

**FIAT
GROUP**

Electronics and Components

**MAGNETI
ARELLI**

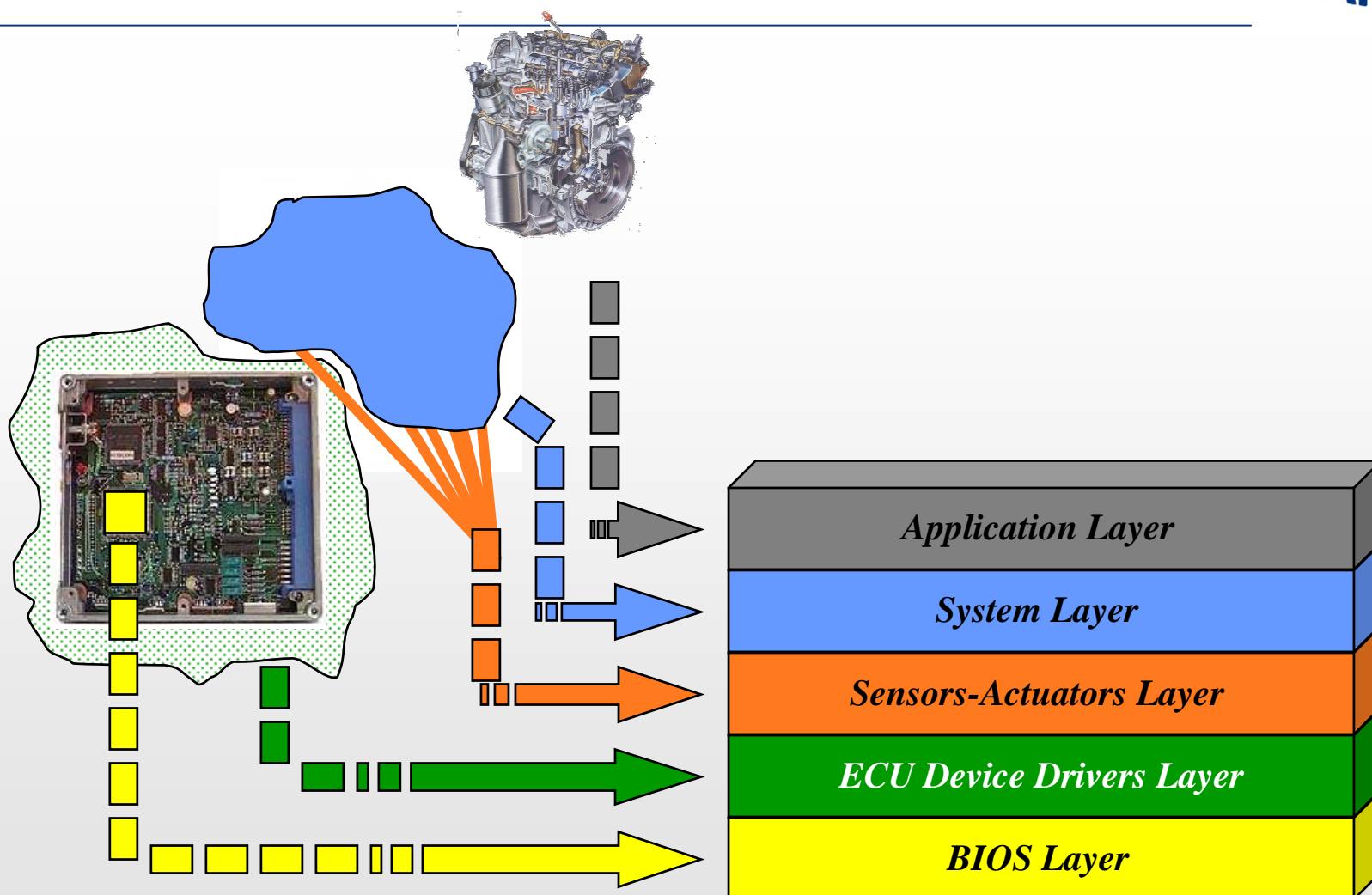


Un'esperienza di Miglioramento di Processo secondo Automotive SPICE attraverso l'Uso di Tools sviluppati ad Hoc.

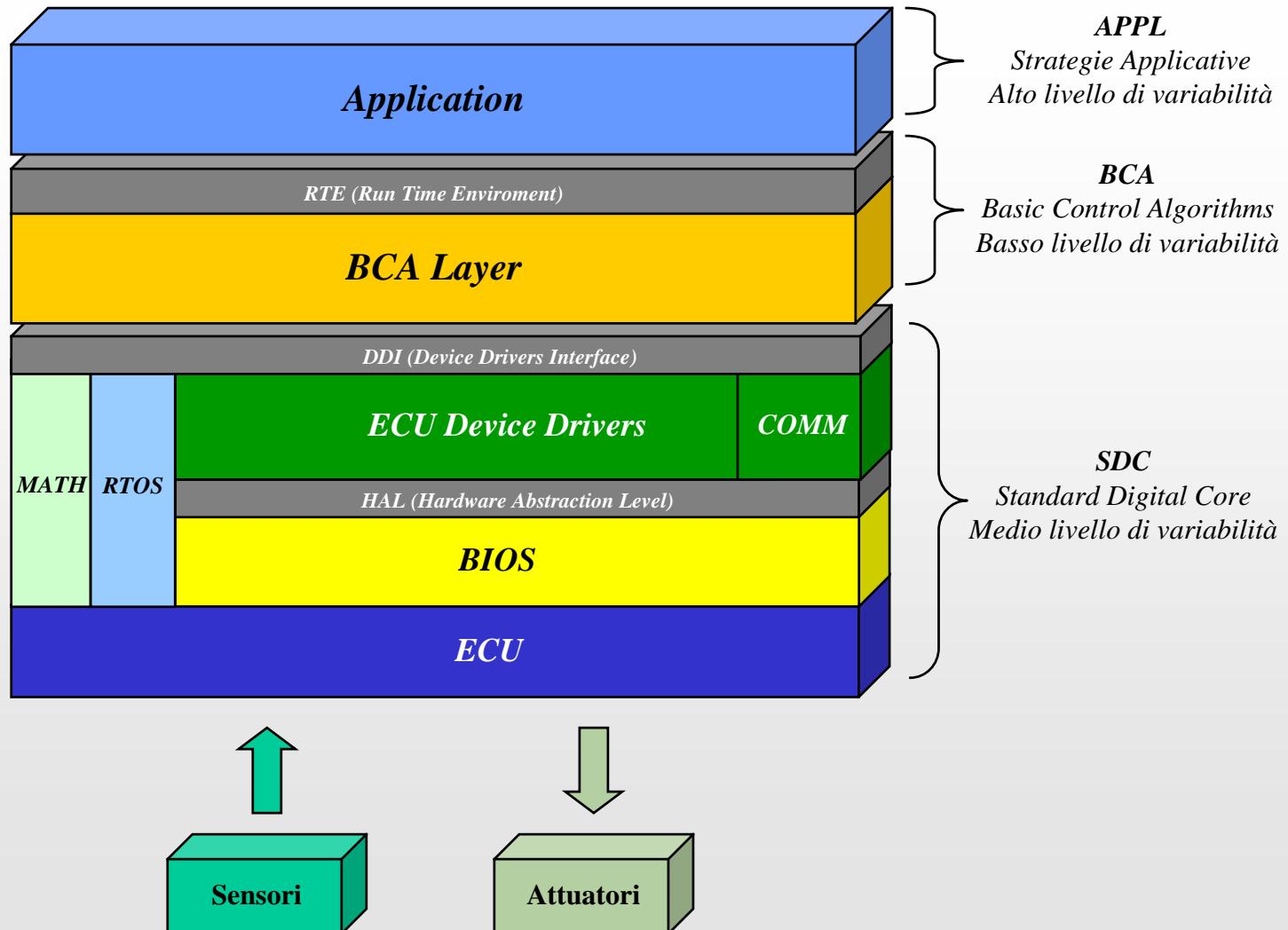
Paolo Marcea
Standard Digital Core Responsible

