Capitolo 1
UniSim

UniSim è un tool per lo sviluppo, validazione e controllo per progetti d'automazione che mette a disposizione degli utenti i seguenti strumenti: un editor grafico per lo sviluppo dei progetti, un motore di simulazione, un motore di controllo che si interfaccia con i dispositivi di I/O di tipo digitale e un XML validating parser per l'importazione dei progetti. Il tool è basato sulle specifiche IEC 61131-3 che definiscono i linguaggi di programmazione utilizzati per la componente software dei progetti di automazione ed utilizza XML Formats for IEC 61131-3, realizzato da PLCopen, che permette l'interscambio di tutte le informazioni di un progetto (configurazioni, dati e programmi) attraverso gli ambienti di sviluppo.

Grazie a questa caratteristica il software sviluppato e testato con UniSim potrà essere semplicemente importato e messo in esecuzione su ambienti di sviluppo di tipo commerciale, a patto che essi adottino lo XML Schema suddetto.

Inoltre, UniSim consente in corso di simulazione di effettuare la modifica dei programmi, di creare, eliminare o di modificare i task, di passare da una gestione del multitasking di tipo preemptive al tipo non preemptive e di variare la frequenza di campionamento delle variabili.

1.1 Piattaforma e strumenti di sviluppo

Il tool è stato sviluppato usando la piattaforma Microsoft .NET Framework. Questa tecnologia fornisce una serie di compilatori per i principali linguaggi supportati da Microsoft (Visual Basic, Visual C#, Visual J# e Visual C++) ed un ambiente d'esecuzione detto Common Language Runtime (CLR), che richiama in parte la tecnologia Java. Infatti, il codice generato in fase di compilazione è indipendente dall'architettura sulla quale deve essere eseguito. I compilatori per piattaforma .NET Framework generano lo stesso tipo di codice detto Mil (Microsoft Intermediate Language) che viene gestito in fase di esecuzione dal CLR. Esso dispone di un compilatore JIT (Just In Time) che a tempo di esecuzione

traduce il codice nel linguaggio macchina del sistema su cui viene eseguito. Generalmente la traduzione di un'istruzione avviene alla prima esecuzione.

.NET è corredato da una serie di strumenti di sviluppo delle applicazioni, progettati in modo da funzionare in modo integrato all'interno della piattaforma .NET. Uno dei principali strumenti è l'IDE (Integrated Development Environment cioè *Ambiente di sviluppo integrato*), denominato Visual Studio, con il quale è stato sviluppato UniSim. Esso è inoltre un RAD (Rapid Application Development) poiché aumenta la produttività aiutando il programmatore con strumenti come il completamento automatico (IntelliSense) e un designer visuale.

Il linguaggio di programmazione utilizzato è il Visual Basic .NET (VB.NET), derivato dalla versione 6.0 e precedenti, ma rivoluzionato, in quanto implementa in modo completo il paradigma della programmazione ad oggetti, ed utilizza, come tutti gli altri linguaggi della serie .NET, il meccanismo di gestione automatica della memoria (garbage collection).

1.2 Lo standard IEC 61131-3

Come già si è detto in precedenza, UniSim è stato sviluppato seguendo le specifiche dello standard IEC 61131-3, che rappresenta la normativa più diffusa nell'ambito dei controllori a logica programmabile (PLC). Il motivo di tale successo è dovuto principalmente al fatto che esso individua un modello software efficiente ed adattabile a qualsiasi progetto d'automazione. Oltre al SFC che è il linguaggio con cui è possibile creare progetti con il tool, lo standard definisce anche altri quattro linguaggi di programmazione:

- il Function Block Diagram consente di definire facilmente blocchi funzionali o funzioni da riutilizzare all'interno dei programmi;
- il Ladder Diagram si presta all'implementazione di operazioni che coinvolgono i singoli bit;
- Structured Text richiama la programmazione procedurale dei linguaggi di alto livello come il Pascal;
- Instruction List si avvicina molto alla programmazione assembly di più basso livello.

