Presented at the IFAC World Congress, San Francisco, 1996

BATCH RECIPE STRUCTURING USING HIGH-LEVEL GRAFCHART

Charlotta Johnsson and Karl-Erik Årzén

Department of Automatic Control,
Lund Institute of Technology,
Box 118, S-221 00 Lund, Sweden,
E-mail: (lotta,karlerik)@control.lth.se

Abstract: The paper presents how object-oriented sequential function charts can be used to implement a batch control recipe management system. Grafchart, a toolbox based on Grafcet, object-oriented programming ideas and High-level Petri nets, is presented. It is shown how the concepts of H-L Grafchart fit the models defined in the forthcoming ISA batch control standard SP88. A simulated batch process cell scenario is presented together with alternative ways of representing control recipes.

Keywords: Batch Control, Discrete Event Dynamic Systems, Sequential control, Petri nets

1. INTRODUCTION

Grafcet, or Sequential Function Charts (SFC), has been widely accepted in industry as a representation format for sequential control logic at the local PLC level through the standards IEC 848 and IEC 1131-3. However, there is also a need for a common representation format for the sequential elements at the supervisory control level. Grafchart is the name of a toolbox based on Grafcet that is aimed at supervisory control applications, Årzén (1994). It is implemented in G2, an object-oriented graphical programming environment developed for supervisory applications, Moore *et al.* (1990). High-Level Grafchart is an extension of Grafchart that combines the graphical language of Grafcet/SFC with high-level and object-oriented programming language constructs and ideas from High-Level Petri Nets, Jensen and Rozenberg (1991). High-Level Grafchart is described in detail in the companion paper, Årzén (1996).

Representation of operation sequences is an important part in batch control. Batch control is currently the subject of large interest. The forthcoming ISA standard SP88 is an important step towards a formal definition of the terminology, models and functionality of batch control systems for multi-product, network structured batch processing cells. SP88 mentions the possibility to use Grafcet but only at the very lowest level for representing recipe phases. This paper shows how the concepts in High-Level Grafchart can be used to structure an entire recipe system including all levels in the procedure hierarchy. The object oriented features of High-Level Grafchart allow a clean separation between process equipment objects and batch recipes. The multi-dimensional possibilities of High-Level Grafchart allow a well-defined division between resource allocation and recipe execution.

Grafchart and High-Level Grafchart are briefly described in Section 2. The SP88 standard is presented in Section 3. A simulated batch processing cell scenario implemented in G2 is described in Section 4. This will be used as an example in Section 5 where different alternatives are given for how batch recipes could be represented in High-Level Grafchart.

2. HIGH-LEVEL GRAFCHART

Grafchart contains steps, transitions and alternative and parallel paths according to Grafcet, see Fig. 1. It differs from Grafcet with respect to how step actions are represented. In Grafchart, step actions are rep-

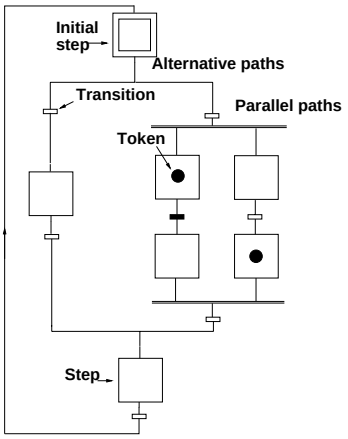

Fig. 1 Grafchart graphical syntax

resented as G2 rules. The actions may be conditional or unconditional and of four basic types: always, initially, finally, and abortive. Always actions are executed periodically while the step is active. Initially actions are executed once when the step becomes active. Finally actions are executed once immediately before the step becomes deactivated and abortive actions are executed once immediately before the step is aborted. To each transition a condition or an event is associated. The transition is enabled when the preceding step is active. When the transition condition becomes true or the transition event occurs, the transition is fired. This means that the preceding step is deactivated and the succeeding step is activated.

