

prove eseguite in accordo con il Piano di Prova di Integrazione Software/Hardware;

v) Il Rapporto di Prova dell'Integrazione Software/Hardware deve documentare l'identità e la configurazione di tutte gli elementi coinvolte.

13 SOFTWARE VALIDATION

VALIDAZIONE DEL SOFTWARE

13.1 Objective

To analyse and test the integrated system to ensure compliance with the Software Requirements Specification with particular emphasis on the functional and safety aspects according to the Software Safety integrity level.

Obiettivi

Analizzare e provare il sistema integrato per assicurare la conformità con la Specifica dei Requisiti del Software con particolare enfasi per gli aspetti funzionali e di sicurezza considerando il livello di integrità della sicurezza del Software.

13.2 Input documents

1) Software Requirements Specification
2) All Hardware and Software Documentation
3) System Safety Requirements Specification

Documenti d'ingresso

1) Specifica dei Requisiti del Software
2) Tutta la Documentazione Hardware e Software
3) Specifica dei Requisiti di Sicurezza del Sistema

13.3 Output documents

1) Software Validation Plan
2) Software Validation Report

Documenti di uscita

1) Piano di Validazione del Software
2) Rapporto di Validazione del Software

13.4 Requirements

Requisiti

13.4.1 Analysing and testing shall be the main validation activity.

L'analisi e le prove devono essere le principali attività di validazione.

13.4.2 Simulation and modelling may be used to supplement the validation process.

La simulazione e la modellazione possono essere usate per completare il processo di validazione.

13.4.3 A Software Validation Plan shall be established and detailed in suitable documentation.

Un Piano di Validazione del Software deve essere stabilito e dettagliato in una appropriata documentazione.

13.4.4 The Software Validation Plan shall be developed, performed and results evaluated by an independent party to the extent required by the software safety integrity level.

Il Piano di Validazione del Software deve essere sviluppato, eseguito ed i risultati valutati da una parte indipendente nella misura in cui è richiesto dal livello di integrità della sicurezza del software.

13.4.5 If required by the software safety integrity level, the scope and contents of the Software Validation Plan shall be agreed with the assessor. This agreement shall also make a statement concerning the presence of the assessor during testing.

Se richiesto dal livello di integrità della sicurezza del software, lo scopo e i contenuti del Piano di Validazione del Software devono essere concordati con il valutatore. Questo accordo deve prevedere anche una dichiarazione riguardante la presenza del valutatore durante le prove.

13.4.6 The Software Validation Plan shall include a summary justifying the validation strategy chosen. The justification shall include consideration, according to the required software safety integrity level, of:

i) manual or automated techniques or both;
ii) static or dynamic techniques or both;
iii) analytical or statistical techniques or both.

Il Piano di Validazione del Software deve comprendere una premessa che giustifichi la strategia di validazione scelta. La giustificazione deve comprendere considerazioni, secondo il livello di integrità della sicurezza del software richiesto, circa:

i) tecniche manuali o automatiche, o entrambe;
ii) tecniche statiche o dinamiche o entrambe;
iii) tecniche analitiche o statistiche o entrambe.

13.4.7 The Software Validation Plan shall identify the steps necessary to demonstrate the adequacy of the:

- Software Requirements Specification;
- Software Architecture Specification;
- Software Design Specification;
- Software Module Design Specification;

in fulfilling the safety requirements set out in the System Safety Requirements Specification. The Validator shall check that the verification process is complete.

Il Piano di Validazione del Software deve identificare i passi necessari a dimostrare l'adeguatezza della:

- Specifica dei Requisiti del Software;
- Specifica dell'Architettura del Software;
- Specifica della Progettazione del Software;
- Specifica della Progettazione del Modulo Software;

nel soddisfare i requisiti di sicurezza prestabiliti nella Specifica dei Requisiti di Sicurezza del Sistema. Il Validatore deve controllare che il processo di verifica è completo.

13.4.8 Measurement equipment used for validation shall be calibrated appropriately. Any tools, hardware or software, used for validation shall be shown to be suitable for the purpose.

Le apparecchiature di misura usate per la validazione devono essere tarate in modo appropriato. Deve essere dimostrato che ogni strumento, hardware o software, usato per la validazione è adatto allo scopo.

13.4.9 The software shall be exercised by simulation of input signals present during normal operation, anticipated occurrences and undesired conditions requiring system action.

Il software deve essere stimolato attraverso la simulazione dei segnali d'ingresso presenti durante il normale funzionamento, eventi anticipati e condizioni non desiderate che richiedono azioni del sistema.

13.4.10 The results of the validation shall be documented in the Software Validation Report in an auditable form.

I risultati della validazione devono essere documentati nel Rapporto di Validazione del Software in un formato che ne consenta la verifica ispettiva.

13.4.11 Once hardware/software integration is finished, a Software Validation Report shall be produced as follows:

i) it shall state whether the objectives and criteria of the Software Validation Plan have been met. If there is a failure, it shall be recorded;
ii) it shall state the tests results and whether the whole software on its target machine fulfils the requirements set out in the Software Requirements Specification;
iii) an evaluation of the test coverage on the requirements of the Software Requirements Specification shall be provided;
iv) the Software Validation Report shall be in a form that is auditable;
v) test cases and their results shall be recorded in a machine readable form for subsequent analysis;
vi) tests should be repeatable and be performed by automatic means, if practicable.

Una volta che l'integrazione hardware/software è finita, deve essere prodotto un Rapporto di Validazione del Software come segue:

i) deve dichiarare se gli obiettivi e i criteri del Piano di Validazione del Software sono stati soddisfatti. Se vi è un mancato funzionamento, questo deve essere registrato;
ii) deve dichiarare i risultati delle prove e se l'intero software sulla sua macchina definitiva soddisfa i requisiti stabiliti nella Specifica dei Requisiti del Software;
iii) deve essere fornita una valutazione della copertura delle prove rispetto ai requisiti della Specifica dei Requisiti del Software;
iv) il Rapporto di Validazione del Software deve essere in un formato che ne consenta la verifica ispettiva;
v) i casi di prova e i loro risultati devono essere registrati in una forma leggibile da macchine per successive analisi;
vi) le prove dovrebbero essere ripetibili ed essere eseguite, se possibile, con mezzi automatici.

13.4.12 The Software Validation Report shall document the identity and configuration of all:

i) the hardware and software used;
ii) the equipment used;
iii) the equipment's calibration;
iv) the simulation models used;
v) the discrepancies found;
vi) the corrective actions performed.

Il Rapporto di Validazione del Software deve documentare l'identità e la configurazione di tutto:

i) l'hardware e il software usato;
ii) le apparecchiature usate;
iii) la taratura delle apparecchiature;
iv) i modelli di simulazione usati;
v) le discrepanze trovate;
vi) le azioni correttive eseguite.

13.4.13 Any discrepancies found, including detected errors, shall be clearly identified in a separate section of the Software Validation Report and included in any release note which accompanies the delivered software.

Ogni discrepanza trovata, compresi gli errori scoperti, deve essere chiaramente identificata in una sezione separata del Rapporto di Validazione del Software e inserita in ogni nota sulla versione che accompagna il software consegnato.

13.4.14 The software shall be tested against the Software Requirements Test Specification. These tests shall show that all of the requirements in the Software Requirements Specification are correctly performed.

The results shall be recorded in a Software Validation Report.

Il software deve essere provato in base alla Specifica di Prova dei Requisiti del Software. Queste prove devono dimostrare che tutti i requisiti della Specifica dei Requisiti del Software sono correttamente eseguiti.

I risultati devono essere registrati in un Rapporto di Validazione del Software.

14 SOFTWARE ASSESSMENT — VALUTAZIONE DEL SOFTWARE

14.1 Objective — Obiettivi

To evaluate that the lifecycle processes and products resulting are such that the software is of the defined software safety integrity level and is fit for its intended application.

Valutare che i processi del ciclo di vita e i prodotti risultanti sono tali che il software è del livello di integrità della sicurezza del software definito ed è adatto per l'applicazione voluta.

14.2 Input documents — Documenti d'ingresso

1) System Safety Requirements Specification
2) All Hardware and Software Documentation

1) Specifica dei Requisiti di Sicurezza del Sistema
2) Tutta la documentazione Hardware e Software

14.3 Output documents — Documenti di uscita

Software Assessment Report

Rapporto di Valutazione del Software

14.4 Requirements — Requisiti

14.4.1 **For software of safety integrity level zero:**

i) the Assessor shall only be required to confirm that this is the appropriate software safety integrity level;
ii) it is also possible to agree this level between the supplier and the user at the time of tendering.

Per il software di livello di integrità della sicurezza zero:

i) al Valutatore deve solo essere richiesto di confermare che questo è il livello appropriato di integrità della sicurezza del software;
ii) è anche possibile concordare questo livello tra il fornitore e l'utente al momento dell'offerta.

14.4.2 Software with a Software Assessment Report from another Assessor does not have to be an object for an entirely new assessment. The second Assessor shall check that the software is of the required software safety integrity level and that it is fit for its intended application on the intended target computer.

Software con un Rapporto di Valutazione del Software di un altro Valutatore non deve essere oggetto di una valutazione interamente nuova. Il secondo Valutatore deve verificare che il software ha il livello di integrità della sicurezza del software richiesto e che è adatto all'applicazione prevista sull'elaboratore di destinazione previsto.

14.4.3 The Assessor shall have access to the design and development process and all project related documentation.

Il Valutatore deve avere accesso al processo di progettazione e di sviluppo e a tutta la documentazione relativa al progetto.

14.4.4 The assessment of the software shall be carried out by an Assessor who is independent from the design team.

La valutazione del software deve essere eseguita da un Valutatore che è indipendente dal gruppo di progetto.

14.4.5 The Assessor shall assess that the software of the system is fit for its intended purpose and responds correctly to safety issues derived from the System Safety Requirements Specification.

Il Valutatore deve valutare che il software del sistema è adatto allo scopo voluto e che risponde correttamente alle condizioni di sicurezza che derivano dalla Specifica dei Requisiti di Sicurezza del Sistema.

14.4.6 To the extent required by the software safety integrity level, the Assessor shall decide if appropriate methods have been selected and applied at each phase of the software lifecycle.

Nella misura in cui è richiesto dal livello di integrità della sicurezza del software, il Valutatore deve decidere se sono stati scelti ed applicati ad ogni fase del ciclo di vita del software metodi appropriati.

14.4.7 If required by the software safety integrity level, the Assessor shall agree the scope and contents of the Software Validation Plan. This agreement shall also make a statement concerning the presence of the Assessor during testing.

Se richiesto dal livello di integrità della sicurezza del software, il Valutatore deve concordare lo scopo e i contenuti del Piano di Validazione del Software. Questo accordo deve anche produrre una dichiarazione riguardo alla presenza del Valutatore durante le prove.

14.4.8 The Assessor may ask for additional verification and validation work if he so chooses.

Il Valutatore, se lo decide, può chiedere ulteriore lavoro di verifica e validazione.

14.4.9 The Assessor shall produce a report for each review which shall detail his assessment results.

Il Valutatore deve produrre per ogni riesame un rapporto che deve dettagliare i risultati della sua valutazione.

14.4.10 If, in the opinion of the Assessor, the software is fit for its intended application, the final Software Assessment Report shall include a statement as to the software safety integrity level achieved by the software.

Se, a giudizio del Valutatore, il software è adatto all'applicazione prevista, il Rapporto finale di Validazione del Software deve includere una dichiarazione circa il livello di integrità della sicurezza del software raggiunto dal software.

14.4.11 When the software is not fit for its purpose or has not achieved the required software safety integrity level then the Assessor shall only report the non-conformities in the Software Assessment Report and shall not give any technical solution.

Quando il software non è adatto al suo scopo o non ha raggiunto il livello di integrità della sicurezza del software richiesto, il Valutatore deve solo riportare le non conformità nel Rapporto di Valutazione del Software e non deve dare alcuna soluzione tecnica.

15 SOFTWARE QUALITY ASSURANCE — ASSICURAZIONE DELLA QUALITÀ DEL SOFTWARE

15.1 Objectives — Obiettivi

15.1.1 To identify, monitor and control all those activities, both technical and managerial, which are necessary to ensure that the software achieves the quality required. This is necessary to provide the required qualitative defence against systematic faults and to ensure that an audit trail can be established to allow verification and validation activities to be undertaken effectively.

Identificare, sorvegliare e controllare tutte quelle attività, sia tecniche che gestionali, che sono necessarie per assicurare che il software raggiunga la qualità richiesta. Questo è necessario per fornire la richiesta difesa qualitativa nei confronti dei guasti sistematici e per assicurare che possa essere stabilito un percorso di verifiche ispettive per permettere di intraprendere efficacemente le attività di verifica e validazione.

15.1.2 To provide evidence that the above activities have been carried out.

Fornire evidenza che le attività di cui sopra sono state eseguite.

15.2 Input documents — Documenti d'ingresso

All the documents available at each stage of the lifecycle

Tutti i documenti disponibili in ogni fase del ciclo di vita.

15.3 Output documents

1) Software Quality Assurance Plan
2) Software Configuration Management Plan

All the above plans shall be issued at the beginning of the project and updated during the life-cycle.

Documenti di uscita

1) Piano di Assicurazione della Qualità del Software
2) Piano di Gestione della Configurazione del Software

Tutti i piani di cui sopra devono essere prodotti all'inizio del progetto ed aggiornati nel corso del ciclo di vita.

15.4 Requirements

Requisiti

15.4.1 The supplier and/or developer shall have and use as a minimum a Quality Assurance System compliant with EN ISO 9000 series, to support the requirements of this European Standard. EN ISO 9001 accreditation is highly recommended.

Il fornitore e/o il realizzatore devono avere ed adottare, come minimo, un Sistema di Assicurazione della Qualità conforme alle norme della serie EN ISO 9000, per supportare i requisiti della presente Norma Europea. È altamente raccomandato che sia conseguita la certificazione EN ISO 9001.

15.4.2 As a minimum, the supplier and/or developer and the customer shall implement for the software development the relevant parts of EN ISO 9001, in accordance with the guidelines contained in EN ISO 9000-3.

Come minimo, il fornitore e/o il realizzatore ed il cliente devono implementare per lo sviluppo del software, le rispettive parti della EN ISO 9001, in accordo con le linee guida contenute nella EN ISO 9000-3.

15.4.3 The supplier and/or developer shall prepare and document, on a project by project basis, a Software Quality Assurance Plan to implement the requirements of 15.4.1 and 15.4.2 of this European Standard, which shall be expressed in measurable terms wherever possible.

Il fornitore e/o il realizzatore devono preparare e documentare, progetto per progetto un Piano di Assicurazione della Qualità del Software per implementare i requisiti degli art. 15.4.1 e 15.4.2 della presente Norma Europea, che devono essere espressi, ove possibile, in termini misurabili.

15.4.4 The Software Quality Assurance Plan shall have a paragraph specifying details about its own updating throughout the project: frequency, responsibility, method.

Il Piano di Assicurazione della Qualità del Software deve prevedere un paragrafo che specifichi i dettagli relativi al suo aggiornamento nel corso del progetto: frequenza, responsabilità, metodo.

15.4.5 All activities, actions, documents, etc. required by all the sections of EN ISO 9000-3 and of this European Standard (annex A included) shall be specified or referenced in the Software Quality Assurance Plan and tailored to the specific development. None of the lists in EN ISO 9000-3 shall be presumed to be exhaustive.

As a minimum, except at software safety integrity level zero, the following items shall also be specified or referenced in the Software Quality Assurance Plan.

This is to ensure that all the safety aspects in the Software with respect to the required Software Safety integrity level will be covered.

The present list is not exhaustive:

i) definition of the life-cycle model definition of each phase including:
   - activities and elementary tasks;
   - entry and exit criteria;
   - inputs and outputs of each phase;
   - major quality activities in each phase;

Tutte le attività, azioni, documenti, ecc. richiesti da tutte le sezioni della EN ISO 9000-3 e della presente Norma Europea (Allegato A compreso) devono essere specificate o richiamate nel Piano di Assicurazione della Qualità del Software ed essere adattate allo sviluppo specifico. Nessun elenco della EN ISO 9000-3 deve essere considerato esaustivo.

Come minimo, eccetto che per il livello di integrità della sicurezza del software zero, devono essere anche specificate o richiamate nel Piano di Assicurazione della Qualità del Software le seguenti indicazioni.

Ciò serve ad assicurare che tutti gli aspetti di sicurezza del Software nei confronti del livello di integrità della sicurezza del Software richiesto, saranno coperti.

Il presente elenco non è esaustivo:

i) definizione del modello del ciclo di vita definizione di ogni fase comprendendo:
   - attività e compiti elementari;
   - criteri di ingresso e di uscita;
   - ingressi ed uscite di ogni fase;
   - principali attività della qualità in ogni fase;

- organisational unit responsible for each activity and elementary task;

ii) requirements traceability;

iii) documentation structure traceability;

iv) documentation associated with the development, verification and validation, operation and maintenance of software;

v) system integration procedures;

vi) coding standards to be used;

vii) assessment of previous validation tests;

viii) the definition of the metrics (quantitative measures) to be carried out on both the product and the process. For the software product metrics carried out, reference shall be made to the quality characteristics and evaluation guidelines defined by ISO/IEC 9126.

- responsabile dell'unità organizzativa per ogni attività e compito elementare;

ii) tracciabilità dei requisiti;

iii) tracciabilità della struttura della documentazione;

iv) documentazione associata allo sviluppo, verifica e validazione, utillizzazione e manutenzione del software;

v) procedure di integrazione del sistema;

vi) standard di codifica da usare;

vii) valutazione delle prove di validazione precedenti;

viii) definizione delle metriche (misure quantitative) da adottare sia per il prodotto che per il processo. Per le metriche adottate per il prodotto software, si deve fare riferimento alle caratteristiche di qualità e alle linee guida di valutazione definite nella ISO/IEC 9126.

15.4.6 As a minimum, configuration management shall be carried out in accordance with the guidelines contained in EN ISO 9000-3.

Each software document shall be placed under configuration control before the release of its first approved version. The software source code shall be placed under configuration control before the commencement of documented module testing.

It shall not be possible to make any unauthorised changes to any item under Configuration Management Control. Precautions shall be taken to prevent or detect errors occurring in machine readable code during storage, transfer, transmission or duplication.

Configuration Management shall not be limited to the strict product development and maintenance, but it shall also cover the environment used during the full lifecycle. This extension, necessary for the reproducibility of the development and for the maintenance activities, shall include computer configuration files, assemblers, compilers, debuggers and all the other used tools.

Come minimo, la gestione della configurazione deve essere eseguita in accordo con le linee guida contenute nella EN ISO 9000-3.

Ogni documento software deve essere sottoposto al controllo di configurazione prima del rilascio della sua prima versione approvata. Il codice sorgente del software deve essere sottoposto al controllo della configurazione prima dell'inizio delle prove di modulo documentate.

Non deve essere possibile eseguire alcun cambiamento non autorizzato per ogni elemento sotto il Controllo di Gestione della Configurazione. Si devono prendere precauzioni per impedire o rilevare errori che si verificano nel codice leggibile dalla macchina durante la memorizzazione, il trasferimento, la trasmissione o la duplicazione.

La Gestione della Configurazione non deve essere limitata allo stretto sviluppo e manutenzione del prodotto, ma deve anche coprire l'ambiente utilizzato durante tutto il ciclo di vita. Questa estensione, necessaria per la riproducibilità dello sviluppo e per le attività di manutenzione, deve comprendere i file di configurazione dell'elaboratore, gli assemblatori, i compilatori, i programmi per localizzare e rimuovere gli errori e tutti gli altri strumenti usati.

15.4.7 The adequacy and results of Software Verification Plans shall be examined.

Devono essere esaminati l'adeguatezza e i risultati del Piano di Verifica del Software.

15.4.8 The supplier and/or developer shall establish, document and maintain procedures for External Supplier Control, including:

- methods to ensure that software provided by external suppliers adheres to established requirements. Previously developed software shall be assured to be compliant with the required software safety integrity level and dependability. New software shall be developed and maintained in conformity with the Software Quality Assurance Plan of the Supplier or with a specific Software Quality Assurance Plan prepared by the external supplier in accordance with the Soft-

Il fornitore e/o il costruttore devono stabilire, documentare e manutenere le procedure per il Controllo dei Fornitori Esterni, compresi:

- metodi per assicurare che il software fornito da fornitori esterni sia aderente ai requisiti stabiliti. Il software sviluppato in precedenza deve essere conforme al livello di integrità della sicurezza del software richiesto e alla fidatezza. Il software nuovo deve essere sviluppato e manutenuto in conformità al Piano di Assicurazione della Qualità del Software del Fornitore o allo specifico Piano di Assicurazione della Qualità del Software preparato dal fornitore esterno in accordo con il Piano di

ware Quality Assurance Plan of the Supplier;
- methods to ensure that the requirements provided to the External Software Supplier are adequate and complete.

Assicurazione della Qualità del Software del Fornitore;
- metodi per assicurare che i requisiti forniti al Fornitore di Software Esterno siano adeguati e completi.

15.4.9 The supplier and/or developer shall establish, document and maintain procedures for Problem Reporting and Corrective Actions. These procedures, as part of the Quality Assurance System, shall implement the relevant parts of EN ISO 9001, especially covering at least the following aspects:

- define the documentation needed for problem reporting and/or corrective actions, with the aim of giving feedback to the responsible management;
- define analysis of the information collected in the problem reports to identify its causes;
- define the practices to be followed for reporting, tracking and resolving problems identified both during the development phase and during software maintenance;
- define preventive actions to deal with problems to a level corresponding to the required software safety integrity level;
- define the specific organisational responsibilities with regard to development and software maintenance;
- define how to apply controls to ensure that corrective actions are taken and that they are effective;
- define the forms to be used;
- define the requirements for re-test, re-verification, re-validation and re-assessment.

As a minimum, problem reporting and corrective action management shall be applied in the software lifecycle starting immediately after Software Integration and before the starting of formal Software Validation, also covering the whole phase of Software Maintenance.

Il fornitore e/o il costruttore devono stabilire, documentare e mantenere procedure per il Rapporto su Problemi ed Azioni Correttive. Queste procedure, come parte del Sistema di Assicurazione Qualità, devono implementare le relative parti della EN ISO 9001, coprendo specialmente, almeno i seguenti aspetti:

- definire la documentazione necessaria per il rapporto su problemi e/o azioni correttive, con lo scopo di darne conto ai gestori responsabilì;
- definire l'analisi delle informazioni raccolte nei rapporti sui problemi per identificarne le cause;
- definire la prassi da seguire per emettere rapporti, tracciare e risolvere problemi identificati sia durante la fase di sviluppo che durante quella di manutenzione del software;
- definire le azioni preventive per trattare i problemi ad un livello corrispondente a quello richiesto dal livello di integrità della sicurezza del software;
- definire le responsabilità specifiche dell'organizzazione con riguardo allo sviluppo e alla manutenzione del software;
- definire come applicare i controlli per assicurare che le azioni correttive siano intraprese e che esse siano efficaci;
- definire i moduli da usare;
- definire i requisiti per ripetere le prove, la verifica, la validazione e la valutazione.

Come minimo, il rapporto dei problemi e la gestione delle azioni correttive devono essere applicati nel ciclo di vita iniziando immediatamente dopo l'Integrazione del Software e prima di iniziare la formale Validazione del Software, coprendo anche l'intera fase di Manutenzione del Software.

16 SOFTWARE MAINTENANCE

MANUTENZIONE DEL SOFTWARE

16.1 Objective

To ensure that the software performs as required, preserving the required software safety integrity level and dependability when making corrections, enhancements or adaptations to the software itself.

Obiettivi

Assicurare che il software operi come previsto, mantenendo il livello di integrità della sicurezza del software richiesto e la fidatezza quando si pongono in atto correzioni, miglioramenti, o adattamenti del software stesso.

16.2 Input documents

All documents

Documenti d'ingresso

Tutti i documenti

16.3 Output documents

1) Software Maintenance Plan
2) Software Change Records
3) Software Maintenance Record

16.4 Requirements

16.4.1 As a minimum, maintenance shall be carried out in accordance with the guidelines contained in EN ISO 9000-3.

In addition, the following requirements concerning software maintenance shall also be met.

16.4.2 Maintainability shall be designed into the software system, in particular, by following the requirements of clause 10 of this European Standard. ISO/IEC 9126 should also be employed in order to require and verify a minimum level of maintainability.

16.4.3 Procedures for the maintenance of software shall be established and recorded in the Software Maintenance Plan. These procedures shall also include:

i) control of error reporting, error logs, maintenance records, change authorisation and software/system configuration;

ii) verification, validation and assessment; and

iii) definition of the Authority which approves the changed software.

16.4.4 The maintenance activities shall be audited against the Software Maintenance Plan, at intervals defined in the Software Quality Assurance Plan.

16.4.5 Maintenance shall be performed with the same level of expertise, tools, documentation, planning and management as the initial development of the system. This shall apply also to configuration management, change control, document control, and independence of involved parties.

16.4.6 This European Standard is not intended to be retrospective. It therefore applies primarily to new developments and only applies in its entirety to existing systems if these are subjected to major modifications. For software safety integrity level 3 or 4, the contracting entities shall, before starting work on any change, decide whether the maintenance actions are to be considered as major or minor or whether the existing maintenance methods for the system are adequate. For software safety integrity levels 0, 1 or 2, the same decision shall be taken by the supplier.

Documenti di uscita

1) Piano di Manutenzione del Software
2) Registrazioni dei Cambiamenti del Software
3) Registrazione delle Manutenzioni del Software

Requisiti

Come minimo, la manutenzione deve essere eseguita in accordo con le linee guida contenute nella EN ISO 9000-3.

Inoltre, devono essere soddisfatti i seguenti requisiti concernenti la manutenzione del software.

La manutenibilità deve essere progettata nel Sistema software, in particolare, seguendo i Requisiti dell'art. 10 della presente Norma Europea. Dovrebbe essere impiegata anche la ISO/IEC 9126 al fine di richiedere e verificare un livello minimo di manutenibilità.

Le procedure di manutenzioni del software devono essere stabilite e registrate nel Piano di Manutenzione del Software. Queste procedure devono comprendere anche:

i) rapporto sul controllo degli errori, raccolta degli errori, registrazioni della manutenzione, autorizzazione di modifiche e configurazione software/sistema;

ii) verifica, validazione e valutazione; e

iii) definizione dall'Autorità che approva i cambiamenti del software.

Le attività di manutenzione devono essere sottoposte a verifica ispettiva a fronte del Piano di Manutenzione del Software, ad intervalli definiti nel Piano di Assicurazione della Qualità del Software.

La manutenzione deve essere eseguita con il medesimo livello di competenza, strumenti, documentazione, pianificazione e gestione, dello sviluppo iniziale del sistema. Questo si deve applicare anche alla gestione della configurazione, al controllo delle modifiche, al controllo della documentazione ed all'indipendenza delle parti coinvolte.

La presente Norma Europea non è previsto che sia retroattiva. Essa si applica pertanto soprattutto a nuovi sviluppi e si applica nella sua interezza solo a quei sistemi esistenti che sono soggetti a modifiche importanti. Per il livello di integrità della sicurezza del software 3 o 4, le entità contraenti devono decidere, prima di iniziare il lavoro per qualsiasi modifica, se le attività di manutenzione devono essere considerate più o meno importanti o se i metodi di manutenzione esistenti per il sistema sono adeguati. Per livelli di integrità della sicurezza del software 0, 1 o 2, la medesima decisione deve essere presa dal fornitore.

16.4.7 External supplier control, problem reporting and corrective actions shall be managed with the same criteria specified in the relevant paragraphs of the Software Quality Assurance clause.

Il controllo dei fornitori esterni, il rapporto sui problemi e sulle azioni correttive devono essere gestiti con i medesimi criteri specificati nei relativi paragrafi dell'articolo dell'Assicurazione della Qualità del Software.

16.4.8 A Software Maintenance Record shall be established for each Software Item before its first release, and it shall be maintained. In addition to the requirements of EN ISO 9000-3 for "Maintenance Records and Reports", this Record shall also include:

i) references to all the Software Change Records for that Software Item;
ii) change consequence information;
iii) test cases for components, including revalidation and regression testing data; and
iv) software configuration history.

Deve essere stabilito e deve essere mantenuto un registro di Manutenzione del Software per ogni elemento di Software prima della sua prima versione. Oltre ai Requisiti della EN ISO 9000-3 per "Registri e Rapporti di Manutenzione", questo registro deve comprendere anche:

i) riferimenti a tutti i Registri di modifica del Software per quell'elemento Software;
ii) informazioni sulle conseguenze delle modifiche;
iii) casi di prova dei componenti, compresa rivalidazione e dati delle prove di regressione; e
iv) storia della configurazione del software.

16.4.9 A Software Change Record shall be established for each maintenance activity. This record shall include:

i) the modification or change request;
ii) an analysis of the impact of the maintenance activity on the overall system, including hardware, software, human interaction and the environment and possible interactions;
iii) the detailed specification of the modification or change; and
iv) revalidation, regression testing and re-assessment of the modification or change to the extent required by the software safety integrity level. The responsibility for revalidation can vary from project to project, according to the software safety integrity level. Also the impact of the modification or change on the process of revalidation can be confined to different system levels (only changed modules, all identified affected modules, the complete system). Therefore the Software Validation Plan shall address both problems, according to the software safety integrity level. The degree of independence of revalidation shall be the same as that for validation.

Deve essere stabilito per ogni attività di manutenzione un registro delle Modifiche del Software. Questo registro deve comprendere:

i) la modifica o la richiesta di modifica;
ii) un'analisi dell'impatto dell'attività di manutenzione sul sistema generale, comprendente l'hardware, il software, l'interazione dell'uomo e l'ambiente e le possibili interazioni;
iii) la specifica dettagliata della modifica o cambiamento; e
iv) rivalidazione, prove di regressione e rivalutazione della modifica o cambiamento nella misura in cui è richiesto dal livello di integrità della sicurezza del software. La responsabilità della rivalidazione può variare da progetto a progetto, secondo il livello di integrità della sicurezza del software. Anche l'impatto della modifica o del cambiamento sul processo di rivalidazione può essere ristretto a livelli di sistema diversi (solo moduli cambiati, tutti i moduli influenzati identificati, il sistema completo). Pertanto il Piano di Validazione del Software deve trattare entrambi i problemi, secondo il livello di integrità della sicurezza del software. Il grado di indipendenza della rivalidazione deve essere lo stesso di quello per la validazione

17 SYSTEMS CONFIGURED BY APPLICATION DATA

SISTEMI CONFIGURATI CON DATI DELL'APPLICAZIONE

17.1 Objectives

Obiettivi

17.1.1 A characteristic feature of railway control and protection systems is the need to design each installation to meet the individual requirements for a specific application. A system configured by application data allows type-approved generic software to be used, with the individual requirements for each installation defined as

Un aspetto caratteristico dei sistemi di controllo e protezione ferroviari è la necessità di progettare ogni installazione per soddisfare i requisiti individuali per una specifica applicazione. Un sistema configurato da dati di applicazione permette di usare un software generico di tipo approvato, con i requisiti specifici per ogni installazione definita come

data (application specific data). This data is normally in the form of tabular information or an application specific language which is interpreted by the generic software.

17.1.2 For a safety critical system, the high level of resources required to develop software to achieve the required system safety integrity level for the system makes the adoption of a system configured by application data very attractive because it allows the re-use of generic software. However, as the safe operation of the system is likely to depend on the correctness of the data, the procedures used for development of the data must also be to an appropriate system safety integrity level.

17.1.3 The sections below describe the requirements of this European Standard for the initial development of the generic software for a system configured by application data, and for the subsequent development of each set of installation-specific data.

17.2 Input documents

1) Software Requirements Specification
2) Software Architecture Specification

17.3 Output documents

1) Application Requirements Specification
2) Data Preparation Plan
3) Data Test Plan
4) Data Test Report

17.4 Requirements

17.4.1 Data Preparation Lifecycle

Figure 6 illustrates a lifecycle for an application making use of hardware and generic software, together with application-specific data. The phases of the lifecycle are as follows.

17.4.1.1 Application Requirements Specification

The requirements for the application shall be defined. This shall include requirements which are specific to the individual installation (e.g. track layout, signal locations, speed limits), and standards which the application must comply with (e.g. signalling principles, system safety integrity levels).

17.4.1.2 Overall Installation Design

The system architecture shall be defined, and the quantity and type of the generic components to be used shall be specified. Those components which contain software shall have been developed in conformance with this European Standard. The functions specified in the requirements shall be allocated to the compo-

dati (dati specifici dell'applicazione). Questi dati sono normalmente nella forma di informazione tabellare o in un linguaggio specifico dell'applicazione che è interpretato dal software generico.

Per un sistema critico per la sicurezza, l'elevato livello di risorse richieste per sviluppare il software per soddisfare il livello di integrità della sicurezza richiesto per il sistema rende vantaggiosa l'adozione di un sistema configurato dai dati dell'applicazione perché consente il riutilizzo di software generico. Tuttavia, poiché il funzionamento sicuro del sistema è verosimilmente dipendente dalla correttezza dei dati, le procedure usate per lo sviluppo dei dati devono anch'esse essere di un appropriato livello di integrità della sicurezza del sistema.

Le sezioni sottostanti descrivono i requisiti della presente Norma Europea per lo sviluppo iniziale del software generico per un sistema configurato dai dati dell'applicazione, e per il successivo sviluppo di ogni insieme di dati specifici dell'applicazione.

Documenti d'ingresso

1) Specifica dei Requisiti del Software
2) Specifica dell'Architettura del Software

Documenti di uscita

1) Specifica dei Requisiti dell'Applicazione
2) Piano di Preparazione dei Dati
3) Piano di Prova dei Dati
4) Rapporto di Prova dei Dati

Requisiti

Ciclo di Vita della Preparazione dei Dati

La figura 6 illustra un ciclo di vita per un'applicazione che utilizza hardware e software generico, con i dati specifici dell'applicazione. Le fasi del ciclo di vita sono le seguenti.

Specifica dei Requisiti dell'Applicazione

I Requisiti per l'applicazione devono essere definiti. Questi possono comprendere requisiti che sono specifici per la singola installazione (ad esempio piano schematico, disposizione dei segnali, limiti di velocità), e norme alle quali l'applicazione deve conformarsi (ad esempio, principi del segnalamento, livelli di integrità della sicurezza del sistema).

Progettazione Generale dell'Installazione

L'architettura del sistema deve essere definita, e la quantità e il tipo dei componenti generici da usare devono essere specificati. Quei componenti che contengono software devono essere sviluppati in conformità con la presente Norma Europea. Le funzioni specificate nei requisiti devono essere allocate nei componenti, e deve essere definita la

nents, and the physical location of each component shall be defined.

localizzazione fisica di ogni componente.

17.4.1.3 Data Preparation

The data preparation process shall include the production of specific information (e.g. control tables), production of the data source code and its compilation, checking and other verification activities, and testing of the application data.

Preparazione dei Dati

Il processo di preparazione dei dati deve comprendere la produzione di specifiche informazioni (ad esempio, tabelle di controllo), la generazione del codice sorgente dei dati e la sua compilazione, il controllo e le altre attività di verifica, e la prova dei dati dell'applicazione.

17.4.1.4 Integration and Acceptance

For some systems the application data will be integrated with the generic hardware and software for a factory test before installation on site. This may not be necessary where a sufficient degree of confidence can be obtained by other means. The equipment shall then be installed on site, and integration tests carried out on the new equipment. Finally the system shall be commissioned as a fully operational system, and a final acceptance process shall be carried out on the complete installation.

Integrazione e Accettazione

Per alcuni sistemi i dati dell'applicazione saranno integrati con l'hardware e il software generico per una prova in fabbrica prima dell'installazione sul sito. Questo può non essere necessario se si può ottenere un sufficiente grado di confidenza con altri mezzi. Le apparecchiature devono quindi essere installate sul sito, ed eseguite le prove di integrazione con le nuove apparecchiature. Infine il sistema deve essere consegnato come sistema completamente funzionante, e deve essere eseguito un processo finale di accettazione sull'installazione completa.

17.4.1.5 Validation and Assessment

Validation and assessment activities shall audit the performance of each stage of the life-cycle.

Validazione e Valutazione

Le attività di validazione e di valutazione devono eseguire una verifica ispettiva della prestazione di in ogni fase del ciclo di vita.

17.4.2 Data Preparation Procedures and Tools

For each new type of system configured by application data, specific data preparation procedures and tools shall be developed to allow the data preparation lifecycle specified in 17.4.1 to be applied to installations of the new system. Development of these procedures and tools shall be carried out in accordance with this European Standard in parallel with the generic software and hardware for the system. The verification, validation and assessment activities shall ensure that the data preparation tools and the generic software are compatible.

Procedure di Preparazione dei Dati e Strumenti (tools)

Per ogni nuovo tipo di sistema configurato dai dati dell'applicazione, devono essere sviluppate procedure di preparazione di dati specifici e strumenti per permettere di applicare all'installazione del nuovo sistema il ciclo di vita della preparazione dei dati specificato in 17.4.1. Lo sviluppo di queste procedure e strumenti deve essere eseguito in accordo con la presente Norma Europea in parallelo al software generico e all'hardware per il sistema. Le attività di verifica, validazione e valutazione devono assicurare che gli strumenti di preparazione dei dati e il software generico siano compatibili.

17.4.2.1 At the Software Design phase for the system configured by application data, a Data Preparation Plan shall be produced to define a documentation structure for the data preparation process. These documents shall be related to the data preparation lifecycle model described in 17.4.1. The plan shall specify the data preparation procedures and tools to be used to meet the required system safety integrity levels. The plan shall specify the requirements for the independence between staff carrying out verification, validation and design tasks.

Nella fase di Progettazione del Software per il sistema configurato dai dati dell'applicazione, deve essere predisposto un Piano di Preparazione dei Dati per definire una struttura della documentazione per il processo di preparazione dei dati. Questi documenti devono essere riferiti al modello del ciclo di vita di preparazione dei dati descritto in 17.4.1. Il piano deve specificare le procedure di preparazione dei dati e gli strumenti da usare per ottenere i livelli di integrità della sicurezza del sistema richiesti. Il piano deve specificare i requisiti per l'indipendenza tra il personale che esegue compiti di verifica, validazione e progettazione.

17.4.2.2 The Data Preparation Plan shall allocate a safety integrity level to any hardware or software tools used in the data preparation lifecycle. This safety integrity level shall be derived from the re-

Il Piano di Preparazione dei Dati deve assegnare un livello di integrità della sicurezza ad ogni strumento, hardware o software, usato nel ciclo di vita di preparazione dei dati. Questo livello di in-

quired system safety integrity level and the degree to which the output of each tool is checked by other manual or automated procedures.

tegrità della sicurezza deve essere derivato dal livello di integrità della sicurezza richiesto per il sistema e dal grado con il quale i dati in uscita di ogni strumento sono verificati con altre procedure automatiche o manuali.

17.4.2.3 Where possible the Data Preparation Plan shall call for notations for specifying requirements and design which are familiar to applications engineers, e.g. standard signalling plans and control tables. Where new notations are introduced, the necessary user documentation must be provided and training shall also be provided where appropriate.

Ove possibile il Piano di Preparazione dei Dati deve richiamare annotazioni per specificare requisiti e progettazione che sono familiari agli ingegneri delle applicazioni, ad esempio, piani di segnalamento normalizzati e tabelle di controllo. Ove sono introdotte nuove notazioni, deve essere fornita la necessaria documentazione per l'utente e deve essere anche fornita l'istruzione se necessario.

17.4.2.4 The verification, test, validation and assessment reports required to demonstrate that the data preparation has been carried out in accordance with the plan can be standardised in the form of checklists to minimise the workload in producing documentation for each installation. This information shall be contained in the Data Test Plan and the results shall be recorded in the Data Test Report.

I rapporti di verifica, prova, validazione e valutazione richiesti per dimostrare che la preparazione dei dati è stata eseguita in accordo con il piano può essere standardizzata nella forma di liste di riscontro per minimizzare il carico di lavoro nella produzione di documentazione per ogni installazione. Queste informazioni devono essere contenute nel Piano di Prova dei Dati e i risultati devono essere registrati nel Rapporto di Prova dei dati.

17.4.2.5 All data and associated documentation shall be subject to the configuration management requirements of section 15 of this standard. The configuration management records shall list the version of generic software with which the data has been designed to operate, and the versions of the tools used in the data preparation process.

Tutti i dati e la documentazione associata deve essere soggetta ai requisiti della gestione della configurazione della Sezione 15 della presente norma. Le registrazioni della gestione della configurazione devono elencare la versione del software generico con la quale i dati sono stati progettati ad operare, e le versioni degli strumenti usati nel processo di preparazione dei dati.

17.4.3 Software Development

Development of the generic software to be used in a system configured by application data shall comply with the requirements in clauses 1 to 16 of this standard. The following additional requirements shall also be observed.

Sviluppo del Software

Lo sviluppo del software generico da usare in un sistema configurato dai dati dell'applicazione deve essere conforme ai requisiti degli art. da 1 a 16 della presente norma. Devono essere osservati anche i seguenti requisiti supplementari.

17.4.3.1 During the Software Requirements Specification phase, those functions which make use of application data in each system and subsystem shall be identified. The system safety integrity level allocated to each sub-system will determine the standards to be applied to the subsequent development of the data for all installations of the system.

