## Circle Language Spec Plan Assignment Spec 2008-07 Project Summary

*Author: JJ van Zon*

*Location: Oosterhout, The Netherlands*

### Super-Project

Circle Language Spec, Command as a Concept Spec

### Goal

Work out the Assignment article group, as it might be needed for working out Commands as a Concept.

### Date & Time

July 24, 2008 – July 28, 2008

5 days

12 hours of work

### Products

The following was produced:

**32** articles were produced

**2** articles were adapted

**1** other item of work was done

*Assignment article group*

version *2008-07-28 00 2.0*

**30** articles:

*Assignment*

*Assignment in a Diagram*

*Assignment in Text Code* (=) (not finished)

*Value Assignment*

*Value Assignment in a Diagram*

*Value Assignment in Text Code*  (v=) (not finished)

*Object Assignment*

*Object Assignment in a Diagram*

*Object Assignment in Text Code* (o=) (not finished)

*Class Assignment*

*Class Assignment in a Diagram*

*Class Assignment in Text Code* (c=) (not finished)

*Interface Assignment*

*Interface Assignment in a Diagram*

*Interface Assignment in Text Code* (i=) (not finished)

*Object Reference Assignment*

*Object Reference Assignment in a Diagram*

*Object Reference Assignment in Text Code* (=🡪) (not finished)

*Object Reference Object Assignment*

*Object Reference Object Assignment in a Diagram*

*Object Reference Object Assignment in Text Code* (o=🡪) (not finished)

*Object Reference Class Assignment*

*Object Reference Class Assignment in a Diagram*

*Object Reference Class Assignment in Text Code* (c=🡪) (not finished)

*Object Reference Interface Assignment*

*Object Reference Interface Assignment in a Diagram*

*Object Reference Interface Assignment in Text Code* (i=🡪) (not finished)

*Alternative Assignments*

*Alternative Assignments in a Diagram*

*Alternative Assignments in Text Code* (not finished)

Object articles: **(2)**

*Nothing*

*Nothing in a Diagram*

(Only added to the development version of the Objects article group.)

Adapted articles: **(2)**

*Basic Diagram Elements*

(Added the wavy line.)

*Sub-Object*

The definition of 'sub-object' does not seem to be right anymore, because the Object Reference article said, that object references are no longer called sub-objects.

Other work: **(1)**

Isolate Storage Principles

(Has nothing to do with the project, but It might be nice to do that too.)

### Project Steps

The notation for assignment was made simpler.

Command assignments was made the same as object assignments.

It was extended with object reference assignments and alternative assignments.

Details:

- Old set of articles' organization:

- Assignment

- Assignment in a Diagram

/ Value Assignment

- Value Assignment in a Diagram

/ Object Assignment

- Object Assignment in a Diagram

- Command Definition Assignment

- Command Definition Assignment in a Diagram

- Class Assignment

- Class Assignment in a Diagram

- Interface Assignment

- Interface Assignment in a Diagram

- Command Interface Assignment

- Command Interface Assignment in a Diagram

The set of articles was changed.

At first there was direct assignment of the following aspects:

- Value

- Object

- Class

- Interface

- Command Definition

- Command Interface

If commands become aspects, this might automatically make Command Definition the same as Class, and Command Interface the same as Interface. That might leave us with the following aspects:

- Value

- Object

- Class

- Interface

It might be a good idea, to mention it in the articles, where an assignment type is also synonym for a command assignment.

Pointer assignment might add more types of assignment. Object, Class and Interface assignment each might get two types of assignment:

- Object assignment:

Assigning the object as the target

- Object reference assignment:

Assigning the related object as the target

Any alternative form of assignment might be accomplished by combining calls to system commands and assignment commands.

