# Circle Docs Revamp Project Notes 2019-08

*Author: Jan Joost van Zon*

*Location: Oosterhout, The Netherlands*

*Date: August, 2019*

## Contents

[Contents 1](#_Toc27316836)

[Project Outline 1](#_Toc27316837)

[Introduction 1](#_Toc27316838)

[Goals 1](#_Toc27316839)

[Limitations 1](#_Toc27316840)

[Strategy 1](#_Toc27316841)

[Steps 1](#_Toc27316842)

[Notes 1](#_Toc27316843)

[2019-08-05 Brainstorm Restructuring Project Docs 1](#_Toc27316844)

[2019-08-05 Brainstorm Circle Docs Writing Style 1](#_Toc27316845)

[2019-08-05 Brainstorm Circle Language Design 1](#_Toc27316846)

[2019-08-05 Brainstorm version control and open source 1](#_Toc27316847)

[Done 1](#_Toc27316848)

[2019-08-04 Done Notes converting version folders to source control history 1](#_Toc27316849)

[2019-08-10 Done Brainstorm complexity in preserving rename history 1](#_Toc27316850)

[2019-08-11 Done Basic Math Conversion to Source Control History? 1](#_Toc27316851)

[2019-08-11 Done Controls Concepts Conversion to Source Control History 1](#_Toc27316852)

## Project Outline

### Introduction

Circle is an unfinished programming language.

I spent thousands of hours spread out over a few decades with this idea of how to express the internals of computers and programming language constructs.

Circle Docs are an unfinished programming language design. The same folder of docs contain texts about all sorts of software applications. But this project 'Circle Docs Revamp' is really about the programming language design.

### Goals

#### Main goals

* 'Soft' goals:
  + Work on project that seems to matter.
  + Keep my head occupied with something else.
* 'Hard' goals:
  + Isolate the Circle Language docs from the rest of the docs.
  + Open source the Circle Language docs.
* Low priority:
  + Get the documents in a state better accessible to others.
  + Make the documents usable for implementation.
  + Make a technical design

To highlight the challenges of how one might implement this as a working piece of software.

### Limitations

* Do not program.
* Limit it to the 'Language' part of the docs (not 'Framework', 'Operating System' or 'Applications & Media')
* Do not open source the prototype apps.

### Strategy

* Before I start changing anything, I should just read the material as-is.

(I have great difficulty maintaining this rule. I tend to immediately want to start changing things.)

### Steps

* Reorganize
  + Convert docs version folders to source control history.
  + Convert project docs version folders to source control history.
  + Categorize loose ideas in in Ideas.doc, so Circle Language and Circle 3 Programming are isolated.
* Reorient
  + Read all the project docs sequentially for thorough orientation.
* Improve project docs:
  + Reuse/reorganize project docs
  + Remove studying goals.
  + Split documentation goals from programming goals.
  + Less resolute language, more wiggle room. (Perhaps search for definites such as 'have to', 'must', 'will', 'should', 'very'.)
* Separate git repositories:
  + Split off Project Docs for Circle language documentation into a separate git repository.
  + Split off Circle Docs' language specification part into a separate git repository.
* Improve docs:
  + Reorganize Circle Docs files and folders (e.g. lone files in folders is not handy).
  + Drop loose ideas from Ideas.doc into the appropriate spot in the documentation.
  + Turn separate Concept/Diagram/Text Code articles into one.
  + Start reading / writing / reformulating.
  + Convert from Word to markdown, so it may get indexed by Google once published.

### Requirements

* Read "New Computer Language, Strategy.doc"
* Read "New Computer Language, Products.doc"
* …

## Notes

### 2020-01-13 Notes

I read and reformulated some texts from "New Computer Language, Strategy.doc". 2 hours or so. I am now too tired.

That document does not cover many things out of scope of language specification. It just briefly talks about programming experimental versions and licensing it and stuff, but little enough to keep it in there were I to isolate this into a pure language specification writing project, which I intend it to.