Grafchart contains three hierarchical abstraction mechanisms: *macro steps, procedure steps* and *process steps*, shown in Fig. 2. Macro steps are used to represent steps that have an internal structure. Sequences that are executed in more than one place in a function chart can be represented as Grafchart procedures. The call to a procedure is represented by a procedure step. The procedure step contains a `procedure` attribute that contains the name of the procedure that should be called. The Grafchart procedures are reentrant and recursive procedure calls are possible. A process step is similar to a procedure step. The difference is that the procedure is started as a separate execution thread, i.e., as a process. Consequently, the transitions after a process step become fireable as soon as the execution has started in the Grafchart procedure whereas the transitions succeeding a macro step or a procedure step do not become fireable until the entire sequence has been executed. An outlined circle token is shown in the process step as long as the process is executing.

High-Level Grafchart is an extension to Grafchart that is currently being implemented. It adds four

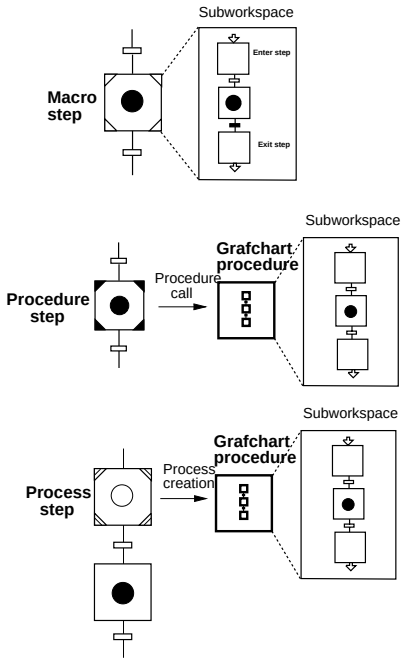

Fig. 2 Macro step, procedure step and process step.

features to Grafchart: parameterization, methods and message passing, object tokens, and multi-dimensional charts. Parameterization denotes the possibility for entire charts and macro steps to have parameters that can be referenced from within steps and transitions. It is also possible for Grafchart procedures to have parameters. The parameters can be referenced from within the procedure body using the dot notation `sup.<parameter-name>`. The actual values of the procedure parameters are set in the procedure step or process step. The `parameters` attribute of a procedure step contains a list of assignments to the formal parameters of the procedure being called. The assigned values can either be constants (numbers, strings or symbols) or the value of a parameter that is visible in the procedure steps context. Using the latter form, it is also possible for a procedure to return values to the procedure step. Similarly, it is possible to determine which procedure that should be called from the value of a parameter.

Methods and message passing is supported by allowing Grafchart procedures to be methods of general G2 objects. For example, an object representing a batch reactor can have Grafchart methods for charging, discharging, agitating, heating etc. Inside the method body, it is possible to reference the object itself and the attributes of this object using the `self` and `self.attribute` notation. The method of an object is called through a procedure or process step in the same way as if the procedure was standalone. Instead of

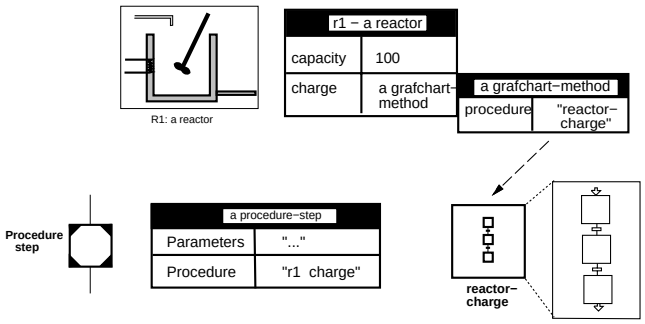

Fig. 3 Grafchart methods.

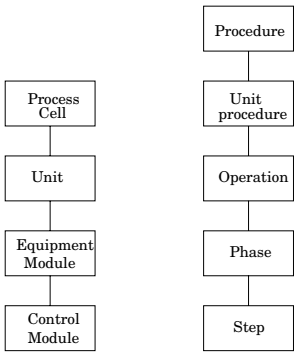

Fig. 4 Physical model and Procedural model

giving a procedure reference, the procedure that will be called is determined by an object reference and a method reference. An example of a Grafchart method is shown in Fig. 3. The reactor object `R1` contains the method `charge`. The method is implemented by the Grafchart procedure `reactor-charge`. The procedure step invokes the `charge` method of the `R1` object.

