GRAFCHEART FOR PROCEDURAL OPERATOR SUPPORT TASKS

Karl-Erik Årzén, Rasmus Olsson, Johan Åkesson

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118, S-221 00 Lund, Sweden
{karlerik,rasmus,jakesson}@control.lth.se

Abstract: Support for procedural operations is important in industry. Within the EU/GROWTH project CHEM aimed at developing advanced decision support systems for the process industry, Grafchart, a toolbox for procedure handling based on Grafcet/Sequential Function Charts, will be used to implement a state-transition support toolbox. A brief overview of Grafchart is given, including a new Java version, JGrafchart. Operator support applications where Grafchart has been used are described.

Keywords: Operating Procedure, Sequence Control, Petri nets, Grafcet, Batch Control

1. INTRODUCTION

Support for procedural operations is important in all industrial plants. The operation procedures define the sequence of steps needed in order to accomplish a certain task, e.g., to start the plant up, to shut the plant down, to perform a grade or product change, to produce a batch according to a recipe, to perform equipment tests in order to ensure safe operation or improve fault isolation, etc. These tasks can either be performed automatically by the control and instrumentation systems, or manually or semi-automatically by the process operators. In both cases it is important that the procedure handling system has a well-defined syntax and semantics. In order to simplify formal approaches to, e.g., verification, it is an advantage if the procedure handling systems is based on a formalism for which formal methods already exists. Examples of such formalisms are Petri nets and Grafcet. If the human operator is involved in procedure execution it is an advantage if the procedure language is graphical in nature and provides visual feedback through animation to the operator.

CHEM is the name of a recently started EU/GROWTH project aimed at developing advanced decision support systems for different operator-assisted tasks, e.g., process monitoring, data and event analysis, fault detection and isolation, and operations support, within the chemical process industries. The operations support system in CHEM will be based on Grafchart, a toolbox for procedure handling that has been developed at Lund University since 1991. The toolbox is based on Grafcet/Sequential Function Charts together with ideas from Petri nets and object-oriented programming. Grafchart has previously been used in several related applications. It has been used to implement recipe handling and resource allocation systems according to the S88 batch standard. It has also been used to structure a model-based diagnosis systems for Steritherm, a mode-changing dairy process, and to implement an alarm filtering system. In industrial applications, it has been used to implement a decision support system for hydrogen balance optimization in an oil refinery, and to implement a robot cell controller.

The paper will give a brief overview of Grafchart and JGrafchart, a recent Java version of Grafchart, present some of the existing applications, and discuss how Grafchart will be used in the implementation of an operating procedure handling system.

1.1 Outline of the Paper

Section 2 gives a brief overview of Grafchart. Existing applications where Grafchart is being used for operator support tasks are presented in Section 3. Finally Section 4, discusses how Grafchart can be used to implement an operating procedure handling system.

* Corresponding author Karl-Erik Årzén, Tel. +46 46 2228782; Fax. +46 46 138118; E-mail karlerik@control.lth.se.
Alternative paths
Parallel paths
Transition
Token
Step

Fig. 1 Grafchart graphical syntax

2. GRAFCHART

Grafet, or Sequential Function Charts (SFC), has been widely accepted in industry for representing sequential control logic at the local PLC level through the standards IEC 848 and IEC 61131-3. A large amount of work has been devoted to formal specifications of Grafet and the use of formal verification methods of Grafet models, e.g., (David (1995); Charbonnier et al. (1999); Lhoste et al. (1997)). There is, however, also a need for a common representation format for the sequential elements at the supervisory control level. Grafchart is the name of a graphical language aimed at supervisory control applications that has been developed at Lund Institute of Technology. It combines the function chart formalism of Grafet and the use of formal verification methods of Grafet models, e.g., (Årzén (1991)), (Årzén (1994b)), (Årzén (1993)). Grafchart is currently implemented in G2 from Gensym Corporation.