Durante la fase della Specificazione dei Requisiti del Software, devono essere identificate quelle funzioni che fanno uso dei dati dell'applicazione in ogni sistema e sottosistema. Il livello di integrità della sicurezza del sistema assegnato ad ogni sottosistema determinerà le norme da applicare allo sviluppo successivo dei dati per tutte le installazioni del sistema.

17.4.3.2 During the Software Design phase the detailed interfaces between the generic software and application data shall be specified, unless this has already been specified at an earlier phase of the lifecycle, for example as a result of a requirement to use an existing application specific language.

Durante la fase di Progettazione del Software devono essere specificate le interfacce dettagliate tra il software generico e i dati di applicazione, salvo che queste non siano già state specificate in una precedente fase del ciclo di vita, per esempio come risultato di un requisito ad usare uno specifico linguaggio di un'applicazione esistente.

17.4.3.3 During the Software Module Design phase a rigid separation between program code and data shall be enforced, i.e. it shall be possible to recompile and update either the generic software or the

Durante la fase di Progetto del Modulo Software deve essere imposta una rigida separazione tra codice del programma e dati, per esempio deve essere possibile ricompilare e aggiornare sia il sof-

data without needing to update the other, unless there has been a change to the defined interface between the software and data. Likewise, application specific data should be separated from other data.

tware generico che i dati senza dover aggiornare l'altro, salvo che non vi sia stato un cambiamento nella interfaccia definita tra il software e i dati. Ugualmente, i dati specifici dell'applicazione dovrebbero essere separati da altri dati.

17.4.3.4 During the Software Maintenance phase, the change control procedures must ensure that any amendment to the generic software may only be installed after it has been established that either the revised software is compatible with the original data, or the data has been revised as necessary.

Durante la fase di Manutenzione del Software, le procedure di controllo delle modifiche devono assicurare che ogni modifica del software generico possa essere installata solo dopo che è stato stabilito che o il software revisionato è compatibile con i dati originali, o i dati sono stati rivisti per quanto necessario.

17.4.3.5 Care must be taken in the Software Verification process and Software Validation phase in order to assure that all relevant combinations of data are considered.

Si deve aver cura di assicurare che nel processo di Verifica del Software e nella fase di Validazione del Software vengano considerate tutte le combinazioni pertinenti dei dati.

17.4.3.6 The generic software shall be designed to detect corrupted configuration data where this is feasible.

Il software generico deve essere progettato per rilevare, quando ciò è fattibile, i dati di configurazione corrotti.

– BLANK PAGE – **– PAGINA BIANCA –**

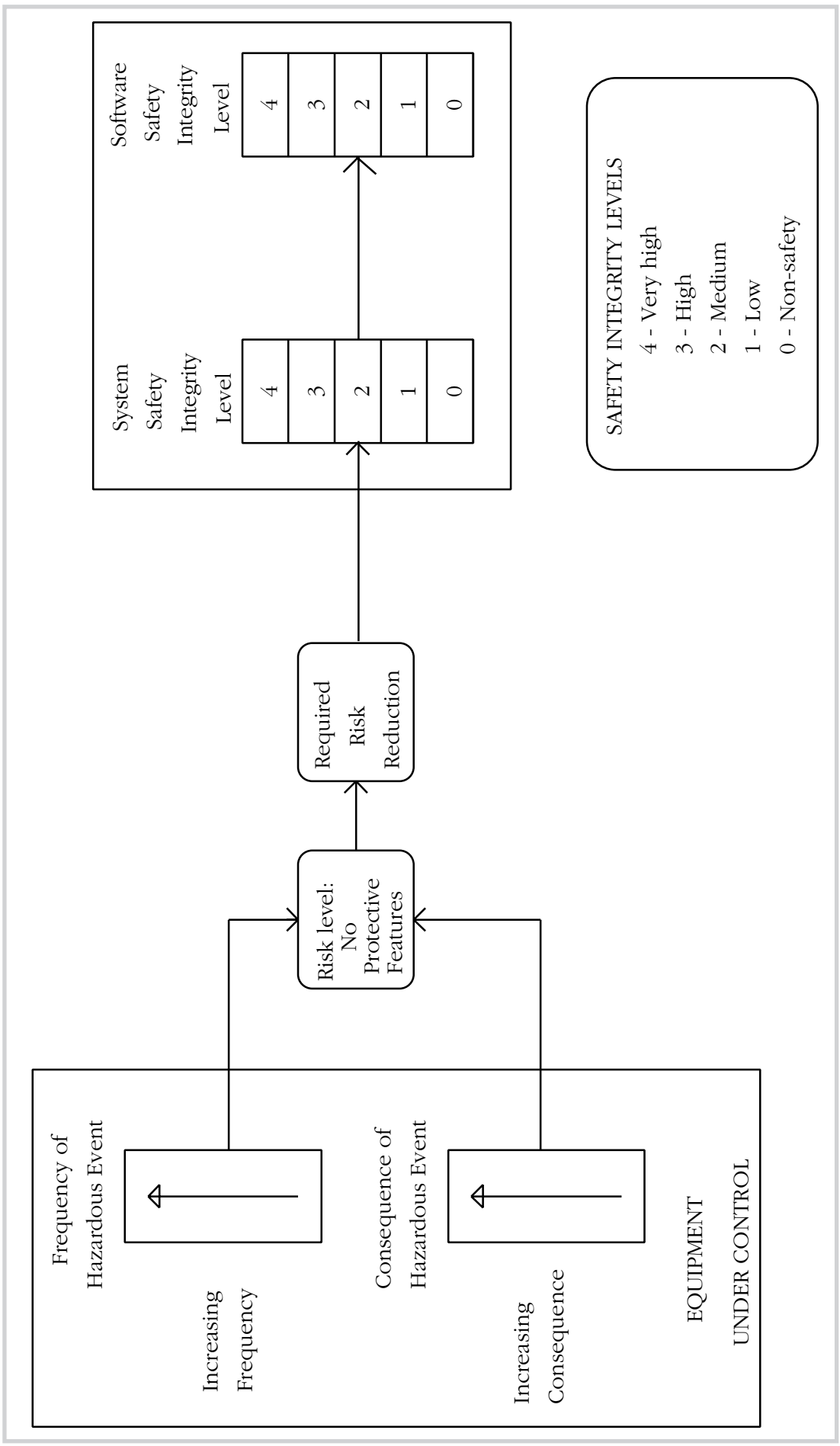


Fig. 1 **Integrity Levels for Safety-Related Systems**

Livelli di Integrità per Sistemi in Sicurezza

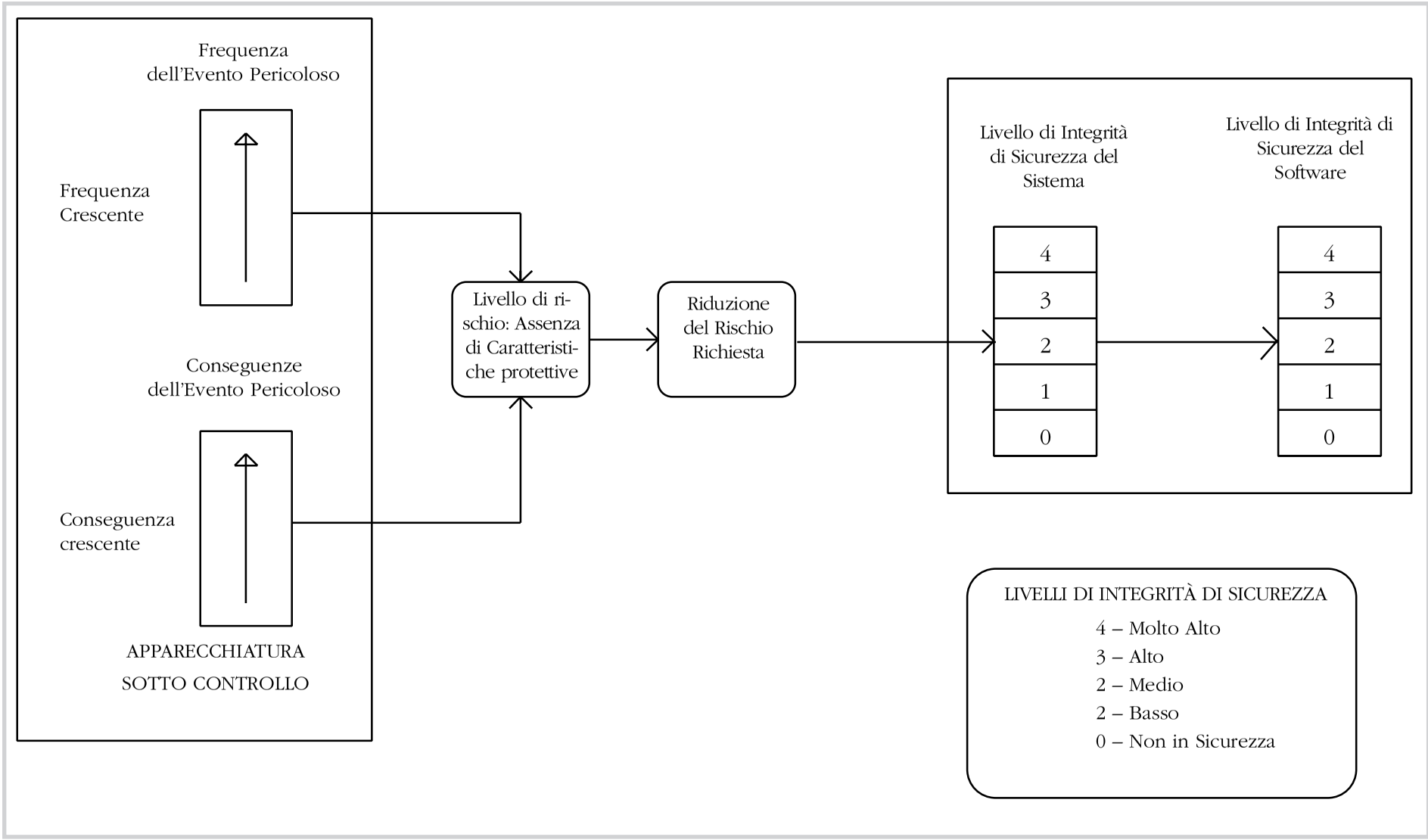
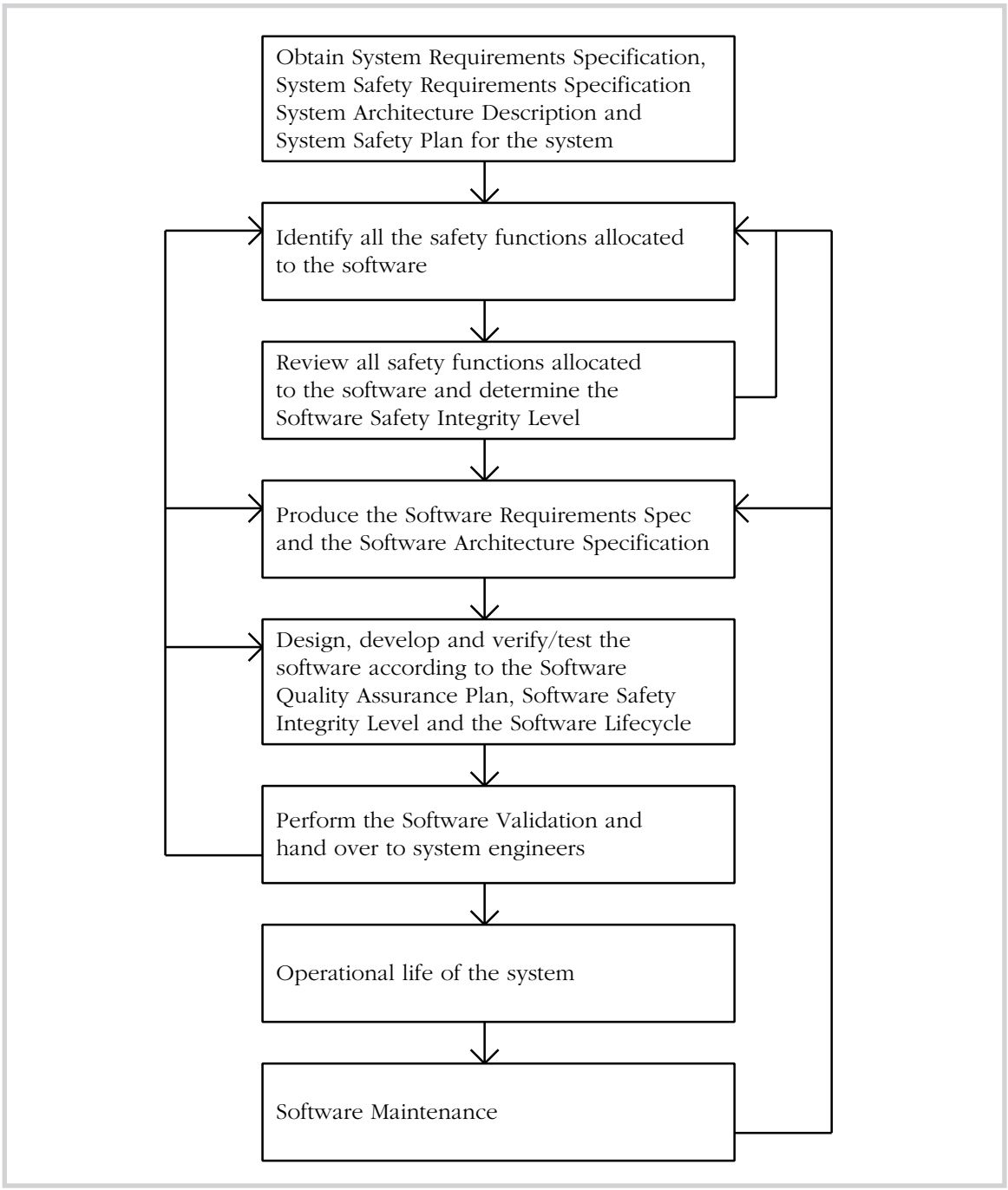


Fig. 2 **Software Safety Route Map**



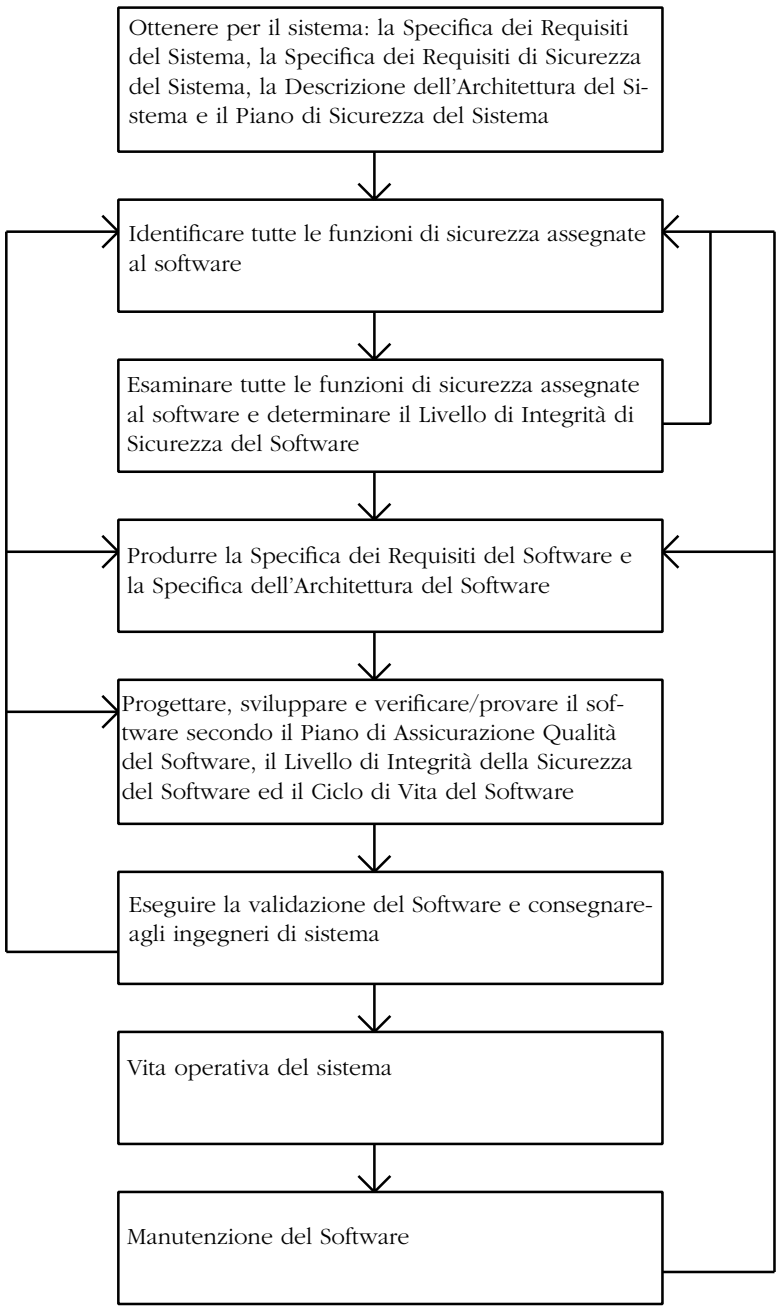
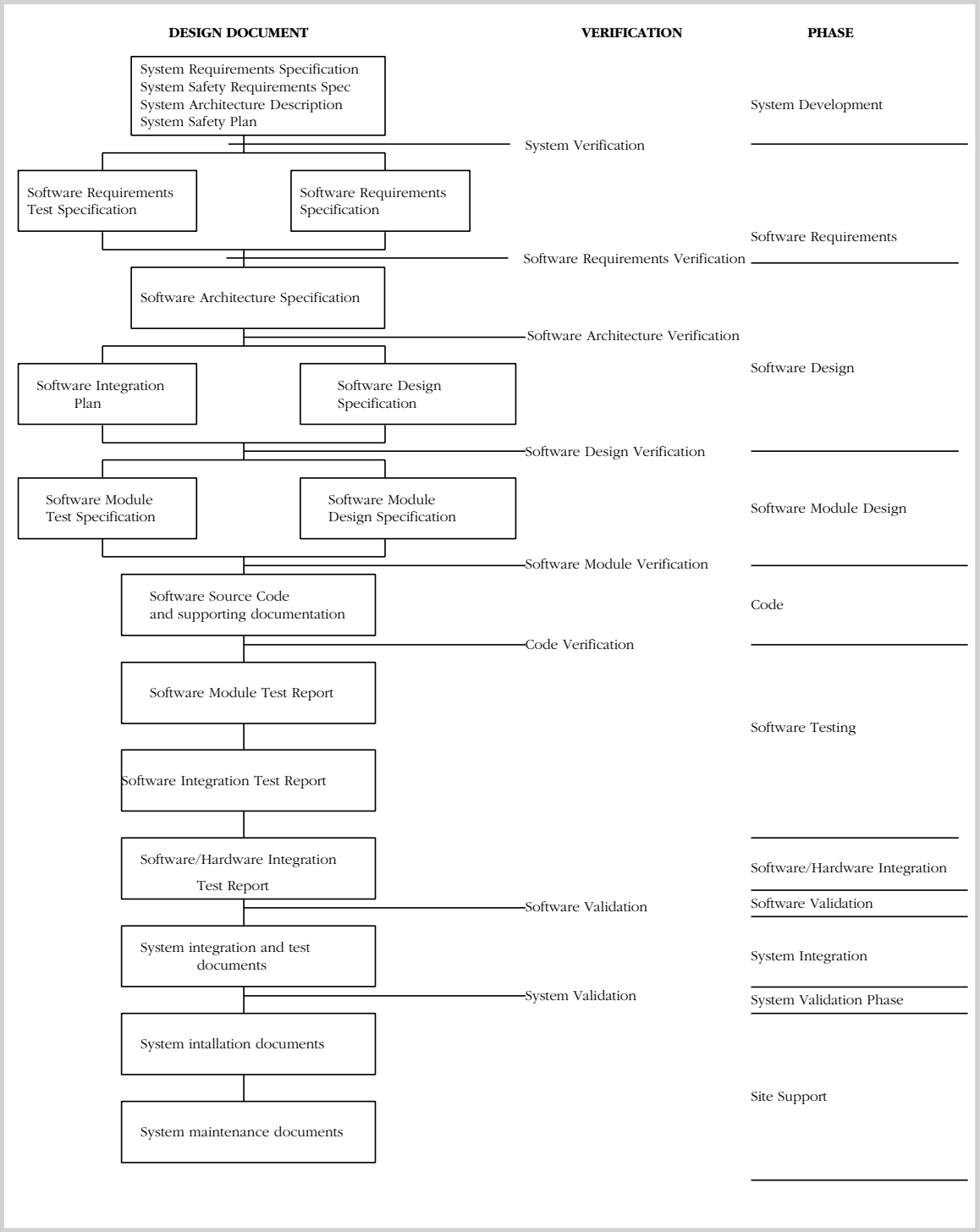


Fig. 3 **Development Lifecycle 1**



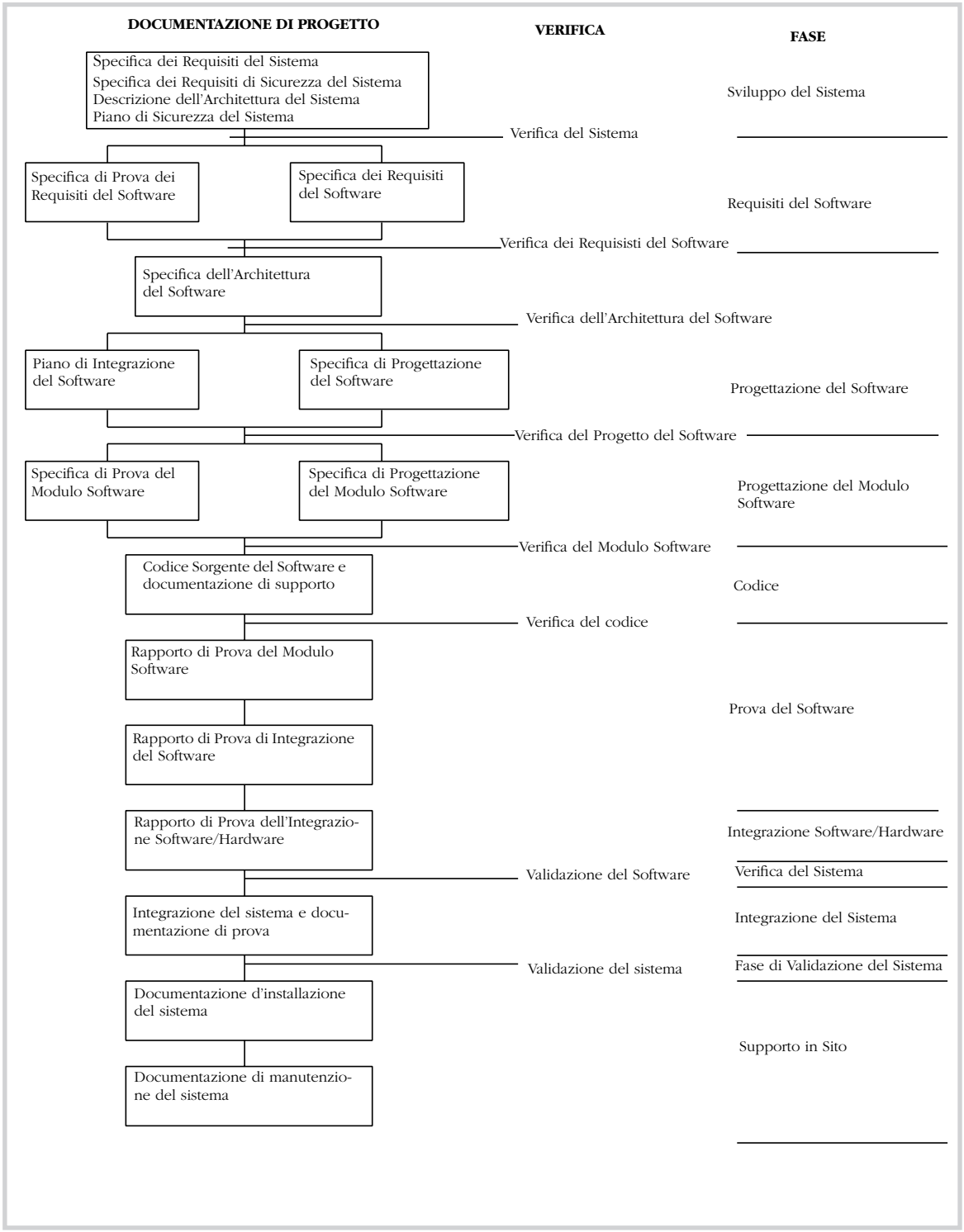
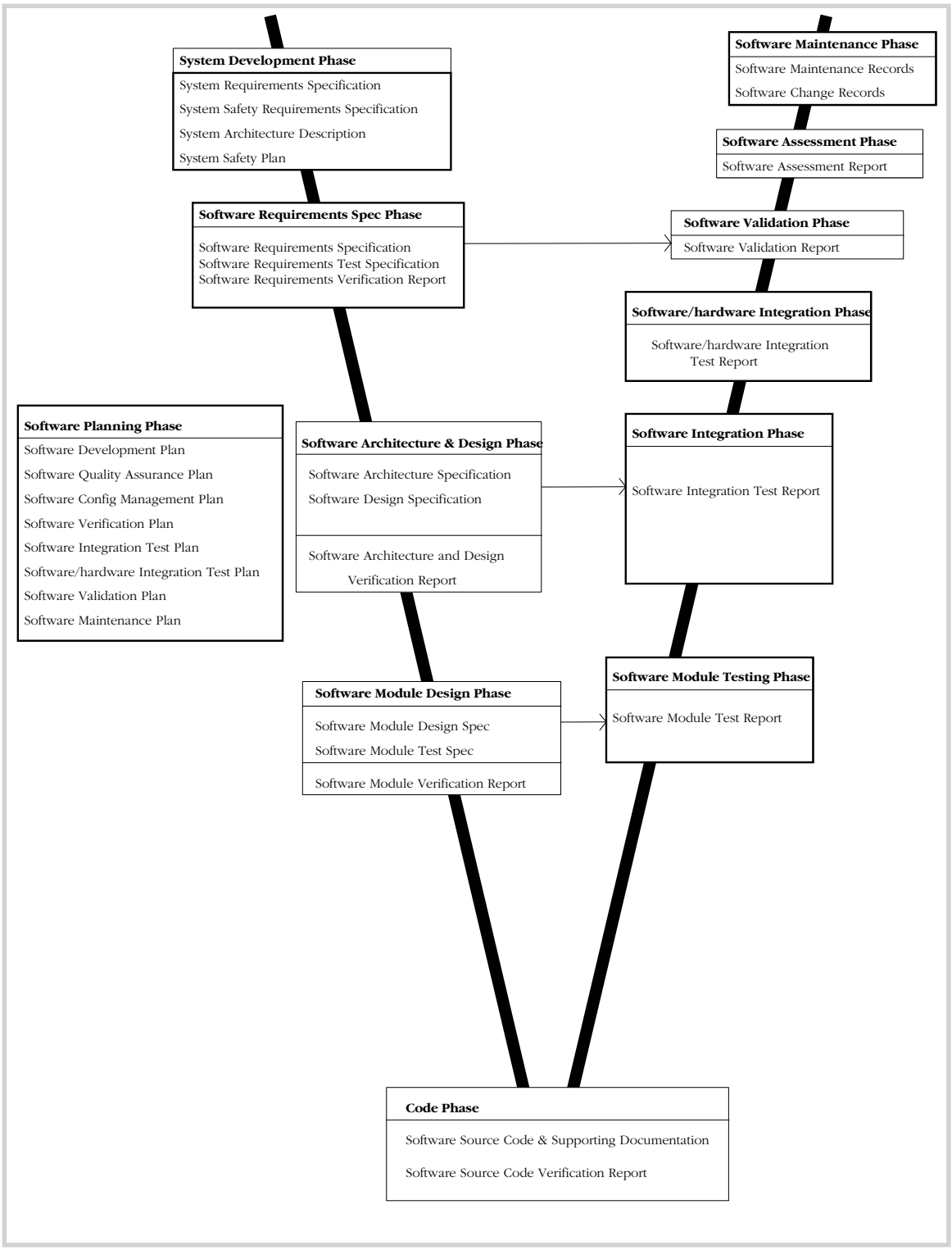


Fig. 4 **Development Lifecycle 2**



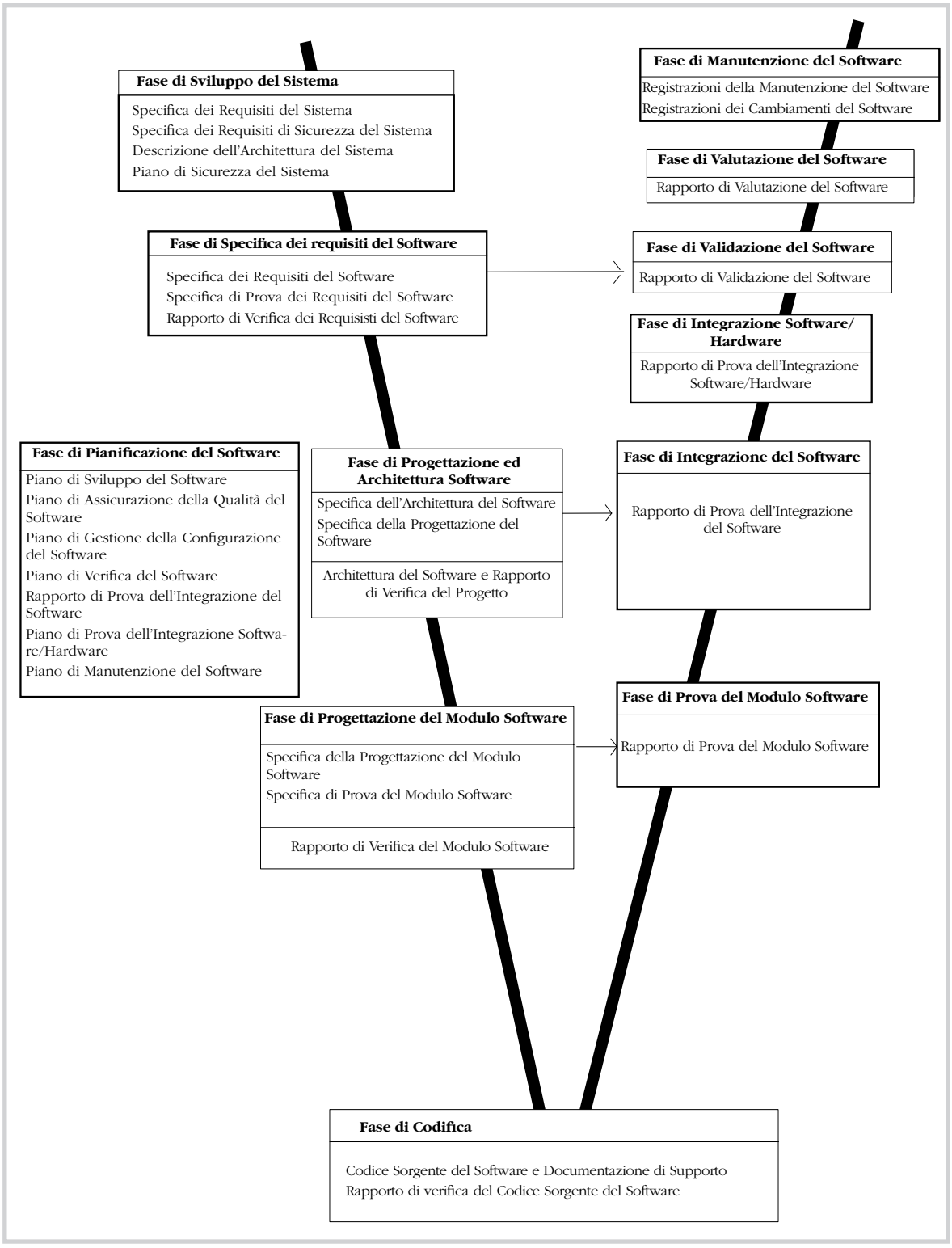
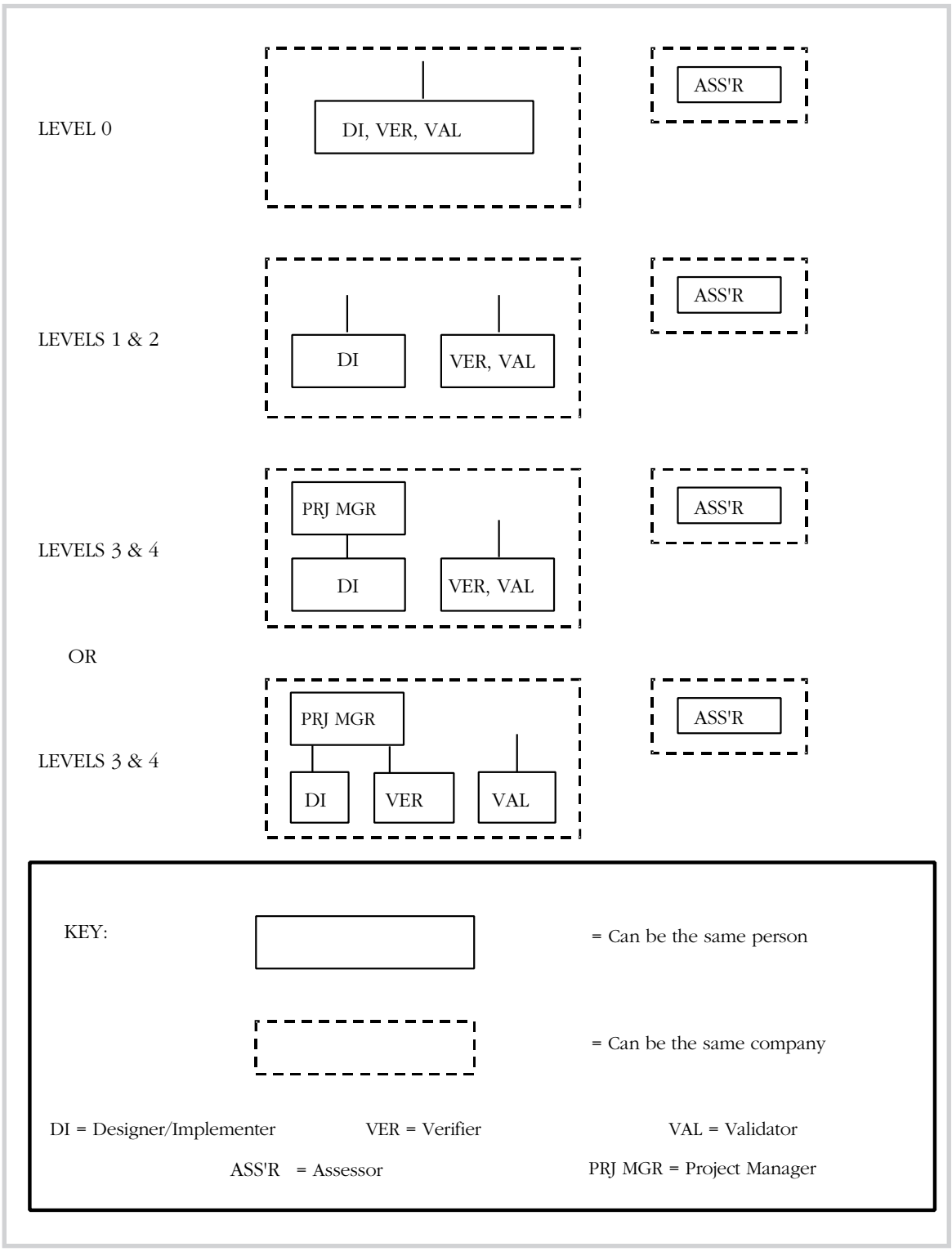


