



# LUND UNIVERSITY

## Remarks and Suggestions Concerning G2 Version 1.1

Årzén, Karl-Erik

1988

*Document Version:*

Publisher's PDF, also known as Version of record

[Link to publication](#)

*Citation for published version (APA):*

Årzén, K.-E. (1988). *Remarks and Suggestions Concerning G2 Version 1.1*. (Technical Reports TFRT-7409). Department of Automatic Control, Lund Institute of Technology (LTH).

*Total number of authors:*

1

### General rights

Unless other specific re-use rights are stated the following general rights apply:

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: <https://creativecommons.org/licenses/>

### Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117  
221 00 Lund  
+46 46-222 00 00

CODEN: LUTFD2/(TFRT-7409)/1-07/(1988)

Remarks and suggestions  
concerning G2 version 1.1

Karl-Erik Årzén

Department of Automatic Control  
Lund Institute of Technology  
December 1988

**TILLHÖR REFERENSBIBLIOTEKET  
UTLÄNAS EJ**

<b>Department of Automatic Control</b> <b>Lund Institute of Technology</b> P.O. Box 118 S-221 00 Lund Sweden		<i>Document name</i> Technical report	
		<i>Date of issue</i> December 1988	
		<i>Document Number</i> CODEN: LUTFD2/(TFRT-7409)/1-7/(1988)	
<i>Author(s)</i> K-E. Årzén		<i>Supervisor</i>	
		<i>Sponsoring organisation</i> STU, The Swedish Board for Technical Development under contract DUP 85-3084P.	
<i>Title and subtitle</i> Remarks and suggestions concerning G2 version 1.1.			
<i>Abstract</i> A summary of the experience that we have had of using G2 from Gensym Corp. as a tool for modelling and simulation and as an expert system tool for monitoring.			
<i>Key words</i>			
<i>Classification system and/or index terms (if any)</i>			
<i>Supplementary bibliographical information</i>			
<i>ISSN and key title</i>		<i>ISBN</i>	
<i>Language</i> English	<i>Number of pages</i> 7	<i>Recipient's notes</i>	
<i>Security classification</i>			

The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Telex: 33248 lubbis lund.

# Remarks and suggestions concerning G2 version 1.1

Karl-Erik Årzén

Department of Automatic Control

Lund Institute of Technology, Box 118, 221 00 Lund, SWEDEN

This paper is a summary of the experiences we have had of using version 1.1 of G2 from Gensym Corp. Our overall impression of G2 is very good. The way real-time issues are integrated with the G2 inference engine is very powerful. Other things which we specially like are the object orientation, the graphical development environment, and the structural editor.

The experience we have of G2 is restricted to the off-line case. We have used G2 in two types of applications: as a modelling and simulation tool and as an expert system tool for monitoring. We have no experiences of the data server interface or real-time performance.

The paper is a mixture of minor technical issues and more fundamental architectural issues. The work has been supported by STU, the National Swedish Board for Technical Development, under contract no. 85-3084P.

## Hierarchical models

An important motivation for using knowledge-based systems in operator assistance applications such as monitoring and diagnosis is to reduce the cognitive overload that operators are exposed to in modern control rooms. One important means for structuring large amounts of information is to use hierarchical abstraction. This is not possible in the current version of G2. Some of the building blocks for achieving it, e.g., subworkspaces, are available but several things are missing.

A very simple example will illustrate what we mean. Consider two cascaded PID controllers. This can be seen as a cascade controller that has an internal structure according to Fig 1. At a quick glance this may seem easy to implement by simply placing the internal PID controllers on the subworkspace of the cascade controller. This is however not the case.

### Objects within objects

In a hierarchical model it is natural to store the internal parts of an object as attributes. The possibility in version 1.1 to have general objects as attribute values seems to make this possible. Indeed it is possible to refer to the internal objects of an object in the following way, **the gain of the outer-loop of cascade-2**. However, a large problem remains.

