
ANSI/IEEE Std 1016-1987 IEEE Recommended Practice for Software Design Description

Approvato il 12-3-1987
Riconfermato il 23-9-1998

Obbiettivi dello standard

Lo standard ANSI/IEEE 1016, specifica il contenuto informativo necessario e raccomanda una organizzazione per la descrizione di un progetto software.

Il documento prodotto mostra come il sistema software deve essere strutturato per soddisfare i requisiti identificati nell'SRS.

Lo standard non supporta esplicitamente nessuna metodologia di progetto e tanto meno una tecnologia per la descrizione dello stesso.

Esso è corredato da una Guida (IEEE Std 1016.1-1993) "*IEEE Guide to Software Design Descriptions*"

Sezioni dello standard

Lo standard è composto da 6 sezioni:

- ◆ La sezione 1 specifica lo scopo dello standard;
- ◆ La sezione 2 contiene riferimenti ad altri standard ANSI/IEEE;
- ◆ La sezione 3 definisce la terminologia usata dallo standard;
- ◆ La sezione 4 illustra come l'SDD si colloca nel ciclo di vita e perché viene usato;
- ◆ La sezione 5 descrive l'informazione minima che deve essere inclusa nell'SDD;
- ◆ La sezione 6 fornisce delle raccomandazioni su come deve essere organizzato l'SDD

Sezione 5: contenuto informativo

Il modello descrittivo di un progetto software può essere rappresentato come una collezione di “**design entities**” ognuna avente **proprietà (attributi)** e **relazioni**.

Una design entity è un elemento di un progetto strutturalmente e funzionalmente distinto dagli altri.

Lo standard presenta una lista minima di attributi per le entities da includere nell'SDD:

- ◆ **Identification:** nome dell'entità;
- ◆ **Type:** tipo di entità (sottoprogramma, modulo, processo, data store, ecc.);
- ◆ **Purpose:** perché esiste l'entità;

Attributi delle entità

- ◆ **Function:** descrizione di cosa fa l'entità, ovvero la trasformazione applicata agli input per produrre gli output desiderati; nel caso di data store descrive la tipologia di dati immagazzinati dall'entità;
- ◆ **Subordinates:** identificazione delle entità che compongono l'entità in questione;
- ◆ **Dependencies:** descrizione delle relazioni esistenti tra l'entità in questione e le altre (es. relazione di USO);
- ◆ **Interface:** descrizione di come le altre entità interagiscono con l'entità in questione;

Attributi delle entità (II)

- ◆ **Resources:** descrizione degli elementi, esterni al progetto, usati dall'entità. Solitamente sono informazioni riguardanti i dispositivi fisici (stampanti, memoria, ecc.), dispositivi software (librerie, servizi del s.o. ecc.). Vanno inoltre identificate situazioni di deadlock e concorrenza;
- ◆ **Processing:** Descrizione di come l'entità esegue le sue funzionalità. E' una descrizione dell'algoritmo usato dall'entità, e pertanto un raffinamento dell'attributo *function*;
- ◆ **Data:** descrizione dei dati immagazzinati dall'entità: metodi di rappresentazione, valori iniziali, uso, semantica, formato, valori accettabili. La descrizione può avere la forma di un dizionario dati.

Struttura dell'SDD

1. **Introduzione**
 - 1.1 **Purpose (intenzioni del documento)**
 - 1.2 **Scope**
 - 1.3 **Definizioni e acronimi**
2. **Riferimenti**
3. **Descrizione della decomposizione del progetto**
 - 3.1 **Decomposizione in moduli**
 - 3.1.1 **Descrizione Modulo 1**
 - 3.1.2 **Descrizione Modulo 2**
 - 3.2 **Decomposizione in processi concorrenti**
 - 3.2.1 **Descrizione Processo 1**
 - 3.2.2 **Descrizione Processo 2**

Struttura dell'SDD (II)

- 3.3 **Decomposizione dei dati**
 - 3.3.1 **Descrizione Entità 1**
 - 3.3.2 **Descrizione Entità 2**
4. **Descrizione delle dipendenze**
 - 4.1 **Dipendenze tra moduli**
 - 4.2 **Dipendenze tra processi**
 - 4.3 **Dipendenze tra dati**
5. **Descrizione delle interfacce**
 - 5.1 **Interfacce dei moduli**
 - 5.1.1 **Descrizione interfaccia Modulo 1**
 - 5.1.2 **Descrizione interfaccia Modulo 2**

