Software: quando il ri-uso fa la differenza

Utilizzare di nuovo applicazioni o componenti già sviluppati, invece di riscrivere da capo il codice, è un'arma strategica per chi si occupa di progettazione dei sistemi embedded.
Per sfruttarla in modo adeguato occorre disporre degli strumenti giusti

Giorgio Fusari

I costi crescenti di sviluppo del software attaccano con sempre maggior forza alla base la competitività delle industrie del settore embedded, portando più che mai alla ribalta la necessità di procedure e strumenti in grado di consentire in maniera conveniente la possibilità di riutilizzo del codice.
La complessità dei moderni dispositivi e l'evoluzione dell'hardware in termini di transistor per millimetro quadrato stanno davvero facendo aumentare in modo sproporzionato e preoccupante il lavoro di sviluppo e debugging. Tanto che attualmente il codice base per un'applicazione embedded (sistema operativo, middleware e applicazione software) può spesso superare i 2 milioni di linee; e questo numero risulta essere in continua crescita.
Gli effetti di questo fenomeno si traducono nello sviluppo di una grande quantità di software, molto del quale risulta spesso duplicato. Ma soprattutto i tempi necessari per condurre questo grande lavoro finiscono per ripercuotersi sul time-to-market dell'azienda, producendo ritardi nella fase di commercializzazione dei prodotti e deteriorando la qualità degli stessi.
Tra i principali effetti si posizionano soprattutto i mancati lanci sul mercato di nuove linee di prodotti e anche l'incompletezza delle funzionalità. Come indicano numerose indagini di mercato, a causa della complessità di sviluppo del software, molti costruttori vanno fuori budget, non riescono a rispettare i ritmi del progetto o a soddisfare le aspettative sulle prestazioni dell'applicazione, abbassando di molto la loro probabilità di successo sul mercato. Inoltre la correzione di ciascun errore di codice, pur costando relativamente poco nella fase di codifica, sale poi in maniera sostanziosa durante le verifiche funzionali, e poi ancor di più se le modifiche avvengono in fase di produzione o addirittura dopo che il prodotto è già stato rilasciato sul mercato.

Massimo Porta, direttore generale di Telelogic

Laurent Accard, general manager Enea France in Enea Embedded Technology

Ricreare da zero componenti software o applicazioni ogni volta che si introduce una nuova linea di prodotti è dispendioso è non più ipotizzabile in aree come l'automazione industriale, il settore automotive o anche i prodotti elettronici destinati all'ambito consumer. In ciascuno di questi comparti la produzione del software, pur essendo ancora frutto dell'elaborazione dell'intelletto e dell'ingegno umano, è sempre più industrializzata e soggetta ai ritmi di creazione e immissione sul mercato dei nuovi prodotti.
Sviluppo più veloce, software di maggior qualità e costi più contenuti sono fra i principali benefici di riutilizzo del codice. "L'importanza del riuso del software – spiega Massimo Porta,

direttore generale di Telelogic – è rappresentata dal fatto che la rapidità di modellazione e simulazione è essenziale per essere competitivi, e ciò vale sia per quanto riguarda il settore embedded che per altre aree". Ma nonostante queste tecniche aiutino fin dalle fasi preliminari del progetto a riutilizzare quanto già sviluppato e a reimpiegare i modelli di codice su target diversi, esistono tuttora ostacoli che in varie aziende rallentano l'adozione di questa filosofia di lavoro. "Le barriere – aggiunge Porta – possono essere costituite dall'approccio metodologico e culturale di ogni impresa; in quelle piccole la cultura dell'investimento ai fini di crescita metodologica si deve ancora affermare".

Non per tutti

Fermo restando che il riuso del codice, in generale, deve rappresentare la norma e va applicato in azienda come una strategia, anche Laurent Accard, general manager di Enea France in Enea Embedded Technology, precisa come oggi, in realtà, ciò non possa dirsi vero per tutte le aziende: "La dimensione dell'impresa, la sua organizzazione corrente in termini di processi e strumenti di sviluppo del software, nonché il prodotto finale hanno una forte influenza su come questo obiettivo può essere raggiunto".

