Dal software di controllo all'Automation Component

Luca Ferrarini, Alessio Dedè

L'aumento della complessità dei sistemi d'automazione più moderni richiede una profonda revisione delle tecniche di progetto e sviluppo. In particolare si rende necessario, a causa della moltitudine di dispositivi eterogenei che si trovano a coesistere, definire un metodo formale basato su modelli in grado di astrarre la fase di progetto da quella di implementazione. Questo articolo presenta i risultati ottenuti nel campo della progettazione model-based per sistemi embedded, introducendo i modelli ed i concetti definiti dal gruppo di ricerca del progetto europeo Medeia.

I sistemi industriali automatizzati hanno visto nell'ultimo decennio una crescita esponenziale della loro complessità, per rispondere ad una domanda sempre più esigente e variegata, nell'ottica dell'allargamento su scala globale dei mercati e della "customizzazione di massa". In particolare, l'esigenza di far fronte a rapide variazioni della domanda e quindi della produzione ha portato alla progettazione ed alla realizzazione di impianti sempre più flessibili e con un elevato grado di integrazione di sistemi diversi. In questo scenario differenti sotto sistemi spesso assai eterogenei tra loro devono coesistere e collaborare per raggiungere l'obiettivo fissato dalle specifiche iniziali. Questa eterogeneità nei moderni impianti è supportata anche dalla nascita di nuovi dispositivi autonomi, in grado cioè di elaborare informazioni e di comunicare con il mondo circostante; l'impianto diventa quindi un insieme ed una rete di tanti differenti attori che interagendo tra loro giungono all'obiettivo finale.

L'aumento di complessità dell'hardware dei sistemi automatici odierni ha portato a un inevitabile aumento di complessità anche nelle architetture del software di controllo. In questo caso algoritmi di controllo e strutture dati modellate in fase di specifica devono poi essere implementate e spesso distribuite su piattaforme differenti, rendendo particolarmente complicata ed onerosa la loro implementazione finale. Questa fase di traduzione porta talvolta a commettere errori particolarmente complessi da individuare e risolvere poiché dipendenti dalla specifica piattaforma e quindi non preventivabili o riconoscibili in fase di specifica.

Tutti questi fattori, necessari per la realizzazione di impianti moderni e competitivi, portano però ad un aumento dei costi di progettazione, sviluppo e messa in opera di questi nuovi sistemi di produzione industriali. Per contenere i costi di sviluppo può anche capitare, se pur raramente, che le specifiche iniziali di progetto risultino così onerose da rispettare che si preferisce rivederle, ottenendo così una parziale risposta al bisogno espresso dal committente che impone ovviamente anche un limite di costo sul sistema finale.

Maggiore flessibilità e riduzione dei costi sono quindi obiettivi contrapposti ma che sempre più in futuro saranno i principali protagonisti nello sviluppo dei sistemi di produzione. Per questo motivo diventa necessario rivedere e riorganizzare le metodologie per la progettazione e la messa in opera dei sistemi automatici industriali, con l'obiettivo di fornire strumenti utili all'integrazione di sistemi eterogenei, alla loro modellistica e, pensando ai dispositivi autonomi accennati in precedenza, al riutilizzo delle architetture, degli algoritmi e dei modelli presenti in libreria rendendoli indipendenti dalla particolare piattaforma.

L'impatto della fase di progettazione sui costi finali di un impianto è caratterizzato principalmente dalla complessità del sistema, data a sua volta dalle specifiche imposte, dalla dipendenza dal particolare dominio e piattaforma e dai tempi di progetto e successivamente di realizzazione. Diventa quindi necessario definire le specifiche del sistema senza pensare all'implementazione e alla piattaforma che verrà utilizzata, esprimendo ciò che si vuole ottenere e non come lo si vuole ottenere. Le specifiche vengono così definite in modo completamente indipendente dalla realizzazione finale del sistema. Infatti con l'idea di riutilizzare algoritmi e soluzioni adottati questi devono essere specificati in un linguaggio di alto livello che non dipende ovviamente dalla particolare piattaforma scelta, ma che è in grado di definire modelli più astratti per il controllo e per l'impianto che verranno quindi implementati a seconda del particolare dominio su una particolare piattaforma. Il riuso e la creazione di librerie di componenti è un altro fattore in grado di determinare una riduzione dei tempi e quindi dei costi di progettazione. L'idea di riusare soluzioni già sviluppate e testate, e di creare gruppi di modelli applicabili in differenti scenari sono concetti definiti e descritti in modo particolare nei linguaggi di programmazione orientati agli oggetti (OO – Object Oriented). Tuttavia, soluzioni analoghe possono essere impiegate anche per la gestione di modelli hardware, software e comportamentali nell'ambito dei sistemi

