## Circle Language Spec Plan | Input Output Spec | Project Summary

*December, 2008*

### Goal

Work out the Input Output article group.

This includes the exact specification of the terms input and output. This should lead to the definition of how automatic execution order can be established. Also, there will be looked at, to which extent this all can solve concurrency issues.

### Super-Project

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

### Products

The product list below is outdated and needs to be redone.

The following articles are to be produced:

Influential concepts:

- Events

- Versioning

Advanced Command Topics:

- Automatic determination of parameter in / out / thru

(do start listing out topics in a general Flat & Structured interchange article)

- Command IO

- Command IO in a Diagram

(taken out of the automatic execution order section)

- Accessing parameters’ sub-objects

- Specific data unknown

- Parameters & IO

- Sub-commands’ IO

- Pre- & Post-conditions

- Conditions

- Conditions in a Diagram

- User Commands

> If executions are only executed by parent executions, then what is the parent execution of the parent execution of the parent execution? Well, the upper parent command is actually comes from a human being. A person can execute a command definition.

Flat & Structured Interchange:

- Commands & Classes Loosely Coupled

Redo, making it a real relation between commands and objects.

Subdividing the commands from different site into different interfaces of an object, is a separate issue to address.

The use of object lines and class lines for commands has changed.

Apply commands & classes loosely coupled to basic command articles.

- Automatic Execution Order

(I am going to have to take a lot of these topics out from under automatic execution order. Because many things are just facilitating and handy for other stuff too. But it’s not priority nr. 1 to do that.)

- Parameters of calls directly tied together

- Parameters tied together

- Parameters tied to objects

- Outcome dependency

- Compared IO

- Implementation of System Commands: **(1)**

You may need to add separate explanations about the implementation of the system commands.

- Scheduling & Waiting

### Project Steps

<…>

- Finally:

- Update the article Flat & Structured Interchange

- Update the articles Coding Principles and Coding Concepts

### Brainstorm

At one point I realized that Parameters (that may be either in/out or thru) might not be what determines input output. I started to suspected input/output just would mean anything read vs anything written. Parameters can be part of that, but internally a procedure may as well write other stuff (e.g. to a data source). But that does not mean it would not be useful if the concept of input/output is also reflected onto parameters, because they often do have a lot to do with it. Also, functions that only read and write parameters, not read anything else, I think those are called pure functions or sometimes functions with no side-effects/ It's an existing concept. But were I to explain (to myself) how that compares to my ideas about Input/Output, I expose it in a theoretical framework how everything compares to each other, might help me. For instance, some parameters can be called input or output parameters. To realize that the parameters are potentially not the only input/output that a procedure has, would disentangle those concepts. Also: if you pass a reference to an object to a procedure, is it input or output? That question used to confuse me. But now I realize it can be both. Just look at the elements written and then elements read: those are the input and the output. So from the viewpoint of a command, the reference is input. If it uses that object reference to e.g. write properties, that is its output. What is also interesting, is something written by one procedure can be read by another, so whether something is input or output is also dependent on context. E.g. the object reference parameters: it is sort of output when a procedure caller writes the reference, I mean if you uphold the definition of output = writing, then it must be. But to the called procedure it is input, so the same piece of information switches role there, just because in that context it is only read, not written. The whole thing can be so confusing and intermixed in my view, that I think I found it a useful idea to maybe write about how all the concepts would work together precisely. To explain it to myself, maybe eventually useful for others too.