Nelle diverse aree di attività (automazione industriale, automotive, transportation, ecc.) ciascuna azienda può infatti avere esigenze diverse, a seconda della pressione di mercato che sperimenta, dei vincoli di progettazione che deve rispettare (hardware standard o specifico, requisiti real-time 'hard' o 'soft', ecc.) e del ciclo di vita dei propri prodotti. "Inoltre – aggiunge Accard - il fattore 'dimensione dell'azienda' è ugualmente determinante, non tanto perché riutilizzare il codice è meno utile per una piccola impresa, ma perché l'effetto del riutilizzo sarà molto più 'visibile' in

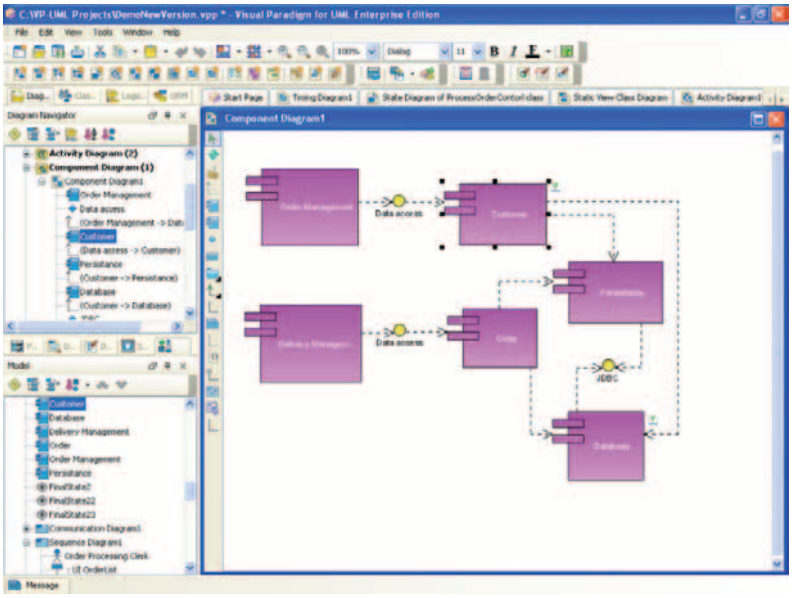
Attrezzature di tlc: il riuso è la regola

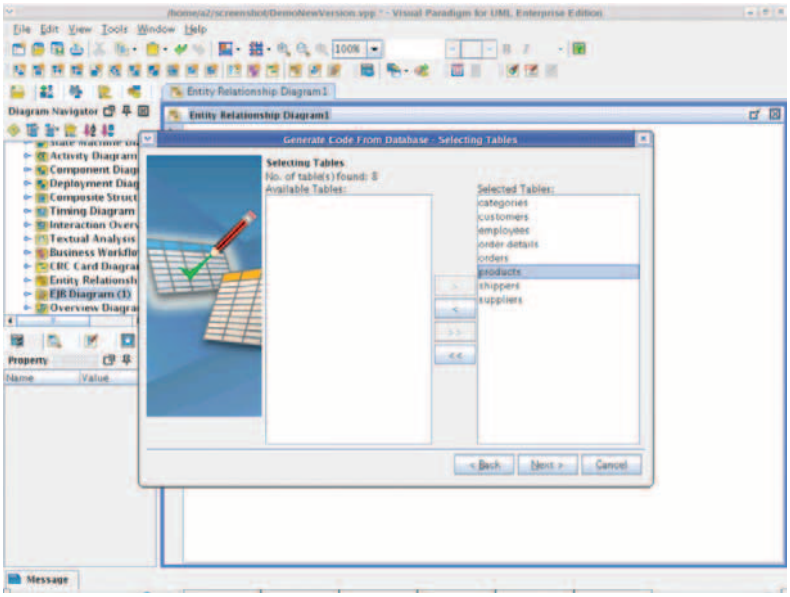
Un settore in cui il riutilizzo del software rappresenta una strategia particolarmente importante è quello dei costruttori di attrezzature di telecomunicazioni, che forniscono un'ampia gamma di prodotti fra cui switch, router e dispositivi di accesso alla rete. Per essere sempre competitive, queste aziende devono seguire in maniera permanente le più recenti tecnologie, innovare per mantenere un alto valore aggiunto, tagliare i costi utilizzando componenti hardware e software di tipo COTS (Commercial Off The Shelf) e ridurre i cicli di sviluppo al fine di rispettare i vincoli di time-to-market. Per queste imprese il riuso del codice è quindi un elemento obbligatorio del processo di sviluppo del software.

Enea Embedded Technology dice di avere, fra i propri utenti, vari esempi 'real life' di riutilizzo del software, molti dei quali provengono dall'area delle infrastrutture di telecomunicazioni o dai costruttori di terminali (handset) wireless. "Queste aziende – commenta Laurent Accard, general manager in Enea France – hanno chiaramente realizzato una strategia di riutilizzo del codice a tutti i livelli. Nei loro processi di sviluppo, 'riusabilità' è una parola chiave che guida la progettazione di ogni componente software". Ci sono invece settori come l'automazione industriale o aziende di questa sfera in cui il riutilizzo del codice avviene solo in modo parziale. Vi possono essere ad esempio il riuso dello stesso algoritmo di un Dsp (Digital signal processing), la migrazione di parte del codice da un Dsp a un microcontroller o anche la migrazione di una parte di codice da un sistema Rtos (Real-time operating system) a un altro.