Architettura a layer teorica

MAGNETI
ARELLI

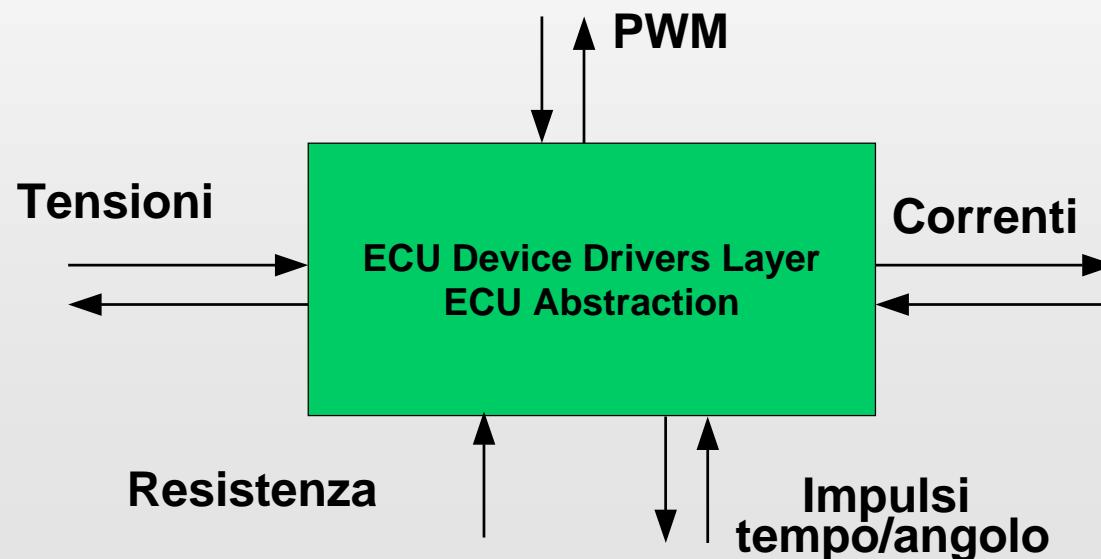


Architettura definitiva



- Si occupa delle dipendenze dal microprocessore
- E' l'unico livello software in grado di accedere direttamente alle risorse del microcontrollore

- Incapsula tutte le dipendenze dall'hardware della centralina
- Il suo perimetro di operazione è il connettore della ECU
- Tratta essenzialmente grandezze di tipo elettrico (es: tensioni/correnti) o di durata (es: tempi/angoli)

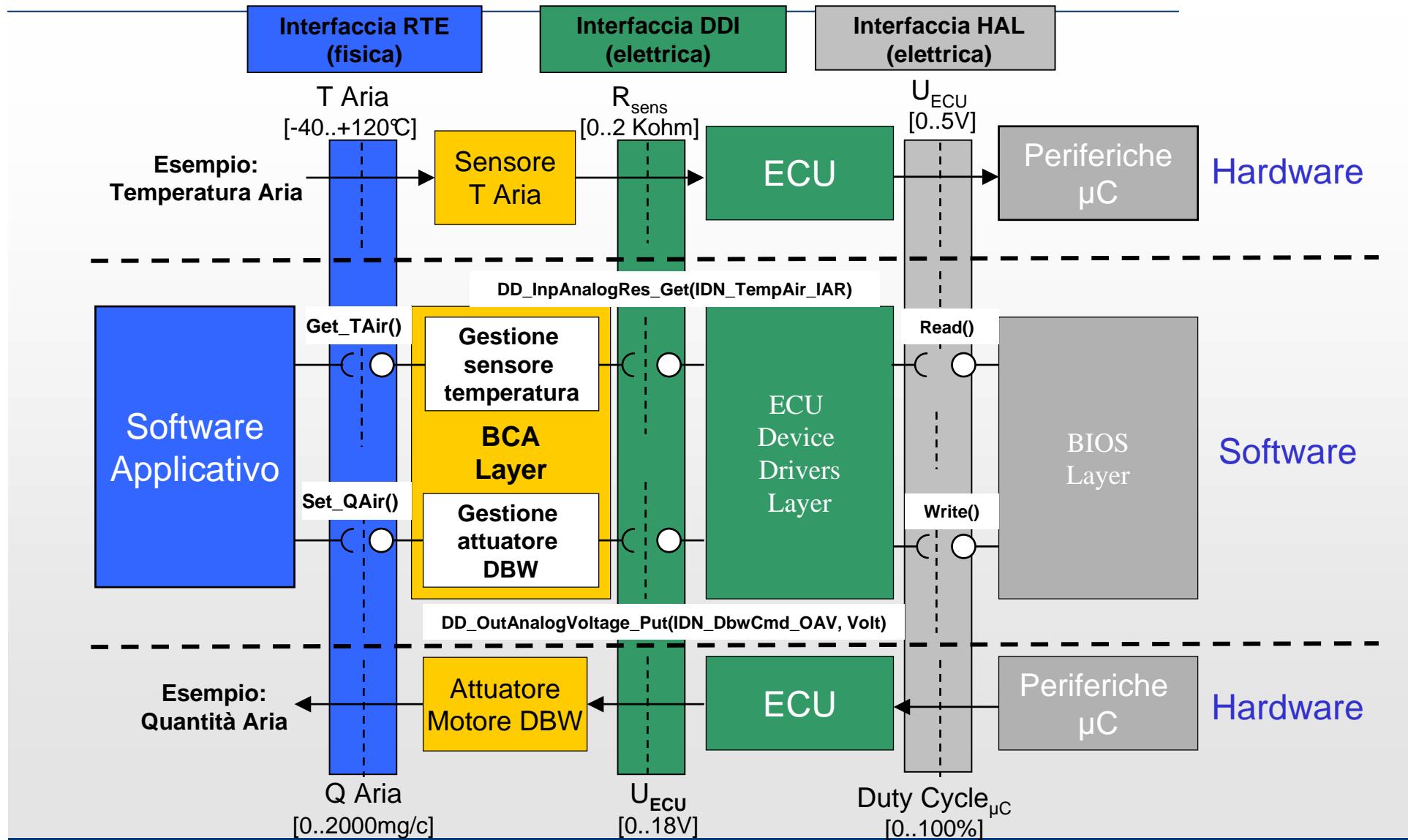


- Acquisire le grandezze analogiche, eseguirne la diagnosi elettrica, l'eventuale calibrazione hardware e la messa in scala per compensare eventuali partizioni del segnale eseguite dall'hardware. Il dato prodotto a livello DDI è la tensione vista sul pin del connettore della ECU
- Acquisire i segnali digitali lenti ed effettuarne il debouncing. Il risultato di questa operazione sarà l'informazione logica del segnale acquisito
- Acquisire i segnali in frequenza fornendo informazioni sulle loro caratteristiche fisiche come il periodo o la durata di un impulso