The object token extension is based on ideas from High-Level Petri Nets, Jensen and Rozenberg (1991). Here, a token is no longer simply a boolean indicator that tells whether a step is active or not. Instead, a token is an abstract data type or object whose attributes can be referenced from within the step that the token is placed in and from the transitions that are enabled by the token. A step may contain multiple tokens that, possibly, are of different type.

Since the tokens are objects they may also have Grafchart methods. This gives a multi-dimensional chart structure where the tokens themselves contain function charts. As will be shown in Section 5, this possibility is very useful for representing recipes.

3. SP88

The ISA batch control standard SP88 aims at standardizing the terminology, models and functionality of batch control systems. The *physical model* of SP88 defines the hierarchical relationships between the physical assets involved in batch manufacturing. The model has seven levels. Here we will focus on the lower four levels as shown in Fig. 4 (left) with the following interpretation. A process cell must contain one or more units which carry out majoring processing activities like react, crystallize, etc. The unit may contain one or more equipment modules which can take care of a finite number of minor processing activities like weighting and dosing. Finally, a control module is typically a collection of sensors, actuators or controllers.

In Fig. 4 (right), the *procedural control model* is shown. A procedure can gradually be broken down into smaller parts. The smallest element that can accomplish a process oriented task is a phase. A phase may be decomposed into steps according to Grafcet/SFC. An operation is a sequence of phases that defines a major processing sequence. Related operations can be combined into unit procedures.

SP88 defines four recipe types: general recipes, site recipes, *master recipes* and *control recipes.* Here, we will only consider the master and control recipes. A recipe contains administrative information, formula information, requirements on the equipment needed and the procedure that defines how the recipe should be produced. The procedure is organized according to the procedural control model. The master recipe is targeted to a process cell. Each individual batch is represented by its control recipe. The control recipe is originally a copy of the master recipe which has been completed and/or modified with scheduling, operational and equipment information. A control recipe can be viewed as an instantiation of a master recipe.

The control recipe itself does not contain enough information to operate a process cell. On some level it must be linked to the process equipment control, i.e. the control actions that are associated with the different equipment objects. SP88 offers large flexibility with respect to at which level the control recipe should reference the equipment control. It is also allowed to omit one or more of the levels in the procedural model. The situation is shown in Fig. 5. The dashed levels could either be contained in the control recipe or in the equipment control. Several examples of how this can be done will be given in Section 5.

4. A BATCH SCENARIO

A simulated scenario consisting of a batch cell has been implemented in G2. The cell consists of raw material storage tanks, mixers, buffers, batch reac-