'Concepts' are almost exactly like 'aspects' from 'aspect oriented programming', except maybe the idea of whether just about everything can be elevated to become an 'aspect' even things that aren't the aspect oriented programming construct. Even hand-written, coded out aspects, such as those System Aspects in the New Computer Language. Cross-cutting concerns that you couldn't isolate out of the system using an aspect, but are still clearly. Can new programming constructs be found that can do that, isolate concerns like that? It is hard to express my ideas about it and explain them well. I don't even have it all clearly in my mind myself yet. Also the comparison requires I know all the details about aspect oriented programming, which I don't.

### 2020-01-04 Brainstorm

Rough plan:

* Remove detail from products doc.
* Split project docs into 2: content about the diagrammatic programming language and content outside of that subject area.
* Or read some sub projects docs.

I had those plans with it, but did nothing about them this day.

### 2019-12-29 Brainstorm

* Basically I want to scope the project.
* What do I do with things, that are out-of-scope? Do I just bluntly remove them from the documentation, or do I go through the trouble of parking the texts elsewhere?
* Out of scope:
  + Operating system components
  + Studying
  + Fundamental principles?
    - Are some of them out-of-scope?
    - Do I limit the ambitions with the project?
    - Do I remove fundamental principles that are questionable or irrelevant, like that the code base is written in C++, or things that may speed up development, things others could figure out, making how I feel about it not add much? Maybe ease up on things, talk less strict about things?
  + 'Software System': an abstraction layer above the new computer language.
    - How do I take interesting parts from the general Software System documentation and drop em somewhere in the Computer Language documentation.
  + Computer language topics out-of-scope:
    - Concepts / aspect oriented programming
    - Database principles
    - Concept libraries
    - Machine language
    - Internet as a single computer
    - So many things, but I want to leave them out.
* Would I rename 'Computer Language' to something else, admitting it is a programming language, and only expressing the hope that it would become a language to a user to, where constructs are simpler. Do I simply admit that these were my ambitions with the project, and if people claim arrogance, then let them?
* I think that somewhere along the way, the language lost its purity. Hypothetically, I may have gotten carried away a few times. For instance, using the dashed line as a conceptual expression of the idea of 'classes' or 'types': I think I tend to introduce ideas about notations that might simplify things visually, but possibly introducing ambiguity. At first, the language, was to be the purest form that I could find, in which you could draw out an object oriented system in a diagram. What happened to that as I started to make drawing something with dashed lines something ambiguous. I am not sure: this might be a non-issue. But maybe I want to be wary of where I got carried away and not think in definites about the final form of the language.
* I am hoping at some point, the project docs get smaller… because these documents are huge and intimidating.
* Maybe I should just make 2 project folders eventually in the Project Docs repository: one for the new computer language and one for the rest, that are much like eachother, but one stripped down to computer language functional design topics, and the other in which to dump the rest: anything deemed out-of-scope of the entire new computer language topic. Those are different than topics out-of-scope because postponed, but still much to do with the new computer language. Maybe at first, even 'worse', I make 2 documents in each folder: One with topics that belong to the new computer language, and another document much like it, in which the rest is put, that I would want to leave out of it.
* I think a new concept to me, introduced in this project is that: I do not need to do everything. Like this from "New Computer Language, Products.doc: "You have to be able to introduce new basic data structures and give them the nonagon symbol, and have different kinds of possible indexers, etcetera." I don't have to. I could do without. Ideas might be viable and interesting without all details being covered, without all proofs being given. I wanted to work out *everything* at some point. I also was a afraid, that if I didn't, people would not believe in the idea. Maybe I got over-ambitious, because I saw so much potential. I think I was able to work out a lot, but then I would get distracted by another project and then it turned out, I never got back to it. Scoping is a trick for that, when managing projects. Setting the boundaries and limitations of what the project would cover. I never wanted to do that back then. I wanted a framework in which everything would fit and then choose seemingly randomly what I would cover next. In one way I like the freedom of that. But on the other hand, it becomes a never ending story. I sometimes had the ambition of actually making *all* of it. I might have been able to create a playground in which I can go wild, but someone else would never want to cover all of that. Someone else would never take over your programming life, just a scoped programming project and then maybe. So I want to scope it. The design of the programming language should lose some ambition and express that only as dreams. And lose the 'programming it out' part. And loose 'it is also a framework and an OS and any commonly used application'. It is actually quite hard for me to let go of that idea. I liked my playground back then. I wanted proof, that this could be used to realize software quicker, so one man can do what would have taken an army of programmers to do before. But I don't have that ambition anymore. Right now I just want to publically give away the programming language idea. I think I notice a lot of insecurities about people thinking it is a good idea or not. Maybe because I was trying to sell the idea, rather than just give it away? I get that I wanted a framework into which all of my ideas fit. I like some of the modularization of the concepts. But I do want to just cut away a few things. I think I am still trying to sell an idea, but then in a different way. I do not have the intention to sell it for cash, but I do want to mot make it too ambitious, cover too much, so large in scope, that no one would pick it up anymore. The data concepts and coding concepts thing, and the aspect oriented-like thing, I may want to put that out of scope. I might want to accept that the idea I present has limited potential, and might not apply to what you can do with a database, or ambitious aspect oriented programming ideas… just object oriented programming expressed in diagrams is good enough. I had no idea back then how to merge the two or three concepts into that diagram language and maybe I should just give up there. That seems more achievable. I am sick, and not sure I am capable of doing anything large anymore ever.
* The time planning document ("New Computer Language, Project Steps & Time Planning.doc") looks far more overviewable and less intimidating. It all seems so manageable there.
* The document with the list of products ("New Computer Language, Products.doc") is overwhelming, because each article written is mentioned separately and that means almost each paragraph of produced writing is mentioned separately. If I would just mention the basic outlines, this might be better. Earlier, back then, it may have helped me see what I did and see how much I wanted to do. But with the goal I have now, I think it loses its purpose, and simplicity of the planning docs is more important than rigorous tooling for detailed planning of my own work.