L. Ferrarini, A. Dedè - Dipartimento di Elettronica e Informazione del Politecnico di Milano

automatici industriali.
La fase di progettazione di sistemi complessi come quelli qui considerati vede l'obbligo di integrare tra loro differenti tecniche ciascuna delle quali è legata ad uno specifico dominio. Diventa così importate l'integrazione e la gestione di queste tecniche per sfruttare in modo corretto i vantaggi di ciascun dominio riducendo il tempo per lo scambio e la comprensione delle informazioni tra domini eterogenei.
Una delle complessità principali nella progettazione di sistemi di automazione industriali diventa quindi l'integrazione di diverse tecniche per la specifica alle quali si aggiungono poi differenti metodologie implementative che dipendono dal particolare caso trattato.

L'approccio Medeia

Per superare le limitazioni imposte dall'elevato numero di possibili metodi modellistici ed implementativi coinvolti nella progettazione e realizzazione di un sistema industriali automatizzato il progetto Medeia vuole proporre un nuovo approccio alla progettazione basata sul concetto di meta-modello e su un'architettura di meta-design, formalizzando le fasi di specifica e implementazione e frapponendo ad esse un meta-modello indipendente da entrambe. Il progetto Medeia (Model-Driven Embedded Systems Design Environment for the Industrial Automation Sector) è supportato e finanziato dalla commissione europea nell'ambito del settimo Research Framework Programme (FP7), ha una durata di 3 anni (2008-2010) e vede il contributo idi importanti aziende europee: MCM SpA (Machining Centre Manufacturing) di Vigolzone (PC), società italiana impegnata nella produzione di centri di lavorazione automatizzati, Schunk GmbH & Co. KG, azienda tedesca produttrice di robot industriali e specializzata nei sistemi di serraggio pezzi, EDF (Électricité de France), primo gestore e produttore per l'energia elettrica francese, LogiCals, società austriaca di informatica e consulenza sviluppatrice di applicativi per l'automazione industriale. Completano il team di lavoro importanti università tecniche e centri di ricerca: Profactor GmbH, società di ricerca austriaca e leader del progetto, Technical University of Vienna, la Scuola Universitaria Professionale della Svizzera Italiana (SUPSI) di Lugano e il Politecnico di Milano tramite il Dipartimento di Elettronica e Informazione
Il principale scopo del progetto è quindi quello di definire un metodo formale basato su modelli impiegato per la definizione e lo sviluppo di sistemi di controllo in grado di integrare dispositivi embedded eterogenei. Incentrato sul modello dell'Automation Component (AC), capace di rappresentare sia parti hardware che software, l'approccio Medeia ha l'intento di definire le regole per utilizzare ed assembleare tra loro differenti AC di differenti produttori, e ovviamente di stabilire come specificarne e definirne di nuovi. L'assemblaggio ed il riutilizzo di AC immagazzinati in librerie riduce drasticamente il tempo di progettazione e di sviluppo e la possibilità di commettere errori. L'obiettivo è quindi quello di arrivare a definire e modellare un impianto come un insieme di AC già sviluppati e testati in precedenza connessi tra loro.
Un secondo obiettivo del framework definito in Medeia è quello di essere facilmente compreso ed utilizzato da esperti di diversi domini, in modo da permettere la gestione integrata delle informazioni e delle specifiche definite sotto diversi punti di vista e con obiettivi differenti. Questa caratteristica aumenta di fatto l'integrità dei modelli finali e permette di sfruttare al meglio i vantaggi che ciascun particolare dominio può apportare in fase di specifica.
La diagnostica gioca un ruolo fondamentale nella qualità che il committente andrà a percepire durante tutta la vita dell'impianto, ed in particolare la capacità di isolare e quindi riparare rapidamente possibili guasti porta ad una riduzione dei fermi macchina e quindi ad un incremento della produttività dell'intero impianto. Per questo motivo nuovi sistemi diagnostici sono stati integrati nell'architettura Medeia, con l'obiettivo di includere un approccio diagnostico già in fase di specifica. Un discorso simile è stato fatto sia per la simulazione che per la verifica del singolo AC, andando ad integrare metodologie di simulazione che permettano il testing di ogni parte del modello specificato anche prima della sua reale implementazione.
I modelli realizzati vengono quindi implementati per particolari piattaforme in modo automatico o semi-automatico, riducendo così al minimo la possibilità di errore nella fase di traduzione delle specifiche in codice eseguibile.
Una presentazione dei concetti chiave affrontati e sviluppati nel progetto Medeia è riportata in [5].