La struttura del modello software definito dallo standard IEC 61131-3 è di tipo gerarchico: al vertice si colloca la *configuration*, che rappresenta l'insieme di tutto il software (ossia i programmi e la configurazione stessa dei dispositivi di controllo), utilizzato da tutti i dispositivi che realizzano il controllo di un processo, poi vi è la *resource*, vale a dire il dispositivo fisico che è in grado di eseguire i programmi. Un PLC monoprocessore può essere associato ad un'unica resource, mentre un PLC multiprocessore ha tante resource quanti sono i suoi processori, così come n PLC monoprocessori collegati da un bus possono essere associati ognuno ad una resource e poiché in una configuration vi possono essere più resource, quest'ultime comunicano tra loro mediante le variabili globali. Una risorsa, infine, contiene compiti (*task*) ed istanze di unità organizzative di programma (Program Organization Unit o *POU*) prive di task. I task, contenenti le POU, possono avere un'esecuzione di tipo periodica o legata al verificarsi di un evento e dispongono di un valore che ne definisce la priorità rispetto agli altri task. Il periodo di esecuzione per i task di tipo ciclico è definito dal relativo attributo Interval, mentre il verificarsi di un evento associato all'esecuzione dei task non ciclici è identificato dal fronte di salita di una variabile booleana associata al task stesso, identificata come Single. Le POU, invece, si riferiscono a programmi(program), o a blocchi funzionali (functional block) o a funzioni (function) definiti attraverso i linguaggi di programmazione specificati dallo standard stesso. La figura di seguito illustra la definizione di una configuration così come è definita dalla normativa:

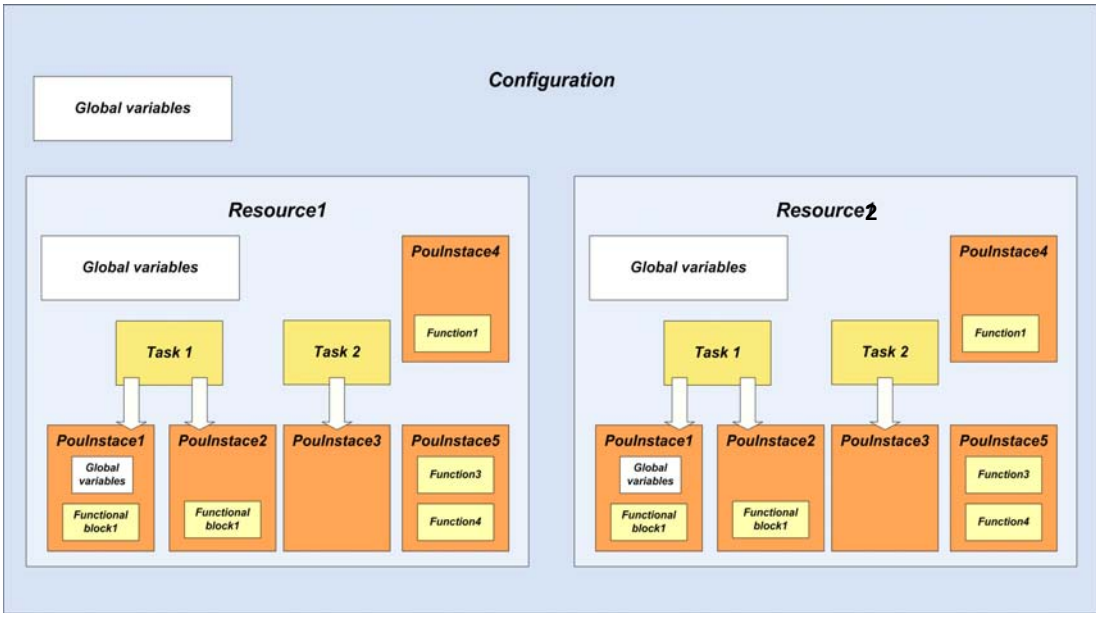

Figura 1 – Definizione di una configuration

1.3 XML Formats for IEC 61131-3

Sebbene lo standard IEC 61131-3 definisca con precisione il modello software da adottare, non stabilisce un'interfaccia per l'interscambio delle informazioni tra gli strumenti che lo utilizzano. Allora l'organizzazione PLCOpen ha strutturato un modello per l'import/export che è rappresentato dal XML Formats for IEC 61131-3. Lo scambio delle informazioni tra i diversi tool avviene attraverso file xml e lo schema definito per tali file è lo stesso per qualunque tipo d'informazione relativa ad un progetto d'automazione, sia esso una libreria di programmi, di funzioni o blocchi funzionali, una definizione di una risorsa, una configurazione di più risorse o un intero progetto.