### 2019-12-15 Notes

* I read over the document "New Computer Language, Strategy.doc" in full and did some reformulations, also removing my never realized studying goals.

### 2019-08-05 Brainstorm Restructuring Project Docs

The Circle project docs took a turn at some point in time. At one point it was mostly about documentation, then it became about both documentation and programming. But the project docs folders do not seem to be fully updated to that change. Maybe I can do that in the context of *this* project. First some more reorientation.

'Program Software System' now looks 'outdated', compared to the programming work described in 'Document Software System'.

I might actually move many of those topics from 'Future\Interesting Now' to 'Postponed'.

I also would want to put a cut into all the project docs and all the circle docs: this is the language and this is the rest.

So it gets isolated. In the past I wanted to put everything I did (and will ever do) with software development at home in a single system so general that I called it 'Software System'. Many docs are general and describe both that language + OS-like topics and applications. I might want to cut that in two: language and the rest. I might like to open source the language at one point and just leave the rest out of it.

I think I interwove these things maybe a little too much. I just liked to subdivide things into a single system of subdivision into which everything fitted. Also, the interweaving may have been stimulated by my wanting to combine this 'Creator' project with the 'Circle' language project. The 'Creator' project was about model-driven development, aspects and framework more than being a real computer language. I wanted to combine the two things into a single system, so that may have lead me to try and put everything into a single system. Now, I think I know that Circle is the computer language and you could program model-driven aspect oriented software with it, if it can provide the aspect construct. Really, I think it helps to not try and solve all problems at once. Even the large 'introduction' document to software system: you might put part of it in one for 'language' and just refer to it from the main document.

### 2019-08-05 Brainstorm Circle Docs Writing Style

* The read uses terminology in a very specific way, that is not shared with my peers, therefor not easing readers into the material. Maybe an introduction would do. The 'older' versions actually seemed to give that more gradual intro, that takes the reader by the hand, leading them to how and why this is actually useful, rather than just plainly stating the shapes that the diagrams can contain.
* That said, I do want to quickly come to the raw definition of how the diagrams are built-up. To introduce the raw basic specs quickly.
* The work might be modularized. I am not attacted to how CSS3 is modularized, each piece of specs with a different state of being finished up. It seems messy. But I can employ the same organization to accept certain concepts are just more crystalized out than others, making it easier to share, even in an unfinished state, and stimulating keeping things separated and separately usable even when other parts are just really still messy.
* I seem to have had several goals fighting over eachother, in projects done long ago about this documentation:
  + Explaining it to myself.
  + Designing the concepts, separately from the notation.
  + Tying together loose ends.
  + Making it easy to read for someone else.

