Computers in Industry 70 (2015) 46–55

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

A new object model of batch equipment and procedural control for better recipe reuse

Giovanni Godena [a,d,*], Tomaž Lukman [a,1], Igor Steiner [b,2], Franc Bergant [c,3], Stanko Strmčnik [a,4]

[a] Jožef Stefan Institute, Department of Systems and Control, Jamova 39, SI-1000 Ljubljana, Slovenia
[b] Inea d.o.o. Stegne 11, SI-1000 Ljubljana, Slovenia
[c] Helios, Tovarna barv, lakov in umetnih smol Količevo, d.o.o. Količevo 65, SI-1230 Domžale, Slovenia
[d] Jožef Stefan International Postgraduate School, Jamova 39, SI-1000 Ljubljana, Slovenia

ARTICLE INFO

Article history:
Received 9 May 2014
Received in revised form 6 February 2015
Accepted 11 February 2015
Available online 11 March 2015

Keywords:
Batch control
Recipes reuse
Information repetition
Equipment object model
Overlapping unit classes

ABSTRACT

Application of the ISA-88 standard in industrial batch-process control often leads to repetition of information in recipes and to a low level of their reuse. This problem stems from the deficiencies of the standard batch-process control object model. A solution to the problem is proposed that is based on a more sophisticated object model of equipment and procedural control, with dynamically defined and potentially overlapping unit classes. The new concept, together with its elements, is described, and its use is illustrated and validated by means of a real batch control project. The validation is carried out as a comparison of the number of master recipes and unit procedures created under the new object model with the number of master recipes and unit procedures needed to perform the same functionality using the standard object model. The comparison demonstrates that the proposed approach has a significant advantage.

1. Introduction

There is broad consensus among the manufacturers of automation equipment, service providers, and end users of the need to comply with the ISA-88 standard [1] in batch-process control, regardless of the level of automation. The reason lies in the many attractive features of this standard, e.g., it represents a common language for engineering and operations management; it is a standard that reduces the engineering cost for automation; it can be used to identify the appropriate level of automation, etc. [2]. Furthermore, it is compatible with the ISA S95 standard, and as such represents an important framework for design and implementation of control, optimization, and decision functions on various hierarchical levels of the enterprise (see e.g., [3,4]). However, there are also certain problems concerning the application of the ISA-88 standard and the batch tools that are based on it, see e.g., [5].

One of the problems encountered in real life applications of batch control is the high degree of repetition of information in the recipes and their low degree of reuse, resulting in the need for increased effort regarding recipe management and in a higher probability of errors in the recipes. This problem stems, in our opinion, from the deficiencies of the standard batch-process control object model, caused by its fixed structure of unit classes and (too) simple class hierarchy.

Surprisingly, the above-mentioned problem is not recognized in the technical and scientific literature. An extensive literature search in several databases, including INSPEC, Science Direct, and Google Scholar, did not turn up any paper dealing with this problem. The search was performed using various combinations of the following keywords: "batch", "process", "control", "recipe", "reuse", "repetition of information", and "reusability".

In our opinion, a proper solution to the problem is the introduction of a more sophisticated object model of equipment and procedural control. A step toward a more sophisticated object model has already been made in Part 2 of the ISA-88 standard, [6] which introduces a more complex kind of class hierarchy in the form of multiple inheritance in the units object model by allowing an equipment entity to belong to more equipment classes.

* Corresponding author: Tel.: +386 1 477 37 59; fax: +386 1 477 39 94.
E-mail addresses: giovanni.godena@ijs.si (G. Godena), tomaz.lukman@ijs.si (T. Lukman), igor.steiner@inea.si (I. Steiner), franc.bergant@helios.si (F. Bergant), stanko.strmcnik@ijs.si (S. Strmčnik).
[1] Tel.: +386 1 477 37 30; fax: +386 1 477 39 94.
[2] Tel.: +386 1 513 81 17; fax +386 1 513 81 70.
[3] Tel.: +386 1 3629 350; fax: +386 1 3625 991.
[4] Tel.: +386 1 477 35 76; fax: +386 1 477 39 94.

http://dx.doi.org/10.1016/j.compind.2015.02.002
0166-3615/

However, this concept concerns assigning different roles to a particular unit, such as a vessel being both a reactor and a holding tank, and has no impact on the repetition of information in the recipes and their reuse.

The aim of this paper is to present a new proposal for the object model of batch equipment and procedural control that to a large extent minimizes the repetition of information and maximizes the reuse of recipes. The proposal is based on a concept involving dynamically defined and potentially overlapping unit classes or a multiple inheritance relationship between the units and the unit classes, i.e., on the possibility of a certain unit belonging to more than one unit class.

The paper is structured in the following manner. Section 2 introduces object technology as a means of information-repetition prevention and reuse enhancement in the domain of batch-process control. Section 3 describes the ISA-88 standard object model of equipment and procedural control as it is used in most batch control tools and comments on its deficiencies. Section 4 presents a new proposal for the object model of equipment and procedural control. Section 5 briefly describes the implementation of the new concept in an in-house-developed batch tool and illustrates the advantages of the new approach in a real-life industrial application. Section 6 presents the validation of the proposed concept and Section 7 the conclusions.