La struttura gerarchica Medeia

La struttura gerarchica definita dall'approccio Medeia vuole colmare le differenze che esistono tra la progettazione meccanica ed elettrica di un impianto, che solitamente vede l'impianto come insieme di dispositivi, e quella invece del controllo che si basa principalmente su una visione d'insieme del sistema da controllare. Con questa scomposizione multi livello si possono identificare due tipi di AC che possono essere definiti come Hardware Specific AC, cioè blocchi che modellizzano una vera parte hardware e software (per esempio punti di I/O, robot, macchine, nastri ecc.), e i Compositional AC che rappresentano blocchi formati dalla composizione di altri blocchi e che aggiungono quindi solo una serie di funzionalità di alto livello e di supervisione.

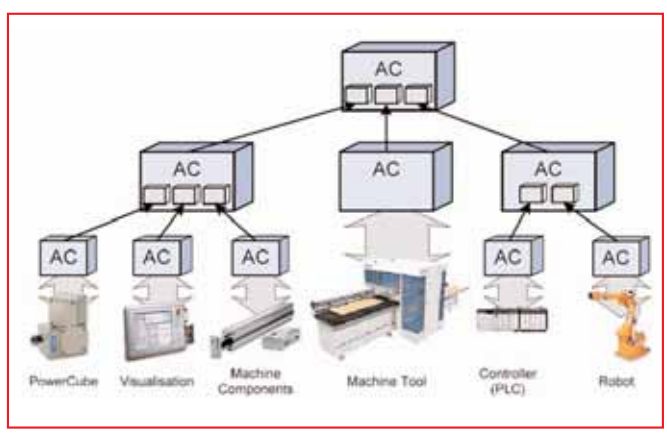

Figura 1 - Struttura gerarchica ad Automation Component

Il primo passo per la progettazione di un nuovo impianto diventa quindi la definizione dell'insieme delle specifiche e della sua architettura di massima organizzata ad AC. Successivamente verranno inclusi gli AC già presenti in libreria in modo da risparmiare tempo per la realizzazione ed il testing di soluzioni già sviluppate in precedenza. Il processo di specifica e creazione dell'impianto segue quindi una serie di raffinamenti operati da attori diversi (tipicamente fornitori e subfornitori) che vede la divisione dell'architettura in una gerarchia multi livello di AC. Ad ogni passo ciascun AC può essere simulato e testato ancora prima di definire e terminare la specifica e l'implementazione dell'intera architettura.
L'impiego del workflow Medeia può ad esempio partire dalla definizione da parte di un system integrator di un'architettura di un nuovo impianto, fase nella quale viene definita l'intera gerarchia di AC e le relative interazioni e connessioni. Questo progetto può quindi essere passato ad un produttore che in base alle specifiche definite in precedenza utilizza una serie di AC di diversi fornitori di dispositivi per realizzare la struttura finale. Seguendo la metodologia proposta da Medeia si può quindi standardizzare l'interazione tra fornitori e sub-fornitori, utilizzando così lo stesso strumento per la definizione ed il passaggio delle specifiche. La descrizione delle metodologie e del workflow Medeia è riportata negli articoli della sessione [7].