Grafchart contains steps, transitions, macro steps, alternatives, and parallel paths according to Grafet, see Fig. 1. It differs from Grafet with respect to how step actions are represented. In Grafchart, step actions are represented as G2 rules. The step actions may be conditional or unconditional. They can be 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. Similarly, finally actions are executed once immediately before the step becomes deactivated and abortive actions are executed once immediately before the step is aborted. The step actions associated with a step are placed on the *sub-workspace* of the step. A sub-workspace is a virtual, rectangular window upon which various G2 items such as rules, procedures, objects, displays, and interaction buttons can be placed. A sub-workspace can be activated or deactivated. When a sub-workspace is deactivated all the items, e.g., rules, on the workspace are inactive and "invisible" to G2. To each transition a condition or an event is associated. The transition is enabled when all the preceding steps are 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. The transition expression is also translated into a G2 rule. This rule is invoked when a transition is enabled and is responsible for the transition firing.

Macro steps are used to represent steps that have an internal structure of (sub)steps, transitions, and macro steps. The internal structure is placed on the sub-workspace of the macro step. 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 Grafchart procedures are reentrant. This makes recursive procedure calls possible. A transition connected after a macro step or procedure step will not become enabled until the execution has reached the last step in the macro step or in the associated Grafchart procedure. An exception transition is a special type of transition that only may be connected to macro steps and procedure steps. An exception transition is enabled all the time while the macro step or procedure step is active. If the exception transition becomes true while the corresponding macro step is executing the execution will be aborted and the step following the exception transition becomes active. Macro steps and procedure steps "remember" the execution state when they were aborted and it is possible to resume them in this state. Macro steps and procedure steps are shown in Fig. 2.

2.1 Parameters and methods

Function charts, macro steps, super steps, and steps may have parameters. The parameters can be accessed from within step actions and transition expressions. The parameterization feature is based on the fact that all Grafchart language elements are objects defined in class definitions. The user can specialize these classes to subclasses in which additional attributes are added. These attributes act as parameters with lexical scoping.

It is also possible for Grafchart procedures to have parameters. The parameters can be accessed from within the procedure step. The values can either be constants (numbers, strings, or symbols) or the value of a parameter that is visible in the context of the procedure step. In the latter case a procedure may also return values to the calling procedure step. It is also possible to let the value of a parameter determine which procedure that will be called by the procedure step.

The method feature denotes the possibility to have Grafchart procedures as methods of general G2 objects. For example, a G2 object representing a batch
reactor could have methods for charging, discharging, agitating, heating, etc. From the body of the procedure realizing the method it is possible to reference the attributes of the object that the method belongs to using the `self` notation. A Grafchart object method is called through a procedure step. The method that will be called is determined by an object reference and by a method reference. The example in Figure 3 shows a reactor object R1 that 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.

2.2 High-Level Grafchart

In High-level Grafchart the tokens that indicate that a step is active are objects with attributes. A step may contain several tokens, of the same or of different class. Each step action is associated with a token class. For example, initially actions associated with the token class `foo` are executed once every time a token that is an instance of the class `foo` or one of its subclasses is entered into the step. The token attributes can be referenced and changed from the step actions. Associated with transitions are receptivities. Each receptivity is associated with a token class. The receptivity is enabled and may fire as soon as at least one token of the associated class is present in the step preceding the transition. The receptivity has the attributes condition, event, and action. When the receptivity is enabled and its condition becomes true or its event occurs the transition fires. The condition may involve the attributes of the tokens in the preceding step. The condition may also involve the presence of other tokens in the preceding step, and their attributes. When the transition fires the actions associated with the receptivity are executed. The default action is to move a token from the preceding step to the succeeding step. It is, however, also possible to delete a token in the preceding step, to create a new instance of a token in the succeeding step, or to execute an arbitrary G2 action.

