CODEN: LUTFD2/(TFRT–7423)/1-007/(1989)

Scones — An Interactive Block Diagram Editor for Simnon

Dag M. Brück

Department of Automatic Control
Lund Institute of Technology
July 1989

Department of Automatic Control **Lund Institute of Technology** P.O. Box 118 S-221 00 Lund Sweden	*Document name* INTERNAL REPORT
	Date of issue July 1989
	Document Number CODEN:LUTFD2/(TFRT–7423)/1–007/(1989)
Author(s) Dag M. Brück	*Supervisor*
	Sponsoring organisation The National Swedish Board of Technical Development (STU project 87-2503)

Title and subtitle

Scones — An Interactive Block Diagram Editor for Simnon

Abstract

Scones is an interactive tool for creating connecting systems in Simnon, a simulator for non-linear systems. The main purpose of this work was to gain additional experience in using the programming language C++, and to evaluate InterViews, a user interface toolkit developed at Stanford University.

The report gives a short overview of Scones and demonstrates a major example. Manual pages describe the operation of Scones.

Scones currently runs on Sun-3 workstations under the X Window System.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title	*ISBN*

Language English	*Number of pages* 7	*Recipient's notes*
Security classification		

1. Overview

Scones is an interactive tool for creating connecting systems in Simnon, a simulator for non-linear systems [Elmqvist et at., 1986]. The main purpose of this work was to gain additional experience in using the programming language C++, and to evaluate InterViews, a user interface toolkit developed at Stanford University [Linton et al., 1989]. InterViews is written in C++ and runs on top of the X Window System on many workstations. Our experiences with InterViews when implementing Scones can be found in [Brück, 1989]; a simpler block diagram editor implemented with the graphics standard PHIGS is described in [Brück, 1988]. C++ is described by Stroustrup [1986] and Lippman [1989].

A connecting system describes the interaction between continuous or discrete systems. A connection from an output of one system to an input of another system is in Simnon represented by an assignment statement. With Scones, the user can interactively define the connecting system by drawing a block diagram. Arithmetic operators are represented by special symbols (e.g., a sum). When the block diagram is complete, Scones will automatically generate the textual representation for Simnon. Scones can handle the following basic objects:

- Continuous or discrete systems are either defined by the user, or read from an existing Simnon `.t` file.
- Generators are general arithmetic or boolean expressions, connected to inputs. A typical example is `if t > 0 then 1 else 0`.
- Scones provides special symbols for sum, product and negation (-1). These symbols are transformed to arithmetic expressions in the connecting system.
- Connections provide the interaction between systems. Connections always go from an output to an input, or from a generator to an input.
- Scones inserts `TIME t`, the global simulation time, in the connecting system.

Scones has a "conventional" Macintosh-inspired graphical user interface. All operations are controlled with the mouse. An object is selected by pointing at it with the cursor and clicking one of the mouse buttons. Actions are chosen from menus. The textual representation of a system can be sent to the user's text editor for addition of simulation code. The operation of Scones is described in the enclosed manual. Scones currently runs on Sun-3 workstations under the X Window System.

2. An example

A fairly complex example has been taken from Åström and Wittenmark [1989], Figure 4.12, page 139. The user's view after drawing the block diagram is shown in Figure 1. Scones will from this graphical representation produce the connecting system shown in Listing 1. All connections in the block diagram have been translated to equations.

When creating new continuous or discrete systems, Scones will create a skeleton `.t` file for the system. The skeleton contains the system heading and declarations of inputs, outputs and states. The user must add the equations of

