# Circle Language Spec Revamp Notes 2019-08

*Author: JJ van Zon*

*Location: Oosterhout, The Netherlands*

*Date: August, 2019 – March 2020*

## 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 visually express the internals of computers and programming languages.

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

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

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

* I might want to focus on splitting apart non-Circle-Language topics from the Circle Language topics.

### 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 project docs sequentially for thorough orientation.
* Improve planning docs:
  + 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'.)
    - (The 'Productive Writing' part of the Strategy doc is quite resolute in the language.)
    - (Maybe I am being too strict towards myself, about having to rewrite everything in a milder tone.)
  + Include hour sheet?
* Improve Circle Language Spec:
  + 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.
* 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.

### Requirements

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

## Information

* For converting to MD:
  + <https://pandoc.org/index.html>
  + <https://word2md.com>

## Notes

### 2020-04-01 Notes Circle Language Spec Planning Docs

* Empty project folders:
  + I could evaluate whether those empty project docs folders are needed.
  + To me it seems odd now, that 'Errors' is put out of scope.
  + I think I want to move the topics 'Errors' and 'Concept Construct' from the Out-of-Scope document to the Circle Language Spec Planning docs.
  + The 'Out-of-Scope' document os currently not just postponed items of the Circle Language Spec, but more than that: not even considered part of the Circle Language Spec proect at all anymore.
  + It is still not a hard rule to be set. But putting Errors and Concepts out of view, makes it sort of not the same project for the sake of Concepts and putting my head into the sand when it is about Errors. If I have ideas about these things, they do belong in the project I think. I have a hard time thinking them away.
* Circle Language Specs Done projects:
  + It is a lot of content. It is not practical for me to go into detail about it now. It is only the past plans, not the future, so less important.
  + Do this also for the Current and Future plan docs.
  + Phase-by-phase.
    - Background black.
    - Name spelling JJ van Zon.
    - No abbreviations in folder name and file name.
    - Rename 'Document' to 'Spec'?
    - Suffix with 'Plan'.
    - Do the above for the Current and Future Projects
    - 'Project' => 'Plan'
    - Leave out the word 'Plan' if obvious.
    - Headings 'Circle Language Spec' + similar to folder and file name.
    - 'Computer Language' => 'Circle Language'
    - 'Articles' => 'Spec'
    - No sub folders, just files?
      * Some sub-projects contain many files.
      * Flattening it out might make a big list.
      * Keeping some folders but have other as loose files, makes you lose that time-line feeling.
      * Too much doubt -> Postpone.
    - Maybe prefix all with Circle Language Spec Plan. Then simplify the rest of the title, because Plan is already implied.
      * Getting path length problems. Collapsing sub-folders with only one file after all.
    - Not sure if that prefixing looks so great.
      * It looks cluttered. And I cannot apply it consistently, because of path length problems. Branch. Try without prefixes.
      * 'Spec Plan' somewhere in there looked ok.
      * My doubts are that if this doc is in the internet, the title means a lot, so why not have it be complete? It does already have that completeness somewhere in the URL, but…
      * Maybe let go of the toilet role principle and just call something 'Automatic Containment for Relations.doc'. Not 'Plan', not 'Spec' just lazy. Maybe it is ok.
      * My being better at systematics than words might be in the way. So that I don't easily decide what's clear enough.
    - Convert to docx without Compatibility Mode.
      * What use is that, if I want to convert it to MD instead?
      * Conversion to MD is not great if I use color coding it the docs.
    - Month in the headings, so it is clear that the fact that it is a time period is leading.
      * In the front or the back of the sub-project name?
      * In the file names too in case they have a folder?
    - 'TODO' => 'Notes'?
    - 'Legend' => 'Legend of Symbols and Markings'?
    - Remove '& Copyright'
    - Remove unnecessary title page white space and 'version XXXX-XX' mentioning.
    - Font Calibri.
    - Set proofing language to English US.
    - Page headers
    - File properties (author and such)
  + Details:
    - Get rid of content about 'studying'?
    - Command as a Concept Spec Plan seems a super project but later turns out not to be…
    - Done project "Document Internet as a Single Computer" is out-of-scope…
  + I would want to read over those Done projects content before publishing. I should know what I publish exactly and have evaluated it and made some adaptations possibly. Probably nothing in it is a secret, so you don't need to remove it from source control history, but slight changes might be good.
  + Some things especially in evaluations may seem cocky when I call my own successes very very good. I don't know if I need to change that. I also use I and you interchangedly when I talk about myself. Not sure if I have to change that. The plans sometimes talk in definites. Maybe openness is better language. Not sure if that's a problem. I worry what people would think of me. If they'd think I'm arrogant… maybe I should not worry about that.
* I somehow don't trust this is all there's to it about Circle Language Spec planning docs. Look over the state of it. It *would* be fun to move on to a next topic and have something finished. But somehow I don't believe it. > It is. I lean towards the actual language spec rather than the planning, now.

