Università degli Studi di Parma

Facoltà di Scienze
Matematiche, Fisiche e Naturali

Corso di Laurea in Informatica

Tesi di Laurea

Progettazione di applicazioni per la supervisione di sistemi domotici

Relatore Prof. Giulio Destri

Correlatore Ing. Cesare Chiodelli

Candidata Irene Bacchi

Anno Accademico 2004/2005

a mia sorella Addy

Ringraziamenti

Desidero innanzitutto ringraziare l'Ing. Cesare Chiodelli e il Prof. Giulio Destri per aver permesso lo svolgimento del mio stage formativo presso la ditta CS Soluzioni, consentendomi di apprendere le basi per lo sviluppo di software a livello professionale e impostando la stesura della mia tesi. Un grazie anche ad Alberto Picca e Luca Lodi Rizzini per il lavoro svolto insieme.

Ringrazio i miei compagni di università Andrea, Danilo, Fabio, Giordano, Dolma, Matteo, Nadia, Paolo, Anna, Delia, Maximiliano e Samuele, con cui ho passato questi quasi quattro anni di studio (e non solo) e i miei amici più "vecchi" Silvia, Stefo, Mundo, Ibou, Ori, Tello, Ferro e Anna.

Ringrazio i miei genitori Carolina e Angelo per essermi stati vicini e per avermi supportato materialmente e moralmente, e le mie sorelle Chiara, per la correzione degli errori grammaticali, e Adele, alla quale dedico questo lavoro.

Un ringraziamento speciale a Matteo, che mi ha sopportato fino ad ora e per tutto il resto che sa già.

Introduzione

L'ambito di lavoro documentato da questa tesi è la domotica, ovvero l'automazione di un edificio dotato di dispositivi ed impianti (agenti domotici) integrati mediante una rete di comunicazione e costituenti un sistema aperto, flessibile e capace di interagire con l'utente in modo diretto ed efficace.

In un "edificio intelligente" molte delle funzioni sono controllate da un sistema basato sulla scienza dell'informazione, sull'automazione, sull'elettronica e su di una strumentazione dotata di interfacce facilmente comprensibili.

Per la progettazione di questo tipo di sistema sono fondamentali una perfetta integrazione e interoperabilità tra i processi di sviluppo che implementano le specifiche relative alla gestione dei seguenti sottosistemi:

- impianti tecnologici, come climatizzazione ambientale, generazione e distribuzione dell'energia;
- impianti di sicurezza e sorveglianza, ad esempio rilevazione fumo e gas, servizi antincendio, controllo barriere di accesso, controllo anti-intrusione e antifurto, sorveglianza mediante monitor a circuito chiuso;
- impianti di comunicazione, come collegamenti telefonici ed interfonici, telefax, videoconferenza, traduzione simultanea, diffusione sonora, ricerca persone;
- impianti informatici, quali servizi di posta elettronica, condivisione di risorse come unità centrali di memoria e unità di stampa, accesso a database.

Il passo successivo necessario per la realizzazione di un edificio intelligente è costituito da un nuovo livello di automazione che integra i precedenti sottosistemi, rendendo possibile una vera e propria gestione centralizzata. Questa integrazione deve avere luogo attraverso la creazione di strumenti informatici ben progettati, utilizzando tecniche proprie dell'Informatica.

La tesi si propone di applicare le metodologie di analisi, progettazione e realizzazione software apprese nel Corso di Laurea al settore della domotica. In particolare verrà illustrato lo sviluppo del progetto *IntelliDomus*, ideato dalla ditta *CS Soluzioni* di Casalmaggiore (CR).

Indice

Capitolo 1

La Domotica e l'Automazione Civile

In questo capitolo viene introdotto il contesto della domotica e dell'automazione civile, con le terminologie e le problematiche relative.

1.1 Home e Building Automation

La domotica può essere suddivisa in diverse specializzazioni: la *Home Automation*, rivolta all'automazione della casa intesa come singola unità abitativa (per esempio una villa o anche un singolo appartamento), e la *Building Automation*, rivolta all'automazione delle funzioni di un edificio di dimensioni maggiori ad uso lavorativo (industriale, amministrativo, commerciale, ecc.).

A differenza del mercato della Home Automation, che da poco si trova in una fase di espansione, il mercato della Building Automation è già consolidato da diversi anni ed ha già sviluppato un'offerta articolata e ben recepita dagli utenti, tramite progettisti e installatori specializzati in questo settore.

Talvolta il termine domotica viene circoscritto a sinonimo di Home Automation. Da parte di molti operatori nell'area della Building Automation si è formata l'opinione che gli stessi prodotti, gli stessi approcci commerciali e tecnici, seppur in scala ridotta, possano essere facilmente applicati e adattati al mercato della Home Automation.

	Building Automation	Home Automation
Decisore	azienda	abitante
Utente	lavoratore	abitante
Gestore sistema	building manager	abitante
Utilizzo	complesso	semplice
Dimensione	edificio o gruppi di edifici	abitazione singola
Gestione spazi	dinamica	statica
Motivazioni	sicurezza	comfort
	risparmio energetico	sicurezza
	automazione utenze elettriche	status symbol
	controllo accessi	intrattenimento

Tabella 1.1: Differenze tra Home e Building Automation.

Questo passaggio in realtà non è così banale poichè le due aree, per quanto presentino caratteristiche simili, hanno ambiti di utilizzo, di amministrazione e di tipologia di utenti molto diversi. In tabella 1.1 sono riassunte le principali differenze.

1.2 Normative nazionali ed europee

Gli ultimi anni hanno portato profonde innovazioni nel modo di pensare e progettare gli impianti tecnologici, anche sotto la spinta di provvedimenti legislativi introdotti nel nostro paese e nella comunità europea.

La legge 46/90, che stabilisce delle norme per la sicurezza degli impianti, e la 10/91, che regola le materie di risparmio energetico e sviluppo di nuove fonti di energia, hanno regolamentato la progettazione e la realizzazione degli impianti elettrici, termici, di rilevamento di incendi e di telecomunicazioni. Il Decreto Legislativo 626/94 ha in seguito introdotto le direttive europee sulla sicurezza, completate successivamente con il regolamento di prevenzione incendi.

Questo ha fatto in modo che il responsabile di una attività si sia trovato ad avere l'obbligo di gestire e tenere in efficienza un elevato numero di

impianti tecnologici e di sicurezza che si presentano sempre più complessi e sofisticati.

Per poter gestire in maniera efficiente queste problematiche viene in aiuto la Building Automation, il cui scopo è quello di integrare tutti gli impianti sotto un unico strumento di controllo, tramite soluzioni elettroniche ed informatiche avanzate, in modo da arrivare all'integrazione in una unica rete dei diversi sottosistemi dedicati alla sicurezza, al risparmio energetico ed alle altre funzioni fondamentali.

1.3 Progettare un sistema di Building Automation

Il processo di progettazione edilizia ha come fine ultimo la costruzione di un immobile che risponda alle esigenze dell'uomo. Questo compito, col passare del tempo, è stato trasferito dall'edificio vero e proprio agli impianti di cui esso è corredato, ma ha comportato un prezzo in termini di costi energetici e sociali. Proprio per questo la progettazione degli edifici, oltre alle relazioni che intercorrono tra l'ambiente esterno, l'uomo, l'involucro e l'impianto, deve tener conto anche di questo aspetto.

Si può osservare, infatti, come lo sviluppo di dispositivi impiantistici ha per certi versi ulteriormente deresponsabilizzato la progettazione edilizia, delegando esclusivamente all'impianto di climatizzazione o di illuminazione il compito di attivare le condizioni ambientali interne per compensare le carenze o gli errori del progetto edilizio.

Le problematiche energetiche e una maggiore attenzione al tema dell'inquinamento ambientale evidenziano oggi l'opportunità di operare direttamente sull'"organismo edilizio", mediante tecnologie sostenibili da un punto di vista energetico ed ambientale. Si deve poter ridurre al massimo la dipendenza energetica nell'attivazione di condizioni ambientali coerenti con le attuali attese di qualità. Infatti queste dipendono da una corretta interazione tra il sistema costruttivo e le componenti distributive e di articolazione volumetrica, che possono contribuire in modo notevole a una mediazione intelligente tra clima interno e clima esterno.

Proprio per le precedenti osservazioni il committente spesso ritiene necessario assegnare alle componenti di controllo ambientale la loro giusta

rilevanza, fissando gli obiettivi del prodotto finale e consegnando la progettazione della sede a più studi tecnici che, individuando le possibili strategie tecniche/morfologiche praticabili per la loro attivazione, realizzano un progetto capace di minimizzare i consumi energetici a costi interessanti, soprattutto se confrontati con edifici costruiti secondo i criteri tradizionali. Si parla spesso a tal proposito di "edificio intelligente".

1.4 Integrazione delle parti

In un edificio intelligente molte delle funzioni sono controllate da un sistema basato sulla scienza dell'informazione, sull'automazione, sull'elettronica e su di una strumentazione dotata di interfacce facilmente comprensibili e gestibili da qualsiasi tipo di utente. L'edificio stesso fa parte di un sistema che è molto più di una somma delle singole automazioni.

Ci sono molti termini che identificano un sistema integrato e molti altri che indicano l'approccio all'integrazione, come ad esempio il *Building Automation System*. Tali sistemi sono già maturi per i mercati del settore terziario (banche, uffici, centri direzionali e tecnologici), dove la disponibilità di un più elevato budget e di esperienze perfezionate negli anni consentono di valutare meglio i vantaggi dell'integrazione. Per il mercato della casa, invece, si ricercano soluzioni più facilmente adattabili ad un'utenza disposta a spendere quantità di denaro limitate e molto meno consapevole delle opportunità offerte.

I sistemi presenti attualmente sul mercato derivano soprattutto da due filoni:

- dall'automazione industriale (sistemi molto affidabili, completi, flessibili ma costosi);
- dai produttori di antifurti (sistemi semplici ed economici ma quasi sempre costruiti senza seguire gli standard e quindi potenzialmente inaffidabili quanto a supporto sul lungo termine).

Non si è ancora riusciti a riempire il segmento intermedio, giustificando quindi la lentezza con cui il mercato risponde a questo tipo di soluzioni.

Per accattivare nuovi potenziali clienti i sistemi devono presentare caratteristiche di affidabilità e modularità. Su una dotazione base, comprendente un sistema di "intelligenza locale" (centralizzata o distribuita) e su una rete di comunicazione, deve essere possibile inserire funzioni diverse, composte da programmi, sensori e attuatori, modulando in maniera agevole i costi. Inoltre deve essere possibile prevedere l'integrazione di ulteriori funzionalità in tempi successivi. Da ciò l'enorme importanza della standardizzazione a garanzia degli investimenti futuri dell'utenza.

In altri termini la maggiore valenza del sistema domotico è quella di monitorare e controllare in tempo reale tutte le funzioni del sistema (edificio, impianti, utenza, clima), considerando tutte le interazioni possibili e ottimizzando le prestazioni complessive secondo criteri prefissati o perfezionabili nel tempo. Gestendo in modo integrato un insieme di funzionalità complesse si può ottenere un significativo miglioramento globale del comfort per gli utenti e dell'efficienza energetica.

Le riduzioni dei consumi per l'energia elettrica, il riscaldamento, ecc. sono difficilmente quantificabili, anche se alcune aziende fornitici di soluzioni promettono risparmi nell'ordine del 25%. Con l'intelligenza distribuita nell'edificio si possono gestire funzioni complesse quali: il controllo dell'illuminazione (complessivo o locale), della qualità dell'aria, del funzionamento dell'impiantistica, degli allarmi tecnici, delle intrusioni, dell'autenticazione degli accessi e delle risorse.

La tecnologia attuale permette di scegliere impianti con o senza fili o con entrambe le modalità. Essi sfruttano sistemi di comunicazione tradizionali (radiofrequenza o linea telefonica PSTN/ISDN) oppure le moderne tecnologie GSM, GPRS, ADSL. Ne deriva una flessibilità generale che si traduce in un'ampia scelta per l'utilizzatore di funzioni particolarmente utili e finora gestibili solo manualmente e soprattutto complesse da installare. Inoltre, tali sistemi possono permettere l'accesso al mercato dei teleservizi in rete permettendo lo scambio di messaggi con l'esterno.

1.5 L'edificio "intelligente"

Gli edifici attuali, destinati ad attività produttive nel settore dei servizi, si stanno evolvendo verso configurazioni chiamate *Edificio Intelligente* oppure *Hi-Tech Building*.

Il vantaggio della concentrazione in un unico sistema edilizio di una quantità crescente di servizi avanzati consiste nell'eliminare i passaggi intermedi, e quindi semplificare il processo di lavorazione a vantaggio della qualità del prodotto finale, della velocità di realizzazione e della produttività del capitale investito in uomini e tecnologie.

La spinta principale nella direzione dell'edificio intelligente è data dal fatto che questa soluzione è il punto d'incontro delle diverse esigenze (architettoniche, funzionali, economiche) che concorrono a definire le prestazioni richieste al sistema edificio e ai suoi componenti. Questa soluzione nasce dall'aver capito che in un'attività di produzione le strutture edilizie, gli impianti generali, gli impianti produttivi vanno considerati come un *sistema unico di risorse concorrenti* sia all'esercizio che allo sviluppo dell'intero sistema produttivo. In altri termini nella progettazione dell'edificio è necessario pianificare lo sviluppo dei mezzi tecnologici di produzione e anche integrare razionalmente tra loro i principali sottosistemi necessari all'attività produttiva.

All'interno di questa logica si possono individuare le principali ragioni di natura tecnica, economica e organizzativa che favoriscono l'espandersi di queste applicazioni:

- la grande diffusione delle tecnologie elettroniche applicate all'informatica, alle comunicazioni, al controllo degli impianti, alla sicurezza e alla sorveglianza che ha favorito la riduzione del costo unitario delle funzioni svolte;

- la necessità di razionalizzare e coordinare l'installazione, la manutenzione e l'espansione degli impianti;

- l'esigenza di evitare la proliferazione di reti di trasmissione di informazioni, di favorire l'integrazione di apparati diversi che svolgono

funzioni simili e la compatibilità dei sottosistemi che sono previsti per la comunicazione;

- la valutazione della produttività dell'edificio che aumenta quando in esso è possibile riunire ed organizzare razionalmente molte risorse tecnologiche ed umane.

Tutte queste considerazioni hanno come conseguenza la scelta di strutture edilizie ed impiantistiche altamente razionali, affidabili e capaci di garantire efficienza, comfort, e sicurezza anche nei confronti di gravi eventi perturbativi come incendi, scosse sismiche, black-out elettrici, furti.

Nel tempo, lo sviluppo di nuove tecnologie elettroniche e informatiche (sensori, reti, terminali, ecc.) ha permesso l'introduzione e l'integrazione di sistemi per controllare e gestire in modo automatico il funzionamento degli impianti produttivi. Tali processi hanno portato le loro innovazioni a modificare il controllo e la gestione dei seguenti sottosistemi:

Impianti tecnologici Climatizzazione ambientale, generazione e distribuzione dell'energia, controllo spaziale e temporale dell'illuminazione interna ed esterna, distribuzione idrica, trasporti verticali e orizzontali;

Impianti di sicurezza e sorveglianza Rilevazione di fumo e gas, servizi antincendio, controllo barriere di accesso, controllo antintrusione e antifurto, sorveglianza mediante monitor a circuito chiuso;

Impianti di comunicazione Collegamenti telefonici ed interfonici, telefax, videoconferenza, traduzione simultanea, riproduzione e diffusione sonora, ricerca persone;

Impianti informatici Trattamento di testi e immagini, posta elettronica, condivisione di risorse come le unità centrali di memoria e le unità di stampa, accesso a database.

Questi quattro sistemi sono utili ed efficaci, ma per realizzare l'edificio intelligente è necessario un successivo passo costituito da un nuovo livello

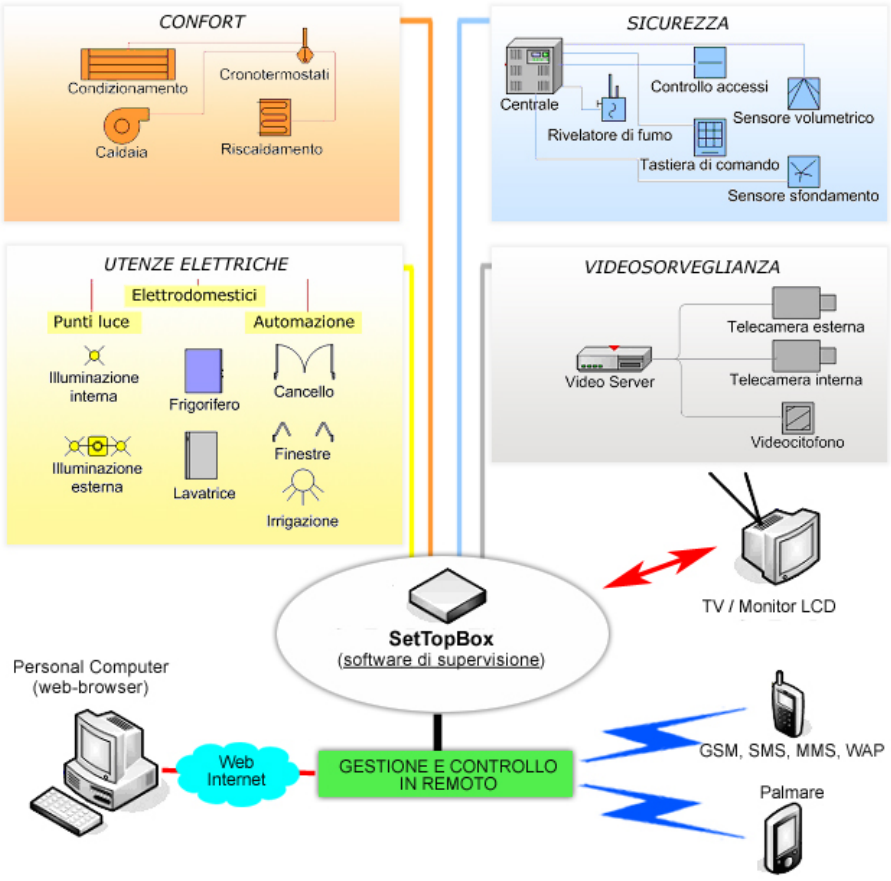


Figura 1.1: Struttura generale di un sistema di Home Automation.

di automazione che integra i precedenti sottosistemi, rendendo possibile una vera e propria *gestione centralizzata*.

Nasce così un sistema integrato di automazione e controllo di tutti gli impianti tecnologici, che consente la gestione complessiva del sistema di produzione e l'ottimizzazione dell'uso delle risorse disponibili, sia in termini di efficienza che di produttività.

Un aspetto importante di una struttura di tipo gerarchico è quella di garantire la crescita modulare, ossia la possibilità di espansione al crescere delle esigenze e l'adattamento a specifiche variabili nel tempo.

I campi tipici di applicazione dell'edificio intelligente sono: edifici per uffici, centri direzionali, centri commerciali, centri per congressi, ospedali,

aeroporti, stazioni ferroviarie, università, musei, alberghi, parcheggi, ecc.. L'obiettivo da raggiungere è la gestione dei flussi di materiali, energia, comunicazioni e informazioni, in modo da garantire la massima sicurezza e comfort degli occupanti, la funzionalità delle comunicazioni, l'accesso ad informazioni e la riduzione delle risorse energetiche.

L'architettura più adatta a gestire simili livelli di automazione è il modello ad intelligenza distribuita con duplicazione dei dispositivi critici. Essa necessita di un centro di controllo specializzato per ognuna delle quattro aree funzionali e di un centro di supervisione per organizzare la gestione globale. Il processo deve essere il più possibile automatico ma è necessario che un operatore abbia la possibilità di intervenire direttamente nel caso di guasti, per ordinaria manutenzione o per l'attivazione di procedure non usuali.

In molte tipologie di edifici come alberghi, villaggi vacanze, parcheggi, banche, uffici, palestre, ecc. si ha l'esigenza di controllare l'accesso ad ambienti, reparti e zone di sicurezza. Tale necessità ha imposto lo sviluppo di dispositivi di autenticazione e di controllo come badge o codice PIN e strumenti per la segnalazione di eventuali infrazioni al personale addetto alla sorveglianza.

Nell'ambito di un edificio intelligente è opportuno pensare ad un'integrazione complessiva dei sistemi che effettuano il controllo degli accessi alla rete con funzioni simili a quelle relative alla sicurezza e al monitoraggio degli ambienti (rilevazione e spegnimento incendi, video-sorveglianza), in modo da semplificare l'installazione e ridurre il cablaggio.

Capitolo 2

L'informatica per l'Automazione

Verrano trattati gli aspetti relativi all'integrazione tra tecnologie informatiche e sistemi di automazione, illustrando i requisiti funzionali e i vincoli che dovrebbe rispettare un progetto software.

Inoltre verrà descritta la struttura del PLC, un componente fondamentale per la realizzazione di un sistema domotico.

2.1 L'informatica "dietro" l'automazione

Si consideri un sistema di automazione elementare composto da un interruttore e da una lampadina: quest'ultima ha il semplice scopo di indicare ad un operatore lo stato dell'interruttore, il quale a sua volta può verificare se un dispositivo è attivo, ad esempio se un motore è in funzione.

Pensando che l'operatore non si trovi in prossimità dell'interruttore, la lampadina, con la sua informazione associata, deve quindi necessariamente essere lontana dall'interruttore. È chiaro che non si può pensare di realizzare un circuito elettrico così esteso e si deve, quindi, prendere in considerazione l'uso di dispositivi di comunicazione per trasportare l'informazione relativa allo stato dell'interruttore.

Se è necessario dover supervisionare lo stato di alcune migliaia di interruttori, non si può certamente prevedere l'adozione di un singolo dispositivo di comunicazione per ognuno di essi ma si dovrebbe trovare il modo di usarne uno solo per tutti (ad esempio inviando le informazioni di stato

in maniera seriale con annessa l'identificazione dell'interruttore a cui esse si riferiscono).

Per quanto riguarda il ruolo dell'operatore, egli dovrebbe osservare e valutare lo stato di migliaia di circuiti. Per facilitare questo compito si potrebbe pensare di utilizzare un calcolatore che memorizzi le informazioni che arrivano e gliele presenti sotto opportune forme (quadri sinottici).

L'operatore inoltre potrebbe decidere di voler modificare qualcosa, fornendo un comando da eseguire in remoto; il sistema di comunicazione adottato dovrebbe quindi permettere comunicazioni bidirezionali.

Il sistema descritto è ciò che si definisce come un sistema di supervisione e acquisizione dati, meglio noto con la sigla SCADA (*Supervisory Control And Data Acquisition*). Il cuore di un sistema di questo tipo è rappresentato dalla sua realizzazione software, di solito una struttura modulare, come si può vedere in figura 2.1): il nucleo di tale struttura è rappresentato dalla *base dati* o *architettura di processo* a cui tutti gli altri componenti fanno riferimento (vedi figura 2.2).

Altri moduli che compongono il sistema SCADA (non è detto che siano tutti necessariamente presenti) sono quello di accesso alla base di dati, di interfaccia operatore, di archiviazione storica dei dati, di gestione allarmi ed eventi, di sistema esperto, di generazione rapporti, di gestione ricette, di controllo statistico di processo, di supporto alla manutenzione, di comunicazione (driver).

Dato che anche un sistema di piccole/medie dimensioni deve tipicamente gestire molti punti di interfaccia con il processo, vi è quindi la necessità di avere un metodo sistematico per processare le informazioni.

I sistemi SCADA possono avere differenti modalità per la creazione dall'architettura di processo in base alle varie situazioni di utilizzo. L'architettura di processo può essere vista come l'insieme di tutte le azioni svolte allo scambio dati tra i dispositivi e l'utente finale ed è sempre affiancata da una struttura di memorizzazione dei dati, ad esempio un database.

Durante la fase di progettazione di uno SCADA occorre analizzare quali sono tutti i possibili driver e canali di comunicazione che devono essere adottati per il corretto funzionamento del sistema. A tal scopo, risulta necessaria la scelta di una architettura di processo che si avvicini il più

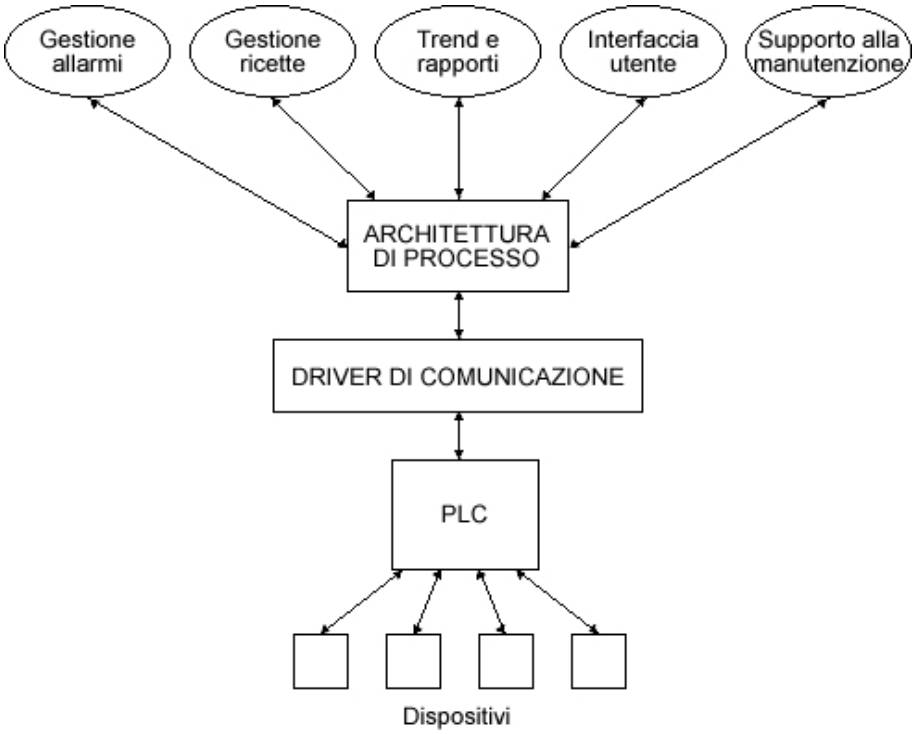


Figura 2.1: Schema funzionale SCADA.

possibile al linguaggio di comunicazione dei diversi driver e che allo stesso tempo fornisca un supporto di archiviazione dati adeguato al flusso delle informazioni.

Il compito principale del software di supervisione è quello di aggiornare costantemente la base di dati, allineando i dati contenuti al suo interno con quelli che vengono forniti dai dispositivi di controllo.

Il driver di comunicazione si occupa essenzialmente della gestione delle comunicazioni con i vari dispositivi e del controllo del trasporto delle informazioni, gestito attraverso tecniche classiche di comunicazione via linea seriale o rete. Il modulo si deve anche occupare dell'interpretazione dei messaggi e del legame bidirezionale tra base dati del processo e dispositivi.

In un sistema SCADA deve essere sempre presente almeno un modulo driver che permetta l'accesso in maniera contemporanea ai diversi dispositivi che sono connessi.

