## Circle Language Spec Plan, Parameters Spec, Project Summary

*Author: JJ van Zon*

*Location: Oosterhout, The Netherlands*

*Date: September 4, 2008 – October 1, 2008*

### Goal

Work out the Parameter articles, as part of working out Commands as a Concept.

### Super-project

This project used to be part of the project ‘Command As A Concept’, which proved to be too large, so it was split up into multiple projects.

### Time

September 4, 2008 – October 1, 2008

**4** weeks

**60** hours of work

### Background

Parameters are the essential connection between commands and objects. If I can use a line merge in bidirectional relations between classes, does that mean that a line merge between a command parameter’s class line and a class command’s definition line should be considered a bidirectional relation between a class and a command as well? Should that be viewed more like a relationship, more than anything else?

When a command definition gets an object with a class, this creates a relation between the command and the class definition. This means, that the class definition also gets a reference back to the command.

It looks like, when you turn a command into an executable object, without any additional effort automatically establishes the interchangability between class commands and command parameters. I didn’t expect that. If two seemingly independent concepts confirm eachother, I must be on the right track.

### Products

At the beginning of the project, the amount of articles to produce or adapt was 47.

Eventually the amount of articles produced or adapted was **27**.

The amount of articles shrunk, but the amount of *work* did not.

The following was produced:

**27** articles

*Parameters documentation section*

version *2008-10-01 00 1.0*

**24** articles

*- Relations Between Commands & Objects*

*- Relations Between Commands & Objects in a Diagram*

*- Access Controlling System Aspects*

*- Access Controlling System Aspects in a Diagram*

*- Simplified Access Control Expression*

*- Simplified Access Control Expression in Text Code*

*- Simplified Access Control Expression in a Diagram*

*- Joint Display of Access Connectors & Object Relations*

*- Legacy Parameter Concepts*

*- Required & Optional*

*- Required & Optional in a Diagram*

*- Required & Optional in a Text Code*

*- Variable Amount of Parameters*

*- Variable Amount of Parameters in a Diagram*

*- Variable Amount of Parameters in Text Code*

*- Return Values*

*- Return Values in a Diagram*

*- Parameter Order*

*- Parameter Order in a Diagram*

*- Parameter Order in Text Code*

*- Miscellaneous Parameter Topics*

*- Parameters & Arguments*

*- Parameters & Arguments in a Diagram*

*- Parameters For The Add Command*

*System Objects articles*

3 articles

*- Parameters For Objects*

*- Parameters For Objects in a Diagram*

*- Parameters For Objects in Text Code*

The original list was **47** articles, which included:

- Value topics

> Moved to Assignment

- Old parameter passing type subdivision

That used the terms input and output

> Moved to Advanced Command Topics

- Legacy and misc parameter topics

> Also there in the eventual list

The original list did not include the idea of Joint Display of Access Connectors & Object Relations.

### Means

It was ignored, that parameter notations would have discrepancies with automatic containment.

It is ignored, that the notation of automatic containment for relations is not worked out yet.

Important phrases out of the super-project description:

- Form a clear idea in your head

- Always look at the broader view too.

- Put more effort into it when the approach changes

- Do not go for producing the articles as fast as possible.

- Do not dismiss ideas, just because they are not easy to work out.

### Reflection

In this reflection I will sum up which turns I needed to take in the project.

At first there were the following pieces of material to organize and turn into a single Parameter documentation section:

- Commands & Classes Loosely coupled

An extensive story about the concept of commands & classes loosely coupled and all sorts of implications on interface reliability and the internet

- Older parameter type subdivision

That was supposed to be converted to the eventual article list, but it was not.

Value In

Value Out

Value Thru

Reference In

Reference Out

Reference Thru

Object Out

- Older, simple descriptions of the terms parameters and arguments

Parameters

Parameters in a Diagram

Arguments

Arguments in a Diagram

- A possible parameter article list:

This eventually became the legacy and misc topics

First all the material was read over and comment was dropped into it.