### 2020-04-09 Brainstorm Explore (Content) Search Options

* Exp: Visual Studio Find in Files
* Obs~ Some paths are too long…
* The file had a path of 260, which I think is the max for Windows 7, but some API's use 255 as a max.
* Exp: Shorten to 255. Find in Files again
* Obs~ Error is gone.
* Exp: Get rid of an intermediate folder to shorten the path and still keep descriptive names.
* Obs~ Visual Studio Find in Files will not search doc contents.
* Hyp: Windows 7 File Explorer will not find whole words. For some things that's relevant for other things it is not. (I thought I saw it).
* Exp: search "\*.doc"
* Obs: Now it does not match "\*.docx".
* Hyp: I thought I saw that some times.
* Exp: Windows 7 File Explorer. "D:\Source\JJs Software\Project Docs\Circle Language Spec Plan". Search "content:Joost".
* Obs~ It shows 2 files, if I open them (docx's) and search for Joost in Microsoft Word, I get no result. Odd.
* Hyp: File properties?

### 2020-03-22 Notes Circle 3 Programming Planning Docs

* I could change titles of projects inside the doc content too.
  + Also for Circle 3 Programming, though less importantly.
* Could I just go with it, call it 'Circle 3 Programming'? And if I want to mention in the planning docs that to documentation was of little concern, just do it with a more open formulation, like 'documentation was of little concern' or 'very little documentation turned out to be written in these projects. The focus turned out to be on programming the code.'?
* I could put a remark or something in the Circle 3 Programming docs that I did not do any documentation, even though it was the initial plan to do that.
* I was going over some sub-project docs to check if any documentation was written during those projects.
* Other sub-projects I scanned were not clearly any doc issues in them.

### 2020-03-07 Brainstorm Circle 3 Programming Planning Docs

(Circle 3 projects are software development projects, unlike Circle Docs projects, which are language design projects.)

It appears around 2010 I started off with programming Circle 3 with the intention of rigorous planning methodology and high quality technical documentation. Around the same time I switched employers. The new employer did not value my doing planning or documentation, just coding. In projects at home it seems I adopted that way of working. So planning docs and tech docs were no concern anymore. I focused on coding. I think I also stopped keeping an hour sheet at home. It was freeing. It felt too much like work logging the hours I spent at hobby projects. But the real motivation for the shift in way of working, seems to be that I cannot have 2 methodologies at the same time: one at home and another one at the office. That seems intrinsic to how my mind works.

So I have these near-perfect planning docs for Circle 3, while the execution of the projects was almost only coding, no planning, no documentation.

I don't think I want to reformulate the goals of Circle 3 programming projects, to exclude software design. It is not about making those planning docs good.