In generale è possibile affermare che la qualità di uno SCADA si misura

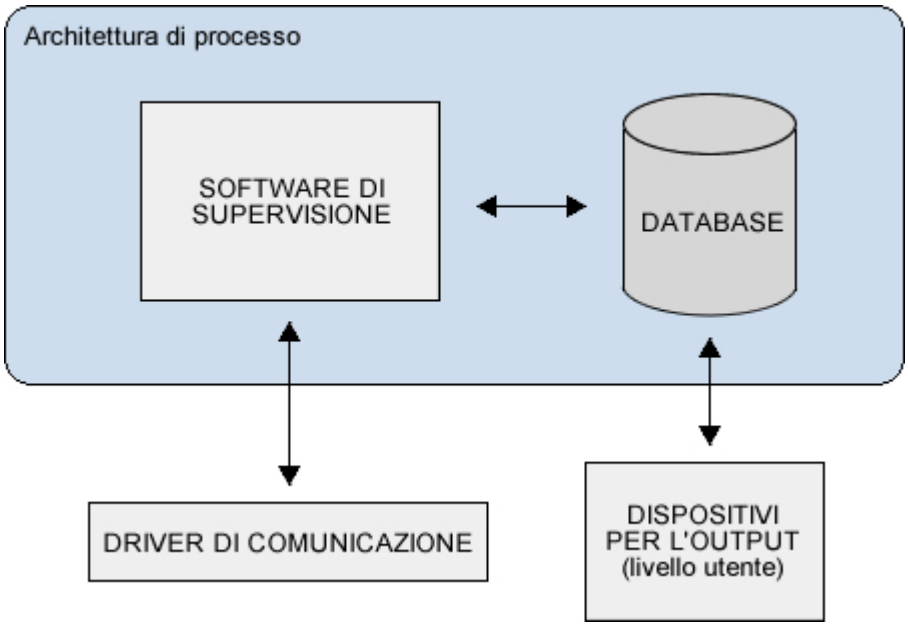


Figura 2.2: Schema dell'architettura di processo.

anche dal numero di driver disponibili. I driver possono essere anche costruiti in laboratorio basandosi sulle specifiche di colloquio delle centraline dei vari sottosistemi.

Nel futuro dei driver è prevista la diffusione di una tecnologia già standardizzata denominata *OLE for Process Control* (OPC), tra cui figurano come creatori le maggiori software house mondiali di sistemi SCADA.

Il modulo che si occupa della gestione di ricette (o lotti o batch) deve essenzialmente gestire delle sequenze di operazioni pianificate, le cosiddette ricette. Tali sequenze possono essere fatte eseguire sulla base di scadenze temporali, al verificarsi di eventi particolari, o immediatamente alla richiesta dell'operatore. In genere l'uso delle ricette serve a impostare i dati di impianto per ciascun processo. Tali dati rappresentano i parametri che debbono essere forniti alle macchine che eseguono effettivamente le varie fasi del processo.

Un altro compito importante che viene di solito affidato ai sistemi di supervisione e acquisizione dati è quello di tenere traccia dei ritmi di produzione sia per il passato, attraverso la costruzione di serie storiche dei valori

di alcune variabili di processo, sia per il presente, attraverso la rappresentazione grafica in tempo reale dell'evoluzione di tali valori. La costruzione e la memorizzazione di serie storiche può essere utile per verificare lo stato dell'impianto o risalire alle cause di eventuali malfunzionamenti, e talvolta viene imposta da obblighi di legge.

2.2 Un componente fondamentale: il PLC

Il PLC (*Programmable Logic Controller*) è un dispositivo digitale industriale programmabile, specializzato nella gestione dei processi industriali. Esso esegue un programma ed elabora i segnali digitali ed analogici provenienti da sensori e diretti agli attuatori presenti in un impianto industriale.

Una delle principali caratteristiche è la sua robustezza, infatti normalmente è posto entro quadri elettrici in ambienti rumorosi, con molte interferenze elettriche, con temperature elevate o con grande umidità. In alcuni casi il PLC è in funzione 24 ore su 24, per 365 giorni all'anno, su impianti che non possono fermarsi mai.

Un altro dei vantaggi dei sistemi a logica programmabile rispetto a quelli a logica cablata è la flessibilità: nel caso in cui si voglia destinare il PLC ad un altro scopo è sufficiente modificare le istruzioni del programma, senza dover rifare nessun cablaggio di fili tra elementi logici e con il completo riutilizzo dell'hardware.

Altri aspetti rilevanti sono i ridotti tempi di manutenzione (grazie ad adeguati sistemi diagnostici), i limitati consumi di energia elettrica e il basso costo, che rendono i PLC economicamente vantaggiosi.

Il PLC è un oggetto hardware componibile, per cui la sua struttura viene adattata in base al processo da automatizzare. Durante la progettazione del sistema di controllo vengono scelte le schede adatte alle grandezze elettriche in gioco, che vengono quindi inserite sul rack del PLC.

L'evoluzione della tecnologia ha generato una notevole diversificazione di modelli, con capacità e velocità diverse in modo da coprire i vari segmenti di mercato. L'architettura dei modelli di fascia bassa è molto più semplice rispetto a quella dei modelli di fascia media o alta, ma la struttura base può essere descritta mediante lo schema in figura 2.3.

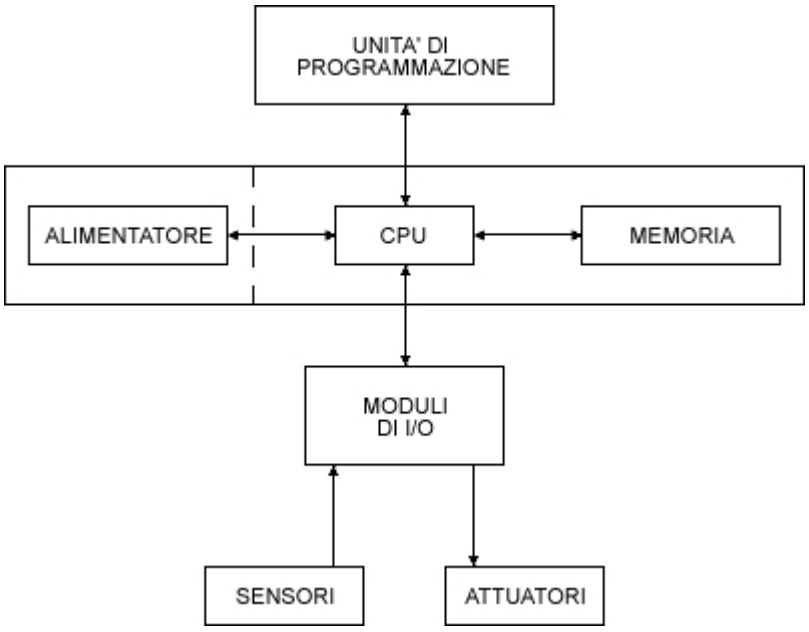


Figura 2.3: Schema a blocchi PLC.

Un PLC è composto da un alimentatore, dalla CPU, dalla memoria (contenente dati e programma), dalle schede di comunicazione e da un certo numero di schede di I/O, con un numero variabile di canali di ingresso e di uscita per la gestione di segnali sia di tipo digitale che analogico. Questi ultimi moduli sono collegati ai sistemi di rilevamento e di attuazione, che hanno lo scopo di generare entro la macchina automatica tutti i vari movimenti e operazioni richiesti.

Per i PLC di fascia media e alta sono inoltre disponibili moduli dedicati a particolari compiti di automazione, ad esempio contatori veloci, schede programmatori a cammes, moduli PID, moduli diagnostici, schede controllo assi. Il vantaggio nell'utilizzare tali schede è quello di avere il controllo di un'operazione o di un evento indipendentemente dal PLC, relegando quest'ultimo alla funzione di controllo e parametrizzazione.

La configurazione, la programmazione del PLC e il monitoraggio delle variabili durante l'esecuzione del programma avviene attraverso un'unità di programmazione, cioè una periferica collegata al PLC tramite una linea di comunicazione seriale (di solito RS232).

Nei PLC di fascia bassa può essere costituita da un semplice tastierino e un display di poche righe che permettono l'inserimento delle istruzioni del programma e la visualizzazione delle memorie del PLC, mentre nei sistemi di fascia alta è invece possibile collegare il PLC ad un personal computer e utilizzare pacchetti software di supporto forniti insieme al sistema operativo dalla casa costruttrice.

In questo modo si può configurare il sistema (interfaccia di I/O e definizione della relativa mappa di memoria e impostazione di altri parametri personalizzabili), scrivere il programma nel linguaggio prescelto, mandarlo in esecuzione, interromperlo e monitorarne l'esecuzione per individuare eventuali errori ed effettuare le relative correzioni.

2.3 Requisiti funzionali e vincoli

Il primo passo nella realizzazione di un progetto domotico deve essere quello di definire e classificare l'insieme di utenze, ossia di servizi, che il sistema deve gestire, in modo da poter analizzare i requisiti che esse pongono alla struttura degli strumenti di controllo.

In base ad un criterio funzionale le utenze possono essere suddivise in due settori principali:

- gestione ambientale, che coinvolge tutte quelle utenze legate al controllo delle condizioni ambientali e al miglioramento del comfort dell'utente. Tra queste sono presenti la distribuzione dell'energia, la climatizzazione, l'illuminazione automatica, l'azionamento remoto di sistemi di aperture e ingresso;

- gestione della sicurezza, che comprende sia sistemi anti-intrusione e anti-rapina che per la gestione di eventi potenzialmente pericolosi come fughe di gas, allagamenti, incendi ed eventi atmosferici.

  Questo settore rappresenta uno degli aspetti salienti nella realizzazione di un impianto di Building o Home Automation perché offre una protezione passiva, rilevando tutti i possibili eventi dannosi attraverso la propria rete di sensori e segnalando tempestivamente il problema all'utente o ad opportune strutture come Polizia, Vigili del Fuoco, 118.

Quello però che rende un impianto di Building o Home Automation "intelligente" è la possibilità di operare una sicurezza attiva; l'impianto è quindi in grado di reagire di fronte alla rilevazione di un allarme evitando o almeno limitando potenziali danni a persone o cose.

Lo scopo del progetto in cui si inserisce la presente tesi è quello di realizzare un controllore per Home Automation modulare, ingegnerizzato ed espandibile.

Per poter programmare il controllore e renderlo effettivamente funzionante si sono dovute però individuare alcune utenze tipo da inserire nell'ambiente virtuale in cui il sistema deve dimostrare le sue effettive potenzialità operative.

Sono state effettuate alcune ricerche sulle esigenze imposte dal mercato oggi esistente e si sono quindi individuate quali devono essere le funzionalità essenziali di un sistema di Home Automation affinché possa essere considerato appetibile per gli utenti finali. Qui di seguito sono riportate le caratteristiche delle quali si è tenuto conto nella realizzazione del progetto.

Temporizzazione Il sistema deve permettere di definire programmi temporizzati per tutte le utenze collegate: ad esempio per gestire le fasce orarie per l'accensione e lo spegnimento del riscaldamento o per attivare l'irrigazione del giardino ad uno o più orari prestabiliti; le programmazioni possono essere definite su base giornaliera, settimanale o annuale;

Rilevazione di presenza/movimento Nelle zone di passaggio come corridoi, rampe scale o camminamenti esterni è particolarmente utile prevedere il controllo automatico dell'illuminazione in funzione del movimento di persone. Il sistema di rilevamento presenze, realizzato attraverso comuni sensori ad esempio ad infrarosso, offre una duplice funzione: oltre a gestire la già citata illuminazione automatica, permette di realizzare senza ulteriori costi un semplice sistema di anti-intrusione;

Scenari possibili Il sistema deve offrire la possibilità di definire scenari

ossia combinazioni di stati e regolazioni per le diverse utenze. È possibile infatti gestire l'attivazione di diversi dispositivi con un singolo comando in modo da sgravare l'utente dalle numerose operazioni altrimenti necessarie.

Un esempio pratico potrebbe essere uno scenario chiamato "Modalità Notturna"; il sistema provvederebbe automaticamente all'abbassamento di tutte le tapparelle, alla regolazione della temperatura ideale nelle varie zone climatizzate, all'inserimento del sistema anti-intrusione e quant'altro fosse prestabilito dall'utente.

L'uso degli scenari può essere utilizzato non solo per applicazioni domestiche ma anche per il controllo di edifici destinati per esempio a contenere molteplici uffici come accade per il settore terziario; con un semplice comando sarebbe infatti possibile gestire l'insieme delle operazioni compiute quotidianamente come potrebbero essere lo spegnimento luci, l'inserimento antifurto, la disattivazione dell'impianto di climatizzazione e quant'altro richiesto dalle esigenze reali;

Comando a distanza Il comando a distanza offre all'utente la possibilità di gestire tutte le utenze in qualsiasi posizione si trovi all'interno o nelle immediate vicinanze dell'edificio attraverso un pannello di controllo senza fili, come ad esempio un touch-panel in cui comandi, controlli e segnalazioni sono rappresentati con simboli e scritte ben visibili e di facile comprensione;

Illuminazione Il sistema gestisce l'illuminazione di più zone indipendenti tra loro, si occupa sia degli ambienti interni che di quelli esterni. Una regolazione automatica dell'illuminazione permette un risparmio energetico attraverso tecniche di controllo dei parametri di luminosità di cui un semplice esempio è il sensore crepuscolare. Il controllo può anche prevedere la gestione degli avvolgibili e delle tende per permettere la regolazione assoluta anche delle fonti luminose naturali;

Climatizzazione Il controllo della climatizzazione di varie zone indipendenti tra loro è uno degli aspetti fondamentali. La gestione avviene sia

attraverso le impostazioni dell'utente (temporizzazioni su base giornaliera o settimanale) sia attraverso valutazioni automatiche effettuate autonomamente dall'impianto attraverso la propria sensoristica. Anche in questo viene promosso il risparmio energetico attraverso la valutazione di condizioni ambientali sfavorevoli all'inserimento dell'impianto di climatizzazione;

Telecontrollo L'intero impianto può essere controllato a distanza attraverso i più comuni mezzi di telecomunicazione come ad esempio internet, telefonia cellulare e telefonia fissa. Il telecontrollo permette in ogni istante la supervisione e la gestione completa dell'intero impianto, ad esempio consentendo all'utente di essere informato tempestivamente dell'attivazione di un allarme;

Rilevamento parametri meteo Rilevando i parametri meteo come pioggia, vento o temperatura esterna il sistema è in grado di intervenire automaticamente su varie utenze quali ad esempio tapparelle ed infissi, in modo da impedire eventuali danni che potrebbero essere prodotti da eventi atmosferici potenzialmente pericolosi;

Monitoraggio aperture Il sistema deve controllare lo stato di apertura o chiusura di serramenti come porte o finestre; ciò è essenziale sia per la gestione di un sistema di anti-intrusione perimetrale sia per esempio per coordinare l'apertura/chiusura dei serramenti in concomitanza con l'inserimento dell'impianto di climatizzazione;

Rilevamento allarmi Il sistema deve essere in grado di rilevare fenomeni tipo l'allagamento degli ambienti, le fughe di gas, la presenza di fumi o incendi ed intervenire tempestivamente. In seguito alla rilevazione di un evento potenzialmente pericoloso l'impianto deve immediatamente avvisare l'utente ed attuare le opportune contromisure quali ad esempio l'attivazione del sistema anti-incendio oppure la chiusura dell'elettrovalvola principale per l'erogazione dell'acqua nel caso di un allagamento.

2.4 La gestione delle variabili in gioco

Tutti i dati relativi alle misurazioni e rilevazioni effettuate da sensori, rilevatori, sonde e interruttori vengono digitalizzate da appositi moduli di input/output presenti sul PLC, che si occupano anche del procedimento inverso, cioè trasformare i parametri stabiliti dall'utente in segnali elettrici da inviare ai dispositivi di attuazione.

I segnali scambiati tra il sistema di controllo e la macchina si possono dividere in due grandi categorie:

- segnali analogici, che possono assumere solo due valori corrispondenti a due stati logici contrapposti (ON/OFF, +/-, ecc.);

- segnali discreti, che possono assumere con continuità tutti i valori compresi in un determinato range.

Per soddisfare le esigenze di collegamento con differenti tipi di sensori e attuatori, è disponibile sul mercato una gamma di moduli in grado di gestire segnali elettrici con diverse caratteristiche.

In un sistema di automazione si ha la necessità di gestire differenti tipologie di dati, ad esempio lo stato di un interruttore o di una valvola, il setpoint impostato per la temperatura di una stanza, l'indicatore di pressione dell'acqua, ecc., ognuno dei quali può assumere determinati valori o range di validità, quindi occorre costruire una struttura dati ad hoc per l'organizzazione di ogni tipo di variabile.

Un altro aspetto importante è la corrispondenza tra dati memorizzati all'interno dei registri del PLC e nel database: sono necessarie opportune funzioni di conversione nel software di sincronizzazione, che si occupa del trasferimento delle informazioni dal PLC al database e viceversa e della loro conversione.

Oltre a questo i dati devono essere presentati all'utente mediante un'interfaccia chiara e comprensibile e, attraverso determinati pulsanti e controlli, l'utente deve essere in grado di impostare i parametri desiderati.

2.5 I requisiti dei programmi

Per qualità del software si intende la misura in cui un prodotto software soddisfa un certo numero di aspettative rispetto sia al suo funzionamento sia alla sua struttura interna.

Gran parte della ricerca nel campo dell'ingegneria del software è dedicata, direttamente o indirettamente, al tema della qualità. In particolare, si è cercato di stabilire che cosa si intenda per qualità del software, definendo un insieme di parametri significativi e tecniche di misurazione rispetto a un dato sistema software, e di sviluppare tecnologie (per esempio linguaggi di programmazione) e metodologie (per esempio di analisi e progettazione) che facilitino la realizzazione di software di qualità.

I fattori rispetto a cui si può misurare o definire la qualità del software possono essere classificati in due famiglie: *parametri esterni* e *parametri interni*. I primi si riferiscono alla qualità del software così come è percepita dai suoi utenti, e includono:

Correttezza Un programma o sistema software si dice corretto se si comporta esattamente secondo quanto previsto dalla sua specifica dei requisiti, cioè se fa esattamente quello che è stato progettato per fare.

La correttezza è una qualità assoluta, ma sostanzialmente non è possibile stabilire con certezza se un sistema sia corretto;

Affidabilità Un sistema è tanto più affidabile quanto più raramente si manifestano malfunzionamenti durante l'uso del sistema, tenendo conto che, nella valutazione dell'affidabilità, errori gravi si considerano solitamente più influenti rispetto a quelli di minore importanza.

Poiché l'affidabilità è concettualmente misurabile, questo parametro viene spesso considerato la controparte "realistica" della correttezza;

Robustezza È la misura in cui il sistema si comporta in modo ragionevole in situazioni impreviste, non contemplate dalle specifiche. Situazioni di questo tipo in genere riguardano errori ed eccezioni di varia natura

(dati di input scorretti, fallimenti di componenti software o hardware esterni al sistema e interagenti con esso, ecc.);

Efficienza Un sistema è efficiente se usa memoria, CPU e altre risorse in modo proporzionato ai servizi che svolge, ovvero senza sprechi. Il termine *prestazioni* ha un significato correlato più specifico; le prestazioni, infatti, sono da considerarsi come uno degli elementi che potrebbe essere specificato dai requisiti (si parla in questo caso di *requisiti non funzionali*).

Efficienza e prestazioni sono difficilmente predicibili e quindi non raramente vengono prese in considerazione solo a sistema realizzato. Fra i modelli utilizzati per misurare l'efficienza di un sistema si possono citare quelli basati sulla complessità algoritmica, le misure sul campo, le misure su modelli matematici o modelli di simulazione;

Usabilità Un sistema si dice usabile se è semplice da utilizzare. È una qualita soggettiva perché dipende dal contesto e dall'esperienza dell'utente, esistono comunque principi condivisi secondo cui valutare il livello di usabilità di un'applicazione (ad esempio lo standard ISO 9241 sugli *ergonomic requirements*);

Scalabilità Indica l'adattabilità del sistema a diversi contesti con forti differenze di complessità, senza che questo richieda la riprogettazione dello stesso sistema.

I parametri interni si riferiscono invece alla qualità del software secondo aspetti legati agli sviluppatori, e comprendono:

Verificabilità Un sistema è verificabile se le sue proprietà di correttezza e affidabilità sono facili da controllare (una progettazione modulare del software aumenta il grado di verificabilità);

Manutenibilità Facilità di apportare modifiche al sistema realizzato (è maggiore se il software è ben progettato), non corrisponde solo alla correzione degli errori ma comprende anche l'evoluzione del software;

Evolvibilità e riusabilità Utilizzando tecniche di progettazione e di programmazione e introducendo una standardizzazione del processo di sviluppo (specifiche, documentazione, testing) è possibile ampliare un progetto in modo meno dispendioso o riutilizzarlo completamente o in parte;

Portabilità Un sistema è portabile se è in grado di funzionare in ambienti diversi. Questo è diventato un aspetto fondamentale perché consente di avere vantaggi economici, in quanto si possono ammortizzare i costi trasportando l'applicazione in diversi ambienti (un esempio sono le applicazioni web).

Non raramente esiste una correlazione fra questi due fattori (il software progettato e sviluppato male tende anche a dare problemi di funzionamento).

Un altro aspetto importante che porta alla qualità del software riguarda la gestione delle situazioni anomale. Sarebbe auspicabile che chi sviluppa un programma ponga una notevole cura nel prevedere tutte le possibili situazioni anomale che potrebbero insorgere durante l'esecuzione e nel predisporre le contromisure che il programma deve adottare in tali casi, per ridurre al minimo le conseguenze di tali anomalie.

La gestione "puntigliosa" di tutte le possibili situazioni anomale in tutti i possibili luoghi del codice in cui possono manifestarsi può rendere meno leggibile il codice ma è importante ai fini della robustezza e affidabilità del software.

2.6 Real time e vincoli temporali

Un sistema per l'automazione deve controllare un processo fisico in modo continuativo, eseguendo le operazioni di acquisizione dei segnali sensoriali, elaborazione dei dati ottenuti e attuazione dei segnali di controllo. Il tempo di elaborazione dati e' fondamentale per la correttezza del funzionamento del sistema di controllo, questo tipo di applicazioni viene detto infatti *Real Time System*.

In un sistema real time la correttezza di un'azione non dipende solo dal risultato logico della computazione, ma anche dal tempo in cui i risultati sono generati: se un'operazione non viene completata nel tempo prefissato (deadline) fallisce o comunque provoca la diminuzione delle prestazioni del sistema.

Nella progettazione di una applicazione real time è necessario effettuare un'analisi per verificare che i vincoli temporali siano rispettati, determinando quali possono essere i fattori di rallentamento dell'applicazione (ad esempio il traffico sulla rete, le prestazioni dell'hardware utilizzato, la velocità di elaborazione dei dati o il tempo necessario per la comunicazione tra i vari componenti) ed eventualmente confrontando diverse metodologie per risolvere il problema.

Nel caso di applicazioni domotiche il principale vincolo temporale da soddisfare consiste nel rendere accettabili i tempi di attesa da parte dell'utente, da quando viene inviato un comando al momento in cui questo viene effettivamente svolto dai sistemi di attuazione.

Un altro possibile fattore di rallentamento può essere il tempo di risposta del PLC, cioè il massimo intervallo di tempo che passa tra la rilevazione di un certo evento e l'esecuzione dell'azione di risposta per esso programmata. Il tempo di risposta quindi tiene conto anche dei ritardi introdotti dai moduli di I/O.

2.7 Automazione e reti: integrazione

Nell'ambito dell'automazione occorre fornire due precise indicazioni:

- qual è la reale posizione dell'operatore rispetto all'architettura di processo;
- qual è la reale posizione del PLC rispetto all'architettura di processo.

Le risposte ad entrambi i punti forniscono uno strumento più approfondito per l'analisi dei possibili mezzi di comunicazione e quindi una scelta ottimale della architettura di rete. In un ambiente domestico, così come

	Massima portata (m)	Velocità di trasferimento (kbps)	Costo (€/m)	Tipo di applicazione
RS232	1,5	$0,3 \sim 3$	5	generica
PROFIBUS	100	$1 \sim 10^2$	2,5	industriale
ETHERNET	90	$10^3 \sim 10^6$	0,5	generica

Tabella 2.1: Comparazione bus di comunicazione.

in quello industriale, è possibile che il raggio d'azione dell'operatore possa essere di qualche metro (quindi si parla di controllo locale), oppure di qualche chilometro (controllo remoto).

Per entrambi è possibile individuare una serie di protocolli standard di comunicazione, in tabella 2.1 viene riportata una comparazione tra alcuni tipi di bus presenti sul mercato, indicandone le principali caratteristiche.

Lo standard Ethernet, noto anche come IEEE 802.3, grazie alla crescita di internet, è diventato il protocollo più utilizzato in applicazioni/reti in ambienti office, ma ultimamente ha avuto una buona diffusione anche nel campo dell'automazione, dove è utilizzato per il controllo e monitoraggio di impianti e sistemi grazie al grande numero di dispositivi ed applicazioni disponibili sul mercato, alla sua economicità ed affidabilità.

La connettività Ethernet offre integrazione di alto livello in sistemi informatici: l'inserimento all'interno di strutture preesistenti garantisce l'interazione del prodotto con le risorse della rete e agevola l'implementazione delle funzioni di controllo remoto. Inoltre, può essere previsto l'utilizzo di tecnologie wireless da utilizzare nei casi in cui si vogliano evitare interventi invasivi sull'edificio esistente.

Anche in ambito domestico un sistema affiancato ad internet permette il controllo dell'intero apparato anche da remoto. Attraverso un gateway è possibile fornire una connessione permanente e a banda larga per consentire l'accesso all'architettura di processo o ai computer dell'abitazione, la trasmissione di dati e il controllo remoto (monitoraggio della temperatura, verifica degli allarmi attivi, gestione dell'impianto di illuminazione, ecc.).

Capitolo 3

I sistemi di supervisione

Un sistema di supervisione viene generalmente impiegato per controllare una realtà complessa, dove l'acquisizione di una grande quantità di dati in tempo reale, la sua elaborazione, interpretazione e presentazione sono essenziali per il funzionamento di un'attività, dal momento che le decisioni riguardanti la conduzione vengono prese proprio in base a tali dati e la loro raccolta non può essere effettuata manualmente.

Verranno illustrati i componenti che costituiscono un sistema di supervisione, come avviene la comunicazione tra essi e quali requisiti devono essere soddisfatti.

3.1 Tipologie di HMI

In un sistema di supervisione è importante il modo in cui avviene l'interazione uomo-macchina.

I dispositivi HMI (*Human Machine Interface*) stanno assumendo un ruolo sempre più decisivo nelle applicazioni di automazione (sia industriale che civile). Infatti, la possibilità di visualizzare in tempo reale messaggi diagnostici, allarmi o istruzioni per l'operatore e, al contempo, modificare i parametri operativi in modo semplice e diretto, è diventata un'esigenza essenziale nella maggioranza delle applicazioni.

Nel caso dell'attività di controllo di un edificio l'utente dovrà essere in grado di visualizzare lo stato degli impianti elettrici, riscaldamento e con-

dizionamento, di impartire comandi alle macchine ed effettuare regolazioni. L'interazione uomo-macchina avviene attraverso la riproduzione di tutti i pannelli di controllo dei regolatori di temperatura, quadri elettrici, allarmi antincendio, antintrusione, ecc.. Inoltre è il supervisore a farsi carico del controllo del regolare funzionamento degli apparati, della segnalazione di anomalie e dell'esecuzione o suggerimento di azioni correttive.

Inoltre si avverte sempre più l'esigenza di visualizzare, controllare e interagire con i sistemi in modo remotizzato, staccandosi dalla classica visione del controllo industriale a bordo macchina (pannelli operatore).

La tendenza del mercato HMI riflette le linee di evoluzione introdotte dal fenomeno della convergenza digitale: le applicazioni e i dispositivi si sono fatti sempre più flessibili e potenti, consentendo *remotizzazione*, *multicanalità*, *mobilità*, *personalizzazione* e *adattamento* a diverse tipologie di rete, mentre le interfacce utente e le architetture applicative si sono uniformate ad alcuni tra i principali standard tecnologici emersi sul mercato (browser, architetture basate su IP, architetture a tre livelli, Web services, ecc.).

3.2 I vincoli relativi alle interfacce utente

Con l'avvento dei servizi Web come punti centralizzati per la condivisione di informazioni e dati, gli utenti hanno sentito la necessità di accedere e gestire le informazioni di lavoro e personali da qualsiasi luogo, in qualsiasi momento e con qualsiasi dispositivo (computer, palmare, cellulare, televisione, ecc.), di conseguenza la progettazione di applicazioni per dispositivi si è considerevolmente evoluta.