2.3 Multi-dimensional charts

Since G2 objects may contain Grafchart methods, and token objects are G2 objects, it is also possible for token objects to have Grafchart methods. This gives a multi-dimensional chart structure with several powerful structuring possibilities. A toy example of this is the Russian philosophers problem, (Lakos (1994)). This problem is like the dining philosophers problem, except that each Russian philosopher is thinking about a dining philosophers problem. A more realistic example where this can be useful is in the context of batch control. A batch is represented by a control recipe. The batch is subject to two types of operations: production of the batch or a part of the batch according to the batch procedure contained in the control recipe and allocation of production units, e.g., unit processes, to the batch. This fits nicely into a multi-dimensional structure were the control recipe is a token object that contains the batch procedure as a method. The control recipe token moves around in a function chart where the resource allocation is performed. A batch control application of H-L Grafchart is described in (Johnsson and Årzén (1994); Johnsson and Årzen (1996); Johnsson (1997)).

2.4 JGrafchart

In order to increase the portability, Grafchart is currently being ported to Java. The graphical editor and runtime environment is named JGrafchart. The user-interface is implemented in Swing using a class package for graphical object editors called JGo from Northwoods Corporation. JGo supports a number of functions, e.g., drag-and-drop, undo/redo, zoom, cut-and-paste, object connectivity, etc. The current version of JGrafchart can either execute on-line against local I/O or run in simulated mode. A JGrafchart screen dump
is shown in Fig.4. In JGrafchart the textual language used in actions and transition expressions is defined in a formal grammar. The parser generator tool JavaCC is used to generate the Java code for the parsing and for the creation of a syntax tree.

Currently JGrafchart only supports the basic Grafchart functionality. The high-level features of Grafchart are currently being implemented.

3. GRAFCHART FOR OPERATOR SUPPORT

Grafchart has been used in several operator support applications. The application that has been running on-line the longest time is an oil refinery application (Årzén (1994a)). The system uses knowledge-based system techniques coupled with numerical optimization in a decision-support system that gives on-line advice regarding the distribution of hydrogen resources in the refinery. The solution to the problem, i.e. the main reasoning cycle, is structured into three sequential sub-steps. Each of these sub-steps are internally decomposed into further sub-steps. The main reasoning cycle is executed once every two minutes. The entire reasoning sequence can be viewed as a decision tree that is being traversed automatically, once every cycle. The animation feedback gives the human operators a good overview of the operations that are taking place.

Grafchart is also used for batch control recipe handling and resource allocation, see e.g., (Johnsson and Årzén (1998a); Johnsson and Årzén (1998b); Johnsson (1999)). Different possibilities for representing recipes and combining recipe execution with resource allocation according to the S88 batch control standard have been explored using both versions of Grafchart. The S88 procedural model is straightforward to model in Grafchart, see Fig. 5. Grafchart has also had a considerable impact on the definition of Procedure Function Charts (PFC), see (Emerson (1999)), in Part 2 of the S88 standard.

The linking between the control recipe and the equipment control is implemented using methods and message passing according to Fig. 6. The element in the control recipe where the linking should take place is represented by a procedure step. Depending on at which level the linking takes place, the procedure step could represent a recipe procedure, recipe unit procedure, recipe operation or recipe phase. The procedure step calls the corresponding equipment control element which is stored as a method in the corresponding equipment object.

A number of different ways to represent recipes have been proposed. The most straightforward way is to represent each control recipe by a separate Grafchart function chart. Another possibility is to use a high-level function chart for representing each master recipe, and to use the object tokens in this function chart to represent the individual batches. This can also be combined with resource allocation in a Petri-net style, and equipment-oriented operations, e.g., CIP (Cleaning-In-Place), see (Johnsson (1999)).
Grafchart has also been used to implement a prototype of a training simulator for a sugar crystallization process, (Nilsson (1991)). Here, Grafchart is used both to structure the simulation model of the process and to implement the control system for the process. In (Årzén (1995)), Grafchart is integrated with DMP, a model-based diagnosis framework. The aim is to integrate the monitoring and diagnostics for sequential processes with the sequencing logic. The steps contain both actions implementing the sequence control logic and objects representing diagnostic model equations and process faults.