L'Automation Component

Come già spiegato in precedenza il cuore della metodologia sviluppata nel progetto Medeia è l'Automation Component, inteso come strumento per modellizzare congiuntamente parti hardware e software. La specifica del modello dell'AC passa attraverso una serie di sotto modelli, ciascuno dei quali viene utilizzato per descrivere una particolare parte dell'AC stesso. Ogni AC ha infatti una propria Interface Specification, che definisce le metodologie e le regole di aggregazione dell'AC e di scambio di informazioni con gli altri AC, ed un Internal Behavior, che modellizza il vero e proprio comportamento dell'AC rispetto al mondo circostante. L'architettura gerarchica permette, come spiegato nel paragrafo precedente, l'aggregazione multi livello degli AC. Ciascun AC ha quindi una Hierarchical Aggregation che definisce l'aggregazione dei suoi sotto componenti, ed un gruppo di Communication Local I/O che modellizza le connessioni dirette con l'hardware dell'AC. Vista la suddivisione in Hardware Specific AC e Compositional AC spiegata nella sezione precedente queste ultime due parti sono praticamente in mutua esclusione. Pertanto, se un AC è di tipo composto la parte che descrive l'interfaccia ai segnali hardware sarà mancante, viceversa se l'AC modellizza una parte hardware mancherà la sua parte che aggrega sotto componenti, non avendo quest'ultimo sotto componenti (figura 2).
La specifica di ciascuna parte dell'AC avviene mediante una serie di viste e metodi definiti Domain Specific Models. Esiste infatti una vasta moltitudine di metodologie e linguaggi utilizzabili per descrivere e specificare modelli di dispositivi automatici, sia per quanto riguarda le parti fisiche che per quelle software. Si avrà quindi un insieme di possibili viste

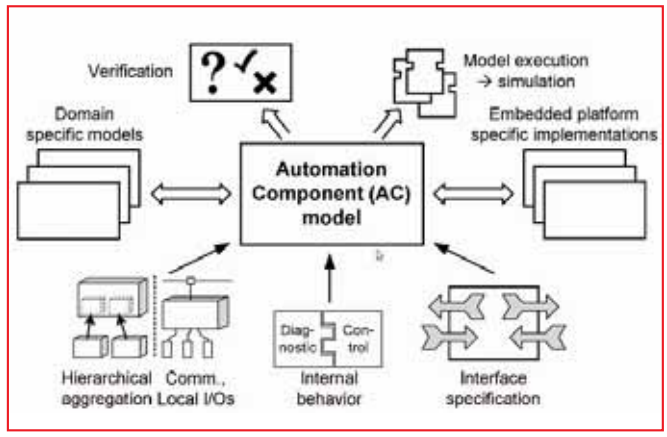

Figura 2 - Modello dell'Automatomation Component

che dipenderanno dal singolo dominio considerato, ciascuna delle quali potrà essere utilizzata per descrivere l'AC. Particolari trasformazioni, che ovviamente saranno legate alla vista scelta (es. UML diagram, Gantt Chart, CAD, Sequential Flow Chart ecc.), permettono di trasformare le informazioni editate tramite la vista in specifiche parti dell'AC e viceversa. La trasformazione inversa permette così l'editing di modelli già sviluppati in precedenza. Per analoghi motivi ciascun AC ha un set di possibili Embedded Platfrom Specific Implementations, ovvero ciascun modello potrà essere implementato su più possibili piattaforme tramite una serie di trasformazioni che permettono la generazione automatica di codice e la traduzione tra differenti modelli. In questo modo Medeia riesce a suddividere la parte di specifica da quella di implementazione inserendo nel mezzo l'AC che può quindi essere considerato come un'interfaccia per lo scambio di informazioni tra le due fasi.

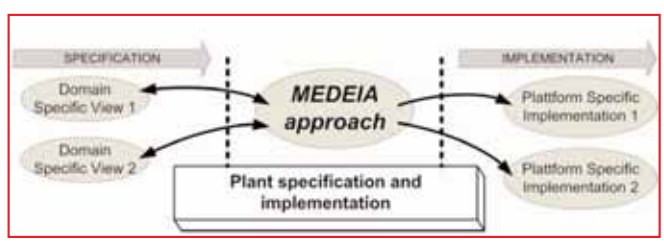

Figura 3 - Razionalizzazione dell'approccio Medeia

Automation Component Implementation Model

La metodologia sviluppata nel progetto Medeia vede tre componenti fondamentali che compongono il framework finale. L'insieme di Domain Specific View (DSV) che permette l'editing dei modelli di ciascun Automation Component Implementation Model (ACIM) tramite una serie di strumenti e linguaggi che vengono scelti per ciascun particolare dominio. Questa parte è l'unica che si interfaccia direttamente con l'utilizzatore finale, ovvero è l'unica che viene messa a disposizione dei progettisti meccanici, elettrici, controllisti ecc. per editare i modelli ed interagire con l'intero framework. Un'analisi ed una catalogazione delle viste considerate in Medeia sono riportate in [4]. La parte di implementazione finale dei

