# Circle Language Spec Revamp Notes 2019-08

*Author: JJ van Zon*

*Location: Oosterhout, The Netherlands*

*Date: August, 2019 – April 2020*

## Contents

[Contents 1](#_Toc37540046)

[Project Outline 2](#_Toc37540047)

[Introduction 2](#_Toc37540048)

[Goals 2](#_Toc37540049)

[Limitations 2](#_Toc37540050)

[Steps 3](#_Toc37540051)

[Requirements 3](#_Toc37540052)

[Information 3](#_Toc37540053)

[Notes 3](#_Toc37540054)

[2020-04-01 Notes Circle Language Spec Planning Docs 3](#_Toc37540055)

[2020-04-09 Brainstorm Explore (Content) Search Options 6](#_Toc37540056)

[2020-03-22 Notes Circle 3 Programming Planning Docs 6](#_Toc37540057)

[2020-03-07 Brainstorm Circle 3 Programming Planning Docs 6](#_Toc37540058)

[2019-12-29 Brainstorm Scope 7](#_Toc37540059)

[2019-12-29 Brainstorm Circle Language Spec: Design 8](#_Toc37540060)

[2019-08-05 Brainstorm Circle Language Spec: Writing Style 9](#_Toc37540061)

[2019-08-05 Brainstorm Circle Language Spec: Large Lists Problem 10](#_Toc37540062)

[2020-01-13 Brainstorm Aspects / Concepts 10](#_Toc37540063)

[2019-08-05 Brainstorm Version Control and Open Source 10](#_Toc37540064)

[Done 11](#_Toc37540065)

[2020-04-01 Done Circle Language Spec Planning Docs 11](#_Toc37540066)

[2020-03-22 Done Project Names 12](#_Toc37540067)

[2020-03-22 Done Organize Planning Docs 12](#_Toc37540068)

[2020-03-16 Done Circle 3 Requirements Docs 13](#_Toc37540069)

[2020-03-15 Done 14](#_Toc37540070)

[2020-03-08 Done Reading Circle 3 Requirements Docs 14](#_Toc37540071)

[2020-03-07 Done No Planning or Docs Back Then 14](#_Toc37540072)

[2019-12-29 Done Brainstorm Scoping 14](#_Toc37540073)

[2020-02-23 Done Reading Circle 3 Strategy 16](#_Toc37540074)

[2020-02-20 Done 16](#_Toc37540075)

[2020-02-16 Done Notes 16](#_Toc37540076)

[2020-02-13 Done Notes 16](#_Toc37540077)

[2020-01-13 Done Notes 16](#_Toc37540078)

[2020-01-30 Done Notes 17](#_Toc37540079)

[2020-01-04 Done Rough Plan 17](#_Toc37540080)

[2019-12-15 Done 17](#_Toc37540081)

[2019-08-10 Done Brainstorm complexity in preserving rename history 17](#_Toc37540082)

[2019-08-11 Done Basic Math Conversion to Source Control History? 18](#_Toc37540083)

[2019-08-11 Done Controls Concepts Conversion to Source Control History 18](#_Toc37540084)

[2019-08-05 Done Brainstorm Restructuring Docs 18](#_Toc37540085)

[2019-08-04 Done Notes converting version folders to source control history 19](#_Toc37540086)

## 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 difficulty maintaining this rule. I tend to want to start changing things.)

* I might want to focus on splitting apart non-Circle-Language topics from the Circle Language topics.
* Maybe not write to clarify new concepts, but rather limit it organizing already worked out concepts.

### 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.
    - (The 'Productive Writing' part of the Strategy doc is quite resolute in the language.)
    - (Perhaps search for definites such as 'have to', 'must', 'will', 'should', 'very'.)
    - (Maybe I am being too strict towards myself, about having to rewrite everything in a milder tone.)
  + Scoping
  + Include hour sheet? (Extracted from hour sheet about multiple projects?)
* Improve Circle Language Spec:
  + Reorganize Circle Docs files and folders (e.g. lone files in folders is not handy).
  + Scoping
  + Turn separate Concept/Diagram/Text Code articles into one.
  + Reduce the number of articles, by grouping topics together.
  + Generally go through content reformulating things?
  + Less resolute language, more wiggle room?
  + Start reading / writing / reformulating.
  + Convert from Word to markdown, so it may get indexed by Google and easier to navigate and read 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.