In a hierarchical model it is necessary for an internal object to also have an icon representation. That is, the internal object has more than one reference. In G2 all objects are distinct, even if they have the same name and are of the same class. This makes it impossible to have an internal object represented

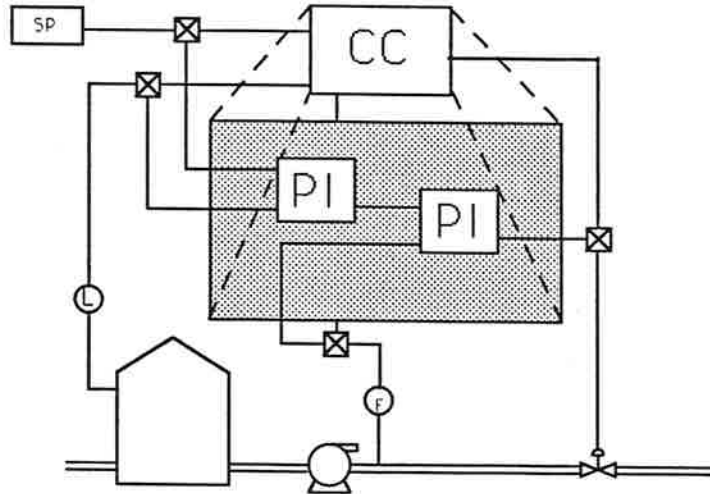


Figure 1. Hierarchical model of a cascade controller

both as an attribute value in the “super object” and as an icon on the “super-objects” subworkspace.

Hierarchical models is probably not the only case where multiple representations of an object would be desirable. Situations where a single object is represented with multiple icons on different workspaces (either with the same icon or with different icons) can be easily thought of.

To achieve this, G2 must treat all references to objects with the same name and of the same class as references to a single object. This does not easily coincide with the current possibility to have more than one name for an object. There will also be problems concerning object deletion, object connections, subworkspaces etc.

Another related issue is the treatment of objects with the same name but of different classes. In G2 these objects are seen as distinct objects which is natural. G2 has, however, no means for referencing the individual objects. Consider the case with the classes *class-1* and *class-2* which both have an attribute with the name *foo*. Assume that *class-1* has an instance named *bar* and that *class-2* has an instance also named *bar*. G2 has no way to refer to the attribute *foo* in the different instances. The expression *the foo of bar* is ambiguous. An straightforward extension to the G2 grammar would be to allow the following attribute references.

**the <attribute> of [<class>] <object>**

### **Junctions between workspaces**

Another issue that must be solved is junctions between different workspaces. In the hierarchical cascade controller example both the cascade controller object and the internal PID objects must be connected to the surrounding objects. What is needed are junctions that allow connections between workspaces. In Fig. 1, these junctions are indicated as crossed squares. They behave as ordinary junction blocks with the exception that they allow connections between different workspaces.

Interworkspace junctions would be very useful also for general structuring purposes. It is often desirable to split a model between different workspaces.

### Hierarchical model classes

The final prerequisite for G2 to have hierarchical models is the possibility to define hierarchical classes. It should be possible to define the cascade controller with its internal PID controllers as a class. To do this it must be possible to specify the layout and connections among the internal objects in the class definition.

## Inheritance

G2 only have single inheritance. In many cases, the class decomposition and structuring of a problem would be much more natural if multiple inheritance was allowed. The possibility to use "mix-in" classes that only adds a certain behavior or a certain set of attributes to a class is often useful. I cannot see any problems with this. The class precedence rules of CLOS or Flavors could be used to calculate class precedence lists that determine how attributes are inherited and shadow each other in case of ambiguities.

## Simulation

The G2 simulator is based on simulation of individual variables. The step length in the integration routine can be set individually for different variables. This gives a system which is very flexible. It is however also very easy to get incorrect simulation results due to, e.g., bad choices of step length. An alternative to variable based simulation is to base the simulation on the system concept. The difference is most significant in the case of systems of difference equations.