modelli avviene tramite una serie di possibili Platform Specific Implementation (PSI) che, utilizzando le informazioni presenti all'interno dell'ACIM, generano in modo automatico il codice o i modelli per particolari piattaforme (PLC, CNC, PC industriali, FPGA, ecc.). L'ACIM diventa quindi il metamodello utilizzato per modellizzare ed organizzare le informazioni di specifica editate tramite le DSV e che vengono quindi utilizzate per la generazione di diverse implementazioni tramite le PSI. L'ACIM è composto a sua volta da tre sottomodelli ciascuno dei quali raccoglie un insieme di informazioni che caratterizzano e descrivono una particolare parte del sistema considerato. In particolare, l'Automation Component Model (ACM) rappresenta la parte dell'ACIM indipendente dalla piattaforma. Esso modellizza il comportamento dell'AC, la sua interazione con gli altri AC e la parte di diagnostica. In questo particolare sottomodello viene anche riportato il modello dell'impianto utile per simulare e testare le regole definite nella parte comportamentale. Il secondo sotto modello dell'ACIM, Mapping Model (MM), descrive una serie di modelli e di regole per il mapping dei segnali e dei modelli di alto livello descritti nell'ACM sulla piattaforma considerata. La piattaforma viene quindi descritta nell'Execution System Model (ESM) nel quale vengono raccolte una serie di informazioni utili a descrivere l'hardware, il middleware, i driver ecc. che appartengono all'AC.

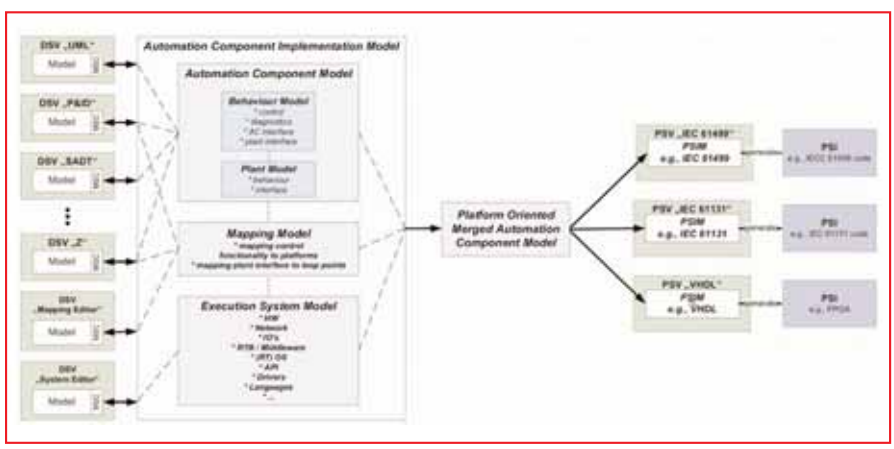

Figura 4 - Panoramica del Frameowork Medeia

L'approccio presentato si basa su regole di trasformazione tra modelli che sono cruciali per l'applicabilità dell'intera metodologia. In particolare una serie di trasformazioni bidirezionali vanno definite per la creazione e l'editing dell'ACIM e per la generazione dell'implementazione per ciascuna piattaforma considerata. Per facilitare la generazione di codice e la traduzione dell'ACIM in modelli eseguibili una prima traduzione e riorganizzazione delle informazioni è stata aggiunta mediante il Platform Oriented Merged Automation Component Model (Pomac). Questo componente è utilizzato appunto per facilitare la traduzione finale, riorganizzando le informazioni e i modelli dell'ACIM e aggiungendo informazioni mancanti che riguardano la particolare piattaforma.
Informazioni dettagliate sui modelli e sull'approccio Medeia sono riportate in [6].

