# New Computer Language Products

***-***

*Author: Jan Joost van Zon*

*Location: Oosterhout, The Netherlands*

*Date: February, 2008 – January 2020*

## Purpose of this Document

This document is a raw list of the products to produce within the project *New Computer Language*. The goal and approach of the project were already described in the document *New Computer Language Strategy.*

## Topics

### Topics To Cover

The only things desirable to work out for the new computer language are:

- Fundamental Principles

- Coding Concepts

- Data Concepts

- Diagrams

- 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* will 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 parts of the 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

The *Fundamental Principles* are more or less the *demands* for the language.

They are very abstract. The main groups of topics are:

- Achievability

- Exchangability

- Extensibility

They also explain how everything might work in harmony to achieve these goals.

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

- Object Storage

Fundamental Principles also covers the basics of how you will be expressing yourself in the new language:

- Diagram & Text Code Expression

### 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

A sub-project may consist of documenting one or more concepts of a certain topic.

The list is not ordered by reading order, but rather by writing order, which is roughly the same as being ordered by level of difficulty.

The reason this list is so massive, is because of the split up into tiny little topics.

Work Towards Implementation

Instead of working these topics oud conceptually in writing, these topics might be worked out in projects for programming an actual implementation of the language.

So these are basically postponed as a documentation project.

##### Coding Concepts

###### Static

The concept of static needs to be worked out to a simple extent

###### Type Control

- Type Control

- Implementation

> Not sure if I will even do that. Might exclude this.

##### Other Requirements

Data Concepts:

- Object Order

##### Diagrams

- Diagram Metrics

- Part (technical) design

- Implementation

Coding Essentials

After this phase the coding concepts are as complete as any OO language.

##### Coding Concepts

###### Object Resolution

- Object Resolution

- Overriding

- Overloading

- Shadowing

- Melding

~ Hiding

- Interface Resolution

- Default Member

- Cascading Overriding

- Ambiguity

- Implicit Conversions

- With Blocks

###### e Control

- Type Control

- Functional Design

###### Classes, Relations & Static

- Static

Static could to be worked out

and classes & relations need a redo accordingly.

##### Other Requirements

Relations:

- Relation Direction

Data Concepts:

- Enums

Objects:

- Multiplicity

> You might want to be able to introduce new kinds of lists and express and use them as any other list.

> You might want to be able to introduce new basic data structures and give them the nonagon symbol, and have different kinds of possible indexers, etcetera.

Automatic Diagram Organization

The coding concepts are complete, but without the diagrams being organized automatically, the effectivity of the language is considerably limited.

##### Diagrams

- Automatic Containment

- Artificial Promotion, Esthetic Reference

- Diagram Metrics

- Functional Design

Evaluate

This may be a crossroads at which you migh switch to implementation or patenting and releasing it into the world.

A benefit of making a working version is that it immediately becomes clear what are the most important open issues, it proves all the concepts and it probably gives you a clearer picture of the language which helps to complete other topics. It also allows you to implement each thing you design, because you have the base for it. It also would be really cool.

I think that some chapters (main modules) need to be split up, for instance into Controls A and Controls B, to cover the controls concepts at the top of the documentation, while the not-worked-out topics are further down in the documentation. That way you will end up with finished up work at the top of the documentation. That makes it easier to isolate a production version.

Math & Integration

Without these libraries the usability of the language is limited.

After this phase things are patentable and 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.

Evaluate

This may be a crossroads at which you migh switch to implementation or patenting and releasing it into the world.

If you release it, you continue your plan, but you do not know what the world is going to make of it.

However, the language could do without anything put here below.

Everything above makes it a complete and usable language.

An advantage of releasing it, is that from then on you might never be lonely working on the new computer language again.

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 concepts coding construct, which replaces code generation by a better solution.

##### Coding Concepts

###### Concepts

- Concepts

- < … >

##### 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.

##### Coding Concepts

###### Errors

- Errors

- Warnings

- Structure Errors

##### Data Concepts

- Binaral

(for the purpose of compiling)

##### Other Topics

- Access Control

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

IO

This phase would introduce a ‘revolutionary’ way to handle the input/output concept and concurrency.