Lo schema specifica che un *progetto* dovrebbe essere composto dai seguenti quattro elementi: *fileHeader*, che contiene informazioni sulla compagnia produttrice; nome, versione e data di creazione del prodotto contenuto nel file xml; *contentHeader* costituito da informazioni generali sul progetto: nome, autore e versione; *types* che include i tipi di dati (ad esempio bool, int, string, array, enum) utilizzati nel progetto (dataTypes) e le definizioni delle POUS ed, infine, *instances* contenente informazioni sulle resource, sui task e sulla configuration. L'elemento *POU* è costituito a sua volta da altri cinque elementi: *transitions*, *actions*, *documentation*, *interface* che contiene le liste dei vari tipi di

variabili utilizzate ed, infine, *body* che implementa una POU con uno dei cinque linguaggi di programmazione definiti dallo standard. Inoltre, ogni oggetto grafico definito da uno dei linguaggi FBD, LD e SFC è rappresentato da un diverso elemento, contenuto in uno dei seguenti quattro gruppi: *commonObjects*, *fdbObjects*, *IdObjects* ed *sfcObjects*. Il gruppo commonObjects contiene gli elementi che definiscono gli oggetti comuni ai tre linguaggi grafici e quello più rilevante è l'elemento *actionBlock* che rappresenta un blocco di azioni all'interno di un body. Un blocco di azioni è definito, oltre che dagli elementi grafici, da una lista d'azioni (action). Il gruppo sfcObjects, invece, è costituito dagli elementi che definiscono gli oggetti che contraddistinguono il linguaggio: step (fase), macroStep (macrofase), jumpSteps (fase di salto), transition (transizione), selectionDivergence (divergenza selettiva), selectionConvergence (convergenza selettiva), simultaneousDivergence (divergenza simultanea) e simultaneousConvergenge (convergenza simultanea). In figura si riporta la struttura generale definita dallo schema del XML Formats for IEC 61131-3:

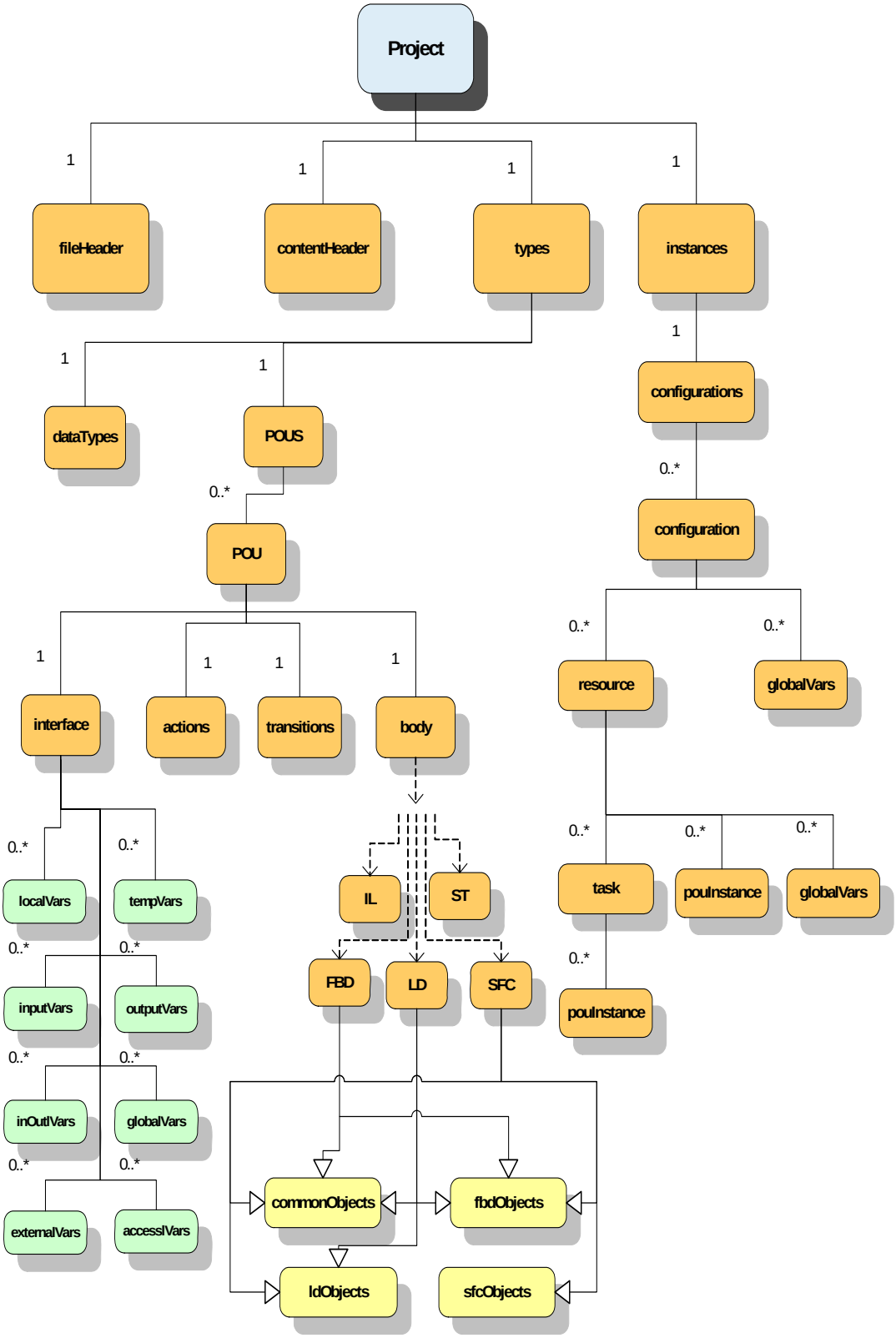

Figura 2 – Struttura generale definita dallo schema

1.4 Architettura generale

L'architettura del tool rispecchia il modello software definito dallo standard IEC 61131-3 e ripreso dal XML Formats for IEC 61131-3. Inoltre dal punto di vista dell'utente, UniSim è suddiviso in due applicazioni indipendenti: la prima per l'editing, la simulazione e l'import/export dei progetti e la seconda per il controllo dei processi reali non critici. Il tool, rispettando il paradigma object-oriented, consta di diverse classi divise in tre gruppi:

- InteractionGroup, contenente le classi per l'implementazione dell'interfaccia grafica;
- ProjectGroup, contenente le classi per l'implementazione dell'intero progetto e dell'interfaccia di import/export. A sua volta, tale gruppo è composto dai seguenti sottogruppi:

- ProjectInfoGroup, contenente le classi per la memorizzazione delle informazioni relative al progetto;
- TypeGroup, contenente le classi per l'implementazione dei tipi di dati e le unità organizzative di programma (POU);
- InstancesGroup, contenente le classi che implementano gli oggetti dei livelli più alti del modello software definito dallo standard.

- PLCGroup, contenente le classi per l'implementazione del motore di simulazione, del motore di controllo e dell'interfaccia verso i dispositivi di I/O.

Tutte le classi sono contenute in un'unica libreria (UniSimClassLibrary) e sono utilizzate dai due moduli eseguibili delle due applicazioni viste prima. Per cui, ognuno di questi moduli implementa solamente le rispettive interfacce utente. Il diagramma sottostante mostra la situazione:

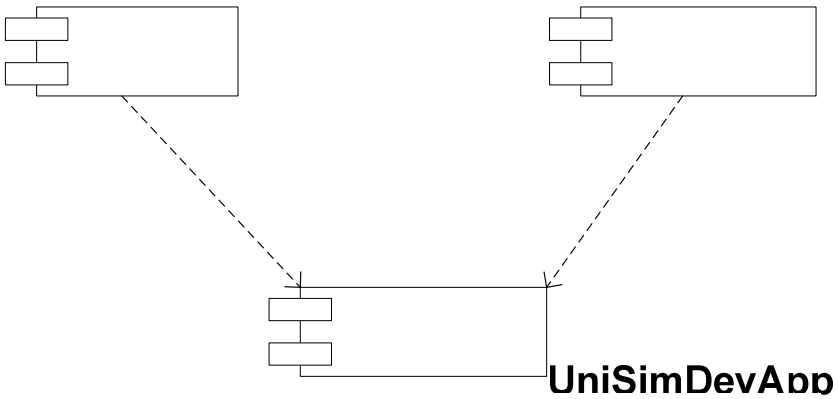

Diagramma dei componenti

UniSim