Modelli di impianto, controllo e diagnosi

Come già spiegato nei paragrafi precedenti l'approccio Medeia prevede una modellizzazione integrata dell'intero sistema. Questo significa che l'AC, principale attore del framework proposto, contiene e descrive i modelli congiunti di impianto, controllo e diagnosi. La centralizzazione dell'approccio modellistico permette una gestione concentrata delle informazioni e la possibilità di verificare l'integrità dei modelli definiti.
L'utilizzo e lo sviluppo di librerie di AC può essere quindi concepito come un sistema di memorizzazione non solo di algoritmi e strutture dati, ma anche di descrizioni e soluzioni elettromeccaniche e funzionalità più o meno avanzate di diagnostica, fornendo così un valido strumento per la gestione dell'intero know-how di produzione e sviluppo. L'integrazione del modello dell'impianto permette inoltre di implementare facilmente modelli di simulazione per il testing delle logiche di controllo specificate nella parte comportamentale dell'AC. L'integrazione di modelli diagnostici cerca di dare una risposta concreta alla forte richiesta di sistemi affidabili e facilmente riparabili. Infatti i lunghi fermi macchina in un mercato caratterizzato da un'elevata concorrenza generano spesso mancati profitti che possono provocare anche gravi difficoltà aziendali. Per questo motivo in Medeia si è cercato di inserire già in fase di progetto un approccio formale e sistematico per l'identificazione e l'isolamento dei guasti con il fine di ridurre i tempi di riparazione e quindi aumentare la produttività degli impianti.

Prototipi sviluppati e strumenti adottati

Un primo prototipo del framework Medeia è oggi disponibile e permette, tramite l'implementazione Mule dell'architettura ESB (Engineering Service Bus), di accedere ad un repository centralizzato e di editare modelli di AC già presenti o di crearne di nuovi, tramite un semplice editor fornito con il pacchetto oppure mediante viste esterne integrate dall'utente.
L'intera applicazione è stata sviluppata utilizzando una serie di plugin per l'IDE Eclipse come EMF (Eclipse Modeling Framework) che supporta lo sviluppo di software model-based. Lo stato attuale dell'implementazione permette inoltre di generare codice per diverse piattaforme, in particolare si è presa in considerazione la piattaforma software PLC Orchestra sviluppata da Sintesi SCpA. L'intera piattaforma e la relativa documentazione possono essere scaricate dal sito web [3]. Orchestra è un motore software hard real-time per l'esecuzione di logiche di controllo discrete e controllo di traiettorie tramite CNC disponibile per sistemi operativi Linux/RTAI (RTAI - RealTime Application Interface). Un'altra piattaforma che è già stata integrata in Medeia è quella definita dallo standard IEC61499, ed in particolare l'implementazione 4DIAC-RTE sviluppata da un gruppo internazionale di enti di ricerca, aziende ed università coordinato da Profactor GmbH.

Maggiori informazioni sul framework 4DIAC sono riportate in [2].
Un insieme di strumenti per la diagnostica e l'isolamento dei guasti sono stati sviluppati ed implementati utilizzando anche le potenzialità di sistemi realtime come RTJS (RealTime Java Specification), partendo da modelli basati su automi a stati finiti e sviluppati da precedenti progetti e lavori di ricerca orientati principalmente ad ottenere risultati teorici, esempi e descrizioni dettagliate possono essere trovate in [1].
I prototipi sviluppati sono stati testati con un insieme di casi di studio forniti dai partner industriali del progetto. In particolare sono stati sviluppati esempi riguardanti la modellistica e la diagnostica di sistemi manifatturieri e robotici basati sull'impianto SPI di MCM SpA e l'architettura Modular Robotic di Schunk GmbH.