Fig. 5 **Independence Versus Software Integrity Level**



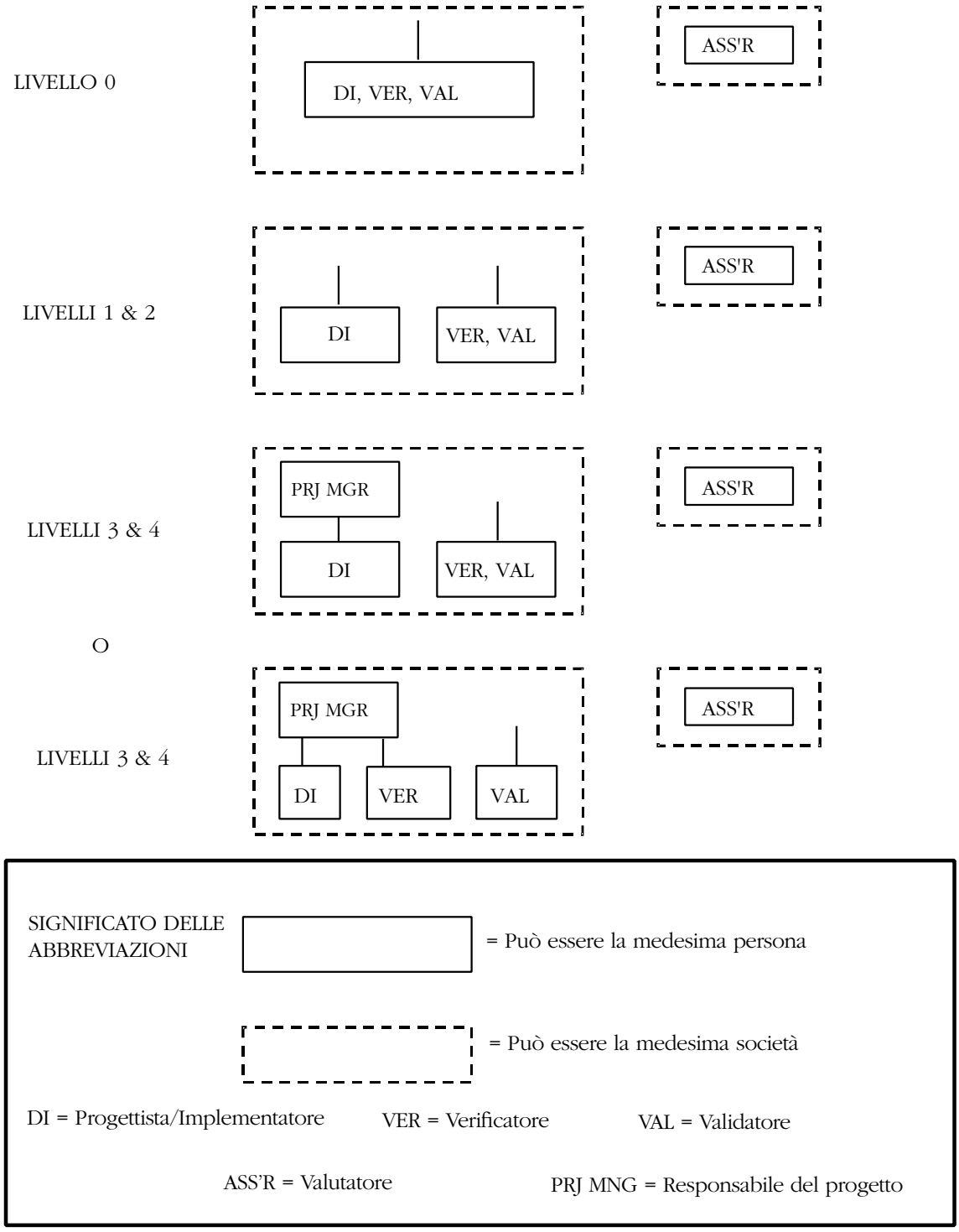
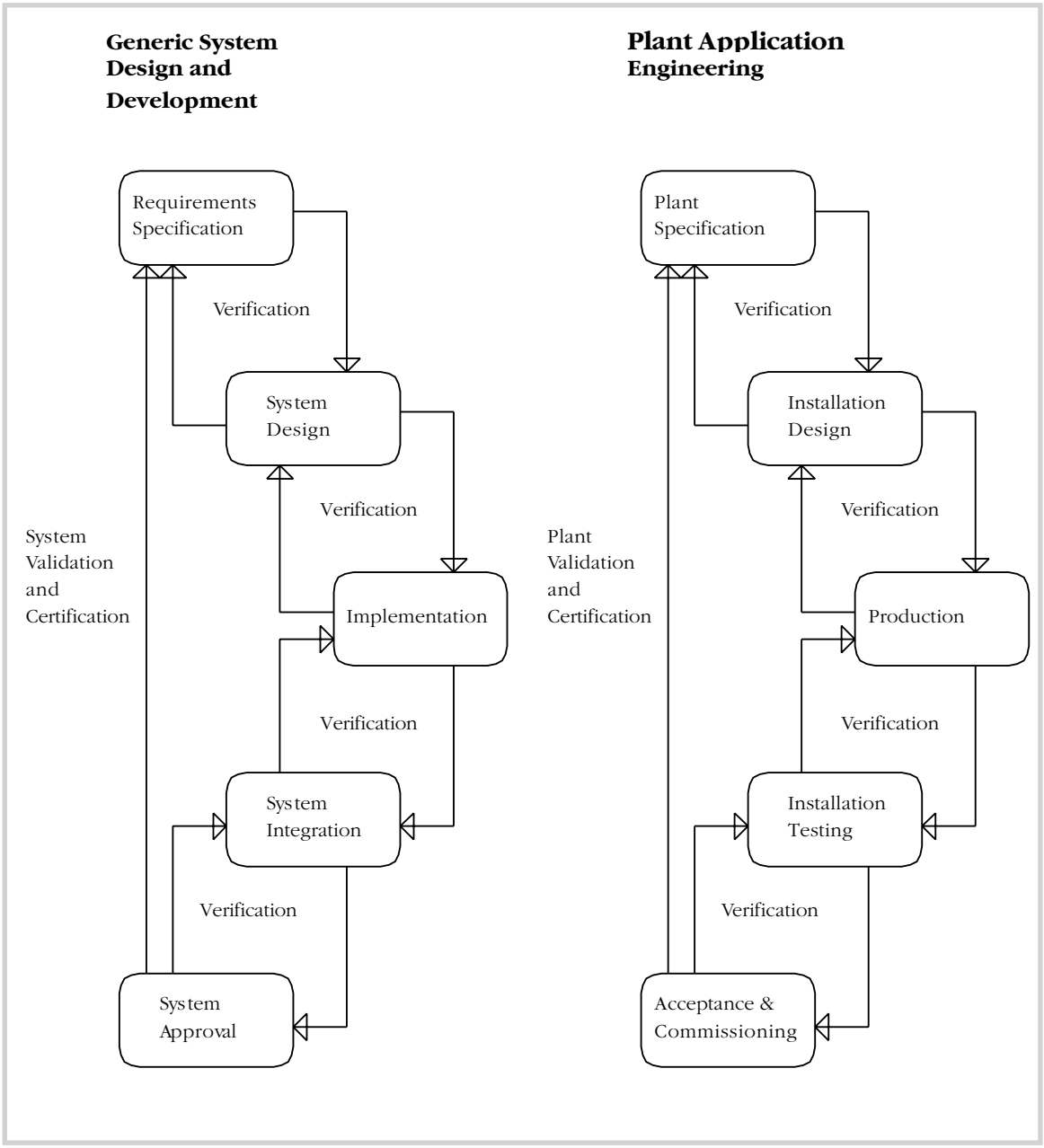
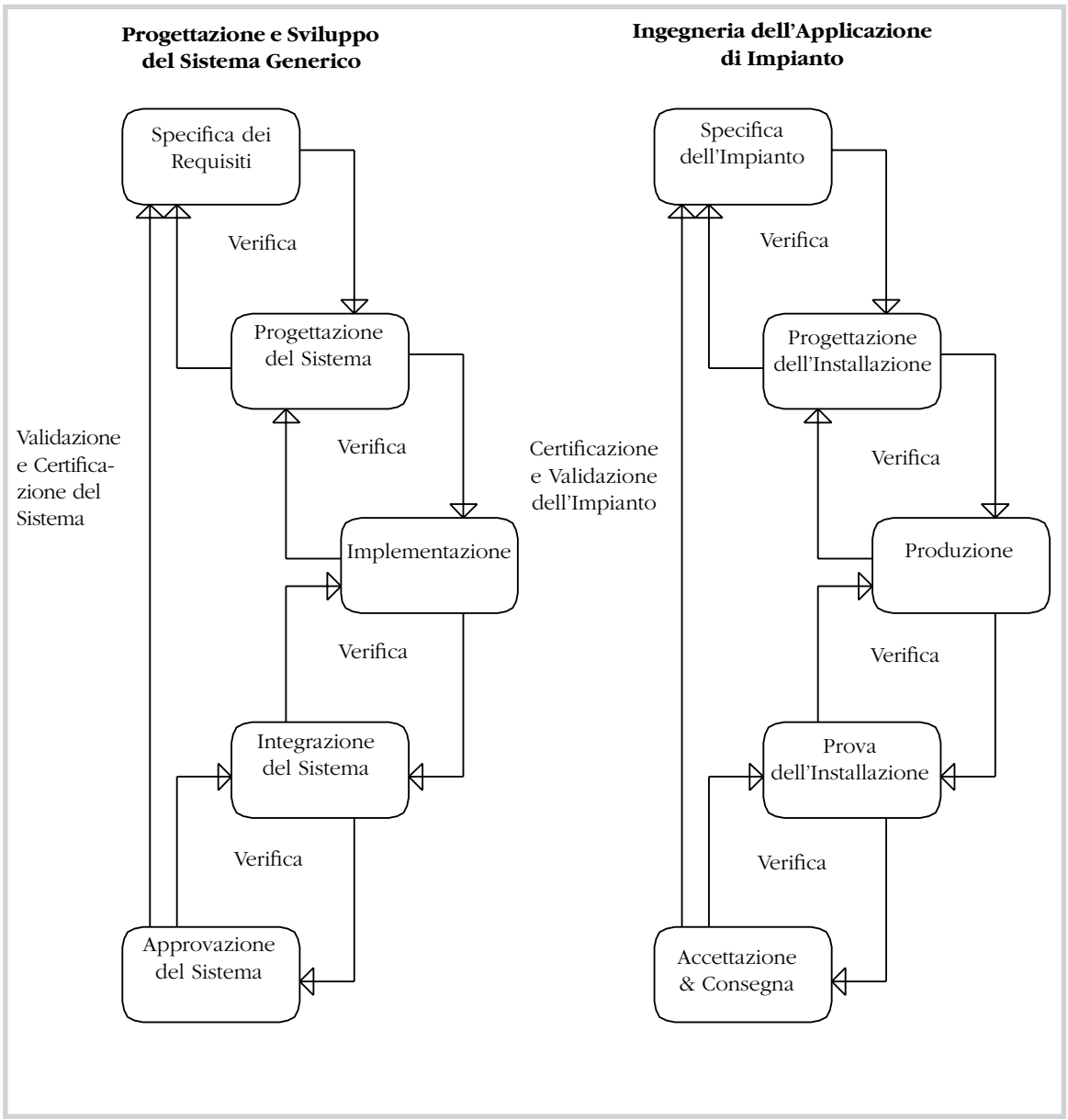


Fig. 6 **Relationship between Generic System Development and Application Development**



Relazione tra Sviluppo del Sistema Generico e Sviluppo dell'Applicazione



CRITERIA FOR THE SELECTION OF TECHNIQUES AND MEASURES

Each of clauses 7 to 16 of this European Standard has an associated clause table to illustrate the means of achieving conformance. There exist lower level tables, the detailed tables, which expand upon certain entries in the clause tables. For example, Semi Formal Methods in the clause 8 table is expanded upon in the detailed Table D.7. There also exists an informative annex B which is referred to from the clause tables.

With each technique or measure in the tables there is a requirement for each software safety integrity level (SWSIL), 1 to 4 and also for the non safety-related level 0. In this version of the document, the requirements for software safety integrity levels 1 and 2 are the same for each technique. Similarly, each technique has the same requirements at software safety integrity levels 3 and 4. These requirements can be:

CRITERI PER LA SCELTA DI TECNICHE E PROVVEDIMENTI

Ciascuna degli art. da 7 a 16 di questa Norma Europea ha una tabella associata all'articolo per illustrare i mezzi per raggiungere la conformità. Vi sono tabelle di livello inferiore, le tabelle di dettaglio, che approfondiscono certe voci nelle tabelle degli articoli. Per esempio, la tabella dei Metodi Semi-Formali dell'art. 8 viene approfondita nella tabella di dettaglio D.7. Vi è anche un Allegato B informativo che fa riferimento alle tabelle dei capitoli.

Nelle tabelle vi è un requisito, associato ad ogni tecnica o provvedimento, per ogni livello di integrità di sicurezza del software (SWSIL), da 1 a 4 e anche per il livello 0 non di sicurezza,. In questa versione del documento, i requisiti per i livelli di integrità di sicurezza del software 1 e 2 sono gli stessi per ogni tecnica. Similmente, ogni tecnica ha gli stessi requisiti per i livelli di integrità di sicurezza del software 3 e 4. Questi requisiti possono essere:

"M" questo simbolo significa che è obbligatorio l'uso di una tecnica;
this symbol means that the use of a technique is mandatory;

"HR" questo simbolo significa che la tecnica o il provvedimento sono Altamente Raccomandati per questo livello di integrità di sicurezza. Se questa tecnica o provvedimento non è usata la ragione per la quale non si usa dovrebbe essere dettagliata nel Piano di Assicurazione Qualità del Software o in altro documento richiamato dal Piano di Assicurazione della Qualità del Software;
this symbol means that the technique or measure is Highly Recommended for this safety integrity level. If this technique or measure is not used then the rationale behind not using it should be detailed in the Software Quality Assurance Plan or in another document referenced by the Software Quality Assurance Plan;

"R" questo simbolo significa che la tecnica o il provvedimento è Raccomandato per questo livello di integrità di sicurezza. Questo, rispetto a "HR" è un livello inferiore di raccomandazione e tali tecniche possono essere combinate per formare parte di un pacchetto;
this symbol means that the technique or measure is Recommended for this safety integrity level. This is a lower level of recommendation than an "HR" and such techniques can be combined to form part of a package;

"-" questo simbolo significa che la tecnica o il provvedimento non hanno raccomandazioni pro o contro il loro uso;
this symbol means that the technique or measure has no recommendation for or against being used;

"NR" questo simbolo significa che la tecnica o il provvedimento è Non Raccomandato per questo livello di integrità di sicurezza. Se questa tecnica o provvedimento è usata, la ragione per la quale si usa dovrebbe essere dettagliata nel Piano di Assicurazione della Qualità del Software o in altro documento richiamato dalPiano di Assicurazione della Qualità del Software.
this symbol means that the technique or measure is positively Not Recommended for this safety integrity level. If this technique or measure is used then the rationale behind using it should be detailed in the Software Quality Assurance Plan or in another document referenced by the Software Quality Assurance Plan.

The combination of techniques or measures are to be stated in the Software Quality Assurance Plan or in another document referenced by the Software Quality Assurance Plan with one or more techniques or measures being selected unless the notes attached to the table makes other requirements. These notes can include reference to approved techniques or approved combinations of techniques. If such techniques or combinations of techniques are used, then the Assessor shall accept them as valid and shall only be concerned that they have been correct-

La combinazione di tecniche o provvedimenti deve essere dichiarata nel Piano di Assicurazione della Qualità del Software o in altro documento richiamato dal Piano di Assicurazione della Qualità del Software selezionando una o più tecniche o provvedimenti da selezionare, salvo che note inserite nella tabella prevedano altri requisiti. Queste note possono contenere riferimenti a tecniche consentite o a combinazioni di tecniche consentite. Se sono usate tali tecniche o combinazioni di tecniche, allora il Valutatore deve accettarle come valide e deve solo essere preoccupato che esse si-

ly applied. If a different set of techniques is used and can be justified, then the Assessor may find this acceptable.

ano correttamente applicate. Se viene usata una diversa serie di tecniche e può essere giustificata, il Valutatore può considerare questo accettabile.

Clause Tables

Tabelle degli articoli

Tab. A.1 **Lifecycle Issues and Documentation (clause 7)**

Argomenti del Ciclo di Vita e Documentazione (art. 7)

DOCUMENTAZIONE *DOCUMENTATION*	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Documenti di Pianificazione del Software *Software Planning Documents*	R	HR	HR	HR	HR
2. Documenti dei Requisiti S/W *S/W Requirements Documents*	R	HR	HR	HR	HR
3. Documenti della Progettazione S/W *S/W Design Documents*	—	HR	HR	HR	HR
4. Documenti del Modulo S/W *S/W Module Documents*	—	HR	HR	HR	HR
5. Codice Sorgente & Documentazione *Source Code & Documentation*	R	HR	HR	HR	HR
6. Rapporti di Prova S/W *S/W Test Reports*	—	HR	HR	HR	HR
7. Rapporto di Prova dell'Integrazione S/W & H/W *S/W & H/W Integration Test Report*	—	HR	HR	HR	HR
8. Rapporto di Validazione S/W *S/W Validation Report*	R	HR	HR	HR	HR
9. Rapporto di Valutazione S/W *S/W Assessment Report*	—	HR	HR	HR	HR
10. Registrazioni di Manutenzione S/W *S/W Maintenance Records*	R	HR	HR	HR	HR

Requisito
Requirement

La conformità con la EN ISO 9000-3 implica la produzione di documentazione adeguata per tutti i Livelli di Integrità di Sicurezza del Software. Per Livello di Integrità di Sicurezza del Software 0 il progettista deve scegliere tipi adeguati di documenti.
Compliance with EN ISO 9000-3 implies the production of adequate documentation for all Software Safety Integrity Levels. For Software Safety Integrity Level 0, the designer shall choose suitable types of document.

Tab. A.2 **Software Requirements Specification (clause 8)**

Specifica dei Requisiti del Software (art. 8)

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Metodi Formali comprendenti per esempio CCS, CSP, HOL, LOTOS, OBJ, Logica Temporale, VDM, Z e B *Formal Methods including for example CCS, CSP, HOL, LOTOS, OBJ, Temporal Logic, VDM, Z and B*	B.30	—	R	R	HR	HR
2. Metodi Semi-Formali *Semi-Formal Methods*	D.7	R	R	R	HR	HR
3. Metodologia Strutturata comprendente per esempio JSD, MASCOT, SADT, SDL, SSADM, e Yourdon *Structured. Methodology including for example JSD, MASCOT, SADT, SDL, SSADM, and Yourdon.*	B.60	R	HR	HR	HR	HR

Requisiti
Requirements

1. La Specifica dei Requisiti del Software richiederà sempre una descrizione del problema in linguaggio naturale ed ogni necessaria notazione matematica riferita all'applicazione.
   The Software Requirements Specification will always require a description of the problem in natural language and any necessary mathematical notation that reflects the application.
2. La tabella riferisce requisiti addizionali per la definizione della specifica in modo chiaro e preciso. Una o più di queste tecniche devono essere scelte per soddisfare il Livello di Integrità di Sicurezza del Software da usare.
   The table reflects additional requirements for defining the specification clearly and precisely. One or more of these techniques shall be selected to satisfy the Software Safety Integrity Level being used.

Tab. A.3 **Software Architecture (clause 9)** **Architettura del Software (art. 9)**

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Programmazione Difensiva *Defensive Programming*	B.15	—	R	R	HR	HR
2. Rilevamento guasto & Diagnosi *Fault Detection & Diagnosis*	B.27	—	R	R	HR	HR
3. Codici di Correzione Errore *Error Correcting Codes*	B.20	—	—	—	—	—
4. Codici Rilevamento Errore *Error Detecting Codes*	B.20	—	R	R	HR	HR
5. Programmazione di Asserzione delmancato funzionamento *Failure Assertion Programming*	B.25	—	R	R	HR	HR
6. Tecnica della valigetta di sicurezza (Safety bag) *Safety Bag Techniques*	B.54	—	R	R	R	R
7. Programmazione Diversificata *Diverse Programming*	B.17	—	R	R	HR	HR
8. Blocco di Recupero (Recovery Block) *Recovery Block*	B.50	—	R	R	R	R
9. Recupero all'indietro (Backward Recovery) *Backward Recovery*	B.5	—	NR	NR	NR	NR
10. Recupero in avanti (Forward Recovery) *Forward Recovery*	B.32	—	NR	NR	NR	NR
11. Meccanismo di ripetizione per il Recupero dal Guasto *Re-try Fault Recovery Mechanisms*	B.53	—	R	R	R	R
12. Memorizzazione dei Casi Eseguiti *Memorising Executed Cases*	B.39	—	R	R	HR	HR
13. Intelligenza Artificiale- Correzione del guasto *Artificial Intelligence - Fault Correction*	B.1	—	NR	NR	NR	NR
14. Riconfigurazione dinamica del software *Dynamic Reconfiguration of software*	B.18	—	NR	NR	NR	NR
15. Analisi dell'Effetto dell'Errore Software *Software Error Effect Analysis*	B.26	—	R	R	HR	HR
16. Analisi dell'Albero dei Guasti *Fault Tree Analysis*	B.28	R	R	R	HR	HR

Requisiti
Requirements

1. Le combinazioni consentite delle tecniche per i Livelli 3 e 4 di Integrità di Sicurezza del Software devono essere le seguenti:
   Approved combinations of techniques for Software Safety Integrity Levels 3 and 4 shall be as follows:
   a 1, 7 e una da 4, 5 o 12
   1, 7 and one from 4, 5 or 12
   b 1, 4 e 5
   1, 4 and 5
   c 1, 4 e 12
   1, 4 and 12
   d 1, 2 e 4
   1, 2 and 4
   e 1 e 4, e una tra 15 e 16
   1 and 4, and one of 15 and 16
2. Con l'eccezione delle voci 3, 9, 10, 13 e 14, una o più di queste tecniche devono essere scelte per soddisfare i requisiti per i Livelli di Integrità di Sicurezza del Software 1 e 2.
   With the exception of entries 3, 9, 10, 13 and 14, one or more of these techniques shall be selected to satisfy the requirements for Software Safety Integrity Levels 1 and 2.
3. Alcune di queste indicazioni possono essere definite a livello di sistema.
   Some of these issues may be defined at the system level.
4. I codici di correzione dell'errore possono essere usati in accordo con i requisiti della EN 50159-1 ed EN 50159-2.
   Error correcting codes may be used in accordance with the requirements of EN 50159-1 and EN 50159-2.

Tab. A.4 **Software Design and Implementation (clause 10)** **Progettazione del Software ed Implementazione (art. 10)**

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Metodi Formali comprendenti ad esempio CCS, CSP, HOL, LOTOS, OBJ, Logica Temporale, VDM, Z e B *Formal Methods including for example CCS, CSP, HOL, LOTOS, OBJ, Temporal Logic, VDM, Z and B*	B.30	—	R	R	HR	HR
2. Metodi Semi Formali *Semi-FormaI Methods*	D.7	R	HR	HR	HR	HR
3. Metodologia Strutturata comprendente ad esempio JSD, MASCOT, SADT, SDL, SSADM e Yourdon. *Structured. Methodology including for example JSD, MASCOT, SADT, SDL, SSADM and Yourdon.*	B.60	R	HR	HR	HR	HR
4. Approccio Modulare *Modular Approach*	D.9	HR	M	M	M	M
5. Standard di Progettazione di Codifica *Design and Coding Standards*	D.1	HR	HR	HR	M	M
6. Programmi Analizzabili *Analysable Programs*	B.2	HR	HR	HR	HR	HR
7. Linguaggio di Programmazione Fortemente Tipizzato *Strongly Typed Programming Language*	B.57	R	HR	HR	HR	HR
8. Programmazione Strutturata *Structured Programming*	B.61	R	HR	HR	HR	HR
9. Linguaggio di Programmazione *Programming Language*	D.4	R	HR	HR	HR	HR
10. Sottoinsieme del Linguaggio *Language Subset*	B.38	—	—	—	HR	HR
11. Traduttore Validato *Validated Translator*	B.7	R	HR	HR	HR	HR
12. Traduttore Provato con l'Uso *Translator Proven in Use*	B.65	HR	HR	HR	HR	HR
13. Libreria di Moduli e Componenti Affidabili/Verificati *Library of Trusted/Verified Modules and Components*	B.40	R	R	R	R	R
14. Prove Funzionali/ a Scatola Chiusa (Black-box) *Functional/ Black-box Testing*	D.3	HR	HR	HR	M	M
15. Prove di Prestazione *Performance Testing*	D.6	—	HR	HR	HR	HR
16. Prove sulle Interfacce *Interface Testing*	B.37	HR	HR	HR	HR	HR
17. Registrazione e Analisi dei Dati *Data Recording and Analysis*	B.13	HR	HR	HR	M	M
18. Logica Fuzzy *Fuzzy Logic*	B.67	—	—	—	—	—
19. Programmazione Orientata all'Oggetto *Object Oriented Programming*	B.68	—	R	R	R	R

Requisiti
Requirements

1. Deve essere scelto un'insieme di tecniche secondo il livello di integrità di sicurezza del software.
   A suitable set of techniques shall be chosen according to the software safety integrity level.
2. Per il livello di integrità di sicurezza del software 3 o 4, l'insieme consentito di tecniche deve comprendere una delle tecniche 1, 2 o 3, insieme ad una delle tecniche 11 o 12. Le restanti tecniche devono essere trattate ancora secondo le rispettive raccomandazioni.
   At software safety integrity level 3 or 4, the approved set of techniques shall include one of techniques 1, 2 or 3, together with one of techniques 11 or 12. The remaining techniques shall still be treated according to their recommendations.

Tab. A.5 **Verification and Testing (clause 11)** **Verifica e Prove (art. 11)**

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Prova Formale *Formal Proof*	B.31	—	R	R	HR	HR
2. Prove Probabilistiche *Probabilistic Testing*	B.47	—	R	R	HR	HR
3. Analisi Statistica *Static Analysis*	D.8	—	HR	HR	HR	HR
4. Analisi Dinamica e Prove *Dynamic Analysis and Testing*	D.2	—	HR	HR	HR	HR
5. Metriche *Metrics*	B.42	—	R	R	R	R
6. Matrice di Tracciabilità *Traceability Matrix*	B.69	—	R	R	HR	HR
7. Analisi dell'Effetto dell'Errore Software *Software Error Effect Analysis*	B26	—	R	R	HR	HR

Requisiti
Requirements

1. Per il Livello di Integrità di Sicurezza del Software 3 o 4, le combinazioni di Tecniche consentite devono essere:
   For Software Safety Integrity Level 3 or 4, the approved combinations of techniques shall be:
   a 1 e 4
   1 and 4
   b 3 e 4
   3 and 4
   o c) 4, 6 e 7
   or 4, 6 and 7
2. Per il Livello di Integrità di Sicurezza del Software 1 o 2 le Tecniche consentite devono essere 1 o 4.
   For Software Safety Integrity Level 1 or 2, the approved technique shall be 1 or 4.

Tab. A.6 **Software/Hardware Integration (clause 12)** **Integrazione Software/Hardware (art. 12)**

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Prove funzionali ed a Scatola Chiusa (Black-box) *Functional and Black-box Testing*	D.3	HR	HR	HR	HR	HR
2. Prove di Prestazione *Performance Testing*	D.6	—	R	R	HR	HR

Requisiti
Requirements

1. Per il livello di integrità di sicurezza del software 0 , sarà consentita la tecnica 1.
   For software safety integrity level 0, technique 1 shall be the approved technique.
2. Per il livello di integrità di sicurezza del software 1, 2, 3 o 4, sarà consentita la combinazione di tecniche 1 e 2.
   For software safety integrity level 1, 2, 3 or 4, the approved combination of techniques shall be 1 and 2.

Tab. A.7 **Software Validation (clause 13)** — **Validazione del Software (art. 13)**

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Prove Probabilistiche *Probabilistic Testing*	B.47	—	R	R	HR	HR
2. Prove di Prestazione *Performance Testing*	D.6	—	HR	HR	M	M
3. Prove Funzionali ed a Scatola Chiusa (Black-box) *Functional and Black-box Testing*	D.3	HR	HR	HR	M	M
4. Modellazione *Modelling*	D.5	—	R	R	R	R

Prescrizioni
Requirement

1. Per il Livello di Integrità di Sicurezza del Software 1, 2, 3 o 4, una combinazione di Tecniche consentite sarà 2 e 3.
   For Software Safety Integrity Level 1, 2, 3 or 4, an approved combination of techniques shall be 2 and 3.

Tab. A.8 **Clauses to be assessed** — **Articoli da valutare**

ARTICOLI DA VALUTARE *CLAUSE TO BE ASSESSED*	Articolo *Clause*	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Livelli di Integrità di Sicurezza S/W *S/W Safety Integrity Levels*	5	HR	HR	HR	HR	HR
2. Personale & Responsabilità *Personnel & Responsibility*	6	—	R	R	HR	HR
3. Ciclo di vita & Documentazione *Lifecycle & Documentation*	7	—	HR	HR	HR	HR
4. Spec. Requisiti S/W *S/W Requirements Spec.*	8	R	HR	HR	HR	HR
5. Architettura S/W *S/W Architecture*	9	—	R	R	HR	HR
6. Progettazione & Sviluppo *Design & Development*	10	—	R	R	HR	HR
7. Verifica *Verification*	11	—	HR	HR	HR	HR
8. Integrazione S/W/H/W *S/W/H/W Integration*	12	—	R	R	HR	HR
9. Validazione S/W *S/W Validation*	13	—	HR	HR	HR	HR
10. Assicurazione Qualità *Quality Assurance*	15	—	HR	HR	HR	HR
11. Manutenzione *Maintenance*	16	—	HR	HR	HR	HR

Tab. A.9 **Software Assessment (clause 14) Assessment Techniques**

Valutazione Software (art. 14) Tecniche di Valutazione

ASSESSMENT TECHNIQUE *ASSESSMENT TECHNIQUE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Liste di controllo *Checklists*	B.8	HR	HR	HR	HR	HR
2. Analisi Statica del Software *Static Software Analysis*	B.14 B.42 D.8	R	HR	HR	HR	HR
3. Analisi Dinamica del Software *Dynamic Software Analysis*	D.2 D.3	—	R	R	HR	HR
4. Diagrammi causa conseguenza *Cause Consequence Diagrams*	B.6	R	R	R	R	R
5. Analisi dell'Albero degli Eventi *Event Tree Analysis*	B.23	—	R	R	R	R
6. Analisi dell'Albero dei Guasti *Fault Tree Analysis*	B.28	R	R	R	HR	HR
7. Analisi dell'Effetto dell'Errore Software *Software Error Effect Analysis*	B.26	—	R	R	HR	HR
8. Analisi dei mancati funzionamenti di Causa Comune *Common Cause Failure Analysis*	B.10	—	R	R	HR	HR
9. Modelli di Markov *Markov Models*	B.41	—	R	R	R	R
10. Diagramma a Blocchi dell'Affidabilità *Reliability Block Diagram*	B.51	—	R	R	R	R
11. Prova sul Campo Prima della Messa in Servizio *Field Trial Before Commissioning*	—	R	HR	HR	HR	HR

Requisito
Requirement

1. Deve essere scelta una o più di queste tecniche per soddisfare il Livello di Integrità di Sicurezza del Software che deve essere usato.
One or more of these techniques shall be selected to satisfy the Software Safety Integrity Level being used.

Tab. A.10 **Software Quality Assurance (clause 15)**

Assicurazione della Qualità del Software (art. 15)

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Articolo o *Clause or* Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Accreditato alla EN ISO 9001 *Accredited to EN ISO 9001*	15	R	HR	HR	HR	HR
2. Conforme con la EN ISO 9000-3 *Compliant with EN ISO 9000-3*	15	M	M	M	M	M
3. Sistema Qualità della Società *Company Quality System*	15	M	M	M	M	M
4. Gestione della Configurazione del Software *Software Configuration Management*	B.56	M	M	M	M	M

Tab. A.11 **Software Maintenance (clause 16)**

Manutenzione del Software (art. 16)

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Analisi dell'Impatto *Impact Analysis*	B.35	R	HR	HR	M	M
2. Registrazione e Analisi dei Dati *Data Recording and Analysis*	B.13	HR	HR	HR	M	M

Tab. A.12 **Design and Coding Standards (D.1) Referenced by clause 10**

Standard di Progettazione e di Codifica (D.1) Riferimento all'art. 10

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Esiste lo Standard di Codifica *Coding Standard Exists*	B.16	HR	HR	HR	HR	HR
2. Guida allo stile di Codifica *Coding Style Guide*	B.16	HR	HR	HR	HR	HR
3. Nessun Oggetto Dinamico *No Dynamic Objects*	B.16	—	R	R	HR	HR
4. Nessuna Variabile Dinamica *No Dynamic Variables*	B.16	—	R	R	HR	HR
5. Uso Limitato di Puntatori *Limited Use of Pointers*	B.16	—	R	R	R	R
a) Uso Limitato di Ricorsioni *Limited Use of Recursion*	B.16	—	R	R	HR	HR
6. Nessun Salto Incondizionato *No Unconditional Jumps*	B.16	—	HR	HR	HR	HR

Requisito
Requirement

1. È accettato che le tecniche 3, 4 e 5 possano essere presenti come parte di un compilatore o traduttore validato.
   It is accepted that techniques 3, 4 and 5 may be present as part of a validated compiler or translator.
2. Deve essere scelto un'insieme adatto di tecniche secondo il livello di integrità di sicurezza del software.
   A suitable set of techniques shall be chosen according to the software safety integrity level.

Tab. A.13 **Dynamic Analysis and Testing (D.2) Referenced by clauses 11 and 14**

Analisi e Prova Dinamiche (D.2) Riferimento all'art. 11 e 14

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Esecuzione del Caso di Prova dall'Analisi del Valore limite *Test Case Execution from Boundary Value Analysis*	B.4	—	HR	HR	HR	HR
2. Esecuzione del Caso di Prova dalla Supposizione di Errori *Test Case Execution from Error Guessing*	B.21	R	R	R	R	R
3. Esecuzione del Caso di Prova dalla Semina di Errori *Test Case Execution from Error Seeding*	B.22	—	R	R	R	R
4. Modellazione delle Prestazioni *Performance Modelling*	B.45	—	R	R	HR	HR
5. Classi di Equivalenza e Prove di Partizione dei dati di Ingresso *Equivalence Classes and Input Partition Testing*	B.19	—	R	R	HR	HR
6. Prove Basate sulla Struttura *Structure-Based Testing*	B.58	—	R	R	HR	HR

Requisiti
Requirements

1. L'analisi per i casi di prova è a livello di sottosistema ed è basata sulla specifica e/o sulla specifica ed il codice.
   The analysis for the test cases is at the sub-system level and is based on the specification and/or the specification and the code.
2. Deve essere scelto un insieme adatto di tecniche secondo il livello di integrità di sicurezza del software.
   A suitable set of techniques shall be chosen according to the software safety integrity level.

Tab. A.14 **Functional/Black Box Test (D.3)**
Referenced by clauses 10,12, 13 and 14

Prove Funzionali/ a Scatola Chiusa (Black-box) (D.3)
Riferimento all'art. 10,12, 13 e14

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Esecuzione del Caso di Prova dai Diagrammi Causa Conseguenza *Test Case Execution from Cause Consequence Diagrams*	B.6	—	—	—	R	R
2. Prototipazione/Animazione *Prototyping/Animation*	B.49	—	—	—	R	R
3. Analisi del Valore limite *Boundary Value Analysis*	B.4	R	HR	HR	HR	HR
4. Classi di Equivalenza e Prova di Partizione dei dati di Ingresso *Equivalence Classes and Input Partition Testing*	B.19	R	HR	HR	HR	HR
5. Simulazione di Processo *Process Simulation*	B.48	R	R	R	R	R

Requisiti
Requirements

1. La completezza della simulazione dipenderà dall'ampiezza del livello di integrità di sicurezza del software, dalla complessità e dall'applicazione.
   The completeness of the simulation will depend upon the extent of the software safety integrity level, complexity and application.
2. Deve essere scelto un'insieme adatto di tecniche secondo il livello di integrità di sicurezza del software.
   A suitable set of techniques shall be chosen according to the software safety integrity level.

Tab. A.15 **Programming Languages (D.4)**
Referenced by clause 10

Linguaggi di Programmazione (D.4)
Riferimento all'art. 10

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. ADA	B.62	R	HR	HR	R	R
2. MODULA-2 *MODULA-2*	B.62	R	HR	HR	R	R
3. PASCAL	B.62	R	HR	HR	R	R
4. Fortran 77	B.62	R	R	R	R	R
5. "C" or C++ (senza restrizioni) *"C" or C++ (unrestricted)*	B.62	R	—	—	NR	NR
6. Sottoinsieme di C o C++ con standard di codifica *Subset of C or C++ with coding standards*	B.62 B.38	R	R	R	R	R
7. L/M	B.62	R	R	R	NR	NR
8. BASIC	B.62	R	NR	NR	NR	NR
9. Assembler	B.62	R	R	R	—	—
10. Diagrammi a Scala *Ladder Diagrams*	B.62	R	R	R	R	R
11. Blocchi Funzionali *Functional Blocks*	B.62	R	R	R	R	R
12. Lista delle Dichiarazioni *Statement List*	B.62	R	R	R	R	R

Requisiti
Requirements

1. Per il livello di Integrità di Sicurezza del Software 3 e 4 quando è usato un sottoinsieme dei linguaggi 1, 2, 3 e 4 la raccomandazione cambia a HR.
   At Software Safety Integrity Level 3 and 4 when a subset of languages 1, 2, 3 and 4 are used the recommendation changes to HR.
2. Per certe applicazioni i linguaggi 7 e 9 possono essere i soli disponibili. Per il livello di Integrità di Sicurezza del Software 3 e 4 se non è disponibile una opzione Altamente raccomandata è fortemente consigliato che ci sia un sottoinsieme di questi linguaggi ed un preciso insieme di standard di codifica per innalzare la raccomandazione ad 'R'.
   For certain applications the languages 7 and 9 may be the only ones available. At Software Safety Integrity Level 3 and 4 where a Highly recommended option is not available it is strongly recommended that to raise the recommendation to "R" there should be a subset of these languages and that there should be a precise set of coding standards.
3. Per informazioni sulla valutazione di idoneità di un linguaggio di programmazione vedere la voce nella bibliografia di "Linguaggi di Programmazione Adatti", B.62.
   For information on assessing the suitability of a programming language see entry in the bibliography for "Suitable Programming Language" B.62.
4. Se un linguaggio specifico non è nella tabella, non è automaticamente escluso. Esso dovrebbe essere comunque conforme a B.62.
   If a specific language is not in the table, it is not automatically excluded. It should, however, conform to B.62.

Tab. A.16 **Modelling (D.5)**
Referenced by clause 13

Modellazione (D.5)
Riferimento all'art. 13

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Diagrammi di Flusso dei Dati *Data Flow Diagrams*	B.12	—	R	R	R	R
2. Macchine a Stato Finito *Finite State Machines*	B.29	—	HR	HR	HR	HR
3. Metodi FormaIi *FormaI Methods*	B.30	—	R	R	HR	HR
4. Modellazione delle Prestazioni *Performance Modelling*	B.45	—	R	R	HR	HR
5. Reti di Petri temporali *Time Petri Nets*	B.64	—	HR	HR	HR	HR
6. Prototipazione/Animazione *Prototyping/Animation*	B.49	—	R	R	R	R
7. Diagrammi Strutturati *Structure Diagrams*	B.59	—	R	R	HR	HR

Requisiti
Requirement

1. Deve essere scelto un'insieme adatto di tecniche secondo il livello di integrità di sicurezza del software..
A suitable set of techniques shall be chosen according to the software safety integrity level.

Tab. A.17 **Performance Testing (D.6)**
Referenced by clauses 10, 12 and 13

Prove di Prestazione (D.6)
Riferimento all'art. 10, 12 e 13

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Prove di sovraccarico (Avalanche /Stress) *Avalanche/Stress Testing*	B.3	—	R	R	HR	HR
2. Tempi di risposta e vincoli di Memoria *Response Timing and Memory Constraints*	B.52	—	HR	HR	HR	HR
3. Requisiti di Prestazione *Performance Requirements*	B.46	—	HR	HR	HR	HR

Requisiti
Requirement

Deve essere scelto un' insieme adatto di tecniche secondo il livello di integrità di sicurezza del software.
A suitable set of techniques shall be chosen according to the software safety integrity level.

Tab. A.18 **Semi-Formal Methods (D.7)**
Referenced by clauses 8 and 10

Metodi Semi-Formali (D.7)
Riferimento all'art. 8 e 10

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Diagr. Dei Blocchi Logici /Funzionali *Logic/Function Block Diags*	—	R	R	R	HR	HR
2. Diagrammi di Sequenza *Sequence Diagrams*	—	R	R	R	HR	HR
3. Diagrammi di Flusso dei dati *Data flow Diagrams*	B.12	R	R	R	R	R
4. Macchine a Stati Finiti/ Diagrammi di Transizione dello stato *Finite State Machines/State Transition Diagrams*	B.29	—	R	R	HR	HR
5. Reti di Petri temporali *Time Petri Nets*	B.64	—	R	R	HR	HR
6. Tavole di Decisione /di Verità *Decision/Truth Tables*	B.14	R	R	R	HR	HR

Requisiti
Requirement

Deve essere scelto un' insieme adatto di tecniche secondo il livello di integrità di sicurezza del software.
A suitable set of techniques shall be chosen according to the software safety integrity level.

Tab. A.19 **Static Analysis (D.8)**
Referenced by clauses 11 and 14

Analisi Statica (D.8)
Riferimento all'art. 11 e 14

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Analisi del Valore limite *Boundary Value Analysis*	B.4	—	R	R	HR	HR
2. Liste di controllo *Checklists*	B.8	—	R	R	R	R
3. Analisi del Flusso di controllo *ControI Flow Analysis*	B.9	—	HR	HR	HR	HR
4. Analisi del Flusso dei Dati *Data Flow Analysis*	B.11	—	HR	HR	HR	HR
5. Supposizione di Errore *Error Guessing*	B.21	—	R	R	R	R
6. lspezioni di Fagan *Fagan lnspections*	B.24	—	R	R	HR	HR
7. Analisi di circuito ingannevole *Sneak Circuit Analysis*	B.55	—	—	—	R	R
8. Esecuzione Simbolica *Symbolic Execution*	B.63	—	R	R	HR	HR
9. Analisi di Percorso (Walkthroughs) / Riesame della progettazione *Walkthroughs/Design Reviews*	B.66	HR	HR	HR	HR	HR

Requisiti
Requirement

Deve essere scelto un' insieme adatto di Tecniche secondo il livello di integrità di sicurezza del software.
A suitable set of techniques shall be chosen according to the software safety integrity level.

Tab. A.20 **Modular Approach (D.9)**
Referenced by clause 10

Approccio– Modulare (D.9)
Riferimento all'art. 10

TECNICA/PROVVEDIMENTO *TECHNIQUE/MEASURE*	Ref	SWSIL0	SWSIL1	SWSIL2	SWSIL3	SWSIL4
1. Dimensione del Modulo Limitata *Module Size Limited*	B.43	HR	HR	HR	HR	HR
2. Occultamento/Incapsulamento delle informazioni *Information Hiding/Encapsulation*	B.36	R	HR	HR	HR	HR
3. Limite del Numero dei Parametri *Parameter Number Limit*	B.43	R	R	R	R	R
4. Punti ad Una Sola Entrata /Una Sola Uscita nelle Subroutine e nelle Funzioni *One Entry/One Exit Point in Subroutines and Functions*	B.43	R	HR	HR	HR	HR
5. Interfaccia Completamente definita *Fully Defined Interface*	B.43	HR	HR	HR	M	M

Requisiti
Requirement

Deve essere scelto un'insieme adatto di tecniche secondo il livello di integrità di sicurezza del software.
A suitable set of techniques shall be chosen according to the software safety integrity level.

ANNEX/ALLEGATO
B informative informativo

BIBLIOGRAPHY OF TECHNIQUES

BIBLIOGRAFIA DELLE TECNICHE

B.1 AI Fault Correction (Referenced by clause 9)

Aim

To be able to react to possible hazards in a very flexible way by introducing a mix (combination) of methods and process models and some kind of on-line safety and reliability analysis.

Description

In particular fault forecasting (calculating trends), fault correction, maintenance and supervisory actions may be supported by AI-based systems in a very efficient way in diverse channels of a system, since the rules might be derived directly from the specifications and checked against these. Certain common faults which are introduced into specifications by implicitly already having some design and implementation rules in mind may be avoided effectively by this approach, especially when applying a combination of models and methods in a functional or descriptive manner.

The methods are selected such that faults may be corrected and the effects of failures be minimised, in order to meet the desired safety and reliability.

Correzione del Guasto AI (Riferimento all'art. 9)

Scopo

Essere in grado di reagire a possibili pericoli in un modo molto flessibile introducendo un mix (combinazione) di metodi e modelli di processo ed alcuni tipi di analisi di sicurezza e affidabilità in linea.

Descrizione

In particolare la previsione guasti (calcolo degli andamenti), la correzione guasti, le azioni di manutenzione e di supervisione possono essere supportate dai sistemi basati su AI in un modo molto efficace in diversi canali di un sistema, poiché le regole possono essere derivate direttamente dalle specifiche e possono essere verificate in base a queste. Certi guasti comuni che sono introdotti nelle specifiche implicitamente avendo già in mente alcune regole di progetto e di implementazione, possono essere evitati in modo efficace con questo approccio, specialmente quando si applica una combinazione di modelli e metodi in un modo funzionale e descrittivo.

I metodi sono selezionati in modo tale che i guasti possono essere corretti ed essere minimizzati gli effetti delle avarie, in modo da ottenere la sicurezza e l'affidabilità desiderate.

B.2 Analysable Programs (Referenced by clause 10)

Aim

To design a program in a way that program analysis is easily feasible. The program behaviour must be testable completely on the basis of the analysis.

Description

The intention is to produce programs which are easy to analysis using static analysis methods. In order to achieve this, the rules of structured programming should be followed, for instance:

- the module control flow should be composed of structured constructs, that is sequences, iterations and selection.
- the modules should be small.
- the number of possible paths through a module is small.
- the individual program parts have to be designed so that they are decoupled as far as possible.
- the relation between the input and output parameters should be as simple as possible.
- complex calculations should not be used as the basis of branching and loop decisions.

Programmi Analizzabili (Riferimento all'art. 10)

Scopo

Progettare un programma in modo che sia facilmente eseguibile la sua analisi. Il comportamento del programma lo si deve poter provare in modo completo sulla base dell'analisi.