Quando si progetta l'interfaccia di una applicazione è necessario considerare molti elementi relativi al dispositivo di destinazione, come la risoluzione, la dimensione e gli angoli di visualizzazione dello schermo, l'ubicazione e la dimensione di destinazione, i colori e i caratteri, in quanto le applicazioni vengono visualizzate diversamente a seconda del dispositivo.

È bene considerare come si stiano sempre più accentuando le differenze tra i dispositivi stessi e i PC desktop e il tipo di attività che gli utenti

desiderano effettuare. L'obiettivo consiste nello sviluppare applicazioni più rapide, produttive, semplici ed esteticamente gradevoli.

I dispositivi si distinguono tra loro e dai computer desktop per diversi fattori:

Dimensioni I dispositivi hanno spesso dimensioni molto più ridotte rispetto a quelle dei PC desktop;

Funzionalità di visualizzazione Mentre la risoluzione dei monitor tradizionali è di almeno 800×600 pixel, per altri dispositivi è inferiore (ad esempio per un palmare 320×240, per una televisione 720×576) e anche la scala di grigi e il numero di colori sono limitati;

Metodi di input e output I dispositivi consentono di utilizzare diversi metodi di interazione, quali pulsanti hardware, stilo, voce o mouse e tastiera.

Per migliorare l'accessibilità dell'applicazione è consigliabile tenere in considerazione questi requisiti durante la progettazione dell'interfaccia.

Un altro aspetto da valutare è l'analisi del tipo di attività che può essere svolta dall'utente, della rapidità con cui tali attività vengono completate e alle prestazioni di applicazioni e dispositivi.

Le attività eseguite dagli utenti possono variare a seconda del dispositivo utilizzato, oppure la stessa attività può essere svolta in modo diverso o con esigenze particolari, è pertanto necessario progettare l'interfaccia dell'applicazione in modo da semplificare il completamento delle attività definite dall'utente, anziché pensare all'applicazione in termini di funzionalità.

A questo scopo, è possibile riprodurre un'interfaccia simile o correlata alle attività compiute dagli utenti con il dispositivo, rendere l'applicazione gradevole esteticamente sapendo quali elementi è possibile visualizzare sullo schermo e quali no, prevedere utilità di accesso facilitato.

3.3 I requisiti utente

I requisiti principali che deve possedere l'interfaccia utente sono *usabilità* e *accessibilità*.

Con il termine usabilità si intende il grado in cui un prodotto può essere usato da particolari utenti per raggiungere certi obiettivi con efficacia, efficienza e soddisfazione in uno specifico contesto d'uso.

Il problema dell'usabilità è emerso dapprima negli anni '80, con la diffusione delle tecnologie informatiche a livello di ufficio e di famiglia, ed è definitivamente esploso negli anni '90, con la diffusione del personal computer. Mentre prima i principali utilizzatori dei prodotti software finivano per essere gli stessi progettisti o persone esperte con una formazione simile ai progettisti, ora gli utenti finali del software (ma naturalmente anche dell'hardware) non sono necessariamente esperti di informatica.

L'usabilità nasce dunque soprattutto come ausilio alla progettazione: l'obiettivo è fare in modo che il modello mentale di chi ha progettato il software (*design model*), da cui deriva il suo reale funzionamento, corrisponda il più possibile al modello mentale del funzionamento del software così come se lo costruisce l'utente finale (*user model*).

Le tecniche di usabilità tentano dunque di porre al centro dell'attenzione progettuale proprio l'utente: ad ogni sua azione l'interfaccia proporrà un risultato, un cambiamento di stato, ma ai fini dell'usabilità non importa come l'interfaccia sia giunta a quello stato, attraverso cioè quali meccanismi di programmazione.

L'accessibilità consiste nella possibilità di rendere fruibili i contenuti dei siti ad utenti disabili o con dotazioni tecnologiche obsolete o poco comuni.

La WAI (*Web Accessibility Initiative*), una sezione del W3C, ha prodotto una serie di raccomandazioni tecniche, mirate a dare agli sviluppatori gli strumenti per rendere accessibili non solo i contenuti del Web, ma anche i programmi per navigare in Rete nonché quelli utilizzati per produrre e pubblicare contenuti. Queste linee guida sono conosciute come WCAG (acronimo di *Web Content Accessibility Guidelines*) e sono giunte attualmente alla versione 1.0, rilasciata dal WAI-W3C come documento ufficiale con valore normativo in data 5 maggio 1999.

I metodi suggeriti dalle WCAG riguardano soprattutto interventi sul codice, che consentano di rendere la struttura della pagina il più possibile flessibile, in modo che i suoi contenuti possano essere fruiti senza perdita d'informazioni per mezzo dei più diversi dispositivi di navigazione: dai normali browser grafici ai browser testuali, dai sintetizzatori vocali alle barre Braille, dai computer palmari ai robot di ricerca, dai telefoni cellulari agli ingranditori di schermo usati dagli ipovedenti.

3.4 Come il canale condiziona l'interfaccia

Un altro degli aspetti che condizionano la progettazione dell'interfaccia sono i fattori fisici, come il traffico presente sulla rete, l'hardware utilizzato e la comunicazione tra i vari componenti.

Se il sistema di supervisione viene eseguito in locale il tempo necessario affinché un comando inviato dall'utente venga effettivamente eseguito sarà ridotto, mentre se vengono utilizzati sistemi di controllo remoti si dovrà tenere presente che il carico della rete può essere variabile, e se elevato provoca un aumento dei tempi di attesa, sia per quanto riguarda l'attuazione dei comandi, sia per il caricamento delle informazioni (grafica e dati).

Oltre al traffico di rete, la comunicazione tra i vari componenti che formano il sistema incide sul tempo di risposta dell'applicazione, dato che il trasferimento dei dati avviene in più passaggi: dai controlli presenti nell'interfaccia al database, dal database al PLC e viceversa.

Inoltre anche l'hardware utilizzato influisce, sebbene in misura minore, sulle prestazioni dell'applicazione, sia per quanto riguarda il lato client (computer, set top box o altri dispositivi) e il lato server, sia per il PLC (sensori di rilevazione, moduli di I/O, dispositivi di attuazione).

3.5 La persistenza e la storicizzazione dei dati

L'utilizzo di un database consente di memorizzare e organizzare tutto ciò che riguarda l'applicazione, garantendo la persistenza dei dati (indipen-

dentemente dall'esecuzione dell'applicazione che lo utilizza) e la loro condivisione tra i componenti che costituiscono il sistema.

Un DBMS (*DataBase Management System*) efficiente cerca di utilizzare al meglio le risorse di spazio di memoria e di tempo, offrendo funzionalità articolate, potenti e flessibili, al fine di consentire la progettazione e l'utilizzo di database veloci ed affidabili (i dati sono risorse fondamentali, da conservare a lungo termine).

Tra le altre proprietà che dovrebbe assicurare un DBMS ci sono quelle della privatezza e dell'affidabilità: un database è una risorsa che può essere in comune tra più applicazioni, quindi si dovrebbero prevedere meccanismi di autorizzazione con assegnamento agli utenti di privilegi per l'accesso e la modifica di determinati dati condivisi, anche effettuando controlli delle concorrenze. L'affidabilità viene invece ottenuta mediante la gestione delle transazioni.

Una transazione è una sequenza di operazioni eseguite sul database che possiede le seguenti proprietà:

Atomicità La transazione è indivisibile nella sua esecuzione e la sua esecuzione deve essere totale o nulla, non sono ammesse esecuzioni intermedie;

Consistenza Quando inizia una transazione il database si trova in uno stato consistente e quando la transazione termina il database deve essere ancora in uno stato consistente, ovvero non deve violare eventuali vincoli di integrità;

Isolabilità Ogni transazione deve essere eseguita in modo isolato e indipendente dalle altre transazioni, l'eventuale fallimento di una transazione non deve interferire con le altre transazioni in esecuzione;

Durabilità Detta anche persistenza, si riferisce al fatto che una volta che una transazione ha richiesto un *commit work*, i cambiamenti apportati non dovranno essere più persi. Per evitare che nel lasso di tempo fra il momento in cui la base di dati si impegna a scrivere le modifiche e quello in cui li scrive effettivamente si verifichino perdite di dati

dovuti a malfunzionamenti, vengono tenuti dei registri di log, dove sono annotate tutte le operazioni sul database.

Le proprietà di atomicità e durabilità vengono implementate utilizzando sistemi di recovery-management, mentre l'isolamento viene raggiunto attraverso schemi di controllo delle concorrenze, che generano sequenze che indicano l'ordine cronologico in cui le istruzioni di transazioni concorrenti devono essere eseguite (*schedule*). Possono essere utilizzati diversi protocolli che utilizzano lock, grafi di precedenza, la data di inizio della transazione o altre tecniche di validazione (ad esempio il protocollo 2-phase commit). Per maggiori informazioni si veda [1] e [2].

Un database può contenere, oltre ai dati in senso stretto (come le impostazioni relative alle utenze gestite nel caso di un sistema di automazione), anche i parametri di configurazione dell'applicazione, ad esempio informazioni relative al PLC o al server (indirizzo IP, stato, eventuali messaggi di errore), impostazioni sulla lingua utilizzata, data e ora, descrizione degli allarmi e dei possibili messaggi di errore/warning.

Un aspetto che deve essere tenuto in considerazione riguarda la storicizzazione dei dati, ovvero l'insieme di tutte quelle informazioni che permettono di creare un quadro temporale su ciò che è accaduto al sistema. In questo contesto è possibile scegliere di memorizzare, oltre che gli eventuali errori del sistema, anche gli allarmi generati in ogni determinato istante, in modo da ottenere uno storico disponibile in un qualsiasi momento.

Si può inoltre prevedere di tenere traccia della navigazione effettuata dall'utente, memorizzando informazioni quali la data di accesso e di uscita dal sistema, il percorso effettuato e i dati visualizzati/modificati.

Bisogna però tenere presente che in questo caso verrà raccolta una grande mole di dati, per cui saranno necessarie procedure periodiche di eliminazione delle informazioni meno recenti o non più necessarie.

3.6 La comunicazione con il PLC

Come illustrato precedentemente la comunicazione tra SCADA e PLC avviene principalmente mediante connessione seriale o Ethernet, ovviamente è

però necessario anche un protocollo di comunicazione che permetta l'invio di comandi e dati tra i componenti del sistema.

I protocolli di livello superiore possono essere differenti a seconda del tipo di applicazione e non è garantita l'interoperabilità tra i differenti standard.

Le più diffuse implementazioni sono: *MODBUS/RS-232*, *Modbus/TCP* (Modbus su Ethernet o Open Modbus), *Ethernet/IP* (ControlNet e DeviceNet su Ethernet) e *Profinet* (Profibus su Ethernet). L'impiego di software in grado di interfacciare i diversi protocolli al livello di applicazione permette di integrare tra loro componenti di diversi produttori.

Modbus è un protocollo stateless a livello di applicazione (vedi figura 3.1) di tipo client/ server, basato su transazioni che comprendono una richiesta da parte del client (*master*) e una risposta inviata dal server (*slave*) (vedi figura 3.2).

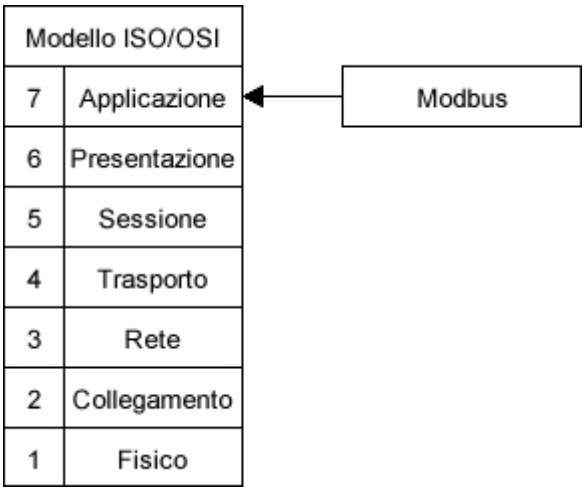


Figura 3.1: Protocollo Modbus.

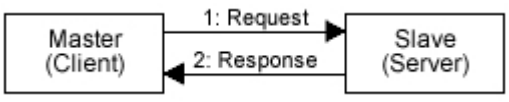


Figura 3.2: Transazione master/slave.

La comunicazione è basata su semplici pacchetti, chiamati *Protocol*

Data Unit (PDU). Le specifiche definiscono tre tipi di PDU:

- *Request PDU*, formato da un campo che specifica il codice della funzione (dimensione 1 byte) e da un campo dati di lunghezza variabile;
- *Response PDU*, composto da un campo che specifica il codice della richiesta (dimensione 1 byte) e da un campo dati di lunghezza variabile;
- *Exception Response PDU*, formato da un campo indicante il codice della richiesta + 0x80 (dimensione 1 byte) e da un campo contenente il codice dell'eccezione (dimensione 1 byte).

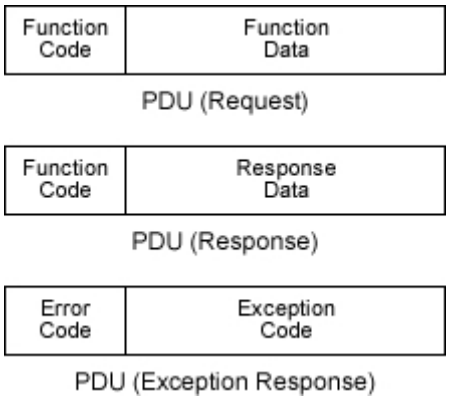


Figura 3.3: Pacchetti PDU.

Le funzioni sono state sviluppate per lo scambio di dati tipici dell'ambito dell'automazione, rappresentati in tabella 3.1.

PROFINET, inizialmente sviluppato da Siemens, è uno standard per l'automazione industriale basato su Ethernet (IEC 61158). A tale scopo utilizza il protocollo TCP/IP e rende possibile l'automazione in tempo reale. Con PROFINET è possibile collegare dispositivi di campo decentralizzati e stabilire sistemi di automazione distribuiti basati su componenti.

Sistemi di bus di campo esistenti, come PROFIBUS, possono essere semplicemente integrati senza necessità di modificare le apparecchiature esistenti.

Per quanto riguarda la comunicazione tra SCADA e PLC si può stabilire che la logica funzionale sia situata all'interno del PLC, cioè che questo

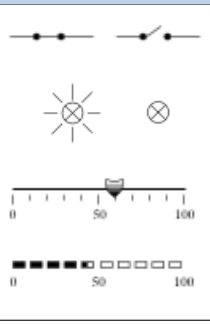
Nome	Tipo	Accesso	Esempio
Discrete Input	single bit	read-only	
Discrete Output	single bit	read-write	
Input Registers	16-bit word	read-only	
Holding Registers	16-bit word	read-write	

Tabella 3.1: Tipi di dato utilizzati nell'ambito dell'automazione.

sia l'unico strumento di elaborazione delle informazioni. Tuttavia questo approccio può risultare molto lento, dato che la capacità di elaborazione di un PLC è inferiore rispetto a quella di un normale computer [3] [4].

Perciò in fase di progettazione si può pensare di inserire un componente intermedio strutturato a blocchi, ciascuno dei quali si occupa della gestione di una determinata utenza. Questo componente sarà situato su un computer che include le funzionalità di base per lo scambio delle informazioni tra l'utente e il PLC e, grazie ad un web-server installato, per il controllo da remoto.

Un particolare computer di questo tipo viene detto *Set Top Box* (STB): fisicamente può essere paragonato ad un decoder digitale, ma al suo interno sono contenuti i comuni componenti di un pc: processore, hard-disk, RAM, scheda video, ecc. Per quanto riguarda i controlli utente è possibile collegare al STB sia le normali tastiere e mouse, sia avanzati sistemi di puntatori, come un telecomando wireless. In sostanza è uno strumento collegato in rete come tutti gli altri apparati che sono presenti nell'ambiente domestico (personal computer, telecamere OverIP, router Adsl).

Al software presente sul STB vengono assegnati compiti molto importati, tra cui:

- verifica del corretto funzionamento del controllore presente sul PLC;
- sincronizzazione dei dati tra database e memoria del PLC (con questa funzionalità si ottiene il disaccoppiamento tra le due entità);

- impostazione delle interfacce utente per i vari strumenti di visualizzazione (TV, monitor LCD);
- gestione dei controlli utente.

3.7 Sistemi di rilevamento e di attuazione

Al PLC è affidata la funzione di controllo in tempo reale di una macchina o di un processo. Per poter assolvere in modo efficace a questi compiti sono state sviluppate due importanti funzionalità:

- acquisizione e gestione dei segnali in ingresso e in uscita provenienti dai sensori e diretti agli attuatori che si trovano nel sistema da controllare, mediante appositi moduli hardware idonei alla conversione di segnali elettrici a valori digitali e viceversa;
- elaborazione della risposta ad un determinato evento previsto da programma in tempi molto ridotti (generalmente dell'ordine di alcuni centesimi di secondo), al fine di garantire stabilità all'interno dell'anello di controllo macchina - PLC - macchina.

Tipicamente un PLC, nell'interfacciarsi con il mondo esterno, è in grado di gestire i seguenti segnali:

Ingressi digitali (DI) Sono segnali provenienti da contatti, pulsanti, termostati, ecc., che tipicamente hanno tensione $0V$ se OFF e $+24Vdc$ se ON;

Uscite digitali (DO) Sono i segnali con i quali il PLC comanda (tramite relé ausiliari e/o contattori) gli attuatori, come ad esempio motori, elettrovalvole, segnalazioni, ed altri circuiti;

Ingressi analogici (AI) Sono segnali provenienti da trasduttori di pressione, portata, o termometri, igrometri, analizzatori chimici, analizzatori di energia elettrica e altri strumenti che trasducono la grandezza fisica analizzata in un segnale elettrico proporzionale (tipicamente $4-20mA$ o anche $0-10V$);

Uscite analogiche (AO) Sono segnali atti a pilotare valvole proporzionali, strumenti indicatori, registratori, regolatori di velocità per motori (drives o inverter) e altre apparecchiature regolatrici.

I moduli di I/O presenti sul PLC rappresentano l'interfaccia tra i segnali elettrici provenienti dalle apparecchiature di campo ed i segnali digitali che viaggiano sul bus dati interno del PLC, all'interno di essi avviene la conversione dei valori memorizzati in forma binaria nei registri in segnali elettrici per comandare degli attuatori. Viceversa, i valori di tensione o di corrente provenienti dai sensori vengono tradotti in valori numerici espressi in forma binaria.

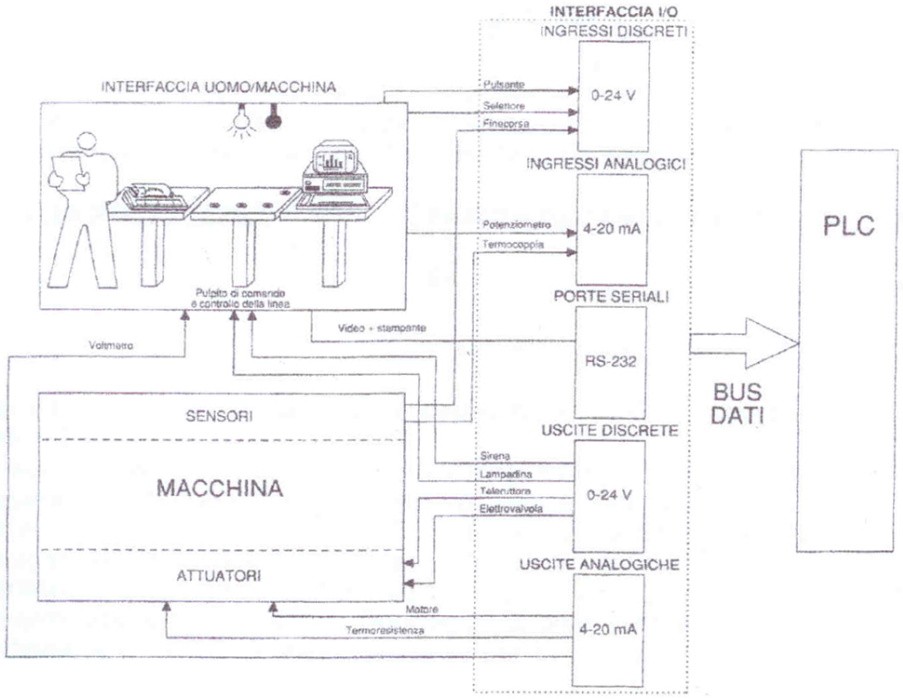


Figura 3.4: Schema a blocchi di un sistema gestito da PLC.

In un sistema di controllo in tempo reale è indispensabile acquisire le variabili in input con la massima risoluzione possibile, in quanto il rilevamento di ogni minima variazione consente una maggiore tempestività nella risposta. La maggiore risoluzione possibile si ottiene sfruttando tutte le linee a disposizione sul bus dati: ad esempio, se si ha a disposizione un bus

a 8 linee e i valori di tensione rilevabili sono compresi nel range $0 - 24V$, si potranno mappare 256 possibili valori di tensione, con una risoluzione di $24/255 = 0.094V$.

3.8 MVC ed approccio multicanale applicato alle HMI

L'intento del design pattern *Model View Controller* (MVC) è di disaccoppiare il più possibile tra loro le parti dell'applicazione adibite al controllo, all'accesso ai dati e alla loro presentazione (vedi [5] e [6]). Questo approccio porta a diversi vantaggi quali:

- indipendenza tra business data (*model*), logica di presentazione (*view*) e logica di controllo (*controller*);
- separazione dei ruoli e delle relative interfacce;
- viste diverse per il medesimo model;
- maggiore semplicità per il supporto a nuove tipologie di client: basta scrivere la vista ed il controller appropriati riutilizzando il model esistente.

Mediante il paradigma MVC vengono identificati i tre componenti fondamentali di una applicazione interattiva:

Model Individua la rappresentazione dei dati dell'applicazione e le regole di business con cui viene effettuato l'accesso e la modifica a tali dati. Il modello non è a conoscenza dei suoi controller e tanto meno delle sue view; non contiene riferimenti ad essi, ma è il sistema che si prende la responsabilità di mantenere i link tra il modello e le sue view e di notificare a quest'ultime le variazioni nei dati del modello;

View È la presentazione visuale dei dati all'utente e interagisce con il modello attraverso un riferimento ad esso. Uno stesso modello può quindi essere presentato secondo diverse viste;

Controller È colui che interpreta le richieste della view in azioni che vanno ad interagire con il model (di cui possiede un riferimento), aggiornando conseguentemente la view stessa.

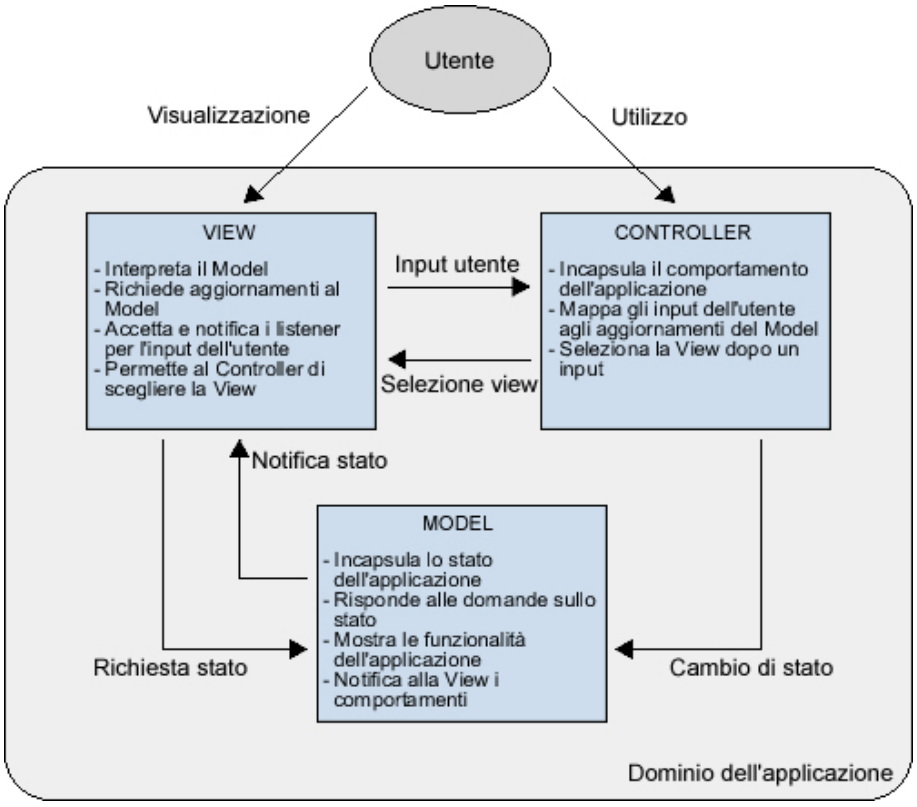


Figura 3.5: Schema funzionale MVC.

La suddivisione in layer dell'applicazione permette la progettazione e la programmazione dei vari componenti in modo indipendente tra loro, collegandoli solamente in runtime. Se un componente dovesse diventare inutilizzabile, può essere sostituito senza influire sugli altri.

In alcuni contesti è necessario sviluppare diverse interfacce di presentazione dei dati che possano essere interpretate correttamente da dispositivi tecnologici differenti da un PC, quali un telefono cellulare, un palmare o un PDA.

In questo caso, separare la business logic dalla presentation logic permette di sviluppare nuove interfacce grafiche senza dover intervenire sulla logica dell'applicazione, rendondene più facile l'evoluzione e la manutenzione. Ad esempio la stessa applicazione può essere sviluppata in due versioni, GUI e web, che si differenziano solo per l'interfaccia: la prima

utilizzerà finestre grafiche tradizionali, la seconda pagine web[1], mentre la business logic rimane comune ad entrambe.

[1] In Java è stata introdotta un'evoluzione di questo pattern, il MVC2, nel quale la view è implementata da una pagina JSP, mentre il controller è rappresentato da un Servlet che, tramite gli oggetti HttpServletRequest e HttpServletResponse, si occupa della gestione dei dati da/per la view.

Capitolo 4

La progettazione informatica

In questo capitolo verrà illustrato il paradigma di programmazione orientato agli oggetti, con particolare riferimento al linguaggio Java, l'utilizzo di UML nel contesto della progettazione del software e l'importanza di avere a disposizione un ambiente di sviluppo potente e flessibile.

4.1 Il paradigma di sviluppo orientato agli oggetti

Attualmente lo sviluppo del software si basa sul paradigma della *programmazione orientata agli oggetti* (OOP), che nasce negli anni '60/'70 con i primi linguaggi di questo tipo (Simula, SmallTalk), ma si diffonde a partire dagli anni '80 con l'introduzione di metodologie di programmazione più adatte a gestire la complessità dei nuovi progetti software.

In questo contesto vegono definiti nuovi concetti come quelli di *oggetto* e di *classe* per definire gli oggetti.

Nel paradigma di programmazione orientata agli oggetti un oggetto è caratterizzato da uno stato (cioè una n-upla di valori memorizzati negli *attributi*) e da un insieme di funzioni (che vengono chiamati *metodi* o servizi).

Durante la fase di progettazione del software non è possibile stabilire fin dall'inizio tutti gli oggetti necessari, sia perché i dati effettivi non sono ancora noti o comunque l'informazione relativa agli oggetti è pittosto generica. La definizione di una classe consente invece di definire un ogget-

to soltanto mediante i tipi dei propri attributi, mentre i valori concreti che identificano l'oggetto sono variabili. Da una classe possono essere generati più oggetti, che prendono il nome di *istanze* della classe.

