International Workshop on
Development and Evolution of Software Architectures for Product Families

position paper

Configurable Software Modules for Families of PLC-based Systems

Flavio Bonfatti [1], Gianni Gadda [2], Paola Daniela Monari [1]

[1] Department of Engineering Sciences, University of Modena
via Campi 213/B, 41100 Modena (Italy)
tel +39 59 378514 fax +39 59 378515 email bonfatti@unimo.it

[2] DemoCenter scrl
viale Virgilio 55, 41100 Modena (Italy)
tel +39 59 848810 fax +39 59 848630 email g.gadda@democenter.it

The market of automated systems based on Programmable Logic Controllers (PLCs) is steadily growing and evolving towards higher levels of product customization. In Europe, and particularly in Italy, most developers are small-medium specialized enterprises (SMEs) which often apply sub-contracting. This means flexibility in facing a market demand that changes rapidly in size and product types, but implies a number of problems arising from uncompleteness and ambiguity of the exchanged information, uneven quality levels, difficulties in taking advantage of previous experiences.

A PLC-based system is made of four main components: mechanical (carpentry, belts, gears), electrical (power supply, engines, sensors), electronic (the control unit, other computers) and software. The system design is split into two main phases: the structural design, with the definition of mechanical and electrical components, and the control design, with the definition of electronic and software components. The latter derives requirements from the former: the system behaviour is decided by the electro-mechanical project team who interprets customer requests.

Many difficulties arise from this situation. Since the two design phases are carried out by project teams with different cultures, a common language is hard to define, so that information lacks and misunderstandings frequently occur. Moreover, the two phases are strictly sequential because of the subordinate role of the control team with respect to the structural team, and this lengthens the design time. Finally, design errors and incompletenesses deriving from the cited difficulties require a number of revisions and iterations of the design process that increase production times and costs.

Further problems make design and realization of PLC-based plants particularly complex. First, these are often one-of-a-kind systems, even though based on similar components. Unless parametrized solution are identified, most of the control functions are developed ex novo or substantially revised every time. Moreover, it happens that different

customers require similar systems, but employing different control units: thus, the same control software must be generated and maintained in different languages. Finally, because of the low level of PLC languages, heavy test sessions are required to assure the needed system safety degree, in particular when sub-contractors are involved in the project. Some design and development defects are detected only during the final test on the field, with negative consequences on relations with the customer.

A significant enhancement of PLC software design and development practice is presently under study. It adapts and integrates two techniques resulting from European Union funded projects: the ESPRIT / CIME project 8224 - RUMS, and the ESPRIT / ESSI project 10542 - EASIER. In short, the proposed approach:

- Provides the structural designer with a modelling tool coming from the RUMS project, conceived to obtain a rule-based, compact and parametrized product definition that relates functional and structural aspects. Similarities and peculiarities of the different system versions are completely captured and represented according to a proper family-based model.

- Provides the software designer with an object-oriented language, derived from the EASIER project, that decomposes the plant control into parts at different composition levels, and assigns to every part the description of its (possibly parametrized) behaviour. This knowledge is expressed in form of declarations and rules, easily readable by all the involved personnel and suitable to support completeness and consistency controls.

The result is a control software package for the whole system family, adaptable to the single version by setting the parameter configuration that identifies it.

Concerning software modularity, the proposed practice conceives the system control as a set of weakly coupled software modules, each managing one of the system component units. In other words, the resulting control structure should mirror the system physical structure so as to substantially improve the reuse potential of its parts in other systems. The control code of complex units is obtained by a specific composition method. Unit interactions are expressed as synchronizations of unit behaviours, and rely on the definition of proper interface variables and information exchange rules.

A bedded encapsulation mechanism explicitly relates the complex unit behaviour to that of its components. The encapsulation approach relies on the following two basic rules:

- a software control module (no matter if simple or complex) is not allowed to access variables of the modules operating at the same level or at higher levels;

- a complex control module is only allowed to access interface variables of its direct components, that is, the modules operating at the immediate lower level.

These rules are mainly aimed at limiting and disciplining the use of global variables, as they constitute the most dangerous obstacle to software reuse. The former rule establishes that the single module has to be designed in such a way to be unaware of the other (external) modules, including those interacting with it or using it as a component. If

this condition is not met, the module functionality results somehow dependent on the context where it has been developed, and this could prevent from using it in different contexts. The latter rule explains where and how the interactions and dependences between the two components, and with the upper level control module, should be expressed. Limiting the access to the modules at the immediate lower level prevents these from missing the control of their own components.

Observe that this bedded encapsulation approach applies recursively to upper level compound parts. This means that the simple control modules, the compound module and those of upper level are all viewed as reusable software packages. Depending on which of the system physical parts (either elementary or compound) are employed elsewhere, the corresponding control module can be reused. Its interactions with the other parts of the new plant will be a task of the higher level module including it.

The EASIER experience showed that an important step forward the introduction of a better software engineering practice consists in coding once for all the controls of the most common component units and the most frequent interactions between them. In fact, although complex systems differ significantly from each other, they often share combinations of a number of basic functional units. The ongoing work is carried out within the ESPRIT / IiM project 22273 - AUTOMAT (Methodology and Set of Tools for the Improvement of Machine Tool Automation). This project pursues the general objectives recalled above, but focusing on a particular class of PLC-based systems: CNC machine tools and flexible manufacturing systems. In spite of this restriction, the composition mechanism of reusable modules is widely studied, taking also into account the need of mapping in onto the IEC1131-3 standard languages.