Descrizione

L'intenzione è di produrre programmi che siano facilmente analizzabili usando metodi di analisi statica. A tale fine, devono essere seguite le regole della programmazione strutturata, per esempio:

- il flusso del controllo del modulo dovrebbe essere composto da costrutti strutturarti, cioè sequenze, iterazioni e selezioni.
- i moduli dovrebbero essere piccoli.
- il numero dei possibili percorsi attraverso il modulo è piccolo.
- le parti individuali del programma devono essere progettate in modo che esse possano essere il più possibile disaccoppiate.
- le relazioni tra i parametri d'ingresso e di uscita dovrebbero essere le più semplici possibili.
- non dovrebbero essere usati calcoli complessi come base delle decisioni di diramazione e di costrutti iterativi

- branch and loop decisions should be simply related to the module input parameters.
- boundaries between different types of mappings shall be simple.

References:

Verification - The Practical Problems. B.J.T. Webb and D.J. Mannering, SARSS 87, Nov. 1987, Altrincham, England, Elsevier Applied Science, ISBN 1-85166-167-0, 1987.

An Experience in Design and Validation of Software for a Reactor Protection System.S. Bologna, E. de Agostino et. al., IFAC Workshop, SAFECOMP 1979, Stuttgart, 16-18 May1979, Pergamon Press 1979.

B.3 Avalanche/Stress Testing (Referenced by D.6)

Aim

To burden the test object with an exceptionally high workload in order to show that the test object would stand normal workloads easily.

Description

There are a variety of test conditions which can be applied for avalanche/stress testing. Some of these test conditions are listed below:

- if working in a polling mode then the test object gets much more input changes per time unit as under normal conditions;
- if working on demands then the number of demands per time unit to the test object is increased beyond normal conditions;
- if the size of a database plays an important role then it is increased beyond normal conditions;
- influential devices are tuned to their maximum speed or lowest speed respectively;
- for the extreme cases, all influential factors, as far as is possible, are put to the boundary conditions at the same time.

Under these test conditions the time behaviour of the test object can be evaluated. The influence of load changes can be observed. The correct dimension of internal buffers or dynamic variables, stacks etc can be checked.

- le decisioni di diramazione e dei costrutti iterativi dovrebbero essere semplicemente correlate con i parametri d'ingresso del modulo.
- i confini tra i diversi tipi di mappatura devono essere semplici.

Riferimenti:

Verification - The Practical Problems. B.J.T. Webb and D.J. Mannering, SARSS 87, Nov. 1987, Altrincham, England, Elsevier Applied Science, ISBN 1-85166-167-0, 1987.

An Experience in Design and Validation of Software for a Reactor Protection System.S. Bologna, E. de Agostino et. al., IFAC Workshop, SAFECOMP 1979, Stuttgart, 16-18 May1979, Pergamon Press 1979.

Prove di sovraccarico (Avalanche/Stress) (Riferimento a D.6)

Scopo

Gravare l'oggetto della prova con un carico di lavoro eccezionalmente elevato per dimostrare che l'oggetto della prova potrebbe sopportare facilmente carichi di lavoro normali.

Descrizione

Vi sono una varietà di condizioni di prova che possono essere applicate per le prove a valanga/da sforzo. Alcune di queste condizioni di prova sono elencate nel seguito:

- se si lavora in una modalità di interrogazione ciclica, l'oggetto della prova riceve molti più cambiamenti d'ingresso per unità di tempo che in condizioni normali;
- se si lavora su domanda, il numero delle domande per unità di tempo all'oggetto della prova è aumentato oltre le condizioni normali;
- se la dimensione della base dei dati gioca un ruolo importante, essa è aumentata oltre le condizioni normali;
- dispositivi influenti sono sintonizzati alla loro massima velocità o rispettivamente alla loro velocità più bassa;
- nei casi estremi, tutti i fattori d'influenza, per quanto possibile, sono posti nelle condizioni limite contemporaneamente.

In queste condizioni di prova può essere valutato il comportamento nel tempo dell'oggetto di prova. Può essere osservata l'influenza dei cambiamenti di carico. Possono essere verificate le corrette dimensioni dei buffer interni o delle variabili dinamiche, stack ecc.

B.4 Boundary Value Analysis (Referenced by D.2, D.3 and D.8)

Aim

To remove software errors occurring at parameter limits or boundaries.

Description

The input domain of the program is divided into a number of input classes. The tests should cover the boundaries and extremes of the classes. The tests check that the boundaries in the input domain of the specification coincide with those in the program. The use of the value zero, in a direct as well as in an indirect translation, is often error-prone and demands special attention:

- zero divisor;
- blank ASCII characters;
- empty stack or list element;
- null matrix;
- zero table entry.

Normally the boundaries for input have a direct correspondence to the boundaries for the output range. Test cases should be written to force the output to its limited values. Consider also, if it is possible to specify a test case which causes output to exceed the specification boundary values.

If output is a sequence of data, for example a printed table, special attention should be paid to the first and the last elements and to lists containing none, 1 and 2 elements.

References:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, USA, 1979.

B.4 Analisi del Valore limite (Riferimento a D.2, D.3 e D.8)

Scopo

Rimuovere gli errori del software che si verificano attraverso parametri ai confini o ai limiti.

Descrizione

Il dominio d'ingresso del programma è diviso in una serie di classi d'ingresso. Le prove dovrebbero coprire i limiti e gli estremi delle classi. Le prove verificano che i limiti nel dominio d'ingresso della specifica coincidano con quelli nel programma. L'uso del valore zero, in una traduzione sia diretta che indiretta è spesso incline all'errore e richiede una attenzione particolare:

- divisore zero;
- caratteri ASCII di spazio;
- stack vuoto o elemento di elenco;
- matrice nulla;
- inserimento di tabella zero.

Normalmente i confini per l'ingresso hanno una corrispondenza diretta con i confini del campo di uscita. I casi di prova dovrebbero essere scritti per forzare l'uscita ai suoi valori limite. Va considerato anche, se è possibile, specificare un caso di prova che preveda che l'uscita superi i valori di confine specificati.

Se l'uscita è una sequenza di dati, per esempio una tabella stampata, deve essere prestata un'attenzione particolare al primo e all'ultimo elemento e ad elenchi che contengono elementi nulli, 1 e 2.

Riferimenti:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, USA, 1979.

B.5 Backward Recovery (Referenced by clause 9)

Aim

To provide correct functional operation in the presence of one or more faults.

Description

If a fault has been detected, the system is reset to an earlier internal state, the consistency of which has been proven before. This method implies saving of the internal state frequently at so-called well defined checkpoints. This may be done globally (for the complete database) or incremental (changes only between checkpoints). Then the system has to compensate for the changes which have taken place in the meantime by using journalling (audit trail of actions), compensation (all effects of these changes are nullified) or external (manual)interaction.

B.5 Recupero all'indietro (Backward Recovery) (Riferimento all'art. 9)

Scopo

Fornire operazioni funzionali corrette in presenza di uno o più guasti.

Descrizione

Se è stato rilevato un guasto, il sistema è ripristinato ad uno stato interno precedente, la coerenza del quale è stata prima provata. Questo metodo implica che sia salvato lo stato interno in modo frequente in cosiddetti punti di controllo ben definiti. Questo può essere fatto in modo globale (per la completa base dei dati) o in modo incrementale (cambiamenti solo tra punti di controllo). Poi il sistema deve compensare i cambiamenti che hanno avuto luogo nel frattempo usando registrazioni (percorsi ispettivi delle azioni), compensazione (tutti gli effetti di questi cambiamenti sono annullati) o interazione esterna (manuale).

B.6 Cause Consequence Diagrams (Referenced by clause 14 and D.3)

Aim

To model, in a diagrammatic form, the sequence of events that can develop in a system as a consequence of combinations of basic events.

Description

It can be regarded as a combination of fault-tree and event-tree analysis. Starting from a critical event, a cause-consequence graph is traced backwards and forwards. In the backwards direction it is equivalent to a fault tree with the critical event as the given top event. In the forward direction the possible consequences arising from an event are identified. The graph can contain vertex symbols which describe the conditions for propagation along different branches from the vertex. Time delays can also be included. These conditions can also be described with fault trees. The lines of propagation can be combined with logical symbols, to make the diagram more compact. A set of standard symbols are defined for use in cause consequence diagrams. The diagrams can be used to compute the probability of occurrence of certain critical consequences.

References:

The Cause Consequence Diagram Method as a Basis for Quantitative Accident Analysis.D.S. Nielsen, Riso-M-1374, 1971.

Diagrammi Causa Conseguenza (Riferimento all'art. 14 e a D.3)

Scopo

Modellare, in forma di diagramma, le sequenze degli eventi che si possono sviluppare in un sistema come conseguenza della combinazione di eventi di base.

Descrizione

Può essere considerata come una combinazione dell'analisi dell'albero dei guasti e dell'albero degli eventi. Partendo da un evento critico, viene tracciato a ritroso ed in avanti un grafico causa-conseguenza. Nella direzione a ritroso è equivalente ad un albero dei guasti con l'evento critico come top event. Nella direzione in avanti sono identificate le possibili conseguenze che hanno origine da un evento. Il grafico può contenere simboli di vertice che descrivono le condizioni per la propagazione lungo i diversi rami dal vertice. Possono essere anche inseriti ritardi di tempo. Queste condizioni possono essere anche descritte con alberi dei guasti. Le linee di propagazione possono essere combinate con simboli logici, per rendere il diagramma più compatto. Per l'impiego in diagrammi causa conseguenza sono definiti una serie di simboli normalizzati. I diagrammi possono essere usati per calcolare la probabilità di accadimento di certe conseguenze critiche.

Riferimenti:

The Cause Consequence Diagram Method as a Basis for Quantitative Accident Analysis.D.S. Nielsen, Riso-M-1374, 1971.

B.7 Certified Tools and Certified Translators (Referenced by clause 10)

Aim

Tools are necessary to help developers in the different phases of software development. Wherever possible tools should be certified so that some level of confidence can be assumed regarding the correctness of the outputs.

Description

A certified tool is one which has been determined to be of a particular quality. The certification of a tool will generally be carried out by an independent, often national, body, against independently set criteria, typically national or international standards. Ideally, the tools used in all development phases (definition, design, coding, testing and validation) and those used in configuration management, should be subject to certification. To date only compilers (translators) are regularly subject to certification procedures; these are laid down by national certifica-

Strumenti e Traduttori Certificati (Riferimento all'art. 10)

Scopo

Strumenti sono necessari per aiutare i progettisti, nelle diverse fasi dello sviluppo del software. Ove possibile gli strumenti dovrebbero essere certificati in modo che si possa assumere un certo livello di fiducia circa la correttezza delle uscite.

Descrizione

Uno strumento è certificato quando è stato stabilito che è di particolare qualità. La certificazione di uno strumento viene generalmente eseguita da un ente indipendente, spesso nazionale, sulla base di una serie di criteri indipendenti, tipicamente da norme nazionali o internazionali. Idealmente, gli strumenti usati nelle fasi di sviluppo (definizione, progettazione, codifica, prova e validazione) e quelli usati nella gestione della configurazione, dovrebbero essere soggetti a certificazione. Attualmente solo i compilatori (traduttori) sono regolarmente soggetti a procedure di certificazione; queste sono elaborate da enti

tion bodies and they exercise compilers (translators) against international standards. such as those for Ada and Pascal.

References:

Pascal Validation Suite. UK distributor: BSI Quality Assurance, PO Box 375, Milton Keynes, MK14 6LL.

Ada Validation Suite. UK distributor: National Computing Centre (NCC), Oxford Road., Manchester England.

di certificazione nazionale che provano i compilatori (traduttori) in base a norme internazionali come quelle per Ada e Pascal.

Riferimenti:

Pascal Validation Suite. UK distributor: BSI Quality Assurance, PO Box 375, Milton Keynes, MK14 6LL.

Ada Validation Suite. UK distributor: National Computing Centre (NCC), Oxford Road., Manchester England.

B.8 Checklists (Referenced by clause 14 and D.8)

Aim

To provide a stimulus to critical appraisal of all aspects of the system rather than to lay down specific requirements.

Description

A set of questions to be completed by the person performing the checklist. Many of the questions are of a general nature and the Assessor must interpret them as seems most appropriate to the particular system being assessed.

To accommodate wide variations in software and systems being validated, most checklists contain questions which are applicable to many types of system. As a result there may well be questions in the checklist being used which are not relevant to the system being dealt with and which should be ignored. Equally there may be a need, for a particular system, to supplement the standard checklist with questions specifically directed at the system being dealt with.

In any case it should be clear that the use of checklists depends critically on the expertise and judgement of the engineer selecting and applying the checklist. As a result the decisions taken by the engineer, with regard to the checklist(s) selected, and any additional or superfluous questions, should be fully documented and justified. The objective is to ensure that the application of the checklists can be reviewed and that the same results will be achieved unless different criteria are used.

The object in completing a checklist is to be as concise as possible. When extensive justification is necessary this should be done by reference to additional documentation. Pass, Fail and Inconclusive, or some similar restricted set of tokens should be used to record the results for each question. This conciseness greatly simplifies the process of reaching an overall conclusion as to the results of the checklist assessment.

Liste di controllo (Riferimento all'art. 14 e a D.8)

Scopo

Fornire uno stimolo alla valutazione critica di tutti gli aspetti di un sistema anziché elaborare prescrizioni specifiche.

Descrizione

Una serie di domande che devono essere completate dalla persona che affronta la lista di controllo. Molte domande sono di natura generale ed il Valutatore deve interpretarle come appare più adatto allo specifico sistema che deve essere valutato.

Per conciliare le ampie variazioni nel software e nei sistemi da validare, la maggior parte delle liste di controllo contengono domande che sono applicabili a molti tipi di sistemi. Come risultato, nella lista di controllo che deve essere usata, vi possono essere domande che non sono pertinenti con il sistema di cui si tratta e che dovrebbero essere ignorate. Per contro può esservi una necessità, per un particolare sistema, di integrare la lista di controllo standard con domande specificamente orientate al sistema che deve essere considerato.

In ogni caso dovrebbe essere evidente che l'uso di liste di controllo dipende in modo critico dall'esperienza e dal giudizio dell'ingegnere che seleziona ed applica la lista. Ne deriva che le decisioni prese dall'ingegnere, nei confronti della/e lista/e di verifica scelta/e, e le domande aggiunte o superflue, dovrebbero essere interamente documentate e giustificate. L'obbiettivo è quello di assicurare che l'applicazione delle liste di controllo possa essere riesaminata e che possano essere raggiunti gli stessi risultati salvo che siano usati criteri diversi.

Nel completare una lista di controllo l'obbiettivo deve essere quello di essere i più concisi possibile. Quando è necessaria un' ampia giustificazione questa dovrebbe esser data facendo riferimento a documentazione aggiuntiva. Valido, Fallito e Non Conclusivo, o una serie ristretta di indicazioni simili dovrebbe essere usata per registrare i risultati di ogni domanda. Questa sinteticità semplifica notevolmente il processo per ottenere una conclusione generale riguardo ai risultati della valutazione della lista di controllo.

References:

Programmable Electronic Systems in Safety Related Applications. Health and Safety Executive, Her Majesty's Stationary Office, London 1987.

Guidelines for the Assessment of the Safety and Reliability of High Integrity Industrial Computer Systems. EWICS TC7, WP 489, 6th October 1987.

The Art of Software Testing. G. Myers, Wiley & Sons, New York, USA 1979.

Software for computers in the safety systems of nuclear power stations. IEC 60880, 1986.

Riferimenti:

Programmable Electronic Systems in Safety Related Applications. Health and Safety Executive, Her Majesty's Stationary Office, London 1987.

Guidelines for the Assessment of the Safety and Reliability of High Integrity Industrial Computer Systems. EWICS TC7, WP 489, 6th October 1987.

The Art of Software Testing. G. Myers, Wiley & Sons, New York, USA 1979.

Software for computers in the safety systems of nuclear power stations. IEC 60880, 1986.

B.9 Control Flow Analysis (Referenced by D.8)

Aim

To detect poor and potentially incorrect program structures.

Description

Control Flow Analysis identifies suspect areas of code which do not follow good programming practice. The program is analysed to form a directed graph which can be analysed for:

- inaccessible code, for instance, unconditional jumps which leaves blocks of code unreachable;
- knotted code, which is well structured code whose control graph is reducible by successive graph reductions to a single node. Poorly structured code can only be reduced to a knot composed of several nodes.

References:

RXVP80 - The Verification and Validation System for FORTRAN. Users Manual. General Research Corporation, Santa Barbara, California, USA.

Information Flow and Data Flow of While Programs. J.F. Bergeretti and B.A. Carre, ACMTrans. on Prog. Lang. and Syst., 1985.

Analisi del Flusso di Controllo (Riferimento a D.8)

Scopo

Rilevare strutture di programma scadenti e potenzialmente non corrette.

Descrizione

L'Analisi del Flusso di Controllo identifica aree sospette del codice che non seguono una buona pratica di programmazione. Il programma è analizzato per formare un grafico orientato che può essere indirizzato per:

- codice inaccessibile, per esempio, salti incondizionati che rendono irraggiungibili i blocchi di codice;
- codice di nodi, che è un codice ben strutturato il cui controllo grafico è riducibile per successive riduzioni del grafico ad un singolo nodo. Un codice poco strutturato può essere ridotto solamente ad un nodo composto di più nodi.

Riferimenti:

RXVP80 - The Verification and Validation System for FORTRAN. Users Manual. General Research Corporation, Santa Barbara, California, USA.

Information Flow and Data Flow of While Programs. J.F. Bergeretti and B.A. Carre, ACMTrans. on Prog. Lang. and Syst., 1985.

B.10 Common Cause Failure Analysis (Referenced by clause 14)

Aim

To identify potential failures in redundant systems or redundant sub-systems which would undermine the benefits of redundancy because of the appearance of the same failures in the redundant parts at the same time.

Description

Computer systems intended to take care of the safety of a plant often use redundancy in hardware and majority voting. This technique is used to avoid random component failures, which would tend to prevent the correct processing of

Analisi dei mancati funzionamenti di Causa Comune (Riferimento all'art.14)

Scopo

Identificare mancati funzionamenti potenziali in sistemi o sottosistemi ridondanti che indebolirebbero i benefici della ridondanza a seguito del presentarsi, nello stesso tempo, delle medesime avarie nelle parti ridondanti.

Descrizione

Sistemi computerizzati progettati per trattare la sicurezza di un impianto, usano spesso la ridondanza dell'hardware con votazione di maggioranza. Questa tecnica è usata per evitare mancati funzionamenti casuali di componenti, che tenderebbero ad impe-

data in a computer system.

However, some failures can be common to more than one component. For example, if a computer system is installed in one single room, shortcomings in the air-conditioning, might reduce the benefits of redundancy. The same is true for other external effects on the computer system such as: fire, flooding, electromagnetic interference, plane crashes, and earthquakes. The computer system may also be affected by incidents related to operation and maintenance. It is essential, therefore, that adequate and well documented procedures are provided for operation and maintenance. Extensive training of operating and maintenance personnel is also essential.

Internal effects are also major contributors to Common-Cause Failures (CCF). They can stem from design errors in common or identical components and their interfaces, as well as ageing of components. CCF-Analysis has to search the system for such potential common failures. Methods of CCF-Analysis are general quality control, design reviews, verification and testing by an independent team, and analysis of real incidents with feedback of experience from similar systems. The scope of the analysis, however, goes beyond hardware. Even if "diverse software" is used in difficult chains of a redundant computer system, there might be some commonality in the software approaches which could give rise to CCF. Errors in the common specification, for example.

When CCF's do not occur exactly at the same tine, precautions can be taken by means of comparison methods between the redundant chains which should lead to detection of a failure before this failure is common to all chains. CCF analysis should take this technique into account.

References:

Review of Common Cause Failures. I.A. Watson, UKAEA, Centre for Systems Reliability, Wigshaw Lane, WA3 4NE, England, NCSR R 27, July 1981.

Common-Mode Failures in Redundancy Systems. I.A. Watson and G.T. Edwards Nuclear Technology Vol, 46, De. 1979.

Programmable Electronic Systems in Safety Related Applications. Health and Safety Executive, Her Majesty's Stationary Office.

dire di processare correttamente i dati in un sistema computerizzato.

Tuttavia, alcuni mancati funzionamenti possono essere comuni a più di un componente. Per esempio, se un sistema computerizzato è installato in un'unica stanza, una deficienza dell'impianto di condizionamento, potrebbe ridurre il beneficio della ridondanza. Lo stesso vale per altri effetti esterni sul sistema computerizzato come: fuoco, allagamento, interferenze elettromagnetiche, incidenti aerei e terremoti. Il sistema computerizzato può essere influenzato anche da incidenti relativi al funzionamento ed alla manutenzione. È essenziale, pertanto, che siano previste procedure ben documentate e adeguate per funzionamento e la manutenzione. È essenziale anche l'addestramento approfondito del personale addetto al funzionamento ed alla manutenzione.

Anche gli effetti interni sono tra i maggiori responsabili dei mancati funzionamenti di Causa Comune (CCF). Essi possono derivare da errori di progetto in componenti comuni o identici e nelle loro interfacce, così come l'invecchiamento dei componenti. L'analisi CCF deve ricercare nei sistemi tali avarie comuni potenziali. I metodi dell'Analisi CCF sono il controllo della qualità generale, la revisione del progetto, la verifica e le prove da parte di un gruppo di lavoro indipendente, e l'analisi di incidenti reali con il riscontro dell'esperienza in sistemi simili. Lo scopo dell'analisi, tuttavia, va oltre l'hardware. Anche se viene usato "software diversificato" in catene difficili di sistemi computerizzati ridondanti, vi può essere qualche comunanza negli approcci software che potrebbe accrescere le CCF. Errori nelle specifiche comuni, ad esempio.

Quando le CCF non si verificano esattamente nel medesimo periodo di tempo, possono essere prese precauzioni mediante metodi comparativi tra le catene ridondanti che condurrebbero al rilievo di un mancato funzionamento prima che tale mancato funzionamento sia comune a tutte le catene. L'Analisi CCF dovrebbe tenere conto di tale tecnica.

Riferimenti:

Review of Common Cause Failures. I.A. Watson, UKAEA, Centre for Systems Reliability, Wigshaw Lane, WA3 4NE, England, NCSR R 27, July 1981.

Common-Mode Failures in Redundancy Systems. I.A. Watson and G.T. Edwards Nuclear Technology Vol, 46, De. 1979.

Programmable Electronic Systems in Safety Related Applications. Health and Safety Executive, Her Majesty's Stationary Office.

B.11 Data Flow Analysis (Referenced by D.8)

Aim

To detect poor and potentially incorrect program structures.

Description

Data Flow Analysis combines the information obtained from the control flow analysis with information about which variables are read or written in different portions of code. The analysis can check for:

- variables that are read before they are written. This is very likely to be an error, and is certainly bad programming practice;
- variables that are written more than once without being read. This could indicate omitted code;
- variables that are written but never read.

This could indicate redundant code. There is an extension of data flow analysis known as information flow analysis, where the actual data flows (both within and between procedures) are compared with the design intent. This is normally implemented with a computerised tool where the intended data flows are defined using a structured comment that can be read by the tool.

References:

RXVP80 - The Verification and Validation System for FORTRAN. Users Manual. General Research Corporation, Santa Barbara, California, USA.

Information Flow and Data Flow of While Programs. J.F. Bergeretti and B.A. Carre, ACMTrans. on Prog. Lang. and Syst., 1985.

Analisi del Flusso dei Dati (Riferimento a D.8)

Scopo

Rilevare strutture di programma insufficienti e potenzialmente non corrette.

Descrizione

L'Analisi del Flusso dei Dati combina le informazioni ottenute dall'analisi del flusso di controllo con informazioni che indicano quali variabili sono lette o scritte in diverse porzioni del codice. Le analisi possono verificare:

- variabili che sono lette prima di essere scritte. Questo molto probabilmente è un errore, ed è certamente una cattiva pratica di programmazione;
- variabili che sono scritte più di una volta senza essere lette. Questo potrebbe indicare omissione di codice;
- variabili che sono scritte ma mai lette. Questo potrebbe indicare ridondanza di codice.

Vi è un'estensione dell'analisi del flusso dei dati conosciuta come analisi del flusso di informazioni, dove i flussi di dati reali (sia entro che tra le procedure) sono confrontati con la progettazione determinata. Questo è normalmente implementato con uno strumento computerizzato ove i flussi dei dati determinati sono definiti con un commento strutturato che può essere letto dallo strumento.

Riferimenti:

RXVP80 - The Verification and Validation System for FORTRAN. Users Manual. General Research Corporation, Santa Barbara, California, USA.

Information Flow and Data Flow of While Programs. J.F. Bergeretti and B.A. Carre, ACMTrans. on Prog. Lang. and Syst., 1985.

B.12 Data Flow Diagrams (Referenced by D.5 and D.7)

Aim

To describe the data flow through a program in a diagrammatic form.

Description

Data Flow Diagrams document how data input is transformed to output, with each stage in the diagram representing a distinct transformation.

Data flow diagrams are made up of three components:

1. annotated arrows - represent data flow in and out of the transformation centres with the annotations specifying what the data is;

Diagrammi di Flusso dei Dati (Riferimento a D.5 e D.7)

Scopo

Descrivere il flusso dei dati attraverso un programma in forma di diagramma.

Descrizione

I Diagrammi di Flusso dei Dati documentano come l'ingresso di dati è trasformato in uscita, con ogni fase nel diagramma che rappresenta una distinta trasformazione.

I diagrammi del flusso dei dati sono costituiti da tre componenti:

1. frecce annotate - rappresentano il flusso dei dati in entrata e in uscita dai centri di trasformazione con le annotazioni che specificano ciò che il dato è;

2. annotated bubbles - represent transformation centres with the annotation specifying the transformation;
3. operators (AND, XOR) - These operators are used to link the annotated arrows.

Data flow diagrams describe how an input is transformed to an output. They do not, and should not, include control information or sequencing information. Each bubble can be considered as a stand alone black box which, as soon as its inputs are available, transforms them to its outputs.

One of the principle advantages of data flow diagrams is that they show transformations without making any assumptions about how these transformations are implemented.

The preparation of data flow diagrams is best approached by considering system inputs and working towards system outputs. Each bubble must represent a distinct transformation - its output should, in some way, be different from its input. There are no rules for determining the overall structure of the diagram and constructing a data flow diagram is one of the creative aspects of system design. Like all design, it is an iterative process with early attempts refined in stages to produce the final diagram.

References:

Software Engineering. I. Sommerville, Addison-Wesley 1982. ISBN 0-201-13795-X.

B.13 Data Recording and Analysis (Referenced by clauses 10 and 16)

Aim

To record all data, decisions and rationale in the software project to allow for easier verification, validation, assessment and maintenance.

Description

Detailed records are maintained during a project, both on a project and individual basis. For instance, an engineer would be required to keep records which could include

- effort expanded on individual modules,
- testing performed on each module,
- decisions and their rationale,
- achievement of project milestones,
- problems and their solutions.

During and at the conclusion of the project these records can be analysed to establish a wide variety of information. In particular data recording is very important for the maintenance of computer systems as the rationale for certain

2. pallogrammi annotati – rappresentano centri di trasformazione con le annotazioni che specificano la trasformazione;
3. operatori (AND, XOR) – Questi operatori sono usati per collegare le frecce annotate.

I diagrammi di flusso dei dati descrivono come un ingresso è trasformato in uscita. Essi non comprendono, e non dovrebbero comprendere, informazioni di controllo o informazioni di sequenza. Ogni pallogramma può essere considerato come una scatola nera (black box) indipendente che, non appena i suoi ingresso sono disponibili, li trasforma nelle sue uscite.

Uno dei principali vantaggi dei diagrammi di flusso dei dati è quello di mostrare le trasformazioni senza fare alcuna assunzione circa il modo in cui le trasformazioni sono implementate.

La preparazione dei diagrammi del flusso di dati è affrontata meglio considerando gli ingressi del sistema e arrivando verso le uscite del sistema. Ogni pallogramma deve rappresentare una trasformazione distinta – la sua uscita dovrebbe, in qualche modo, essere diversa dal suo ingresso. Non vi sono regole per determinare la struttura generale del diagramma e la costruzione di un diagramma di flusso dei dati è uno degli aspetti creativi della progettazione del sistema. Come tutta la progettazione, è un processo iterativo con tentativi iniziali raffinati per fasi per produrre il diagramma finale.

Riferimenti:

Software Engineering. I. Sommerville, Addison-Wesley 1982. ISBN 0-201-13795-X.

Registrazione e Analisi dei Dati (Riferimento all'art. 10 e 16)

Scopo

Registrare tutti i dati, le decisioni e le ragioni del progetto software per permettere una più facile verifica, validazione, valutazione e manutenzione.

Descrizione

Durante un progetto sono mantenute dettagliate registrazioni, sia sulla base del progetto che su base individuale. Per esempio, ad un ingegnere dovrebbe essere richiesto di tenere registrazioni che potrebbero comprendere

- sforzo sviluppato sui singoli moduli,
- prove eseguite su ogni modulo,
- decisioni e loro ragioni,
- raggiungimento di pietre miliari del progetto,
- problemi e loro soluzioni.

Durante e alla conclusione del progetto, queste registrazioni possono essere analizzate per stabilire un'ampia varietà di informazioni. In particolare la registrazione dei dati è molto importante per la manutenzione dei sistemi computerizzati, poiché

decisions made during the development project is not always known by the maintenance engineers.

la ragione di certe decisioni prese durante lo sviluppo del progetto non è sempre nota agli ingegneri della manutenzione.

B.14 Decision Tables (Truth Tables) (Referenced by clause 14 and D.7)

Aim

To provide a clear and coherent specification and analysis of complex logical combinations and relationships.

Description

These related methods use two dimensional tables to concisely describe logical relationships between Boolean program variables.

The conciseness and tabular nature of both methods makes them appropriate as a means of analysing complex logical combinations expressed in code.

Both methods are potentially executable if used as specifications.

Tavole di Decisioni/di Verità (Riferimento all'art. 14 e a D.7)

Scopo

Fornire una specifica chiara e coerente ed una analisi delle combinazioni logiche complesse e loro relazioni.

Descrizione

Questi metodi correlati usano due tabelle bidimensionali per descrivere in modo conciso le relazioni logiche tra le variabili di programma booleane.

La sinteticità e la natura tabellare dei due metodi li rende adatti come mezzo di analisi di combinazioni logiche complesse espresse in codice.

Entrambi i metodi sono potenzialmente eseguibili se usati come specifica.

B.15 Defensive Programming (Referenced by clause 9)

Aim

To produce programs which detect anomalous control flow, data flow or data values during their execution and react to these in a predetermined and acceptable manner.

Description

Many techniques can be used during programming to check for control or data anomalies. These can be applied systematically throughout the programming of a system to decrease the likelihood of erroneous data processing.

Two overlapping areas of defensive techniques can be identified. Intrinsic error-safe software is designed to accommodate its own design shortcomings. These shortcomings may be due to plain error of design or coding, or to erroneous requirements. The following lists some of the defensive techniques:

- variables should be range checked;
- where possible, values should be checked for plausibility;
- parameters to procedures should be type, dimension and range checked at procedure entry.

These first three recommendations help to ensure that the numbers manipulated by the program are reasonable, both in terms of the pro-

Programmazione Difensiva (Riferimento all'art. 9)

Scopo

Produrre programmi che rilevano flusso di controllo, flusso di dati o valori di dati anomali nel corso della loro esecuzione e che reagiscono a questi in modo predeterminato ed accettabile.

Descrizione

Durante la programmazione possono esser usate molte tecniche per verificare le anomalie del controllo o dei dati. Queste possono essere applicate per tutto il corso della programmazione di un sistema per ridurre la probabilità di processare dati in modo scorretto.

Si possono identificare due aree sovrapposte di tecniche difensive. Il software intrinsecamente privo di errori è progettato per conciliare le proprie deficienze di progettazione. Queste deficienze possono essere dovute ad un puro errore di progettazione o di codifica, o a requisiti sbagliati. Nel seguito sono elencate alcune tecniche difensive:

- le variabili dovrebbero essere verificate nel loro campo di variazione;
- ove possibile, dovrebbe essere verificata la credibilità dei valori;
- i parametri per le procedure dovrebbero essere verificati come tipo, dimensione e campo di variazione in sede di introduzione della procedura.

Queste prime tre raccomandazioni aiutano ad assicurare che i numeri manipolati dal programma siano ragionevoli, sia in termini di funzione

gram function and physical significance of the variables.

- Read-only and read-write parameters should be separated and their access checked. Functions should treat all parameters as read-only. Literal constants should not be write-accessible. This helps detect accidental overwriting or mistaken use of variables.

Error tolerant software is designed to "expect" failures in its own environment or use outside nominal or expected conditions, and behave in a predefined manner. Techniques include the following:

- input variables and intermediate variables with physical significance should be checked for plausibility;
- the effect of output variables should be checked, preferably by direct observation of associated system state changes;
- the software should check its configuration. This could include both the existence and accessibility of expected hardware and also that the software itself is complete. This is particularly important for maintaining integrity after maintenance procedures.

Some of the defensive programming techniques such as control flow sequence checking, also cope with external failures.

References:

Dependability of Critical Computer Systems - Part 1. E.F. Redmill (ed) Elsevier Applied Science 1988.

Dependability of Critical Computer systems - Part 2. E.F. Redmill (ed) Elsevier Applied Science 1988.

Software Engineering Aspects of Real-time Programming Concepts. E. Schoitsch, Computer Physics Communications 41 (1986) North Holland, Amsterdam.

del programma che di significato fisico delle variabili.

- I parametri di sola lettura e di lettura - scrittura dovrebbero essere separati e il loro accesso verificato. Le funzioni dovrebbero trattare tutti i parametri come sola lettura. Costanti letterali non dovrebbero essere accessibili in scrittura. Questo aiuta nel rilievo di sovrascrittura accidentale o nell'errato uso delle variabili.

Il software che tollera errori è progettato "in previsione" di mancati funzionamenti nel suo proprio ambiente o per l'uso al di fuori delle condizioni nominali o previste, e comportarsi in modo predefinito. Le tecniche comprendono quanto segue:

- per le variabili d'ingresso e le variabili intermedie con significato fisico, dovrebbe essere verificata la credibilità;
- dovrebbe essere verificato l'effetto delle variabili in uscita, preferibilmente per diretta osservazione dei cambiamenti di stato del sistema associato;
- il software dovrebbe verificare la propria configurazione. Questo potrebbe comprendere l'esistenza e l'accessibilità dell'hardware previsto ed anche che il software stesso sia completo. Questo è particolarmente importante per mantenere l'integrità dopo le procedure di manutenzione.

Alcune delle tecniche di programmazione difensiva come la verifica di sequenza del flusso di controllo copre anche i mancati funzionamenti esterni.

Riferimenti:

Dependability of Critical Computer Systems - Part 1. E.F. Redmill (ed) Elsevier Applied Science 1988.

Dependability of Critical Computer systems - Part 2. E.F. Redmill (ed) Elsevier Applied Science 1988.

Software Engineering Aspects of Real-time Programming Concepts. E. Schoitsch, Computer Physics Communications 41 (1986) North Holland, Amsterdam.

B.16 Design and Coding Standards (Referenced by D.1)

Aim

To ensure a uniform layout of the design documents and the produced code, enforce egoless programming and to enforce a standard design method.

Standard di Progettazione e di Codifica (Riferimento a D.1)

Scopo

Assicurare una organizzazione uniforme dei documenti di progettazione e del codice prodotto, imporre una programmazione spersonalizzata e un metodo di progettazione standardizzato.

Description

The rules to be adhered to are agreed at the outset of the project between the participants. These rules must comprise as a minimum

- the development method and the related coding standards to be followed, for example JSP, MASCOT, Petri-Nets etc;
- details of modularisation, for example, interface shapes, module sizes;
- use of encapsulation, inheritance and polymorphy, in case of object oriented languages;
- use or avoidance of certain language constructs, for example GOTO, EQUIVALENCE, dynamic objects, dynamic data, recursion, pointers, exits etc.; and
- the what, where and how to comment.

These rules are made to allow for ease of development, verification, assessment and maintenance. Therefore they shall address the available tools, in particular analysers and reverse engineering tools.

Descrizione

Le regole alle quali uniformarsi sono concordate all'inizio del progetto tra i partecipanti. Queste regole devono comprendere come minimo

- metodo di sviluppo e corrispondenti standard di codifica da seguire, per esempio JSP, MASCOT, reti di Petri ecc.;
- dettagli di modularizzazione, per esempio, condizioni delle interfacce, dimensioni del modulo;
- uso di incapsulamento, ereditarietà e polimorfismo, nel caso di linguaggi orientati all'oggetto;
- usare od evitare certi costrutti di linguaggio per esempio GOTO, EQUIVALENCE, oggetti dinamici, dati dinamici, ricorrenza, puntatori, uscite ecc.; e
- cosa, dove e come commentare.

Queste regole sono create per permettere facilità di sviluppo, verifica, valutazione e manutenzione. Pertanto esse devono rivolgersi agli strumenti disponibili, in particolare agli strumenti analizzatori e agli strumenti di reverse engineering.

B.17 Diverse Programming (Referenced by clause 9)

Aim

Detect and mask residual software design faults during execution of a program, in order to prevent safety critical failures of the system, and to continue operation for high reliability.

Description

In diverse programming a given program specification implemented N times in different ways. The same input values are given to the N versions, and the results produced by the N versions are compared. If the result is considered to be valid, the result is transmitted to the computer outputs.

The N versions can run in parallel on separate computers, alternatively all versions can be run on the same computer and the results subjected to an internal vote. Different voting strategies can be used on the N versions depending on the application requirements.

- If the system has a safe state, then it is feasible to demand complete agreement (all N agree) otherwise a fail-safe output value is used. For simple trip systems the vote can be biased in the safe direction. In this case the safe action would be to trip if either version demanded a trip. This approach typically uses only two versions (N=2).
- For systems with no safe state, majority voting strategies can be employed. For cases where there is no collective agreement,

Programmazione Diversificata (Riferimento all'art. 9)

Scopo

Rilevare e mascherare guasti residui della progettazione del software durante l'esecuzione di un programma, al fine di prevenire mancati funzionamenti critici per la sicurezza del sistema, e dare continuità di funzionamento ad elevata affidabilità.

Descrizione

Nella programmazione diversificata una determinata specifica di programma è implementata N volte in modi differenti. Alle N versioni vengono dati i medesimi valori d'ingresso, e vengono confrontati i risultati prodotti dalle N versioni. Se il risultato è considerato valido, questo viene trasmesso alle uscite del computer.

Le N versioni possono essere eseguite in parallelo su computer separati, in alternativa tutte le versioni possono essere eseguite sul medesimo computer ed i risultati possono essere soggetti a voto interno. Sulle N versioni possono essere usate diverse strategie di voto in relazione ai requisiti dell'applicazione.

- se il sistema ha uno stato sicuro, è possibile richiedere l'accordo completo (tutte le N versioni concordano) altrimenti è usato un valore di uscita sicuro. Per sistemi che non hanno ritorno il voto può essere indirizzato nella direzione sicura. In questo caso l'azione sicura dovrebbe essere di scattare se entrambe le versioni richiedessero uno scatto. Questo approccio usa tipicamente solo due versioni (N=2).
- per sistemi con nessun stato sicuro, possono essere usate strategie di voto maggioritario. Per casi in cui non vi è accordo comune, pos-

probabilistic approaches can be used in order to maximise the chance of selecting the correct value, for example, taking the middle value, temporary freezing of outputs until agreement returns etc.

This technique does not eliminate residual software design faults, but it provides a measure to detect and mask before they can affect safety.

References:

Dependable Computing: From Concepts to Design Diversity. A. Avizienis, J.C. Laprie, Proc IEEE, Vol 74, 5, May 1986.

A Theoretical Basis for the Analysis of Multi-version Software subject to Co-incident Failures. D.E. Eckhardt and L.D. Lee, IEEE Trans. SE-11, No 12, 1985.

sono essere usati approcci probabilistici al fine di massimizzare la possibilità di scegliere il valore corretto, per esempio, prendendo il valore medio, congelamento temporaneo delle uscite fino al ritorno di accordo ecc.

Questa tecnica non elimina i guasti residui della progettazione del software, ma fornisce un provvedimento per rilevarli e mascherarli prima che essi possano influenzare la sicurezza.

Riferimenti:

Dependable Computing: From Concepts to Design Diversity. A. Avizienis, J.C. Laprie, Proc IEEE, Vol 74, 5, May 1986.

A Theoretical Basis for the Analysis of Multi-version Software subject to Co-incident Failures. D.E. Eckhardt and L.D. Lee, IEEE Trans. SE-11, No 12, 1985.

B.18 Dynamic Reconfiguration (Referenced by clause 9)

Aim

To maintain system functionality despite an internal fault.

Description

The logical architecture of the system has to be such that it can be mapped onto a subset of the available resources of the system. The architecture needs to be capable of detecting a failure in a physical resource and then remapping the logical architecture back onto the restricted resources left functioning. Although the concept is more traditionally restricted to recovery from failed hardware units, it is also applicable to failed software units if there is sufficient "run-time redundancy" to allow a software re-try or if there is sufficient redundant data to make the individual and isolated failure a little import.