- Gestire le attuazioni, sia di tipo frequenziale su base tempo/angolo sia digitali lente, ed eseguirne la diagnosi elettrica
- Gestire la comunicazione SPI per la ricezione delle informazioni trasmesse dai custom, tipicamente diagnosi ed input digitali, e per la trasmissione delle informazioni verso i custom, tipicamente le attuazioni seriali

- Gestire i dispositivi fisici montati sulla ECU, come l'eeprom seriale ed i custom, che eseguono il trattamento di segnali complessi quali sonda lambda lineare e detonazione

Flusso segnali Hw/Sw ed astrazione Sw



- Esistono tipicamente 4 tipologie di operazioni:
 1. Le operazioni di configurazione (Config) del layer EDD
 2. Le operazioni di ricezione delle informazioni prodotte dal layer EDD (Get)
 3. Le operazioni di ricezione delle informazioni di diagnosi delle uscite prodotte dal layer EDD (GetDiag)
 4. Le operazioni di trasferimento delle informazioni prodotte dal layer BCA verso il layer EDD (Put)

Esempi:

```
t_MStatus DD_InpDigital_Config( t_DD_InpDigital_Idn           Idn,  
                                t_DD_InpDigital_CfgDyn *     DD_InpDigital_CfgDyn)
```

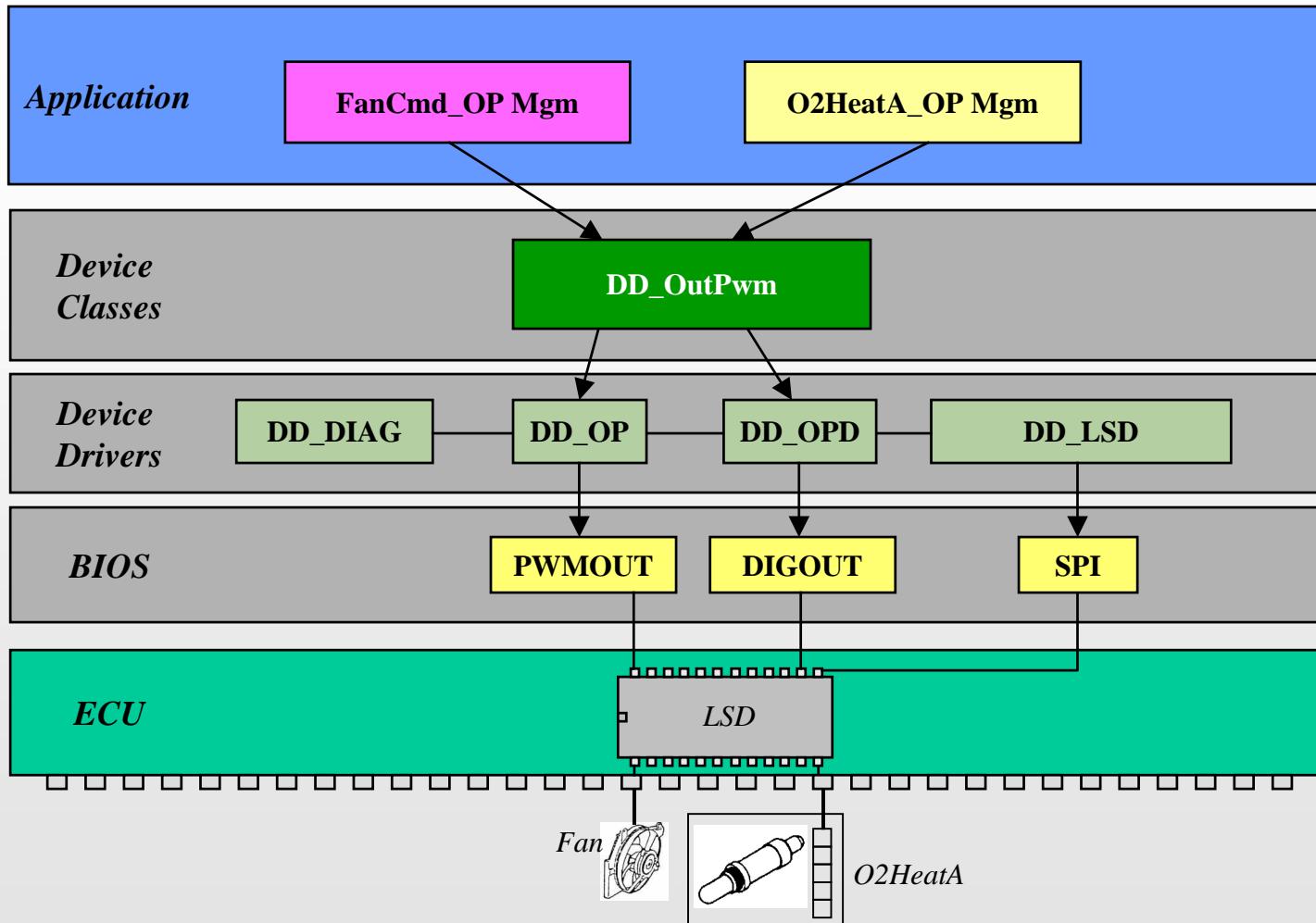
```
t_MStatus DD_InpDigital_Get(   t_DD_InpDigital_Idn           Idn,  
                                t_DD_InpDigital_Data *     Data)
```

```
t_MStatus DD_OutDigital_Put(   t_DD_OutDigital_Idn   Idn,  
                                t_switch     Cmd)
```