Questo meccanismo ha il vantaggio di rappresentare solo gli oggetti veramente necessari e consente di implementare i metodi degli oggetti solo una volta per classe. Inoltre la definizione dei tipi di dato all'interno della dichiarazione di una classe permette al compilatore di effettuare controlli sui tipi.

Gli oggetti e le classi di un sistema orientato agli oggetti si basano sui principi di:

Identità dell'oggetto Ogni oggetto, durante il suo ciclo di vita, rappresenta un'identità univoca in tutto il sistema. Due oggetti sono uguali se gli attributi di entrambi hanno gli stessi valori, ma non necessariamente hanno la stessa identità. Due oggetti identici sono invece lo stesso oggetto (ad esempio due reference che si riferiscono allo stesso oggetto);

Information hiding La descrizione interna dei dati di un oggetto non deve essere visibile all'esterno, né in lettura né in scrittura, ma è resa accessibile soltanto definendo opportune interfacce (metodi *get* e *set*, vedi figura 4.1). Nei linguaggi orientati agli oggetti questo principio viene ottenuto utilizzando determinati specificatori di accesso, che consentono diversi livelli di protezione.

Legato al concetto di information hiding è quello di *structure hiding*, che consiste nel disaccoppiare fra loro gli elementi strutturali di un sistema (un esempio è il software di sincronizzazione dati tra database e memoria del PLC);

Ereditarietà Gli oggetti con strutture dati o comportamenti simili possono essere organizzati secondo un certo livello gerarchico, ereditando o lasciando in eredità attributi o metodi. Un software che utilizza l'ereditarietà, se ben progettato, consente di risparmiare tempo nello sviluppo sfruttando i concetti di *generalizzazione* e *specializzazione*;

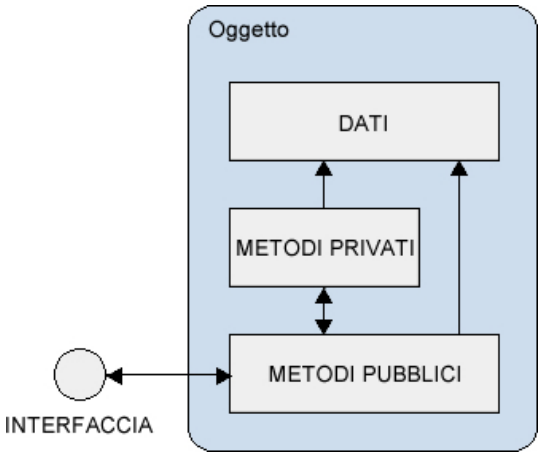


Figura 4.1: Occultamento delle informazioni interne all'oggetto.

Polimorfismo Metodi con lo stesso nome hanno una semantica diversa, secondo il contesto in cui vengono richiamati (concetto di *overriding*).

La progettazione del software secondo il paradigma orientato agli oggetti cerca di ottenere i seguenti obiettivi:

- la scomposizione del problema in sottoproblemi, ognuno dei quali viene risolto attraverso un singolo modulo software. La soluzione al problema generale è ottenuta mediante l'interazione tra questi componenti, che espongono interfacce ben definite;
- riunione delle classi in archivi per facilitarne il riutilizzo;
- riuso del prodotto a vari livelli (tutta l'applicazione o solo alcuni componenti, documenti di analisi o di progetto, ecc.).

4.2 UML e il contesto dell'automazione

Durante gli anni '90 furono introdotte nel mercato dell'Information Technology parecchie metodologie per il design e la progettazione di sistemi software. Vi era un problema, però: ognuna di queste tecnologie aveva il suo insieme proprio di notazioni e simboli che differiva, a volte in modo rilevante, dalle altre. In particolare, eccellevano tre di queste metodologie:

- OMT (Rumbaugh);
- Booch 1991;
- OOSE (Jacobson).

Ognuno di questi metodi aveva, naturalmente, i suoi punti di forza e i suoi punti deboli. Ad esempio, l'OMT si rivelava ottimo in analisi e debole nel design. Booch 1991, al contrario, eccelleva nel disegno e peccava in analisi. OOSE aveva il suo punto di forza nell'analisi dei requisiti e del comportamento di un sistema ma si rivelava debole in altre aree.

Successivamente, Booch scrisse il suo secondo libro che adottava i principi di analisi utilizzati da Rumbaugh e Jacobson nelle loro rispettive metodologie. A sua volta, Rumbaugh pubblicò una serie di articoli (conosciuti come OMT-2) che descrivevano parecchie delle tecnologie di disegno di Booch. In sostanza, i tre metodi stavano convergendo verso un'unica visione che incorporasse le qualità migliori che ognuno di essi aveva mostrato. L'unico problema che restava era il fatto che ogni metodo portava ancora con sè la propria notazione.

Tale problema non era da sottovalutare in quanto l'uso di simbologia differente portava facilmente confusione sul mercato a causa del fatto che un determinato simbolo poteva avere un significato differente per analisti e disegnatori differenti.

Finalmente, dopo un periodo di tempo in cui andò avanti la cosidetta "guerra della metodologia" ci si rese conto che era assolutamente necessario produrre uno standard che unificasse anche la notazione utilizzata. Fu così che, nell'Ottobre del 1995, nacque la prima bozza dell'UML (*Unified Modeling Language*) [7], ovvero l'unificazione delle notazioni e delle idee prodotte da Booch, Rumbaugh e Jacobson per modellare un sistema software. La prima versione ufficiale, prodotta dall'OMG (*Object Management Group*) [8] fu rilasciata nel Luglio del 1997 e nel Novembre dello stesso anno l'UML venne adottato come standard.

Elenchiamo, qui di seguito, alcuni dei benefici derivanti dall'utilizzo del linguaggio UML:

- un sistema software, grazie al linguaggio UML, viene *disegnato professionalmente* e documentato ancor prima che ne venga scritto il relativo codice da parte degli sviluppatori. Si sarà così in grado di conoscere in anticipo il risultato finale del progetto su cui si sta lavorando;

- poiché la fase di disegno del sistema precede la fase di *scrittura del codice*, ne consegue che questa è resa più *agevole* ed *efficiente*, oltre al fatto che in tal modo è più facile scrivere del codice riutilizzabile in futuro. I costi di sviluppo, dunque, si abbassano notevolmente con l'utilizzo del linguaggio UML;

- è più facile *prevedere e anticipare eventuali "buchi"* nel sistema. Il software che si scrive, si comporterà esattamente come ci si aspetta senza spiacevoli sorprese finali;

- l'utilizzo dei diagrammi UML permette di avere una chiara idea, a chiunque sia coinvolto nello sviluppo, di tutto l'insieme che costituisce il sistema. In questo modo, si potranno sfruttare al meglio anche le risorse hardware in termini di memoria ed efficienza, senza sprechi inutili o, al contrario, rischi di sottostima dei requisiti di sistema;

- grazie alla documentazione del linguaggio UML diviene ancora più facile effettuare eventuali *modifiche future* al codice. Questo, ancora, a tutto beneficio dei costi di mantenimento del sistema.

La comunicazione e l'interazione tra tutte le risorse umane che prendono parte allo sviluppo del sistema è molto più efficiente e diretta. Parlare la stessa "lingua" aiuta ad evitare rischi di incomprensioni e quindi sprechi di tempo.

L'applicazione dei modelli UML a un contesto come quello dell'automazione di edifici consente di massimizzare la produttività dello sviluppatore potendo fornire al cliente una rappresentazione comprensibile e favorisce la stesura delle specifiche tecniche da considerare per l'implementazione del sistema.

4.3 UML e metodologie per la progettazione del software

L'utilizzo di una metodologia rigorosa nel design e nella progettazione di un software (vedi [9] e [10]) aumenta la complessità del lavoro di documentazione da effettuare, ma fornisce indicazioni preziose sulle fasi da seguire e migliora la cognizione di causa sull'intero progetto.

Una sintesi delle operazioni da svolgersi per una corretta serializzazione del lavoro può essere riassunta nei seguenti passaggi:

Raccolta dei requisiti È la prima fase del lavoro congiunto con il cliente che, partendo dalle sue intenzioni iniziali e dai suoi desideri, ha lo scopo di produrre un documento informale, scritto in linguaggio naturale, che elenchi i requisiti e le specifiche richiesti. In pratica questo primo documento serve a circoscrivere i limiti del lavoro successivo. Deve essere breve e il più possibile accurato;

Stesura del Glossario In questa fase si deve definire la terminologia del progetto, identificando con precisione le entità (persone, ruoli, luoghi, oggetti materiali, eventi, strutturazioni, ecc.) coinvolte nel sistema del mondo reale (ovvero del dominio di business) che hanno importanza per il sistema informatico obiettivo del progetto. È importante identificare con precisione le entità allo scopo sia di definire meglio i loro scenari d'uso (*Use Case*), sia di individuare le classi entità (*Class Diagram* di analisi). Il risultato di questa fase è il *glossario*;

Stesura degli Use Case In questa fase devono essere individuati con precisione gli scenari di interazione fra il sistema e gli attori, ovvero le entità esterne al sistema con cui esso interagisce e comunica. I passi necessari in questa fase possono essere così suddivisi:

1. Definizione esatta del *boundary* o confine del sistema (entro sistemi particolarmente complessi questa fase può anche essere applicata a sottosistemi);
2. Identificazione e definizione degli *attori*, ossia delle entità ester-

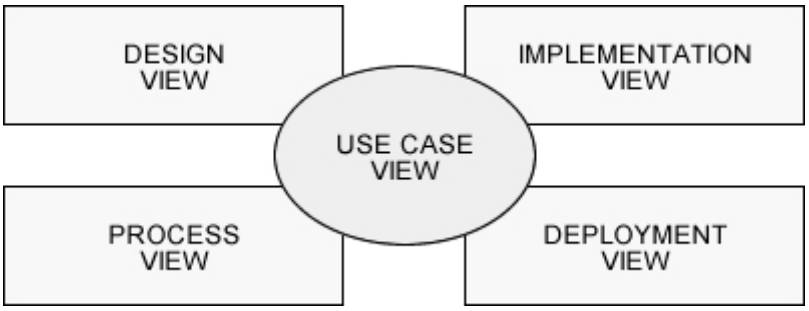


Figura 4.2: Struttura e interdipendenze delle fasi di progettazione e analisi di un software.

ne con cui il sistema (o i sottosistemi) oggetto dell'analisi interagiscono e comunicano;

3. Individuazione dei vari *scenari* di uso/interazione fra sistema ed attori, che corrisponderanno ai singoli casi d'uso, identificati da ellissi nel diagramma;
4. Definizione delle *interazioni* entro i singoli casi d'uso; tali interazioni, strutturate nella forma di richiesta dell'attore cui corrisponde una risposta del sistema (tenendo conto anche di eventuali comunicazioni asincrone o autonome quali ad esempio gli allarmi), andranno a costituire i campi descrizione dei singoli casi d'uso (operazione detta in gergo "srotolamento" dello use case);
5. Esame dei diagrammi così ottenuti e delle loro descrizioni per procedere alla raccolta a fattore comune di parti fra i singoli use case entro diagrammi, facendo uso delle relazioni *extends* ed *include* definibili tra i vari casi d'uso.

   Il passo 5 può essere iterato più volte; occorre tenere conto della granularità del problema e del grado di definizione e precisione che si vuole raggiungere. Inoltre si deve considerare che un singolo caso d'uso spesso dà origine ad una singola maschera (sia essa un menu testuale, una singola finestra in ambiente grafico o una pagina Web).

Il prodotto di questo passo è l'insieme completo degli use case inse-

riti entro uno o più diagrammi, ciascuno dei quali corredato da una adeguata descrizione, strutturata chiaramente in forma di request-response, e considerando sia il percorso principale di interazione (*basic course*) sia gli eventuali percorsi alternativi (*alternative course*). Il diagramma e le descrizioni devono essere ben strutturati, chiari ed esaurienti in quanto tutti i passi successivi si baseranno su di essi;

Stesura del Class Diagram di analisi In questa fase deve essere realizzato il diagramma delle classi di analisi. Esso deve indicare chiaramente tutte le classi entità, ossia le classi definibili come "proiezioni" nel dominio dell'applicazione software delle entità del problema in cui l'applicazione software andrà ad operare, più eventuali altre classi individuate nel corso dell'analisi e che siano di una certa rilevanza per i concetti funzionali che definiscono i requisiti del progetto. In pratica nel diagramma, che è l'equivalente da un punto di vista del ruolo (e l'evoluzione da un punto di vista storico e metodologico) del diagramma Entità-Relazione (ER) usato nelle metodologie di sviluppo più tradizionali, devono essere chiaramente indicate:

- tutte le classi *entità* che fanno parte del dominio del problema;
- gli *attribuiti* caratteristici di tali classi, eventualmente procedendo alla individuazione dei singoli attributi o dei gruppi che consentano una identificazione univoca delle istanze delle classi, ovvero dei singoli oggetti;
- le *associazioni* che intercorrono tra tali classi; queste associazioni (che corrispondono alle relazioni dei diagrammi ER) sono importanti perché in sede di implementazione del codice indicheranno anche la visibilità necessaria tra le classi, cioè quali altre classi (eventualmente appartenenti ad altri *package* o *namespace*) potranno essere viste da una certa classe e definendo quindi la loro interdipendenza;
- i versi di tali associazioni (ad esempio, se la classe magazzino deve conoscere la classe prodotto, non è sempre vero il contrario);

- le molteplicità di tali associazioni (es. uno-a-molti, molti-a-molti) e l'eventuale necessità di definire *classi di associazione*. Per esempio il concetto di proprieta' di un'auto, una volta rappresentato nel dominio delle classi, puo' costuire una classe di associazione fra l'auto e la persona che ricopre il ruolo di proprietario;
- eventuali rapporti di inclusione legati a tali associazioni e suddivisi fra aggregazione e composizione. Si ricordi che l'eliminazione di una composizione, indicata con il diamante nero, elimina anche tutti i suoi elementi componenti, mentre l'eliminazione di una aggregazione, indicata nella rappresentazione UML dei class diagram con il diamante bianco/nero, non elimina anche i componenti che comunque hanno un ambito di sopravvivenza indipendente;
- eventuali rapporti di ereditarietà fra le classi ottenuti applicando i principi di generalizzazione e specializzazione, ovvero *raccogliendo a fattor comune* attributi e metodi o aggiungendone di nuovi.

Il processo che conduce al diagramma finale è ovviamente iterativo e può dirsi stabilizzato quando tutte le relazioni (in senso ampio) fra le classi sono chiaramente individuate. Il Class Diagram di analisi è fondamentale per tutti i passi successivi;

Scelta architetturale e definizioni conseguenti La scelta architetturale è un passo fondamentale in quanto le fasi successive ne verranno pesantemente condizionate. Esistono comunque regole generali che ne aiutano lo svolgimento quali il pattern *Model-View-Controller* (MVC) ed il conseguente *approccio multicanale* alla realizzazione delle interfacce utenti.

Seguendo tale metodo si separa nettamente l'interfaccia utente vera e propria (*View*) che ha lo scopo di presentare i dati all'utente ed è ovviamente soggetta a vincoli, dal tipo di canale di comunicazione utilizzato (interfaccia a finestre grafiche, shell a caratteri, ecc.), dal

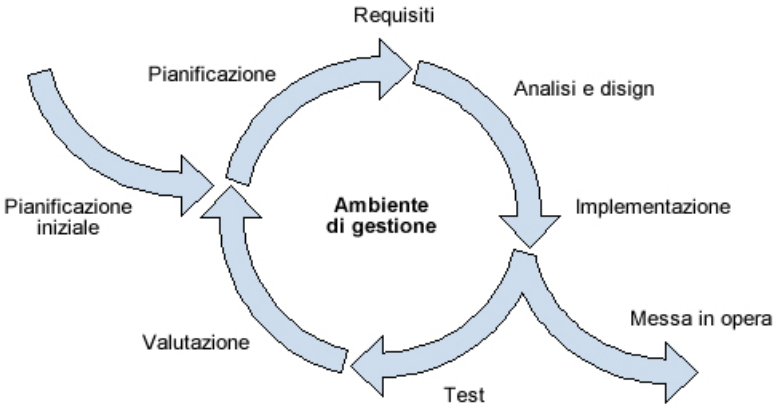


Figura 4.3: Fasi di sviluppo del software.

reattore agli eventi trasmessi dall'utente (*Controller*) che usa i metodi forniti dagli strati interni dell'applicazione (*Model* e relativi *Adapter*) per garantire all'utente i servizi associati alle richieste effettuate.

Grazie all'approccio multicanale, eventualmente corredato dall'uso di altri strati di Adapter, diviene possibile riutilizzare (almeno in buona parte) il Controller (ed ovviamente gli strati sottostanti) cambiando solo la View quando si cambia canale, passando, ad esempio, da una applicazione GUI ad una Web.

La scelta dell'architettura deve anche segnalare limiti e criticità nel sistema che sarà realizzato. L'output di questa fase sono documenti tecnici architetturali che saranno poi corredati da eventuali *Component Diagram* e *Deployment Diagram* solo al termine della fase di progetto vera e propria;

Definizione del Class Diagram di progetto In questa fase occorre definire chiaramente tutte le classi che fanno parte dell'applicazione software da implementare. Il *Class Diagram di Progetto* è l'elenco completo delle classi e di tutte le relazioni e su di esso si basa anche il dimensionamento della fase di sviluppo (ovvero la scrittura vera e propria del codice).

Il processo che permette di giungere al diagramma delle classi di progetto è necessariamente iterativo. Si parte dal diagramma delle classi di analisi e progressivamente vengono inserite tutte le classi di servizio che permettono al programma nel suo insieme di operare correttamente ed in modo efficiente. Le classi di servizio sono ovviamente fortemente dipendenti nella loro struttura dall'architettura scelta e da eventuali framework utilizzati nel progetto.

Se un diagramma di analisi ben fatto può essere spesso utilizzato con diverse tecnologie ad oggetti, ovvero essere punto di partenza per progetti analoghi realizzati su piattaforme diverse, un diagramma di progetto è chiaramente molto più influenzato dalla tecnologia usata. Il processo usa anche altri diagrammi UML:

- i diagrammi di interazione (*sequence diagram*, che evidenzia la sequenza temporale delle interazioni, e *collaboration diagram*, che chiarisce la dipendenza fra le classi) sono di importanza fondamentale sia per la definizione dei metodi che le classi offrono (e dei loro argomenti e valori di ritorno), sia per l'individuazione di eventuali colli di bottiglia che vengono risolti con l'inserimento di nuove classi o la cancellazione di quelle ridondanti. In teoria ad ogni use case corrisponde almeno un sequence o un collaboration diagram: infatti ogni corso di eventi individuato nell'analisi con gli use case dovrebbe produrre una precisa sequenza temporale di invocazione di metodi all'interno dell'insieme delle classi costituenti il sistema software. Non sempre è però indispensabile un'implementazione completa, specie se gli eventi presentano evidenti analogie e similitudini, nel qual caso basta apportare le opportune descrizioni di accompagnamento;
- i diagrammi di attività (*activity diagram*) derivano dagli Use Case e danno loro una sequenza temporale e logica. Inoltre possono aiutare molto nella definizione della mappa di navigazione tra le finestre, consentendo di definire completamente l'interfaccia utente di un applicativo ed eventualmente di realizzare i prototipi d'analisi;

- i diagrammi di stato (*statechart diagram*) sono anch'essi molto importanti per valutare l'evoluzione temporale delle singole classi (o meglio degli oggetti da esse istanziati) o di sottosistemi che esse vanno a costituire, aiutando ad individuare eventuali condizioni critiche o colli di bottiglia dell'applicazione.

L'obiettivo finale è comunque la realizzazione del Class Diagram di Progetto, completo di tutte le classi.

Spesso per motivi di chiarezza (specialmente in progetti grandi dove le classi sono molto numerose) il diagramma viene diviso in package o namespace, associazioni di classi corrispondenti ad unità funzionali, che indicano esternamente solo le reciproche relazioni. Ciascun package viene poi rappresentato completamente entro un diagramma di secondo livello. Quasi sempre questa suddivisione funzionale viene anche portata a livello implementativo servendosi delle aggregazioni tipiche dei linguaggi, come i package di Java o i namespace di C#. L'obiettivo deve essere sempre quello di avere un diagramma leggibile, che serve come mappa per lo sviluppo.

Da questo diagramma possono anche essere generati gli scheletri delle classi attraverso opportuni strumenti (ad esempio *Poseidon for UML*), oppure essere estrapolati i *Fogli di Specifica*, ossia i documenti che descrivono ciascuna classe con attributi, metodi, vincoli e controlli da implementare;

Definizione delle strutture di contorno Usando i diagrammi realizzati in precedenza si arriva a definire le parti implementative di contorno del progetto, che devono essere opportunamente documentate come segue:

- definizione della *base di dati*, attraverso un EER, eventualmente corredato dagli script di creazione delle tabelle e vincoli che genera la base dati nello specifico DBMS scelto;
- definizione dell'insieme dei singoli *componenti software* (packages, librerie statiche o dinamiche, archivi JAR, ecc.) che devo-

no essere prodotti, con l'indicazione delle loro interdipendenze, attraverso un opportuno Component Diagram;

- definizione della *distribuzione dei componenti* sulla o sulle piattaforme di produzione prescelte attraverso uno o più opportuni Deployment Diagram;
- stesura di opportuni *documenti Readme* ed altro che corredino il progetto e l'installazione; in particolare devono essere chiaramente indicati eventuali limiti e/o malfunzionamenti delle piattaforme software e hardware utilizzate;
- stesura dell'opportuno *manuale utente* dell'applicazione secondo i criteri stabiliti;
- definizione delle *scadenze* e pianificazione dell'esecuzione temporale del progetto in base ai dimensionamenti svolti e alle risorse a disposizione;
- definizione dei *test* e dei singoli casi di test;
- pianificazione del collaudo e dell'*entrata in produzione*;
- definizione della successiva fase di *manutenzione*.

4.4 Il linguaggio Java

Il linguaggio Java è stato sviluppato da Sun Microsystems nel 1991 come parte del progetto Green che si occupa di sviluppo di software per il controllo di dispositivi elettronici a larga diffusione. I ricercatori del progetto Green svilupparono un controllore chiamato Star7, una sorta di dispositivo di controllo remoto in grado di comunicare con altri esemplari dello stesso tipo. L'idea originale era di sviluppare il sistema operativo Star7 in C++, il potente linguaggio di programmazione orientato agli oggetti derivato dallo Standard C e ideato da Bjiarne Stroustrup.

Ma questo nuovo sistema ben presto non rispettò le aspettative, quindi Gosling fece partire un progetto totalmente nuovo sulla base di specifiche che aveva in mente. Nacque così, intorno alla metà del 1991, Oak (poi ribattezzato Java), un linguaggio di programmazione object oriented, platform-indipendent, robusto e sicuro, in grado di gestire meglio lo Star7.

Progettato tenendo presenti le applicazioni anziché lo sfruttamento completo delle risorse dei PC, Java doveva essere di piccole dimensioni, efficiente, affidabile e facilmente portabile verso un'ampia gamma di dispositivi hardware.

Gli elementi che hanno reso adatto il linguaggio Java per lo sviluppo dello Star7 si sono rivelati adatti anche per la programmazione del World Wide Web. Infatti esso garantisce dimensioni esigue per le proprie applicazioni, impedisce di scrivere programmi che possano accedere illimitatamente alle risorse dell'hardware e può essere eseguito su Windows, Macintosh, Linux e altre piattaforme senza modifiche rilevanti del codice sorgente. Inoltre con l'introduzione del Remote Method Invocation (RMI) è molto adatto per realizzare applicazioni distribuite.

Anche se Java ricevette parecchia attenzione da parte della comunità del Web, il linguaggio fu consacrato realmente solo dopo che Netscape diventò la prima società ad averne la licenza nell'agosto 1995.

L'obiettivo principale di Java è quello di avere un linguaggio indipendente dalla piattaforma (*platform indipendent*), in modo che funzioni su macchine e sistemi operativi differenti, e quindi anche su sistemi embedded. Java è un linguaggio orientato agli oggetti che eredita caratteristiche peculiari del C++ ma ne estende le potenzialità migliorando la produttività dello sviluppatore.

Gli ambienti di compilazione ed esecuzione sono differenti rispetto ad un linguaggio di programmazione "compilato". Infatti, dopo aver scritto un sorgente in altri linguaggi come C e C++, si procede alla fase di compilazione, nella quale si controlla la sintassi e si generano i file oggetto (*object*), i quali vengono poi collegati (fase di *linking*) a librerie statiche per la creazione del file eseguibile.

Se poi si volesse utilizzare lo stesso programma su sistemi operativi diversi sarebbe necessario ricompilare il codice sorgente, eventualmente apportando modifiche ad alcune parti dell'applicazione. In Java invece queste fasi sono differenziate e la compilazione non porta ad ottenere istruzioni codificate in un linguaggio macchina ma al *bytecode*, che è simile al codice macchina prodotto da altri linguaggi, ma non è specifico per alcun processore; in questo modo si aggiunge un livello di astrazione tra il codice

sorgente e il codice eseguibile, ottenendo l'indipendenza dalla piattaforma.

Dopo la fase di compilazione interviene il *Class Loader* che carica le classi in memoria dinamicamente (in modo da risparmiare spazio) ed è in grado di prelevare i dati dal file system locale oppure dalla rete.

In seguito al caricamento delle classi viene eseguito il *Verifier* che ha il compito di verificare la correttezza del bytecode prima che esso sia eseguito. Esso verifica che il bytecode utilizzi una quantità finita di risorse, che i salti avvengano ad indirizzi legali del bytecode e che le operazioni siano sempre applicate ad argomenti di tipo opportuno.

Al di sotto del Verifier si trova la *Java Virtual Machine* (specifica per ogni architettura hardware/software) che ha il compito di eseguire le istruzioni, cioè di trasformarle in azioni comprensibili al sistema operativo della macchina sul quale il programma è in esecuzione.

Per quanto riguarda le caratteristiche del linguaggio, l'exception handling di Java deriva direttamente (anche da un punto di vista sintattico) da quello del linguaggio C++. Tuttavia, il meccanismo di Java deve considerarsi forse più oneroso ma più sicuro, grazie alla cosiddetta regola dell'*handle or declare* (gestisci o dichiara), che in sostanza obbliga il programmatore a prevedere esplicite contromisure per ogni situazione anomala (prevedibile) all'interno del metodo, oppure a dichiarare un ulteriore rilancio dell'eccezione, rimandando al chiamante l'obbligo di gestirla.

Qualsiasi programma di un certo livello di complessità può incorrere, durante la propria esecuzione, in situazioni anomale che richiedano di essere trattate eseguendo azioni che differiscono da quello che sarebbe stato, altrimenti, il "flusso normale" del programma.

Ogni metodo di un programma Java dovrebbe avere un compito ben preciso da portare a termine. In presenza di anomalie o situazioni impreviste, è possibile che un metodo fallisca, cioè non sia in grado di portare a termine tale compito. Questa evenienza deve essere evidentemente segnalata al metodo chiamante il quale poi potrà, a seconda dei casi, prendere qualche contromisura che gli consenta di concludere il proprio compito nonostante il fallimento del metodo chiamato, oppure, se questo è impossibile, dichiarare a sua volta il proprio fallimento nei confronti del proprio chiamante (e così via).

Per segnalare il proprio fallimento, un metodo Java può sollevare una eccezione, che viene gestita dal chiamante attraverso un'apposita struttura di controllo, chiamata blocco try-catch. Con la clausola try viene controllato un blocco di codice all'interno del quale compare il metodo "a rischio": se il metodo ha successo si passa alle istruzioni successive al blocco try-catch, altrimenti l'eccezione (che può essere un qualsiasi oggetto che implementi l'interfaccia Throwable) viene catturata e gestita nel blocco catch.

Un'altra delle differenze tra Java e C++ è rappresentata dal meccanismo di ereditarietà: in Java, a differenza del C++, si può avere solo ereditarietà singola tra le classi. Questa scelta va a favore della robustezza, ma a discapito della potenza del linguaggio, tuttavia è possibile utilizzare l'ereditarietà multipla tra le interfacce.