Although traditionally applied to hardware, this technique is being developed for application to software and, thus, the total system. It must be considered at the first system design stage.

References:

Critical Issues in the Design of a Reconfigureable Control Computer. H. Schmid, J. Lam,R. Naro and K. Weir, FTCS 14 June 1984 IEEE 1984.

Assigning Processes to Processors: A Fault-tolerant Approach. June 1984 G. Kar and C.N.Nikolaou, Watson Research Centre, Yorktown

Riconfigurazione dinamica (Riferimento all'art. 9)

Scopo

Conservare la funzionalità del sistema nonostante un guasto interno.

Descrizione

L'architettura logica del sistema deve essere tale che essa possa essere mappata su un sottoinsieme di risorse disponibili del sistema. L'architettura necessita di essere in grado di rilevare un mancato funzionamento in una risorsa fisica e quindi mappare nuovamente l'architettura logica verso le risorse limitate che rimangono funzionanti. Per quanto il concetto sia tradizionalmente ristretto al ripristino di unità hardware malfunzionanti, esso è applicabile anche ad unità software malfunzionanti se vi è sufficiente "ridondanza nel tempo di esecuzione" da consentire un nuovo tentativo del software, o se vi sono dati ridondanti sufficienti da rendere di scarsa importanza il mancato funzionamento individuale ed isolato.

Benché tradizionalmente applicata all'hardware, questa tecnica è sviluppata per applicazioni software e, quindi all'intero sistema. Essa deve essere presa in considerazione nella prima fase la di progettazione del sistema.

Riferimenti:

Critical Issues in the Design of a Reconfigureable Control Computer. H. Schmid, J. Lam,R. Naro and K. Weir, FTCS 14 June 1984 IEEE 1984.

Assigning Processes to Processors: A Fault-tolerant Approach. June 1984 G. Kar and C.N.Nikolaou, Watson Research Centre, Yorktown

B.19 Equivalence Classes and Input Partition Testing (Referenced by D2 and D.3)

Aim

To test the software adequately using a minimum of test data. The test data is obtained by selecting the partitions of the input domain required to exercise the software.

Description

This testing strategy is based on the equivalence relation of the inputs, which determines a partition of the input domain.

Test cases are selected with the aim of covering all subsets of this partition. At least one test case is taken from each equivalence class.

There are two basic possibilities for input partitioning which are:

- equivalence classes may be defined on the specification. The interpretation of the specification may be either input oriented, for example the values selected are treated in the same way or output oriented, for example the set of values leading to the same functional result; and
- equivalence classes may be defined on the internal structure of the program. In this case the equivalence class results are determined from static analysis of the program, for example the set of values leading to the same path being executed.

References:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, USA, 1979.

B.19 Classi di Equivalenza e Prova di Partizione dei dati d'ingresso (Riferimento a D2 e D.3)

Scopo

Provare il software in modo adeguato usando un minimo di dati di prova. I dati di prova sono ottenuti scegliendo le partizioni del dominio d'ingresso richieste per sollecitare il software.

Descrizione

Questa strategia di prova è basata sulla relazione di equivalenza degli ingressi, che determina una partizione del dominio dell'ingresso.

Sono scelti dei casi di prova con lo scopo di trattare tutti i sottoinsiemi di questa partizione. Almeno un caso di prova è preso da ogni classe di equivalenza.

Vi sono due possibilità basilari per la partizione d'ingresso, che sono:

- le classi di equivalenza possono essere definite nella specifica. L'interpretazione della specifica può essere, sia orientata all'ingresso, per esempio i valori scelti sono trattati nel medesimo modo, o orientati all'uscita, per esempio l'insieme di valori che conducono al medesimo risultato funzionale; e
- le classi di equivalenza possono essere definite nella struttura interna del programma. In questo caso i risultati della classe di equivalenza sono determinati dall'analisi statica del programma, per esempio l'insieme di valori che conducono al medesimo percorso che deve essere eseguito.

Riferimenti:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, USA, 1979.

B.20 Error Detecting and Correcting Codes (Referenced by clause 9)

Aim

To detect and correct errors in sensitive information.

Description

For an information of n bits, a coded block of k bits is generated which enables errors to be detected and corrected. Different types of code include:

- hamming codes;
- cyclic codes;
- polynomial codes.

References:

The Technology of Error Correcting Codes. E.R. Berlekamp, Proc. of the IEEE, 68, 5, 1980.

A Short Course on Error Correcting Codes. N.J.A. Sloane, Springer-Verlag, Wien, 1975.

B.20 Codici di Correzione e di Rilevamento dell'Errore (Riferimento all'art. 9)

Scopo

Rilevare e correggere errori con informazioni significative.

Descrizione

Per informazioni a n bit è generato un blocco codificato di k bit che rende possibile il rilievo e la correzione degli errori. I diversi tipi di codice comprendono:

- codici hamming;
- codici ciclici;
- codici polinomiali.

Riferimenti:

The Technology of Error Correcting Codes. E.R. Berlekamp, Proc. of the IEEE, 68, 5, 1980.

A Short Course on Error Correcting Codes. N.J.A. Sloane, Springer-Verlag, Wien, 1975.

B.21 Error Guessing (Referenced by D.2 and D.8)

Aim

To remove common programming errors.

Description

Testing experience and intuition combined with knowledge and curiosity about the system under test may add some uncategorised test cases to the designed test case set. Special values or combinations of values may be error-prone. Some interesting test cases may be derived from inspection checklists. It may also be considered whether the system is robust enough. Can the buttons be pushed on the front-page too fast or too often? What happens if two buttons are pushed simultaneously?

Supposizione di Errore (Riferimento a D.2 e D.8)

Scopo

Eliminare gli errori di programmazione comuni.

Descrizione

L'esperienza nelle prove e l'intuizione unita alla conoscenza e alla curiosità circa il sistema in prova può portare all'aggiunta di qualche caso di prova che non rientra nelle categorie dell'insieme di prove progettate. Valori speciali o combinazioni di valori possono essere inclini all'errore. Alcuni casi di prova interessanti possono essere derivati dall'ispezione delle liste di controllo. Può anche essere esaminato se il sistema è abbastanza robusto. I pulsanti sul pannello frontale possono essere spinti troppo rapidamente o troppo spesso? Che cosa succede se due pulsanti sono premuti simultaneamente?

B.22 Error Seeding (Referenced by D.2)

Aim

To ascertain whether a set of test cases is adequate.

Description

Some known error types are inserted in the program, and the program is executed with the test cases under test conditions. If only some of the seeded errors are found, the test case set is not adequate. The ratio of found seeded errors to the total number of seeded errors is an estimate of the ratio of found real errors to total number errors. This gives a possibility of estimating the number of remaining errors and thereby the remaining test effort.

Semina di Errori (Riferimento a D.2)

Scopo

Accertare se una serie di casi di prova è adeguata.

Descrizione

Sono inseriti alcuni tipi di errori nel programma e il programma viene quindi eseguito con i casi di prova nelle condizioni di prova. Se vengono trovati solo alcuni degli errori seminati, l'insieme dei casi di prova non è adeguato. Il rapporto degli errori seminati trovati rispetto al numero totale degli errori seminati è una stima del rapporto degli errori effettivi trovati sul numero totale degli errori. Questo da una possibilità di stimare il numero degli errori che rimangono e quindi il residuo impegno per le prove.

$$\frac{\text{Errori seminati trovati}\ /\ \textit{Found seeded errors}}{\text{Numero totale degli errori seminati}\ /\ \textit{Total number of seeded errors}} = \frac{\text{Errori effettivi trovati}\ /\ \textit{Found real errors}}{\text{Numero totale degli errori effettivi}\ /\ \textit{Total number of real errors}}$$

The detection of all the seeded errors may indicate either that the test case set is adequate, or that the seeded errors were too easy to find. The limitations of the method are that in order, to obtain any usable results, the error types as well as the seeding positions must reflect the statistical distribution of real errors.

Lo scoprire tutti gli errori seminati può indicare, sia che l'insieme dei casi di prova è adeguato, sia che gli errori seminati sono stati trovati troppo facilmente. I limiti del metodo sono dovuti al fatto per ottenere qualche risultato utile, sia i tipi di errore che le posizioni di semina devono rispecchiare la distribuzione statistica degli errori effettivi.

B.23 Event Tree Analysis (Referenced by clause 14)

Aim

To model, in a diagrammatic form, the sequence of events that can develop in a system after an initiating event, and thereby indicate how serious consequences can occur.

Description

On the top of the diagram is written the sequence conditions that are relevant in the development following the initiating event which is the target of the analysis. Starting under the initiating event, one draws a line to the first condition in the sequence. There the diagram branches off into a "yes" and a "no" branch, describing how the future developments depend on the condition. For each of these branches, one continues to the next condition in a similar way. Not all conditions are, however, relevant for all branches. One continues to the end of the sequence, and each branch of the tree constructed in this way represents a possible consequence. The event tree can be used to compute the probability of the various consequences based on the probability and number of conditions in the sequence.

References:

Event Trees and their Treatment on PC Computers. N. Limnious and J.P. Jeannette, Reliability Engineering, Vol. 18, No. 3 1987.

B.23 Analisi dell'Albero degli Eventi (Riferimento all'art. 14)

Scopo

Modellare, in forma di diagramma, la sequenza degli eventi che si possono sviluppare in un sistema dopo un evento iniziale, indicando così come possono verificarsi serie conseguenze.

Descrizione

Alla sommità del diagramma sono indicate le condizioni della sequenza che sono attinenti nello sviluppo successivo l'evento iniziale che è l'obbiettivo dell'analisi. Partendo dall'evento iniziale si disegna una linea verso la prima condizione della sequenza. Qui il diagramma si dirama in rami "si" e "no", descrivendo come gli sviluppi futuri dipendono dalle condizioni. Per ciascuno di questi rami si continua verso la condizione seguente in modo simile. Non tutte le condizioni tuttavia, sono attinenti per tutti i rami. Si continua verso la fine della sequenza, e ogni ramo dell'albero così costruito rappresenta una possibile conseguenza. L'albero degli eventi può essere usato per calcolare la probabilità delle varie conseguenze basate sulla probabilità e sul numero delle condizioni della sequenza.

Riferimenti:

Event Trees and their Treatment on PC Computers. N. Limnious and J.P. Jeannette, Reliability Engineering, Vol. 18, No. 3 1987.

B.24 Fagan Inspections (Referenced by D.8)

Aim

To reveal errors in all phases of the program development.

Description

A "formal" audit on quality assurance documents aimed at finding errors and omissions. The inspection process consists of five phases; Planning, Preparation, Inspection, Rework and Follow up. Each of these phases has its own separate objective. The complete system development (specification, design, coding and testing) must be inspected.

References:

Design and Code Inspections to Reduce Errors in Program Development. M.E. Fagan, IBM Systems Journal, No. 3, 1976.

B.24 Ispezioni di Fagan (Riferimento a D.8)

Scopo

Rilevare errori in tutte le fasi dello sviluppo del programma.

Descrizione

Una ispezione "formale" su documenti di assicurazione qualità con lo scopo di trovare errori ed omissioni. Il processo di ispezione consiste in cinque fasi, Pianificazione, Preparazione, Ispezione, Modifica e Rafforzamento. Ciascuna di queste fasi ha il proprio separato obbiettivo. Deve essere ispezionato lo sviluppo completo del sistema (specifica, progettazione, codifica e prova).

Riferimenti:

Design and Code Inspections to Reduce Errors in Program Development. M.E. Fagan, IBM Systems Journal, N3, 1976.

B.25 Failure Assertion Programming (Referenced by clause 9)

Aim

To detect residual software design faults during execution of a program, in order to prevent safety critical failures of the system and to continue operation for high reliability.

Description

The assertion programming method follows the idea of checking a pre-condition (before a sequence of statements is executed, the initial conditions are checked for validity) and a post-condition (results are checked after the execution of a sequence of statements). If either the pre-condition or the post-condition is not fulfilled, the processing stops with an error.

For example,

assert <pre-condition>;
action 1;
:
:
action x;
assert <post-condition>;

References:

A Discipline of Programming. E.W. Dijkstra, Prentice Hall, 1976.

The Science of Programming. D. Gries, Springer-Verlag, 1981.

Software Development - A Rigorous Approach. C.B. Jones, Prentice-Hall, 1980.

Programmazione di Asserzione del mancato funzionamento (Riferimento all'art. 9)

Scopo

Rilevare i malfunzionamenti residui della progettazione del software durante l'esecuzione di un programma, con il fine di impedire mancati funzionamenti del sistema e continuare operativamente con elevata affidabilità.

Descrizione

Il metodo di programmazione di asserzione segue l'idea di verificare una precondizione (prima che una sequenza di comandi sia eseguita, viene verificata la validità delle condizioni iniziali) e una postcondizione (i risultati vengono verificati dopo l'esecuzione di una sequenza di comandi). Se, o la precondizione, o la postcondizione non sono soddisfatte, il processo si arresta con un errore.

Per esempio,

asserisce <precondizione>;
azione 1;
:
:
azione x;
asserisce <post-condizione>;

Riferimenti:

A Discipline of Programming. E.W. Dijkstra, Prentice Hall, 1976.

The Science of Programming. D. Gries, Springer-Verlag, 1981.

Software Development - A Rigorous Approach. C.B. Jones, Prentice-Hall, 1980.

B.26 SEEA - Software Error Effect Analysis (Ref'd by clause 9, 11 & 14)

Aim

To identify software modules, their criticality; to propose means for detecting software errors and enhancing software robustness; to evaluate the amount of validation needed on the various software components.

Description

The analysis is done in three phases:

- Vital software modules identification;
  Determination of the depth of the analysis (at the level of a single instruction line, a group of instructions, a module, etc...) needed for each software module, from its specification.

SEEA - Analisi dell'Effetto dell'Errore Software (Riferimento all'art. 9, 11 & 14)

Scopo

Identificare i moduli software, la loro criticità; proporre mezzi per rilevare errori del software e migliorare la robustezza del software; valutare l'ammontare delle necessità di validazione nelle varie componenti del software.

Descrizione

L'analisi è eseguita in tre fasi:

- Identificazione dei moduli software vitali;
  Determinazione della profondità dell'analisi (a livello di una linea di istruzione singola, di un gruppo di istruzioni, di un modulo, ecc.....) richiesta, per ciascun modulo software, dalla sua specifica.

- Software error analysis

  The result of this phase is a table listing the following information:
  - module name;
  - error considered;
  - consequences of the error at the module level;
  - consequences at the system level;
  - violated safety criterion;
  - error criticality;
  - proposed error detection means;
  - violated criterion if the detection means is implemented;
  - residual criticality if the detection means is implemented.
- Synthesis

  The synthesis identifies the remaining unsafe scenarios and the validation effort needed given the criticality of each module.

SEEA, being an in-depth analysis carried out by an independent team, is a powerful bug-finding method.

References:

Railway fixed equipment and rolling stock. Data processing. Software dependability - Adapted methods for software safety analysis, French standard NF F 71-013, December 1990.

IRSE International Technical Committee Report No. 1

SAFETY SYSTEM VALIDATION with regard to cross acceptance of signalling systems by the railways

P. THIREAU

Méthodologie d'Analyse des Effets des Erreurs Logiciel (AEEL) appliquée à l'étude d'un logiciel de haute sécurité.

5th International Conference on Reliability and Maintainability.

Biarritz - France - 1986

D. J. REIFER

Software failures modes and effects analysis

Transaction on reliability IEE - Vol. R28 No. 3

August 79 - Pages 247/249

J. R. FRAGOLA and J. F. SPAMM

The software error effects analysis, a qualitative design tool

IEEE Symposium Computer on Software Reliability

1973 - pages 90/93

- Analisi dell'errore software

  Il risultato di questa fase è una tabella che elenca le seguenti informazioni:
  - nome del modulo;
  - errore considerato;
  - conseguenze dell'errore a livello del modulo;
  - conseguenze a livello del sistema;
  - criterio di sicurezza violato;
  - criticità dell'errore;
  - mezzi di rilievo dell'errore proposti;
  - criterio violato se sono implementati i mezzi di rilievo;
  - criticità residua se sono implementati i mezzi di rilievo.
- Sintesi

  La sintesi identifica gli scenari non sicuri rimanenti e lo sforzo di validazione necessario data la criticità di ciascun modulo.

Poiché la tecnica SEEA è un'analisi in profondità eseguita da un gruppo indipendente, è un metodo potente per trovare errori di programmazione.

Riferimenti:

Railway fixed equipment and rolling stock. Data processing. Software dependability - Adapted methods for software safety analysis, French standard NF F 71-013, December 1990.

IRSE International Technical Committee Report No. 1

SAFETY SYSTEM VALIDATION with regard to cross acceptance of signalling systems by the railways

P. THIREAU

Méthodologie d'Analyse des Effets des Erreurs Logiciel (AEEL) appliquée à l'étude d'un logiciel de haute sécurité.

5th International Conference on Reliability and Maintainability.

Biarritz - France - 1986

D. J. REIFER

Software failures modes and effects analysis

Transaction on reliability IEE - Vol. R28 No. 3

August 79 - Pages 247/249

J. R. FRAGOLA and J. F. SPAMM

The software error effects analysis, a qualitative design tool

IEEE Symposium Computer on Software Reliability

1973 - pages 90/93

B.27 Fault Detection and Diagnosis (Referenced by clause 9)

Aim

To detect faults in a system, which might lead to a failure, thus providing the basis for countermeasures in order to minimise the consequences of failures.

Description

Fault detection is the process of checking a system for erroneous states (which are caused, as explained before, by a fault within the (sub)system to be checked). The primary goal of fault detection is to inhibit the effect of wrong results. A system which delivers either correct results, or no results at all, is called "self checking".

Fault detection is based on the principles of redundancy (mainly to detect hardware faults) and diversity (software faults). Some sort of voting is needed to decide on the correctness of results. Special methods applicable are assertion programming, N-version programming and the safety bag technique and on hardware level by introducing sensors, control loops, error checking codes, etc.

Fault detection may be achieved by checks in the value domain or in the time domain on different levels, especially on the physical (temperature, voltage etc.), logical (error detecting codes), functional (assertions) or external level (plausibility checks). The results of these checks may be stored and associated with the data affected to allow failure tracking.

Complex systems are composed of subsystems. The efficiency of fault detection, diagnosis and fault compensation depends on the complexity of the interactions among the subsystems, which influences the propagation of faults.

Fault diagnosis isolates the smallest subsystem that may be identified. Smaller subsystems allow a more detailed diagnosis of faults (identification of erroneous states).

B.28 Fault Tree Analysis (Referenced by clauses 9 & 14)

Aim

To aid in the analysis of events, or combinations of events, that will lead to a hazard or serious consequence.

Rilievamento Guasto e Diagnosi (Riferimento all'art. 9)

Scopo

Rilevare malfunzionamenti in un sistema che potrebbero condurre ad un mancato funzionamento, fornendo pertanto le basi per contromisure finalizzate a minimizzare le conseguenze dei mancati funzionamenti.

Descrizione

Il rilievo del malfunzionamento è il processo di verifica di esistenza di stati erronei in un sistema (che sono causati, come spiegato in precedenza, da un malfunzionamento nel sistema/sottosistema da verificare). Il fine primario della rilevazione di un malfunzionamento è quello di inibire l'effetto di risultati sbagliati. Un sistema che produce sia risultati corretti che nessun risultato affatto, è detto che "si autocontrolla".

Il rilievo del malfunzionamento è basato sui principi della ridondanza (principalmente per rilevare malfunzionamenti hardware) e sulla diversità (malfunzionamenti software). Per decidere sulla correttezza dei risultati è necessario qualche tipo di votazione. Metodi speciali applicabili sono: programmazione di asserzione, programmazione con N-versioni e tecnica della safety bag; a livello di hardware introducendo sensori, costrutti iterativi di controllo, codici di verifica dell'errore, ecc.

Il rilievo del malfunzionamento può essere conseguito con verifiche nel dominio del valore o nel dominio del tempo su livelli differenti, specialmente fisico (temperatura, tensione, ecc.), logico (codici di rilievo dell'errore), funzionale (asserzioni) o a livello esterno (verifiche di accettabilità). I risultati di queste verifiche possono essere memorizzati e associati con dati sui quali hanno influenza per permettere di seguire le tracce del mancato funzionamento.

I sistemi complessi sono composti da sottosistemi. L'efficienza del rilievo del malfunzionamento, della diagnosi e della compensazione del guasto dipende dalla complessità delle interazioni tra i sottosistemi, che influenzano la propagazione dei guasti.

La diagnosi del guasto isola il sottosistema più piccolo che possa essere identificato. Il sottosistema più piccolo permette una diagnosi più dettagliata dei malfunzionamenti (identificazione degli stati erronei).

Analisi dell'Albero dei Guasti (Riferimento all'art. 9 & 14)

Scopo

Aiutare nell'analisi degli eventi, o nella combinazione degli eventi, che condurranno ad un pericolo o ad una seria conseguenza.

Description

Starting at an event which would be the immediate causes of a hazard or serious consequence (the "top event") analysis is carried out along a tree path. Combinations of causes are described with logical operators (and, or, etc). Intermediate causes are analysed in the same way, and so on back to basic events where analysis stops.

The method is graphical, and a set of standardised symbols are used to draw the fault tree. It is mainly intended for the analysis of hardware systems, but there have also been attempts to apply this approach to software failure analysis.

References:

System Reliability Engineering Methodology: A Discussion of the State of the Art. J.B. Fusseland J.S. Arend, Nuclear Safety 20(5), 1979.

Fault Tree Handbook. W.E. Vesely et. al., NUREG-0942, Division of System Safety Office of Nuclear Reactor Regulation, U.S. Nuclear Regulatory Commission Washington, D.C.20555, 1981.

Reliability Technology. A.E. Greene and A.J. Bourne, Wiley-Interscience, 1972.

B.29 Finite State Machines/State Transition Diagrams (Ref'd by D.5 & D.7)

Aim

To define or implement the control structure of a system.

Description

Many systems can be defined in terms of their states, their inputs, and their actions. Thus when in state S1, on receiving input I a system might carry out action A and move to state S2. By defining a system's actions for every input in every state we can define a system completely. The resulting model of the system is called a Finite State Machine. It is often drawn as a so-called state transition diagram showing how the system moves from one state to another, or as a matrix in which the dimensions are state and input and the matrix cells contain the action and new state resulting from receipt of the input in the given state.

Where a system is complicated or has a natural structure this can be reflected in a layered FSM.

A specification or design expressed as an FSM can be checked for completeness (the system must have an action and new state for every input in every state), for consistency (only one state change is defined for each state/input

Descrizione

Partendo da un evento che potrebbe essere la causa immediata di un pericolo o di una seria conseguenza ("l'evento critico") l'analisi è sviluppata lungo il percorso ad albero. Le combinazioni delle cause sono descritte con operazioni logiche (and, or, ecc.). Le cause intermedie sono analizzate nel medesimo modo, e così via a ritroso fino agli eventi di base ove l'analisi si arresta.

Il metodo è grafico, ed è usato un insieme di simboli normalizzati per disegnare l'albero dei guasti. Esso è destinato principalmente all'analisi dei sistemi hardware, ma vi sono stati anche tentativi di applicare questo approccio all'analisi del mancato funzionamento del software.

Riferimenti:

System Reliability Engineering Methodology: A Discussion of the State of the Art. J.B. Fusseland J.S. Arend, Nuclear Safety 20(5), 1979.

Fault Tree Handbook. W.E. Vesely et. al., NUREG-0942, Division of System Safety Office of Nuclear Reactor Regulation, U.S. Nuclear Regulatory Commission Washington, D.C.20555, 1981.

Reliability Technology. A.E. Greene and A.J. Bourne, Wiley-Interscience, 1972.

Macchine a Stati Finiti /Diagrammi di Transizione dello Stato (Riferimento a D.5 & D.7)

Scopo

Definire o implementare la struttura di controllo di un sistema.

Descrizione

Molti sistemi possono essere definiti nei termini dei loro stati, dei loro ingressi e delle loro azioni. Perciò quando nello stato S1, al ricevimento dell'ingresso I un sistema potrebbe svolgere l'azione A e andare nello stato S2. Con la definizione delle azioni di un sistema per ogni ingresso in ogni stato possiamo definire un sistema in modo completo. Il modello risultante del sistema è chiamato una Macchina a Stati Finiti. Esso è spesso disegnato come un cosiddetto diagramma di transizione di stato che mostra come il sistema passa da uno stato all'altro, o come una matrice, nella quale le dimensioni sono stati e ingressi e le celle della matrice contengono l'azione ed il nuovo stato risultante dal ricevimento dell'ingresso in un determinato stato.

Quando un sistema è complicato o ha una struttura naturale, questo può riflettersi in un FSM a strati.

Una specifica o una progettazione espressa come un FSM può essere verificata per completezza (un sistema deve avere un'azione e un nuovo stato per ogni ingresso in ciascun stato), per coerenza (è definito un solo cambiamento di stato per ogni

pair) and reachability (whether or not it is possible to get from one state to another by any sequence of inputs). These are important properties for critical systems and they can be checked. Tools to support these checks are easily written. Algorithms also exist that allow the automatic generation of test cases for verifying an FSM implementation or for animating an FSM model.

References:

Introduction to the theory of Finite State Machines. A. Gill, McGraw-Hill 1962.

coppia stato/ingresso) e raggiungibilità (reachability) (se è possibile o no raggiungere uno stato dall'altro con qualunque sequenza di ingressi). Queste sono proprietà importanti per sistemi critici ed esse possono essere verificate. Strumenti per supportare queste verifiche sono scritti facilmente. Esistono anche algoritmi che consentono la generazione automatica di casi di prova per verificare una implementazione FSM o per animare un modello FSM.

Riferimenti:

Introduction to the theory of Finite State Machines. A. Gill, McGraw-Hill 1962.

B.30 Formal Methods (Referenced by clause 8, clause 10 and D.5)

Metodi Formali (Riferimento l'art. 8, 10 e D.5)

Aim

The development of software in a way that is based on mathematics. This includes formal design and formal coding techniques.

Description

Formal methods provide a means of developing a description of a system at some stage in its development specification, design or code. The resulting description takes a mathematical form and can be subjected to mathematical analysis to detect various classes of inconsistency or incorrectness. Moreover, the description can in some cases be analysed by machine with a rigour similar to the syntax checking of a source program by a compiler, or animated to display various aspects of the behaviour of the system described. Animation can give extra confidence that the system meets the real requirement as well as the formally specified requirement.

A formal method will generally offer a notation (generally some form of discrete mathematics being used), a technique for deriving a description in that notation, and various forms of analysis for checking a description for different correctness properties.

Several examples of Formal Methods are described in the following subsections of this bibliography. The Formal Methods described are CCS, CSP, HOL, LOTOS, OBJ, Temporal Logic, VDM and Z.

Scopo

Lo sviluppo di software in modo che sia su basi matematiche. Questo comprende progettazione formale e tecniche di codifica formali.

Descrizione

I metodi formali forniscono i mezzi per sviluppare la descrizione di un sistema ad un certo momento nella sua specifica di sviluppo, progettazione o codice. La descrizione risultante assume una forma matematica e può essere soggetta ad un'analisi matematica per rilevare varie classi di inconsistenza o scorrettezza. Inoltre, in alcuni casi la descrizione può essere analizzata con una macchina con un rigore simile alla verifica della sintassi di un programma sorgente mediante un compilatore, o essere animata per mostrare vari aspetti del comportamento del sistema descritto. L'animazione può dare ulteriore fiducia che il sistema soddisfa il requisito reale così come il requisito specificato formalmente.

Un metodo formale generalmente offre un'annotazione (generalmente vengono usate alcune forme di matematica discreta), una tecnica per derivare una descrizione in tale annotazione, e varie forme di analisi per verificare una descrizione per diverse proprietà di correttezza.

Nella seguente sottosezione di questa bibliografia sono descritti esempi vari di Metodi Formali. I Metodi Formali descritti sono: are CCS, CSP, HOL, LOTOS, OBJ, Logica di Tempo, VDM e Z.

B.30.1 CCS - Calculus of Communicating Systems

CCS - Calcolo dei sistemi di comunicazione

Aim

CCS is a means for describing and reasoning about the behaviour of systems of concurrent, communicating processes.

Description

Similar to CSP, CCS is a mathematical calculus concerned with the behaviour of systems. The

Scopo

CCS è un mezzo per descrivere e ragionare sul comportamento dei sistemi di processi di comunicazione concorrenti.

Descrizione

Come il CSP, il CCS è un calcolo matematico correlato con il comportamento dei sistemi. La progettazio-

system design is modelled as a network of independent processes operating sequentially or in parallel. Processes can communicate via ports (similar to CSP's channels), the communication only taking place when both processes are ready. Non-determinism can be modelled. Starting from a high-level abstract description of the entire system (known as a trace), it is possible to carry out a step-wise refinement of the system into a composition of communicating processes whose total behaviour is that required of the whole system. Equally, it is possible to work in a bottom up fashion, combining processes and deducing the properties of the resulting system using inference rules related to the composition rules.

References:

A Calculus of Communicating Systems. R Milner, Report number ECS-LFCS-86-7, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, UK.

The Specification of Complex Systems. B Cohen, W T Harwood, M I Jackson. Addison-Wesley, 1986.

ne del sistema è modellata come una rete di processi indipendenti che funzionano in sequenza o in parallelo. I processi possono comunicare attraverso porte (simili ai canali del CSP) e la comunicazione ha luogo solo quando entrambi i processi sono pronti. Il non determinismo può essere modellato. Partendo da una descrizione astratta ad elevato livello dell'intero sistema (conosciuta come traccia), è possibile eseguire un procedimento per raffinamenti successivi del sistema in una composizione di processi di comunicazione il cui comportamento totale è quello richiesto dall'intero sistema. Ugualmente, è possibile lavorare in una modalità bottom-up, combinando i processi e deducendo le proprietà del sistema risultante usando regole di interferenza correlate con le regole di composizione.

Riferimenti:

A Calculus of Communicating Systems. R Milner, Report number ECS-LFCS-86-7, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, UK.

The Specification of Complex Systems. B Cohen, W T Harwood, M I Jackson. Addison-Wesley, 1986.

B.30.2 CSP - Communicating Sequential Processes

Aim

CSP is a technique for the specification of concurrent software systems, i.e. systems of communicating processes operating concurrently.

Description

CSP provides a language for the specification of systems of processes and proof for verifying that the implementation of processes satisfies their specifications (described as a trace - permissible sequences of events).

A system is modelled as a network of independent processes. Each process is described in terms of all of its possible behaviours. A system is modelled by composing processes sequentially or in parallel. Processes can communicate (synchronise or exchange data) via channels, the communication only taking place when both processes are ready. The relative timing of events can be modelled.

The theory behind CSP was directly incorporated into the architecture of the INMOS transputer, and the OCCAM language allows a CSP-specified system to be directly implemented on a network of transputers.

References:

Communicating Sequential Processes. C A R Hoare, Prentice-Hall, 1985

CSP - Processi Sequenziali di Comunicazione

Scopo

Il CSP è una tecnica per la specifica dei sistemi software concorrenti, cioè sistemi di processi di comunicazione che funzionano in concorrenza.

Descrizione:

Il CSP fornisce un linguaggio per la specifica di sistemi di processi e prova per verificare che l'implementazione dei processi soddisfa le loro specifiche (descritto come traccia - ammissibile serie di eventi).

Un sistema è modellato come una rete di processi indipendenti. Ogni processo è descritto nei termini di tutti i suoi possibili comportamenti. Un sistema è modellato componendo i processi in modo sequenziale o in parallelo. I processi possono comunicare (sincronizzare o scambiare dati) attraverso canali, e le comunicazioni hanno luogo soltanto quando entrambi i processi sono pronti. La temporizzazione relativa degli eventi può essere modellata.

La teoria di supporto al CSP è stata direttamente incorporata nell'architettura del transputer INMOS, ed il linguaggio OCCAM consente che un sistema specificato come CSP possa essere implementato direttamente su una rete di transputers.

Riferimenti:

Communicating Sequential Processes. C A R Hoare, Prentice-Hall, 1985

B.30.3 HOL - Higher Order Logic

Aim

This is a formal language intended as a basis for hardware specification and verification.

Description

HOL (Higher Order Logic) refers to a particular logic notation and its machine support system, both of which were developed at the University of Cambridge Computer Laboratory. The logic notation is mostly taken from Church's Simple Theory of Types and the machine support system is based upon the LCF (Logic of Computable Functions) system.

References:

HOL: A Machine Orientated Formulation of Higher Order Logic. M. Gordon, University of Cambridge Technical Report, Number - 68.

HOL - Logica di Ordine Superiore (Higher Order Logic)

Scopo

Questo è un linguaggio formale destinato come base per la specifica e la verifica dell'hardware.

Descrizione

HOL (Higher Order Logic) fa riferimento ad una particolare annotazione logica e al suo sistema di supporto della macchina ed entrambi sono stati sviluppati al "Computer Laboratory" della università di Cambridge. L'annotazione logica è per lo più presa dalla "Church's Simple Theory of Types" e il sistema di supporto della macchina è basato sul sistema LCF (Logic of Computable Functions).

Riferimenti:

HOL: A Machine Orientated Formulation of Higher Order Logic. M. Gordon, University of Cambridge Technical Report, Number - 68.

B.30.4 LOTOS

Aim

LOTOS is a means for describing and reasoning about the behaviour of systems of concurrent, communicating processes.

Description

LOTOS (Language for Temporal Ordering Specification) is based on CCS with additional features from the related algebras CSP and CIRCAL (Circuit Calculus). It overcomes the weakness of CCS in the handling of data structures and value expressions by combining it with a second component based on the abstract data type language ACT ONE. The process definition component of LOTOS could, however, be used with other formalisms for the description of abstract data types.

Reference:

Information Processing Systems - Open Systems Inter-connection - LOTOS - A Formal Description Technique based on the Temporal Ordering of Observational Behaviour. ISO DIS 8807, July 20, 1987.

LOTOS

Scopo

LOTOS è un mezzo per descrivere e ragionare sul comportamento dei sistemi di processi di comunicazione concorrenti.

Descrizione

LOTOS (Language for Temporal Ordering Specification) è basato sul CCS con caratteristiche addizionali prese dall'algebra correlata di CSP e CIRCAL (Circuit Calculus). Esso supera la debolezza di CCS nel maneggiare strutture di dati ed espressioni di valori combinandoli con una seconda componente basata sul linguaggio ACT ONE per i tipi di dati astratti. La componente di definizione del processo di LOTOS potrebbe, tuttavia, essere usata con altri formalismi per la descrizione di tipi di dati astratti.

Riferimento:

Information Processing Systems - Open Systems Inter-connection - LOTOS - A Formal Description Technique based on the Temporal Ordering of Observational Behaviour. ISO DIS 8807, July 20, 1987.

B.30.5 OBJ

Aim

To provide a precise system specification with user feed-back and system validation prior to implementation.

Description

OBJ is an algebraic specification language. Users specify requirements in terms of algebraic equation. The behavioural, or constructive, aspects of the system are specified in terms of operations acting on abstract data types (ADT). An

OBJ

Scopo

Fornire una specifica di sistema precisa con retroazione dell'utente e un sistema di validazione prima dell'implementazione.

Descrizione

OBJ è un linguaggio di specificazione algebrica. Gli utenti specificano i requisiti in termini di equazioni algebriche. Gli aspetti di comportamento o costruttivi del sistema sono specificati in termini di operazioni che agiscono sui tipi di dati

ADT is like an ADA package where the operator behaviour is visible whilst the implementation details are "hidden".

An OBJ specification, and subsequent step-wise implementation, is amenable to the same formal proof techniques as other formal approaches. Moreover, since the constructive aspects of the OBJ specification are machine-executable, it is straightforward to achieve system validation from the specification itself. Execution is essentially the evaluation of a function by equation substitution (re-writing) which continues until specific output value is obtained. This executability allows end-users of the envisaged system to gain a "view" of the eventual system at the system specification stage without the need to be familiar with the underlying formal specification techniques.

As with all other ADT techniques, OBJ is only applicable to sequential systems, or to sequential aspects of concurrent systems. OBJ has been widely used for the specification of both small and large-scale industrial applications.

References:

An introduction to OBJ; A language for Writing and Testing Specifications. J A Goguen and JTardo, Specification of Reliable Software, IEEE Press 1979, reprinted in Software Specification Techniques, N Gehani, A McGettrick (eds), Addison-Wesley, 1985.

Algebraic Specification for Practical Software Production. C Rattray, Cogan Press, 1987.

An Algebraic Approach to the Standardisation and Certification of Graphics Software. R Gnatz, Computer Graphics Forum 2(2/3), 1983.

DTI STARTS Guide. 1987, NCC, Oxford Road, Manchester, UK.

B.30.6 Temporal Logic

Aim

Direct expression of safety and operational requirements and formal demonstration that these properties are preserved in the subsequent development steps.

Description

Standard First Order Predicate Logic contains no concept of time. Temporal logic extends First Order logic by adding modal operators (e.g. "Henceforth" and "Eventually"). These operators can be used to qualify assertions about the system. For example, safety properties might be required to hold "henceforth", whilst other desired system states might be required to be attained "eventually" from some other initiating

astratti (ADT). Un ADT è come un pacchetto ADA dove il comportamento dell'operatore è visibile, mentre i dettagli di implementazione sono "nascosti".

Una specifica OBJ, e la successiva implementazione per raffinamenti, è riconducibile alle medesime tecniche di prova formale come altri approcci formali. Inoltre, dal momento che gli aspetti costruttivi della specifica OBJ sono eseguibili dalla macchina, la validazione del sistema è ottenuta direttamente dalla specifica stessa. L'esecuzione è essenzialmente la valutazione di una funzione per mezzo di sostituzione dell'equazione (riscrittura) che continua fino a quando è ottenuto un valore di uscita specifico. Questa eseguibilità consente all'utente finale del sistema previsto di ottenere una "panoramica" del sistema definitivo nella fase di specifica del sistema senza la necessità di avere familiarità con le tecniche di specifica formali sottostanti.

Come con tutte le altre tecniche ADT, OBJ è applicabile solo a sistemi sequenziali, o ad aspetti sequenziali di sistemi concorrenti. OBJ è stato ampiamente usato per la specifica di applicazioni industriali sia in scala ridotta che ampia.

Riferimenti:

An introduction to OBJ; A language for Writing and Testing Specifications. J A Goguen and JTardo, Specification of Reliable Software, IEEE Press 1979, reprinted in Software Specification Techniques, N Gehani, A McGettrick (eds), Addison-Wesley, 1985.

Algebraic Specification for Practical Software Production. C Rattray, Cogan Press, 1987.

An Algebraic Approach to the Standardisation and Certification of Graphics Software. R Gnatz, Computer Graphics Forum 2(2/3), 1983.

DTI STARTS Guide. 1987, NCC, Oxford Road, Manchester, UK.

B.30.6 Logica Temporale

Scopo

Espressione diretta di requisiti di sicurezza e operativi e dimostrazione formale che queste proprietà sono conservate nei passaggi di sviluppo successivi.

Descrizione

Lo Standard First Order Predicate Logic non contiene alcun concetto di tempo. La logica temporale estende la logica First Order aggiungendo operatori modali (ad esempio. "Henceforth" (d'ora innanzi) e "Eventually" (in fine)). Questi operatori possono essere usati per qualificare asserzioni inerenti il sistema. Per esempio, proprietà di sicurezza potrebbero essere richieste per sostenere "henceforth", mentre altri stati del sistema desiderato potreb-

state. Temporal formulas are interpreted on sequences of states (behaviours). What constitutes a "state" depends on the chosen level of description. It can refer to the whole system, a system component or the computer program. Quantified time intervals and constraints are not handled explicitly in temporal logic. Absolute timing has to be handled by creating additional time states as part of the state definition.

bero essere richiesti per ottenere "eventually" da qualche altro inizio di stato. Formule temporali sono interpretate su sequenze di stati (comportamenti). Quello che costituisce uno "stato" dipende dal livello di descrizione scelto. Esso può riferirsi all'intero sistema, ad una componente del sistema o al programma del computer. Nella logica temporale intervalli di tempo quantificati e vincoli non sono trattati espicitamente. La temporarizzazione assoluta deve essere trattata con la creazione di stati di tempo addizionali come parte della definizione dello stato.

References:

Temporal Logic of Programs. F Kroger EATCS Monographs on Computer Science, Vol 8, Springer Verlag, 1987.

Design for Safety using Temporal Logic. J Gorski. SAFECOMP 86, Sarlat France, Pergamon Press, October 1986.

Logics for Computer Programming. D Gabay. Ellis Horwood.

Riferimenti:

Temporal Logic of Programs. F Kroger EATCS Monographs on Computer Science, Vol 8, Springer Verlag, 1987.

Design for Safety using Temporal Logic. J Gorski. SAFECOMP 86, Sarlat France, Pergamon Press, October 1986.

Logics for Computer Programming. D Gabay. Ellis Horwood.

B.30.7 VDM - Vienna Development Method

VDM – Metodo di Sviluppo Vienna

Aim

The systematic specification and implementation of sequential programs.

Scopo

La specifica e la implementazione sistematica di programmi sequenziali.

Description

VDM is a mathematically based specification technique and a technique for refining implementations in a way that allows proof of their correctness with respect to the specification.