4. GRAFCHART AND OPERATOR PROCEDURE HANDLING

A operation procedure system needs support for procedure representation, procedure execution, and procedure synthesis. In Grafchart operating procedures are represented as sequence charts or as Grafchart procedure. The parameterization feature increases the possibilities for reuse of operating procedures. The different exception handling facilities makes it possible to separate the sequential logic representing the normal, fault-free executions of a procedure, from all exception handling logic that is needed to handle abnormal situations arising during the procedure execution.

An important issue for an operator procedure handling systems is visual feedback to the human operators. Grafchart provides this in several ways. The currently active steps in a function chart are animated. In JGrafchart, also the truth values of the transitions are animated. A true transition is shown in green color, whereas a false transition is shown in red color. It would also be possible to show the exact part of a compound boolean expression that is true or false in a transition condition using different colors for the part of the corresponding text string representing the expression. Finally, the object oriented implementation of Grafchart makes it easy to modify the graphical syntax of the entire language. For example, it is possible to create subclasses of the step and transition classes, with different graphical appearance.

Animation of the currently active steps only shows the present state of the execution. It is in many situations important to also show the execution history. This can be done in two different ways. In the oil refinery application mentioned earlier the path followed in the decision tree is marked by changing the color of the steps that have been active. Before a new reasoning cycle is started all step colors are reset. The approach is illustrated in Fig. 7. In very rapid execution scenarios it may be difficult to see which step that is currently active. JGrafchart therefore supports token tracing. When a step becomes active a token remains in the step. This token gradually fades away, generating a tail effect for the tokens. The idea is illustrated in Fig. 8.

Synthesis of operating procedures can be approached in a number of different ways. One possibility is to use AI-based symbolic planning methods, see, e.g. (Aylett et al. (2001)). Another possibility is to use numerical optimization based approaches based on linear or nonlinear programming. The latter requires the availability of process models. The problem that will receive special focus within CHEM is process state transitions, e.g., grade changes, for continuous flow processes. A model predictive control (MPC) approach will be taken. The optimization will only be active when a new grade change should be performed. The calculated control sequence will be translated into a Grafchart procedure dynamically. This procedure will then be effectuated. During the state transition the execution will be monitored, and if the signals deviate too much from their predicted values a new optimization will be performed.

When the state transition is completed the state transition will be stored, for possible future reuse. An interesting possibility is to adopt the structuring mechanisms of the S88 batch standard also for continuous-flow processes. In S88 each process unit may only be accessed through a set of predefined operations, rather than by direct manipulation of its associated variables and signals. This can be likened to object-oriented programming and shares the benefits of this. Using the S88 approach can be advantageous for the state-transition problem. The state-transition problem can be viewed as trying to find a path from one point in a high-dimensional space to another point. The S88 predefined operations can be viewed as precalculated
paths, that are guaranteed to be safe and not violate any signal constraints. Finding the path from a start point to an end point can then be reduced to the problem of finding the path to the start point of the nearest pre-calculated path that takes the system sufficiently close to the desired end point, followed by the subproblem of finding the path from the end point of the pre-calculated path to the desired end point. If the latter point are sufficiently close this problem may be solved by the basic regulatory control functions in the plant control system.

5. CONCLUSIONS

Using a Grafcet and Petri net-based language such as Grafchart as the basis for an operation procedure handling system within a decision support systems aimed at the chemical process industry has several advantages. The language has a well-defined syntax and semantics. It supports modularization and encapsulation and it provides visual feedback to the process operators. Grafchart has already been used in industrial operator support applications and in the implementation of recipe-based batch control systems. Within the CHEM project Grafchart and JGrafchart will be combined with optimization-based synthesis of operating procedures to generate state-transition procedures for grade changes in continuous-flow processes.

5.1 Acknowledgments

This work has been supported by the EU project CHEM, by the Center for Process Design and Control (CPDC) funded by the Swedish Foundation for Strategic Research (SSF), and by LUCAS - Center for Applied Software Research. The CHEM project is funded by the European Community under the Competitive and Sustainable Growth Programme (1998-2002) under contract G1RD-CT-2001-00466.

6. REFERENCES