### 2019-12-29 Brainstorm Scope

* Basically I want to scope the project.
* 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.
* 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.
* *Diagram notation / redesigned constructs / gap lifting*: Eventually, you might split the in-scope topics apart in pieces that are in-scope and others that are out-of-scope. Some ideas just aren't the original ones, not the hot idea, and not prone to be adopted. For instance: The spaces in identifiers, text code ideas… I might have done good by keeping conceptual thoughts apart from diagram thoughts. I tend to want to merge those two aspects for the benefit of the reader. But I also tend to want a diagram notation that can be applied to other programming languages as a source for the diagrams. In that case, not all wild conceptual ideas are relevant. I did have thoughts: how would I implement this if I did it from the ground up, how would I implement that if I did it from the ground up, like interfaces, inheritance, ref-ness etc. Also I had ideas on the other end of the spectrum, more macroscopically: how far can I take the application of this diagram notation? Could they even substitute user-interfaces? Could boundaries be lifted, for instance, between things running on different computers and the boundary between users and programmers. The language gaps between things like files and folders, databases, programming languages, user interfaces… gaps between the physical and the logical. All that. Interesting stuff. The application of them are different (of *diagrams, conceptual constructs* and *boundaries lifting*). Another example is the automatic diagram organization topics. The diagram notation idea can live totally without some of the wild ideas in that. For instance, interchangeability between containment and referential structure or inversibility of containment in case of bidirectional relationships. Also the striving to want almost all relationships between objects to be bidirectional, does not apply if you want to use the diagrams to express systems in which you have a choice if relationships are bidirectional or unidirectional. Also giving things a different name (aspects are all of a sudden called concepts) is not a priority, and perhaps even alienating. Those are just some ideas I have about how to pull things apart. I think maybe those differences in application call for a rigorous split up: *diagram notation / redesigned constructs / gap lifting*. Diagram notation is a bit of a grey area, because it wants to use constructs. Some of the notation capabilities imply different variations of otherwise fixed constructs. Black box is a notation I dislike. It works, but it is not elegant (I think). Also, if you take C# as a source, rules about private/public are set, and some things just don't have to be denoted, because they are implied. A method contained by a a class can access all the other private members of that class. In my efforts to generalize and make things interchangeable, this 'had to be' explicitly denoted… I think a separation of some main groups of concepts is appropriate here, so the ideas are better transferable and perhaps better usable and applicable. It might make the general idea about the diagram notation more pure and also more freely usable. An argument that falls away with this open sourcing compared to patenting is that things do not need to have a closed, unambiguous definition. Things can be just part usable, using and replacing rules as one wishes. You can say at some point the diagram notation surpassed the original goal on two ends. I had an explosion of ideas then surrounding this notation. Boundaries between ideas were not well defined. That's a (good?/bad?) quality of my way of thinking, I guess. But I tend to want to focus things, by splitting things apart in the main blocks, so it that it might become more practical (for others or myself).

### 2019-12-29 Brainstorm Circle Language Spec: Design

* Rename the term 'Code Base' to something like 'Base Code' or 'Base of the Code'.
* 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.
* 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 Language Spec: 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.
* The design of the programming language should lose some ambition and express that only as dreams.
* I took a look at some of the postponed work. I worry about the messiness of the content. And if the loose ends will make the idea fall apart. And whether this makes it even fit for publishing. I just don't know at all. But I think I should come back to it later. Because I had strategies for this. And I might be too hard on myself. A clear 'flag' [Preliminary documentation] in red somewhere at the top usually does the trick. Might tells people clearly not to not too much from the text that follows. Just being clear about that might be enough.
* Reading over the Interfaces planning docs I noticed I said there I like how the Interfaces chapter reads. I started reading it over. I notice, I only like how it reads where lots of little diagrams are shown. I like it not so much when I just see walls of texts. I think it is my 'brain type' so to say. But I think having both text and pictures would help a lot of brain types. So: more pictures. I like pictures.
* I notice I talk a lot about implementation rather than notation. For instance: Does a dashed circle mean it is used as a class, enforced to be a class, static inside its container, how does it work in the system interfaces? What if it is just the notation that is a good idea, what if the implementation isn't. That might even make system interfaces' precise definition not important or maybe just subjected to diagrams drawn out to represent things from another language, like C#. C# getters and setters might be in a system interface notation. But setting an object reference's interface dynamically in runtime… may be too much of an implementation detail. I think it is a language definition / runtime implementation separation. In think the engineers at Microsoft might be right about developing language spec / runtime / framework / compiler quasi-independently. Maybe I can be inspired by that and make my language definition a litte simpler. I am subjected to the pitfall of wating to cover every little minor edge-case, of which I have a fear that it may make the whole system fall to pieces. I already warned myself about that in the Circle Language Spec Strategy document. But now I think other people might actually read this, I start to think: maybe limit the scope. Somehow define the diagram notation and what it represents and not want to work out how things would work in a runtime. Runtime would be a system where the diagrams and actually the data that internally describes the diagram, to be loaded and run as computer programs. I think I wanted to check the usability of the notation by shining light on any little aspect of it, I could find. But I think some details are not that important. Maybe those are to be demoted to possible implementation details, to keep the main part of the story clean. I am OK with apologizing in the documentation, that this might not be usable or something. The description in the Strategy document is pretty much spot on, I think.
* I get the problem that next to introducing new notation, I also wanted to introduce new concepts. A new conceptual take on things. I think it all became a little much. Maybe I can improve that during this project 'Circle Language Spec Revamp'.
* I think the 'being blunt' might help. It is not blunt unfriendly, because the explanations might become so much simpler if you say: this is that, this is that, instead of and this far-fetched edge case is solved in this difficult, abstract, theoretical, but precisely determined way, that I'm not sure I even understand anymore… : )