Consider the following example. Assume that a discrete PID controller should be simulated. The controller needs three discrete states: the old value of the process variable, the old value of the integration part, and the old value of the derivative part. It is desirable to write the simulation equations as generic equations. The problem with variable based simulation arises when one consider the sampling time of the controller. The sampling time is used in the simulation equations and can have different values for different PID controllers in the simulation. Therefore, the sampling time is stored as an attribute, `dt`, in the PID objects. However, for the simulation to work the value of the sampling time must be stored in three additional places in a PID object. These places are the **Time increment for update** attribute in the simulation subframes of the three discrete state variables. The value of this attribute is restricted to `none` or `<number> second(s)`. Hence, if the sampling time is changed the change must be performed in four different places.

A partial solution would be to allow the **Time increment for update** attribute to take a general expression as a value, i.e., `the dt of pid-1`. An even better solution would be to view a simulated discrete PID controller object as a discrete system that has a time increment attribute that is default for all simulated discrete variables inside the system.

## Connections as abstract relationships

The connection concept of G2 is mostly used to represent physical connections among objects. As indicated in the manual, connections may also represent abstract relationships among objects.

According to our opinion the latter is not sufficiently supported in current G2. One example is the explicit use of the word connected in the G2 grammar for referring to objects that are connected. When using abstract relations this sounds strange and differs from the usual natural english style.

Suppose that an object,  $x$  is related to five other objects through five different relationships. The relationships are represented as different connection classes with different intersection patterns. Suppose now that that one would like to refer to the object which is related to  $x$  with the specific relation *relation-a*. The G2 reference to this object is as follows.

**the object connected to the relation-a connected to  $x$**

If stubs were used, things would be simpler. Due to our opinion, stubs are not natural for abstract relationships.

## Multiple layers

Realistic process models contains various types of connections. Examples are flow of different materials, flow of energy, etc. A single schematic with all objects and connections soon gets very complex. A possible way to deal with the complexity would be to have different layers of connections. Each connection class on the workspace constitutes a layer. A layer can be visible or hidden. Connections and stubs of a hidden layer are invisible. Objects whose stubs and connections are hidden are also invisible. The workspace menu has means for hiding and showing the different layers. With all layers visible, the total process schematic is shown. The user can concentrate on different functions of the process by hiding irrelevant connection layers.

## Procedures

Eagerly awaited.

## Dynamic objects

The possibility to create and delete objects from the inference engine would be very useful.

## Demonstration knowledge bases

It is very useful that G2 is followed by a set of demonstration knowledge bases. There seems, however, to be a certain time lag before the new features of G2 are used in the demonstrations. It would be useful to have new demonstrations that, e.g., give examples on how to use the new operator controls.

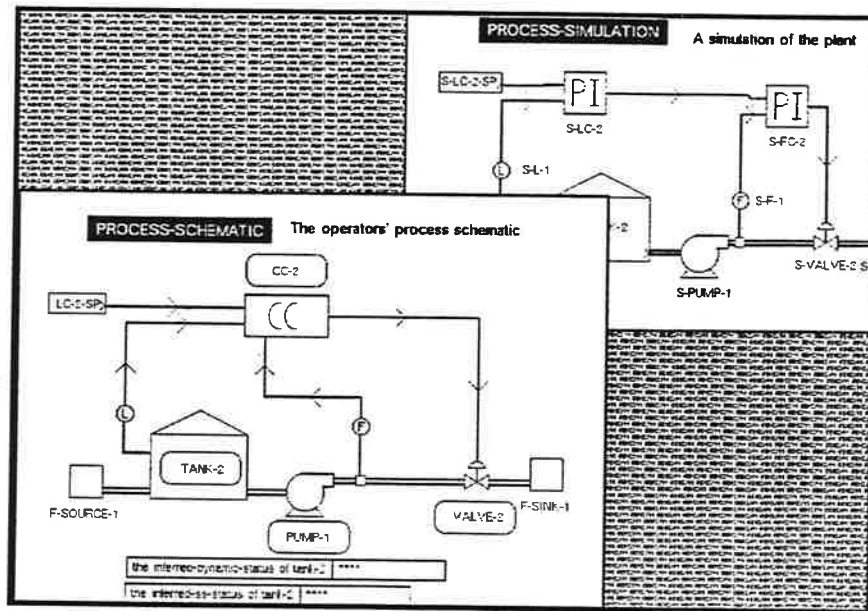


Figure 2. G2 simulation organization.