I rephrased and reorganized the texts of the older parameter type subdivision, that was supposed to become the eventual article list, but it didn’t.

Then I read over the Commands Basics documentation section in search of ideas about more parameter passings. The parameter passings that needed to be added had to do with the class aspect of a parameter and the fact that parameters can also be commands. I thought I had a clear spectrum of the possibilities.

Then I started trying to convert the idea of Commands & Classes Loosely Coupled into Relations Between Commands & Objects.

Quite early I become completely confused. When you look at parameters as relations between commands & objects, then object parameters usually point outwards. I could not match it to the reference situations I had thought of: pointing *to* a parameter and let the parameter point to something.

After a lot of brainstorming the whole idea landed on its feet again. The display of relational structure between commands and objects will be displayed separate from the parameter passings: access connectors of the ways you can access the parameter. By totally separating those two things, the system could function again. Most of it went very smoothly then, but after working out some topics smoothly, I had a hard time coming up with simplified parameter access control literals, and eventually I actually gave that up and just described my existing unfinished ideas about simplified parameter access control literals.

Next, most of the material from Commands & Classes Loosely Coupled was moved to the documentation sections Interfaces and Public & Private.

Next I had a hard time crossing out the articles about old parameter passing types. At one point I even thought I had a good plan, but when I started at it, the whole plan fell apart. The thing was: they all talked about input and output and I had decided, that I could not really define the terms input and output yet. I was not sure whether output was what’s read on the outside, or output was what’s written on the inside, and I figured or thought it was not always both.

Eventually I decided to move most of the sections about the old parameter passing types to the Advanced Command Topics, and decided the whole definition of the terms input and output should be determined in the project of working out the Advanced Command Topics, because those already go deeper into input and output.

Sections of the old parater passing types, that were about value transfer, were reformulated and moved to the Assignment topics.

Then the remaining brainstorm texts in the project description were crossed out. Lastly, the even older parameter articles were crossed out.

Then there was a simpler phase: documenting all the legacy and miscellaneous parameter concepts. That was actually simple. I was able to finish almost all of it in one evening when I was very very tired.

Lastly, I made a good start at the miscellaneous topic Parameters For Objects, but moved that to the documentation section System Objects.

The last phase in the project, was wrapping up the project.

In this project things were also stressful at work. It influenced the amount of time I could put in it, and it influenced how clear I could think.

It was a very hard project. I had to wreck my brain to come up with the right ideas.

### Project Steps

#### Preparation work Parameters

(0h)

- Do consider the general approach, described in the super-project  
*New Computer Language Functional Design*

- Commands & classes loosely coupled is moved to the parameter articles

- Record the amount of articles, as planned to be written at the beginning of the project

#### Read over existing texts

(9h)

- Read over the material and drop comment into it

except Commands & Classes Loosely Coupled

- Read over Commands & Classes Loosely Coupled

- Read it over

- Verbal brainstorming

- It will be used as a cross out list later.

#### Prepare texts

- Adapt parameter texts

- The Input, Output, Throughput, Downput introduction covers too many details, that have to moved to the more specific articles, because they are later repeated there.

- Remove the separation between direct and indirect value transfer

- I did not mention anything about reference situations, before the parameter passing summary

> Renamed *Parameter Passing Summary* to *Three Parameter Passing Elements*

> Came up to Object Out in a Diagram

- Remaining ideas will only be worked out later

- Extend with more types of parameter passing:

- Based on Command Basics

- Gather up issues

- The issues:

- The issues have to do with:

- Parameters that are commands

- Classes

- Specific issues:

- Parameters that are command references

- Parameters that are output commands

- Existing command

- New command

- The class of a parameter

- The class of a parameter, that is a reference

- Public active command references inside commands

- Sub-commands are never referenced / sub-commands are never output objects

- Issues were partly written about.

- But more issues were found.

- Extend with more types of parameter passing:

- Actually, there are even more possibilities

- The spectrum of it is partly worked out on paper.

- But now I just want to start at the beginning of the story.