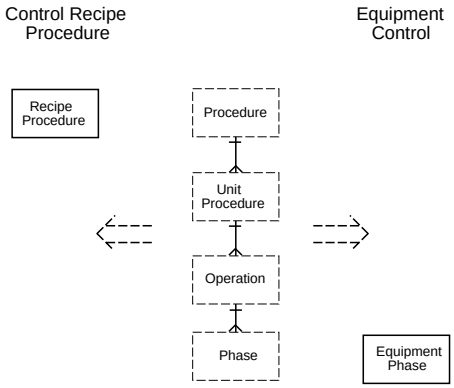

Fig. 5 Procedure/Equipment Control Recipe Separation

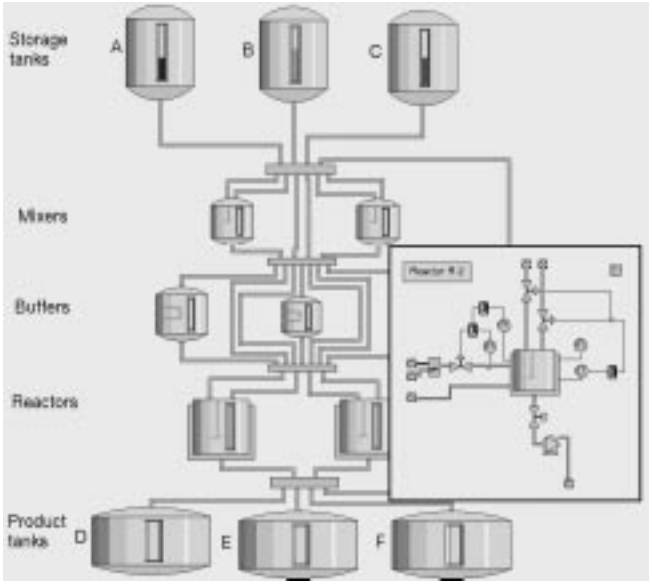

Fig. 6 Process schematic

tors, and product storage tanks. The units are interconnected through valve batteries. The cell can produce two products `D` and `E` using three reactants `A`, `B` and `C`. The scenario consists of a dynamic simulator, the process control system and the operator interface. The dynamic simulator simulates the mass balances, energy balances and chemical reactions in real time. The control system contains sensors, actuators, PID controllers and the High-Level Grafchart based recipe management system. The control system also contains equipment objects representing the unit processes, the equipment modules and the control modules. The operator interface is shown in Fig. 6.

The equipment objects are implemented as classes defined in a class hierarchy. The process equipment objects have an internal structure of equipment and control module objects, attributes and methods. The

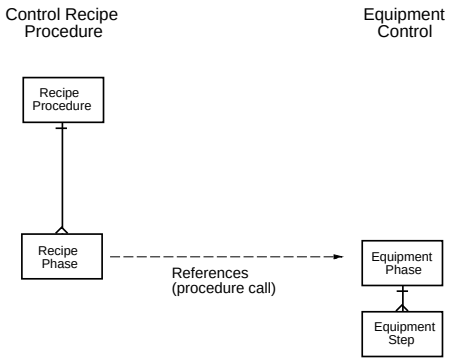

Fig. 7 Recipes based on recipe phases.

methods represent equipment operations or equipment phases and are expressed in Grafchart.

5. RECIPES IN HIGH-LEVEL GRAFCHART

The hierarchical levels in the procedural model of SP88 can conveniently be expressed with the hierarchical abstraction facilities of Grafchart. For example, a procedure could be represented as a function chart with the unit procedures represented as macro steps or Grafchart procedures. A unit procedure macro step could contain operations that also are represented as macro step or Grafchart procedures etc.

The separation between the control recipe procedure and the equipment control is implemented using the method and message passing facilities of Grafchart. The equipment operations and equipment phases are represented as methods of the equipment objects. The control recipe calls the equipment control by invoking a method in an equipment object.

In the following we will give four alternative representations of recipes. The first two alternatives are currently operational, the others are under implementation. As an example we will use the simple recipe for producing product `D`. This is done by mixing reactants `A` and `B`, agitating, and then performing the reaction in a batch reactor.

5.1 *Recipes based on recipe phases*

In the first example the control recipe procedure consists solely of phases. Each phase references the associated equipment phase through a procedure call according to Fig. 7. The resulting control recipe is shown in Fig. 8. Each column represents a unit process that the batch passes through. The recipe starts by an assignment step in which the resources necessary to produce the recipe are allocated. Two version of the recipe have been implemented. One where the opera-

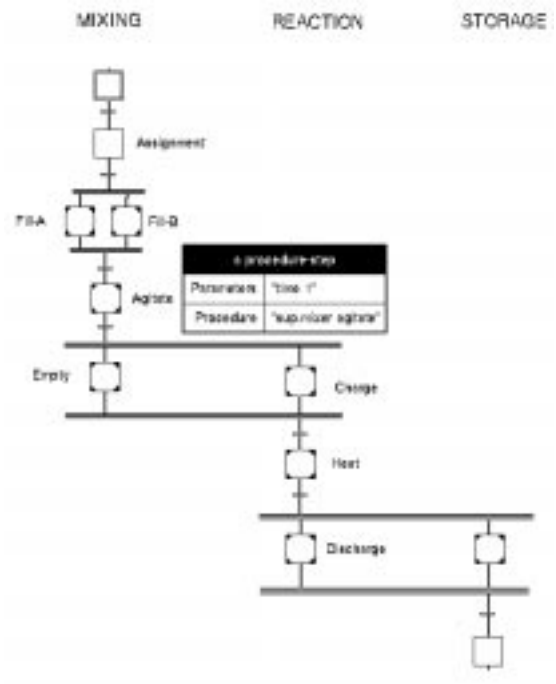

Fig. 8 Control recipe based on recipe phases.