### 2019-08-05 Brainstorm Circle Language Spec: Large Lists Problem

* 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 (with poor judgement?) 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.

### 2020-01-13 Brainstorm Aspects / Concepts

'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 an aspect from a conceptual point of view. 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.

### 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

### 2020-04-01 Done Circle Language Spec Planning Docs

* Circle Language Spec Products doc: Move postponed work from Done section to the Postponed section.
* The document "Circle 3 Excluded Requirements" has topics that might be interesting for the Circle Language Spec project.
  + Was at: Round Up looking for things to add/move from Circle 3 Programming to Circle Language Spec.
* Circle 3 Programming, Preliminary Designs:
  + Yes, definitely quite some useful texts in that document, that belong in the Circle Language Spec project.
  + The texts seem to belong in the actual Circle Language Spec docs, not its planning docs.
  + In the past I made effort to dump even unfinished ideas right inside the docs.
  + I had doubts about it, since it would litter the docs with unfinished, lower quality material.
  + But I remember doing it, because otherwise loose ideas would be spread over numerous sub-project documentation and would get out of view.
  + Do I move these texts to the appropriate spot in the Circle Docs now?
  + Do I also move ideas from Sub-Projects there? Maybe those are too short to matter. And not really loose texts, but goals. Somewhere in the middle. So maybe I should copy/move some texts to the loose idea boxes of Circle Language Spec.
  + I'm staring at 'Concepts' now. But Concepts is out-of-scope. Very out-of-scope. Does it matter? Any content about it is to be dumped into the appropriate spot in the documentation.
  + Object Order: Some of it is in Data Concepts. But I fundamentally need it in the Circle language. It's not just list order. It's more fundamental. It can be command execution order, that is dependent on it. I just don't know where it belongs in the chapters. Maybe just its own chapter I was making a separation between coding concepts and data concepts once. Now most coding concepts are just part of the programming language and most data concepts are not. Just the separation is not as precisely clear as that.
* Circle 3 Programming, Requirements:
  + Has technical design requirements.
  + I could look at them, take them out, compare them to the Circle Docs requirements, to see if they are in there or have a place in there.
  + Nah.
* Concepts construct:
  + I am missing writing the Concepts construct documentation from the planning docs. I only see it in the Circle 3 Programming, Excluded Requirements.
* Lower contents:
  + The idea of lower contents is described in project docs not in Circle Docs Planning, but in '1. Circle 3 Programming' planning.
  + You can also find some texts about it here: Circle 3, Lower Contents, TODO.doc

### 2020-03-22 Done Project Names

* Inspecting the sub-projects just to find whether documentation was of any concern in those projects, might be too intense for me right now.
* The question I was trying to answer with that was: Is it accurate to call the super-project 'Circle 3 Programming'?
* The reason for calling it that, is to make it clearer what the project entains, separating it better from the super-project with the name 'Circle Docs', so that there is a clear distinction that one is about programming and the other is about documentation.
* But from the top of my head I kind of already know that the focus of those Circle 3 projects was programming, not documentation.
* I just want clarity on the distinction between projects Circle Docs and Circle 3 Programming, but giving it a clearly distinctive name.