Infine Java, diversamente da tutti gli altri linguaggi di programmazione, contiene già le primitive di multithreading per l'esecuzione concorrente di unità operative diverse che possono condividere dati e svolgere le loro attività in contemporanea. In pratica è possibile esplicitare nell'applicazione tutte quelle parti di codice che richiedono un'esecuzione sincronizzata senza ricorrere a chiamate alle primitive di multithreading del sistema operativo, consentendo una maggiore indipendenza dalla piattaforma utilizzata.

4.5 Caratteristiche dell'ambiente di sviluppo

Per quanto riguarda lo sviluppo del software risulta importante la scelta di un determinato ambiente, che può avvenire tenendo conto delle caratteristiche principali del progetto in fase di realizzazione oppure per criteri indipendenti da questo.

Il primo approccio è più flessibile e innovativo, perché viene ricercata e utilizzata la tecnologia più adatta al progetto (ad esempio per la disponibilità di particolari componenti o il supporto a determinati concetti di architettura), tuttavia può essere rischioso per il fatto che apprendere una nuova tecnologia porta a una dilatazione di tempi e di costi.

Per questo motivo si può scegliere di riutilizzare lo stesso ambiente di sviluppo perché già noto ai programmatori o già presente in azienda. La

Ambiente di sviluppo	Licenza	Costo
NetBeans (SUN)	Sun Public License	0 €
Java Studio Enterprise (SUN)	Commerciale	~ 600 €
Eclipse (IBM)	Eclipse Public License	0 €
WebSphere Studio (IBM)	Commerciale	~ 1000 €
JBuilder (Borland)	Commerciale	~ 500 €
JDeveloper (Oracle)	Commerciale	0 €

Tabella 4.1: Confronto tra ambienti di sviluppo.

produzione del software potrebbe però essere limitata da quella particolare tecnologia, e comunque questo approccio impedisce l'innovazione.

Indipendentemente dal progetto, la caratteristica più importante che deve possedere un ambiente di sviluppo è quella di fornire strumenti di qualità quali debugger, repository, editor di testi (con syntax highlighting, formattazione codice e correzione errori), metodi per lo sviluppo visuale e sistemi di gestione delle versioni, di essere espandibile, personalizzabile per esigenze particolari e in grado di integrarsi con il sistema operativo o con altri strumenti. Inoltre può incidere sulla scelta anche il costo, sia per il software in sé sia in termini di formazione (vedi tabella 4.1).

Capitolo 5

Lo SCADA di IntelliDomus

Verrà descritta l'applicazione dei concetti illustrati nei precedenti capitoli a un contesto pratico, la realizzazione di un sistema di Home Automation, IntelliDomus, ideato dalla ditta *CS Soluzioni* di Casalmaggiore (CR).

5.1 La struttura dell'applicativo

Il progetto IntelliDomus ha come obiettivo quello di fornire all'utente tutti i servizi necessari per la gestione delle utenze domestiche: riscaldamento, illuminazione, irrigazione, automatismi (finestre, tapparelle, cancelli) e allarmi.

Si vuole creare un sistema multipiattaforma, per cui si è scelto di utilizzare Java come linguaggio di sviluppo e di impiegare sistemi disponibili sia per ambiente Windows che Linux e possibilmente free, al fine di ridurre i costi relativi all'acquisto delle licenze. Infatti, dato che IntelliDomus ha come target la gestione di un ambiente domestico, si vuole cercare anche di mantenere un prezzo accessibile.

Il software di supervisione è costituito da diversi componenti, in quanto la struttura modulare consente una migliore organizzazione del sistema e la personalizzazione del software in base alle esigenze del cliente. Inoltre in futuro, mediante l'aggiunta di nuovi moduli software, risulterà più semplice l'ampliamento del progetto.

Come si può vedere dalla figura 5.1 il Set Top Box rappresenta il cuore

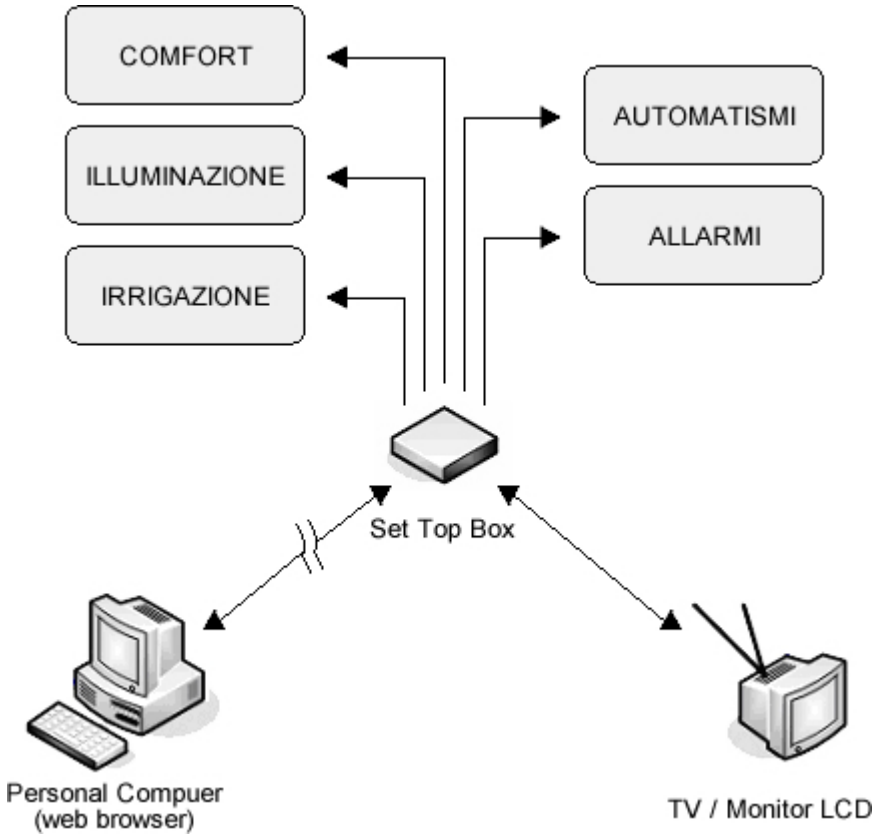


Figura 5.1: Il sistema IntelliDomus.

del sistema, in quanto contiene il software di supervisione che si occupa della comunicazione con il PLC e dell'interazione con l'utente, che può avvenire via web o attraverso un'interfaccia grafica apposita.

Il lavoro di controllo operativo sul PLC è invece gestito da un software realizzato da CS Soluzioni, scritto in Codesys, un linguaggio fornito dalla casa produttrice Wago che aderisce allo standard IEC-61131-3.

5.2 Analisi e progettazione

Il software di supervisione del sistema IntelliDomus si occupa dell'interazione tra l'utente e i sistemi per la gestione delle utenze domestiche. Analizzando in dettaglio tutte le aree vengono definite le seguenti specifiche di progetto:

Riscaldamento Il programma destinato a gestire il riscaldamento è organizzato in zone termiche (ovvero le parti dell'edificio che sono riscaldate in modo indipendente tra loro) con una programmazione settimanale, quindi ha necessità di un'ampia area di memoria riservata alle impostazioni delle fasce orarie di riscaldamento, per ogni zona termica che si intende controllare.

Si ipotizza di gestire un massimo di 16 zone differenti, utilizzando fasce di tempo della durata di 30 minuti[1], quindi in totale 2 (suddivisione in mezz'ore) $\times$ 24 (ore giornaliere) $\times$ 7 (giorni per settimana) $\times$ 16 (zone termiche) valori, cioè 5.376 setpoint da memorizzare.

Le temperature sono gestite e salvate in decimi di grado (la visualizzazione è invece in gradi con un cifra decimale), per compatibilità con le misurazioni date dai sensori di tipo PT-100. La trasmissione dei dati al controllore può essere effettuata all'inizio di ogni giorno con una procedura ad-hoc. Per ogni zona termica vi è una serie di dati necessari alla sua completa gestione:

- tipo di funzionamento (riscaldamento o raffrescamento);
- differenziale termico per la gestione dell'inseguimento del setpoint;
- ritardo di isteresi;
- setpoint manuale.

La necessità di avere un meccanismo di gestione e spostamento ulteriore dei dati di ricetta copre anche i requisiti inerenti altri dati di altri programmi che presentano una organizzazione con carattere temporale e in particolar modo giornaliero;

Illuminazione La parte di impianto elettrico relativa all'illuminazione che si intende gestire in automatico si ipotizza venga realizzata secondo le indicazioni di CS Soluzioni. Come evidenziato nei documenti specifici

[1] La scelta è dovuta al fatto che generalmente i termostati disponibili sul mercato utilizzano queste impostazioni, consentendo una ragionevole granularità di programmazione.

[11], i punti luce associati a lampadari, lampade fisse, faretti, ecc. che si intendono controllare anche in modalità automatica programmata, vanno connessi elettricamente a dei relè passo-passo.

Questi dispositivi, collocati nei box dell'impianto, possono essere azionati in base ad eventi e comandi provenienti dal controllore centrale oppure dalla semplice operazione manuale dell'utente sui pulsanti inseriti nelle normali placche a muro (pulsanti, non interruttori). Dalla possibile gestione automatizzata sono escluse abatjour o altre apparecchiature di illuminazione con interruttore diretto posto sul dispositivo stesso.

Il sistema di illuminazione così approntato offre diversi vantaggi: al singolo pulsante è possibile cambiare facilmente "ruolo": si ha l'opportunità di azionare un punto luce diverso da quello inizialmente collegato spostando semplicemente i fili nel box, senza bisogno di lavorare su tubazioni o altro, inoltre il controllore centrale può agire contemporaneamente su più punti luce così come è agevole associare più pulsanti al medesimo punto luce. Un altro vantaggio è che la tensione tra i capi dei fili collegati ai pulsanti è a $24V$ invece di $220V$, quindi con l'impiego di fili di minor diametro si ottiene anche un risparmio economico.

Dal punto di vista funzionale il programma che gestisce l'illuminazione ha caratteristiche meno complesse rispetto a quello di riscaldamento o irrigazione. Non sono previsti calendari settimanali o una suddivisione in fasce orarie in quanto in questo caso non ha molto senso fornire tali funzionalità, è invece contemplata la possibilità, a livello di gruppo di punti luce, di programmare due accensioni e due spegnimenti giornalieri per tutte le luci configurate per essere comandate in modo automatico.

Il numero massimo di gruppi gestibile è 9, identificati da variabili il cui contenuto viene deciso in fase di inizializzazione del sistema . Per ogni gruppo sono presenti al massimo 16 punti luce per un totale di 144 variabili interessate;

Irrigazione La parte di impianto elettrico relativa all'irrigazione è in grado di gestire un massimo di 9 settori differenti che corrispondono ad altrettante elettrovalvole programmabili indipendentemente tra loro, su base giornaliero-settimanale, per un totale di 8 eventi di irrigazione al giorno.

Le funzionalità e l'interfaccia utente di questa sezione sono molto simili a quelle che si ritrovano per il controllo delle zone termiche; è consentito utilizzare la singola valvola di settore in manuale o in automatico ed è disponibile una funzione di copia delle impostazioni tra settori e giorni differenti;

Automatismi La parte di impianto elettrico relativa agli automatismi è suddivisa in sei sottocategorie:

- finestre;
- tende;
- serrande;
- tapparelle;
- basculanti;
- cancelli.

Le prime quattro entità vengono gestite nel medesimo modo, offrendo la possibilità di considerare un numero massimo di 20 applicazioni. Le ultime due vengono considerate come entità simili e quindi sono gestite in modo molto analogo: in questo contesto è possibile considerare fino a 3 applicazioni;

Allarmi La parte di impianto elettrico relativa agli allarmi non può essere definita a priori ma analizzando le reali necessità dell'utente. In linea del tutto generale si è pensato di considerare comunque un numero massimo di zone sensibili pari a 30.

In figura 5.2 si può vedere lo use case diagram relativo alla gestione delle zone termiche (i diagrammi relativi alle altre utenze hanno una struttura più semplificata ma simile).

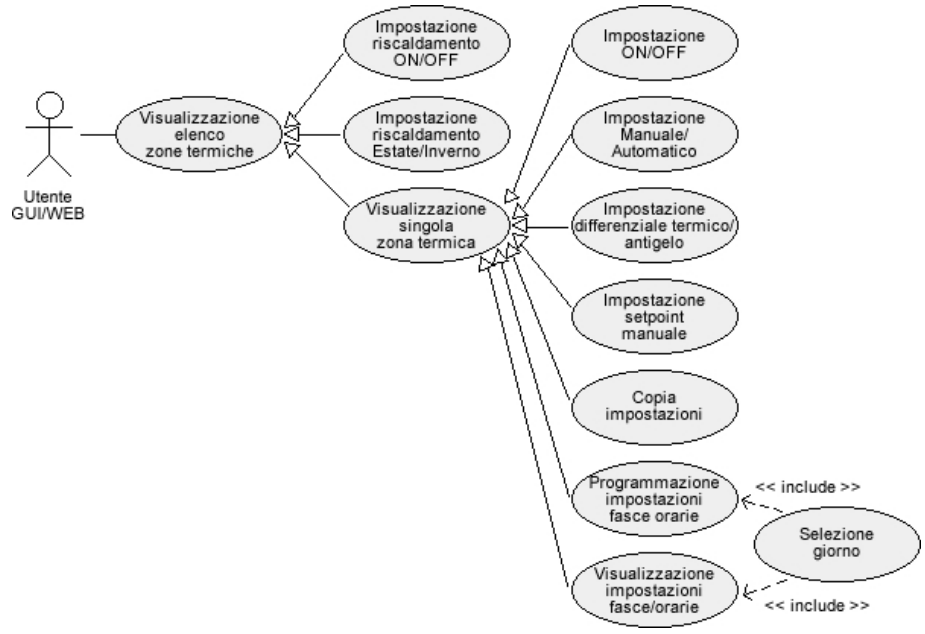


Figura 5.2: Use case diagram relativo alla funzionalità di gestione delle zone termiche.

L'utente, che nel caso di un sistema di Home Automation è un abitante della casa, deve essere in grado di gestire e controllare lo stato di tutte le utenze descritte. Sono stati scelti due sistemi di interazione: il primo consiste in un software con una tradizionale interfaccia grafica contenente tutti i controlli necessari per la gestione del sistema, installata sul STB a cui sono collegati un televisore o un monitor LCD e dispositivi di input quali mouse e tastiera (oppure un telecomando con funzionalità avanzate). Il software viene caricato all'avvio del sistema operativo, presentando subito la schermata principale.

Si vuole inoltre dare la possibilità di accedere al sistema via web, quindi il STB è collegato ad internet per essere in grado accettare richieste remote (e quindi è necessario anche un web server, in questo caso è stato adottato Apache con il servlet container Tomcat). Per questo motivo è stata creata un'applicazione web, sviluppata in Java utilizzando servlet e Java Server Pages, avente un'interfaccia grafica del tutto simile a quella della versione GUI, che comunica con il sistema utilizzando per default la porta 8080.

Sul STB è inoltre necessario un DBMS (si è scelto di utilizzare MySQL versione 4.1), che garantisce la persistenza e la storicizzazione dei dati che verranno impostati o letti dal client e trasferiti da/verso il PLC per mezzo di un software di sincronizzazione.

In un sistema di Home Automation si suppone che il numero di utenti connessi nello stesso momento sia piuttosto esiguo, quindi per il database non si hanno problemi di gestione di elevate contemporaneità. Perciò, considerando anche le capacità del PLC, si è stabilito di prevedere un massimo di 5 sessioni di connessione simultanee.

Il componente più importante è rappresentato dal software di sincronizzazione, in quanto si occupa dell'allineamento dei dati tra database e memoria del PLC, consentendo l'applicazione dei comandi e delle impostazioni inviate dal client e il recupero delle informazioni relative allo stato delle varie utenze gestite.

Le richieste inviate sia dall'interfaccia Web che dalla GUI vengono interpretate e gestite dal software HMI, che si occupa della restituzione dei dati all'utente, dell'aggiornamento del database con i valori impostati e di segnalare al software di sincronizzazione la navigazione dell'utente (e di conseguenza le eventuali variabili modificate). Per il software HMI, sviluppato in Java, è stata progettata e sviluppata un'organizzazione a package stratificata, come verrà illustrato nel paragrafo 5.6.

Un aspetto importante è legato alla comunicazione tra i componenti che costituiscono il sistema: software HMI e sincronizzatore comunicano via socket, utilizzando un semplice protocollo che segnala al software di sincronizzazione le informazioni necessarie per l'allineamento dei dati, come illustrato nel paragrafo 5.5.

Per il collegamento tra STB e PLC viene invece utilizzato il protocollo Modbus su TCP/IP, la comunicazione avviene mediante il controller Ethernet situato sul PLC Wago.

Il software di sincronizzazione, sviluppato in VB .NET, utilizza apposite librerie messe a disposizione dalla casa produttrice Wago (disponibili solo per Microsoft Windows) che sfruttano la libreria di sistema Windows Socket 2.0 per poter dialogare con il controller.

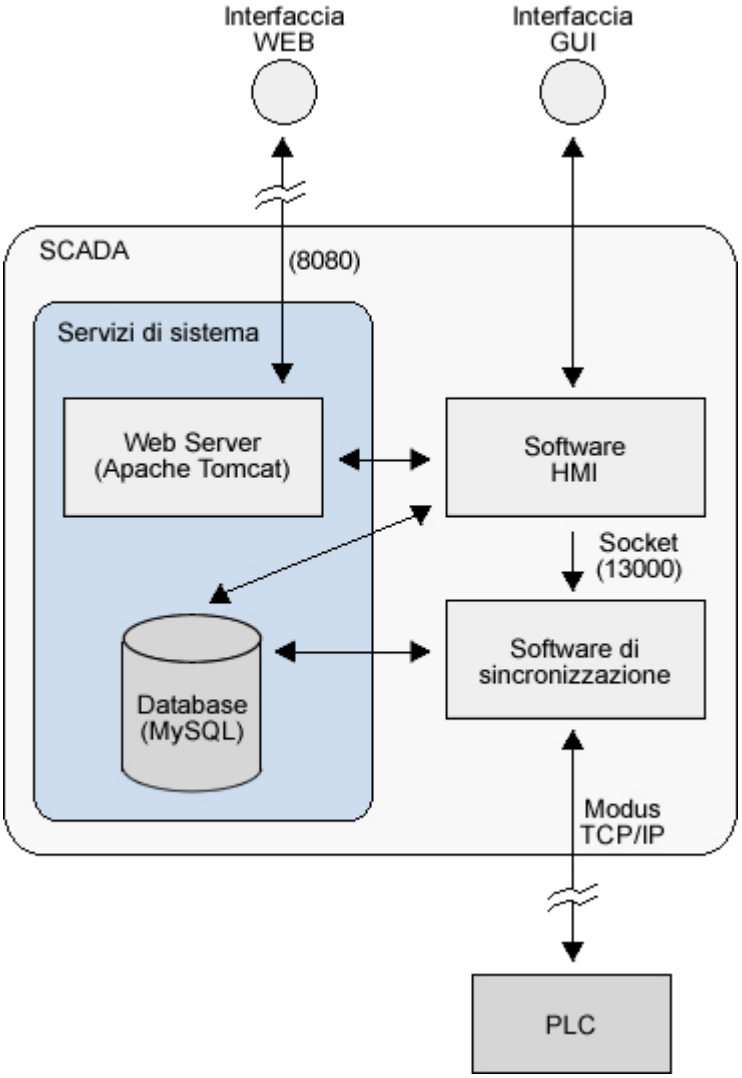


Figura 5.3: Interazione tra i componenti dello SCADA.

Per consentire la portabilità su sistemi non Microsoft è in corso lo sviluppo del porting in Java del software di sincronizzazione, utilizzando *Java Modbus Library* (jamod), un'implementazione free e open source del protocollo Modbus realizzata in Java, che permette di realizzare applicazioni master/slave che comunicano tramite connessione seriale o di rete (protocolli TCP/IP o UDP/IP).

Uno schema dei componenti di sistema e dell'interazione tra loro è presentato in figura 5.3.

Per quanto riguarda la configurazione del sistema è stato installato come sistema operativo Microsoft Windows XP Home Edition, ma al fine di ridurre i costi per l'acquisto delle licenze si prevede la migrazione a Linux. I requisiti software sono i seguenti:

- *J2SE Runtime Environment* (JRE) versione 5.0 Update 6, che contiene la *Java Virtual Machine* (JVM), le librerie e il software necessari per l'esecuzione di programmi scritti in Java;
- web server *Apache Tomcat* versione 5.5, per la gestione delle richieste e dell'interfaccia web del sistema (pagine JSP e servlet);
- DBMS *MySQL Server* versione 4.1;
- driver *MySQL Connector/J* versione 3.1.11, che si occupa della conversione delle chiamate JDBC (*Java Database Connectivity*) nel protocollo di rete utilizzato da MySQL;
- *Microsoft .NET framework*, che mette a disposizione librerie di classi e il motore di esecuzione per applicativi .NET;
- driver *MySQLDriverCS* versione 3.0.18, per la connessione a database MySQL all'interno di applicazioni .NET;
- libreria Modbus *MBT.dll*, per la comunicazione tra software di sincronizzazione e PLC, che viene messa a disposizione dalla casa produttrice Wago.

Gli ultimi tre componenti non saranno più necessari quando il software di sincronizzazione verrà riscritto in Java, sarà invece richiesta la libreria jamod citata precedentemente.

Sfruttando sistemi free si può vedere come i costi sostenuti da parte dell'utente per l'acquisto delle licenze vengano azzerati. Inoltre anche per lo sviluppo del software Java è stato utilizzato NetBeans nella versione 5.0, un ambiente di sviluppo free e open source, distribuito sotto licenza SPL (*Sun Public License*), mentre per la progettazione del sistema è stato scelto Poseidon for UML 4.1, gratuito nella versione CE (*Community Edition*).

Ulteriori informazioni e riferimenti sul software utilizzato sono reperibili in appendice.

5.3 La base di dati

Il database ha il compito di fornire la persistenza e la storicizzazione dei dati per quanto riguarda i seguenti aspetti:

- mappatura della memoria del PLC (mediante variabili sincrone e asincrone);
- parametri di configurazione, stato e messaggi di errore generati dal PLC;
- gestione utenze (riscaldamento, illuminazione, irrigazione, automatismi, allarmi);
- gestione utenti e navigazione;
- gestione transazioni e log;
- localizzazione (immagini, label e messaggi di errore).

Il database deve rappresentare una copia esatta della memoria del PLC, quindi occorre prima di tutto analizzare la struttura delle celle di memoria del controllore, che in generale sono costituite come in figura 5.4.

Per ogni indirizzo di memoria è necessario sapere se esso verrà destinato alle sole azioni di lettura e scrittura oppure ad entrambe. Inoltre bisogna definire il tipo di valore che può essere scritto e che deve essere letto, ovvero se si tratta di un singolo bit, di una word oppure di una double-word. Si supponga di avere due aree di memoria: una riferita ad

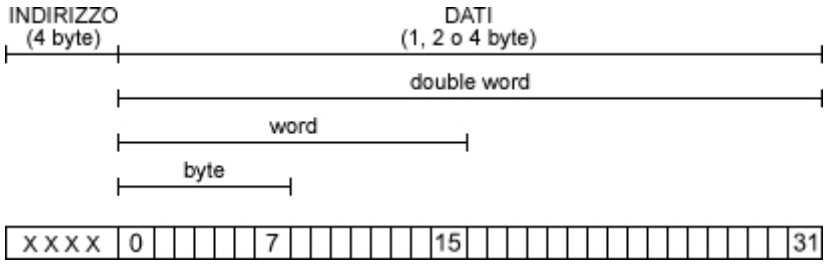


Figura 5.4: Il contenuto di un registro di memoria del PLC

	Sonda di temperatura	Sensore di presenza
Tipo valore	analogico	digitale
Tipo variabile	word	bit
Valori ammessi	0..20	0-1
Indirizzo di memoria	XXXX	XXXX.Y
Permessi	Read	Read

Tabella 5.1: Comparazione di due aree di memoria.

un valore letto da una sonda di temperatura, mentre l'altra riferita ad un sensore di presenza. Analizzando le caratteristiche di entrambe è possibile individuare caratteristiche comuni e peculiarità in contesti differenti.

Quando viene letto un valore di tipo boolean occorre sempre specificare il bit di riferimento, infatti nel caso del sensore di presenza l'indice del bit è identificato dalla variabile Y (vedi figura 5.1). Si osservi invece che per leggere il valore della sonda di temperatura occorre eseguire una conversione da binario a decimale, infatti avendo a disposizione una word a 16 bit sarà possibile ottenere un numero decimale da 0 a 65535.

È inoltre possibile distinguere due tipologie di variabili che intervengono in questo contesto: le variabili dette *asincrone*, il cui aggiornamento dipende dalle effettive necessità dell'utente (ad esempio l'accensione del riscaldamento), e le variabili *sincrone*, che al contrario sono completamente indipendenti dalle azioni compiute dell'utente (un esempio di variabile sincrona è lo stato degli allarmi, che deve essere controllato periodicamente ed in modo del tutto trasparente all'utente).

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
var_name	varchar(70)	Nome della variabile
word_adr	char(16)	Indirizzo di memoria nel PLC
bit_index	int(2)	Indice del bit da leggere se si tratta di valore boolean
type_code	int(2)	Tipo della variabile [0=bit, 1=byte, 2=word, 4=Dword, 8=Long]
var_value	bigint	Valore della variabile
var_definition	char(2)	Definizione dell'area di memoria [R: read, W:write, RW:read/write]
ID_associated_var1	int(10)	FK Riferimento all'ID della prima variabile da aggiornare
plc_code	int(2)	Riferimento al PLC

Tabella 5.2: Definizione tabella transfert_async.

Queste ultime solitamente saranno agganciate ad un timer che periodicamente segnala al sincronizzatore dati di effettuare una lettura nell'area di memoria del controllore per verificarne lo stato. A questa categoria appartengono anche tutti i valori letti dalle sonde di temperatura, che richiedono un aggiornamento costante in modo da fornire all'utente una visione reale ed istantanea della situazione domestica.

Per questo si è pensato di costruire tabelle separate, così da poter gestire in modo indipendente le categorie di dati. La tabella delle variabili asincrone, denominata `transfert_async`, è descritta in tabella 5.2, la 5.3 rappresenta invece l'entità `transfert_sync`.

Dato che il database (e in particolare queste due tabelle) rappresenta il punto per lo scambio di dati tra PLC e client si è dovuta prestare attenzione ai tipi di dato utilizzati nei vari componenti, identificando le corrispondenze tra uno e l'altro per evitare problemi di conversione. In tabella 5.4 sono visualizzati i tipi di dato utilizzati all'interno del database, con i rispettivi tipi per Java.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
var_name	varchar(70)	Nome della variabile
word_adr	char(16)	Indirizzo di memoria nel PLC
bit_index	int(2)	Indice del bit da leggere se si tratta di valore boolean
type_code	int(2)	Tipo della variabile [0=bit, 1=byte, 2=word, 4=Dword, 8=Long]
var_value	bigint	Valore della variabile
var_definition	char(2)	Definizione dell'area di memoria [R: read, W:write, RW:read/write]
is_alarm	bool	Indica se la variabile deve essere considerata come un allarme
ID_alarm_description	int(10)	FK della descrizione dell'allarme

Tabella 5.3: Definizione tabella transfert_sync.

MySQL	Java
bool	boolean
tinyint	byte
smallint	short
int	int
bigint	long
char(n), varchar(n)	String

Tabella 5.4: Corrispondenze tra tipi di dato MySQL/Java.