Figura 5 - Schunk GmbH Modular Robotic

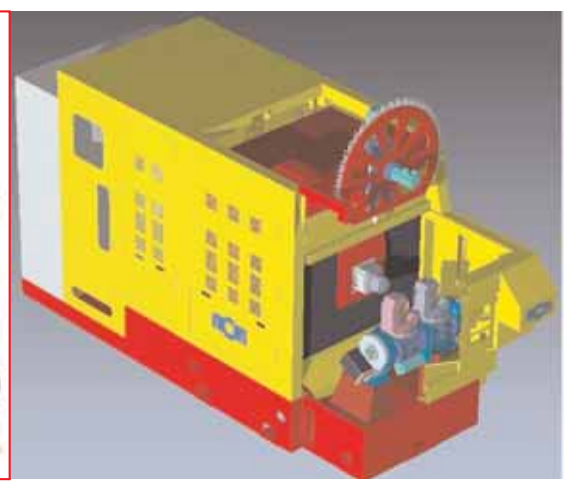

Figura 6 - MCM SpA Macchina SPI

La chiusura del progetto prevede quindi la creazione di una base dati centralizzata accessibile tramite ESB, in grado di gestire e memorizzare l'intera libreria di AC di un'azienda. L'architettura fornirà un insieme di interfacce software utilizzabile per integrare tool e applicativi esterni da utilizzare come viste per l'editing e la creazione di modelli di AC. Un sistema integrato per la gestione della diagnostica verrà sviluppato ed inserito nel framework finale, con lo scopo di dare un supporto completo e quanto più possibile automatico allo sviluppo di sistemi che facilitino e permettano un rapido isolamento di guasti e la gestione di possibili malfunzionamenti. Le due piattaforme descritte in precedenza rimarranno il punto di partenza per la creazione di nuovi traduttori e generatori di codice in grado di integrare soluzioni embedded e di controllo in base alle esigenze dell'utilizzatore finale, lasciando quindi spazio ad una personalizzazione dell'intero framework.

Conclusioni

Medeia vuole definire un innovativo metodo per la progettazione e la realizzazione di complessi sistemi industriali automatizzati. In particolare tramite la metodologia proposta, fortemente basata su modelli in ogni sua fase, si vogliono ridurre tempi e costi attribuibili alla progettazione ed alla successiva realizzazione degli impianti.
Il framework specificato nel progetto può essere applicato in differenti realtà industriali (produzione di energia, manufacturing, sistemi robotizzati, impianti di packaging ecc.). Grazie infatti alla definizione e all'integrazione di viste personalizzate per editare i modelli è possibile adattare l'approccio proposto a diverse esigenze, rendendone semplice la comprensione e l'impiego. La generazione automatica di codice e la traduzione automatica in modelli eseguibili rende poi semplice l'implementazione finale dei modelli sviluppati per le particolari piattaforme scelte. La modellizzazione integrata di controllo, impianto e diagnostica permette un approccio sistematico per la simulazione e quindi il testing delle logiche sviluppate. Inoltre l'approccio diagnostico inserito nativamente nel framework facilita l'isolamento e la riparazione di guasti e malfunzionamenti aumentando la produttività del sistema controllato.
Medeia affronta così la sfida di ridurre i costi di produzione degli impianti pur mantenendo un buon grado di flessibilità e un'elevata qualità attraverso un approccio basato sui modelli.

Bibliografia

[1] M. Sampath, R. Sengupta, S. Lafortune, K. Sinaamohideen, and D. Teneketzis. "Failure Diagnosis Using Discrete Event Models". *IEEE Transactions on Automatic Control*, 4(2):105-124, Marzo 1996.
[2] 4DIAC Framework for Distributed Industrial Automation and Control, www.fordiac.com .
[3] Sintesi SCpA, Orchestra Control Engine. www.orchestracontrol.com.
[4] L. Ferrarini, A. Dedè, P. Salaün, T. Dang and G. Fogliazza, "Domain Specific Views in Model-driven Embedded Systems Design in Industrial Automation", *Proceedings of the 7th IEEE International Conference on Industrial Informatics (INDIN'09)*, Cardiff , Wales, United Kingdom, June 23-26, 2009.
[5] T. Strasser, C. Sünder, A. Valentini, "Model-Driven Embedded Systems Design Environment for the Industrial Automation Sector", *Proceedings of the 6th IEEE International Conference on Industrial Informatics*, Daejeon, South Korea, July 13-16, 2008.
[6] T. Strasser, M. Rooker, I. Hegny, M.Wenger, A. Zoitl, L. Ferrarini, A. Dedè and M. Colla, "A Research Roadmap for Model-Driven Design of Embedded Systems for Automation Components", *Proceedings of the 7th IEEE International Conference on Industrial Informatics (INDIN'09)*, Cardiff , Wales, United Kingdom, June 23-26, 2009.
[7] Consorzio Medeia, "Applying advanced software design concepts to industrial automation systems", *10th IFAC Workshop on Intelligent Manufacturing Systems*, Lisbon, July 2010.