##### Coding Concepts

###### Input Output

- In, Out & Thru

- Auto In, Out & Thru

- Command IO

- Compared IO

- Accessing parameters’ sub-objects

- Specific data unknown

- Parameters & IO

- Sub-commands’ IO

- Pre- & Postconditions

- 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 IO

- Legacy Parameter IO

(from old parameter passing types)

- 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

…

#### Postponed

These topics are not essential or they are adequately, but not fully described already.

##### Overview

Trivial, Less Important:

- Create Objects, Ensure Objects

- Attributes as Tree Layers

Nice To Have:

- Namespaces, Aliases

- Misc Coding Concepts: Conversion, Conditions, Handy Access, Circularity, Apply

- Data Concepts Trivial

- Data Concepts Misc

- Diagram Topics Misc

- Expression Misc

- Text Code

- Internet Concepts Trivial

- Fundamental Principles Already Worked Out

- Uncategorized Coding Concepts

- Language Summary

##### Coding Concepts

- Introduction

###### Identifiers

- Namespaces

- Aliases

###### Conversion

- Conversions

###### Conditions

- Conditions

- On Commands

- On Object Values

###### Handy Access

- Handy Access

- Progression

- Dedimensionality

- Skipping Structure Layers

###### Circularity

- Circularity

- Circularities

- Circularity Handling

- Pointer Circularity

###### Apply

- Apply

###### Uncategorized Coding Concepts

- Special Access

- Global Access

- Clause Access

- Interface Access

ultiple Language Layers

- This

- Redirection

- Retry, Skip Stop

Where did these go?

- Value Direction

- Clone

###### New Computer Language Summary

##### 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

- Expression Mixing

- Specialized Expressions

- 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

- Coloring

- Abusing Diagram Expression

- Abstract Diagram Expression

- UML

##### Fundamental Principles

The list of fundamental principles will be extended in a future project, but not written straight away then.

- Introduction

###### Exchangability Principles

- Introduction

- Diagram & Text Code Expression

- Object Oriented = Relational

- Attributes Are Objects

- Command = Executable Object

- Hand Signs

- Hand Writing

- 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

###### Extensibility Principles

- Introduction

###### Achievability Principles

- Introduction

#### Finished

##### Coding Concepts

###### Identifiers

- Identifiers

- Omitted Identifiers

###### Objects

- Objects

- Sub-Objects

- Object Reference

- Related Objects

- Nothing

- Values

- Multiplicity

- Attributes

- Target Objects

###### Classes

- Classes

- Class Commands

- Class Reference

- Target Classes

###### Relations

- Referrers

- Class Referrers

- Related Classes

- Relations

- Dual & Unary

­

­­- Relations Between Objects

- Relation to a Pointer

- Relation Synchronization

###### System Objects

- System Objects

- System Aspects

- Object-Bound & Reference Bound Aspects

- System Commands

- System Commands for the Referene Aspect

- System Commands for the Object Aspect

- System Commands for the Class Aspect

- The Extra Commands & Overloads

- System Commands for the Value Aspect

- System Commands for the Clone Aspect

- System Commands for the Name Aspect

- System Commands for the Data Aspect

- System Commands for the Execute Aspect

- System Commands for the Existance Aspect

- System Commands for the List Aspect

- System Interfaces

- System Interface of an Object

- System Interface of a Related Item

- System Interface of a Related Item

- System Interface of a Related List

- System Interface of a Related List Item

- Use-Command Gets Another Aspect

- Aspect-In-A-Triangle

- System Interfaces of Objects & References

- Preliminariness of System Interface Notation

- Assignment

- Pointer Assignments

- Assignment With Pointer Source

- Cross-Aspect Assignments

- System Command Call Notations

- System Interface Comm Call Notation

- Simplified System Comm Call Notation

- System Comm Call with Argument Notation

- Explicit Get & Set Notation

- Explicit Get & Set Arguments Notation

- Completely Explicit Display

- Connectors

- Connections

- System Command Calls by User

- Objects Floating Around

- System Command Extension

- Parameters For Objects

- Ancestry Terms

- Assignment in Text Code

The System Objects article group needs extensions in the future.

It needs to be revisited to make complete the set of system aspects and system commands and cross out remaining ideas and topics.