tor manually assigns all the equipment before starting the procedure and one where the equipment is assigned automatically immediately before the equipment is needed. After the assignment step the two reactants `A` and `B` are filled into a mixer, this is done in parallel (the mixers have two inlets). Then the agitation is started. When the agitation is finished the mixer is emptied and at the same time the reactor is charged. This is expressed with the Grafchart parallel construct. When the charging is finished, the reaction is started by heating the batch. Each recipe phase is represented by a procedure step that calls the corresponding equipment phase. In Fig. 8 the attribute table for the agitate step is shown. The procedure step calls the agitate methods in the mixer that has been assigned to the batch. This mixer is contained in the `mixer` attribute of the recipe. The parameter `time` is given the value `1`. This denotes how long the agitation will take. The agitate equipment phase consists of a single step where the agitator motor is active until the defined time has elapsed.

5.2 *Recipes based on recipe operations*

The second alternative also contains recipe operations. The link between the control recipe and the equipment control is still performed at the phase level. The structure is shown in Fig. 9. The resulting control recipe is shown in Fig. 10. The control recipe is a straight sequence of recipe operations. Each recipe

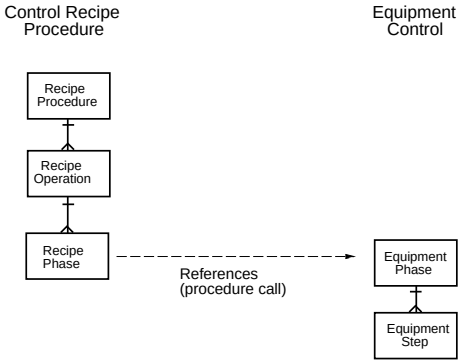

Fig. 9 Recipes based on recipe operations.

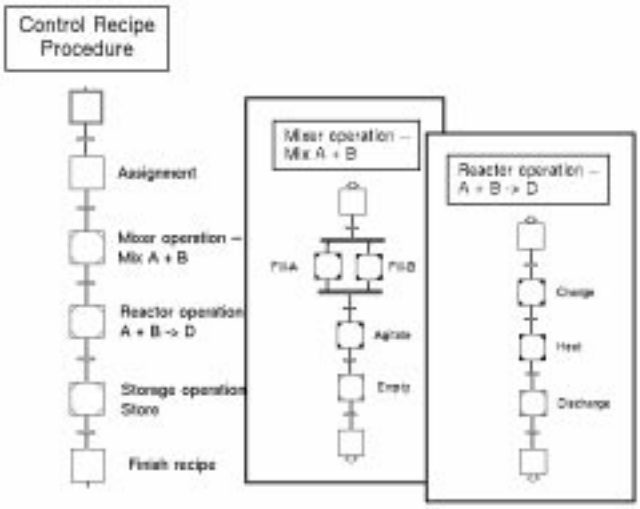

Fig. 10 Control recipe based on recipe operations.

operation is represented as a macro step that contains the recipe phases. The phases are represented as procedure steps that calls the corresponding equipment phases. A problem with this approach is how to handle the parallelism between the operations. The emptying of the mixer and the charging of the reactor should be performed in parallel. Here this is solved using process steps. The `Empty` phase is started through a process step, i.e. as a separate execution thread. The result will be that the `Empty` phase will execute at the same time as the `Charge` phase of the reactor. A drawback of this approach is that the parallelism is not expressed explicitly with the Grafchart constructs and therefore not directly visible to the operator.