What is most disturbing with the demonstrations, however, is their organization. A good example of this is the valve-and-tank knowledge base. Here, quantitative simulation and rule-based monitoring are mixed in very confusing way. Sensors with simulation formulas are used to provide values for the numerical simulation of flows and levels. In the same time, rules are used to compute qualitative values for the same variables and to inform the operator when the qualitative value are not normal. What causes confusion is that numerical, simulated process variables and qualitative values, calculated by the monitoring rules, are stored in the same objects. The underlying problem is what the process equipment icons on the process schematic really represent. In this demonstration, the icons both represent the operators and monitoring systems view of the process component and the actual, physical components (in terms of their simulated behavior). In our opinion it is better to separate the numerical simulation and the monitoring into different objects. The organization is shown in Fig. 2.

A separate workspace is used for the numerical simulation. The objects contain the process variables and has associated generic simulation equations. The objects on the process schematic contains the numerical process variables that are measured through sensors and the qualitative abstractions of these values derived through monitoring rules. The sensors are used to connect the simulation workspace and the process schematic. With this organization the difference between running against a simulation and against a real, physical process through data servers is very small.

## History expressions

The history expressions in G2 are aimed only at quantitative variables. The only history expression that is relevant for symbolical and logical variables are the value of <variable> as of <time variable> ago. It is easy to think of other types of history expression for non-quantitative variables that would be very useful. Some examples are: "the number of times the status variable has been high during the last 30 minutes", "the percentage of time the status



variable has been high during the last 30 minutes”, “the time when status variable received its current value” etc.

## Graphs

The restriction that the **Expression to display** attribute must be the name of a variable and not a general expression is disturbing. As long as the referenced variable has a stored history it should not make any difference. The requirement that a displayed variable should have a stored variable should also be possible to relax. Consider the case when a displayed variable gets its values from another variable which has a stored history. In the current system, the displayed variable is required to also store a history.

Another issue is the way graphs are drawn. In the current system the graphs are drawn in exclusive-or mode. If two variables with the same values are shown in the same graph the two variables overwrite each other and none of them are shown. The situation is common. One example is the when both the inflow and the outflow of a process component are shown during steady state operation. Another example is the set-point and the process variable of a PI controller during steady state operation.

The meaning of the **Show simulated values** attribute is not exactly the same as in the other types of displays. If a simulated variable is shown in a graph it may well be that the **Show simulated values** attribute should have the value **no** as long as the history is stored in the variable frame and not in the simulation subframe.

## Operator controls

### Action buttons

It would be nice if action buttons were treated as general objects in the sense that the user may specialize them. One immediate need is to be able to define the action button icon in the same way as other icons are defined. In many cases we have found it useful to place several smaller action buttons on top of an ordinary object icon. This is difficult to do with fixed, one-size button icons.

Multiple action buttons where the choice between different actions is done with a pop-up menu would also be nice. In some cases it may also be useful to be able to enter text input from the keyboard.

### Sliders

It is difficult to use sliders to change values in a simulation. One example could be to change regulator parameters in a simulated controller. Another could be to change set points and operating points in the simulation. In order to change a numerical parameter in a simulation, the parameter must be a quantitative variable. Since the G2 simulator expects simulation formulas to depend only on other simulated values or time states, the data server of the parameter must be **G2 simulator**. The only way the inference engine can change the value of a simulated variable is through an explicit **set** action.

Due to this, the only way we have found to solve the problem is to let the slider affect a quantitative dummy variable which has the inference engine as data server. A whenever rule is associated with the dummy variable. This rule executes an explicit set action on the simulated parameter whenever the dummy variable receives a new value. In order for this to work the simulation formula of the parameter looks like **state variable:  $d/dt = 0$ , with initial value "the default number"**. Even though it works, it is not a convenient solution.