###### Commands

- Commands

- Commands Implementation (not finished)

- Execute Once

- Executable Command

- Inactive Command

- Command Definition

- Command Call

- Command Reference

- Clause

- Inactive Clause

- Active Clause

- Start & Stop

- Start & Stop Implementation (not finished)

- Command Anywhere

- Changing Inactive to Executable

- Sub-Commands

- Command References Inside Commands

- Executables & Executions

- Procedure

- Resolution When Not Allowed For Commands

- Parent Controls Its Sub-Executions

- Sub-Commands Are Never Referenced

- Beware of Active Command References

- Sub-Commands Not Manually Started

- Public Inactive Clause = Command Out Parameter

- Creation Behavior of Calls

- Creation Behavior of Clauses

- Creation Behavior of ‘Inactive Calls’

- No Overhead Of Command Creation

- No Circular Command Creation

- No Private Contents in a Call in a Definition

- A Call in a Call Shows Privates When Running

- Active Command in Inactive Command

- Reading & Writing Parameters

- Comparison to CPU-Like Calls (not finished)

- Example Diagrams (folder)

- Command Calls in a Command Definition in a Diagram

- Command Calls in a Command Call in a Diagram

- Command Calls in a Command Reference in a Diagram

- Parameters in a Command Definition in a Diagram

- Parameters in a Command Call in a Diagram

- Parameters in a Command Reference in a Diagram

- Private Objects in a Command Definition in a Diagram

- Private Objects in a Command Call in a Diagram

- Private Objects in a Command Reference in a Diagram

- Command References in a Command Definition in a Diagram

- Command References in a Command Call in a Diagram

- Command References in a Command Reference in a Diagram

- Clauses in Clauses in a Diagram

- Inactive Clauses in a Command Definition in a Diagram

- Inactive Clauses in a Command Call in a Diagram

- Inactive Clauses in a Command Reference in a Diagram

- Active Clauses in a Command Definition in a Diagram

- Active Clauses in a Command Call in a Diagram

- Active Clauses in a Command Reference in a Diagram

- Command Calls in an Inactive Clause in a Diagram

- Command Calls in an Active Clause in a Diagram

- Parameters in an Inactive Clause in a Diagram

- Parameters in an Active Clause in a Diagram

- Private Objects in an Inactive Clause in a Diagram

- Private Objects in an Active Clause in a Diagram

- Command References in an Inactive Clause in a Diagram

- Command References in an Active Clause in a Diagram

- Inactive Command Object Redirection

- Inactive Command Class Redirection

- Executable Command Class Redirection

- Executable Command Object Redirection

- Target Command Object

- Target Command Definition

- Recursion

- Implementation of System Commands

- Command Object Referrers (not finished)

- Command Definition Referrers (not finished)

###### Parameters

- Relations Between Commands & Objects

- Parameter Passing

- Joint Display of Access Connectors & Object Relations

- Legacy Parameter Concepts

- Required & Optional

- Variable Amount of Parameters

- Return Values

- Parameter Order

- Miscellaneous Parameter Topics

- Parameters & Arguments

- Parameters For The Add Command

The article group might need a redo in the future.

###### Globality

- Globality

###### Execution Control

- Execution Control

- Conditional Execution

- If

- Else If

- Select Case

- Select Case (exact value)

- Select Case (split formula)

- Loops

- For

- For (range)

- For (conditional)

- For Each

- While

- Until

- Jumps

- Normal Execution Order

- Label & Goto

- Call & Return

- Exit Command

- Loop-Related Jumps:

- Continue

- Exit Loop

###### Black Box

- Introduction

- Black Box Main Concepts

- Public & Private

- Friend Declaration

- Notations of Private

- Friend Declaration in Instances

- Public & Friend Connections

- Access Controlling Aspects

- Object-Bound, Reference-Bound & Access Control

- Public & Private Connectors

- Public & Private Connections

- Public & Private Assignment

- Assignment between Friend Members

- Friend = Wavy Line

- Miscellaneous Issues

- Deeper Exclusion

- Protected

- Internal

- Objects Take Over Class Access Control

- Side-Issues

- Private Names

- Inclusion

- Black Boxing and User Access Control

- Programmers and Users

- Details to Cover Last

- Details, May Not Cover

###### Interfaces

- Introduction

- Interfaces Main Concepts

The general notation

Interface = Publics

Interface & implementation

Interface = Objects Melting Together

Interface = Contract

Mutliple Interfaces

Interface Type Controls

Explicit Interfaces

Implicit Interfaces

Polymorphism

Interface Object Resolution

Inheritance

Main Usages of Contracts:

a contract = a guarantee

a contract = a contract of usage

a contract = a contract of participation

- Interface Aspect

- Interface Commands for the Interface Aspect

- Interface Commmands for the Interface Merging Aspect

- Interface Aspect in System Interface

- Interface Assignment Types

- Interface System Command Calls

- Interface Connectors

- Interface Connectors

- New Command with Interface Parameter

- Command Interfaces

- Interface Referencing and Redirectioning (examples)

- Member Grouping

- Relations Through Interfaces

- Interface Referrers

- Reliability of Interfaces

- Group By Source

- Imaginary Backward Relations

- Target Interfaces

- Mutual Commands, Mutual Interfaces (Unfinished)

- Extensive Classification with Interfaces (Unfinished)

###### Events

Basic Concepts:

*- Events Introduction*

*- Prime Event Example: Button Clicked*

*- Events Main Concepts*

Main Notation of Events

Event Procedure / Event Implementation

Event Connector

Event Raising

Call to Parent

Event Implementation & Connector

*- Event Situations*

Multi-Cast Events

Multi-Cast in Same Parent

Event with Parameters

Event from Deeper Object

*- Explicit Implementation of Event Object*

*- Explicit Interface of Event Object*

For the sender

For the receiver

*- Event Interface Reference*

*- System Events*

Changed Events

Changing Events

Get Events

Use Events

List Events

Execution & Existence Events

Pointer to Pointer Events

*- System Events in Normal Notation*

*- Simplified System Event Notation*

*- System Event Parameters*

*- System Event Design Choices*

Misc topics:

*- Qualified Event Names*

*- Interaction Events*

*- Black Boxing Events*

*- Event Alternatives*

###### Inheritance

- Inheritance Introduction

- Inheritance Main Concepts

- Class Inheritance

- Object Inheritance

- List Inheritance

- System Inheritance

- Interface Inheritance

- Specialization

- Specialization & Data Replacement

- Altering the Member Set

- Member Addition

- Member Exclusion

- Member Inclusion

- Detouring Members

- Shadowing

- Overriding

- Altering Command Implementations

- Command Extension

- System Command Extension

- System Command Overriding

- System Command Shadowing

- Sys Comm Exten By Shadowing

- Sys Comm Exten By Overr (Questionnable)

- Sys Comm Overr By Extension

- Destructive & Non-Destr Spec Methods

- Misc Inheritance Situations

Command Inheritance

Backward Relation to Derived Classes

Multiple Objects Sharing one Base Object

Inheriting Multiply from the Same Class

- Enforcing & Preventing Specialization

Protected

Overriding

Optional / Required

Requirements for Other Side of Connection

Gut Feeling

Applied to Concepts from Other Languages

Protected

Virtual

Abstract Member

Abstract Class

Sealed / Final Class

Interface / Interface Member

Non-Overridable

More Combinations

Courtesy

Enforcing & Preventing Other Specialization Methods

Preventing Pre-Extension with Cancellation

Preventing & Enforcing Data Replacement

- Deeper Specialization

Deeper Exclusion

Deeper Inclusion

Deeper Member Addition

Even Deeper Member Addition

Deeper Shadowing

Deeper Overriding

Deeper Command Extension

Deeper System Command Extension

Deeper System Command Overriding

Deeper System Command Shadowing

The Deeper Specializations Structure

- Alternate Version Through Inheritance

- Extending System Objects (Older)

(This older article was left in tact to save time.)

##### Diagrams

- Basic Diagram Elements

##### Fundamental Principles

###### Exchangability Principles

- Data = Code

- Programming Language = Database

- Designtime = Runtime

- User = Programmer

- Clear Cut Coding Principles

- Hyperlinks = Referential Structure

- User Interface Not Procedure Oriented

- Symbol = Creator

###### Extensibility Principles

- Reflection

- Module Integration

###### Achievability Principles

- Generic, No Generators

- Small Code Base

- Computer Language Programmed Within Itself

- Everything Only ( Lack Of Choice = Guarantees )

- C++