### Language Design / Content Changes

* Gentler intro
* Static notation
* Large lists problem
* Friend notation (too prominent?)
* Fully OO (is it?)
* Language lost its purity (with implicit notations?)
* Simplify rules
* Comment notation
* List concept more generic
* Pointers (separate chapter)
* Relationships 'always' bidirectional (might not be a good choice)
* Object-commands / command-objects exchangeability doubts
* Enriching misc diagram topics
* Access marks: Make them arrows by default? think of something different for protected?
* Writing style:
  + More pictures
  + More open, less resolute language
  + Use common IT terms
  + Write unfinished articles
  + Is it a *notation* or is it a *run-time*?
    - (Texts might insist that Circle is a run-time, while the notation might be the main point of things.)
  + Over-awareness of pointers?

## Information

* For converting to Markdown:
  + <https://pandoc.org/index.html>
  + [https://word2md.com](https://word2md.com/)

## TODO

### 2020-05-18 TODO Content Changes for Commands Chapter

* It seems to me that I wanted to evaluate back then:
* What concepts from other programming languages could look like in the diagram notation.
* What each diagram notation situation could mean in terms of other programming languages like C#.
* What would happen if you try to draw a parallel between objects and commands. Where things overlap, what the implications of being a command is, compared to an object.
* That.
* But it may have resulted in a bit of an overwhelming set of situations being shed light on.
* Not sure to what extent this is a problem.
* It's more of a content issue, rather than a formatting issue. I was focusing on formatting issues for now, so maybe postpone this.
* What if you would just remove most of the rules imposed onto commands compared to regular objects? It seems that maybe no disasters would happen. I would like to evaluate which rules can be lived without so the idea might be presented in a simpler way.
* The rules seem simpler in Circle than in e.g. C#. It seems that derived from basics, several construct from e.g. C# might have a corresponding expression in Circle using more basic rules. I seem to go through an effort to describe how supposedly more familiar constructs might be represented in Circle. It seems quite an effort to explain something that might be simpler in terms that might be more complicated. But it might have a purpose as to build a bridge between what a reader might already be familiar with to a different idea, that the reader is attempted to be introduced to. I have some doubt about whether the story could be simpler. Maybe it could be simpler by adding something: first the basic rules with no edge cases, then an admission of it e.g.: "Familiar constructs from other programming languages might be expressed using these basic building blocks from Circle. Each of those familiar constructs seems to be a specific use-case of basic building blocks in Circle."
* "Changing Inactive to Executable" section:
  + Seems quite Circle Broader View-ish. Use-cases of how you might use it in a less conventional setting. What it might be like if the internet was a single computer that everyone was programming and using at the same time, with more parts changeable and accessible to multiple people at the same time.
* The 'many rules' around commands seem circumstantial: practical behavior that might be found in other programming languages. At the other end of the spectrum might be a hypothetical system, in which no control is imposed with rules as such, which may only cause possibly inconveniences of e.g. concurrency issues like reading out half-written objects and such. The 'rule richness' might already be in place if the notation were to be used in an existing system or a new system might be developed that uses the notation and the programmers might use their own heads on how to make it practical. The 'rule free' variation might also be an experimental funny thing to develop. It might still function as a coherent base for programming, be it there could be concurrency issues for instance. I really wonder if the rules might have to be se embedded in the text. I might be more comfortable to extract these rules as a separate article that just aims to expose a system of rules that a programming environment might impose onto command objects, to make it more practically workable and evasive of troublesome situations. Some programming environments just support this behavior out of the box. "Local variables and parameters are automatically thread-safe" is one of those rules that I come to rely on when programming concurrent systems. (I personaly find concurrency one of the more difficult issues to deal with in programming.) If rules are not imposed in Circle, that thread safety could fall to pieces in a way. I just think a description of the notation does not need to rely on the existence of those rules. If I want to go into it, I might want to separate that from the main article, because it might make the idea easier to absorb. The rules I state were maybe just my attempt to prove to myself what would it take for objects and commands to be exchangeable and what rules does it take to make object behave like commands in an orderly fashion and does this notation work at all for the many command-oriented constructs that exist in programming languages today. Now I am more confident that it works, the main explanation might not have to be a stack-up of 'proof', just an exposition of the main point.
* Where I use parameter notation with connectors, I might want to be honest to for now ignore what those connectors mean, just accept that it makes them public and makes them parameters.
* Commands Main Concepts article:
  + A lot seems to be hung up on definition of words and expression of concepts from other languages.
  + But I thing just the concepts and notation could become more centric.
  + From those concepts and notation might follow how existing constructs in the world might have a (unique?) expression in Circle.
  + I might want to take the reader along with *that* idea. A smaller set of elements in Circle might translate to a larger set of constructs known from other languages.
  + I see an analogy between the lifecycle of running a command and an object's creation, constructor, initial setting of required values, and running of active commands directly in it.
* My main concern is that I do not like that much the main structure upon which the text is laid out.
  + The current structure is based on constructs and rules from other languages.
  + It is my opionin that that is not an impractical subdivision.
  + But what seems to happen is lots of details covered that do not describe the core of what the notation might do.
  + So perhaps it can be replaced with the raw diagram expression elements and their meaning.
  + And then more of a sum op of how constructs fit into that, then lastly more detailed rules perhaps.
* Possible new writing structure:
  + Short definition of the concept of commands.
  + The raw diagram expression elements and their meaning.
  + A bridge saying constructs might be derived fom these basic elements.
  + Then name and diagram expression for each construct. (Possibly use existing texts as inspiration.)
    - Short definition of the construct?
    - Less explanation?
    - A short sentence covering diagram elements' usages?
    - Vary grammar constructs?
    - Pseudo-code
      * It might be vague to compare it to other programming language and not show some pseudo-code how it might look there. I know it sort of breaks my idea of not comparing to other languages that might, but here I explicitly decide to do it anyway, so I may as well 'own' it.)
  + Synonyms as sum-ups. (Intended to be deprecated to a detail.)
  + Extracting rules and put them in the bottom?
* Clauses:
  + Inactive clause could also be a local function.
  + Inactive clause could be the clauses of an if-statement for instance.
  + Active clause could be a braced code block.
  + I did seem to reinvent the word clause there. I wonder if that is necessary. It seems to raise the issue of perhaps you might use IT terminology more as known by the industry and not reassign and reinvent terminology.
  + The structure choices indicate I might have sections specifically for local functions, clauses (e.g. if clauses and what have you), local code blocks.

