SKILL Tutorial

(*prepared by*: Charlie Boecker, EcpE Department, Iowa State University)

0. Introduction

This is meant as a very quick tutorial to get you acquainted with development of SKILL code. You don't have to do this, if you don't want to. Although you don't have to do this, I advise you to because you will learn how easy it is to program in SKILL. I advise you to download the reference documentation before attempting this lab. They should be on the web site. Also, if you have any function that you are looking for, type the following in a Unix window.

```
>openbook &
```

This will start up the online help for Cadence. When nobody is around that can help, this is how you will be able to answer most of the questions you might have about Cadence.

1. Familiarization with SKILL Programming

1.1 In the CIW type the following and hit return:

```
>1 + 2
```

three should be the result in the CIW window. Ok, enough easy stuff.

1.2 In the CIW type the following and hit return:

```
>myList = list( 1 3 5 7 a
```

nothing happens, why? If you were paying attention in lecture, you should know. If not, then you may be there for awhile. (HINT: It has something to do with a missing parenthesis)

1.3 Now we have a list. Get the first element from the list using a function in the reference material and set it equal to the variable b. (The function is also something you drive.)

```
> b = function( myList )
```

1.4 Let's do something interesting now. Create a new cell using the method that we normally use. Call the cell test_layout and use the layout view. The window should be open now. We need to get a pointer to the database object that cadence created for the layout. To do this type the following in the CIW and hit enter.

```
>cvID = geGetWindowCellView()
```

You should see that cvID is now equal to some long expression like db:123456789.

1.5 Create a rectangle of Metal1 in the layout window using the "normal" way.

1.6 Create a rectangle of Active using SKILL by typing the following in the CIW and hitting enter.

```
>dbCreateRect(cvID list( "Active" "drawing" ) list( 10:0 20:10 ) )
```

You should see a rectangle appear in the layout window. Now you know two ways of creating a rectangle in cadence. The way just describe is used to do automated layout. This can be used in structures that are regular and repeated. Instead of laying out several thousand of these by hand, a program can do it in no time at all.

2. Making a SKILL Function

2.1 You will be shown the format of creating a new function in this section. You will also be shown how to load the new function into the SKILL interpreter, so the interpreter knows that it is a function when it is entered into the CIW.

2.2 Open an text editor in your home directory and type the following into it.

```
procedure( rectanglefury( num_rect )
        cvID = geGetWindowCellView()
        for( i 1 num_rect
          llx = random( 1000 )
          lly = random( 1000 )
          urx = random( 1000 )
          ury = random( 1000 )
          x = random( 10 )
          if( ( x > 6 ) dbCreateRect(cvID list("Metal1" "drawing" ) list( llx:lly urx:ury ) ) )
          if( ( x>3 && x<6 ) dbCreateRect(cvID list("Metal2" "drawing" ) list( llx:lly urx:ury ) ) )
          if( ( x < 3 ) dbCreateRect(cvID list("Poly1" "drawing" ) list( llx:lly urx:ury ) ) )
        )
)
```

After you have typed it in save the file as */home/user/tutorial.il.* Where *user* is your 465 account name.