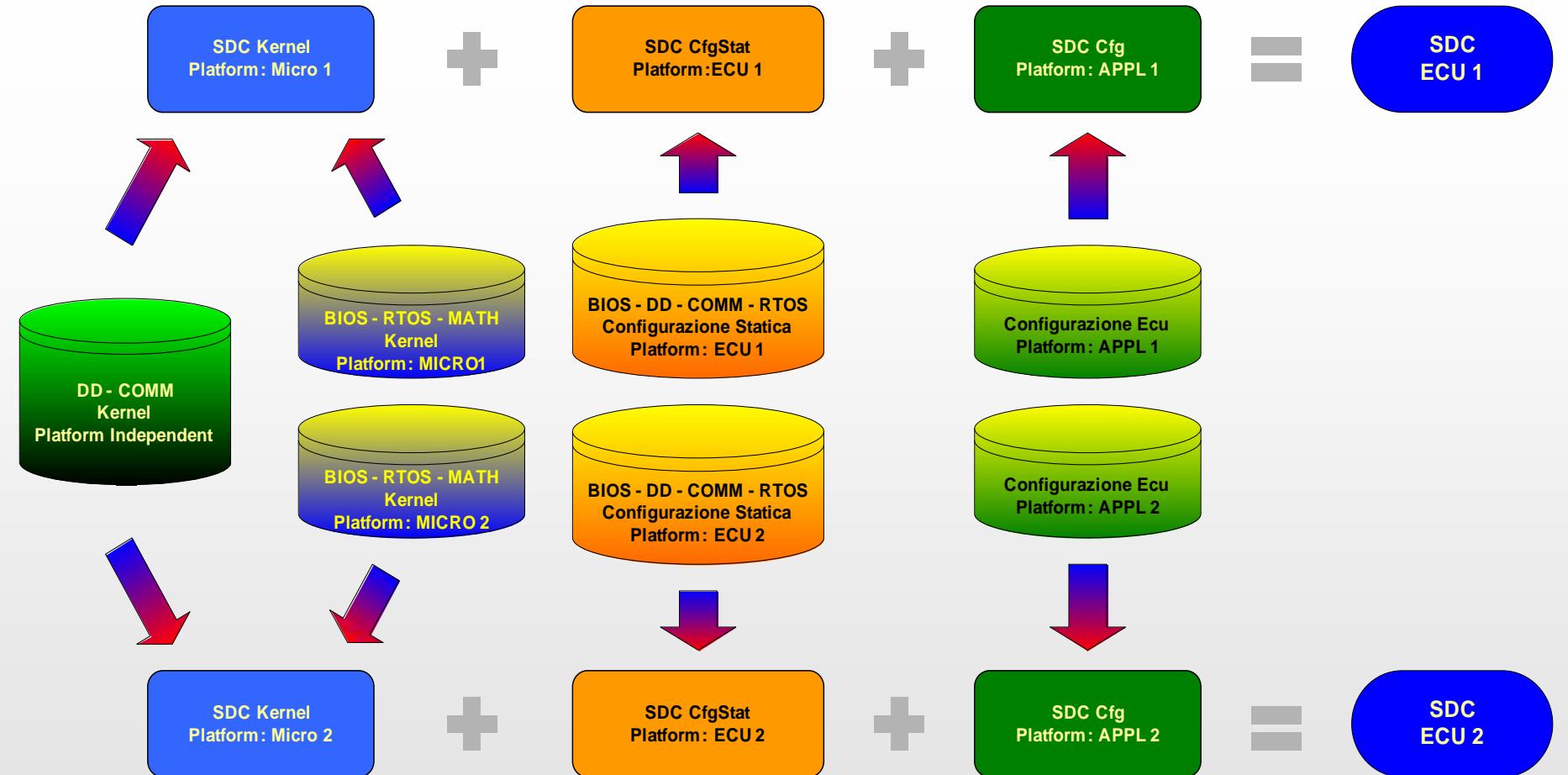
```
t_MStatus DD_OutDigital_GetDiag(t_DD_OutDigital_Idn           Idn,  
                                t_DD_OutDiag_Data *     DiagData)
```

Esempio di mapping software

MAGNETI
ARELLI



SDC- Generazione progetto



- *ISEDDA* si basa su una serie di database front-end contenenti tutte le maschere di accesso ai dati e di creazione dei report, e su una serie di database back-end che contengono tutti i dati memorizzati.
- L'utilizzo di questo ambiente, in cui è possibile accedere da qualsiasi DB front-end alle informazioni memorizzate in tutti i DB back-end, permette di evitare la duplicazione della scrittura delle informazioni ed il rapido accesso ad esse per la generazione di tutti gli output generati.

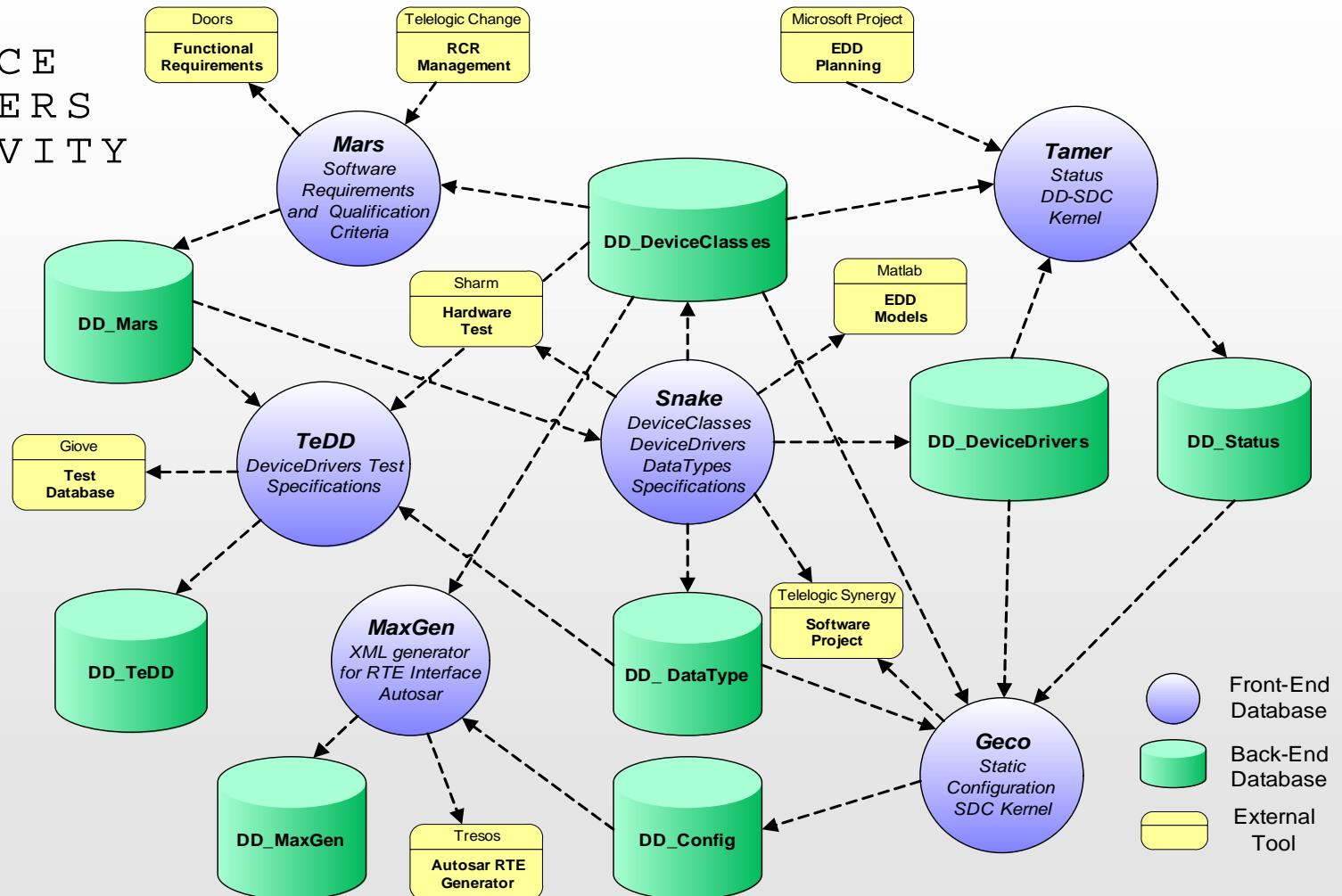
I tool (database di front-end) utilizzati nello sviluppo del progetto del layer EDD sono i seguenti:

- Mars – gestione dei requisiti software
- Snake – scrittura specifiche funzionali ed implementative
- TeDD – progettazione test layer EDD
- MaxGen – generatore file xml per interfaccia Autosar
- Tamer – gestione dello sviluppo e dei rilasci dei kernel
- Geco – gestione configurazione statica layer EDD

Una ulteriore integrazione di questo sistema si ha nell'import-export delle informazioni verso tool esterni.