Per quanto riguarda il PLC i dati sono memorizzati in forma binaria all'interno di registri Word (2 byte), quindi è il software di sincronizzazione che si occupa delle conversioni da binario a decimale e viceversa. I possibili tipi di dato sono due: intero e booleano: per i primi basta una semplice conversione, mentre per i secondi si dovrà conoscere la posizione del bit nel registro in cui è memorizzato il valore.

In ogni caso all'interno del database sono stati utilizzati solo tipi semplici (int, bool, char e varchar), disponibili per tutti i principali DBMS in commercio, in modo da facilitare un'eventuale migrazione da MySQL a un altro DBMS (ad esempio PostgreSQL).

Per alcuni dati riguardanti il PLC (parametri di configurazione come l'IP, variabili di stato e messaggi di errore) è stata creata una tabella apposita, contenente coppie di attributi [chiave, valore] utilizzati dal software di sincronizzazione.

Per quanto riguarda la gestione delle singole utenze sono state create tabelle corrispondenti ad ognuna delle entità coinvolte nel sistema. I campi definiti come `int(10)` sono chiavi esterne che si riferiscono ai rispettivi record presenti nella tabella `transfert_async`.

Per la parte legata al riscaldamento sono state create le tabelle 5.6, 5.7 e 5.8, analogamente per l'illuminazione si sono definite le tabelle 5.9, 5.10 e 5.11, mentre per l'irrigazione vedere le tabelle 5.12, 5.13 e 5.14.

Infine per le funzionalità relative agli automatismi sono state create le tabelle 5.15 per le basculanti, 5.16 per i cancelli, 5.17 e 5.18 relative a tapparelle, tende e finestre.

Per quanto riguarda la gestione degli utenti, occorre prevedere delle entità che permettano di tenere traccia della navigazione utente tra le varie finestre che compongono l'interfaccia grafica.

Vengono quindi definite tre tabelle, la cui struttura si può vedere nelle tabelle 5.19, 5.20 e 5.21.

I campi denominati `ui_level_X`, con `X` = 1..6 sono tutti i possibili livelli in cui l'utente può accedere, mentre i campi `ui_param_X`, con `X` = 1..4 servono a specificare i parametri necessari al livello (ad esempio se ci si trova nella schermata di una zona termica verrà inserito l'ID corrispondente).

Il meccanismo di navigazione per livelli verrà illustrato nel paragrafo 5.6.

Un altro aspetto rilevante riguarda le transazioni e i log, mediante i quali si gestiscono gli eventuali errori di comunicazione tra software di sincronizzazione e PLC. Per tenere traccia dei valori necessari al recupero dei dati è stata definita la tabella `transfert_log` (5.22), che memorizza le infomazioni relative a tutte le variabili modificate: nel paragrafo 5.5 viene spiegato il funzionamento del modulo che si occupa di questa funzionalità.

Infine è stata creata una tabella necessaria per la gestione della localizzazione del software in diverse lingue (italiano, inglese, francese, tedesco e spagnolo): essa contiene tutte le immagini e le label che vengono utilizzate nell'interfaccia grafica del client, oltre ai possibili messaggi di errore che si possono verificare.

Ogni record è identificato da un nome univoco, che viene utilizzato per il caricamento di tutte queste variabili al momento dell'avvio dell'HMI, in cui viene definita la lingua da utilizzare. La struttura è visibile in tabella 5.23.

Il database viene popolato inizialmente mediante un software scritto appositamente, che si occupa dell'inserimento di tutti i dati relativi alle utenze (nome, descrizione, indirizzo di memoria e altri parametri), come ad esempio le zone termiche, i settori di irrigazione, i punti luce, ecc.. Perciò le operazioni effettivamente eseguite dal software HMI coinvolgono soltanto l'aggiornamento dei valori per le variabili selezionate.

Nome	Tipo	Descrizione
key_name	varchar(30)	Chiave
key_value	varchar(250)	Valore

Tabella 5.5: Definizione tabella ini_element.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
onoff_status	int(10)	Termoregolazione in funzione
winter_status	int(10)	Inverno
summer_status	int(10)	Estate
external_temperature	int(10)	Temperatura rilevata dalla sonda esterna

Tabella 5.6: Definizione tabella global_termczone.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
termiczone_idx	int(3)	Indice della zona termica entro il programma, con posizione entro la finestra
name	varchar(30)	Nome della zona termica
is_present	bool	Flag che indica se la zona è presente
man_status	int(10)	Status manuale entro la zona termica
man_status_write	int(10)	Status manuale entro la zona termica, riservata per la scrittura
auto_status	int(10)	Status automatico entro la zona termica
auto_status_write	int(10)	Status automatico entro la zona termica, riservata per la scrittura
current_temperature	int(10)	Temperatura effettiva presente entro la zona termica
current_auto_setpoint	int(10)	Setpoint attivo al momento presente
manual_setpoint	int(10)	Setpoint manuale
iceprevention_setpoint	int(10)	Setpoint antigelo
offset_delay	int(10)	Ritardo di applicazione dei cambiamenti
temperature_offset	int(10)	Differenziale per l'isteresi

Tabella 5.7: Definizione tabella local_termiczone.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
ID_termiczone	int(10)	FK zona termica
ID_weekday	int(10)	FK giorno della settimana
list_idx	int(3)	Posizione entro la lista (0-47)
auto_setpoint	int(10)	Setpoint temperatura

Tabella 5.8: Definizione tabella timezone_termiczone.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
group_idx	int(3)	Indice del gruppo per la visualizzazione sulla finestra
name	varchar(30)	Nome del gruppo
description	varchar(250)	Descrizione lunga
is_present	bool	Flag che indica se il gruppo è presente

Tabella 5.9: Definizione tabella group_timedlight.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
ID_group_timedlight	int(10)	FK del gruppo luce di appartenenza
name	varchar(20)	Nome del punto luce
is_present	bool	Flag che indica se il punto luce è presente
manauto_status	int(10)	Status automatico/manuale del singolo punto luce
switch_status	int(10)	Status dell'interruttore [NetMaster]
onoff_status	int(10)	Status effettivo impostato in uscita
feedback_status	int(10)	Status della luce (feedback) [NetMaster]
feedback_enabled	int(10)	Indica se il feedback è abilitato [NetMaster]
presence_sensor_existence	int(10)	Indica se esiste il sensore di presenza per questo punto luce
presence_sensor_enabled	int(10)	Indica se il sensore di presenza è abilitato
presence_sensor_status	int(10)	Status segnalato dal sensore
switchoff_delay	int(10)	Ritardo espresso in secondi per lo spegnimento quando il sensore non rileva più presenze

Tabella 5.10: Definizione tabella local_timedlight.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
ID_group	int(10)	FK gruppo luci
ID_weekday	int(10)	FK giorno della settimana
list_idx	int(3)	Posizione entro la lista
on_hour	int(10)	Ora di inizio
on_minute	int(10)	Minuto di inizio
off_hour	int(10)	Ora di fine
off_minute	int(10)	Minuto di fine

Tabella 5.11: Definizione tabella timezone_timedlight.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
water_present	int(10)	Rilevamento presenza acqua (pressostato)
pump_switch_status	int(10)	Stato dell'interruttore della pompa
pump_onoff_status	int(10)	Stato effettivo per la pompa impostato in uscita
pump_actual_status	int(10)	Status reale della pompa (da feedback)
valve_switch_status	int(10)	Stato dell'interruttore della valvola
valve_onoff_status	int(10)	Stato effettivo per la valvola impostato in uscita
valve_actual_status	int(10)	Status reale della valovla (da feedback)

Tabella 5.12: Definizione tabella global_irrigation.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
sector_idx	int(3)	Indice del settore irriguo per la visualizzazione sulla finestra
name	varchar(30)	Nome del settore
description	varchar(250)	Descrizione lunga
is_present	bool	Flag che indica se il settore irriguo è presente
auto_status	int(10)	Status automatico del singolo settore
auto_status_write	int(10)	Status automatico del singolo settore, riservata per la scrittura
man_status	int(10)	Status manuale del singolo settore
man_status_write	int(10)	Status manuale del singolo settore, riservata per la scrittura
feedback_status	int(10)	Status della valvola (feedback) [NetMaster]
switch_status	int(10)	Status dell'interruttore [NetMaster]
onoff_status	int(10)	Status per la valvola impostato in uscita (attuazione)

Tabella 5.13: Definizione tabella local_irrigation.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
ID_sector	int(10)	FK settore irriguo di appartenenza
ID_weekday	int(10)	FK giorno della settimana
list_idx	int(3)	Posizione entro la lista
on_hour	int(10)	Ora di inizio
on_minute	int(10)	Minuto di inizio
off_hour	int(10)	Ora di fine
off_minute	int(10)	Minuto di fine

Tabella 5.14: Definizione tabella timezone_irrigation.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
list_idx	int(3)	Indice della basculante per la visualizzazione sulla finestra
name	varchar(30)	Nome della basculante
description	varchar(250)	Descrizione lunga
open_hmi	int(10)	Apertura basculante HMI
close_hmi	int(10)	Chiusura basculante HMI
block_hmi	int(10)	Arresto basculante HMI
alert_hmi	int(10)	Emergenza basculante HMI
open_status	int(10)	Apertura in corso
close_status	int(10)	Chiusura in corso
coast_pneumatic_status	int(10)	Segnale temporizzato costa pneumatica
micro_status	int(10)	Segnale micro di sicurezza (basculante totalmente aperta)
balancing_status	int(10)	Stato basculante

Tabella 5.15: Definizione tabella local_balancing.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
list_idx	int(3)	Indice del cancello per la visualizzazione sulla finestra
name	varchar(30)	Nome del cancello
description	varchar(250)	Descrizione lunga
open_hmi	int(10)	Apertura cancello HMI
close_hmi	int(10)	Chiusura cancello HMI
block_hmi	int(10)	Arresto cancello HMI
alert_hmi	int(10)	Emergenza cancello HMI
open_status	int(10)	Apertura in corso
close_status	int(10)	Chiusura in corso
internal_sensor_status	int(10)	Segnale fotocellula interna sicurezza chiusura cancello
external_sensor_status	int(10)	Segnale fotocellula esterna (pilastro) sicurezza chiusura cancello
micro_status	int(10)	Segnale micro stato cancello
gate_status	int(10)	Stato cancello

Tabella 5.16: Definizione tabella local_gate.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
list_idx	int(3)	Indice della tapparella per la visualizzazione sulla finestra
name	varchar(30)	Nome della tapparella
description	varchar(250)	Descrizione lunga
type_code	int(2)	Indica il tipo di automatismo (0: Tapparella; 1: Tenda; 2: Finestra; 3: Serranda)
is_present	bool	Flag che indica se la tapparella è presente
manauto_status	int(10)	Status automatico/manuale della singola tapparella
up_feedback_status	int(10)	Status fine corsa su della tapparella (feedback)
down_feedback_status	int(10)	Status fine corsa giù della tapparella (feedback)
up_used	int(10)	Utilizzo il fine corsa su
down_used	int(10)	Utilizzo fine corsa giù
up_switch_status	int(10)	Status dell'interruttore (su/apri)
down_switch_status	int(10)	Status dell'interruttore (giù/chiudi)
up_onoff_status	int(10)	Status effettivo impostato in uscita su
down_onoff_status	int(10)	Status effettivo impostato in uscita giù
meteo_alarm	int(10)	Allarme meteorologico (digitale)
meteo_alarm_is_used	int(10)	Utilizzo allarme meteo

Tabella 5.17: Definizione tabella local_shutter.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
ID_shutter	int(10)	FK tapparella di appartenenza
ID_weekday	int(10)	FK giorno della settimana
list_idx	int(3)	Posizione entro la lista
on_hour	int(10)	Ora di inizio
on_minute	int(10)	Minuto di inizio
off_hour	int(10)	Ora di fine
off_minute	int(10)	Minuto di fine

Tabella 5.18: Definizione tabella timezone_shutter.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
username	varchar(20)	Username
password	varchar(10)	Password
name	varchar(30)	Nome dell'utente
ID_group	int(10)	Gruppo di appartenenza

Tabella 5.19: Definizione tabella user_user.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
ID_user	int(10)	FK utente
login_time	char(20)	Data e ora di accesso al sistema
logout_time	char(20)	Data e ora di uscita dal sistema
ip	char(15)	Indirizzo IP dell'utente

Tabella 5.20: Definizione tabella user_session.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
ID_session	int(10)	FK sessione
ui_level_1	int(2)	Primo livello della navigazione utente
ui_level_2	int(2)	Secondo livello della navigazione utente
ui_level_3	int(2)	Terzo livello della navigazione utente
ui_level_4	int(2)	Quarto livello della navigazione utente
ui_level_5	int(2)	Quinto livello della navigazione utente
ui_param_1	int(10)	Primo parametro disponibile
ui_param_2	int(10)	Secondo parametro disponibile
ui_param_3	int(10)	Terzo parametro disponibile
ui_param_4	int(10)	Quarto parametro disponibile
is_validate	bool	Indica se il software di sincronizzazione ha aggiornato i dati per la lettura
is_committed	bool	Indica se è stato validata la transazione
SCN	bigint	Identificativo della transazione

Tabella 5.21: Definizione tabella user_currentpage.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
ID_transfert_async	int(10)	FK relativa alla variabile presente nella tabella delle variabili d'appoggio
ID_currentpage	int(10)	FK relativa alla navigazione corrente
last_update	char(20)	Giorno e l'ora dell'ultimo aggiornamento
last_upload	char(20)	Data della memorizzazione definitiva nel PLC
old_value	bigint	Valore della variabile prima della modifica da parte dell'utente
new_value	bigint	Valore della variabile prima della modifica da parte dell'utente

Tabella 5.22: Definizione tabella transfert_log.

Nome	Tipo	Descrizione
ID	int(10)	Chiave primaria
image_name	varchar(30)	Chiave per la ricerca da codice (Nome univoco utilizzato all'interno del software)
image_fs_path	varchar(50)	Path dell'immagine nella versione GUI
image_web_path	varchar(50)	Path dell'immagine nella versione Web
image_file_name_ita	varchar(50)	Nome del file nella versione italiano
image_file_name_eng	varchar(50)	Nome del file nella versione inglese
image_file_name_fra	varchar(50)	Nome del file nella versione francese
image_file_name_esp	varchar(50)	Nome del file nella versione spagnolo
image_file_name_deu	varchar(50)	Nome del file nella versione tedesco
is_label	bool	Flag che indica se l'elemento è una label oppure una immagine

Tabella 5.23: Definizione tabella ui_element_description.

5.4 Il PLC Wago

Il controllore utilizzato per IntelliDomus è il WAGO-I/O-SYSTEM 750-841. Si tratta del più piccolo sistema modulare di ingressi e di uscite indipendente dal bus di campo concepito per l'automazione decentralizzata. Permette di guadagnare costi e spazio durante il montaggio del nodo del bus di campo con la libera combinazione di moduli I/O digitali, analogici e complessi.

È basato su una CPU a 32 bit in grado di lavorare in multitasking ed è un componente modulare e flessibile, grazie alla combinazione di ingressi e di uscite digitali/analogici e di funzioni specifiche con diversi potenziali, potenze e segnali sullo stesso nodo bus di campo.

Possiede inoltre un controller Ethernet, quindi, sfruttando uno dei protocolli che mette a disposizione (ad esempio Modbus su TCP/IP), consente di inviare e ricevere comandi o di prelevare e impostare i dati nell'area di memoria.

Oltre ai vantaggi citati precedentemente, il controllore Wago introduce un plug-in innovativo all'interno della CPU, che ha portato alla scelta di utilizzare questo PLC: si tratta di un web-server integrato che offre un supporto diretto alla memoria del controllore e altre funzionalità (come il recupero della data di sistema) attraverso il semplice uso di pagine HTML e SSI.

All'interno del web-server vengono messe a disposizione alcune pagine HTML per il controllo remoto del sistema, attraverso le quali è possibile impostare alcuni parametri, come ad esempio l'abilitazione dei vari protocolli e l'impostazione dell'indirizzo IP se non è stato abilitato un server DHCP nella rete di appartenenza.

È possibile comunicare con il web-server attraverso l'utilizzo dei seguenti protocolli: TCP/IP, UDP, HTTP, FTP, SMTP e questo rappresenta un vantaggio dal punto di vista della diagnostica e del testing da remoto (riduzione di tempo di azione e dei costi di manutenzione e di trasferta), nonostante sia la velocità di elaborazione delle pagine che la memoria a disposizione siano inferiori rispetto a un web-server tradizionale.

Il controllore è in grado di mantenere attive un numero simultaneo di

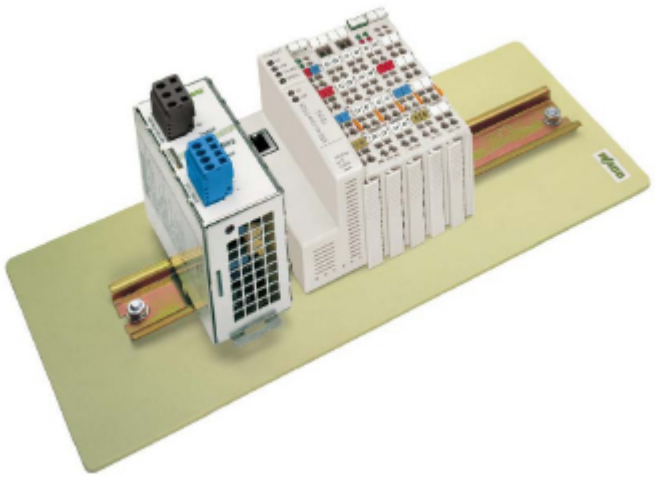


Figura 5.5: Il PLC Wago 750-841.

connessioni remote:

- 3 connessioni per l'HTTP (interrogazione di pagine web);
- 5 connessioni per MODBUS TCP/IP (lettura o scrittura in aree di memoria);
- 128 connessioni Ethernet IP.

5.5 Il sincronizzatore dati

Il software di sincronizzazione è la parte centrale del sistema e riveste un ruolo di fondamentale importanza nel contesto di questo progetto, esso ha infatti come compito principale quello di allineare i dati presenti nel database con quelli presenti nella memoria del PLC.

Per questa funzionalità è importante definire una politica di aggiornamento: si potrebbe pensare di aggiornare periodicamente tutte le variabili in gioco, ma ciò comporterebbe uno spreco delle risorse di sistema e una lentezza nell'elaborazione dei dati. Infatti il numero di dati da leggere varia da un minimo di 3400 ad un massimo di 7000 circa e, sapendo che la lettura di un dato dal PLC e la scrittura dello stesso nel database richiede quasi

1 ms, i tempi di attesa a livello utente sarebbero molto alti (dell'ordine di decine di secondi) e quindi inaccettabili.

È necessario quindi determinare un meccanismo che permetta di ridurre il numero delle variabili da aggiornare in modo da ridurre i tempi di elaborazione.

Sapendo che la navigazione dell'utente è organizzata a livelli (vedi figure 5.14 e 5.15 nel prossimo paragrafo) è facile rintracciare la reale posizione dei dati in memoria relativi a un singolo livello: se per ognuno di questi si cerca di contenere il numero di variabili, i tempi di elaborazione delle informazioni si riducono notevolmente.

Ad esempio il caso computazionalmente più oneroso è quello del livello in cui si programmano le temperature delle fasce orarie giornaliere di una zona termica (48 variabili), mentre negli altri casi il numero di dati da modificare risulta molto più ridotto. In ogni caso i tempi di elaborazione rimangono inferiori ad un secondo, quindi questa politica di aggiornamento risulta essere molto efficiente.

Per effettuare l'aggiornamento delle variabili il software di sincronizzazione deve essere a conoscenza di *quando* e a *quale* livello l'utente ha accesso.

Per quanto riguarda quest'ultimo punto, si può ottenere la posizione in cui si trova l'utente mediante il tracciamento che viene effettuato inserendo le informazioni necessarie all'interno della tabella `user_currentpage`. Invece per sapere in quale momento l'utente accede ad un livello o ne modifica i dati occorre inviare un "segnale" al sincronizzatore, in modo tale che possa avviare tutte le procedure occorrenti per l'aggiornamento dei dati.

Secondo questo principio si è pensato di realizzare un comando di *wake-up*, che tramite socket TCP [12] indica al software di sincronizzazione che l'utente ha modificato il proprio livello di navigazione. Quindi lato client vi sarà una procedura che, quando si verifica un passaggio tra due livelli adiacenti o vengono variate le impostazioni di una o più variabili, memorizza nel database la nuova posizione dell'utente e invia una stringa di wake-up contenente il codice della nuova navigazione al software di sincronizzazione.

Quest'ultimo preleverà l'identificativo della navigazione, per mezzo del

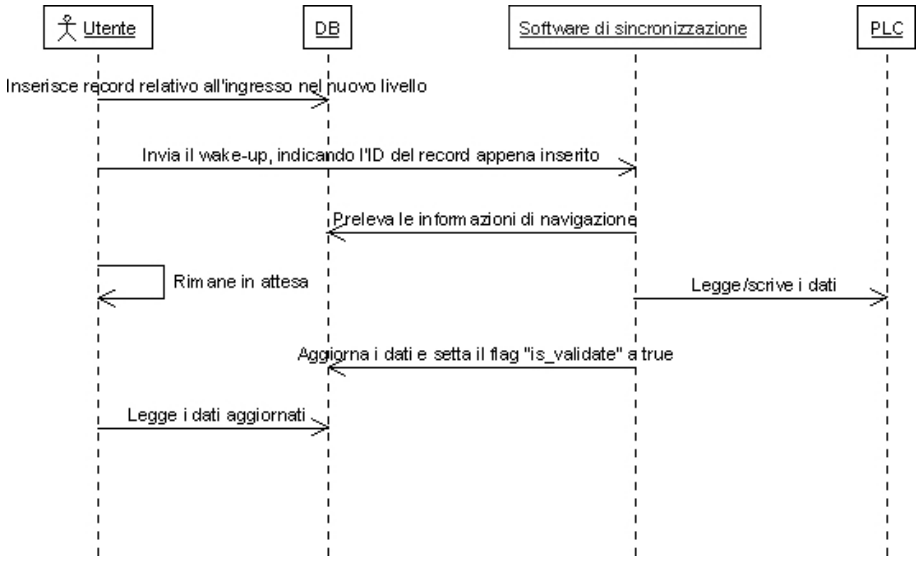


Figura 5.6: Sequence diagram relativo alla fase di sincronizzazione dati, l'utente rappresenta il componente HMI.

quale potrà risalire al livello di navigazione e quindi alle variabili coinvolte. Ovviamente il software di sincronizzazione deve mantenere un processo attivo in ascolto su una determinata porta (13000) per poter intercettare i comandi inviati dal client.

Di seguito viene indicato un sequence diagram che descrive l'interazione tra attore e componenti software.

L'utilizzo di uno stream socket risulta essere molto funzionale in quanto multipiattaforma e semplice da realizzare, prevedendo che sia solo il client ad inviare un messaggio al server e non viceversa.

Per IntelliDomus è stato realizzato un semplice protocollo che gestisce in modo unidirezionale lo scambio di informazioni tra client e server. Prima di tutto è stata definita la tipologia del messaggio da inviare, ovvero quale informazione deve contenere. A tal scopo sono state individuate tre particolari situazioni da segnalare al software di sincronizzazione:

- l'entrata nel sistema;
- la navigazione tra i livelli;
- l'uscita dal sistema.

Azione	Messaggio
Ingresso	`USER:ENTRY = [codice utente]`
Navigazione	`USER:NAVIGATION = [codice navigazione]`
Uscita	`USER:EXIT = [codice utente]`

Tabella 5.24: Definizione dei messaggi per il protocollo di segnalazione.

Seguendo queste linee guida si è pensato di costruire una stringa per ogni tipologia di messaggio, definita in questo modo:

Il codice utente viene prelevato dal campo `ID` dalla tabella `user_user`, mentre il codice della navigazione viene ottenuto dal campo `ID` dalla tabella `user_currentpage`.

Lato client è stato realizzata la classe `ClientSocket.java` che si occupa dell'invio dei messaggi al software di sincronizzazione, fornendo i metodi

```
sendEnter(String p_idUser)
sendNavigation(String p_idCurrentNavigation)
sendExit(String p_idUser)
```

che vengono richiamati ogni qualvolta sia necessario un allineamento dei dati.

Lato server viene effettuato il parsing del messaggio recuperato per ottenere le informazioni necessarie per la sincronizzazione dei dati tra database e memoria interna del PLC. Dopo aver effettuato l'aggiornamento dei dati il software di sincronizzazione segnala all'utente il completamento della procedura settando a `true` il flag `is_validate` presente nella tabella `user_currentpage`.

Il client rimane in attesa fino a quando, leggendo questo flag, si rende conto che l'operazione è stata completata correttamente. Se allo scadere del tempo massimo prestabilito (30 secondi) il flag `is_validate` è ancora a `false` viene generata un'eccezione e l'utente viene reindirizzato verso una pagina di errore.

Un altro aspetto importante è quello della gestione delle transazio-

ni: dato che generalmente ad ogni aggiornamento corrisponde la modifica di più valori si rende necessario un meccanismo che consenta di verificare che l'aggiornamento di *tutte* le variabili avvenga correttamente, in caso contrario va ripristinata la situazione precedente (nessuna modifica deve essere applicata, garantendo le proprietà di atomicità e consistenza dell'operazione).

Per fare questo ogni "blocco" di aggiornamenti viene identificato da un numero progressivo (campo `SCN` nella tabella `user_currentpage`) e per ogni variabile modificata viene scritto un record nella tabella `transfert_log` contenente tutte le informazioni necessarie per ritornare alle condizioni iniziali in caso di errore:

- ID della variabile modificata;
- ID della navigazione corrente;
- vecchio valore della variabile;
- nuovo valore della variabile;
- data e ora di aggiornamento.

Inoltre durante l'allineamento dei dati con la memoria del PLC il software di sincronizzazione scrive all'interno del record la data e ora di caricamento dei dati in memoria. Vengono effettuati tre tentativi di aggiornamento, se non è in grado di completare tutte le operazioni, ad esempio per problemi di rete, la transazione fallisce (e quindi viene settato a `false` il flag `is_committed` presente nella tabella `user_currentpage`), ma grazie ai record scritti precedentemente il database viene ripristinato alle condizioni iniziali.

In caso contrario la transazione ha successo e il sincronizzatore lo segnala all'utente impostando il flag `is_committed` a `true`.

Il fatto di tenere traccia della navigazione dell'utente e delle variabili modificate è utile non solo per la comunicazione tra i vari componenti e la correzione degli errori, ma anche per avere uno storico dell'utilizzo del sistema, inoltre rappresenta un metodo diagnostico nel caso in cui si verifi-

chi qualche anomalia, per aiutare a capire che cosa ha generato l'eventuale errore.

Per la gestione delle variabili sincrone (data/ora del PLC e generazione allarmi) l'aggiornamento dei valori avviene in modo diverso: è il software di sincronizzazione che si occupa dell'aggiornamento delle variabili presenti nel database verificando periodicamente il valore assunto dai registri di memoria corrispondenti situati nel PLC.

A sua volta il client ogni 60 secondi richiama una procedura che legge i valori presenti nella tabella `transfert_sync`, aggiornando l'interfaccia utente.

5.6 Struttura del software HMI

Per quanto riguarda l'interazione tra utente e PLC vengono messi a disposizione due metodi: l'utilizzo di un'interfaccia grafica tradizionale e l'accesso via remoto, attraverso un'applicazione web appositamente sviluppata.

Utilizzando il paradigma di progettazione MVC si è cercato di sviluppare un'applicazione modulare, strutturata in tre layer generali (*wrapper*, *controller* e *panel*/*servlet* + *JSP*). Il vantaggio di questo approccio consiste nel fatto che i layer di livello più basso (wrapper e controller) sono comuni a entrambe le versioni del software (GUI e Web) e non sono influenzati dal layer di interfaccia, che può essere modificato liberamente.