Struttura dell'SDD (III)

- 5.2 Interfacce dei processi
 - 5.2.1 Descrizione interfaccia Processo 1
 - 5.2.2 Descrizione interfaccia Processo 2
- 6. Progettazione dettagliata
 - 6.1 Progettazione dettagliata dei moduli
 - 6.1.1 Dettagli del Modulo 1
 - 6.1.2 Dettagli del Modulo 2
 - 6.2 Progettazione dettagliata dei dati
 - 6.2.1 Dettagli Entità 1
 - 6.2.2 Dettagli Entità 2

I paragrafi 3,4,5,6 del documento costituiscono le differenti “viste” del progetto.

Vista 1: descrizione della decomposizione

- ◆ **Scope:** partizionare il sistema in entità di progetto;
- ◆ **Uso:** identificare le entità, mappare su di esse i requisiti funzionali e permettere al project management di monitorare ogni singola parte del progetto;
- ◆ **Attributi coinvolti:** Identification, Type, Purpose, Function, Subordinates;
- ◆ **Possibili rappresentazioni:** Decomposizione gerarchica, linguaggio naturale.

Vista 2: descrizione delle dipendenze

- ◆ **Scope:** Identificare le relazioni tra entità;
- ◆ **Uso:** Fornire una visione globale del progetto per identificare entità causa di failures e colli di bottiglia; stabilire priorità sullo sviluppo; fornire uno strumento per il testing di integrazione;
- ◆ **Attributi coinvolti:** Identification, Type, Purpose, Dependencies, Resources;
- ◆ **Possibili rappresentazioni:** Structure Charts, Transaction Diagrams, Class Diagrams.

Vista 3: descrizione delle interfacce

- ◆ **Scope:** Fornire a progettisti, programmatori e tester quanto necessario per utilizzare le entità;
- ◆ **Uso:** “Contratto” tra progettisti, programmatori e tester per permettere a ciascuno l’utilizzo dell’entità che non ha sviluppato;
- ◆ **Attributi coinvolti:** Identification, Function, Interfaces;
- ◆ **Possibili rappresentazioni:** Descrizione dettagliata di input, output, interfacce utente; uso di data-dictionary per entità data-driven.

Vista 4: descrizione dettagliata

- ◆ **Scope:** Descrizione dei dettagli interni di ogni singola entità;
- ◆ **Uso:** Fornire ai programmatori le informazioni per l'implementazione; strumento per l'unit testing;
- ◆ **Attributi coinvolti:** Identification, Processes, Data;
- ◆ **Possibili rappresentazioni:** PDL (Program Design Language), Pseudocodice, Inglese strutturato, Nassi-Schneidermann charts, Flow-Charts.

Metodologie: Function Oriented

Il sistema viene modellato dividendolo in componenti, identificando gli input richiesti da ciascuna parte e gli output prodotti.

Rappresentazioni:

- ◆ Data dictionary;
- ◆ Structure Charts;
- ◆ Process Specifications

Metodologie: Data Oriented

La struttura del programma è derivata dalle strutture dei dati.

Rappresentazioni:

- ◆ Jackson Structured Programming (JSP)

Metodologie: Real Time Control-Oriented

Estensione dell'approccio *Function Oriented* per dominare la complessità di sistemi real-time. Fornisce viste per rappresentare la concorrenza tramite rappresentazioni di task e comunicazioni tra essi.

Rappresentazioni:

- ◆ Ward and Mellor approach

Metodologie: Object Oriented

Produce una rappresentazione basata sugli oggetti manipolati dal sistema, piuttosto che dalle sue funzioni.

Rappresentazioni:

- ◆ OOD (Object Oriented Design) di Coad & Yourdon;
- ◆ UML (Unified Modeling Language) di Booch, Rumbaugh e Jacobson.

Metodologie: Formal Language Oriented

Rappresentazione del sistema tramite appositi linguaggi formali.

Rappresentazioni:

- ◆ Z;
- ◆ Paisley;
- ◆ Vienna Design Method (VDM).