# Circle Language Spec Product List (Out of Scope)

*Author: JJ van Zon*

*Location: Oosterhout, The Netherlands*

*Date: February, 2008 – February 2020*

## Contents

Contents 2

Purpose of this Document 2

Topics Roughly 2

Topic List Detailed 4

Internet Complete 5

Other Topics 5

Concepts 5

Coding Concepts 5

Concept Libraries 5

Fundamental Principles 5

Querying & Collection Operations 5

Data Concepts 5

Overview 6

Coding Concepts 6

Handy Access 6

Circularity 7

Apply 7

Uncategorized Coding Concepts 7

Data Concepts, Trivial 7

Data Concepts, Misc 7

Internet Concepts 8

Other Expression Topics 8

Diagrams 8

Fundamental Principles 8

Exchangability Principles 8

## Purpose of this Document

This document is a raw list of the products to produce in a project. The topics mentioned here are put out of scope of the project *New Computer Language* and may be moved away to another place eventually.

## Topics Roughly

***Topics To Cover***

The only (out-of-scope) things that might be desirable to work out for the new computer language are:

- Data Concepts

- Text Code

For the internet to function as a single computer, also the following could be worked out:

- Internet Concepts

- Access Control

- Caching

The following non-computer-language topics could also be covered, to show the power of the language:

- Editing Concepts

- Controls Concepts

However, *Controls Concepts* and *Editing & Authoring* would not be covered anymore. Even though they are power-enhancements to the new computer language they already have a rough description that is adequate.

***Topics Not To Cover***

The rest of the ideas could be worked out by others.

These subjects might not need much explanation:

- Utilities

- Optimization

- Parser

- Other Text Codes

- Legacy Text Codes

- Text Templates

- Text Operations

There are also topics about computer language, that just go too far for now:

- Automatic Object Formation

- Object Algebra

- Neural Networks

- Natural Language

- Artificial Intelligence

It would be nice to see, if these could be given a place in the new computer language, but I would not insist on realizing that myself, and could leave that to others, if they even would.

***Fundamental Principles***

How the storage of objects is handled might also be covered in:

- Object Storage

***Borrowed***

<JJ van Zon, 2020-01: It might not be appropriate anymore to include the following information, because it seems too much based on the idea, that almost a whole operating system would be programmed. The scope of the project might be more limited than that.>

Core functionality will be borrowed from the operating system:

- Internet Protocol

- Kernel functionality:

- Threads

- Memory

- Disk

- Device drivers

< JJ van Zon, 2020-01: The following information might not be appropriate to mention anymore, because it references a document, not involved in the project anymore, that is more grander in scope: an OS, a programming language and any commonly used application.>

The new computer language consists of most of the 'Foundation Layer', as described in the document *Software System*. But 'Core' modules described there, will be borrowed from the operating system. The *Core* modules, which consitute an operating system kernel, will at first not be made. One of the ideas, is that the kernel could later be programmed inside the new computer language itself anyway, because you can program machine code in it.

## Topic List Detailed

***TODO***

#### Input Output

This phase might introduce a way to handle the input/output concept and concurrency.

##### Coding Concepts

###### Input Output

- In, Out & Thru

- Auto In, Out & Thru

- Command Input/Output

- Compared Input/Output

- Accessing Parameters’ Sub-Objects

- Specific Data Unknown

- Parameters & Input/Output

- Sub-Commands’ Input/Output

- Pre- & Post-Conditions

- Conditions

- User Commands

- Commands & Classes Loosely Coupled

- Automatic Execution Order

- Parameters of Calls Directly Tied Together

- Parameters Tied Together

- Parameters Tied to Objects

- Outcome Dependency

- Compared Input/Output

- Integration of Parameter Input/Output Concepts

*(Parameter passing types from other languages and from a previous attempt to document this language (the 'Symbol Language' documentation) but now put in the context of in/out/thru and input/output in general and extending to automatic execution order and some form of concurrency resolution).*

- Parameters

- Parameter Types

- In Parameter

- Out Parameter

- Thru Parameter

- By Value

- By Reference

- Value In

- Value Out

- Value Thru

- Reference In

- Reference Out

- Reference Thru

- Object Out

- New Object Out

- Existing Object Out

- Three Parameter Passing Elements

- Parameters of Calls Directly Tied Together

- Strict about Parameter Passings

- The Class of a Parameter

- Sub-Commands are Never Output Objects

- In, Out, Thru Parameters

- Downput Parameter

- Data Direction

- Indirect Value Transmission

- Input/Output not Always Values

###### Concurrency

…

#### Automatic Diagram Organization

##### Diagrams

- Automatic Containment

- Artificial Promotion, Esthetic Reference

- Fixed Logical Residence

**Math & Integration**

Without these libraries the usability of the language is limited.

After this phase things are usable, even though some of my much-wanted constructs are left out.