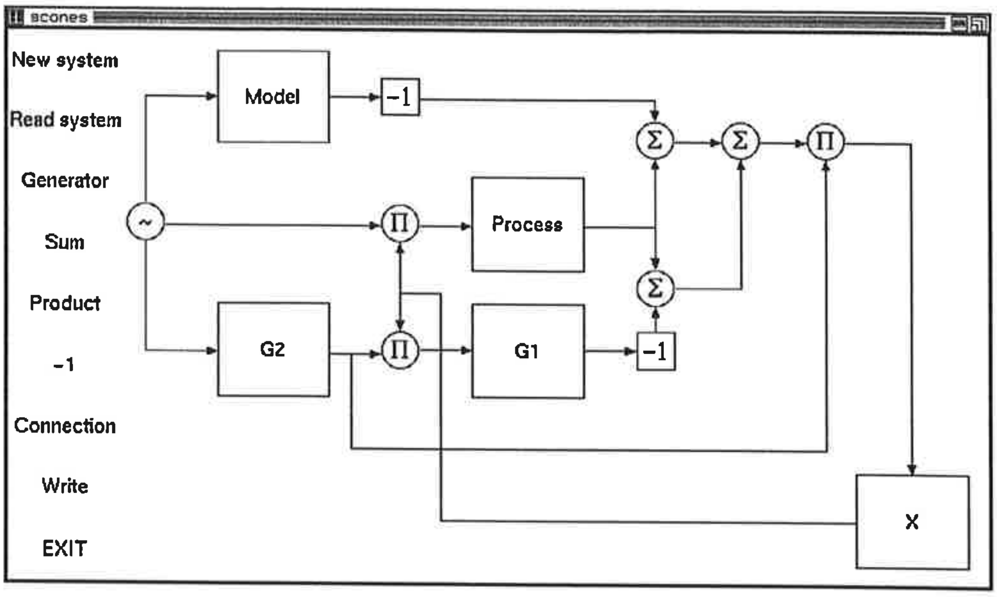

Figure 1. A complex block diagram adopted from Åström and Wittenmark [1989], Figure 4.12, page 139.

```
CONNECTING SYSTEM Consys
"Filename consys.t
"Created Mon Jul 10 14:22:13 1989
TIME t
"System: Model
uc[Model] = uc
"System: G2
in[G2] = uc
"System: G1
in[G1] = out[G2] * theta[X]
"System: Process
in[Process] = theta[X] * uc
"System: X
in[X] = out[G2] * ((-ym[Model]) + y[Process] + (-out[G1]) + y[Process])
"Generator: uc
END
```

Listing 1. The connecting system produced by Scones.

the system using his ordinary text editor. The .t files created for the systems in Figure 1 are shown in Listing 2.

3. References

Åström, Karl Johan and Björn Wittenmark (1989): *Adaptive Control*, Addison-Wesley Publishing Company, Reading, MA, USA.

Brück, Dag M. (1988): "Modelling of Control Systems with C++ and PHIGS," *Proc. USENIX C++ Conference*, October 17–20 1988, Denver, CO, USA. Also available as Report TFRT-7400.

Brück, Dag M. (1989): "Experiences of Object-Oriented Development in C++ and InterViews," Report TFRT-7418, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Elmqvist, Hilding, Karl Johan Åström and Tomas Schönthal

```
CONTINUOUS SYSTEM Model
"Filename model.t
"Created Mon Jul 10 14:00:15 1989
INPUT uc
OUTPUT ym
END

CONTINUOUS SYSTEM Process
"Filename process.t
"Created Mon Jul 10 14:01:52 1989
INPUT in
OUTPUT y
END

CONTINUOUS SYSTEM G1
"Filename g1.t
"Created Mon Jul 10 14:01:18 1989
INPUT in
OUTPUT out
END

CONTINUOUS SYSTEM G2
"Filename g2.t
"Created Mon Jul 10 14:00:47 1989
INPUT in
OUTPUT out
END

CONTINUOUS SYSTEM X
"Filename x.t
"Created Mon Jul 10 14:02:25 1989
INPUT in
OUTPUT theta
END
```

Listing 2. Skeleton code for the systems used in Figure 1.

(1986): *Simnon User's Guide for MS-DOS Computers*, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

LINTON, MARK A., JOHN M. VLISSIDES and PAUL R. CALDER (1989): "Composing User Interfaces with InterViews," *IEEE Computer*, **22**, 2, February 1989.

LIPPMAN, STANLEY B. (1989): *C++ Primer*, Addison-Wesley Publishing Company, Reading, MA, USA.

STROUSTRUP, BJARNE (1986): *The C++ Programming Language*, Addison-Wesley Publishing Company, Reading, MA, USA.

NAME

Scones – an interactive tool for creating Simnon's connecting system.

SYNOPSIS

scones

DESCRIPTION

A connecting system describes the interaction between continuous or discrete systems. A connection from an output of one system to an input of another system is in Simnon represented by an assignment statement.

With *Scones,* the user can interactively define the connecting system by drawing a block diagram. Arithmetic operators are represented by special symbols (e. g., a sum). When the block diagram is complete, *Scones* will automatically generate the textual representation for Simnon.