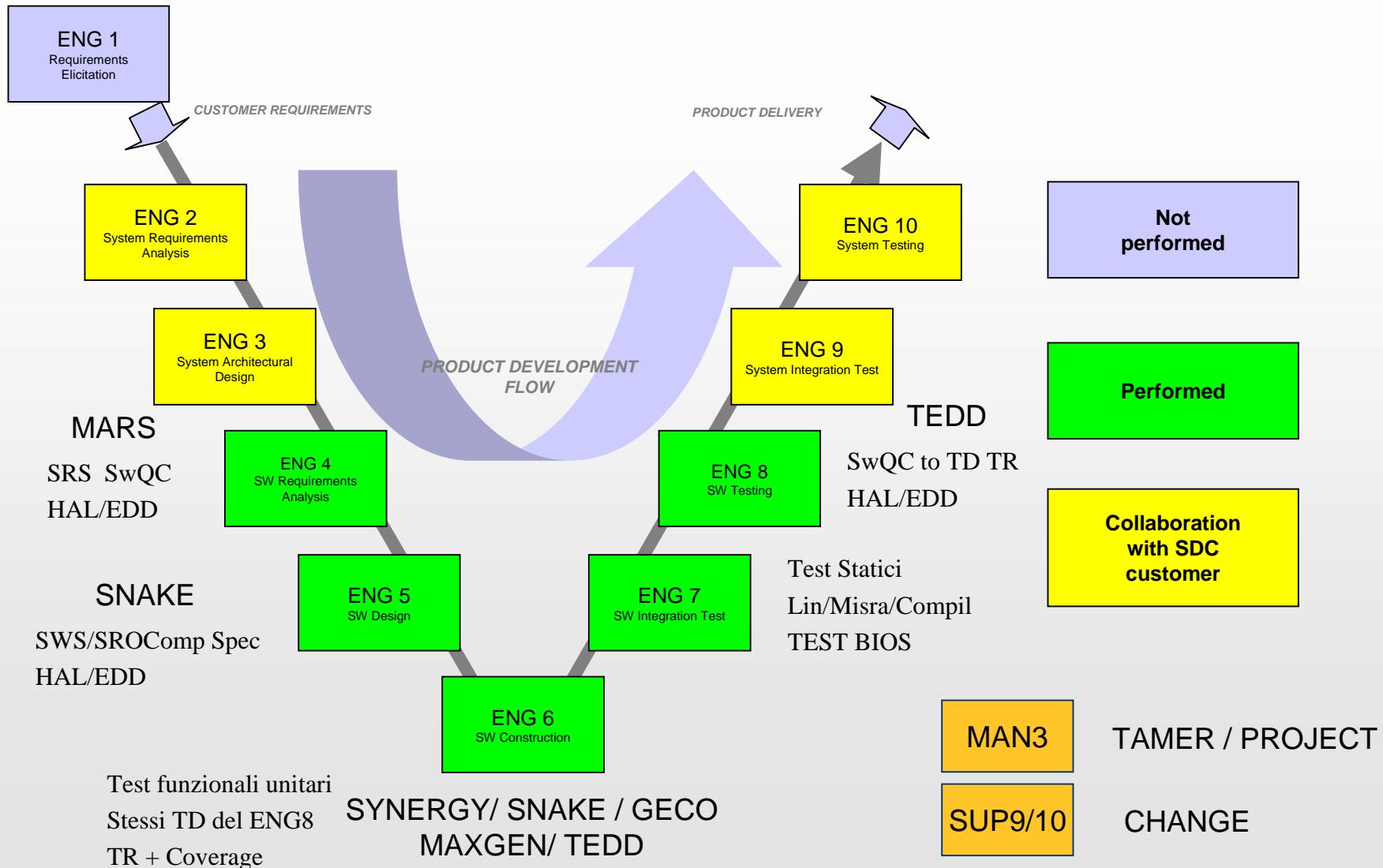
- Nello specifico è possibile trasferire informazioni verso/da:
- Doors – esportazione Software Requirements
- Microsoft Project – importazione informazioni relative alla pianificazione dei kernel EDD
- Matlab – esportazione file per la creazione dei modelli di interfaccia EDD
- Giove – esportazione verso il tool aziendale di gestione dei test
- Tresos RTE generator – esportazione dei file XML verso il tool per la generazione dell'interfaccia RTE Autosar
- Telelogic Change – importazione delle RCR (Requirements Change Request) da associare alle SRS
- Synergy – esportazione dei file sorgente, dei file di include e dei file di configurazione statica verso il Configuration Management Tool SW

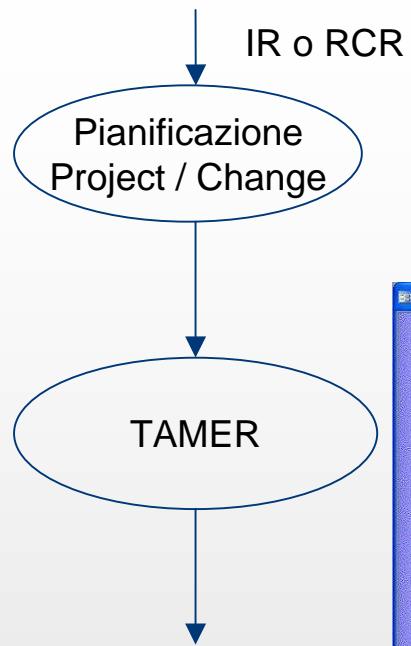
INTEGRATED SYSTEM ECU DEVICE DRIVERS ACTIVITY



- Le specifiche vengono scritte con SNAKE che aiuta, indirizza e gestisce la redazione della specifica e permette la generazione automatica della parte statica del codice e delle parti che si ripetono.
- Tutte le informazioni vengono memorizzate una volta sola in un sistema di database, che permettono il loro riutilizzo da tutti i tools, la gestione in automatico di molte fasi della progettazione dei test, della configurazione, delle interfacce verso altri tools, della pianificazione etc..
- La tracciabilità attraverso i requisiti, la specifica, il software, i Qualification Criteria ed i test è garantita dai database e da un serie di report che possono essere generati in automatico.