The specification technique is model-based in that the system state is modelled in terms of set-theoretic structures on which are defined invariants (predicates), and operations on that state are modelled by specifying their pre and post-conditions in terms of the system state. Operations can be proved to preserve the system invariants.

The implementation of the specification is done by the reification of the system state in terms of data structures in the target language and by refinement of the operations in terms of a program in the target language. Reification and refinement steps give rise to proof obligations that establish their correctness. Whether or not these obligations are carried out is a choice made by the designer.

VDM is principally used in the specification stage but can be used in the design and implementation stages leading to source code. It can only be applied to sequential programs or the sequential processes in concurrent systems.

Descrizione

Il VDM è una tecnica di specificazione su base matematica ed una tecnica di implementazioni per raffinamento in un modo che consente la prova della loro correttezza nei confronti della specifica.

La tecnica di specificazione è basata su modello, nel senso che lo stato del sistema è modellato in termini di strutture di insieme teoriche sulle quali sono definite invarianti (predicati), e operazioni su quello stato sono modellate specificando le loro pre e post condizioni in termini di stato del sistema. Le operazioni possono essere provate per preservare le invarianti del sistema.

L'implementazione della specifica è eseguita mediante la conversione dello stato del sistema in termini di struttura dei dati nel linguaggio previsto e per raffinamento delle operazioni nei termini di un programma nel linguaggio previsto (target). Gli stadi di conversione e di affinamento suscitano l'obbligo di prove che stabiliscano la loro correttezza. È una scelta del progettista quella di eseguire o meno tali obblighi.

Il VDM è principalmente usato nella fase di specificazione, ma può essere usato nelle fasi di progettazione e di implementazione conducendo al codice sorgente. Può essere applicato solo a programmi sequenziali o a processi sequenziali in sistemi concorrenti.

References:

Software Development - A Rigorous Approach. C B Jones. Prentice-Hall, 1980.

Formal Specification and Software Development. D Bjorner & C B Jones. Prentice-Hall, 1982.

Systematic Software Development using VDM. C B Jones. Prentice-Hall, 1986.

The Specification of Complex Systems. B Cohen, W T Harwood and M I Jackson. Addison-Wesley, 1986.

Riferimenti:

Software Development - A Rigorous Approach. C B Jones. Prentice-Hall, 1980.

Formal Specification and Software Development. D Bjorner & C B Jones. Prentice-Hall, 1982.

Systematic Software Development using VDM. C B Jones. Prentice-Hall, 1986.

The Specification of Complex Systems. B Cohen, W T Harwood and M I Jackson. Addison-Wesley, 1986.

B.30.8 Z and B

Aim

Z is a specification language notation for sequential systems and a design technique that allows the developer to proceed from a Z specification to executable algorithms in a way that allows proof of their correctness with respect to the specification.

Z is principally used in the specification stage but a method has been devised to go from specification into a design and an implementation. It is best suited to the development of data oriented, sequential systems.

B is an associated method.

Description

Like VDM, the specification technique is model-based in that the system state is modelled in terms of set-theoretic structures on which are defined invariants (predicates), and operations on that state are modelled by specifying their pre and post-conditions in terms of the system state. Operations can be proved to preserve the system invariants thereby demonstrating their consistency. The formal part of a specification is divided into schemas which allow the structuring of specifications through refinement.

Typically, a Z specification is a mixture of formal Z and informal explanatory text in natural language. (Formal text on its own can be too terse for easy reading and often its purpose needs to be explained, while the informal natural language can easily become vague and imprecise).

Unlike VDM, Z is a notation rather than a complete method. However an associated method (called B) has been developed which can be used in conjunction with Z. The B method is based on the principle of step-wise refinement

Z e B

Scopo

Il metodo Z è un'annotazione di linguaggio di specificazione per sistemi sequenziali ed una tecnica di progettazione che consente allo sviluppatore di procedere da una specifica Z ad algoritmi eseguibili in modo da consentire la prova della loro correttezza nei confronti della specifica.

Z è usato principalmente nella fase di specificazione, ma è stato escogitato un metodo per passare dalla specifica fino a una progettazione ed un'implementazione. Esso è più adatto allo sviluppo di sistemi sequenziali ed orientati ai dati.

Il B è un metodo associato.

Descrizione

Come VDM, la tecnica di specifica è basata su modello, nel senso che lo stato del sistema è modellato in termini di strutture di insieme teoriche sulle quali sono definite invarianti (predicati), e le operazioni su questi stati sono modellate specificando le loro pre e post condizioni in termini di stato del sistema. Le operazioni possono essere provate per preservare le invarianti del sistema dimostrando così la loro coerenza. La parte formale della specifica è suddivisa in schemi che consentono la strutturazione delle specifiche tramite affinamento.

Tipicamente, una specifica di Z è un misto di Z formale e di testo esplicativo informale in linguaggio naturale. (Il testo formale, per se stesso può essere troppo conciso per una facile lettura e spesso è necessario che sia spiegato il suo fine, mentre il linguaggio naturale informale può diventare facilmente vago e impreciso).

A differenza di VDM, Z è un'annotazione piuttosto che un metodo completo. Tuttavia un metodo associato (chiamato B) è stato sviluppato in modo da poterlo usare in associazione con Z. Il metodo B è basato sul principio del procedimento per affinamenti successivi.

References:

The Z Notation - A Reference Manual. J M Spivey, Prentice Hall, 1988.

Specification Case Studies. Edited by I Hayes, Prentice-Hall, 1987.

Specification of the UNIX Filestore. C Morgan and B Sufrin. IEEE Transactions on Software Engineering, SE-10, 2 March 1984.

The B Book - Assigning Programs to Meanings. J R Abrial, Cambridge United Press, 1996.

Riferimenti:

The Z Notation - A Reference Manual. J M Spivey, Prentice Hall, 1988.

Specification Case Studies. Edited by I Hayes, Prentice-Hall, 1987.

Specification of the UNIX Filestore. C Morgan and B Sufrin. IEEE Transactions on Software Engineering, SE-10, 2 March 1984.

The B Book - Assigning Programs to Meanings. J R Abrial, Cambridge United Press, 1996.

B.31 Formal Proof (Referenced by clause 11)

Prova Formale (Riferimento all'art. 11)

Aim

Using theoretical and mathematical models and rules it is possible to prove the correctness of a program without executing it.

Description

A number of assertions are stated at various locations in the program, and they are used as pre and post conditions to various paths in the program. The proof consists of showing that the program transfers the preconditions into the post-conditions according to a set of logical rules, and that the program terminates.

Several Formal Methods are described in this bibliography, for instance, CCS, CSP, HOL, LOTOS, OBJ, Temporal Logic, VDM and Z. The descriptions of these can be found under the section "Formal Methods".

References:

Can Program Proving be made Practical. J. Dahl, Research Report, ISBN 82-90230-26-5 No33, Oslo, May

Scopo

Usando regole e modelli teorici e matematici è possibile provare la correttezza di un programma senza eseguirlo.

Descrizione

Sono stabilite una serie di asserzioni in varie posizioni nel programma, ed esse vengono usate come pre e post condizioni per vari percorsi nel programma. La prova consiste nel mostrare che il programma trasferisce le precondizioni nelle post-condizioni secondo un'insieme di regole logiche, e che il programma termina.

Nella bibliografia sono descritti vari Metodi Formali, per esempio, CCS, CSP, HOL, LOTOS, OBJ, Logica di Tempo, VDM e Z. La loro descrizione può essere trovata nella sezione "Metodi Formali".

Riferimenti:

Can Program Proving be made Practical. J. Dahl, Research Report, ISBN 82-90230-26-5 No33, Oslo, May

B.32 Forward Recovery (Referenced by clause 9)

Recupero in avanti (Forward Recovery) (Riferimento all'art. 9)

Aim

To provide correct functional operation in the presence of one or more faults.

Description

If a fault has been detected, the current state of the system is manipulated to obtain a state, which will be consistent some time later. This concept is especially suited for real-time systems with a small database and fast rate of change of the internal state. It is assumed, that at least part of the system state may be imposed onto the environment, and only part of the system states are influenced (forced) by the environment.

Scopo

Fornire attività funzionale corretta in presenza di uno o più malfunzionamenti.

Descrizione

Se è stato rilevato un malfunzionamento, lo stato corrente del sistema è manipolato per ottenere uno stato, che sarà coerente nel seguito. Questo concetto è particolarmente adatto per sistemi in tempo reale con una piccola base dati ed un elevato tasso di modifiche dello stato interno. Si assume che almeno parte dello stato del sistema può essere imposto sull'ambiente, e solo parte degli stati del sistema sono influenzati (forzati) dall'ambiente.

B.33 Graceful Degradation

Aim

To maintain the more critical system functions available despite failures by dropping the less critical functions.

Description

This technique gives priorities to the various functions to be carried out by the system. The design then ensures that should there be insufficient resources to carry out all the system functions, then the higher priority functions are carried out in preference to the lower ones. For example, error and event logging functions may be lower priority than system control functions. System control would continue if the hardware associated with error logging were to fail. Further, should the system control hardware fail, but not the error logging hardware, then the error logging hardware would take over the control function. This is predominantly applied to hardware but is applicable to the total system. It must betaken into account from the top-most design phase.

References:

Space Shuttle Software. C.T. Sheridan, Datamation, Vol 24, July 1978

The Evolution of Fault-Tolerant Computing. Vol 1 of Dependable Computing and Fault Tolerant Systems, Edited by A. Avizienis, H. Kopetz, and J.C. Laprie, Springer-Verlag, ISBN 3-211-81941-X, 1987.

Fault Tolerance, Principle and Practices. Vol 3 of [2] above, T. Anderson and P.A. Lee, Springer-Verlag, ISBN 3-211-82077-9, 1987.

Degrado Soffice

Scopo

Mantenere disponibili le funzioni di sistema più critiche nonostante i mancati funzionamenti, con caduta delle funzioni meno critiche.

Descrizione

Questa tecnica dà priorità a varie funzioni da effettuare da parte del sistema. La progettazione pertanto assicura che se ci fossero risorse insufficienti a svolgere tutte le funzioni del sistema, allora le funzioni con più alta priorità sono svolte preferendole a quelle con più bassa priorità. Per esempio, funzioni di registrazione di errori ed eventi possono avere priorità inferiore a quelle delle funzioni di controllo del sistema. Il controllo del sistema continuerebbe se l'hardware associato con la registrazione di errori stesse per andare in avaria. Inoltre, se andasse in avaria l'hardware del controllo di sistema, ma non l'hardware di registrazione degli errori, allora l'hardware della registrazione degli errori subentrerebbe nella funzione di controllo. Questo è prevalentemente applicato all'hardware, ma è applicabile all'intero sistema. Di esso si deve tenere conto dalla fase più in alto della progettazione.

Riferimenti:

Space Shuttle Software. C.T. Sheridan, Datamation, Vol 24, July 1978

The Evolution of Fault-Tolerant Computing. Vol 1 of Dependable Computing and Fault Tolerant Systems, Edited by A. Avizienis, H. Kopetz, and J.C. Laprie, Springer-Verlag, ISBN 3-211-81941-X, 1987.

Fault Tolerance, Principle and Practices. Vol 3 of [2] above, T. Anderson and P.A. Lee, Springer-Verlag, ISBN 3-211-82077-9, 1987.

B.34 Hazard and Operability Study (HAZOP)

Aim

To establish, by a series of systematic examinations of the component sections of the computer system and its operation, failure modes which lead to result in potentially hazardous situations in the controlled system.

Typical hazardous events on the controlled systems are fire, explosion, release of toxic material (chemical or nuclear) or serious economic loss.

It is assumed that the hazardous events on the system being controlled have been identified in a separate Hazard Analysis and the consequence of the hazardous events classified into degrees of seriousness.

The analysis which covers all stages of the project life-cycle, from specification through design to maintenance and modification, and is intended to identify at each stage events/failure

Studi del Pericolo e di Funzionalità (HAZOP)

Scopo

Stabilire, con una serie di esami sistematici delle sezioni componenti del sistema computerizzato e della sua funzionalità, i modi di mancato funzionamento che portano al risultato in situazioni potenzialmente pericolose nel sistema controllato.

Eventi tipici pericolosi nei sistemi controllati sono il fuoco, l'esplosione, l'emissione di materiali tossici (chimici o nucleari) o danni economici importanti.

Si assume che gli eventi pericolosi nel sistema da controllare siano stati identificati con una individuale Analisi dei Pericoli e che le conseguenze degli eventi pericolosi siano state classificate secondo gradi di severità.

L'analisi copre tutti le fasi del ciclo di vita del progetto, dalla specifica, attraverso la progettazione, alla manutenzione e alla modifica, ed è previsto che identifichi per ogni fase eventi/modi di man-

modes which could lead to potential hazards and thus eliminate them.

Description

The analysis is carried out by a team of engineers (covering computer, instrument, electrical, process, safety and operational disciplines) led by a trained specialist in hazard analysis techniques in a series of scheduled meetings.

It is important that a fixed time schedule is allocated within the project for the meetings; each one scheduled for at least half a day and for effectiveness no more than four per week to allow for maintaining the flow of accompanying documentation.

Prior to the study, agreed checklists for the systematic examination will have been compiled and for each section of the system leading questions such as; What if it happens? How can it happen? When can it happen? Does it matter? are asked. When positive answers are given further questions are asked, such as; What can be done about it? When must it be done? Is there an alternative? etc.

For the first part of the analysis it is suggested that the computer installation as a whole is examined. The second and subsequent parts involve detailed examination of the relevant parts of the computer systems itself.

At each part or stage of the analysis, the objective is to identify failure modes which lead to potential hazards on the controlled system and the degree of their effect. The major component parts of the computer system are further sub-divided and where necessary subjected to a separate analysis.

At suitable points throughout the systematic analysis, action review meetings will be arranged.

It is essential that comprehensive records of the meetings are kept, for they will form a substantial part of the system hazard/safety dossier.

After a number of meetings it is suggested that a review meeting be held to ensure that actions are followed up, modifications suggested during the Study meetings are incorporated into the study etc.

References:

HAZOP and HAZAN. T.A. Kletz, 1986, 2nd Edition, Institution of Chemical Engineers, 165-171 Railway Terrace, Rugby, CV1 3HQ, UK.

Reliability and Hazard Criteria for Programmable Electronic Systems in the Chemical Industry. E. Johnson, Proc of "Safety and Reliability of PES", PES 3 Safety Symposium, B.K. Daniels

cato funzionamento che potrebbero condurre a potenziali pericoli e di conseguenza eliminarli.

Descrizione

L'analisi è eseguita da un gruppo di ingegneri (coprendo discipline informatiche, strumentali, elettriche, del processo, la sicurezza e operative) guidata da uno specialista istruito nelle tecniche dell'analisi del pericolo in una serie di incontri programmati.

È importante che sia assegnato nell'ambito del progetto un programma di incontri con tempi stabiliti; ciascuno programmato per almeno mezza giornata e non più di quattro per settimana per assicurare l'efficienza e per permettere di garantire il flusso della documentazione associata.

Prima dello studio, devono essere compilate liste di controllo concordate per un esame sistematico e per ogni parte del sistema devono essere poste domande di rilievo quali; Che cosa può accadere? Come può accadere se? Quando può accadere? È di interesse?. Quando sono date risposte positive vanno poste ulteriori domande quali: Che cosa può essere fatto in merito? Quando deve essere fatto? C'è una alternativa? ecc.

Per la prima parte dell'analisi si suggerisce che sia esaminata l'installazione del computer nel suo complesso. La seconda parte e le successive coinvolgono l'esame dettagliato delle parti di rilievo degli stessi sistemi computerizzati.

In ogni parte o stadio dell'analisi, l'obbiettivo è quello di identificare i modi di mancato funzionamento che conducono a pericoli potenziali sul sistema controllato ed il grado del loro effetto. Le principali parti componenti del sistema computerizzato sono ulteriormente suddivise e, ove necessario, soggette ad analisi separata.

In punti adatti per tutta l'analisi sistematica devono essere organizzate riunioni di revisione delle azioni.

È essenziale che siano tenute registrazioni esaurienti delle riunioni, poiché esse costituiranno una parte sostanziale del dossier pericolo/sicurezza del sistema.

Dopo un certo numero di riunioni si suggerisce di tenere una riunione di revisione per assicurare che sia stato dato seguito alle azioni, all'inserimento delle modifiche suggerite durante le riunioni di studio, ecc.

Riferimenti:

HAZOP and HAZAN. T.A. Kletz, 1986, 2nd Edition, Institution of Chemical Engineers, 165-171 Railway Terrace, Rugby, CV1 3HQ, UK.

Reliability and Hazard Criteria for Programmable Electronic Systems in the Chemical Industry. E. Johnson, Proc of "Safety and Reliability of PES", PES 3 Safety Symposium, B.K. Daniels (ed), 28-30

(ed), 28-30 May 1986, Guernsey Channel Islands, Elsevier Applied Science, 1986.

Reliability Engineering and Risk Assessment. E.J. Henlty and H. Kumamoto, Prentice Hall,1981.

Systems Reliability and Risk Analysis. E.G. Frenkel, Martinus Nijhogg 1984.

B.35 Impact Analysis (Referenced by clause 16)

Aim

To identify the effect that a change or an enhancement to a software system will have to other modules in that software system as well as to other systems.

Description

Prior to a modification or enhancement being performed on the software an analysis shall be undertaken to identify the impact of the modification or enhancement on the software and to also identify the affected software systems and modules.

After the analysis has been completed a decision is required concerning the reverification of the software system. This depends on the number of modules affected, the criticality of the affected modules and the nature of the change. The possible decisions are:

i) only the changed module to be reverified;
ii) all identified affected modules are reverified; and
iii) the complete system is reverified.

B.36 Information Hiding / Encapsulation (Referenced by D.9)

Aim

To increase the reliability and maintainability of software.

Description

Data that is globally accessible to all software components can be accidentally or incorrectly modified by any of these components. Any changes to these data structures may require detailed examination of the code and extensive modifications.

Information hiding is a general approach for minimising these difficulties. The key data structures are “hidden” and can only be manipulated through a defined set of access procedures. This allows the internal structures to be modified or further procedures to be added without affecting the functional behaviour of the remaining software. For example, a name directory might have access procedures Insert, Delete and Find. The access procedures and internal data structures could be re-written (e.g. to use a

May 1986, Guernsey Channel Islands, Elsevier Applied Science, 1986.

Reliability Engineering and Risk Assessment. E.J. Henlty and H. Kumamoto, Prentice Hall, 1981.

Systems Reliability and Risk Analysis. E.G. Frenkel, Martinus Nijhogg 1984.

Analisi dell’Impatto (Riferimento all’art. 16)

Scopo

Identificare l’effetto che una modifica od un miglioramento ad un sistema software avrà su altri moduli in quel sistema software, così come su altri sistemi.

Descrizione

Prima che sia introdotta una modifica od un miglioramento nel software, deve essere eseguita una verifica per identificare l’impatto della modifica o del miglioramento sul software ed anche per identificare i sistemi software ed i moduli interessati dal cambiamento.

Dopo il completamento dell’analisi è richiesta una decisione circa la riverifica del sistema software. Questa dipende dal numero dei moduli interessati dal cambiamento, la criticità dei moduli interessati e la natura del cambiamento. Le decisioni possibili sono:

i) solo i moduli modificati sono riverificati;
ii) tutti i moduli interessati dal cambiamento sono riverificati; e
iii) il completo sistema è riverificato.

Occultamento/Incapsulamento delle informazioni (Riferimento a D.9)

Scopo

Aumentare l’affidabilità e la manutenibilità del software.

Descrizione

I dati che sono globalmente accessibili a tutte le componenti del software possono essere modificati accidentalmente o in modo scorretto da qualunque di queste componenti. Ogni cambiamento a queste strutture di dati può richiedere un esame dettagliato del codice e delle modifiche estese.

L’Occultamento dell’informazione è un approccio generale per minimizzare queste difficoltà. Le strutture dei dati chiave sono “occultate” e possono essere manipolate solo attraverso un’insieme definito di procedure di accesso. Questo consente che la struttura interna sia modificata o di aggiungere ulteriori procedure senza influenzare il comportamento funzionale del software rimanente. Per esempio, una directory stabilita potrebbe avere procedure di accesso Inserisci, Cancella e Trova. Le procedure di accesso e le strutture dei dati

different look-up method or to store the names on a hard disk) without affecting the logical behaviour of the remaining software using these procedures.

This concept of an abstract data type is directly supported in a number of programming languages, but the basic principle can be applied whatever programming language is used.

References:

Software Engineering: Planning for Change, D A Lamb, Prentice Hall, 1988.

On the Design and Development of Program Families, D L Parnas, IEEE Trans SE-2, Mar 1976.

interni potrebbero essere riscritte (ad esempio per usare un metodo diverso di ricerca o per immagazzinare i nomi su un disco rigido) senza influenzare il comportamento logico del software rimanente usando queste procedure.

Questo concetto di tipo di dato astratto è supportato direttamente in molti linguaggi di programmazione, ma il principio di base può essere applicato a qualsiasi linguaggio di programmazione usato.

Riferimenti:

Software Engineering: Planning for Change, D A Lamb, Prentice Hall, 1988.

On the Design and Development of Program Families, D L Parnas, IEEE Trans SE-2, Mar 1976.

B.37 Interface Testing (Referenced by clause 10)

Aim

To demonstrate that interfaces of subprograms do not contain any errors or any errors that lead to failures in a particular application of the software or to detect all errors that may be relevant.

Description

Several levels of detail or completeness of testing are feasible. The most important levels are testing:

- all interface variables at their extreme positions;
- all interface variable individually at their extreme values with other interface variables at normal values;
- all values of the domain of each interface variable with other interface variables at normal values;
- all values of all variables in combination. this may only be feasible for small interfaces;
- the specified test conditions relevant to each call of each subroutine.

These tests are particularly important if their interfaces do not contain assertions that detect incorrect parameter values. They are also important after new configurations of pre-existing subprograms have been generated.

Prove sulle Interfacce (Riferimento all'art. 10)

Scopo

Dimostrare che le interfacce di sottoprogrammi non contengono alcun errore o alcun errore che conduca a mancati funzionamenti in una particolare applicazione del software o rilevare tutti gli errori che possono essere pertinenti.

Descrizione

Sono praticabili vari livelli di dettaglio o di completezza di prova. I più importanti livelli di prova sono:

- tutte le variabili di interfaccia nei loro stati estremi;
- individualmente tutte le variabili di interfaccia ai loro valori estremi con altre variabili di interfaccia al valore normale;
- tutti i valori del dominio di ogni variabile di interfaccia con altre variabili di interfaccia al valore normale;
- tutti i valori di tutte le variabili in combinazione. Questo può essere fattibile solo per piccole interfacce;
- le condizioni di prova specificate relative ad ogni chiamata di ogni subroutine.

Queste prove sono particolarmente importanti se le loro interfacce non contengono asserzioni che rilevano valori di parametri scorretti. Esse sono importanti anche dopo che sono state generate nuove configurazioni di sottoprogrammi preesistenti.

B.38 Language Subset (Referenced by clause 10 and D.4)

Aim

To reduce the probability of introducing programming faults and increase the probability of detecting any remaining faults.

Sottoinsieme del Linguaggio (Riferimento all'art. 10 e D.4)

Scopo

Ridurre la probabilità di introdurre malfunzionamenti di programmazione ed aumentare la probabilità di rilevare ogni restante malfunzionamento.

Description

The language is examined to identify programming constructs which are either error-prone or difficult to analyse, for example, using static analysis methods. A language subset is then defined which excludes these constructs.

Descrizione

Il linguaggio è esaminato per identificare i costrutti di programmazione che sono, o inclini all'errore, o difficili da esaminare, per esempio, usando metodi di analisi statica. Un sottoinsieme del linguaggio è quindi definito per escludere questi costrutti.

B.39 Memorising Executed Cases (Referenced by clause 9)

Memorizzazione dei Casi Eseguiti (Riferimento all'art. 9)

Aim

To force the software to fail safe if it executes an unlicensed path.

Description

During licensing a record is made of all relevant details of each program execution. During normal operation each program execution is compared with the set of the licensed executions. If it differs, a safety action is taken.

The execution record can be the sequence of the individual decision-to-decision paths (DD-paths) or the sequence of the individual accesses to arrays, records or volumes, or both.

Different methods of storing execution paths are possible. Nash-coding methods can be used to map the execution sequence onto a single large number or sequence of numbers. During normal operation the execution path value must be checked against the stored cases before any output operation occurs.

Since the possible combinations of decision-to-decision paths during one program is very large, it may not be feasible to treat programs as a whole. In this case, the technique can be applied at module level.

Reference:

Fail-safe Software - Some Principles and a Case Study. W. Ehrenberger, Proc. SARSS 1987,Altringham, Manchester, UK, Elsevier Applied Science, 1987.

Scopo

Forzare il software nella condizione di fail safe se esegue un percorso non consentito.

Descrizione

Durante l'autorizzazione**,** un record è costituito da tutti i dettagli relativi ad ogni esecuzione di programma. Durante il normale funzionamento ciascuna esecuzione di programma è confrontata con l'insieme di esecuzioni autorizzate**.** Se differisce, è intrapresa un' azione in sicurezza

Il record di esecuzione può essere la sequenza di percorsi individuali, decisione per decisione, (DDpaths) o la sequenza di accessi individuali ad una schiera (arrays), record o volumi o entrambi.

Sono possibili diversi metodi di memorizzazione dei percorsi di esecuzione. Il metodo di codifica Nash può essere usato per mappare la sequenza di esecuzione su un unico grande numero o su una sequenza di numeri. Durante il normale funzionamento il valore del percorso di esecuzione deve essere confrontato con i casi memorizzati prima che si verifichi alcuna operazione di uscita.

Poiché le possibili combinazioni di percorsi, decisione per decisione, nel corso di un programma è molto grande, può non essere fattibile trattare i programmi nella loro interezza. In questo caso, la tecnica può essere applicata a livello di modulo.

Riferimento:

Fail-safe Software - Some Principles and a Case Study. W. Ehrenberger, Proc. SARSS 1987, Altringham, Manchester, UK, Elsevier Applied Science, 1987.

B.40 Library of Trusted/Verified Modules and Components (Ref'd by clause 10)

Libreria di Moduli e Componenti Affidabili/Verificati (Riferimento all'art. 10)

Aim

To avoid the need for software modules and hardware component designs to be extensively revalidated or redesigned for each new application. Also to advantage designs which have not been formally or rigorously validated but for which considerable operational history is available.

Scopo

Evitare che sia ampiamente rivalidata o riprogettata la progettazione di moduli software e di componenti hardware per ogni nuova applicazione. Anche per avvantaggiare le progettazioni che non sono state validate formalmente o rigorosamente, ma per le quali sono disponibili considerevoli esperienze di funzionamento.

Description

Well designed and structured PESs are made up of a number of hardware and software components and modules which are clearly distinct and which interact with each other in clearly defined ways.

Different PESs designed for differing applications will contain a number of modules or components which are the same or very similar. Building up a library of such generally applicable modules allows much of the resource necessary for validating the designs to be shared by more than one application.

Furthermore the use of such modules in multiple applications provides empirical evidence of successful operational use. This empirical evidence justifiably enhances the trust which users are likely to have in the modules.

Descrizione

PES ben progettati e strutturati sono costituiti da molti componenti hardware e software e moduli chiaramente distinti e che interagiscono tra loro in modi chiaramente definiti.

Differenti PES progettati per diverse applicazioni conterranno un numero di moduli o componenti che sono identici o molto simili. Costruire una libreria di tali moduli generalmente applicabili, permette di condividere su più di una applicazione le risorse necessarie a validare le progettazioni.

Inoltre, l'uso di tali moduli in più applicazioni fornisce l'esperienza empirica di un uso con funzionamento di successo. Questa evidenza empirica legittimamente migliora la fiducia che gli utenti avranno probabilmente nei moduli.

B.41 Markov Models (Referenced by clause 14)

Modelli di Markov (Riferimento all'art. 14)

Aim

To evaluate the reliability, safety or availability of a system.

Description

A graph of the system is constructed. The graph represents the status of the system with regard to its failure states (the failure states are represented by the nodes of the graph). The edges between nodes, which represent the failure events or repair events, are weighted with the corresponding failure rates or repair rates. Note that the failure events, states and rates can be detailed in such a way that a precise description of the system is obtained, e.g. detected or undetected failures, manifestation of a larger failure etc.

The Markov technique is suitable for modelling redundant systems in which the level of redundancy varies with time due to component failure and repair. Other classical methods, for example, FMEA and FTA, cannot readily be adapted to modelling the effects of failures throughout the life-cycle of the system since no simple combinatorial formulae exist for calculating the corresponding probabilities.

In the simplest cases, the formulae which describe the probabilities of the system are readily available in the literature or can be calculated manually. In more complex cases, some methods of simplification (absorbing states) exist. For very complex cases results can be calculated by computer simulation of the graph.

Scopo

Valutare l'affidabilità, la sicurezza o la disponibilità di un sistema.

Descrizione

È costruito un grafico del sistema. Il grafico rappresenta lo stato del sistema con riguardo ai suoi stati di mancato funzionamento (gli stati di mancato funzionamento sono rappresentati dai nodi del grafico). I margini tra i nodi, che rappresentano gli eventi di mancato funzionamento o di riparazione, sono pesati con i corrispondenti tassi di mancato funzionamento e di riparazione. Da notare che eventi di mancato funzionamento, stati e tassi possono essere dettagliati in modo tale da ottenere una precisa descrizione del sistema, ad esempio, mancati funzionamenti rilevati e non rilevati, manifestazione di un mancato funzionamento più grande ecc.

La tecnica di Markov è adatta alla modellazione di sistemi ridondanti nei quali il livello di ridondanza varia nel tempo a seguito di mancato funzionamento e riparazione del componente. Altri metodi classici, per esempio FMEA e FTA, non possono essere adattati prontamente per modellare gli effetti dei mancati funzionamenti in ogni momento del ciclo di vita del sistema dato che non esiste alcuna formula combinatoria semplice per il calcolo delle corrispondenti probabilità.

Nei casi più semplici, le formule che descrivono le probabilità del sistema sono già disponibili nella letteratura o, possono essere calcolate manualmente. In casi più complessi, esistono alcuni metodi di semplificazione (stati di assorbimento). Per casi molto complessi i risultati possono esser calcolati con la simulazione al computer del grafico.

References:

The Theory of Stochastic Processes. R.E. Cox and H.D. Miller, Methuen and Co. Ltd,London, UK, 1968

Finite MARKOV Chains. J.G. Kemeny and J.L. Snell, D. Van Nostrand Company Inc.,Princeton, 1959

Reliability Handbook. B.A. Koslov and L.A. Ushakov, Holt Rinehart and Winston Inc.,New York 1970

The Theory and Practice of Reliable System Design. D.P. Siewiorek and R.S. Swarz, Digital Press 1982.

Riferimenti:

The Theory of Stochastic Processes. R.E. Cox and H.D. Miller, Methuen and Co. Ltd,London, UK, 1968

Finite MARKOV Chains. J.G. Kemeny and J.L. Snell, D. Van Nostrand Company Inc., Princeton, 1959

Reliability Handbook. B.A. Koslov and L.A. Ushakov, Holt Rinehart and Winston Inc., New York 1970

The Theory and Practice of Reliable System Design. D.P. Siewiorek and R.S. Swarz, Digital Press 1982.

B.42 Metrics (Referenced by clause 11 and clause 14)

Metriche (Riferimento all'art. 11 e 14)

Aim

To predict the attributes of programs from properties of the software itself rather than from its development or test history.

Description

These models evaluate some structural properties of the software and relate this to a desired attribute such as reliability or complexity. Software tools are required to evaluate most of the measures. Some of the metrics which can be applied are given below:

- Graph Theoretic Complexity: this measure can be applied early in the lifecycle to assess trade-offs, and is based on the complexity of the program control graph, represented by its cyclomatic number;
- number of ways to activate a certain module (accessibility): the more a module can be accessed, the more likely it is to be debugged;
- Software Science: this measure computes the program length by counting the number of operators and operands. It provides a measure of complexity and estimates development resources;
- number of entries and exits per module: minimising the number of entry/exit points is a key feature of structured design and programming techniques.

References:

A Complexity Measure. T.J. McCabe, IEEE Trans. on Software Engineering, Vol. SE-2, No. 4, Dec 1976.

Models and Measurements for Quality Assessments of Software. S.N. Mohanty, ACMComputing Surveys, Vol 11, No. 3, Sep 1979.

Elements of Software Science. M.H. Halstead, Elsevier, North Holland, New York, 1977.

Scopo

Predire gli attributi dei programmi dalle proprietà del software stesso piuttosto che dal suo sviluppo o dalla storia delle sue prove.

Descrizione

Questi modelli valutano alcune proprietà strutturali del software e mettono in relazione questo con un attributo desiderato quale affidabilità o complessità. Strumenti software sono richiesti per valutare la maggior parte delle misure. Alcune delle metriche che possono essere applicate sono stabilite qui sotto:

- complessità Teorica del Grafico (Graph Theoretic Complexity): questa misura può essere applicata all'inizio del ciclo di vita per valutare gli scambi, ed è basata sulla complessità del grafico di controllo del programma, rappresentata dal suo numero ciclomatico**;**
- numero di modi per attivare un certo modulo (accessibilità): più un modulo può essere accessibile, più è probabile che sia sottoponibile a debugging
- Scienza del Software: questa misura calcola la lunghezza del programma contando il numero degli operatori e degli operandi. Essa fornisce una misura di complessità e stima le risorse di sviluppo;
- numero di entrate ed uscite per modulo: minimizzare il numero dei punti di entrata/uscita è una caratteristica chiave della progettazione strutturata e delle tecniche di programmazione.

Riferimenti:

A Complexity Measure. T.J. McCabe, IEEE Trans. on Software Engineering, Vol. SE-2, No. 4, Dec 1976.

Models and Measurements for Quality Assessments of Software. S.N. Mohanty, ACMComputing Surveys, Vol 11, No. 3, Sep 1979.

Elements of Software Science. M.H. Halstead, Elsevier, North Holland, New York, 1977.

B.43 Modular Approach (Referenced by D.9)

Aim

Decomposition of a software systems into small comprehensible parts in order to limit the complexity of the system.

Description

A Modular Approach or modularisation contains several rules for the design, coding and maintenance phases of a software project. These rules vary according to the design method employed during design. Most methods contain the following rules:

- a module shall have a single well defined task or function to fulfil;
- connections between modules shall be limited and strictly defined, coherence in one module shall be strong;
- collections of subprograms shall be built providing several levels of modules;
- subprogram sizes shall be restricted to some specified value typically 2 to 4 screen sizes;
- subprograms shall have a single entry and a single exit only;
- modules shall communicate with other modules via their interfaces. Where global or common variables are used they shall be well structured, access shall be controlled and their use shall be justified in each instance;
- all module interfaces shall be fully documented;
- each module shall hide something from its environment;
- any modules interface shall contain the minimum number of parameters necessary for the modules function; and
- a suitable restriction of parameter number shall be specified, typically 5.

Approccio Modulare (Riferimento a D.9)

Scopo

Decomposizione di sistemi software in piccole parti comprensibili al fine di limitare la complessità del sistema.

Descrizione

Un Approccio Modulare o modularizzazione contiene svariate regole per le fasi di progettazione, codifica e manutenzione di un progetto software. Queste regole variano secondo il metodo di progettazione impiegato durante lo sviluppo. La maggior parte dei metodi contiene le seguenti regole:

- un modulo deve avere un' unico e ben definito compito o funzione da soddisfare;
- il collegamento tra moduli deve essere limitato e strettamente definito, la coerenza in un modulo deve essere elevata;
- la raccolta di sottoprogrammi deve essere costruita prevedendo molti livelli di moduli;
- la dimensione dei sottoprogrammi deve essere ristretta ad alcuni valori specificati tipicamente da 2 a 4 dimensioni dello schermo;
- i sottoprogrammi devono avere un'entrata sola e una sola uscita;
- i moduli devono comunicare con altri moduli tramite le loro interfacce. Ove sono usate variabili globali o comuni essi devono essere ben strutturati, l'accesso deve essere controllato e il loro uso deve essere giustificato in ogni situazione;
- tutte le interfacce dei moduli devono essere completamente documentate;
- ogni modulo deve nascondere qualcosa del suo ambiente;
- ogni interfaccia dei moduli deve contenere il minor numero di parametri necessari per la funzione dei moduli; e
- un'idonea restrizione del numero dei parametri deve essere specificata, tipicamente 5.

B.44 Monte-Carlo Simulation

Aim

To simulate phenomenon of the real world in the software using random numbers.

Description

Monte-Carlo simulations are used to solve problems. These problems fall into two classes:

- probabilistic, where random numbers are used to generate a real world stochastic phenomenon; and
- deterministic, which are mathematically translated into an equivalent probabilistic problem.

Monte-Carlo simulation injects a random number streams to simulate noise on an analytic

Simulazione Monte-Carlo

Scopo

Simulare fenomeni del mondo reale nel software usando numeri casuali.

Descrizione

Le simulazioni Monte-Carlo sono usate per risolvere problemi. Questi problemi ricadono in due classi:

- probabilistici, ove i numeri casuali sono usati per generare fenomeni stocastici del mondo reale; e
- deterministici, che sono trasformati matematicamente in un problema probabilistico equivalente.

La simulazione Monte-Carlo inserisce una serie di numeri casuali per simulare disturbi in un segnale

signal or to add random biases or tolerances. The Monte-Carlo simulation is run to produce a large sample from which statistical results are obtained.

When using Monte-Carlo simulations care must be taken to ensure that the biases, tolerances or noise are reasonable values.

A general principle of Monte-Carlo simulations is to restate and reformulate the problem so that the results obtained are as accurate as possible rather than tackling the problem as initially stated.

analitico o aggiungere polarizzazioni casuali o tolleranze. La simulazione di Monte-Carlo è attuata per produrre un ampio campione dal quale sono ottenuti risultati statistici.

Quando si usano simulazioni Monte-Carlo si deve aver cura di assicurare che le polarizzazioni, le tolleranze i disturbi abbiano valori ragionevoli.

Un principio generale nelle simulazioni Monte-Carlo è quello di riesporre e formulare nuovamente il problema in modo che i risultati ottenuti siano i più accurati possibili piuttosto che affrontare il problema come stabilito all'inizio.

B.45 Performance Modelling (Referenced by D.2 and D.5)

Aim

To ensure that the working capacity of the system is sufficient to meet the specified requirements.

Description

The requirements specification includes throughput and response requirements for specific functions, perhaps combined with constraints on the use of total system resources. The proposed system design is compared against the stated requirements by

- defining a model of the system processes, and their interactions,
- identifying the use of resources by each process, for example, processor time, communications bandwidth, storage devices etc),
- identifying the distribution of demands placed upon the system under average and worst-case conditions,
- computing the mean and worst-case throughput and response times for the individual system functions.

For simple systems, an analytic solution may be possible whilst for more complex systems, some form of simulation is required to obtain accurate results.

Before detailed modelling, a simpler "resource budget" check can be used which sums the resources requirements of all the processes. If the requirements exceed designed system capacity, the design is infeasible. Even if the design passes this check, performance modelling may show that excessive delays and response times occur due to resource starvation. To avoid this situation engineers often design systems to use some fraction (e.g. 50 %) of the total resources so that the probability of resource starvation is reduced.

Modellazione delle prestazioni (Riferimento a D.2 e D.5)

Scopo

Assicurare che la capacità di lavoro del sistema sia sufficiente a conseguire i requisiti specificati.

Descrizione

La specifica dei requisiti comprende requisiti di trattamento e di risposta per funzioni specifiche, combinate magari con vincoli nell'uso delle risorse totali del sistema. La progettazione del sistema proposto è confrontata con i requisiti stabiliti da

- definizione di un modello dei processi del sistema e delle loro interazioni,
- identificazione dell'uso di risorse da parte di ogni processo, per esempio, tempo del processore, larghezza di banda della comunicazione, dispositivi di memorizzazione, ecc.,
- identificazione della distribuzione delle richieste poste al sistema nelle condizioni medie e nei casi peggiori,
- calcolo della capacità di trattamento nel caso peggiore e dei tempi di risposta per le funzioni singole del sistema.

Per sistemi semplici, può essere possibile una soluzione analitica, mentre per sistemi più complessi, è richiesta qualche forma di simulazione per ottenere risultati accurati.

Prima della modellazione dettagliata, può essere usata una verifica più semplice di "budget delle risorse" che somma i requisiti delle risorse di tutti i processi. Se i requisiti eccedono la capacità del sistema progettato, la progettazione non è realizzabile. Anche se la progettazione supera questa verifica, la modellazione delle prestazioni può mostrare che si verificano eccessivi ritardi e che occorrono tempi di risposta dovuti alla ristrettezza di risorse. Per evitare questa situazione, gli ingegneri spesso progettano i sistemi usando alcune frazioni delle risorse totali (ad esempio il 50%) in modo da ridurre la probabilità di ristrettezza di risorse.

B.46 Performance Requirements (Referenced by D.6)

Aim

To establish that the performance requirements of a software system have been satisfied.

Description

An analysis is performed of both the system and the software requirements specifications to identify all general and specific, explicit and implicit performance requirements.