### 2020-03-22 Done Organize Planning Docs

* I moved around folders, for a preliminary split up between Circle Docs and Circle 3 Programming projects.
* I could change titles of projects inside the doc content too.
  + I am unsure whether I want to call the projects 'Circle Docs'. Maybe 'Circle Language Specification'.
  + I do want to go for the name 'Circle' as opposed to 'New Computer Language'.
    - I don't mind that this makes my documentation stick to a name that in theory might later change. It probably will not change at all.
  + But Circle Docs is too general, I think. It should be Circle Language Specification.
  + I don't care for the abbreviation 'Spec'. I am not a fan of abbreviation and I don’t think it makes it much more concise or clear.
  + Also: I am hitting my limit here. I'm tired and getting all sorts of symptoms.
  + Circle Language Design is also an option. 'Specification' may raise expectations about finishedness.
  + Circle Language Design Planning
  + Circle Language Specification
  + 'Specification': suggests finishedness, but does it? CSS is all sorts unfinished modules.
  + Leaving out 'Specification' or 'Design': Makes distinction with Circle 3 Programming hard to see.
  + The word 'Circle' should be in it.
  + The word 'Language' at least sheds light a bit on what it actually is.
  + But Circle Language Specification Plan is just a long name.
  + But 'New Computer Language Functional Design' was too, so that is not an argument?
  + 'Spec': Is an abbreviation. I generally dislike those, but I think in this case it is appropriate. It is adopted as a word by the community.
* I could evaluate whether those empty project docs folders are needed.
  + The 'update … articles' sub-projects
    - Defined as empty project doc folders: I see some of it back in the Circle Language Spec's Strategy doc, Project Steps and Time Planning, but not in the Products doc. Some redo's are defined, be it not in Products doc. It seems the reason for redoing is not always stated. I think I should make the list complete in all the planning docs where appropriate. Then I can get rid of the remaining empty sub-project folders.
    - Mentioned them in the Products doc now.
    - Removed the empty project doc folders.
    - Look at Circle Language Spec Strategy for things to put in Circle Language Spec Products.
    - Look at Circle Language Spec Project Steps & Time Planning for things to put in Circle Language Spec Products.
      * There is a lot there that specifies the reasons to change the articles.
    - Consider that some elaborations in the Circle Language Spec Products document might belong elsewhere.
      * Do they belong in Circle Language Spec Strategy?
      * Sort of collides with the Sub-Project folders idea. Maybe details belong in a sub-project folder and a general reference to it in the strategy doc.
      * Maybe go the other way: put sub-projects strategy in the main strategy doc and get rid of separate sub-projects. (My gut says: no, keep sub-project descriptions. That's what they are for: elaboration on a sub-project's scope.)
      * Decision: Rule: Any elaboration now in the Products planning doc, belongs in a sub-project document.
* Sub-Projects:
  + The documents could probably be shorter. Lots of those Project Summaries are not so useful template texts, not filled in. The notes at the bottom would do.
  + Then it is basically just loose ideas, that might have been given a too prominent place. Maybe put it elsewhere, if it is just unhelpful for visual overview, perhaps. > Some are more than that. I keep em this way.
* I like that I seem to build tolerance (back) for not exactly knowing where I am going with it, gradually shaping it, and just taking my time to make things more overviewable, regardless of whether someone might think it is time well spent or not. I myself just create more overview and understanding of the material that way.

### 2020-03-16 Done Circle 3 Requirements Docs

* I moved content from Circle 3 Requirements to New Computer Language Products doc.
* I could rename it to Circle 3 programming, rather than Circle 3 software development, because programming was all I did, not full blown software development cycles. Right? ('Programming' is a bit ambiguous too. It could mean program the dev environment or program using the new language. But I am OK with it.)
* Maybe check sub-project docs later to verify that I didn't do any documentation.

This is spreading my attention over too many different things. Is there a more practical approach?

This is too intense. I have to stop again.

### 2020-03-15 Done

I read over notes to know where I was, and mark some things of as 'done'.

This was too intense. I wonder if starting to change documentation will be too intense too or if I should stop now. Or if other activities or non-activities will be even more intense. I choose to stop for a while now. After a little over 20 minutes.

