Batch Processing in a Wider Perspective

Charlotta Johnsson
Product Manager Cube-XBatch
Orsi-Group
Corso Europa, 799
16148 Genoa
Italy
+39 010 3434 515
+39 010 380 309
johnsson@it.orsigroup.com

KEY WORDS
Synchronization, Coordination, Component-Based Architecture

ABSTRACT
With the “new economy” comes new requirements on all manufacturing and processing activities, including batch processing. The entire chain of activities – from order placement to delivery – must work seamlessly. This implies that batch processing must be dealt with in a wider concept. In batch systems of today there are generally no means for synchronization or coordination of activities outside the scope of the batch. The solution presented in this paper is to have a flexible but yet structured component-based system where the batch-processing system will constitute one part – nevertheless an important part - of the entire system. The entire system corresponds to a MES system, which can be structured according to the standard ISA S95. A framework is used to synchronize and coordinate the activities of the different components.

The execution of a batch is handled by the batch-execution-component, which is S88 compliant. The batch execution component can be synchronized with other components at the stop and start of a batch but also during the execution in a sequential or parallel way. Examples are given in the paper.

The benefits of this approach are vital in the “new economy”, where quicker “time-to-market” and “production-agility” become even more important. A system with well designed components and a flexible and user-friendly framework will shorten the project design time, the project implementation
time, and the required time for introducing a new product on the market. By having a well-organized MES system in which the execution of the different cells can be synchronized, the production capacity of the area can increase.

**INTRODUCTION**

With the “new economy” come new requirements on all manufacturing and processing activities, including batch processing. The entire chain of activities – from order placement to delivery – must work seamlessly. This is important in order to keep pace with the accelerating speed of business and thereby e.g., assuring a closer delivery-date, and reducing warehouse stocks. This implies that batch processing must be dealt with in a wider concept than it is today. An entire batch solution must be responsible not only for executing a planned sequence of operations but also for the wider repercussion of its actions. In batch systems of today there are generally no means for synchronization or coordination of activities outside the scope of the batch. There are of course many solutions to this problem. The most trivial one would be to simply enlarge the scope of traditional batch execution systems. Another solution would be to have a component-based architecture where the different components are responsible for one well-defined task or group of tasks, and to have a framework that coordinates and synchronizes the activities of the different components. In the latter solution alternative, the traditional batch execution system would constitute one (1) of the many components included in a complete MES solution. This article will present the latter solution alternative.

**COMPONENT-BASED ARCHITECTURE**

Component-based Software Engineering is the discipline of developing components and developing products with components. A component can be seen as a collection of related functionality/tasks. Products are no longer developed from scratch; they are instead an assembly of components developed independently of the product and of each other. This means that components are developed without complete knowledge about their execution environment. To assemble components, proprietary code that joins the components is usually needed. This code, often referred to as the “glue code”, may take longer time to develop than the components concerned [1].

Object-oriented programming had the same approach; objects were reusable entities that could be assembled as programs. Component-based development can be seen as an extension of object-orientation, but goes one step further. Object-oriented programming binds the implementation to a particular class library and language. Components on the other hand, are generally not bound to particular language and they communicate through independent means [1]. Today, there are three major component models that are used successfully: COM [2], JavaBeans [3] and CORBA [4].

The solution presented in this paper uses the component model of COM and its “glue code” is a framework. The COM components expose their capabilities in terms of methods and events [2]. The functionality of the components can be developed in-house or can be a third-party component encapsulated by a COM-wrapper. The framework is used to coordinate and synchronize the methods of the different components and to react upon their events. A component with its methods and events is shown in Figure 1.
The components needed in a MES-application focused upon batch processing are of course customer dependent and changes from application to application. However, some fundamental examples are; ERP integration component, Batch execution component, Recipe manager component, Electronic Batch recorder component, Scheduler, etc.