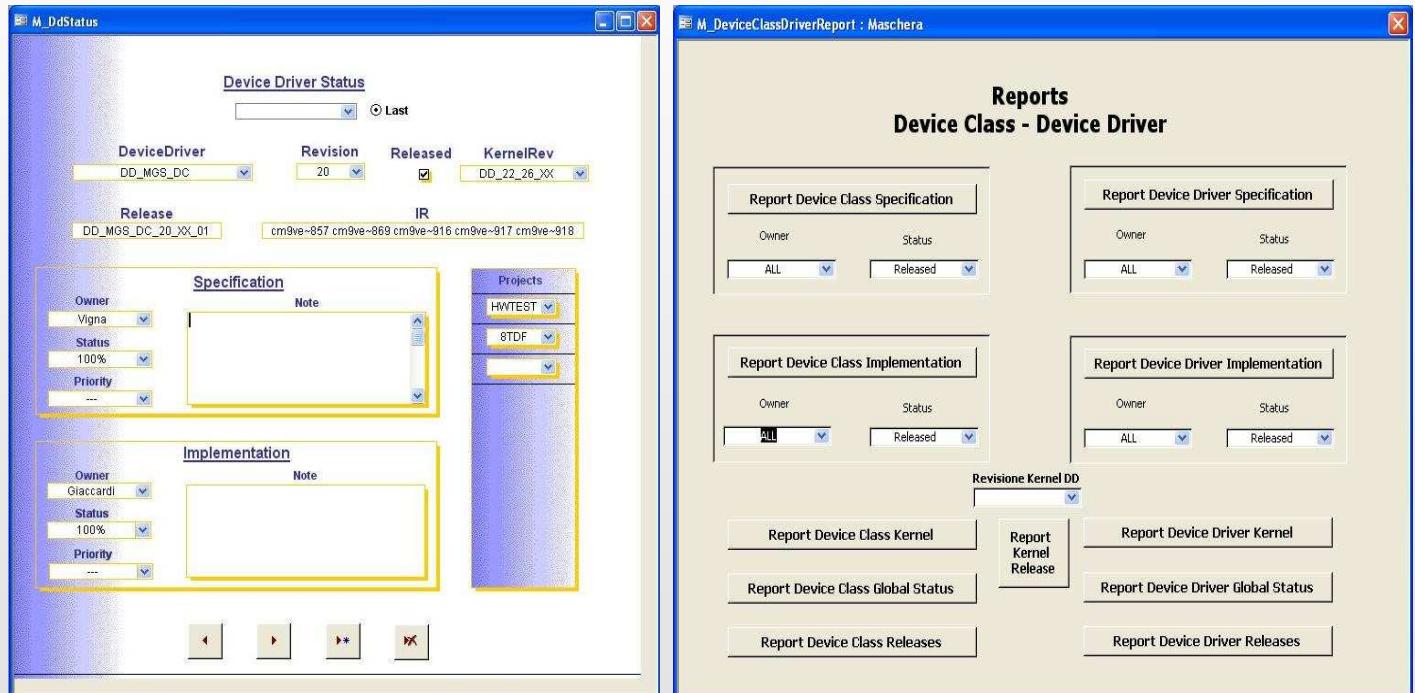
Mapping dei Tools sul processo di sviluppo

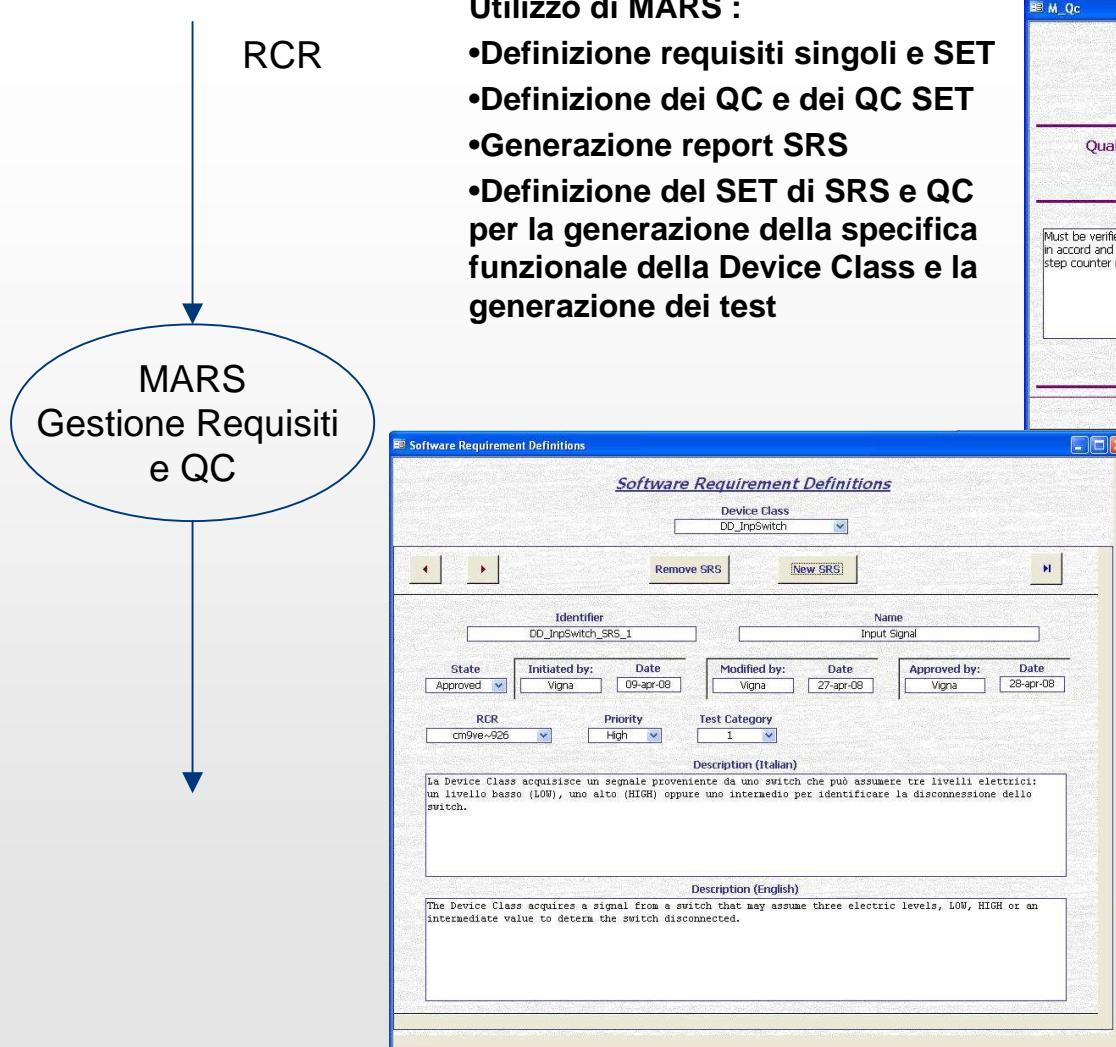


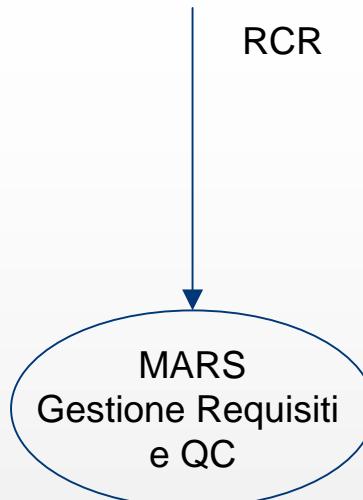


Utilizzo di TAMER :

- Report su stato avanzamento sviluppo DD, DC
- Report su contenuti dei Kernel DD e SDC
- Report sull'uso dei DD e DC nel progetti
- Report di previsione su rilasci futuri di DD e SDC Kernel







Utilizzo di MARS :

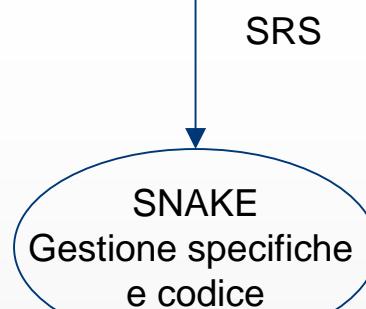
- Definizione requisiti singoli e SET
- Definizione dei QC e dei QC SET
- Generazione report SRS
- Definizione del SET di SRS e QC per la generazione della specifica funzionale della Device Class e la generazione dei test