Since the relational paradigm is not supported yet, you might integrate with relational databases as a provisionary solution.

***Math***

Math will actually facility those normal expressions you are used to having in any language: arithmetic, comparison, boolean operations: expressions if you will. A programming language is incomplete without them. However, in the new computer language it is no more than a module.

Programming the Math module without Text Code actually being developed, does make you unable to generate the expression into text code, which some people may use as an execuse for calling the language ridiculous or incomplete.

***Integrate***

To really prove the language’s power, you might want to be able to navigate existing systems with it. So you might want to be able to express file systems, other text codes, other module systems, relational database, web services and everything in the diagram language and navigate through the systems, navigating through a single diagram. It is not very easy to realize that. The problem is the multiplicity of it: you would have to write an integration module for each system you want to integrate with. But if you could just prove the concept with a couple of systems, or your favorite systems, it may inspire people to say: if you make these integration modules this would be a really really valuable thing.

#### Internet Complete

In this phase the essential parts to make the internet function as a single computer are completed.

##### Other Topics

- Storage \ Caching

- The Physical & The Logical

#### Concepts

This phase finally introduces the concept libraries , which might replace code generation by a better solution.

##### Concept Libraries

A lot of concepts already have a good functional description.

But at this point it may already be about implementing it.

###### Coding Concepts Misc

...

###### Editing Concepts

...

###### Controls Concepts

...

###### Data Concepts

...

##### Fundamental Principles

###### Extensibility Principles

- Concepts As External Modules

- Relational As Carbon Base

#### Querying & Collection Operations

This phase is a belated introduction of the relational paradigm into the language.

##### Data Concepts

- Collection

- Filters, Sorts, Searches, Joins & Indexes

- Collection

- SQL & Tables

- Collection Operations

**Politically Correct**

These topics are far less interesting to me, but could be introduced for political correctness’ sake, because otherwise people will call it incomplete.

***Data Concepts***

- Binaral

(for the purpose of compiling)

***Other Topics***

- Access Control

(These are also important for turning the internet into a single computer.)

***Postponed***

#### Overview

Trivial, Less Important:

- Create Objects, Ensure Objects

- Attributes as Tree Layers

Nice To Have:

- Concept Construct

- Coding Concepts Misc: Handy Access, Circularity, Apply

- Data Concepts Trivial

- Data Concepts Misc

- Text Code

- Internet Concepts Trivial

#### Concept Construct

This phase finally introduces the concepts coding construct, which might replace code generation by a better solution. The concept is pretty much equal to the aspect construct known from aspect oriented programming.

- The Concept Construct

#### Coding Concepts

##### Handy Access

- Handy Access

- Progression

- Dedimensionality

- Skipping Structure Layers

##### Circularity

- Circularity

- Circularities

- Circularity Handling

- Pointer Circularity

##### Apply

- Apply

##### Uncategorized Coding Concepts

- Retry, Skip Stop

#### Data Concepts, Trivial

- Clear

- Registration Lists

- Item Remove

- Undouble

- Undouble

- Undoubled Relations

- Search

- Find

- Find Or Add

- Persistance Delay

- Apply, Ok, Cancel

- Custom Sorting Methods

- User Sorting

- Multi-Lingual

- Is Deleted

- Attributes As Tree Layers

- Defaults:

- Default Values

- Create Objects

- Ensure Objects

- Boolean & Enum Methods

- Delete only when no references

- Extended Multiplicity

#### Data Concepts, Misc

- Search

- Wildcards

- Import & Export

- Import & Export

- XML

- Persistence Delay

- Locking

- Transactions

- Streaming Access

- Strings

#### Internet Concepts

- Mirroring

- Copies

- Internet Synchronization

- Internet Threads

- Publishing

- Backup Machine

- … (more to come)

**Text Code**

- Literals

- Object Literals

- Command Literals

- Concept Literals

- Collection Operation Literals

- Access Operators

- Declaration on first use

- Half-case-sensitive identifiers

- Macro Keywords

- Pointer Operators

Basically each element of the language specification has to be given a textual representation as well.

#### Other Expression Topics

- More Text Codes

- Additional Text Codes

- Math Code

- Assembly

- Legacy Text Codes

(the other topics will not be covered at all)

#### Diagrams

- Diagrams As A Concept

- UML

#### Fundamental Principles

##### Exchangability Principles

- Object Oriented = Relational

- Hardware & Software

- System Engineering = Software Engineering

- Internet as a single computer

Topics:

- A unified paradigm for all digital objects

- A single bulk of storage

- Execution of programs running across machine boundaries

- Security

- Control of concurrent use

- Communication between computers

- ID’s

- Site merging

- Parallel processing

- Mirroring & synchronized copies

- Implications for other concepts

- Legacy modules

- Flat & Structured Interchange