**FRAMEWORK**

The task of the framework is to coordinate and synchronize the functionality exposed by the components, and to show this in an attractive an easily understandable way. This is done through a graphical language. The operator simply specifies the sequential order in which he/she would like the functionality of the components to be executed. An example of such a rule, referred to as a production operation, is shown in Figure 2. The production operation should be read from left to right. The first graphical element indicates the start of the rule and this element can be associated to an event. The second graphical element, which is the graphical element most frequently used in a production operation, is an element that calls a method of a component. It is referred to as method-caller. The production operation in Figure 2 contains one parallel and one alternative branch.
The operator can get information about the current execution status in the plant. The execution of a production operation is color-coded which means that the operator easily can see which steps have been executed and which step is currently executing.

**BATCH COMPONENTS**

The three main important components in a S88 compliant batch solution are the recipe definition component, the batch execution component and the batch recording component.

The recipe definition component allows master recipes to be defined. This includes specifying a header, creating the procedure (graphical description), and specifying the formula and the equipment requirements, all according to the ISA S88 standard. An example of a procedure is shown in Figure 3. The procedure is composed of 4 unit-procedures, each unit-procedure is composed of a number of operations and each operation is composed by phases (not shown in the figure).

The batch execution engine assures the execution, i.e., it creates and starts the execution of control recipes, and it controls the execution and the interaction with the field. The execution of a batch is initiated through a call to the batch execution component from the production operation. In order for the operator to get a good overview of the current executions in the plant, he/she can look at the color-coded procedure of the recipe to see the current execution status of a recipe.

The batch-recording component logs the execution of the batch, i.e., it logs the start and stop time of all phases/operations/unit-procedures in the recipe as well as the actual values of the associated parameters. The batch-recording component generates “a classical batch report”. The batch recording component is linked to the batch execution component.

**Figure 3: A recipe with its procedure. The procedure consists of four unit-procedures**
It is also possible to manage batches without the recipe definition component, the batch execution component and the batch recording component. It is possible to create components that corresponds to the physical entities in the plant and to let these components expose their methods, e.g., a mixer component can expose methods for charging, mixing, and discharging etc. Using this approach, the recipes are not structured in a SFC-like manner, instead the “recipes” are defined as production operations in the framework.

SYNCHRONIZATION WITH THE BATCH COMPONENT

There are three main ways in which the framework can interact with the batch components and thereby synchronize the execution of the batch component with that of other components. The three ways are; start a recipe, monitor the execution of a recipe, and detailed recipe synchronization.

1.1 Start a recipe

Start a recipe is the simplest synchronization between the batch component and the framework. From a method-caller within the production operation, the “start-recipe” method of the batch component is called. The production operation waits for the batch component to complete the recipe execution before it continues executing the following step in the production operation.

An example is given in Figure 4. The batch execution component is called from the first method-caller in the production operation. A control recipe is created and started. When the execution of the control recipe has completed, the execution of the production operation continues and calls for some reporting to be performed, see the second method-caller in the production operation in Figure 4.

![Figure 4: Start a recipe from a production operation.](image)

1.2 Monitor the recipe execution

From the framework it is possible to monitor the execution of a recipe. This feature allows synchronizing with the batch execution without interfering with it. An example is given in Figure 5.
The production operation is triggered by some event. The recipe execution component is called from the first method-caller in the upper parallel branch. A control recipe, see Figure 3, is created and started. A monitoring element, monitoring the second unit-procedure (mixingA) of the recipe, is present in the lower parallel branch of the production operation. This element does nothing but waits for the execution of the second unit-procedure (mixingA) in the recipe to complete. Upon completion, the execution of the lower parallel branch continues and calls for some analysis to be done. When the execution of the entire recipe has completed, the upper parallel branch of the production operation continues and calls for some reporting to be done. The report could e.g., be written in XML format and then sent directly to the relevant recipient (a person or a machine).

The benefit of the monitoring feature is that it allows actions to be performed at the right point in time and without any delays. This could help increasing e.g., the throughput of the plant.

1.3 Detailed recipe synchronization

From the framework it is possible to interfere with the execution of a recipe and to synchronize it with the execution of other components. An example of this is given in Figure 6.

The production operation is triggered by an event. The batch execution component is called from the first method-caller in the upper parallel branch in the production operation. A control recipe, e.g. the one shown in Figure 3, is created and started.