SRS SET

DD_InpSwitch_SRS_Set,1

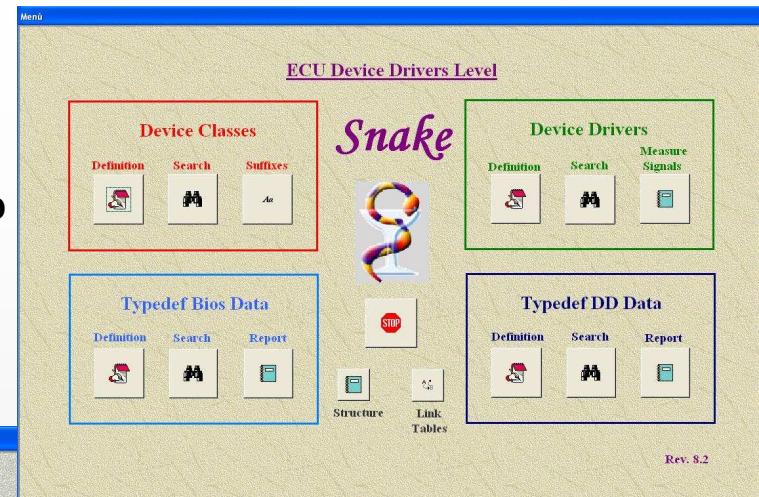
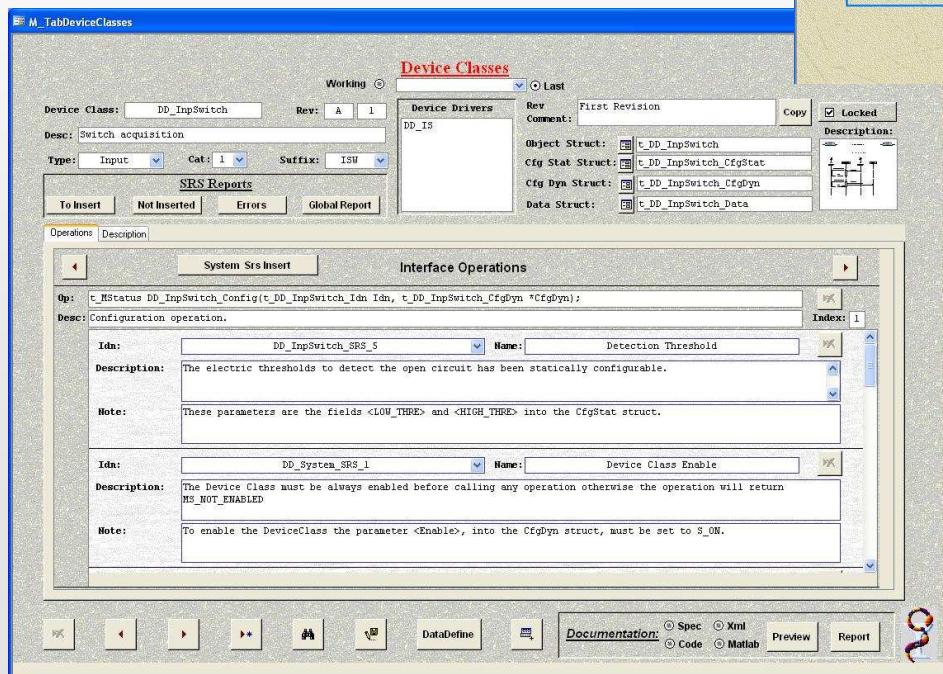
Idn	Name	Descr
DD_InpSwitch_SRS_1	Input Signal	The Device Class acquires a signal from a switch that may assume three electric levels, LOW, HIGH or an intermediate value to determine the switch disconnected.
DD_InpSwitch_SRS_2	Read Signal	The Device Class, every user request, acquire the electric input signal and returns the level.
DD_InpSwitch_SRS_3	Signal Disconnect	Without input signal (not connected) has been returned an open circuit fault information.
DD_InpSwitch_SRS_4	Read Signal Disconnected	Without input signal (not connected) has been returned the last status signal read before the fault. If is not available an acquired value will be returned the low level.
DD_InpSwitch_SRS_5	Detection Threshold	The electric thresholds to detect the open circuit has been statically configurable.

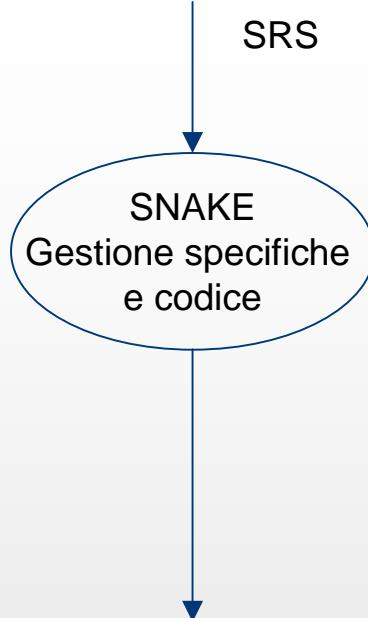
Traceability Matrix DD_InpSwitch Rev. A		Qualification Criteria									
		DD_InpSwitch		DD_System							
		Coverage	QC_1	QC_2	QC_1	QC_2	QC_3	QC_4	QC_5	QC_6	QC_7
SRS	DD_InpSwitch_SRS_1	Input Signal	Direct								
	DD_InpSwitch_SRS_2	Read Signal	Direct								
	DD_InpSwitch_SRS_3	Signal Disconnect		Direct							
	DD_InpSwitch_SRS_4	Read Signal Disconnected		Direct							
	DD_InpSwitch_SRS_5	Detection Threshold	Indirect	Direct							
	DD_System_SRS_1	Device Class Enable			Direct	Indirect	Indirect	Indirect			
	DD_System_SRS_2	Device Class Disable				Direct		Indirect			
	DD_System_SRS_3	Device Class Busy					Direct	Indirect			
	DD_System_SRS_4	Device Class Return			Indirect	Indirect	Indirect	Direct			
	DD_System_SRS_5	Object Identifier			Indirect	Direct	Direct	Direct			
	DD_System_SRS_6	Config Operation			Indirect	Indirect	Indirect	Indirect	Direct	Indirect	Indirect
	DD_System_SRS_7	Cfg Dyn Structure			Indirect	Indirect		Indirect	Direct	Indirect	Indirect
	DD_System_SRS_8	Cfg Dyn Data Access									
	DD_System_SRS_9	Read Operation			Indirect	Indirect	Indirect	Indirect	Indirect	Direct	Indirect
	DD_System_SRS_10	Output Data Structure								Direct	
	DD_System_SRS_11	Output Data Access									Direct



Utilizzo di SNAKE :

- Scrivere le specifiche funzionali di DC e implementative dei DD
- Generazione del codice di DC
- Generazione del codice statico di DD
- Generazione dei tipi dei dati





Utilizzo di SNAKE :