Eventualmente è possibile anche implementare una nuova interfaccia (ad esempio per un dispositivo palmare), che si collega al sistema sfruttando i metodi forniti dai controller, senza il bisogno di dover sviluppare una nuova applicazione da zero.

Inoltre si è prestata attenzione alla nomenclatura dei package e delle classi, ad esempio utilizzando determinati suffissi che consentono di capire subito quali sono i servizi offerti.

In figura 5.7 è possibile vedere l'organizzazione modulare dell'applicazione, dove vengono evidenziati i seguenti componenti e relativi package:

Componente HMI Comune ad entrambe le versioni dell'applicativo, contiene i package relativi ai layer di livello più basso (wrapper e controller),

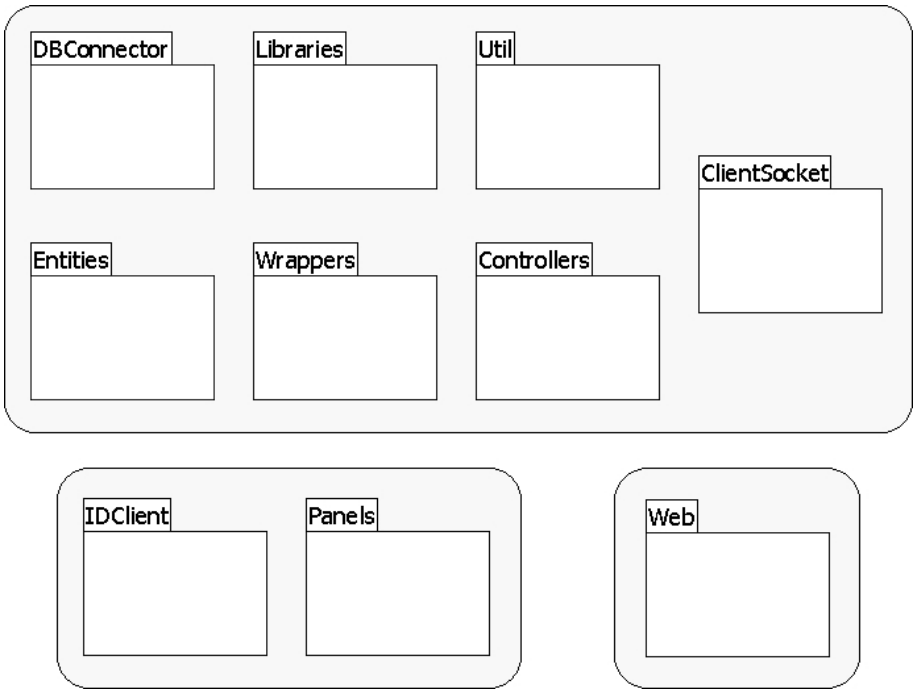


Figura 5.7: Struttura a package del software HMI.

le classi entità, il modulo per la comunicazione via socket, i driver e la classe `DbConnector` per la connessione al database e altre classi di utilità generale, come formattatori e parser per numeri e date;

Interfaccia GUI Contiene tutti i panel relativi all'interfaccia grafica dell'applicazione e classi per la loro gestione, inoltre il package IDClient comprende le classi `Main` per l'avvio del programma e `GuiConst`, in cui sono definite tutte le variabili di configurazione necessarie (informazioni utente, identificatori dei panel, impostazioni per i timeout e altri paramteri per la corretta gestione dell'interfaccia utente);

Interfaccia Web Contiene i package relativi a servlet e pagine JSP, inoltre, a differenza della versione GUI, comprende classi per la gestione delle sessioni Web.

Analizzando in particolare il componente principale dell'applicativo è possibile comprenderne meglio la struttura e capire in che modo avviene l'interazione tra utente, database e software di sincronizzazione. I class

diagram che in seguito verrano rappresentati nelle immagini si riferiscono in particolare alla gestione del sistema di riscaldamento, le altre utente hanno un'organizzazione analoga.

La classe `ClientSocket` fornisce i metodi necessari per la creazione del socket di comunicazione e l'invio al software di sincronizzazione delle informazioni relative all'utente e al livello di navigazione corrente, come esposto precedentemente nel paragrafo 5.5.

Il package `Entities` contiene tutte le classi entità, distinte per categoria: `GlobalServices`, `TermicZone`, `Illumination`, `Irrigation`, `Automatisms`, `Alarms`, `INIElement`, `TSElement`, `UIElement`, `Users`, `Weekday` e `Exceptions` (vedi figure 5.8 e 5.9). Questa suddivisione verrà utilizzata anche per tutti gli altri layer.

Le classi entità vengono utilizzate per la gestione degli oggetti in memoria e gli attributi rappresentano l'esatta copia dei campi delle tabelle del database, in più per ogni attributo sono forniti i metodi get/set per la lettura/scrittura dei valori.

In generale per l'istanziazione dele classi viene sempre definito un costruttore pubblico di default (Java richiede che sia presente per tutte le classi usate nelle pagine JSP), un costruttore che prende come parametro l'ID del record corrispondente e un costruttore per copia.

Per la gestione delle eccezioni viene definita la classe `ProtoExc` che estende `java.lang.Exception`, aggiungendo due attributi: un codice rappresentante il tipo di errore generato (tutti gli errori possibili, con i rispettivi messaggi, sono definiti nella tabella `ui_element_description`) e, per le eccezioni riguardanti l'interfaccia GUI, il panel di provenienza.

Da questa classe derivano poi tipi particolari di eccezione: errore generato dall'HMI, errore generato dal software di sincronizzazione, warning. Questi ultimi si differenziano dagli errori veri e propri perché rappresentano solo avvisi, quindi l'esecuzione dell'operazione che genera un warning non viene bloccata ma può proseguire.

I wrapper si occupano principalmente del recupero dei dati dal database e del loro aggiornamento. Dato che questo layer fa ancora molto riferimento alla struttura del database, anche qui le classi sono raggrup-

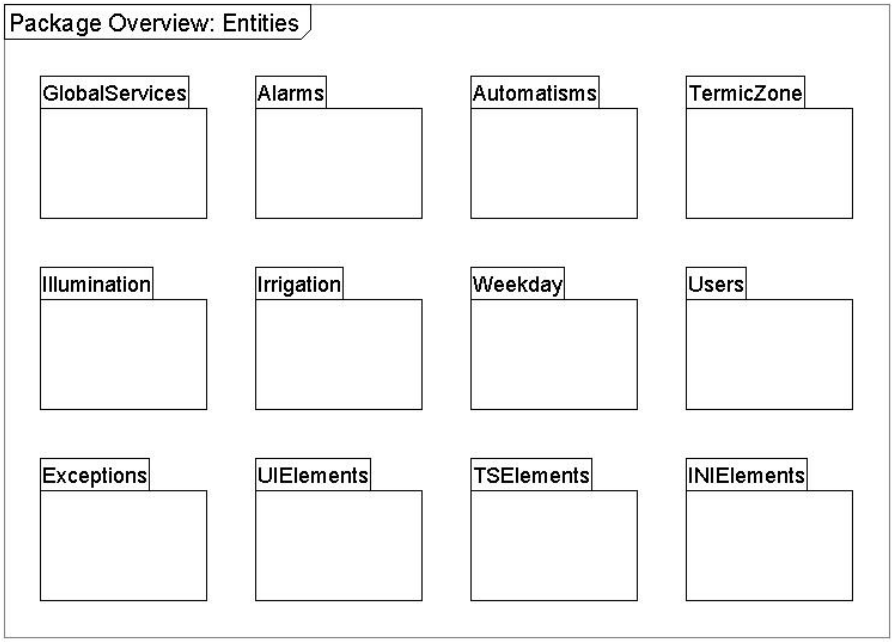


Figura 5.8: Struttura del package Entities.

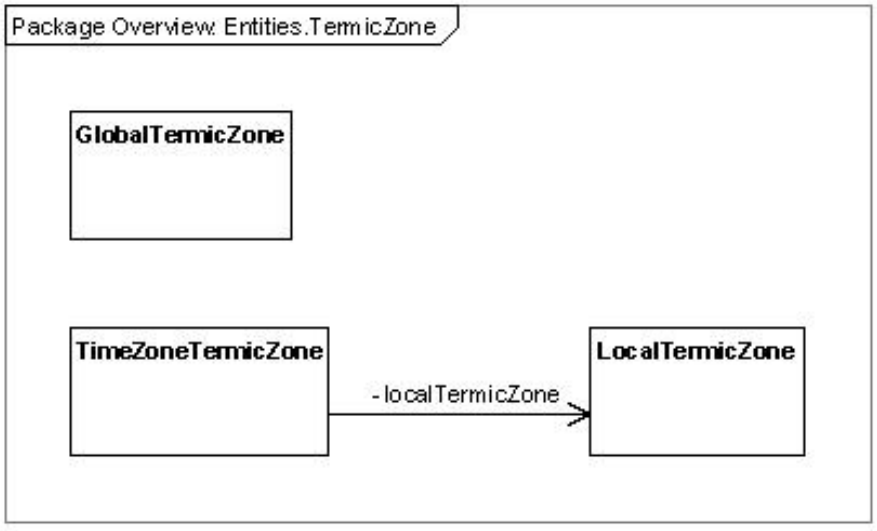


Figura 5.9: Struttura Entities.TermicZone.

pate per categoria ed è stato creato un wrapper per ogni entità presente, come si può vedere nelle figure 5.10 e 5.11.

In particolare viene definita la classe `ProtoWrp`, che viene poi estesa da tutti gli altri wrapper. Essa contiene come attributi gli oggetti `DBConnector`, `Connection` e `DataFormatter` (per la formattazione delle date), tutti dichiarati come static in quanto possono essere condivisi tra tutte le istanze create.

Vi è poi il metodo `setDBConnector` per l'impostazione dei primi due attributi e `insertTransfertLogDB`, che si occupa dell'inserimento dei record di log nella tabella `transfert_log`.

Gli altri wrapper possiedono in più tutti i metodi relativi alle entità a cui si riferiscono, necessari per la lettura e la scrittura dei dati nel database. Anche in questo caso è stata adottata una nomenclatura e uno stile comune.

Se si verifica un errore durante l'esecuzione delle operazioni vengono catturate le eccezioni prodotte, rilanciando un oggetto di tipo `ErrorExc` contenente il codice dell'errore corrispondente.

Per la gestione delle transazioni si osservi che di default la proprietà auto-commit dell'oggetto `Connection` è abilitata, cioè l'applicazione effettiva di uno statement SQL avviene subito dopo l'esecuzione. Per gestire più statement in un'unica transazione è necessario disabilitare questo flag mediante il metodo `setAutoCommit` e utilizzare i metodi `commit`/`rollback`, messi a disposizione dalla classe `Connection`.

I controller rappresentano le classi di collegamento tra interfaccia e wrapper, come descritto dal paradigma MVC. È stato deciso di definire un controller per ogni panel dell'interfaccia, che fornisce tutti i metodi per la gestione dei dati da presentare all'utente e per il controllo e l'aggiornamento dei valori impostati.

Anche in questo caso abbiamo un controller generico (`ProtoCtrl`) da cui ereditano tutti gli altri (figure 5.12 e 5.13): esso contiene i wrapper relativi alle entità da gestire e mette a disposizione funzionalità di utilità generale quali l'impostazione della connessione, la verifica della validazione della navigazione utente e il recupero delle impostazioni del sistema, di informazioni sullo stato del PLC e delle immagini e label relative alla lingua scelta.

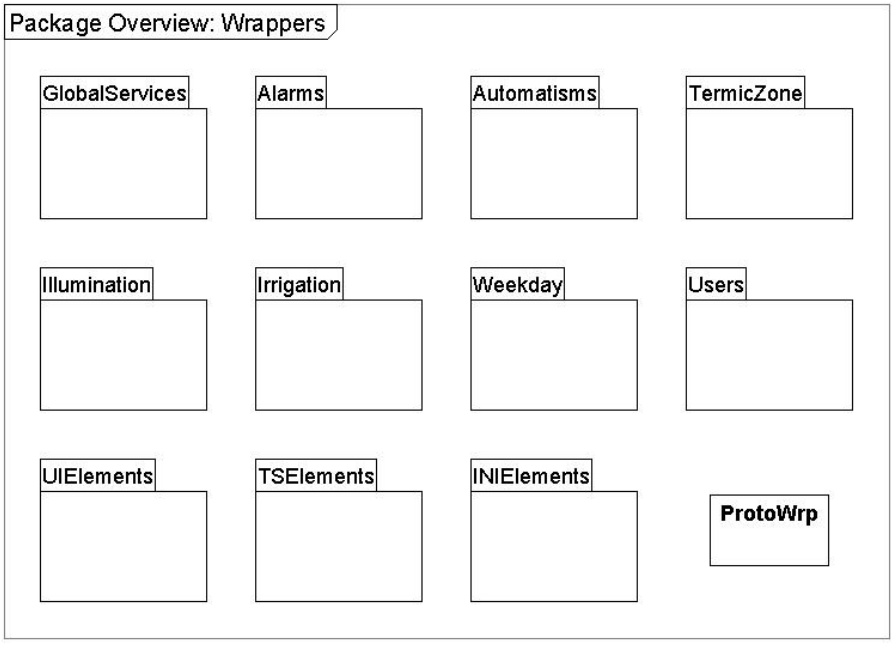


Figura 5.10: Struttura del package Wrappers.

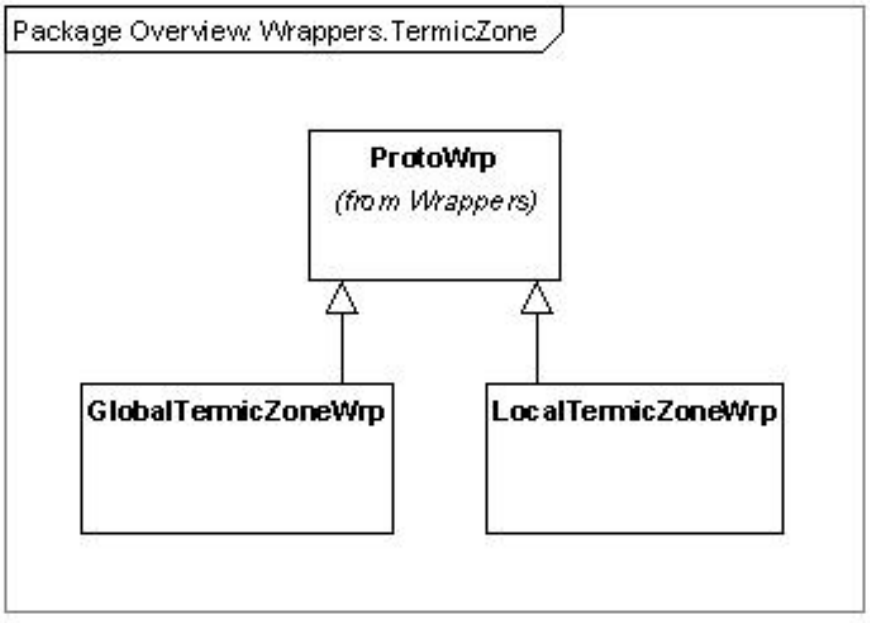


Figura 5.11: Struttura Wrappers.TermicZone.

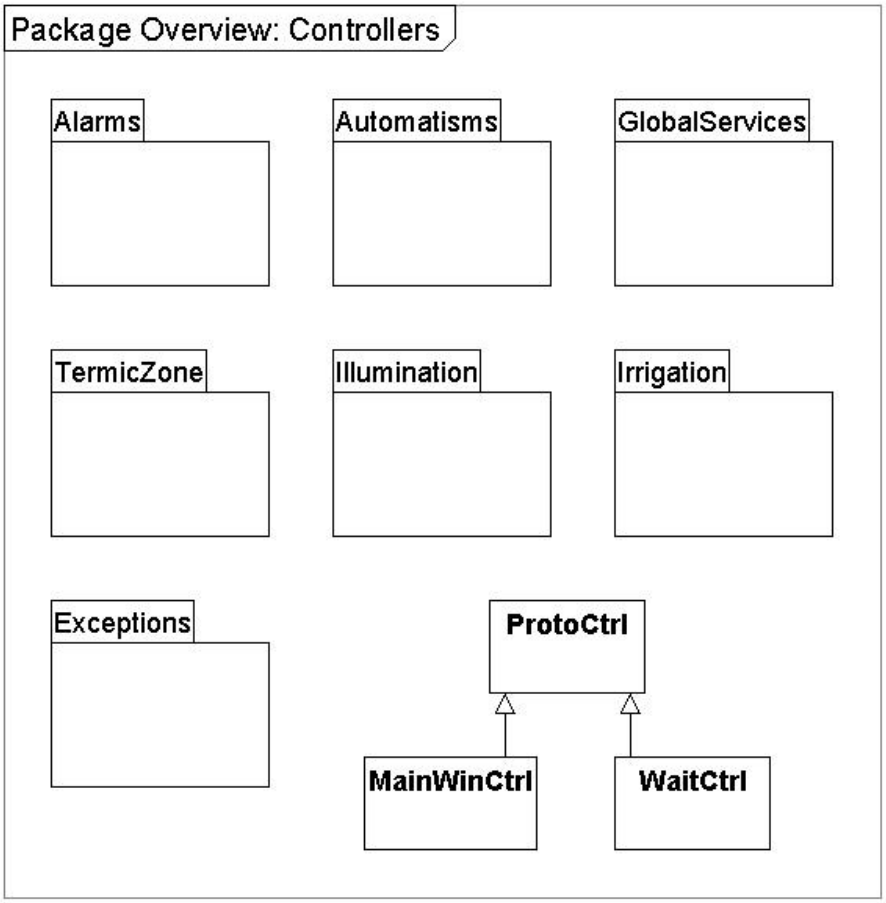


Figura 5.12: Struttura del package Controllers.

Gli altri controller in più forniscono i metodi relativi ai dati che interessano l'interfaccia corrispondente.

Per quanto riguarda la gestione degli errori vengono catturate le eccezioni eventualmente sollevate dal layer sottostante, decidendo di volta in volta se risolvere il problema o rilanciare l'eccezione: in quest'ultimo caso l'utente verrà reindirizzato ad una schermata apposita dove verrà visualizzato il messaggio di errore corrispondente e un pulsante per tornare alla schermata precedente se si tratta di un errore dovuto al software HMI, o per ritentare la connessione se l'errore è dovuto ad un problema di comunicazione con il software di sincronizzazione.

Per quanto riguarda l'interfaccia utente nelle figure 5.14 e 5.15 è indi-

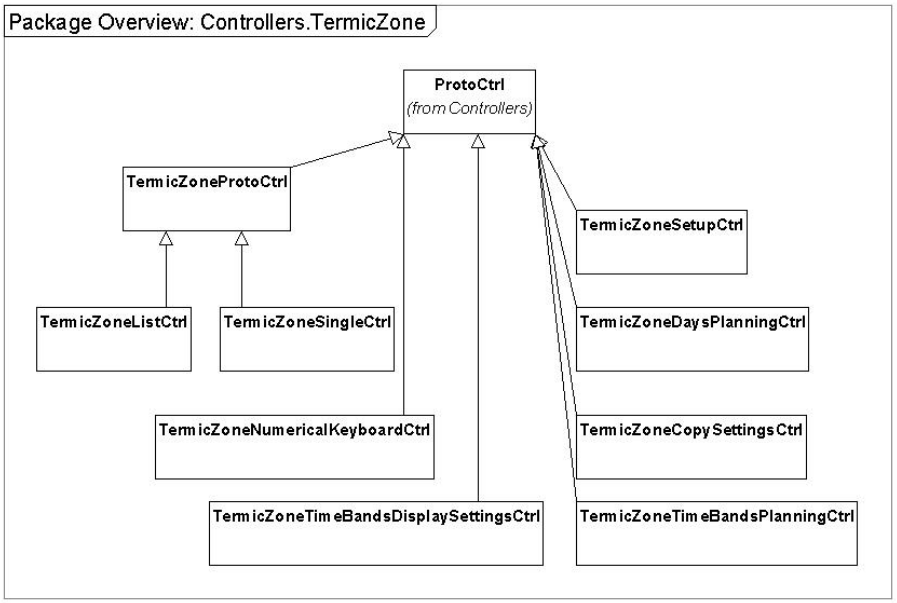


Figura 5.13: Struttura Controllers.TermicZone.

cata la struttura dei pannelli di controllo, dove viene evidenziata l'impostazione gerarchica: ad ogni schermata è riferito un livello di navigazione univoco che permette di indicare la precisa posizione dell'utente durante la navigazione. Di seguito verranno analizzati tutti i livelli relativi alle zone termiche (le altre utente sono analoghe), descrivendo la struttura delle schermate e i controlli presenti, comuni a entrambe le interfacce, mentre nei paragrafi 5.7 e 5.8 verranno illustrate le caratteristiche particolari di ognuna delle due versioni.

All'interno della schermata principale viene presentato l'elenco delle utenze disponibili, vi si accede cliccando sulle apposite icone (vedi figura 5.16).

La schermata in figura 5.17 mostra l'elenco di tutte le zone termiche presenti nell'abitazione e indica lo stato generale del riscaldamento (estate/inverno, acceso/spento).

Nella schermata in figura 5.19 si può vedere la singola zona termica precedentemente selezionata nella schermata principale. Il suo aspetto dipende dalle correnti impostazioni di funzionamento: manuale, on/off, automatico.

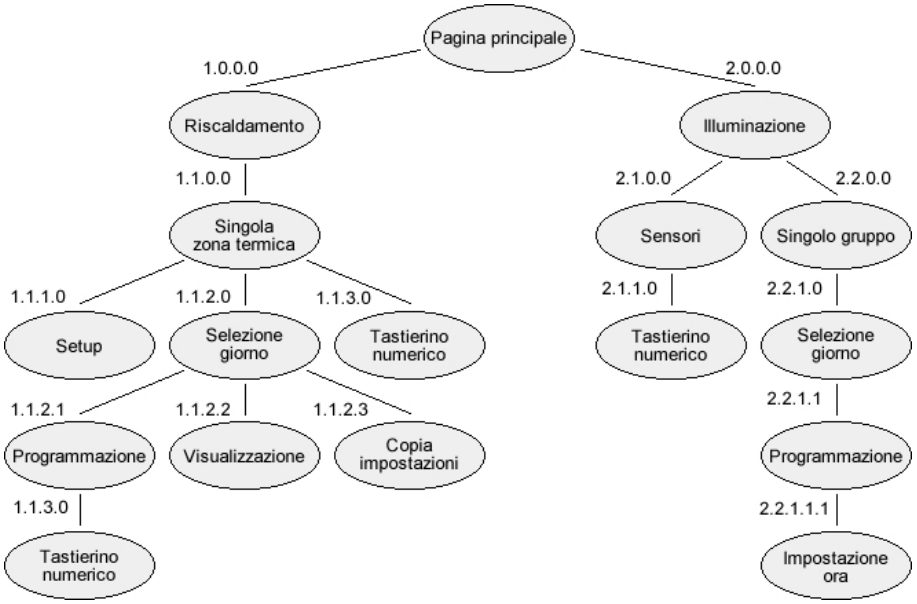


Figura 5.14: Livelli di navigazione (1).

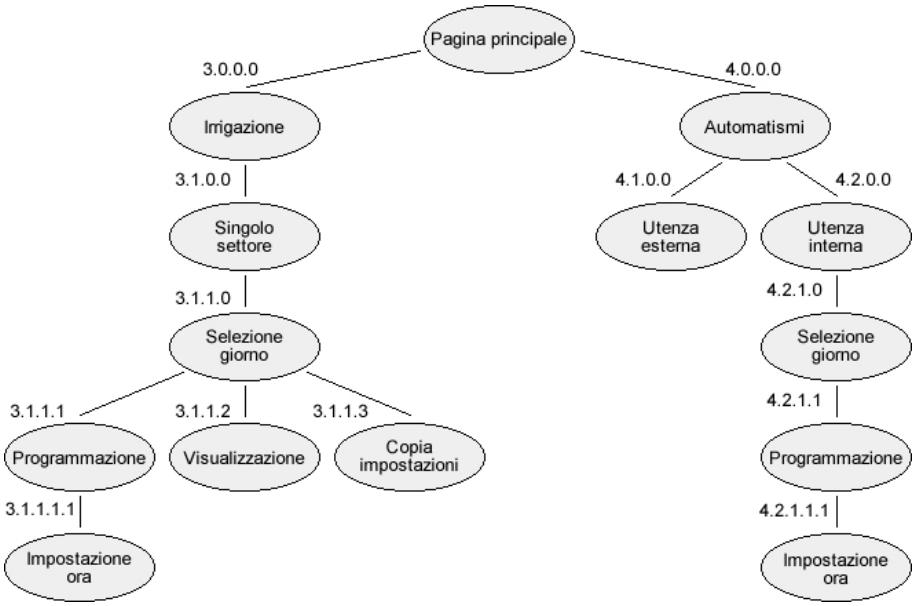


Figura 5.15: Livelli di navigazione (2).

La schermata in figura 5.20 permette la scelta dei giorni di programmazione automatica delle zone termiche, evidenziando il giorno corrente. Da qui si può accedere alla schermata per la copia delle impostazioni giornaliere da una zona all'altra.

La funzionalità di copia delle impostazioni (vedi figura 5.21)permette di modificare la programmazione di un determitato giorno di una zona termica utilizzando i parametri giornalieri di un'altra zona termica.

Nella schermata di programmazione l'utente seleziona una o più fasce orarie e, cliccando sul pulsante a destra, accede al tastierino numerico, dove inserisce la temperatura desiderata, ovvero il setpoint che il sistema deve cercare di mantenere durante tale intervallo di tempo, nel caso in cui si trovi in modalità automatico. La fascia oraria è fissa ed ha durata di 30 minuti, come da figura 5.22.

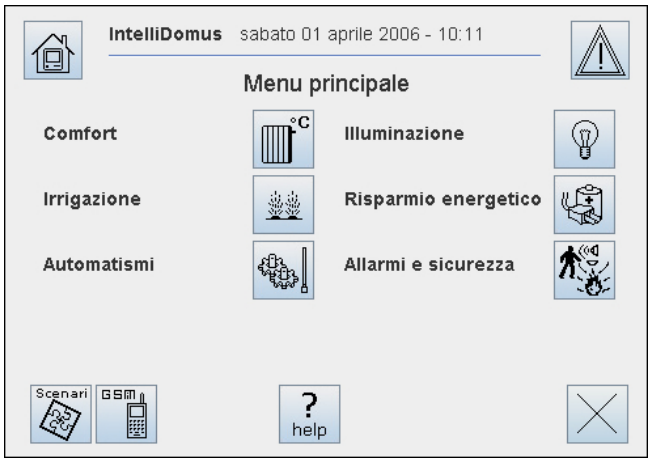
Una schermata simile permette la visualizzazione delle impostazioni della zona termica (vedi figura 5.22).

La schermata in figura 5.18 permette all'utente ma soprattutto all'installatore di impostare alcuni parametri di setup per la singola zona termica:

- differenziale termico utilizzato dal sistema per eseguire il controllo della temperatura, ovvero il raffronto tra la temperatura misurata e quella di setpoint;

- temperatura minima antigelo. Il valore "Temperatura" viene impiegato per determinare la soglia oltre la quale (in positivo o in negativo) la temperatura rilevata viene considerata effettivamente diversa dal setpoint impostato, mentre con "Ritardo" viene stabilito l'intervallo minimo (in minuti) dopo il quale il sistema agisce sulla zona e sul riscaldamento centrale in presenza di uno scostamento tra lettura e setpoint.

Per l'impostazione di alcuni dati numerici viene utilizzato un tastierino numerico comune, come quello in figura 5.20.



Home Riporta l'utente alla pagina iniziale.

Allarmi Pulsante per la visualizzazione degli allarmi. Se ci sono allarmi in corso il simbolo è di colore giallo.