2. Object technology in batch-process control

Object technology has enabled a significant positive shift in the general process of software development. Two of the positive effects of introducing object technology that are the most relevant for the research presented in this paper are enhancing reuse [7,8] and reducing information repetition [9]. This should also hold true for batch-process control tools in accordance with the ISA-88 standard, i.e., the introduction of object models for equipment and procedural control should prevent the repetition of information in the recipes and enable their reuse.

At this point, some important terms and concepts of the ISA-88 standard should be introduced. The standard describes batch control by means of three models, namely the process model, physical model, and procedural control model. The process model describes the hierarchical subdivision of a batch process in terms of entities relevant from a process engineering perspective, namely processes, process stages, process operations, and process actions. The physical model describes the hierarchy of physical assets of an enterprise in terms of enterprises, sites, areas, process cells, units, equipment modules, and control modules. The procedural control model describes (equipment) procedural elements that are combined in a hierarchical manner to accomplish the task of a complete process as defined by the process model. The hierarchy consists of the procedure level (consisting of a sequential-concurrent combination of unit procedures), the unit procedure level (consisting of a sequential-concurrent combination of operations), and the operation level (consisting of a sequential-concurrent combination of phases). The entities of the procedural control model in fact represent the processing capabilities of the entities of the equipment model, where phase is the smallest processing entity. The entities of the three models have the following relation: the entities from the procedural control model, combined with the entities from the physical model, provide process functionality to carry out entities from the process model. Besides the three described models, there are also recipes with their recipe procedures, which provide a hierarchical-sequential-concurrent decomposition of the processing required to produce a certain product. The elements of a recipe procedure are in fact references to the elements of equipment procedural control. Thus, for example, at the lowest level of a recipe procedure, a recipe phase is an abstract notation giving a reference (link) to the corresponding equipment phase. There can be many recipe phases for a given equipment phase, as the processing of an equipment phase may be used by many recipe procedures. For more detailed information on the above-described concepts, see Sections 5.1, 5.2, and 5.3. of [1]. Detailed information on the recipe procedure/equipment procedural control linking can be found in [1], Section 5.3.3.3.

The equipment and procedural control models implemented in various tools compliant with the ISA-88 standard are based on classes and instances of equipment, and on unit procedures that can be written for a class or an instance of an equipment unit. The basis of unit classification is its capability, i.e., the set of phases of a particular unit. A unit class represents units of the same capability, i.e., the same set of phases. A unit procedure for a unit class is written only once for the unit class and may be executed on any unit of that class. A unit class can also represent units that are not completely identical. In that case, the unit class only has the phase classes whose instances can be found in all the units of that class. The other phases appear as specific phases of particular units. A unit procedure written for a unit class can only contain the common phases. If we would like to use some of the specific phases in the recipe, the unit procedure has to be written for a specific unit.

The above-described model works fine in simple cases, in particular where the equipment consists of unit classes with identical unit instances, without specific phases. On the other hand, in more complicated cases with not fully homogeneous unit classes (unit instances having specific phases), which in our experience is quite common in industry, the model fails to avoid the duplication of information in the recipes and does not allow the maximization of their reuse. In these cases, a model based on multiple inheritance, instead of single inheritance, would give much better results.

Inheritance is one of the most prominent concepts of object-oriented technology and the key mechanism that facilitates reuse [10,11] and reduces redundancy, i.e., the repetition of information [9]. Inheritance should be applied to model commonality and specialization [12,13]; however, it can also be used to model kind-of roles, transactions, and devices [14].

Nevertheless, its use must be carefully applied since it can be dangerous when used incorrectly [15]. It is effectively just a mechanism for extending an application's functionality by reusing the functionally of the parent classes [13]. Inheritance can be categorized into two distinct types, i.e., single inheritance, where a child class inherits from only one parent class, and multiple inheritance, where a child class inherits from two or more parent classes. Multiple inheritance is more powerful and expressive than single inheritance [16]. It also increases reuse (because of multiple sub-classing) [13] and allows a more natural definition of the relationships between classes [17]. The implementation and use of multiple inheritance is non-trivial [18], since it introduces several potential problems, e.g., repeated inheritance [19]. Consequently, multiple inheritance is only used in some of the available object-oriented programming languages (e.g., C++ and Eiffel). Whether multiple inheritance simplifies or complicates the object-oriented technology remains a matter of debate [16]. However, the benefits of multiple inheritance have outweighed its drawbacks in some specific domains where it is used in object-oriented modeling. For instance, the use of multiple inheritance in geographic data modeling is essential [20,21].

Despite the controversy, we believe that multiple inheritance is the right mechanism for addressing the problem of the high repetition of information in recipes and their low reuse in the domain of batch-process control. In the remainder of this paper we present an approach to solving the above-mentioned problem. The approach is based on the definition of a new object model of