- Rewrite Commands & Classes Loosely Coupled

- It will be called ‘Relations Between Commands & Objects’

- It is not clear yet exactly how the article set will look or the parameter types at all.

- First go about relations between commands & objects

- Then go about all the different reference situations a parameter can be in

- Then go about how you can access control each situation

- And why to access control each situation

(object algebra, deduction, concurrency resolution, control by user)

(only command IO can lead to that, actually… anything about sub-object acces is not covered. That will all be covered in Advanced Command Topics)

- The use of each reference situation

- Then go about access control literals

#### Changed approach

The whole approach was changed, because starting at Relations Between Commands & Objects the whole idea about refence situations was messed up. After a lot of brainstorming the idea landed on its feat again.

- Preparation:

- Brainstorm about different approach

- Isolate version of unfinished article group

- Articles and work:

- Relations Between Commands & Objects

- Relations Between Commands & Objects in a Diagram

- Object relations

- Class relations

- Access Controlling System Aspects

- Access Controlling System Aspects in a Diagram

- Simplified Access Control Expression

- Simplified Access Control Expression in Text Code

- Simplified Access Control in a Diagram

> I also want to see all the possible access connectors with the new textual literals.

- The Data aspect

> I forgot about indirect value access from the inside

> It should be seen as another system aspect.

- Joint Display of Access Connectors & Object Relations

#### Cross out old material

- Cross out ideas in project document

- Cross out Commands & Classes Loosely Coupled

- Because the whole idea of Commands & Classes Loosely Coupled will be based on relations between commands and objects.

- Most texts have been moved to other article groups:

- Interfaces

- Public & Private

- Cross out old parameter passing type articles

- Read over brainstorm sections near the end, that I made before I changed the approach. Probably delete all those texts.

> Two pieces of text were usable in other article groups.

> And two sections could become an article in the Parameters article group.

- Set aside assignment topics

- Topics:

- Clone

- Value Direction

- Indirect Value Transmission

- Move old parameter passings to Advanced Command Topics

because they talk too much about In, Out and Thru, and the definition of those terms is harder to do, and has too much to do with the Advanced Command Topics.

- Added it as an objective to the project Work Out Advanced Command Topics

- Put a disclaimer in the articles for Simplified Access Control Expression about the use of the terms In Out and Thru.

- Put in the plan to adapt the articles Simplified Access Control Expression, being more reserved about the terms In, Out and Thru

- Assignment related

- Write Clone section for the assignment articles

- Move to the assignment articles:

- Clone

- Value Direction

- Move to the Advanced Command Topics:

- Indirect Value Transmission

- Write something about cloning access connectors

- Register project Update Assignment Articles

- Cross out remaining ideas in the old parameters document

- Change the articles Simplified Access Control Expression, being more reserved about the terms In, Out and Thru.

- Wrap up brainstorm texts in this Project document.

- Cross out older parameter articles

- The older articles contain mostly information about Commands & Objects Loosely Coupled, and will be integrated into the story.

#### Misc articles

- Mainly texts of miscellaneous articles still need to be written

- Write them: (**19**)

- Legacy Parameter Concepts

- Required & Optional

- Required & Optional in a Diagram

- Required & Optional in Text Code

- Variable Amount

- Variable Amount in a Diagram

- Variable Amount in Text Code

- Return Values

- Return Values in a Diagram

- Parameter Order

- Parameter Order in a Diagram

- Parameter Order in Text Code

- Miscellaneous Parameter Topics

- Parameters & Arguments

- Parameters & Arguments in a Diagram

- Parameters for the Add Command

(preliminarily finished and moved to the System Objects documentation section:)

- Parameters For Objects

- Parameters For Objects in a Diagram

- Parameters For Object in Text Code

- Read them over

#### Process leftover ideas

#### Wrap up

- Give articles a parameters-specific article name.

- Set up article list

- Isolate version

- Turn Perparation work Computer Language General into a separate project.

- Make the next project also an Organize Computer Language Ideas project.

- Wrap up project description

- Reflect on how this project progressed