### 2019-08-05 Brainstorm Circle Language Design

* One point of failure I see in this computer language, is that it works well when there are a limited number of symbols, but as lists get big, the language seems to lose its effectivity. One way to still make it useful, is perhaps to filter, or only partially display lists in the diagrams, just like a normal grid or list would. There was a prototype app that would generate diagrams out of vast sources of symbols. The problem became apparent there and it has been in the back of my head since then.

The 'large lists' problem in Circle also applies to large lists of commands, that might apply to an object.

Also that UI's are often optimized to show the most relevant options and then I just say: no none of that, everything only.

Maybe it's just that this UI will have its place along side other techniques.

It's just that the large lists problem should be solved in my view.

Maybe permanent filtering and reordering, which is often hard to customize in windows programs. Like a menu customizer.

Try to make it easier to do that. Construct your own limited view.

### 2019-08-05 Brainstorm version control and open source

Is it really such a good plan to open source it? Am I really not just throwing away money? Maybe it is better to wait with that decision, after I know where things are heading with work and income?

There is an '… IPC Parse.doc' text in the source control history. It should be deleted. I guess I have to search for intellectual property problems. This in docs that are so intellectual property sensitive. Before open sourcing it, you need to do this intellectual property check.

There is a folder 'Previous Versions' that would make you think you forgot to put those at the beginning of the conversion from version folders source control history. But they are not necessarily previous versions of documentation, but more like previous versions of systems.

2004-00-00 XX Symbol Language\Symbol Pictures (Keep Packed, Paths Too Long).zip has intellectual property problems in Pictures\Diagram Examples.

## Done

### 2019-08-04 Done Notes converting version folders to source control history

This is a lot of work.

* Exp: Search for XXXX in the folder "Circle Docs\1. Language".
* Obs: 54 matches.
* Hyp: 54x a version folder structure to convert to source control history.
* Less than I thought and probably doable.
* Maybe rename all commits, putting the version number in front, so it is more apparent that it is alternative source control history.

### 2019-08-10 Done Brainstorm complexity in preserving rename history

The version folders I have left to convert to source control history are complex, if you also try to get a rename mapping in check. Even if you ignore the rename mapping, you have quite some work to do.

It was not expected that there would be such complexity in preserving rename history, but noticing that there is, makes me re-evaluate my plans.

To get overview of the amount of version folders still to cover: The version folders are visible on the 'root' level per chapter. The amount of version control on *sub-*topics is limited to just one (24. Creation Behavior Of Calls). Its about the *main* topics: Commands, Parameters, Globality, Execution Control, Black Boxing, Interfaces, Events and Inheritance. One of those topics has intensely many version folders (Black Boxing). About 3 topics have a 'normal' amount of version folders. The others have a quite small amount of version folders. If you ignore trying to preserve rename history, then you might be done today. If you try to preserve rename history, it will take you probably many days, like 4 or something. And I think the motivation will recede if I do that.

So I have already made my choice. I will not make effort to preserve rename history. I will just methodically convert the version folders to source control commits.

### 2019-08-11 Done Basic Math Conversion to Source Control History?

The 2 versions' contents:

2005-01-06 00 Former Documentation:

Any programming topic about math I had at the time is thrown into one document here:

\* JMath 0.9 docs in Dutch

\* Simple Math operators

\* Ideas about 'regulated systems'

\* Number Bases

\* Brainstorm: Some alternative wordings and loose ideas written down.

The XXX version is only Simple math operators and how they can be implemented as objects.

Idea bout converting version folder to source control history:

So they are not mutually exclusive at all.

I have doubt whether I should even do the conversion from version folders to source control history here.

Using my rules, I'd put the older docs in archive, but neither 'version' is more deprecated than the other. They are both old, and they are both the latest version of things.

The math as objects does have a link to how math can be made not intrinsic to the language, but an extension library, which can still be compiled to good old CPU instructions. That concept is interesting for the idea of the new computer langauge. But... none of this documentation is well worded to suppord that concept...

### 2019-08-11 Done Controls Concepts Conversion to Source Control History

* I can convert the version folders to source control history the regular way, because each successive version does seem to replace the former version.
* Where is that navigation model brainstorm?
  + It is in the control concepts' root folder. It has a doc in it directly, which I overlooked.