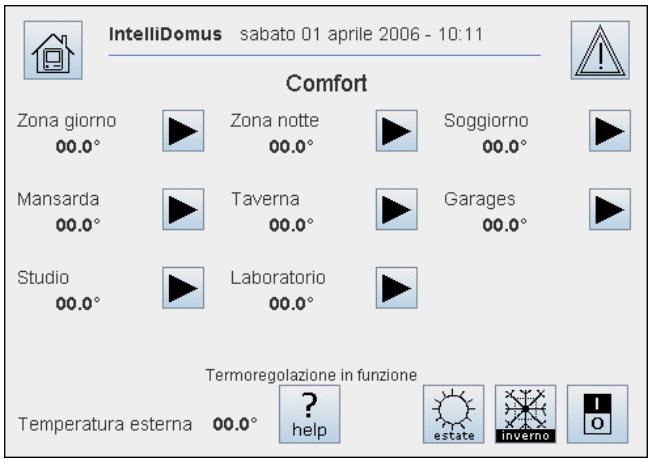
Scenari Pulsante per la programmazione delle utenze in base ad un determinato evento (risveglio, uscita, rientro, vacanza, ecc.).

GSM Pulsante per la gestione del sistema via GSM.

Help Pulsante per la sezione di aiuto che descrive (tramite le sue varie schermate) come utilizzare il sistema.

Exit Pulsante per la chiusura del client IntelliDomus.

Figura 5.16: Schermata principale.



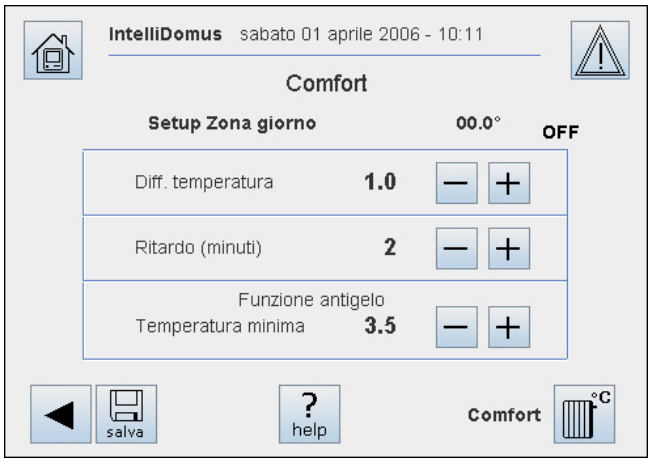
Pulsante di funzionamento - modalità Estate Premendolo si disabilita il suo complementare (Inverno) e si mette l'intero impianto in modalità di funzionamento "Estate", in tal modo il relè di attivazione della valvola di zona lavora in modo opposto rispetto al modo "Inverno", chiudendosi se la temperatura scende sotto al setpoint ed impedendo al fluido refrigerante di circolare ancora. Quando attivato la scritta "inverno" (rispettivamente per "estate") appare su sfondo nero.

Pulsante di funzionamento - modalità Inverno Premendolo si disabilita il suo complementare (Estate) e l'intero impianto opera in modo che le singole valvole di zona si chiudono se la temperatura sale sopra al setpoint impostato.

Pulsante generale di accensione/spegnimento impianto di condizionamento Pulsante per l'accensione/spegnimento dell'intero sistema di riscaldamento/raffrescamento. Se spento vengono disabilitate anche le funzionalità di base come l'"Antigelo".

Con questi pulsanti si accede alla singola zona termica intesa come pagine per la sua visualizzazione ed impostazione (funzionamento manuale, automatico, fasce orarie e calendario settimanale).

Figura 5.17: Schermata lista zone termiche.



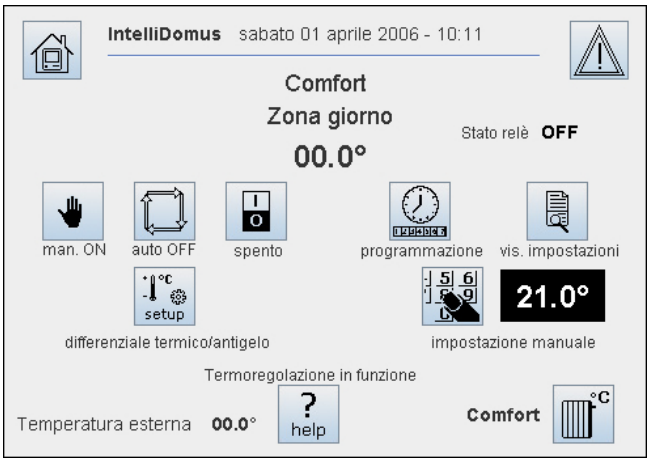
Pulsante generico per incremento valore Aumenta di 1 decimo di grado il valore del differenziale di temperatura o della temperatura minima per la funzionalità antigelo, mentre aumenta di 1 minuto il parametro ritardo.

Pulsante generico per decremento valore Diminuisce di 1 decimo di grado il valore del differenziale di temperatura o della temperatura minima per la funzionalità antigelo, mentre decrementa di 1 minuto il parametro ritardo.

Pulsante di salvataggio Pulsante per confemare e memorizzare i nuovi valori.

Pulsante back Pulsante per tornare alla schermata precedente senza memorizzare i parametri impostati.

Figura 5.18: Schermata impostazione parametri differenziale termico/antigelo.



Pulsante accensione zona termica - stato OFF In questa situazione la zona termica è spenta: non partecipa all'attivazione del sistema centrale di riscaldamento né tanto meno è riscaldata dallo stesso (valvola chiusa).

Pulsante accensione zona termica - stato ON In questa situazione la zona termica è attiva: portandolo da OFF a ON la zona passa in automatico di default.

Pulsante funzionamento automatico - stato OFF Appare in questa forma se la zona è spenta o accesa in manuale.

Pulsante funzionamento automatico - stato ON Appare in questa forma quando lo si preme e la zona termica è accesa.

Pulsante funzionamento manuale - stato di OFF Appare in questa forma se la zona è spenta o accesa in automatico.

Pulsante funzionamento manuale - stato di ON Appare in questa forma quando lo si preme e la zona termica è accesa.

Pulsante programmazione Consente di accedere alla prima pagina della sottosezione di programmazione, da cui si procede a tutte le schermate di configurazione per il funzionamento automatico della zona termica in questione. La prima pagina di sottosezione consente (vedi figura 5.20) di scegliere il giorno su cui operare, di default è evidenziato il giorno corrente.

Pulsante visualizzazione impostazioni Tramite questo pulsante si accede alle schermate che visualizzano le impostazioni delle fasce orarie per il giorno che verrà selezionato nella schermata successiva.

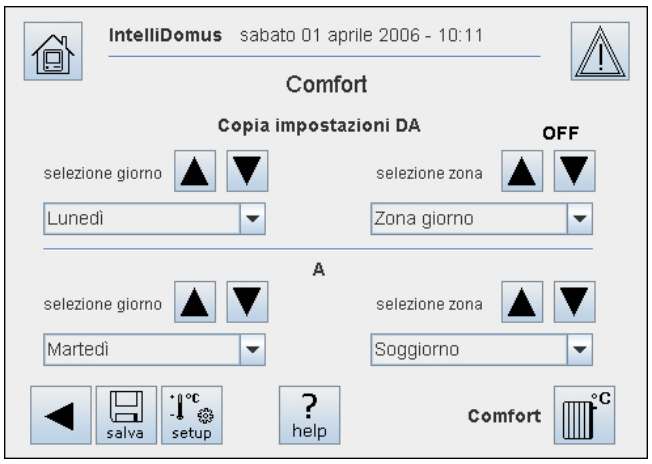
Pulsante setup Permette di accedere alla pagina di configurazioni relative al differenziale termico tra setpoint e valore letto e alla funzione antigelo.

Pulsante di ritorno pagina principale zone termiche Per tornale alla schermata iniziale della sezione. È presente in tutte le schermate relative alla gestione delle zone termiche descritte in seguito.

Figura 5.19: Schermata singola zona termica.

Pulsante copia Pulsante per accedere alla schemata di copia dei dati da un giorno all'altro (giorni selezionabili liberamente), anche tra zone termiche differenti, il tutto per velocizzare l'immissione delle impostazioni (vedi figura 5.21).

Figura 5.20: Schermate tastierino numerico e selezione giorno.



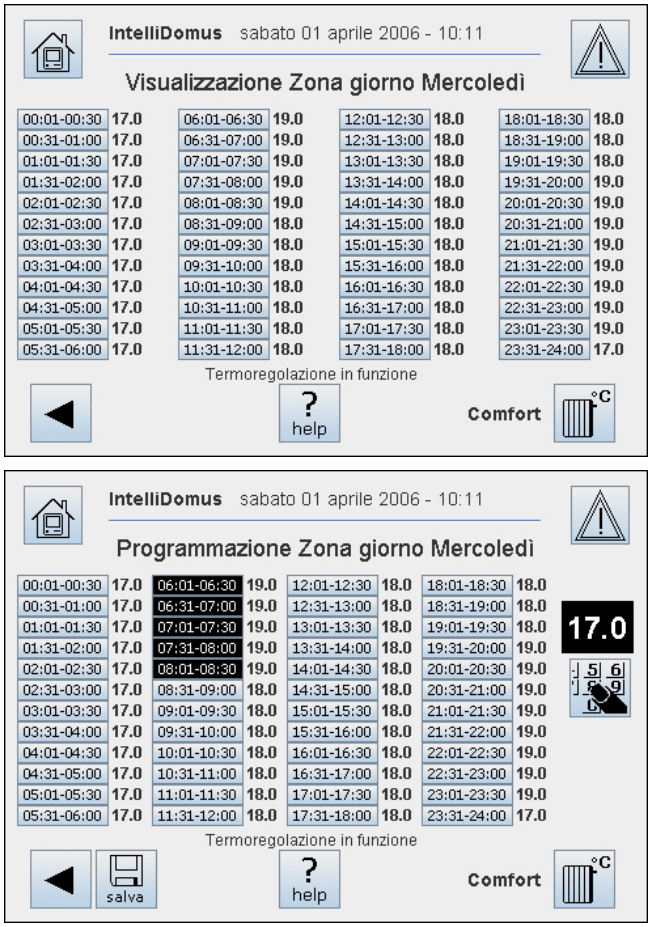
Pulsante scorrimento elenco - precedente Premendolo si passa al giorno o alla zona precedente quello/a correntemente selezionato: in tutto sono presenti 4 pulsanti di questo tipo, ciascuno dei quali interviene su una zona o su un giorno ben specifico.

Pulsante scorrimento elenco - successivo Premendolo si passa al giorno o alla zona successiva a quello/a correntemente selezionato.

Pulsante di salvataggio Pulsante per confemare e memorizzare le nuove impostazioni.

Pulsante back Pulsante per tornare alla schermata precedente senza memorizzare i parametri impostati.

Figura 5.21: Schermata copia impostazioni.



Fascia oraria È rappresentata una generica fascia oraria non selezionata e su cui quindi non si può modificare il setpoint di temperatura.

00:01-00:30

Fascia oraria selezionata La scelta della fascia oraria si effettua selezionando il pulsante corrispondente, possono essere effettuate scelte multiple.

00:01-00:30

Pulsante di salvataggio Al termine delle modifiche apportate ai vari setpoint si utilizza questo pulsante per confemare e memorizzare i nuovi valori affinchè diventino operativi.

Pulsante back Pulsante per tornare alla schermata precedente senza memorizzare i parametri impostati.

Figura 5.22: Schermate visualizzazione e programmazione fasce orarie.

5.7 Versione GUI

L'interfaccia GUI è gestita attraverso JPanel e anche in questo caso si sfrutta il meccanismo dell'ereditarietà (vedi figure 5.23 e 5.24), inoltre è stata definita la classe `PanelMaster` che si occupa della loro gestione e visualizzazione: essa contiene un array in cui vengono inseriti i riferimenti a tutti i panel che compongono l'applicazione (la loro creazione avviene all'avvio del programma), un riferimento al panel corrente, che viene aggiornato in base alla navigazione dell'utente, e tutti i metodi per consentire il passaggio di oggetti da un panel all'altro.

La caratteristica di mantenere in memoria tutti i panel (in tutto circa 40) consente di rendere praticamente nulli per l'utente i tempi di attesa necessari per il passaggio da un panel all'altro.

L'area grafica dell'applicazione è suddivisa in due parti: la parte superiore è fissa e contiene informazioni relative alla data/ora corrente (riferita al PLC) e i pulsanti per il ritorno alla pagina principale e per la visualizzazione degli allarmi; la parte inferiore contiene invece il panel corrente, all'avvio viene visualizzata la pagina principale con l'elenco delle utente.

I dati relativi alla data/ora e agli allarmi sono rappresentati da variabili sincrone, quindi è il software che si occupa della lettura dei valori aggiornati, mediante un thread eseguito periodicamente (ogni 60 secondi) che esegue la lettura dei valori presenti nella tabella `transfert_sync` del database.

Con l'impostazione della variabile `FULL_SCREEN` definita in `GuiConst` è invece possibile specificare se si vuole visualizzare l'applicazione in modalità normale o fullscreen. In quest'ultimo caso le dimensioni dell'area grafica verranno adattate alla dimensione dello schermo, centrando il contenuto del panel. Questa modalità è utile nel caso della visualizzazione su TV.

Le altre operazioni eseguite all'avvio del programma sono quelle di creare la connessione al database e di passarla ai wrapper, recuperare dalla tabella `ui_element_description` tutti gli oggetti (sia immagini che label) che consentono la localizzazione del software nella lingua scelta, leggere dal file di configurazione i dati relativi all'utente registrando le informazioni

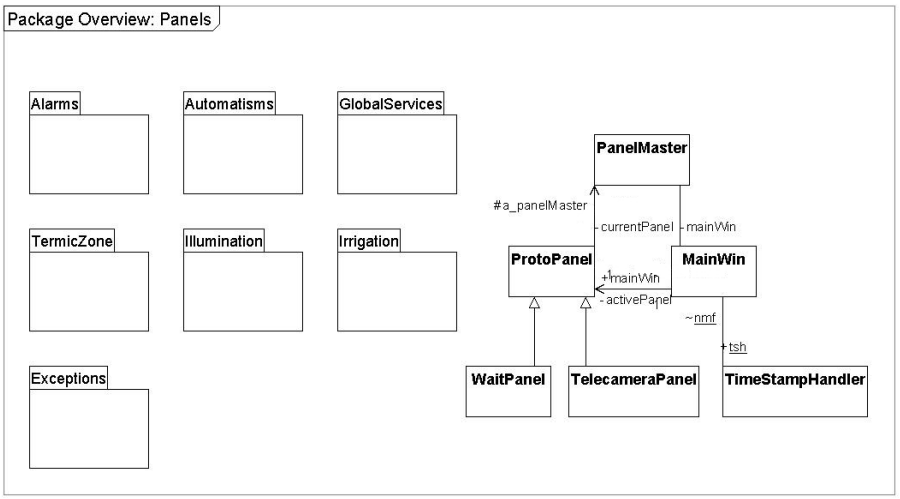


Figura 5.23: Struttura del package Panels.

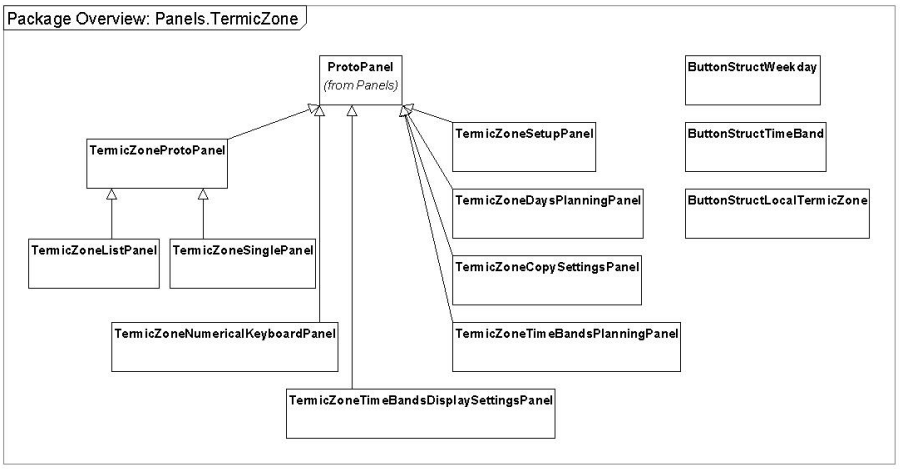


Figura 5.24: Struttura Panels.TermicZone.

relative alla sessione appena iniziata e collegarsi al software di sincronizzazione, verificando se la connessione va a buon fine (in caso contrario l'utente viene reindirizzato al panel di errore corrispondente).

Con la creazione dei panel viene creato il controller corrispondente e viene impostato il riferimento al gestore dei panel, inoltre vengono recuperati dal database tutti i dati relativi agli oggetti da visualizzare in base al panel in cui ci si trova (ad esempio per il panel con la lista delle zone termiche viene creato un vettore che le contiene) e le label vengono aggiornate di conseguenza, mediante la chiamata del metodo `updateValues`. Se si verifica un errore l'utente viene reindirizzato al panel apposito.

Quando avviene il passaggio da un panel all'altro solo i dati vengono ricaricati (non le immagini e le label) e viene registrata la navigazione dell'utente nelle apposite tabelle, eventualmente aggiornando i dati modificati.

Nel caso in cui ci sia una lista di oggetti da presentare si è deciso di introdurre oggetti appositi per la gestione di più label e pulsanti: ad esempio per la programmazione delle fasce orarie viene creato un vettore di 48 elementi che contiene i riferimenti alle label e ai pulsanti di ogni fascia oraria, con il rispettivo stato (selezionato/non selezionato). Questo metodo consente una maggiore chiarezza e semplicità nella scrittura del codice.

Oltre ai panel che contengono i controlli per la gestione del sistema sono stati definiti altri panel speciali: un panel per gli errori durante l'accesso al database o generati dal software HMI (che consente di tornare alla schermata precedente), un panel per gli errori relativi alla comunicazione con il software di sincronizzazione (che permette solo di riprovare a connettersi), un panel relativo ai warning (per ora non utilizzato) e un panel di attesa.

Quest'ultimo viene attivato quando l'utente applica delle modifiche ai dati: gli aggiornamenti sul database avvengono in modo immediato, mentre per le modifiche ai registri di memoria del PLC potrebbe essere necessario qualche secondo. Quindi, per evitare che nel frattempo l'utente esegua operazioni indesiderate, viene visualizzato temporaneamente questo panel con un messaggio di attesa: quando gli aggiornamenti sono stati appli-

cati (cioè il software di sincronizzazione ha validato la navigazione) viene riattivato il panel corretto.

5.8 Versione Web

La versione web sfrutta gli stessi moduli per la connessione al database e la gestione dei dati (controller e wrapper), si differenzia solo per il layer di interfaccia, composto da pagine JSP e servlet.

Un servlet non è altro che un'applicazione Java in esecuzione su una JVM residente su un server. Questa applicazione tipicamente esegue una serie di operazioni che producono in output codice HTML che verrà poi inviato al client per essere interpretato da un qualsiasi browser.

A differenza di uno script CGI i servlet vengono caricati solo una volta, al momento della prima richiesta, e rimangono residenti in memoria pronti per servire le richieste fino a quando non vengono chiusi, con ovvi vantaggi in fatto di prestazioni (risultano infatti un po' più lenti in caricamento, ma una volta aperti le successive richieste vengono evase molto velocemente).

Mediante l'oggetto `HttpServletRequest` il servlet ha accesso alle informazioni di intestazione specifiche del protocollo HTTP ed è quindi in grado di gestire le richieste web ricevute. L'oggetto `HttpServletResponse` fornisce invece tutti gli strumenti per inviare i risultati dell'esecuzione della pagina JSP, metre attraverso l'oggetto `Session` è possibile gestire i dati relativi alla sessione.

I dati agganciati all'oggetto `HttpServletResponse` vengono poi resi disponibili attraverso l'istruzione

```
<jsp:useBean id="nomeBean"
             scope="ambito_oggetto"
             class="classe" />
```

all'interno della pagina JSP, che si occupa della presentazione delle informazioni all'utente.

Per quanto riguarda le differenze di impostazione grafica rispetto ai panel, si avranno immagini invece di pulsanti e layer al posto di label,

che verranno aggiornati utilizzando DHTML (*Dynamic HTML*), un linguaggio di scripting lato client che permette di cambiare in modo dinamico la rappresentazione e il contenuto di un documento.

Inoltre, dato che in questo caso sarà sicuramente disponibile una tastiera come dispositivo di input, non sono più necessari i tastierini numerici presenti invece nella versione GUI, ma le modifiche avvengono inserendo direttamente i dati tramite form.

Un altro aspetto importante consiste nella gestione degli utenti: in questo caso viene utilizzato un modulo di autenticazione che nella versione GUI non è necessario, quindi all'ingresso nel sistema viene presentata una schermata di login, inoltre dopo l'autenticazione verrà creata e gestita una sessione web, che consente di mantenere nel passaggio da una pagina all'altra le informazioni relative all'utente.

L'ultima differenza riguarda i tempi di attesa: nella versione GUI i tempi di risposta saranno discretamente inferiori, in quanto l'applicazione viene eseguita localmente. Al contrario nella versione web le richieste al PLC saranno inviate da remoto, e quindi non è possibile conoscere a priori le prestazioni della rete, che possono dipendere da diversi fattori (disponibilità di banda, traffico, ecc.).

Capitolo 6

Conclusioni

6.1 Bilancio del lavoro svolto

L'utilizzo di adeguate tecniche di progettazione e di sviluppo del software ha consentito di ottenere un sistema che soddisfa tutti i requisiti richiesti: modularità, indipendenza dalla piattaforma e semplicità d'uso.

Durante il corso del progetto sono state effettuate alcune prove di visualizzazione dell'interfaccia utente, sia su monitor LCD che su semplici televisioni. Quest'ultima situazione ha portato alla luce alcune problematiche legate alla gestione dei driver della scheda video integrata nel Set Top Box: sia le dimensioni che i colori delle schermate non risultano paragonabili qualitativamente a quelle dei monitor, per questo motivo si è reso necessario aumentare le dimensioni di tutti i pulsanti del 30%.

6.2 Esperienze e conoscenze acquisite

Nei mesi di stage mi sono occupata sia della progettazione che dello sviluppo del software HMI e dell'interfaccia GUI, e attualmente sto riscrivendo il software di sincronizzazione utilizzando il linguaggio Java, per rendere il sistema IntelliDomus completamente multipiattaforma.

Ho migliorato le mie capacità sulla programmazione in Java a un livello professionale, grazie alle competenze e alle esperienze che mi hanno trasmesso il Prof. G. Destri e l'Ing. C. Chiodelli, coordinatori del mio ope-

rato. In particolare ho ampliato le mie conoscenze sulla progettazione, la documentazione e l'implementazione di alcuni pattern fondamentali di programmazione (MVC, struttura modulare delle applicazioni e interazione tra componenti software), applicando le competenze acquisite in un contesto concreto.

Inoltre è stata positiva l'esperienza di lavorare all'interno di un team di sviluppo nel quale, operando in parallelo, è fondamentale la comunicazione e la condivisione di metodologie e standard per migliorare la reciproca comprensione delle attività svolte individualmente.

6.3 Sviluppi possibili e scenari futuri

Le prospettive di evoluzione sono rivolte al completamento e all'ampliamento del sistema attraverso l'introduzione di nuovi moduli software, come la funzionalità di videosorveglianza mediante videocamere IP, la gestione degli allarmi mediante tecnologia GSM o l'implementazione degli scenari di utilizzo, che consentiranno di raggruppare e svolgere diverse operazioni al verificarsi di un deteminato evento.

Attualmente il target per questo tipo di applicativi è quello dei sistemi domestici, con un privato dotato di un piccolo budget e che desidera soluzioni personalizzabili a costi ragionevoli. In seguito sarebbe auspicabile la realizzazione di soluzioni di fascia medio/alta per edifici di dimensioni maggiori e utilizzati per scopi professionali (uffici, alberghi, centri commerciali, ecc.), aggiungendo nuove funzionalità o adattando quelle esistenti a contesti particolari, sulla base delle richieste del cliente.

Per quanto riguarda invece l'ambito domestico, si potrebbe pensare di sfruttare le potenzialità del Set Top Box per la realizzazione di una soluzione all-in-one, che consenta di riunire diverse funzionalità (gestione delle utenze, videosorveglianza, media center, accesso web, ecc.) in un unico sistema semplice e intuitivo.

Appendice A

Strumenti software impiegati nel progetto

- Netbeans IDE versione 5.0 ©1995-2006 Sun Microsystems Inc., distribuito sotto licenza SPL.

  `http://www.netbeans.org/`

- J2SE Development Kit (JDK) versione 5.0 Update 6 ©1995-2006 Sun Microsystems Inc..

  `http://java.sun.com/j2se/1.5.0/`

- Poseidon for UML Community Edition versione 4.0.1 ©2005 Gentleware AG.

  `http://www.gentleware.com/`

- MySQL Server versione 4.1 ©2005 MySQL AB, distribuito sotto licenza GPL.

  `http://dev.mysql.com/downloads/mysql/`

- MySQL Administrator versione 1.1.3 ©2005 MySQL AB, distribuito sotto licenza GPL.

  `http://dev.mysql.com/downloads/administrator/`

- MySQL Query Browser versione 1.1.5 ©2004 MySQL AB, distribuito sotto licenza GPL.

`http://dev.mysql.com/downloads/query-browser/`

- MySQL Connector/J versione 3.1.11 ©2005 MySQL AB, distribuito sotto licenza GPL.

  `http://www.mysql.com/products/connector/j/`

- Apache Tomcat versione 5.5 ©1999-2005 The Apache Software Foundation, distribuito sotto licenza Apache License, versione 2.0.

  `http://tomcat.apache.org/`

- Java Modbus Library versione 1.2 ©2000-2004 jamod development team, distribuito sotto licenza BSD.

  `http://jamod.sourceforge.net/`

Bibliografia

[1] P. Atzeni, S. Ceri, S. Paraboschi e R. Torlone. *Basi di dati - Modelli e linguaggi di interrogazione*. McGraw-Hill, 2002.

[2] A. Silberschatz, H. F. Korth e S. Sudarshan. *Database System Concepts*. McGraw-Hill, quinta edizione, 2005.
URL: `http://db-book.com/`.

[3] F. Di Francesco. *Ingegnerizzazione software e sviluppo interfacce per un controllore di Building Automation*, 2004. Tesi di Diploma.

[4] S. Boschi. *Sviluppo di un prototipo per Building Automation*, 2004. Tesi di Diploma.

[5] Design pattern MVC.
URL: `http://ootips.org/mvc-pattern.html`.

[6] Sun Microsystems, Inc. Design pattern MVC.
URL: `http://java.sun.com/blueprints/patterns/MVC.html`.

[7] Object Management Group, Inc. Unified Modeling Language.
URL: `http://www.uml.org/`.

[8] Object Management Group, Inc. Consorzio Object Management Group (OMG).
URL: `http://www.omg.org`.

[9] W. Zuser, S. Biffl, T. Grechenig, e M. Kohle. *Ingegneria del Software con UML e Unified Process*. McGraw-Hill, 2004.

[10] G. Destri. *UML nella progettazione software*, 2003. Slide per il corso di Ingegneria del Software.

[11] C. Chiodelli. *Sistema IntelliDomus IP - Specifiche ed Analisi Funzionale del Software per il Sistema NetMaster + PC*, 2005.

[12] J. F. Kurose e K. W. Ross. *Internet e Reti di Calcolatori*. McGraw-Hill, seconda edizione, 2003.

[13] C. Chiodelli. *Building & Home Automation - Le proposte di CS Soluzioni*, 2004.

[14] E. Gamma, R. Helm, R. Johnson e J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 2004.

[15] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad e M. Stal. *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, volume 1. Wiley & Sons, 1996.