Each performance requirement is examined in turn to determine

- the success criteria to be obtained,
- whether a measure against the success criteria can be obtained,
- the potential accuracy of such measurements,
- the project stages at which the measurements can be estimated, and
- the project stages at which the measurements can be made.

The practicability of each performance requirement is then analysed in order to obtain a list of performance requirements, success criteria and potential measurements. The main objectives are;

i) each performance requirement is associated with at least one measurement;
ii) where possible, accurate and efficient measurements are selected which can be used as early in the development process as possible;
iii) essential and optional performance requirements and success criteria are identified and
iv) where possible, advantage shall be taken of the possibility of using a single measurement for more than one performance requirement.

B.46 Requisiti di Prestazione (Riferimento a D.6)

Scopo

Stabilire che i requisiti di prestazione di un sistema software sono stati soddisfatti.

Descrizione

Viene eseguita un'analisi della specifica dei requisiti sia del software che del sistema per identificare tutti i requisiti di prestazione, generali e specifici, espliciti ed impliciti.

Ciascun requisito di prestazione è esaminato a turno per determinare

- i criteri del successo da ottenere,
- se può essere ottenuta una misura dei criteri di successo,
- la precisione potenziale di tali misurazioni,
- le fasi della progettazione nelle quali possono essere valutate le misurazioni, e
- le fasi della progettazione nelle quali possono essere fatte le misurazioni.

L'attuabilità di ogni requisito di prestazione è quindi analizzata al fine di ricavare un elenco di requisiti di prestazione, criteri di successo e misurazioni potenziali. Gli obbiettivi principali sono;

i) ogni requisito di prestazione è associato con almeno una misurazione;
ii) ove possibile, sono scelte misurazioni accurate ed efficienti che possano essere usate, nei limiti del possibile, agli inizi dello sviluppo del progetto;
iii) sono identificati requisiti di prestazione essenziali ed opzionali e criteri di successo, e
iv) ove possibile, devono essere perseguiti i vantaggi della possibilità di usare una singola misurazione per più di un requisito di prestazione.

B.47 Probabilistic Testing (Referenced by clause 11 and clause 13)

Aim

To gain a quantitative figure about the reliability properties of the investigated software. This figure may address the related levels of confidence and significance and

i) a failure probability per demand,
ii) a failure probability during a certain period of time, and
iii) a probability of error containment.

From these figures other parameters may be derived such as

- probability of failure free execution,
- probability of survival,
- availability,
- MTBF or failure rate, and
- probability of safe execution.

B.47 Prove Probabilistiche (Riferimento all'art. 11 e 13)

Scopo

Ottenere un dato quantitativo circa le proprietà di affidabilità del software in osservazione. Questo dato può indirizzarsi ai livelli correlati di fiducia e significatività e

i) una probabilità di mancato funzionamento per richiesta,
ii) una probabilità di mancato funzionamento durante un certo periodo di tempo, e
iii) una probabilità di limitazione dell'errore.

Da questi dati possono essere derivati altri parametri quali

- probabilità di esecuzione priva di mancati funzionamenti,
- probabilità di sopravvivenza,
- disponibilità,
- MTBF o tasso di mancato funzionamento, e
- probabilità di esecuzione sicura.

Description

Probabilistic considerations are either based on a probabilistic test or on operating experience. Usually the number of tests cases of observed operating cases is very large.

In order to facilitate testing, usually automatic aids are taken. They concern the details of test data provision and test output supervision. Large tests are run on large host computers with the appropriate process simulation periphery. Test data is selected both according to systematic and random view points. The first concern the overall test control, for example, guarantee a test data profile. The random selection takes the individual test cases in detail.

Individual test harnesses, test executions and test supervisions are determined by the detailed test aims as described above. Other important conditions are given through the mathematical prerequisites to be fulfilled in order to enable the test evaluation in view of the intended test aim.

Probabilistic figures about the behaviour of any test object may also be derived from operating experience. Provided the same conditions are met, the same mathematics can be applied as for the evaluation of test results.

Descrizione

Le considerazioni probabilistiche sono basate o su una prova probabilistica o su un'esperienza operativa. Normalmente il numero dei casi di prova di casi operativi osservati è molto elevato.

Al fine di facilitare le prove, si adottano normalmente ausili automatici. Essi riguardano i dettagli dell'acquisizione dei dati di prova e della supervisione del dato in uscita dalla prova. Prove estese sono eseguite su grossi computer con opportune periferiche di simulazione del processo. I dati di prova sono scelti sia secondo il punto di vista sistematico che casuale. Il primo riguarda il controllo della prova in generale, per esempio, garantire un profilo dei dati di prova. La scelta casuale considera nel dettaglio i casi di prova individuali.

L'impostazione delle prove individuali, l'esecuzione delle prove e la loro supervisione sono determinate dagli scopi delle prove di dettaglio come sopra descritto. Altre importanti condizioni sono date tramite i prerequisiti matematici da soddisfare al fine di consentire la valutazione della prova in vista dello scopo della prova previsto.

Caratteristiche probabilistiche circa il comportamento di ogni oggetto della prova può essere derivato anche dall'esperienza operativa. Purché siano ottenute le medesime condizioni, gli stessi algoritmi matematici possono essere applicati per la valutazione dei risultati di prova.

B.48 Process Simulation (Referenced by D.3)

Simulazione di Processo (Riferimento a D.3)

Aim

To test the function of a software system, together with its interface to the outside world, without allowing it to modify the real world in any way.

Description

The creation of a system, for testing purposes only, which mimics the behaviour of the system to be controlled by the system under test.

The simulation may be software only or a combination of software and hardware. It must

- provide all the inputs of the system under test which will exist when the system is installed,
- respond to outputs from the system in a way which faithfully represents the controlled plant,
- have provision for operator inputs to provide any perturbations with which the system under test is required to cope.

When software is being tested the simulation may be a simulation of the target hardware with its inputs and outputs.

Scopo

Provare le funzioni di un sistema software, con le sue interfacce con il mondo esterno, senza permettergli in alcun modo di modificare il mondo reale.

Descrizione

La creazione di un sistema, per soli intendimenti di prova, che imita il comportamento del sistema da controllare mediante il sistema in prova.

La simulazione può essere solo software o una combinazione di software e hardware. Essa deve

- fornire tutti gli ingressi di dati del sistema in prova che esisteranno quando il sistema è installato
- rispondere alle uscite dal sistema in modo che rappresentino fedelmente l'impianto controllato,
- avere cura che gli ingressi di dati da parte dell'operatore non producano perturbazioni al quale il sistema in prova è richiesto che faccia fronte

Quando il software viene provato la simulazione può essere una simulazione dell'hardware previsto con i suoi ingressi e le sue uscite.

References:

A Software Simulator - An Aid to Plant Commissioning. S. Nunns, EWICS TC7.

Physical Fault Simulation. F. Morillon, EWICS Document number WP460.

Riferimenti:

A Software Simulator - An Aid to Plant Commissioning. S. Nunns, EWICS TC7.

Physical Fault Simulation. F. Morillon, EWICS Document number WP460.

B.49 Prototyping/Animation (Referenced by D.3 and D.5)

Aim

To check the feasibility of implementing the system against the given constraints. To communicate the specifier's interpretation of the system to the customer, in order to locate misunderstandings.

Description

A sub-set of system functions, constraints, and performance requirements are selected. A prototype is built using high level tools. At this stage, constraints such as the target computer, implementation language, program size, maintainability, reliability and availability need not be considered. The prototype is evaluated against the customer's criteria and the system requirements may be modified in the light of this evaluation.

References:

Proc. Working Conference on Prototyping. Namur Oct 1983, Budde et al, Springer-verlag, 1984.

Using an executable specification language for an information system. S. Urban et. al, IEEE Trans Software Engineering, Vol. SE-11 No. 7 July 1985.

Prototipazione/Animazione (Riferimento a D.3 e D.5)

Scopo

Verificare la fattibilità di implementare il sistema in base ai vincoli dati. Comunicare l'interpretazione di chi specifica il sistema al cliente, al fine di individuare malintesi.

Descrizione

Vengono scelti un sottoinsieme di funzioni del sistema, vincoli e requisiti di prestazione. È costruito un prototipo usando strumenti di elevato livello. In questa fase, vincoli come il computer stabilito, il linguaggio di implementazione, la dimensione del programma, la manutenibilità, l'affidabilità e la disponibilità non è necessario che vengano considerate. Il prototipo è valutato in base ai criteri del cliente e i requisiti del sistema possono essere modificati alla luce di questa valutazione.

Riferimenti:

Proc. Working Conference on Prototyping. Namur Oct 1983, Budde et al, Springer-verlag, 1984.

Using an executable specification language for an information system. S. Urban et. al, IEEE Trans Software Engineering, Vol. SE-11 No. 7 July 1985.

B.50 Recovery Block (Referenced by clause 9)

Aim

To increase the likelihood of the program performing its intended function.

Description

Several different program sections are written, often independently, each of which is intended to perform the same desired function. The final program is constructed from these sections. The first section, called the primary, is executed first. This is followed by an acceptance test of the result it calculates. If the test is passed then the result is accepted and passed on to subsequent parts of the system. If it fails, any side effects of the first are reset and the second section, called the first alternative, is executed. This too is followed by an acceptance test and is treated as in the first case. A second, third or even more alternatives can be provided if desired.

Blocco di Recupero (Riferimento all'art. 9)

Scopo

Aumentare la probabilità che il programma esegua le sue funzioni previste.

Descrizione

Vengono scritte molte sezioni di programma differenti, spesso in modo indipendente, ciascuna di queste è inteso realizzi la medesima funzione desiderata. Il programma finale è costruito in base a queste sezioni. La prima sezione, chiamata primaria, è eseguita per prima. Questa è seguita da una prova di accettazione dei risultati che esse calcola. Se la prova è superata il risultato è accettato e si passa alla parte successiva del sistema. Se la prova va male, alcuni effetti collaterali della prima sono ripristinati e la seconda sezione, chiamata la prima alternativa, è eseguita. Anche questa è seguita da una prova di accettazione ed è trattata come nel primo caso. Se desiderato possono essere previste una seconda, una terza o anche più alternative.

Reference:

System Structure for Software Fault Tolerance. B. Randall, IEEE Trans Software Engineering, Vol SE-1, No 2, 1975.

Riferimento:

System Structure for Software Fault Tolerance. B. Randall, IEEE Trans Software Engineering, Vol SE-1, No 2, 1975.

B.51 Reliability Block Diagram (Referenced by clause 14)

Aim

To model, in a diagrammatic form, the set of events that must take place and conditions which must be fulfilled for a successful operation of a system or a task.

Description

The target of the analysis is represented as a success path consisting of blocks, lines and logical junctions. A success path starts from one side of the diagram and continues via the blocks and junctions to the other side of the diagram. A block represents a condition or an event, and the path can pass it if the condition is true or the event has taken place. If the path comes to a junction it continues if the logic of the junction is fulfilled. If it reaches a vertex, it may continue along all outgoing lines. If there exists at least one success path through the diagram the target of the analysis is operating correctly.

References:

System Reliability Engineering Methodology: A Division of the State of the Art. J.B. Fusseland J.S. Arend, Nuclear Safety 20(5), 1979.

Fault Tree Handbook. W.E. Vesely et al, NUREG-0942, Division of System Safety Office at Nuclear Reactor Regulation, U.S. Nuclear Regulatory Commission, Washington, D.C. 20555, 1981.

Diagramma a Blocchi dell'Affidabilità (Riferimento all'art. 14)

Scopo

Modellare, in forma di diagramma, l'insieme di eventi che deve aver luogo e di condizioni che devono essere soddisfatte per il riuscito funzionamento di un sistema o un processo.

Descrizione

L'obbiettivo dell'analisi è rappresentata come un percorso che abbia successo consistente in blocchi, linee e giunzioni logiche. Un percorso riuscito parte da un lato del diagramma e continua lungo i blocchi e le giunzioni fino all'altro lato del diagramma. Un blocco rappresenta una condizione od un evento, ed il percorso è accettato se la condizione è vera o se l'evento ha avuto luogo. Se il percorso arriva alla giunzione, esso continua se la logica della giunzione è soddisfatta. Se esso raggiunge un vertice, esso può continuare lungo tutte le linee uscenti. Se esiste almeno un percorso nel diagramma che abbia successo quanto stabilito dall'analisi è correttamente funzionante

Riferimenti:

System Reliability Engineering Methodology: A Division of the State of the Art. J.B. Fusseland J.S. Arend, Nuclear Safety 20(5), 1979.

Fault Tree Handbook. W.E. Vesely et al, NUREG-0942, Division of System Safety Office at Nuclear Reactor Regulation, U.S. Nuclear Regulatory Commission, Washington, D.C. 20555, 1981.

B.52 Response Timing and Memory Constraints (Referenced by D.6)

Aim

To ensure that the system will meets its temporal and memory requirements.

Description

The requirements specification for the system and the software includes memory and response requirements for specific functions, perhaps combined with constraints on the use of total system resources. An analysis is performed which will identify the distribution demands under average and worst case conditions. This analysis requires estimates of the resource usage and elapsed time of each system function. These estimates can be obtained in several ways, for example, comparison with an existing

Tempo di Risposta e Vincoli di Memoria (Riferimento a D.6)

Scopo

Assicurare che il sistema soddisfi i suoi requisiti di tempo e memoria.

Descrizione

La specifica dei requisiti del sistema e del software comprende i requisiti di memoria e di risposta per le funzioni specifiche, eventualmente combinate con i limiti nell'uso delle risorse totali del sistema. È eseguita un'analisi che identificherà le richieste di distribuzione nelle condizioni medie e nel caso peggiore. Questa analisi richiede stime dell'uso delle risorse e del tempo trascorso per ciascuna funzione del sistema. Queste stime possono essere ottenute in molti modi, per esempio, per comparazione con un sistema esistente o per

system or the prototyping and bench-marking of time critical systems.

prototipazione e per prestazione comparativa di sistemi con criticità di tempo.

B.53 Re-Try Fault Recovery Mechanisms (Referenced by clause 9)

Aim

To attempt functional recovery from a detected fault condition by re-try mechanisms.

Description

In the event of a detected fault or error condition, attempts are made to recover the situation by re-executing the same code. Recovery by re-try can be as complete as a re-boot and a re-start procedure or a small re-scheduling and re-starting task, after a software time-out or a task watchdog action. Re-try techniques are commonly used in communication fault or error recovery and re-try conditions could be flagged from a communication protocol error (check sum etc.) or from a communication acknowledgement response time-out.

Reference:

The Theory and Practise of Reliable System Design. D.P. Siewiorek and R.S. Schwarz, Digital Press.

Meccanismi di Ripetizione per il Recupero del Guasto (Riferimento all'art. 9)

Scopo

Tentare il recupero della funzionalità da una condizione di rilevamento di malfunzionamento con meccanismi di ripetizione.

Descrizione

Nell'evenienza di un rilevamento di guasto o condizione di errore, vengono fatti tentativi per recuperare la situazione rassegnando il medesimo codice. Il ricupero per ripetizione può essere completo quanto le procedure di "re-boot" e quelle di riavviamento oppure con un processo ridotto di riprogrammazione e riavviamento, dopo un'azione di sospensione del software o un'azione di un processo watchdog. Le tecniche di ripetizione sono comunemente usate nei guasti di comunicazione o recupero di errori e le condizioni di ripetizione potrebbero essere segnalate da un errore di protocollo di comunicazione (check sum, ecc.) o da una interruzione di una risposta di riconoscimento della comunicazione.

Riferimento:

The Theory and Practise of Reliable System Design. D.P. Siewiorek and R.S. Schwarz, Digital Press.

B.54 Safety Bag (Referenced by clause 9)

Aim

To protect against residual specification and implementation faults in software which adversely affect safety.

Description

A safety bag is an external monitor, implemented on an independent computer to a different specification. This safety bag is solely concerned with ensuring the main computer performs safe, not necessarily correct, actions. The safety bag continuously monitors the main computer. The safety bag prevents the system from entering an unsafe state. In addition if it detects that the main computer is entering a potentially hazardous state, the system has to be brought back to a safe state either by the safety bag or the main computer.

References:

Using AI Techniques to Improve Software Safety. Proc. IFAC SAFECOMP 88, Sarlat, France, Oct 1986, Pergamon Press 1986.

Tecnica della Valigetta di Sicurezza (Safety Bag) (Riferimento all'art. 9)

Scopo

Proteggere da guasti residui di specificazione ed implementazione nel software che influenzano negativamente la sicurezza.

Descrizione

La Safety Bag è un monitor esterno, implementato in un computer indipendente con una specifica diversa. Questa Safety Bag è dedicata solamente ad assicurare che il computer principale si comporti in modo sicuro, non necessariamente con azioni corrette. La Safety Bag sorveglia continuamente il computer principale. La Safety Bag impedisce al sistema di entrare in uno stato insicuro. Inoltre se rileva che il computer principale sta entrando in uno stato potenzialmente pericoloso, il sistema deve essere riportato ad uno stato sicuro o dalla Safety Bag o dal computer principale.

Riferimenti:

Using AI Tecniche to Improve Software Safety. Proc. IFAC SAFECOMP 88, Sarlat, France, Oct 1986, Pergamon Press 1986.

B.55 Sneak Circuit Analysis (Referenced by D.8)

Aim

To detect an unexpected path or logic flow within a system which, under certain conditions initiates an undesired function or inhibits a desired function.

Description

A sneak circuit path may consist of hardware, software, operator actions, or combinations of these elements. Sneak circuits are not the result of hardware failure but are latent conditions inadvertently designed into the system or coded into the software programs, which can cause it to malfunction under certain conditions.

Categories of sneak circuits are:

- Sneak Paths which cause current, energy, or logical sequence to flow along an unexpected path or in an unintended direction;
- Sneak Timing in which events occur in an unexpected or conflicting sequence;
- Sneak Indications which cause an ambiguous or false display of system operating conditions, and thus may result in an undesired action by the operator;
- Sneak Labels which incorrectly or imprecisely label system functions, for example, system inputs, controls, displays, buses, etc., and thus may mislead an operator into applying an incorrect stimulus to the system.

Sneak circuit analysis, relies on the recognition of basic topological patterns with the hardware or software structure (e.g. six basic patterns are identified for software). Analysis takes place with the aid of a checklist of questions about the use and relationships between the basic topological components.

References:

Sneak Analysis and Software Sneak Analysis. S.G. Godoy and G.J. Engels, J. Aircraft Vol. 15, No. 8, 1978.

Sneak Circuit Analysis. J.P. Rankin, Nuclear Safety, Vol. 14, No. 5, 1973.

Analisi del Circuito Ingannevole (Riferimento a D.8)

Scopo

Rilevare un percorso inatteso o un flusso logico dentro un sistema che, in certe condizioni inizia una funzione indesiderata o inibisce una funzione desiderata.

Descrizione

Il percorso di un circuito ingannevole può consistere in elementi hardware, software, azioni dell'operatore, o in una combinazione di questi elementi. I circuiti ingannevoli non sono il risultato di mancato funzionamento dell'hardware, ma sono condizioni latenti progettate inavvertitamente nel sistema o codificate nei programmi software, che possono causare malfunzionamento in certe condizioni.

Categorie di circuiti ingannevolio sono:

- Percorsi ingannevoli che causano il flusso di corrente, energia o sequenza logica lungo un percorso inatteso o in una direzione non prevista;
- Tempistica ingannevole nella quale si verificano eventi in una sequenza inattesa o conflittuale;
- Indicazioni ingannevoli che causano una rappresentazione ambigua o falsa delle condizioni operative del sistema, e pertanto ne può derivare una azione indesiderata dell'operatore;
- Identificazione ingannevole che identifica in modo scorretto o impreciso funzioni del sistema, per esempio, ingressi nel sistema, controlli, visualizzazioni, bus, ecc., che pertanto possono fuorviare un operatore nell'applicare uno stimolo non corretto al sistema.

L'analisi dei circuiti ingannevoli, conta sul riconoscimento di modelli topologici di base della struttura hardware o software (ad esempio per il software sono identificati sei modelli di base). L'analisi ha luogo con l'aiuto di una lista di controllo di domande circa l'uso e le relazioni tra i componenti topologici di base.

Riferimenti:

Sneak Analysis and Software Sneak Analysis. S.G. Godoy and G.J. Engels, J. Aircraft Vol. 15, No. 8, 1978.

Sneak Circuit Analysis. J.P. Rankin, Nuclear Safety, Vol. 14, No. 5, 1973.

B.56 Software Configuration Management (Referenced by clause 15)

Aim

Software Configuration management aims to ensure the consistency of groups of development deliverables as those deliverables change. Configuration Management, in general, applies to both hardware and software development.

Gestione della Configurazione del Software (Riferimento all'art. 15)

Scopo

La Gestione della Configurazione Software ha lo scopo di assicurare la coerenza di gruppi di elementi dichiarati quando questi cambiano. La gestione della Configurazione, in generale si applica allo sviluppo sia dell'hardware che del software.

Description

Software Configuration Management is a technique used throughout development. In essence, it requires the recording of the production of every version of every "significant" deliverable and of every relationship between different versions of the different deliverables. The resulting records allow the developer to determine the effect on other deliverables of a change to one deliverable (especially on of its components). In particular, systems or subsystems can be reliably re-built from consistent sets of component versions.

References:

Configuration Management Practices for Systems, Equipment, Munitions and Computer Programs. MIL-STD-483.

Software Configuration Management. J K Buckle, Macmillan Press, 1982.

Software Configuration Management. W A Babich, Addison-Wesley, 1986.

Configuration Management Requirements for Defence Equipment. UK Ministry of Defence Standard 05-57 Issue 1, 1980.

Descrizione

La Gestione della Configurazione Software è una tecnica usata in ogni momento dello sviluppo. In sostanza, essa richiede la registrazione della produzione di ogni versione di ogni elemento dichiarato "significativo" e di ogni relazione tra diverse versioni di elementi dichiarati differenti. I record risultanti consentono allo sviluppatore di determinare l'effetto su altri elementi dichiarati di un cambiamento ad un elemento dichiarato (specialmente sui suoi componenti). In particolare, sistemi o sottosistemi possono essere ricostruiti in modo affidabile da insiemi coerenti di versioni di componenti.

Riferimenti:

Configuration Management Practices for Systems, Equipment, Munitions and Computer Programs. MIL-STD-483.

Software Configuration Management. J K Buckle, Macmillan Press, 1982.

Software Configuration Management. W A Babich, Addison-Wesley, 1986.

Configuration Management Requirements for Defence Equipment. UK Ministry of Defence Standard 05-57 Issue 1, 1980.

B.57 Strongly Typed Programming Languages (Referenced by clause 10)

Aim

Reduce the probability of faults by using a language which permits a high level of checking by the compiler.

Description

Such languages usually allow user-defined data types to be defined from the basic language data types (such as INTEGER, REAL). These types can then be used in exactly the same way as the basic types, but strict checks are imposed to ensure the correct type is used. These checks are imposed over the whole program, even if this is built from separately compiled units. The checks also ensure that the number and the type of procedure arguments match even when referenced from separately compiled modules.

Strongly typed languages also support other aspects of good software engineering practice such as easily analysable control structures (e.g. IF .. THEN .. ELSE, DO .. WHILE, etc) which lead to well-structured programs.

Typical examples of strongly typed languages are Pascal, Ada and Modula 2.

Linguaggi di Programmazione Fortemente Tipizzati (Riferimento all'art. 10)

Scopo

Ridurre la probabilità di guasti usando un linguaggio che permette un elevato livello di verifica da parte del compilatore.

Descrizione

Tali linguaggi consentono normalmente tipi di dati definiti dall'utente che devono essere definiti dai tipi di dati del linguaggio di base (quali INTEGER, REAL). Questi tipi possono quindi essere usati esattamente nello stesso modo dei tipi di base, ma sono imposte verifiche strette per assicurare che sia usato il tipo corretto. Queste verifiche sono imposte su tutto il programma, anche se è costruito partendo da unità compilate separatamente. Le verifiche assicurano anche che il numero e il tipo di argomenti della procedura si accordino anche quando riferiti a moduli compilati separatamente.

I linguaggi fortemente tipizzati sono di supporto anche ad altri aspetti di buona pratica dell'ingegneria del software quali strutture di controllo facilmente analizzabili (ad esempio IF .. THEN .. ELSE, DO .. WHILE, ecc.) che conducono a programmi ben strutturati.

Tipici esempi di linguaggi tipizzati fortemente sono Pascal, Ada e Modula 2.

References:

Reference Manual for the Ada Programming Language, ANSI/MIL-STD-815A 1983.

In search of Effective Diversity: a Six Language Study of Fault-Tolerant Flight Control Software, A Avizienis, M R Lyu, W Schutz, The Eighteenth International Symposium on Fault Tolerant Computing, Tokyo, Japan 27-30 June 1988, IEEE Computer Society Press,ISBN 0-8186-0867-6.

Riferimenti:

Reference Manual for the Ada Programming Language, ANSI/MIL-STD-815A 1983.

n search of Effective Diversity: a Six Language Study of Fault-Tolerant Flight Control Software, A Avizienis, M R Lyu, W Schutz, The Eighteenth International Symposium on Fault Tolerant Computing, Tokyo, Japan 27-30 June 1988, IEEE Computer Society Press,ISBN 0-8186-0867-6

B.58 Structure Based Testing (Referenced by D.2)

Aim

To apply tests which exercise certain subsets of the program structure.

Description

Based on an analysis of the program a set of input data is chosen such that a large fraction of selected program elements are exercised. The program elements exercised can vary depending upon the level of rigour required.

- Statements: this is the least rigorous test since it is possible to execute all code statements without exercising both branches of a conditional statement.
- Branches: both sides of every branch should be checked. This may be impractical for some types of defensive code.
- Compound Conditions: every condition in a compound conditional branch (i.e. linked by AND/OR is exercised).
- LCSAJ: a linear code sequence and jump is any linear sequence of code statements including conditional jumps terminated by a jump. Many potential sub-paths will be infeasible due to constraints on the input data imposed by the execution of earlier code.
- Data Flow: the execution paths are selected on the basis of data usage for example a path where the same variable is both written and wrote.
- Call Graph: a program is composed of subroutines which may be invoked from other subroutines. The call graph is the tree of subroutine invocations in the program. Tests are designed to cover all invocations in the tree.
- Entire Path: execute all possible path through the code. Complete testing is normally infeasible due to the vary large number of potential paths.

References:

Reliability of the Path Analysis Testing Strategy. W. Howden, IEEE Trans Software Engineering, Vol SE 3, 1976.

Prove Basate sulla Struttura (Riferimento a D.2)

Scopo

Applicare prove che esercitano alcuni sottoinsiemi della struttura del programma.

Descrizione

È scelta una serie di dati d'ingresso sulla base dell'analisi del programma in modo che sia sollecitata un'ampia frazione di elementi del programma scelto. Gli elementi del programma esercitato possono variare secondo il livello di rigore richiesto.

- Dichiarazioni: questa è la prova rigorosa minima poiché è possibile eseguire tutti i comandi del codice senza l'attivazione di entrambe le ramificazioni di un comando condizionato.
- Ramificazioni: entrambi i lati di ogni ramificazione dovrebbero essere provati. Questo potrebbe essere non praticabile per alcuni tipi di codice difensivo.
- Condizioni Composte: ogni condizione in una ramificazione condizionata composta (ad esempio collegata da AND/OR è esercitata).
- LCSAJ: una sequenza di codice lineare e salto è ogni sequenza lineare di comandi del codice compresi salti condizionali terminati da un salto. Molti sottopercorsi potenziali non saranno realizzabili per i vincoli sui dati d'ingresso imposti dall'esecuzione del codice precedentemente.
- Flusso dei Dati: i percorsi di esecuzione sono scelti in base all'uso dei dati, per esempio un percorso ove la stessa variabile richiami sia written che wrote.
- Call Graph: un programma è composto da subroutine che possono essere chiamate da altre subroutine. Il call graph è l'albero delle chiamate delle subroutine nel programma. Le prove sono progettate per trattare tutte le chiamate nell'albero.
- Percorso Totale: eseguire tutti i possibili percorsi attraverso il codice. Prove complete non sono normalmente attuabili per il numero molto grande di percorsi potenziali.

Riferimenti:

Reliability of the Path Analysis Testing Strategy. W. Howden, IEEE Trans Software Engineering, Vol SE 3, 1976.

B.59 Structure Diagrams (Referenced by D.5)

Aim

To show the structure of a program diagrammatically.

Description

Structure Diagrams are a notation which complements Data Flow Diagrams. They describe the programming system and a hierarchy of parts and display this graphically, as a tree. They document how elements of a data flow diagram can be implemented as a hierarchy of program units.

A structure chart shows relationships between program units without including any information about the order of activation of these units. They are drawn using the following three symbols:

i) a rectangle annotated with the name of the unit;

ii) an arrow connecting these rectangles;

iii) a circled arrow, annotated with the name of data passed to and from elements in the structure chart. Normally, the circled arrow is drawn parallel to the arrow connecting the rectangles in the chart.

From any non trivial data flow diagram, it is possible to derive a number of different structure charts.

Structure charts derived from data flow diagrams represent a first level structure of the system, where each box on the structure chart represents a bubble in the data flow diagram. Naturally, deeper levels can be described using the same technique.

References:

Software Engineering. I. Sommerville, Addison-Wesley 1982. ISBN 0-201-13795-X.

Structured Design. L.L Constantine and E, Yourdon, Englewood Cliffs, New Jersey, Prentice Hall, 1979.

Reliable Software Through Composite Design. G.J Myers, New York, Van Nostrand, 1975.

B.60 Structured Methodology (Referenced by clause 8 and clause 10)

Aim

The main aim of Structured Methodologies is to promote the quality of software development by focusing attention on the early parts of the life-cycle. The methods aim to achieve this through both precise and intuitive procedures and notations (assisted by computers) to identify the existence of requirements and implementation features in a logical order and a structured manner.

Diagrammi Strutturati (Riferimento a D.5)

Scopo

Mostrare la struttura di un programma mediante diagramma.

Descrizione

I Diagrammi Strutturati sono un'annotazione che integra i Diagrammi di Flusso dei Dati. Essi descrivono il sistema di programmazione ed una gerarchia delle parti e le rappresentano graficamente come un albero. Essi documentano come elementi di un diagramma del flusso dei dati possono essere implementati come una gerarchia delle unità di programma.

Una carta strutturata mostra le relazioni tra le unità di programma senza inserire alcuna informazione sull'ordine di attivazione di queste unità. Esse sono disegnate usando i tre simboli seguenti:

i) un rettangolo annotato con il nome dell'unità;

ii) una freccia che congiunge questi rettangoli;

iii) una freccia circolare, annotata con il nome del dato passato agli e dagli elementi nella carta strutturata. Normalmente la freccia circolare è disegnata parallela alla freccia che congiunge i rettangoli nella carta.

Da ogni diagramma di flusso dei dati non privo di significato, è possibile derivare molte carte di struttura diverse tra loro.

Carte di struttura derivate da diagrammi di flusso di dati rappresentano una struttura di primo livello del sistema, ove ogni rettangolo sulla carta della struttura rappresenta un pallogramma nel diagramma del flusso dei dati. Naturalmente, livelli più profondi possono essere descritti usando la medesima tecnica.

Riferimenti:

Software Engineering. I. Sommerville, Addison-Wesley 1982. ISBN 0-201-13795-X.

Structured Design. L.L Constantine and E, Yourdon, Englewood Cliffs, New Jersey, Prentice Hall, 1979.

Reliable Software Through Composite Design. G.J Myers, New York, Van Nostrand, 1975.

Metodologia Strutturata (Riferimento all'art. 8 e 10)

Scopo

Lo scopo principale della Metodologia Strutturata è quello di promuovere la qualità dello sviluppo del software focalizzando l'attenzione sulle prime fasi del ciclo di vita. I metodi mirano a raggiungere questo attraverso procedure precise, intuitive ed annotazioni (assistite dai computer) per identificare l'esistenza di requisiti e caratteristiche di implementazione in un ordine logico ed in modo strutturato.

Description

A range of Structured Methodologies exist. Some such as SSADM, LBMS are designed for traditional data-processing and transaction processing functions, while others (MASCOT, JSD, real-time Yourdon) are more oriented to process-control and real-time applications (which tend to be more safety-critical).

Structured Methods are essentially "thought tools" for systematically perceiving and partitioning a problem or system. Their main features are:

- a logical order of thought, breaking a large problem into manageable stages;
- identification of total system, including the environment as well as the required system;
- decomposition of data and function in the required system;
- checklists, i.e. lists of the sort of things that need definition;
- low intellectual overhead - simple, intuitive, pragmatic.

The supporting notations tend to be precise for identifying problem and system entities (e.g. processes and data flows), but the processing functions performed by these entities tend to be expressed using informal notations. However some methods do make partial use of(mathematically) formal notations (for example JSD makes use of regular expressions: Yourdon, SOM and SDL utilise finite state machines). This precision not only reduces the scope for misunderstanding, it provides scope for automatic processing.

Another benefit of structured notation is their visibility, enabling a specification or design to be checked intuitively by a user, against his powerful but unstated knowledge.

This bibliography describes some of these Structured Methodologies in more detail, for instance, Controlled Requirements Expression, Jackson System Development, MASCOT, real-time Yourdon and Structured Analysis and Design Technique (SADT).

References:

Structured Development for real-time Systems (3 Volumes) P T Yourdon Press, 1985.

Essential Systems Analysis. St M McMenamin, F Palmer, Yourdon Inc, 1984. N Y.

The Use of Structured Methods in the Development of Large Software-Based Avionic Systems. D J Hatley, Proceedings DASC, Baltimore, 1984.

Descrizione

Esiste una gamma di Metodologie Strutturate. Alcune, come: SSADM, LBMS sono progettate per l'elaborazione dei dati tradizionale e funzioni di elaborazione delle transazioni, mentre altri (MASCOT, JSD, Yourdon in tempo reale) sono più orientati al controllo di processo e alle applicazioni in tempo reale (che tendono ad essere più critiche per la sicurezza).

I Metodi Strutturati sono essenziali "strumenti di raffigurazione" per percepire sistematicamente e realizzare una partizione di un problema o di un sistema. Le loro caratteristiche principali sono:

- un ordine logico di raffigurazione, suddividendo un grosso problema in fasi maneggevoli;
- identificazione del sistema totale, comprendendo l'ambiente oltre che il sistema richiesto;
- decomposizione di dati e funzioni nel sistema richiesto;
- liste di controllo, cioè elenchi di un tipo di elementi che richiedono una definizione;
- basso sforzo/intellettuale - semplice, intuitivo, pragmatico,.

Le annotazioni di supporto tendono ad essere precise per identificare entità di problemi e sistemi (ad esempio flussi di processo e dati), ma le funzioni di elaborazione eseguite da queste entità tendono ad essere espresse usando annotazioni informali. Tuttavia alcuni metodi fanno uso parziale (matematicamente) di annotazioni formali (per esempio il JSD fa uso di espressioni regolari: Yourdon, SOM e SDL utilizzano macchine a stati finiti). Questa precisione, non solo riduce le incomprensioni, ma fornisce un'elaborazione automatica.

Un altro beneficio dell'annotazione strutturata è la sua visibilità, rendendo possibile la verifica intuitiva della specifica o della progettazione da parte di un'utente, di fronte ad una conoscenza notevole ma non definita.

Questa bibliografia descrive alcune di queste Metodologie Strutturate con maggior dettaglio, per esempio, Espressione dei requisiti Controllati, Sviluppo del Sistema Jackson, MASCOT, tempo reale e Analisi Strutturata e Tecnica di Progettazione (SADT).

Riferimenti:

Structured Development for real-time Systems (3 Volumes) P T Yourdon Press, 1985.

Essential Systems Analysis. St M McMenamin, F Palmer, Yourdon Inc, 1984. N Y.

The Use of Structured Methods in the Development of Large Software-Based Avionic Systems. D J Hatley, Proceedings DASC, Baltimore, 1984.

B.60.1 Controlled Requirements Expression (CORE)

Aim

To ensure that all the requirements are identified and expressed.

Description

This approach is intended to bridge the gap between the customer/end user and the analyst. It is not mathematically rigorous but aids the communication process - CORE is designed for requirements expression rather than specification. The approach is structured and the expression goes through various levels of refinement. The CORE process encourages a wider view of the problem, bringing in a knowledge of the environment in which the system will be used and the differing viewpoints of the various types of user. CORE includes guidelines and tactics for recognising departures from the "grand design". Departures can be corrected or explicitly identified and documented. Thus specifications may not be complete, but unresolved problems and high-risk areas are identified and have to be addressed in the subsequent design.

B.60.1 Espressione dei Requisiti Controllati (CORE)

Scopo

Assicurare che tutti i requisiti siano identificati ed espressi.

Descrizione

Questo approccio intende superare il gap tra cliente/utente finale e l'analista. Non è matematicamente rigoroso ma aiuta il processo di comunicazione - CORE è progettato per l'espressione dei requisiti piuttosto che per la specificazione. L'approccio è strutturato e l'espressione procede attraverso vari livelli di raffinamento. Il processo CORE incoraggia una visione più ampia del problema, conducendo ad una conoscenza dell'ambiente nel quale il sistema sarà usato e dei diversi punti di vista dei vari tipi di utenti. CORE prevede linee guida e tattiche per riconoscere scostamenti dai "grandi progetti". Gli scostamenti possono essere corretti od esplicitamente identificati e documentati. Perciò le specifiche possono non essere complete, ma sono identificati i problemi irrisolti e le aree ad elevato rischio e devono essere trattate nella susseguente progettazione.

B.60.2 JSD - Jackson System Development

Aim

A development method covering the development of software systems from requirements through to code, with special emphasis on real-time systems.

Description

JSD is a staged development process in which the developer models the real world processes upon which the system functions are to be based, determines the required functions and inserts them into the model, and transforms the resulting specification into one that is realisable in the target environment. It therefore covers the traditional phases of definition, design and implementation but takes a somewhat different view from the traditional methods in not being top-down.

Moreover it places great emphasis on the early stage of identifying the entities in the real world that are the concern of the system being built and on modelling them and what can happen to them. Once this analysis of the "real-world" has been done and a model created, the system's required functions are analysed to determine how they can fit into this real-world model. The resulting system model is augmented with structured descriptions of all the processes in the model and the whole is then transformed into programs that will operate in the target software and hardware environment.

B.60.2 JSD –Sviluppo del Sistema Jakson

Scopo

Un metodo di sviluppo che copre lo sviluppo di sistemi software dai requisiti fino alla codifica, con enfasi particolare per i sistemi in tempo reale.

Descrizione

JSD è un processo di sviluppo per fasi dove lo sviluppatore modella i processi del mondo reale sui quali le funzioni del sistema devono essere basate, determina le funzioni richieste e le inserisce nel modello, e trasforma la specifica risultante in una che è realizzabile nell'ambiente previsto. Pertanto esso copre le tradizionali fasi di definizione, progettazione ed implementazione, ma assume in qualche modo una visione differente rispetto ai metodi tradizionali non essendo top-down.

Inoltre esso pone molta enfasi sulla fase iniziale di identificazione delle entità nel mondo reale che sono implicate nel sistema che deve essere costruito e che lo modellano e su quanto può verificarsi. Allorché questa analisi del "mondo reale" è stata eseguita e creato un modello, le funzioni richieste dal sistema sono analizzate per determinare come esse possono adattarsi in questo modello del mondo reale. Il modello di sistema risultante è completato con descrizioni strutturate di tutti i processi del modello, ed il complesso viene trasformato in programmi che funzioneranno nell'ambiente software ed hardware stabilito.

References:

An Overview of JSD. J R Cameron, IEEE Trans SE-12 no 2 Feb 1986.

System Development. M Jackson, Prentice-Hall, 1983.

Riferimenti:

An Overview of JSD. J R Cameron, IEEE Trans SE-12 no 2 Feb 1986.

System Development. M Jackson, Prentice-Hall, 1983.

B.60.3 MASCOT

MASCOT

Aim

The design and implementation of real-time systems.

Description

MASCOT (Modular Approach to Software Construction, Operation and Test) is a design method supported by a programming system. It is a systematic method of expressing the structure of real-time system in a way that is independent of the target hardware or implementation language. It imposes a disciplined approach to design that yields a highly modular structure, ensuring a close correspondence between the functional elements in the design and the construction elements appearing in system integration. A system is designed in terms of a network of concurrent processes that communicate through channels. Channels can be either pools of fixed data or queues (pipelines of data). Control of access to channels is defined independently of the processes. It is defined in terms of access mechanisms that also enforce scheduling rules on the processes. Recent versions of MASCOT have been designed with Ada implementation in mind.

MASCOT supports an acceptance strategy based on the test and verification of single modules and larger collections of functionally related modules. A MASCOT implementation is intended to be built upon a MASCOT kernel - a set of scheduling primitives that underline the implementation and support the access mechanisms.

Reference:

MASCOT 3 User Guide. MASCOT Users Forum, RSRE, Malvern, England, 1987.

Scopo

La progettazione e l'implementazione di sistemi in tempo reale.