This would create a different set of articles:

- Assignment =

- Assignment in a Diagram

- Assignment in Text Code (not finished)

- Value Assignment v=

- Value Assignment in a Diagram

- Value Assignment in Text Code (not finished)

- Object Assignment o=

- Object Assignment in a Diagram

- Object Assignment in Text Code (not finished)

- Class Assignment c=

- Class Assignment in a Diagram

- Class Assignment in Text Code (not finished)

- Interface Assignment i=

- Interface Assignment in a Diagram

- Interface Assignment in Text Code (not finished)

- Object Reference Assignment =🡪

- Object Reference Assignment in a Diagram

- Object Reference Assignment in Text Code (not finished)

- Object Reference Object Assignment

- Object Reference Object Assignment in a Diagram

- Object Reference Object Assignment in Text Code o=🡪 (not finished)

- Object Reference Class Assignment

- Object Reference Class Assignment in a Diagram

- Object Reference Class Assignment in Text Code c=🡪 (not finished)

- Object Reference Interface Assignment

- Object Reference Interface Assignment in a Diagram

- Object Reference Interface Assignment in Text Code i=🡪 (not finished)

- Alternative Assignments

- Alternative Assignments in a Diagram

- Alternative Assignments in Text Code (not finished)

It may be an idea to take the following details into consideration:

> It may be possible to assign a class to an *object reference*, but also to assign a class to an *object*. Which is applied when and how?

> It only seems to be described how to assign it to an object reference.

> Perhaps a class can be assigned to an object reference and then create the object reference. It is probably not possible to change the class of an object after it is created.  
This might be mentioned this in class assignment.

> It seems, an object is always accessed through an object reference, and an object reference is always part of the object reference’s parent. So it seems an object is always accessed object through another object.

> What could assigning a pointer to a class as the class of a symbol look like, instead of assigning a class to a symbol?

> Perhaps the normal case is to just assign the class of a symbol to another symbol. Possibly, if that class is a pointer to a pointer to a class, it is the pointer to the pointer to the other symbol that is accessed. Assigning the target class may need to be another type of assignment.

> In that case, it may be an idea to leave out the term target in the normal assignments, and add a new set of possible assignments: target object assignment, target class assignment, etcetera.

- Alternative assignments, not target assignments

What about assigning the Target Class as the class target of an object reference? This might be an option: c=c🡪🡪

Maybe the same can apply to interfaces.

But the following could be done too:

A c=🡪 B . Target Class

The diagram was worked out on paper.

Object reference assignment seemed a useful part of the solution.

But target object reference assignment does not seem needed in the solution. Target object reference assignment seems less common, and possible in an alternative way. Target object reference assignment also lead to thinking about assigning other types of targets and it might clutter the language with a bunch of alternative notations for things that are less common, for which a general notation could be used.

> Commands and objects more the same

Class and definition assignment would be the same concept in that case, and then there would be only one type of interface assignment.

> Calls seem created briefly, but definitions seem created permanently.

- How does that influence the different types of assignment?

- It seems to mean, definitions could get object lines anyway, and calls can not.

- Articles that might change:

- Object assignment

- Object reference object assignment

- Class assignment

(possibly rephrase small parts)

- Object reference class assignment

(possibly rephrase small parts)

- Incorporate:

To make direct conversion between a command and an object possible without any loss of structure, a call maybe should be able to have an object line. In that case the rule might be: when a call symbol has an object line, the object line behaves as a class line.

But does that not require a warning to be generated?

- Explained unsimplified notations of assignment

Perhaps display the notation of an assignment calling upon system commands.

It was not clear how to do it before, but now most seems known about the system interface, it seems clear enough now, how to express it.

Object assignment:

(Picture -1)

B . Object . Set ( A . Object . Get )

(Picture 0)

B . Object . Set ( A . Object . Get . Value )

(Simplified) (The words Get and Set might not need to be shown in normal assignment, but a general assignment notation for this was not introduced.)

B . Object = A . Object

(Picture 2 ½)

o= ( B , A )

B o= A

(Picture 3)

B o= A

Object reference object assignment:

B . Other Related Item . Set ( A . This )

B . Other Related Item . Set = A . This

B o=🡪 A

- Articles that might change:

- Assignment

- Assignment in a Diagram

- Assignment in Text Code

- Object Reference Assignment

- Object Reference Assignment in a Diagram

- Object Reference Assignment in Text Code

- Wrapped it up with more references to other articles:

- In the main article referred to all the sub-articles.

- In all the sub-articles referred to the main article.

- Maybe add references to the articles target objects, target classes and target interfaces

> Only referenced target objects in the article Alternative Assignments.

/ In Target Object Reference Assignment and Object Reference Assignment maybe refer to the article Automatic Containment for an explanation on imaginary reference lines.

- Perhaps put all references to other articles in the see also list.

- Class and interface assignment:

If an object does not have a class,

then the object itself might be assigned as a class.

If an object does not have an interface,

then the object itself might be assigned as an interface