5.3 *Recipe tokens and equipment tokens*

The token object and multi-dimensional chart features of Grafchart fit very nicely into the batch control problem. Previously, each control recipe has been

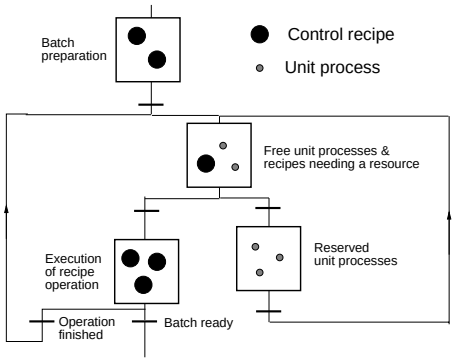

Fig. 11 Resource allocation function chart.

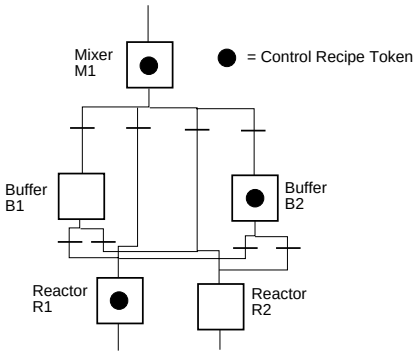

Fig. 12 Process cell structured resource allocation.

represented as a function chart object. Another possibility is to represent a control recipe as a token object, containing the recipe procedure as a method. The execution of a recipe involves two activities or dimensions: resource allocation and recipe execution. These activities are interleaved. The resource allocation can also be described in term of a function chart. This function chart contains two types of tokens: control recipes and unit processes according to Fig. 11. The control recipe procedure is decomposed into recipe operations. The resource allocation associates a unit process with a control recipe. The control recipe operation is executed and the unit process is reserved. When the operation is finished, the control recipe needs a new unit process to execute next recipe operation.

5.4 *Recipe tokens and equipment steps*

An alternative to the previous solution is to represent the unit processes as steps instead of tokens. This gives a resource allocation chart that resembles the physical structure of the process cell according to Fig. 12 where a simplified process cell is shown. The presence of a control recipe token in an equipment step indicates that the recipe is executing a operation in that unit process. The resource allocation mechanism is hidden in the transitions between the equipment steps. The net structure obtained is very similar to the Petri net structures used for analyzing batch control cells in Genrich *et al.* (1994), Yamalidou and Kantor (1991). The difference is that here the resource allocation is integrated with recipe execution through the multi-dimensional chart nature.

6. CONCLUSIONS

The paper has shown how the concepts of High-Level Grafchart can be used to implement a batch recipe management system according to SP88. The SFC formalism is applied at all levels from the recipe procedure down to the control steps. The possibility to use function chart as object methods provides a convenient way of separating the equipment control logic from the recipe procedure. The possibility to use token object and multi-dimensional charts increases the structuring possibilities and allows an integration between Petri net based analysis methods and recipe execution.

This work is supported by the NUTEK REGINA project “High-Level Grafcet for Supervisory Sequential Control”. The authors want to thank the members of the industrial steering committee and Bernt Nilsson for many valuable discussions.

7. REFERENCES

ÅRZÉN, K.-E. (1994): “Grafcet for intelligent supervisory control applications.” *Automatica*, **30:10**.

ÅRZÉN, K.-E. (1996): “Grafchart: A graphical language for sequential supervisory control.” In *Proc. of the IFAC World Congress 1996*.

GENRICH, H. J., H.-M. HANISCH, and K. WÖLLHAF (1994): “Verification of recipe-based control procedures by means of predicate/transition nets.” In *15th International Conference on Application and Theory of Petri nets, Zaragoza, Spain*.

JENSEN, K. and G. ROZENBERG (1991): *High-level Petri Nets*. Springer Verlag.

MOORE, R., H. ROSENOF, and G. STANLEY (1990): “Process control using a real time expert system.” In *Preprints 11th IFAC World Congress*. Tallinn, Estonia.

YAMALIDOU, E. C. and J. C. KANTOR (1991): “Modeling and optimal control of discrete-event chemical processes using Petri nets.” *Computers chem. Engng*, **15**, pp. 503–519.