Scones can handle the following basic objects:

Systems	Continuous or discrete systems are either defined by the user, or read from an existing **.t** file.
Generators	Generators are general arithmetic or boolean expressions, connected to inputs. A typical example is **if t > 0 then 1 else 0**. *Scones* will accept any string as a generator and use it in assignments; the meaning of the string is not checked, and parameter initializations are not generated.
Arithmetic	*Scones* provides special symbols for sum, product and negation (–1). These symbols are transformed to arithmetic expressions in the connecting system.
Connections	From an output to an input, or from a generator to an input.
Time	*Scones* always inserts **TIME t** in the connecting system.

OPERATIONS

All operations are controlled with the mouse. An object is selected by pointing at it with the cursor and clicking one of the mouse buttons.

New system

Select continuous or discrete system by clicking the button icons. Enter system name and the filename of the corresponding **.t** file. Enter any number of blank separated inputs, outputs and states. *Scones* will create a skeleton **.t** file with this information; any existing file with the same name is deleted.

After clicking **OK,** move the cursor to the drawing area on the right. A rectangle the size of the system will slide with the cursor. Clicking the mouse defines the position of the system.

Read system

Enter the name of a **.t** file describing a continuous or discrete system. *Scones* will extract the system name and any inputs, outputs or states. Move the system to the drawing area.

Generator

Enter a text string defining the generator. Move the generator to the drawing area.

Sum, Product, –1

Move the symbol to the drawing area.

Connection

First select an output terminal, then an input terminal. States are regarded as both inputs and outputs. A terminal of a system is selected by first selecting the system, then selecting one of the alternatives from the popup form. Generators and other symbols are selected directly.

The endpoints of the connection will be located at the edges of the components, close to the points of selection. The layout of the connection is controlled by selecting breakpoints in the void between components.

An input can be connected to only one output. An output can be connected to many inputs. The sum and product symbols will accept any number of input connections.

Write

Enter name and filename of the connecting system. Selecting **OK** will create a file with the connecting system. Selecting **CANCEL** will continue operations.

Exit

Enter name and filename of the connecting system. Selecting **OK** will create a file with the connecting system and exit *Scones*. Selecting **CANCEL** will continue operations. Selecting **QUIT** will exit *Scones* without creating a file.

Selecting a component

Selecting a component by clicking the *right* mouse button will present a menu with the following alternatives:

Display	Show name, filename and terminals of a system, or the definition of a generator.
Move	Move rectangle and click at desired new location.
Edit	Systems are edited by invoking your editor on the .t file. The internal representation of the system will not reflect any changes. The definition of a generator is edited in a popup input box.
Delete	The component, and any connections to or from it, are deleted.

Selecting a component by clicking the *left* mouse button is a short form for Display. Selecting a component by clicking the *middle* mouse button is a short form for Move.

Text input

Text input boxes understand simple Emacs-like commands. To edit text, it is *necessary* to select the box and to keep the cursor inside the box. Input boxes extend as required to hold long lines.

A single mouse click selects a new insertion point between characters. A double click selects a whole word. Dragging across the text selects a range of characters. Keyboard commands are: move left (^B), right (^F), to beginning of line (^A), to end of line (^E); delete selection or previous character (DELETE); delete selection or next character (^D); select word (^W); select all (^U); restart edit, undoing all changes (^R).

Graphical output

Scones has no facility for printing the block diagram, so the X screen dump must be used. Enter the following command in a shell window:

```
xdpr -device ps
```

When the cursor changes to a cross-hair, move it to the *Scones* window and click a mouse button. Clicking in the grey background dumps the screen.

SEE ALSO

SIMNON User's Guide for MS-DOS Computers.

Emacs(1), xdpr(1), xwd(1), xpr(1).

ENVIRONMENT

The environment variables VISUAL or EDITOR define the editor used for editing systems. Running Emacs in server mode is particularly nice.

Scones is only available on Sun workstations running the X Window System, release 3.

AUTHOR

Dag M. Bruck, Department of Automatic Control, Lund Institute of Technology, P. O. Box 118, S-221 00 Lund, SWEDEN. E-mail: dag@control.lth.se.

BUGS

This is an early experimental version of *Scones.*

1. A connection cannot be explicitly deleted (although connections to/from a deleted

component are deleted).

2. Connections do not follow moved components, but point into the void.
3. The internal representation of a continuous or discrete system is not updated when the text file for the system is edited.
4. Diagonal connections get a square arrow head.
5. Breakpoints of a connection cannot be moved, i. e., the shape of a connection is fixed.