### 2020-03-08 Done Reading Circle 3 Requirements Docs

The requirement group 'Priority C: Classes' has 1 language design feature: <Diagram Notation Design> Static. So that is to be moved to the language design project docs.

'Priority E: Integration' contains brainstorming instead of a list of items.

(I notice I get inspired to like implementation projects for Circle 3. I feel the enthusiasm in it. I somehow stopped working on it, though. Also it is not my goal right now. The goal is publish Circle language design eventually.)

Some of the content in 'Priority E: Integration' could be part of language design, at least diagram metrics design and automatic containment.

'Priority F: Large Amounts of Items': Spiraling could be part of diagram metrics documentation. And object order.

'Round-Up': Most are documentation issues, which in theory could be moved to the language design project instead. Except, they are 'technical design' documentation issues, which do not have a place in either the language design project or the circle 3 programming project, because 'technical design' I did not consider language design, because I would have called language design 'functional design'. 'Technical design' according to my views back then, would have be document how I implemented things in the programming of Circle 3, not how the language functionally works, but how Circle 3's .NET code works internally. However, I could see the topics up for 'technical design' as topic that also could use an update to the functional design. So I could consider those topics for extending the Circle language spec project's requirements. Then I could consider removing documentation issues from the Circle 3 programming project docs, because I wasn't going to do them anyway.

I ran over all the content of Circle 3 Requirements and above are the conclusions of what to possibly do.

### 2020-03-07 Done No Planning or Docs Back Then

Circle 3 Strategy is pretty much done.

Circle 3 Requirements: change coloring and formatting.

That does not take away I want to split topics in these planning docs between language design one one end and programming at the other.

### 2019-12-29 Done Brainstorm Scoping

* 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?
* 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 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 new 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. 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 not make it too ambitious, cover too much, so large in scope, that no one would pick it up anymore.
* 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.

### 2020-02-23 Done Reading Circle 3 Strategy

An idea for today would be to read "Circle 3 Strategy". That document is supposed to be about software development, not language design. At least, that is the new goal I have with that document. Some things in it are relevant for the Circle Docs.

"Goal of the Language" is where it is part about the language design, not so much the software development. I might use/move this text to the Circle Docs project docs.

Was at "More Tips" processing things.

At what point am I going to be more rigorous in splitting off the Circle Docs project docs from the rest? Not yet, I think. It's still a mash up of both in the project docs I am reading now.

### 2020-02-20 Done

Today finished splitting time planning and projects step into in-scope and out-of-scope documents.

### 2020-02-16 Done Notes

Working on splitting time planning and projects step into in-scope and out-of-scope documents. (worked on it for 45 minutes this day.

### 2020-02-13 Done Notes

Working on New Computer Language, Products.docx:

I do not want to necessarily want to shorten the list of products any further. Even the list of Done work. Because some parts are part done and it would be easier for me, would I work on it, to have the same subdivision of the pieces TODO as the pieces that are done.

What I do might want to change is the rough order in phases of the stuff TODO.

I might just go over it a few times, reformulating.

I am splitting off parts of New Computer Language, Products.docx into a separate document with the out-of-scope things.

I want to go over the Postponed topics, to see what content can be moved to the 'out-of-scope' document.

Now I still want to weed out the 'Topics Roughly' moving things to the 'out-of-scope' document with the products.

Also the Strategy can be stripped of things 'out-of-scope'.

### 2020-01-13 Done 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.

### 2020-01-30 Done Notes

I read over New Computer Language, Strategy.doc and reformulated stuff.

I am cleaning up New Computer Language, Products.doc: simplified color coding, removed mentioning 'in a Diagram' and 'in Text Code' article variations. I might remove detail from done work, but keep it in the proposed work. May remove some 'musts' by 'mays'. Might add intro docs to calm the reader's nerves down, on the overwhelming amount of topics. Do I need to excuse myself for introducing topic names, without actually describing what it entails? Don't know. That description would *be* the product. I have a canundrum. I cannot describe the product without making the product, because the description is the product.

### 2020-01-04 Done Rough Plan

Rough plan:

* Remove detail from products doc.
* Split main 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.

### 2019-12-15 Done

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

### 2019-08-10 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.

### 2019-08-05 Done Brainstorm Restructuring 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.

### 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.