Descrizione

MASCOT (Approccio Modulare per Costruzione Funzionamento e Prova del Software) e un metodo di progettazione supportato da un sistema di programmazione. Esso è un metodo sistematico di esprimere la struttura di un sistema in tempo reale in modo indipendente dall'hardware previsto e dal linguaggio di implementazione. Esso impone un approccio disciplinato alla progettazione che produce una struttura altamente modulare, che assicura una stretta corrispondenza tra gli elementi funzionali nella progettazione gli elementi costruttivi che appaiono nell'integrazione del sistema. Un sistema è progettato in termini di una rete di processi concorrenti che comunicano tramite canali. I canali possono essere sia insiemi di dati stabiliti che code (canali di dati). Il controllo di accesso ai canali è definito in modo indipendente dai processi. Esso è definito in termini di meccanismi di accesso che rafforzano anche la programmazione di regole sui processi. Recenti versioni di MASCOT sono state progettate avendo in mente un'implementazione in Ada.

MASCOT supporta una strategia di accettazione basata su prove e verifiche dei singoli moduli e di una più ampia raccolta di moduli funzionalmente correlati. Una implementazione di MASCOT si intende sia costruita su un kernel di MASCOT - un'insieme di primitive programmate che sottolineano l'implementazione e supportano i meccanismi di accesso.

Riferimento:

MASCOT 3 User Guide. MASCOT Users Forum, RSRE, Malvern, England, 1987.

B.60.4 Real-time Yourdon

Yourdon nel Tempo Reale

Aim

This aims to be a complete software development method consisting of specification and design techniques oriented towards the development of real-time systems.

Description

The development scheme underlying this technique assumes a three phase evolution of a system being developed. The first phase involves

Scopo

Questo intende essere un metodo di sviluppo del software completo che consiste in specificazione e tecniche di progettazione orientate allo sviluppo di sistemi in tempo reale.

Descrizione

Lo schema di sviluppo sottolineando queste tecniche assume una evoluzione in tre fasi di un sistema da sviluppare. La prima fase coinvolge la co-

the building of an "essential model", one that describes the behaviour required by the system. The second involves the building of an implementation model which describes the structures and mechanisms that, when implemented, embody the required behaviour. The third phase involves the actual building of the system in hardware and software. The three phases correspond roughly to the traditional definition, design and implementation phases but lay greater emphasis on the fact that at each stage the developer is engaged in a modelling activity.

The essential model is in two parts: the environmental model containing a definition of the boundary between the system and its environment and a description of the external events to which the system must respond, and the behavioural model which contains schemas defining the transformation the system carries out in response to events and a definition of the data the system must hold in order to respond.

The implementation model also divides into sub-models: in this case covering the allocation of individual processes to processors and of the decomposition of the processes into modules.

To capture these models the technique combines a number of well-known techniques: data-flow diagrams, structured English, state transition diagrams and Petri Nets.

Additionally the method contains techniques for simulating a proposed system design either on paper or mechanically from the models that are drawn up.

Reference:

Structured Development for Real-time Systems (3 Volumes) PT Ward and SJ Mellor. Yourdon Press, 1985.

B.60.5 SADT- Structured Analysis and Design Technique

Aim

To model and identify, in a diagrammatic form using information flows, the decision making processes and the management tasks associated with a complex system.

Description

In SADT, the concept of an Activity-Factor Diagram plays a central role. An A/F diagram consists of activities grouped in so called "action boxes". Each action box has a unique name, and is linked to other action boxes by factor relations (drawn as arrows) which are also given unique names. Each action box can be hierarchically decomposed into subsidiary action

struzione di un "modello essenziale", che descrive il comportamento richiesto al sistema. La seconda coinvolge la costruzione di un modello di implementazione che descrive le strutture ed i meccanismi che, quando sviluppati, concretizzano il comportamento richiesto. La terza fase coinvolge la costruzione reale del sistema hardware e software. Le tre fasi corrispondono grossomodo alle fasi tradizionali di definizione, progettazione ed implementassimo, ma pongono maggiore enfasi sul fatto che per ogni fase lo sviluppatore è impegnato in un'attività di modellazione.

Il modello essenziale è in due parti: il modello ambientale contiene una definizione dei confini tra il sistema e il suo ambiente e una descrizione degli eventi esterni ai quali il sistema deve rispondere, e un modello comportamentale che contiene gli schemi che definiscono la trasformazione che il sistema esegue in risposta agli eventi e alla definizione dei dati che il sistema deve possedere per rispondere.

Il modello di implementazione si suddivide inoltre in due sottomodelli: in questo caso coprendo l'allocazione dei singoli processi sui processori e la decomposizione dei processi in moduli.

Per ottenere questi modelli la tecnica mette insieme un numero di tecniche ben note: diagrammi di flusso dei dati, inglese strutturato, diagrammi di stato-transizione e Reti di Petri.

Inoltre il metodo contiene tecniche per simulare una progettazione del sistema proposto o sulla carta o meccanicamente da modelli che sono redatti.

Riferimento:

Structured Development for Real-time Systems (3 Volumes) PT Ward and SJ Mellor. Yourdon Press, 1985.

SADT – Analisi Strutturata e Tecnica di Progettazione

Scopo

Modellare ed identificare, in forma di diagramma usando flussi di informazione, i processi di decisione ed i compiti di gestione associati ad un sistema complesso.

Descrizione

Nel SADT, il concetto di Diagramma Attività–Fattore gioca un ruolo centrale. Un diagramma A/F consiste in attività raggruppate in così dette "scatole di azione". Ciascuna scatola di azione ha un unico nome, ed è collegata alle altre scatole di azione da fattori di relazione (disegnati come frecce) ai quali vengono dati nomi unici. Ogni scatola di azione può essere gerarchicamente decomposta in scatole di azione sussidiarie e rela-

boxes and relations. There are four types of factors: inputs, controls mechanisms and outputs:

- Input: indicated by an arrow that enters an action box at the left hand side. Inputs can represent material or immaterial things and they are suitable for manipulation by one or more activities in an action box;
- Control: are typically instructions, procedures, choice criteria and so on. Controls guide the execution of an activity and they are shown by arrows entering the top side of an action box;
- Mechanism: is a resource such as personnel, organisational units or equipment, that is needed for an activity to perform its task;
- Output: can denote anything that an activity produces, and it is pictured by an arrow leaving an action box at the right hand side.

When activities are strongly related to each other by many factor relations then it perhaps better to consider these activities as an indivisible group that is contained in one action box which does not lend itself to further detailing of its content. The guiding principle for grouping of activities into action boxes is that the resulting boxes are coupled pairwise by only a few factors.

The model hierarchy of A/F diagrams is pursued until a further detailing of the action boxes is meaningless. This stage is reached when the activities within the boxes are inseparable or when further detailing of the action boxes falls outside the scope of the system analysis.

References:

Structured Analysis for Requirements Definition, DT Ross, KE Schoman Jr, IEEE Trans.Software Eng. Vol SE-3, 1977, 6-15.

Structured Analysis (SA): A language for communicating ideas, DT Ross, IEEE Trans.Software Eng., Vol SE-3,1, 16-34.

Applications and Extensions of SADT, DT Ross, Computer, April 1985, 25-34.

Structured Analysis and Design Technique - Application on Safety Systems, W Heins, Risk Assessment and Control Courseware, Module B1, chapter 11. 1989, Delft University of Technology, Safety Science Group, PO Box 5050, 2600 GB Delft, Netherlands.

B.61 Structured Programming (Referenced by clause 10)

Aim

To design and implement the program in a way which makes practical the analysis of the pro-

zioni. Vi sono quattro tipi di fattori: ingressi, controlli, meccanismi ed uscite:

- Ingressi: indicato da una freccia che entra in una scatola di azione dal lato sinistro. Gli ingressi possono rappresentare cose materiali o immateriali e sono adatti per manipolare una o più attività in una scatola di azione;
- Controllo: sono tipicamente istruzioni, procedure, criteri di scelta e così via. I controlli guidano l'esecuzione di un'attività e sono indicati da frecce entranti dal lato superiore di una scatola di azione;
- Meccanismo: è una risorsa quale il personale, un'unità dell'organizzazione o apparecchiatura, che sono necessari per un'attività per eseguire il suo compito;
- Uscita: può denotare qualsiasi cosa che è prodotta da un'attività, ed è rappresentata da una freccia uscente da una scatola di azione dal lato sinistro.

Quando le attività sono fortemente correlate tra loro da relazioni di molti fattori è meglio considerare queste attività come un gruppo indivisibile che è contenuto in una scatola di azione che non si presta a dettagliare ulteriormente il proprio contenuto. Il principio guida per raggruppare le attività in scatole di azione è quello per cui le scatole risultanti siano accoppiate solo da pochi fattori.

La gerarchia del modello gerarchico dei diagrammi A/F è portata avanti finché un dettaglio ulteriore delle scatole di azione è privo di significato. Tale fase è raggiunta quando le attività dentro le scatole sono inseparabili o quando un ulteriore dettaglio delle scatole di azione ricade all'esterno del campo di applicazione dell'analisi del sistema.

Riferimenti:

Structured Analysis for Requirements Definition, DT Ross, KE Schoman Jr, IEEE Trans.Software Eng. Vol SE-3, 1977, 6-15.

Structured Analysis (SA): A language for communicating ideas, DT Ross, IEEE Trans.Software Eng., Vol SE-3,1, 16-34.

Applications and Extensions of SADT, DT Ross, Computer, April 1985, 25-34.

Structured Analysis and Design Technique - Application on Safety Systems, W Heins, Risk Assessment and Control Courseware, Module B1, chapter 11. 1989, Delft University of Technology, Safety Science Group, PO Box 5050, 2600 GB Delft, Netherlands.

Programmazione Strutturata (Riferimento all'art.10)

Scopo

Progettare ed implementare il programma in modo da rendere pratica l'analisi del programma. Questa

gram. This analysis should be capable of discovering all significant program behaviour.

Description

The program should contain the minimum of structural complexity. Complicated branching should be avoided. Loop constraints and branching should (where possible) be simply related to input parameters. The program should be divided into appropriately small modules, and the interaction of these modules should be explicit. Features of the programming language which encourage the above approach should be used in preference to other features which are (allegedly) more efficient, except where efficiency takes absolute priority (e.g. some safety-critical systems).

References:

A Discipline of Programming. E W Dijkstra, Englewood Cliffs N J, Prentice-Hall, 1976.

Assessing a Class of Software Tools. M A Hennell et al, 7th International conference on Software Engineering, March 1984, Orlando.

A Software Tool for Top-down Programming. D C Ince, Software - Practice and Experience, Vol 13, No 8, August 1983.

B.62 Suitable Programming Languages (Referenced by D.4)

Aim

To support the requirements of this International Standard as much as possible, in particular, defensive programming, strong typing, structured programming and possibly assertions. The programming language chosen should lead to easily verifiable code with a minimum of effort and facilitate program development, verification and maintenance.

Description

The language should be fully and unambiguously defined. The language should be user or problem oriented rather than machine oriented. Widely used languages or their subsets are preferred to special purpose languages.

In addition to the already referenced features the language should provide for

- block structure,
- translation time checking,
- run time type and array bound checking, and
- parameter checking.

analisi potrebbe essere in grado di scoprire il comportamento significativo di tutti i programmi.

Descrizione

Il programma dovrebbe contenere il minimo di complessità strutturale. Ramificazioni complicate dovrebbero essere evitate. Vincoli di costrutti iterati e di diramazione dovrebbero (ove possibile) essere semplicemente correlati ai parametri d'ingresso. Il programma dovrebbe essere diviso in moduli opportunamente ridotti, e le interazioni di questi moduli dovrebbero essere esplicite. Le caratteristiche del linguaggio di programmazione che favorisce l'approccio di cui sopra dovrebbero essere usate in preferenza ad altre caratteristiche che sono (presumibilmente) più efficienti, salvo quando l'efficienza assume assoluta priorità (ad esempio alcuni sistemi in sicurezza).

Riferimenti:

A Discipline of Programming. E W Dijkstra, Englewood Cliffs N J, Prentice-Hall, 1976.

Assessing a Class of Software Tools. M A Hennell et al, 7th International conference on Software Engineering, March 1984, Orlando.

A Software Tool for Top-down Programming. D C Ince, Software - Practice and Experience, Vol 13, No 8, August 1983.

Linguaggi di Programmazione Adatti (Riferimento a D.4)

Scopo

Appoggiare i requisiti di questa Norma Internazionale nella maggiore misura possibile, in particolare, la programmazione difensiva, i linguaggi fortemente tipizzati, la programmazione strutturata, e le possibilità di asserzioni. Il linguaggio di programmazione scelto dovrebbe condurre ad un codice facilmente verificabile con uno sforzo minimo e facilitare lo sviluppo di programmi, la verifica e la manutenzione.

Descrizione

Il linguaggio dovrebbe essere definito in modo completo e senza ambiguità. Il linguaggio dovrebbe essere orientato all'utente o al problema piuttosto che alla macchina. I linguaggi di ampio utilizzo o i loro sottoinsiemi sono preferiti ai linguaggi con scopi particolari.

In aggiunta alle caratteristiche già riferite i linguaggi dovrebbero prevedere

- struttura a blocchi,
- verifica del tempo di traduzione,
- verifica del tipo del tempo di esecuzione e di accesso a locazioni di memoria riservate, e
- verifica del parametro.

The language should encourage

- the use of small and manageable modules,
- restriction of access to data in defined modules,
- definition of variable sub-ranges, and
- any other type of error limiting constructs.

It is desirable that the language is supported by a suitable translator, appropriate libraries of pre-existing modules, a debugger and tools for both version control and development.

Features which make verification difficult and therefore should be avoided are:

- unconditional jumps excluding subroutine calls;
- recursion;
- pointers, heaps or any type of dynamic variables or objects;
- interrupt handling at source code level;
- multiple entries or exits of loops, blocks or subprograms;
- implicit variable initialisation or declaration;
- variant records and equivalence; and
- procedural parameters.

Low level languages, in particular assembly languages, present problems due to their machine oriented nature.

Il linguaggio dovrebbe stimolare

- l'uso di moduli piccoli e maneggevoli,
- restrizione di accesso ai dati in moduli definiti,
- definizione di variabili sub-ranges, e
- ogni altro tipo di errore che limita i costrutti.

È desiderabile che il linguaggio sia supportato da un adatto traduttore, opportune librerie di moduli preesistenti, un debugger e strumenti sia per il controllo delle versioni che per lo sviluppo.

Caratteristiche che rendono difficile la verifica e che pertanto dovrebbero essere evitate sono:

- saliti incondizionati che escludono chiamate di subroutine;
- ripetizioni;
- puntatori, heaps od ogni tipo di variabile o oggetto dinamico;
- gestione delle interruzioni a livello del codice sorgente;
- ingressi o uscite multiple di costrutti iterativi, blocchi o sottoprogrammi;
- inizializzazione o dichiarazione di variabile implicita;
- registrazioni di variante ed equivalenze; e
- parametri procedurali.

I linguaggi di basso livello, in particolare linguaggi in assembler, presentano problemi dovuti alla loro natura orientata alla macchina.

B.63 Symbolic Execution (Referenced by D.8)

Aim

To show the agreement between the source code and the specification.

Description

The program is executed substituting the left hand side by the right hand side in all assignments. Conditional branches and loops are translated into Boolean expressions. The final result is a symbolic expression for each program variable. This can be checked against the expected expression.

References:

Formal Program Verification using Symbolic Execution. R.B. Dannenberg and G.W. Ernst,IEEE Trans Software Engineering, Vol. SE-8, No 1, 1982.

Symbolic Execution and Software Testing. J.C. King, Comm. ACM, Vol. 19, No 7, 1976.

Esecuzione Simbolica (Riferimento a D.8)

Scopo

Mostrare la concordanza tra il codice sorgente e la specifica.

Descrizione

Il programma è eseguito sostituendo il lato sinistro con il lato destro in tutte le assegnazioni. Ramificazioni condizionate e costrutti iterativi sono tradotti in espressioni Booleane. Il risultato finale è un'espressione simbolica per ogni variabile di programma. Questo può essere verificato nei confronti dell'espressione attesa.

Riferimenti:

Formal Program Verification using Symbolic Execution. R.B. Dannenberg and G.W. Ernst,IEEE Trans Software Engineering, Vol. SE-8, No 1, 1982.

Symbolic Execution and Software Testing. J.C. King, Comm. ACM, Vol. 19, No 7, 1976.

B.64 Time Petri Nets (Referenced by D.5 and D.7)

Aim

To model relevant aspects of the system behaviour and to assess and possibly improve safety

Reti di Petri Temporali (Riferimento a D.5 e D.7)

Scopo

Modellare aspetti attinenenti al comportamento del sistema e valutare migliorando possibilmente

and operational requirements through analysis and re-design.

Description

Petri nets belong to a class of graph theoretic models which are suitable for representing information and control flow in systems exhibiting concurrency and asynchronous behaviour.

A Petri net is a network of places and transitions. The places may be "marked" or "unmarked". A transition is "enabled" when all the input places to it are marked. When enabled, it is permitted (but not obliged) to "fire". If it fires, the input marks are removed, and each output place from the transition is marked instead.

The potential hazards are represented as particular states (markings) in the model. Extended Petri nets allow timing features of the system to be modelled. Although "classical" Petri nets concentrate on control flow aspects, several extensions have been proposed to incorporate dataflow into the model.

References:

Net Theory and Applications. W. Brauer (ed), Lecture Notes in Computer Science, Vol. 84, Springer 1980.

Petri Net Theory and Modelling of Systems. J.L. Peterson, Prentice Hall, 1981.

Safety Analysis using Petri Nets. N. Leveson and J. Stolzy, Proc. FTCS 15, Ann Arbor,Michigan, June 1985, IEEE 1985.

A Tool for Requirements Specification and Analysis of Real Time Software Based on Timed Petri Nets. S. Bologna, F. Pisacane, C Ghezzi, D. Mandrioli, Proc. SAFECOMP 88, 9-11Nov. 1988. Fulda Fed. Rep. of Germany 1988.

B.65 Translator Proven In Use (Referenced by clause 10)

Aim

To avoid any difficulties due to translator failures which can arise during development, verification and maintenance of a software package.

Description

A translator is used, whose correct performance has been demonstrated in many projects already. Translators without operating experience or with any serious known errors are prohibited.

If the translator has shown small deficiencies the related language constructs are noted down and carefully avoided during a safety related project.

la sicurezza e i requisiti funzionali tramite l'analisi e la riprogettazione.

Descrizione

Le reti di Petri appartengono ad una classe di modelli teorici grafici che sono adatti a rappresentare informazioni e flussi di controllo in sistemi esistenti che presentano concorrenza e comportamento asincrono.

Una rete di Petri è una rete di comunicazione di luoghi e transizioni. I luoghi possono essere "marcati" o "non marcati". Una transizione è "abilitata" quando tutti i suoi luoghi di ingresso sono marcati. Quando abilitata, è permesso (ma non obbligato) "accendere". Se acceso, le marcature d'ingresso sono rimosse, ed ogni luogo di uscita dalla transizione è marcato.

I pericoli potenziali sono rappresentati come stati particolari (marcature) nel modello. Reti di Petri estese permettono di collocare nel tempo le caratteristiche del sistema da modellare. Per quanto le reti di Petri "classiche" si concentrino sugli aspetti del flusso di controllo, sono state proposte molte estensioni per includere il flusso dei dati nel modello.

Riferimenti:

Net Theory and Applications. W. Brauer (ed), Lecture Notes in Computer Science, Vol. 84, Springer 1980.

Petri Net Theory and Modelling of Systems. J.L. Peterson, Prentice Hall, 1981.

Safety Analysis using Petri Nets. N. Leveson and J. Stolzy, Proc. FTCS 15, Ann Arbor,Michigan, June 1985, IEEE 1985.

A Tool for Requirements Specification and Analysis of Real Time Software Based on Timed Petri Nets. S. Bologna, F. Pisacane, C Ghezzi, D. Mandrioli, Proc. SAFECOMP 88, 9-11Nov. 1988. Fulda Fed. Rep. of Germany 1988.

Traduttore provato con l'uso (Riferimento all'art. 10)

Scopo

Evitare ogni difficoltà dovuta ai mancati funzionamenti del traduttore che possono sorgere durante lo sviluppo, verifica e manutenzione di un pacchetto software.

Descrizione

È usato un traduttore la cui corretta prestazione è già stata dimostrata in molti progetti. Traduttori senza esperienza di funzionamento o con qualsiasi errore serio noto sono proibiti.

Se il traduttore ha mostrato piccole deficienze i costrutti del linguaggio relativo sono annotati ed accuratamente evitati nel corso di un progetto in sicurezza.

Another version to this way of working is to restrict the usage of the language to only its commonly used features.

This recommendation is based on the experience from many projects. It has been shown that immature translators are a serious handicap to any software development. They make a safety-related software development generally infeasible.

It is also known, presently, that no method exits to prove the correctness for all compiler parts.

Un'altra versione di tale modo di lavorare è di restringere l'uso del linguaggio alle sole sue caratteristiche più comunemente usate.

Tale raccomandazione è basata sull'esperienza di molti progetti. È stato mostrato che traduttori non maturi sono un serio handicap per lo sviluppo di qualsiasi software. Essi rendono generalmente non realizzabile lo sviluppo di un software in sicurezza.

È anche noto che, al momento, non esiste alcun metodo per provare la correttezza di tutte le parti del compilatore.

B.66 Walkthroughs/Design Reviews (Referenced by D.8)

Aim

To detect errors in some product of the development process as soon and as economically as possible.

Description

IEC/TC 56, have published a Guide on Formal Design Reviews, which includes a general description of formal design reviews, their objectives, details of the various design review types, the composition of a design review team and their associated duties and responsibilities. The IEC document also provides general guidelines for planning and conducting formal design reviews, as well as specific details concerning the role of independent specialists within a design review team. Examples of specialist functions include, amongst others, Reliability, Maintenance Support and Availability.

The IEC recommend that a "formal design review shall be conducted for all new products/processes, new applications, and revisions to existing products and manufacturing processes which affect the function, performance, safety, reliability, ability to inspect maintainability, availability, ability to cost, and other characteristics affecting the end product/process, users or bystanders".

A code walk through consists of a walk through team selecting a small set of paper test cases, representative sets of inputs and corresponding expected outputs for the program. The test data is then manually traced through the logic of the program.

References:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

Reliability and Maintainability Guide on Formal Design Review (Draft) International Electrotechnical Commission, Technical Committee No 56, January 1988.

Analisi del Percorso (Walkthroughs)/Riesame della progettazione (Riferimento a D.8)

Scopo

Rilevare errori in alcuni prodotti del processo di sviluppo il più presto ed il più economicamente possibile.

Descrizione

Il comitato IEC/TC 56 ha pubblicato una Guida sui Riesami della progettazioni Formali che comprende una descrizione generale del riesame della progettazione formale, i suoi obbiettivi, dettagli dei vari tipi di riesame della progettazione, la composizione del gruppo di lavoro per il riesame della progettazione e le loro relative funzioni e responsabilità. Il documento IEC fornisce anche linee guida generali per pianificare e condurre i riesami della progettazione formali, oltre che i dettagli specifici concernenti il ruolo degli specialisti indipendenti entro il gruppo di riesame della progettazione. Esempi delle funzioni degli specialisti comprendono, tra l'altro, quelli per l'Affidabilità, il Supporto alla Manutenzione e la Disponibilità.

La IEC raccomanda che un "riesame della progettazione formale sia eseguito per tutti i nuovi prodotti/processi, per le nuove applicazioni e le revisioni di prodotti esistenti o processi manifatturieri che influenzano la funzione, la prestazione, la sicurezza, l'affidabilità, la capacità di ispezionare la manutenibilità, la disponibilità, la capacità di spesa e altre caratteristiche che influenzano il prodotto/processo finale, gli utenti o gli soggetti secondari".

L'analisi del percorso del codice consiste in un gruppo di analisi che sceglie un piccolo insieme di casi di prova cartacei, insieme rappresentativi dei dati di ingresso e delle corrispondenti uscite attese per il programma. I dati di prova sono quindi tracciati manualmente attraverso la logica del programma.

Riferimenti:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

Reliability and Maintainability Guide on Formal Design Review (Draft) International Electrotechnical Commission, Technical Committee No 56, January 1988.

B.67 Fuzzy Logic (Referenced by clause 10)

Aim

Fuzzy Logic is a mathematical discipline, based on the fuzzy set theory that allows for degrees of truth and falseness. It is a generalisation of bi-level logic which provides a model for approximate reasoning. It enables the incorporation of human intelligence into automatic systems. By acknowledging the difficulty of defining precise boundaries, Fuzzy Logic reduces the required precision, and hence leads to high level and simple solutions which are easy to control.

Description

The essential part of a Fuzzy Logic solution is a set of linguistic rules (IF - THEN rules) where the antecedents and the consequents are associated with fuzzy sets.

EXAMPLE IF speed is fast and distance_to_stop is medium THEN accelerator is near zero and brake is light

In this example, "speed" is a linguistic variable characterised by a fuzzy set "fast", which can be interpreted as "speed is above about 70 km/h". If speed is less than 60 km/h, the membership value of "fast" is 0. If speed is above 80 km/h, the memberships value of "fast" is 1. If speed is between 60 and 80 km/h, the membership value of "fast" varies between 0 and 1.

The decision making logic is based on mathematical classes of operators: the triangular norms and triangular co-norms.

Fuzzy Logic rule-based systems differ from other expert systems in many ways as: (1) the small number of rules, (2) the use of forward chaining only, (3) non-chaining of inferences (all rules are executed in parallel in one iteration), (4) the statistically defined rules, (5) the speed of execution due to simplicity of the solutions, and (6) the determinism.

During the past few years, Fuzzy Logic has found numerous applications in fields ranging from finance to earthquake engineering. In particular, fuzzy control has emerged to control highly non-linear systems, or systems which mathematical description is unknown or too complex to be treated analytically, or systems in which the available measurements are of poor quality.

Notable applications of fuzzy logic control include aircraft flight control, and power systems and nuclear reactor control. More recently, fuzzy control has been applied successfully to automatic train operation systems.

Logica Fuzzy (Riferimento all'art. 10)

Scopo

La Logica Fuzzy è una disciplina matematica, basata sulla teoria dell'insieme fuzzy che valuta i gradi di vero e falso. È una generalizzazione della logica a due livelli che fornisce un modello per il ragionamento per approssimazione. Essa consente di introdurre l'intelligenza dell'uomo nei sistemi automatici. Con il riconoscimento delle difficoltà di definire confini precisi, la Logica Fuzzy riduce la precisione richiesta, e conduce pertanto ad elevati livelli e a semplici soluzioni che sono facili da controllare.

Descrizione

La parte essenziale di una soluzione a Logica Fuzzy è un'insieme di regole linguistiche (regole IF - THEN) ove gli antecedenti e i conseguenti sono associati con insiemi fuzzy.

ESEMPIO IF la velocità è rapida e la distanza dall'arresto è media THEN l'acceleratore è quasi zero ed il freno è ridotto

Nell'esempio, "la velocità" è una variabile linguistica caratterizzata da un'insieme fuzzy "rapida", che può essere interpretata come "la velocità è circa oltre 70 km/h". Se la velocità è inferiore a 60 km/h, il valore associato di "rapida" è 0. Se la velocità è oltre 80 km/h, il valore associato di "rapida" è 1. Se la velocità è tra 60 e 80 km/h, il valore associato di "rapido" varia tra 0 e 1.

La logica di assunzione delle decisioni è basata su classi matematiche di operatori: le norme e le co-norme triangolari.

I sistemi basati sulle regole della Logica Fuzzy differiscono dagli altri sistemi esperti in molti modi come: (1) il piccolo numero di regole, (2) il solo uso di concatenamento in avanti, (3) non concatenamento delle interferenze (tutte le regole sono eseguite in parallelo in una iterazione), (4) le regole definite statisticamente, (5) la velocità di esecuzione dovuta alla semplicità delle soluzioni, e (6) il determinismo.

Nei pochi anni trascorsi, la Logica Fuzzy ha trovato numerose applicazioni nei campi che spaziano dalla finanza all'ingegneria dei terremoti. In particolare, il controllo fuzzy è emerso per controllare sistemi altamente non lineari o sistemi la cui descrizione matematica è sconosciuta o troppo complessa per essere trattata analiticamente, o sistemi nei quali le misure disponibili sono di scarsa qualità.

Notevoli applicazioni del controllo a logica fuzzy comprendono il controllo di volo degli aerei ed il controllo dei sistemi di generazione di energia e dei reattori nucleari. Più recentemente, il controllo fuzzy è stato applicato con successo ai sistemi ATO (automatic train operation).

References:

Fuzzy Sets: Zadeh. Information and Control, 1965, vol 8, pp338-353

Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part I & II. Chuen Chien Lee. IEEE Transactions on systems, man, and cybernetics, vol. 20, No2, March/April 1990

Industrial Applications of Fuzzy Control: M.Sugeno Ed., Amsterdam North Holland, 1985

Automatic Train Operation System by Predictive Fuzzy Control: Yasunobu, Miyamoto, in Industrial Applications of Fuzzy Control, M.Sugeno Ed., Amsterdam North Holland, 1985

An automatic operation method for control rods in BWR plants: Kinoshita, Fukusaki, Satoh, Miyake, Proc. Specialists Meeting on In-Core Instrumentation and Reactor Core Assessment. Cadarache, France 1988

Use of rule-based system for process control. Bernard. IEEE Contr. Syst. Mag., Vol.8, No.5, pp.3-13, 1988

Riferimenti:

Fuzzy Sets: Zadeh. Information and Control, 1965, vol 8, pp338-353

Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part I & II. Chuen Chien Lee. IEEE Transactions on systems, man, and cybernetics, vol. 20, No2, March/April 1990

Industrial Applications of Fuzzy Control: M.Sugeno Ed., Amsterdam Nord Holland, 1985

Automatic Train Operation System by Predictive Fuzzy Control: Yasunobu, Miyamoto, in Industrial Applications of Fuzzy Control, M.Sugeno Ed., Amsterdam North Holland, 1985

An automatic operation method for control rods in BWR plants: Kinoshita, Fukusaki, Satoh, Miyake, Proc. Specialists Meeting on In-Core Instrumentation and Reactor Core Assessment. Cadarache, France 1988

Use of rule-based system for process control. Bernard. IEEE Contr. Syst. Mag., Vol.8, No.5, pp.3-13, 1988

B.68 Object Oriented Programming (Referenced by clause 10)

Aim

To enable rapid prototyping, to more easily reuse existing software components, to achieve information hiding, to reduce the likelihood of errors during the whole lifecycle, to reduce the necessary effort during the maintenance phase, to break down complex problems into more easily manageable small problems, to reduce the dependencies between software components, to create more easily extendible applications.

Description

Object oriented programming is a fundamentally new way of thinking about software based on abstractions that exist in the real world rather than based on computational abstractions. Object oriented programming organises software as a collection of objects that incorporate both data structure and behaviour. This is in contrast to conventional programming where data structure and behaviour are only loosely connected.

Object: an object consists of a private data area and set of operations - so called methods - on that object. Methods may be public or private. No other software component is allowed to read or change the private data of an object directly. Every other software component has to use the public methods on that object to read or write data in the private data area of an object.

Programmazione Orientata all'Oggetto (Riferimento all'art. 10)

Scopo

Consentire una rapida prototipazione, per riusare più facilmente le componenti software esistenti, per ottenere l'occultamento dell'informazione, per ridurre la probabilità di errori durante tutto il ciclo di vita, per ridurre l'impegno necessario durante la fase di manutenzione, per spezzare problemi complessi in piccoli problemi gestibili più facilmente, per ridurre le dipendenze tra componenti software, per creare più facilmente applicazioni estensibili.

Descrizione

La programmazione orientata agli oggetti è una via essenzialmente nuova di pensare il software basata sulle astrazioni che esistono nel mondo reale piuttosto che basata su astrazioni di calcolo. La programmazione orientata agli oggetti organizza il software come una collezione di oggetti che incorpora sia la struttura che il comportamento dei dati. Questo è in contrasto con la programmazione convenzionale dove la struttura dei dati e comportamento sono solo relativamente connessi.

Oggetto: un oggetto consiste di un'area privata di dati e di un insieme di operazioni – anche chiamati metodi – su quell'oggetto. I metodi possono essere pubblici o privati. A nessun altro componente software è permesso di leggere o cambiare direttamente i dati privati di un oggetto. Ogni altro componente software deve usare i metodi pubblici su quell'oggetto per leggere o scrivere dati nell'area privata di un oggetto.

Object Class: by specifying an object class (often in the form of a type definition) you enable the instantiation of numerous objects of the same class, i.e., all instantiations have the private data area and the methods defined in the object class.

(Multiple) Inheritance: an object class can inherit the private data area and the methods of one (or more) superclasses (object classes above it in the class hierarchy) with being allowed to add some private data, to add some methods or to modify the implementations of the inherited methods. Using Inheritance multiple object class trees can be built.

Polymorphism: the same operation may behave different on different object classes, e.g., the write operation for a terminal object writes characters to that terminal and a write operation to a file object writes characters to that file.

References:

Object Oriented Software Construction, Bertrand Meyer, England: Prentice Hall International, 1988.

Classification as a paradigm for computing, Peter Wegner, Technical Report CS-86-11, Brown University, May 1986.

Learning language, Peter Wegner, Byte (McGraw-Hill publication) March 1989.

All OOPSLA (Object-Oriented Programming Systems, Languages and Applications) and ECOOP (European Conference on Object-Oriented Programming) conference proceedings.

Classe dell'oggetto: specificando una classe dell'oggetto (spesso nella forma di una definizione di tipo) si rende possibile la replica di numerosi oggetti della medesima classe, cioè, tutte le repliche hanno l'area dei dati privata e i metodi definiti nella classe dell'oggetto.

Ereditarietà (Multipla): una classe dell'oggetto può ereditare l'area di dati privati ed i metodi di una (o più) superclassi (classi di oggetti al di sopra di questa nella gerarchia delle classi) essendo consentito aggiungere alcuni dati privati, per aggiungere alcuni metodi o modificare le implementazioni dei metodi ereditati. Usando l'ereditarietà multipla possono essere costruiti alberi di classe dell'oggetto.

Polimorfismo: la medesima operazione si può comportare in modo diverso in differenti classi di oggetti, ad esempio, l'operazione di scrittura per un oggetto terminale scrive caratteri su quel terminale ed un'operazione di scrittura su un oggetto archivio scrive caratteri su quell'archivio.

Riferimenti:

Object Oriented Software Construction, Bertrand Meyer, England: Prentice Hall International, 1988.

Classification as a paradigm for computing, Peter Wegner, Technical Report CS-86-11, Brown University, May 1986.

Learning language, Peter Wegner, Byte (McGraw-Hill publication) March 1989.

All OOPSLA (Object-Oriented Programming Systems, Languages and Applications) and ECOOP (European Conference on Object-Oriented Programming) conference proceedings.

B.69 Traceability (Referenced by clause 11)

Aim

The objective of Traceability is to ensure that all requirements can be shown to have been properly met and that no untraceable material has been introduced.

Description

Traceability to requirements shall be an important consideration in the validation of a system and means shall be provided to allow this to be demonstrated throughout all phases of the lifecycle.

Traceability shall be considered applicable to both functional and non-functional requirements and shall particularly address

a) traceability of requirements to the design or other objects which fulfil them,
b) traceability of design objects to the implementation objects which instantiate them,
c) traceability of requirements and design objects to the operational and maintenance

Tracciabilità (Riferimento all'art. 11)

Scopo

L'obbiettivo della Tracciabilità è quello di assicurare che si possa mostrare che tutti i requisiti siano stati soddisfatti in modo appropriato e che non sia stato introdotto materiale non rintracciabile.

Descrizione

La tracciabilità dei requisiti deve essere considerata importante nella validazione del sistema e devono essere previsti mezzi per consentire che questo sia dimostrato in tutte le fasi del ciclo di vita.

La tracciabilità deve essere considerata applicabile sia ai requisiti funzionali che a quelli non funzionali e deve trattare in particolare

a) tracciabilità dei requisiti per la progettazione o altri oggetti che la soddisfano,
b) tracciabilità degli oggetti della progettazione con gli oggetti dell'implementazione che li replica,
c) tracciabilità dei requisiti e degli oggetti della progettazione con oggetti funzionali e di ma-

objects required to be applied in the safe and proper use of the system,

d) traceability of requirements, design, implementation, operation and maintenance objects, to the verification and test plans and specifications which will determine their acceptability,

e) traceability of verification and test plans and specifications to the test or other reports which record the results of their application.

Where requirements, design or other objects are instantiated as a number of separate documents, traceability shall be maintained within the document structures and in a hierarchical manner.

The output of the Traceability process shall be the subject of formal Configuration Management.

nutenzione che si richiede siano applicati nell'uso sicuro ed appropriato del sistema,

d) tracciabilità di requisiti, progettazione, implementazione, funzionamento e oggetti di manutenzione, per la verifica e i piani di prova e le specifiche che determineranno la loro accettabilità,

e) tracciabilità della verifica e dei piani di prova e delle specifiche per la prova o altri rapporti che registrano i risultati della loro applicazione.

Ove requisiti, la progettazione o altri oggetti sono replicati come una serie di documenti separati, la tracciabilità deve essere mantenuta nelle strutture del documento ed in modo gerarchico.

Il risultato della Tracciabilità deve essere oggetto di Gestione della Configurazione formale.

Fine Documento

La presente Norma è stata compilata dal Comitato Elettrotecnico Italiano
e beneficia del riconoscimento di cui alla legge 1º Marzo 1968, n. 186.

Editore CEI, Comitato Elettrotecnico Italiano, Milano - Stampa in proprio
Autorizzazione del Tribunale di Milano N. 4093 del 24 luglio 1956
Responsabile: Ing. A. Alberici

9 – Trazione

CEI 9-7
Relè neutri a corrente continua per impianti fissi di segnalamento e di sicurezza di ferrovie, tranvie e metropolitane

CEI EN 50215 (CEI 9-13)
Applicazioni ferroviarie, tranviarie, filotranviarie, metropolitane Prove del materiale rotabile dopo il completamento della costruzione e prima dell'entrata in servizio

CEI 9-21
Caratteristiche e prove dei sistemi di frenatura elettrodinamici ed elettromagnetici

CEI 9-22
Guida per la compilazione delle disposizioni per la sicurezza e la regolarità dell'esercizio ferroviario Esercizio degli impianti di trazione elettrica

CEI EN 50153 (CEI 9-29)
Applicazioni ferroviarie Materiale rotabile Misure di protezione contro i pericoli di origine elettrica

CEI EN 50155 (CEI 9-30)
Applicazioni ferroviarie Equipaggiamenti elettronici utilizzati sul materiale rotabile

CEI EN 50163 (CEI 9-31)
Applicazioni ferroviarie Tensioni di alimentazione dei sistemi di trazione

CEI EN 50121-1 (CEI 9-35/1)
Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane Compatibilità elettromagnetica Parte 1: Generalità

CEI EN 50121-2 (CEI 9-35/2)
Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane - Compatibilità elettromagnetica Parte 2: Emissione dell'intero sistema ferroviario verso l'ambiente esterno

CEI EN 50121-4 (CEI 9-35/4)
Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane Compatibilità elettromagnetica Parte 4: Emissione ed immunita' delle apparecchiature di segnalamento e telecomunicazioni

CEI R009-001
Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane Sistemi di comunicazione, di segnalamento e di processo Tassi di guasto pericoloso e livelli di integrità della sicurezza (SIL)

CEI EN 50261 (CEI 9-47)
Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane Montaggio dell'apparecchiatura elettronica

CEI ENV 50129 (CEI 9-55)
Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane Sistemi elettronici di sicurezza per il segnalamento

CEI EN 50126 (CEI 9-58)
Applicazioni ferroviarie, tranviarie, filotranviarie, metropolitane La specificazione e la dimostrazione di Affidabilità, Disponibilità, Manutenibilità e Sicurezza (RAMS)

CEI EN 50125-1 (CEI 9-60)
Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane Condizioni ambientali per gli equipaggiamenti Parte 1: Equipaggiamenti nel materiale rotabile

CEI EN 50239 (CEI 9-62)
Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane Sistemi di radiocomando a distanza di veicoli di trazione per trasporto merci

CEI EN 50124-1 (CEI 9-65/1)
Applicazioni ferroviarie, tranviarie, filotranviarie, metropolitane - Coordinamento degli isolamenti Parte 1: Requisiti base - Distanze in aria e distanze superficiali per tutta l'apparecchiatura elettrica ed elettronica

CEI EN 50124-2 (CEI 9-65/2)
Applicazioni ferroviarie, tranviarie, filotranviarie, metropolitane - Coordinamento degli isolamenti Parte 2: Sovratensioni e relative protezioni

CEI EN 50159-1 (CEI 9-66/1)
Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane - Sistemi di telecomunicazione, segnalamento ed elaborazione Parte 1: Comunicazioni di sicurezza in sistemi di trasmissione di tipo chiuso

CEI EN 50159-2 (CEI 9-66/2)
Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane - Sistemi di telecomunicazione, segnalamento ed elaborazione Parte 2: Comunicazioni di sicurezza in sistemi di trasmissione di tipo aperto

CEI R009-005
Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane Sistemi di radiocomando a distanza di veicoli di trazione per trasporto merci in trazione multipla

€ 125,00

NORMA TECNICA
CEI EN 50128:2002-04
Totale Pagine 136

Sede del Punto di Vendita e di Consultazione
20134 Milano - Via Saccardo, 9
tel. 02/21006.1 • fax 02/21006.222
http://www.ceiuni.it e-mail:cei@ceiuni.it