In Siemens A&D un repository comune per il codice

Risale a novembre 2005 l'annuncio da parte di Siemens Automation and Drives (A&D) dell'acquisizione di una soluzione per lo sviluppo di software embedded 'safety critical', necessario al funzionamento dei prodotti, sistemi e soluzioni che l'azienda indirizza a svariati settori industriali, fra cui le aziende manifatturiere, le tecnologie per impianti elettrici e per l'automazione dei processi negli impianti chimici o nelle unità produttive di veicoli. Come ha spiegato Peter Weichhold, software engineer senior di Siemens A&D, l'obiettivo con l'uso della nuova soluzione (ARTiSAN Studio, di ARTiSAN Software Tools) è stato sviluppare tutti i prodotti seguendo lo standard europeo IEC 61508 per i sistemi safety-critical. Ma alla società è stata di notevole ausilio anche la capacità di supportare con flessibilità il processo di certificazione dei prodotti, attraverso la generazione automatica dai modelli Uml (Unified modeling language) dei documenti desiderati. Ciò ha contribuito molto a migliorare la comunicazione fra i progettisti e l'organismo di certificazione. "Dato che il nostro attuale progetto di sviluppo coinvolge team distribuiti in siti diversi, la possibilità di condividere e scambiare l'informazione dei modelli aiuta i nostri team a collaborare in modo veramente efficiente" ha aggiunto Weichhold. E questo grazie anche alla strategia di sviluppo collaborativo del software embedded, facilitato dalla presenza di un repository multi-utente.





Due schermate, in ambiente Windows e Linux, dello strumento di modellazione Visual Paradigm per Uml, di Visual Paradigm International

un'azienda che per esempio ha venti linee di prodotto diverse e svariati progetti in parallelo, piuttosto che in un'azienda che ha solo una linea di prodotti e in cui i vantaggi del riutilizzo del codice appaiono soltanto durante la progettazione del prodotto di nuova generazione".
Accard parla anche di aspetti tecnologici che talvolta rendono difficoltoso applicare i processi di riutilizzo del codice. I sistemi embedded sono infatti tipicamente composti da una piattaforma hardware (unità di elaborazione costituite da processori, microcontroller, Dsp, ecc.), da un sistema operativo (sistemi proprietari sviluppati 'in house', Rtos – Real-time operating systems commerciali, sistemi Linux, ecc.), da servizi aggiuntivi (stack Tcp/Ip, file system, ecc.) e dal codice dell'applicazione. Inoltre tutti questi componenti sono molto spesso accuratamente ottimizzati per un utilizzo specifico e strettamente integrati per ragioni di dimensioni del codice e massimizzazione delle prestazioni: "Al fine di ottenere il riutilizzo del codice dell'applicazione, qualunque siano l'hardware o il sistema operativo, la sfida è scrivere codice in grado di essere completamente portabile e indipendente dagli strati software sottostanti".
Non tutti i sistemi embedded possono poi, per esempio, utilizzare la piattaforma Linux come sistema operativo, a causa di alcune limitazioni (supporto del processore, potenza della memoria, prestazioni, tempi di avvio, ecc.). Inoltre, in confronto agli Rtos commerciali, vi sono ovvie differenze in termini di Api (Application program interface) e di device driver. In questo caso allora il problema può essere parzialmente risolto con un lavoro di porting basato sull'utilizzo di linguaggi ad alto livello (ad esempio SDL - Specification and Description Language), che attraverso tool appropriati consentono di generare modelli per uno o l'altro Rtos.

Alto livello di astrazione

Per sviluppare meglio il software e riuscire a riutilizzarlo con profitto occorrono procedure e strumenti sofisticati. La struttura o, meglio, l'architettura del software deve essere definita con sufficiente chiarezza, in modo da consentire a chi si occupa di aggiornare e migliorare le applicazioni di trovare e risolvere rapidamente i bug e i difetti lasciati dagli autori originali. Inoltre se l'architettura è modulare, il progettista potrà raccogliere tali componenti in una libreria di modelli, da riutilizzare ogni volta che in un'altra applicazione siano richieste le medesime funzionalità. Al