From the batch execution component an event is sent to the framework at a certain point in the process. The event could e.g., correspond to the need of having some off-line tests performed on the intermediate product. The execution in the batch component pauses. The production operation catches the event (first graphical element in the lower parallel branch). The off-line testing tool or person is notified (second graphical element in the lower parallel branch). When the test results are ready, the results are passed down to the batch component and the execution in the batch component resumes (third graphical element in the lower parallel branch). When the execution of the entire recipe has completed, the upper parallel branch of the production operation continues and calls for some reporting to be done. The
report could e.g., be written in XML format and then sent directly to the relevant recipient (a person or a machine).

A MES SCENARIO WITHIN BATCH PROCESSING

Below follows an example of a combined MES and batch scenario. The terminology used complies with the terminology of the ISA S95.03-draft3 standard.

A production schedule (also referred to as production plan) is sent from the ERP system down to the MES framework used in one area within an enterprise. The arrival of the file triggers a production operation at the area level to start. The production operation calls the scheduling-component that divides the production schedule into one or many detailed schedules (also referred to as local production plans). The detailed schedules, each one corresponding to the execution of one production operation at the cell level, are distributed to the dispatching-component of the relevant cells. The operator at the cell level sees the list of production operations to be executed, and he/she can use the dispatching-component to manually decide when to dispatch a production operation. The production to be performed in the cell can be discrete, continuous or batch. In case of batch processing the production operation at the cell can correspond to any of the production operations presented in Figure 4, Figure 5 or Figure 6. The production operation at the cell level is used to coordinate the execution of the batch component with the execution of other components within the cell (as described earlier). This can be done in a parallel way or in a serial way, see Figure 2. When the work scheduled for one process cell is completed, a notification is sent back to the production operation at the area level. In this way the area production operation can synchronize the work performed by the different cells. Finally, when the entire production schedule has completed, a notification as well as a report of the production performance can be sent back to the ERP system. During the execution of a production operation, logging is performed (by the production-operation-recorder and/or by the batch-recorder component), and location and status of materials are updated and stored (by the material-manager), etc.

Figure 6: Detailed synchronization of the recipe execution.
BENEFITS

The benefits of having a system implemented with a component-based architecture are:

- reduce project design time and implementation time
  The implementation and testing of the functionality of the components can be done separately and do not have to be part of a project, this decreases the project implementation time. The project designer and/or implementer can focus solely on the functionality of the components and do not have to bother about its implementation. This helps decrease the design time. In addition to this, the graphical language in the framework makes it easy for the designer and/or the implementer to quickly design and implement the desired behavior of the plant.

- possibility to easily encapsulate legacy systems
  Since the framework works with COM technology, any system that can be encapsulated by a COM-wrapper can successfully be included in an application.

- increase reusability of software
  The functionality of the components is not designed to suite one particular application, instead they are generic and they can therefore be reused.

- increase through-put (plant capacity)
  The framework with its graphical language makes it very easy to synchronize and coordinate the execution in the plant, and to optimize the use of the plant equipment in order to increase the throughputs.

SUMMARY

In the new economy of today, it has become even more important to synchronize and coordinate your production within your plant. This implies that batch processing must be dealt with in a wider concept than it is today. An entire batch solution must be responsible not only for executing a planned sequence of operations but also for the wider repercussions of its actions. In batch systems of today there are generally no means for synchronization or coordination of activities outside the traditional scope of the batch. One possible solution to this problem is to have a component-based architecture where the different components are responsible for one well-defined task or group of tasks, and to have a framework that coordinates and synchronizes the activities of the different components. The traditional batch execution system will constitute one (1) of the many components included in the solution. The batch component can be made ISA S88 compliant whereas the entire system, being a MES system, could align with the ISA S95 standard.

There are several benefits with this approach, both for end-users and for suppliers. The supplier can produce small self-standing components that can be tested and documented separately. Components increase the reusability of software code. By adding desired components or leaving out undesired one, the end-user has a good opportunity in getting exactly the type of solution desired. The component philosophy will also help decreasing the project design time and the project implementation time.
REFERENCES