2.3 Now go to the CIW and type in the following and hit enter.

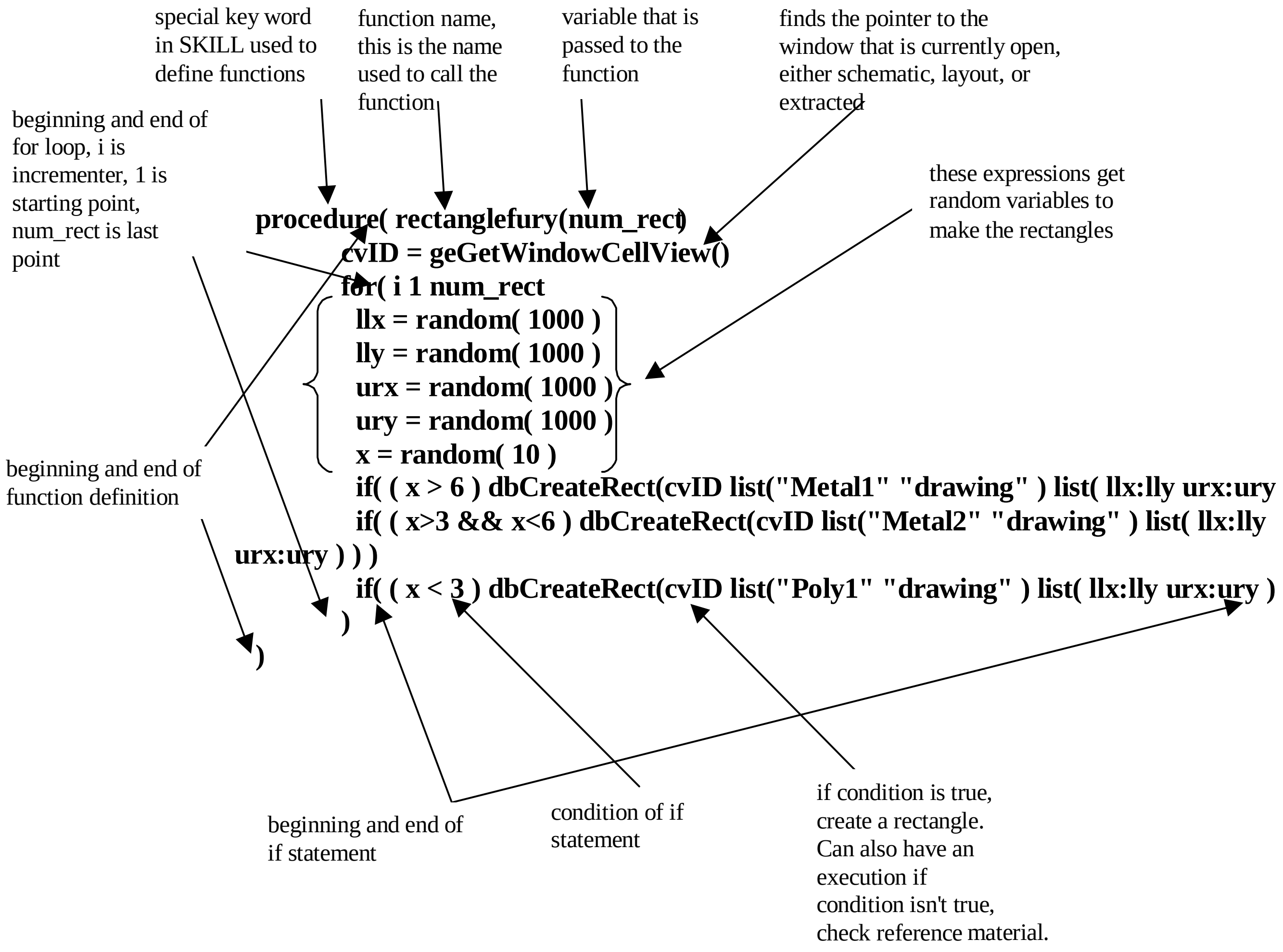
```
>load( "/home/user/tutorial.il")
```

Your new function, rectanglefury, should now be ready to be tested. First, we need to have a layout view that the program will execute its function on. Create a new layout cellview. Make sure the cellview is open when you type the following in the CIW and hit enter.

```
>rectanglefury(100)
```

What happened? If it worked, you just created a bunch of Metal1, Metal2, and Poly1 rectangles. This was just to give you an idea of what can be done in SKILL. Hopefully, with this tutorial and the reference material, the final project programming will be easy. The next section contains an analysis of the function that you created in section 2.2, so that you will be able to create your own function.

2.4 Analysis of the function in section 2.2, rectanglefury.



```
procedure( rectanglefury(num_rect)
    cvID = geGetWindowCellView()
    for( i 1 num_rect
      llx = random( 1000 )
      lly = random( 1000 )
      urx = random( 1000 )
      ury = random( 1000 )
      x = random( 10 )
      if( ( x > 6 ) dbCreateRect(cvID list("Metal1" "drawing" ) list( llx:lly urx:ury
      if( ( x>3 && x<6 ) dbCreateRect(cvID list("Metal2" "drawing" ) list( llx:lly
urx:ury ) ) )
      if( ( x < 3 ) dbCreateRect(cvID list("Poly1" "drawing" ) list( llx:lly urx:ury )
    )
)
```

special key word in SKILL used to define functions
function name, this is the name used to call the function
variable that is passed to the function
finds the pointer to the window that is currently open, either schematic, layout, or extracted
beginning and end of for loop, i is incrementer, 1 is starting point, num_rect is last point
these expressions get random variables to make the rectangles
beginning and end of function definition
beginning and end of if statement
condition of if statement
if condition is true, create a rectangle. Can also have an execution if condition isn't true, check reference material.