La home page dell'OMG (Object Management Group)

momento della codifica lo sviluppatore non deve far altro che importare il modulo di codice all'interno dell'applicazione.
Per ottenere una buona architettura, la modellazione, cioè la fase di progettazione delle applicazioni software prima della codifica, è un momento essenziale, specialmente nei grandi progetti.
Utilizzando modelli che consentono di lavorare a un più alto livello di astrazione, i responsabili del successo del progetto di sviluppo del software possono assicurarsi che le funzionalità siano complete e corrette, che le esigenze degli utenti finali siano soddisfatte e anche che siano rispettati i vari requisiti di scalabilità, robustezza, sicurezza ed espandibilità dell'applicazione. E naturalmente tutto ciò prima di passare a fasi più avanzate del progetto, in cui apportare correzioni sarebbe troppo difficoltoso o costoso.
La modellazione consente, attraverso opportune rappresentazioni di tipo grafico, di focalizzare l'attenzione sui diversi aspetti del prototipo, osservando in modo puntuale il comportamento dell'applicazione e visualizzando le sue connessioni con le altre.
In particolare, i linguaggi di modellazione come il già citato Sdl, i linguaggi UML (Unified Modeling Language) e SysML (Systems Modeling Language) o gli strumenti di sviluppo ad alto livello (un esempio può essere Visual Paradigm for Uml, di cui si possono osservare due schermate in queste pagine) aiutano a semplificare questo genere di attività.

Da Proven Software un design più 'intelligente'

Per aiutare le industrie a migliorare lo sviluppo del software embedded attraverso il riutilizzo del codice, che consente di ridurre notevolmente sia il time-to-market, sia i costi e i rischi di sviluppo, la startup inglese Proven Software Solutions (Pss) ha annunciato in dicembre 2005 la strategia di voler diventare il primo broker indipendente dell'industria elettronica nello spazio del software embedded. L'obiettivo della società è cambiare il modo in cui il software viene sviluppato, fornendo a qualunque progettista non solo la possibilità di diminuire i rischi di design, ma anche di gestire la complessità e di attuare l'ottimizzazione del software dei sistemi embedded (Device software optimization, Dso). La quantità di software provato, affidabile e documentato che può essere riutilizzato cresce ogni giorno, così Pss ha pensato di realizzare un framework e metodologie per consentire agli Oem (Original equipment manufacturer), agli Odm (Original device manufacturer) e alle altre parti in gioco di contrattare, attraverso un sistema di licenze, il riuso del software anziché re-svilupparlo. Alla base del framework di Pss esiste infatti un repository di software già provato e affidabile, nel quale Oem e Odm possono scaricare, archiviare e registrare il loro software proprietario, mettendolo a disposizione per un eventuale riutilizzo. Se il licenziatario del software dovesse richiedere supporto oltre alla documentazione fornita, e il licenziante non desidera fornirlo, Pss è in grado di offrire tali servizi attraverso la propria rete di supporto. In tal modo il pieno servizio di assistenza rende il software più attrattivo per chi lo acquisisce, sgravando al tempo stesso da tale onere il fornitore del codice.

Definito dalle specifiche dell'OMG (Object Management Group), Uml è indipendente dalla metodologia di analisi e progettazione e, utilizzando XMI (Xml Metadata Interchange), un altro standard Omg, permette di trasferire in un repository i modelli creati con un determinato tool software oppure di elaborarli con altri strumenti, per effettuare ad esempio operazioni di rifinitura del codice.
SysML è in sostanza un linguaggio visuale che estende le funzionalità di Uml 2.0, permettendo ai progettisti di eseguire anche attività di specifica, analisi, design, verifica e validazione di sistemi complessi, costituiti da numerosi componenti (hardware, software, informazioni, processi, ecc.). È complementare a Uml 2.0 essendo in grado di riutilizzare un sottoinsieme dei suoi diagrammi, arricchendoli con altri schemi di modellazione. Ciò consente di aumentare il livello di standardizzazione, l'accessibilità e la condivisione delle informazioni, permettendo di sviluppare una maggior collaborazione fra coloro che partecipano al processo di sviluppo dei sistemi e facilitando l'interoperabilità fra i tool di modellazione.
Nel novembre 2005 si è arrivati ad apportare nuovi miglioramenti al linguaggio SysML, attraverso la pubblicazione di una versione completa (SysML v.1.0 alfa) e disponibile sul sito Web del SysML Open Source Project, del quale sono partner produttori di tool come ARTiSAN Software, EmbeddedPlus Engineering, I-Logix, IBM, Telelogic, ma anche società e organizzazioni come Motorola e Nasa (National aeronautics and space administration). Tutti hanno collaborato strettamente nel progetto.

	readerservice.it
ARTiSAN Software	n. 44
EmbeddedPlus Engineering	www.embeddedplus.com
Enea Embedded Technology (Salesteam)	n. 45
IBM	n. 46
I-Logix (TESS-COM Italia)	n. 47
Motorola	n. 48
Nasa	www.nasa.gov
OMG (Object Management Group)	www.omg.org
Proven Software Solutions (Pss)	www.proven-software.com
Siemens A&D	n. 49
SysML Open Source Project	www.sysml.org
Telelogic	n. 50