- Scrivere le specifiche funzionali di DC e implementative dei DD
- Generazione del codice di DC
- Generazione del codice statico di DD
- Generazione dei tipi dei dati

```

/*
 *----- SOURCE FILE NAME : DD_OAVH.c
 *----- FUNCTION NAME : DD_OAVH_Excep
 *----- CREATION DATE :
 *----- SW WRITER :
 *----- UNIPHASE :
 *----- DESCRIPTION : - Exception management, request from the device driver at the end of command period,
 *----- used for diagnosis validity. */
/*$-----$*/
void DD_OAVH_Excep(t_DD_OutAnalogVoltageH *Obj)
{
    /* Data Declaration Section */

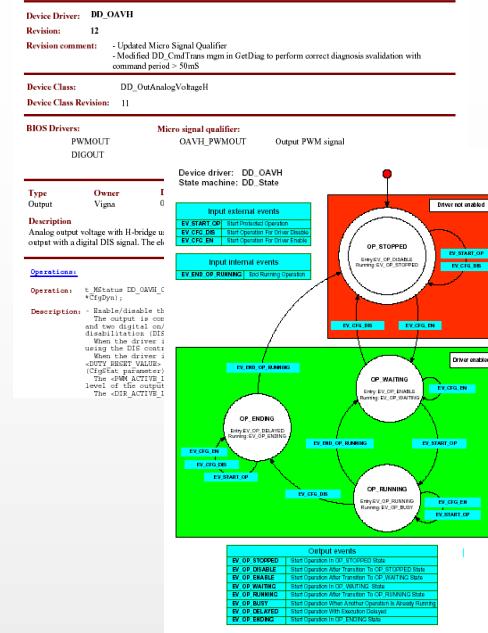
    /* Data Init Section */
    t_DD_OAVH_PrivateData *PtrPrivateData = Obj->PrivateData;
    t_DD_OAVH_CfgStat      *PtrCfgStat     = Obj->CfgStat;

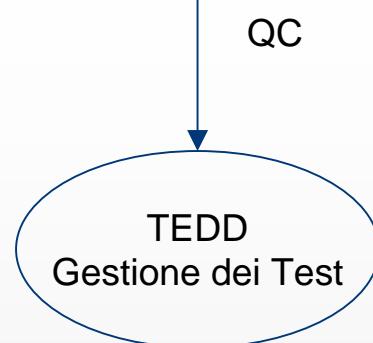
    /* Function body */
}

#ifndef DEBUG_ACTIVE
void DD_OAVH_Debug(t_DD_OutAnalogVoltageH *Obj)
{
    t_DebugData DebugData;
    t_DD_OAVH_PrivateData *PtrPrivateData = Obj->PrivateData;
    t_DD_OAVH_CfgStat      *PtrCfgStat     = Obj->CfgStat;

    DebugData.DD_Obj   = (longword)&Obj;
    DebugData.DD_State = (unsigned_word)PtrPrivateData->DD_State;
    DD_InternalDebug(&DebugData);
}

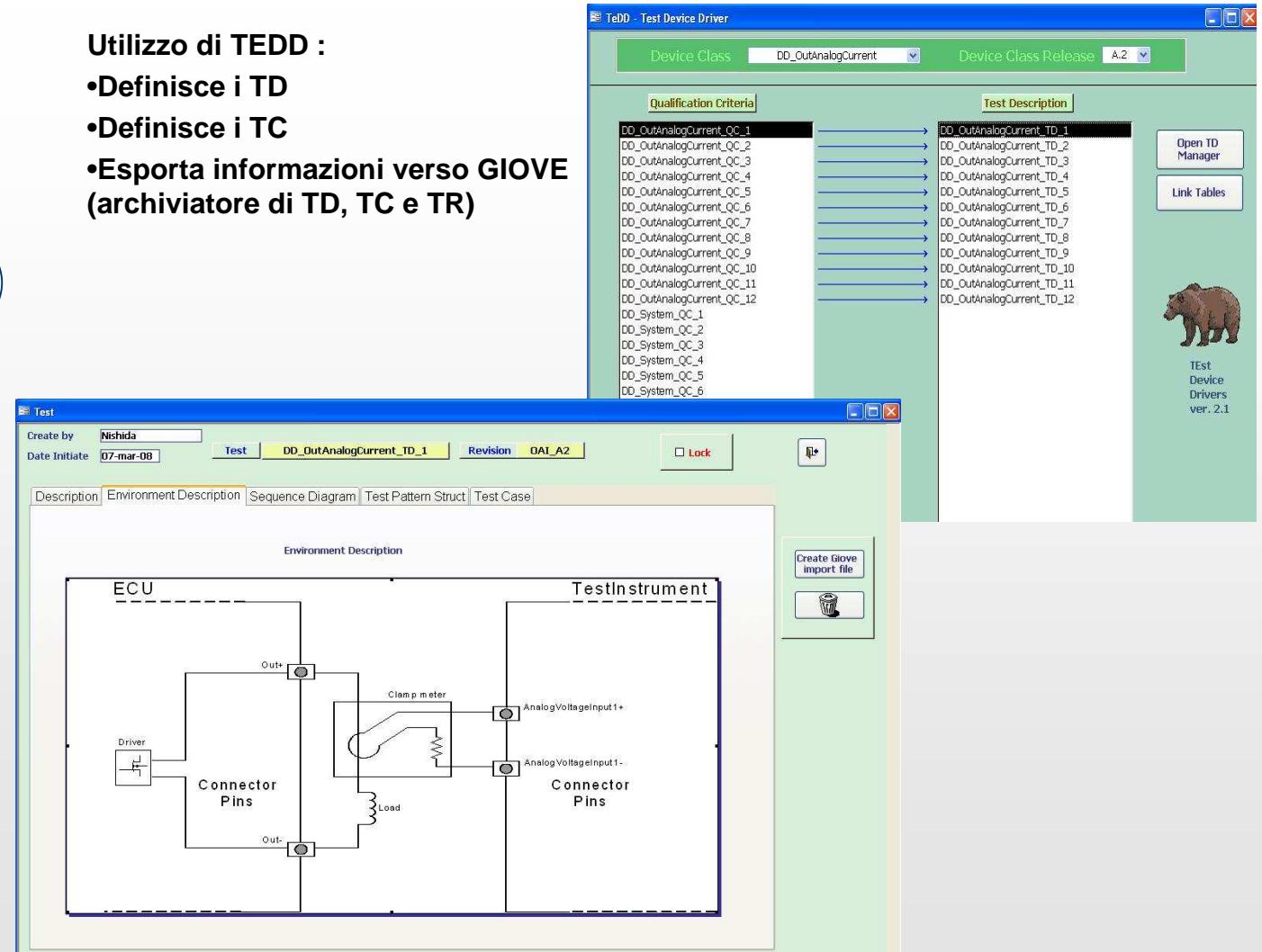
```

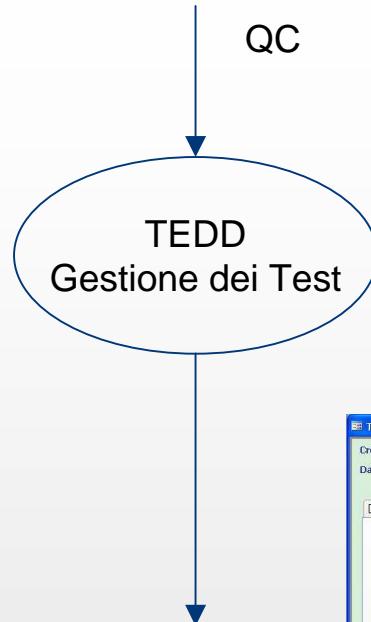




Utilizzo di TEDD :

- Definisce i TD
- Definisce i TC
- Esporta informazioni verso GIOVE (archiviatore di TD, TC e TR)





Utilizzo di TEDD :

- Definisce i TD
- Definisce i TC
- Esporta informazioni verso GIOVE (archiviatore di TD, TC e TR)