### 2020-07-24 Postponed Checklist Reformulating Commands Main Concepts Article

* Fine-tune styling.
* Evaluating if texts are in scope, possibly moving them.
  + But most sections seem appropriate and in-scope to me, except maybe the bottom edge-cases, but I think I will get to that in time eventually.
  + Commands Anywhere has a Broader View feel to it according to my opinion.
  + Terms "In" and "Out" might be avoided.
    - I seemed to be juggling the concepts Get and Set on one hand and In and Out on the other, before realizing they might not be synonymous.
    - I seem to go into specifics about Input/Output. I might want to move those specifics out at one point. I think I would like to evaluate that as I sequentially go through the text.
    - I might actually move the section where those terms are used: move them from the Commands chapter to the Parameters chapter.
    - Perhaps later be honest in the Parameters article about this paradox. Other people may also feel uneasy about it and it makes it maybe more understandable why the terms are avoided.
* Marking trigger words in red.
  + With find and replace
  + Manually
  + Over-used words? "The"?
* Finding replacement words, possibly using online thesaurus.
  + Going through text sequentially.
  + Sometimes moving things to an 'Out of Scope' section.
  + Terms "In" and "Out" might be avoided.
    - Because it seems ambiguous (for instance an object reference (**'Object In**' parameter?) can be considered both input and output at the same time.)
    - Replacement terms might be ones such as **Get Object** Parameter or **Set Value** Parameter. It may not look as pretty, but it might help me avoid the paradox that I experience if I were to call them **In** and **Out**.
* Put subjectivity in perspective.
* Change the orange markings (pictures, texts to move or change).
* Pictures:
  + Maybe draw new ones in one go, on paper and then scan it.
  + Sometimes borrow (pieces of) other pictures
  + Or simplify pictures by editing them.
* Merge conceptual/diagram explanations:
  + When I reformulate things top to bottom, I might at one point feel that the merging of conceptual explanation and diagram explanation is actually finished, because right now, often they are put near each other, but not really became one merged text.
* I have difficulty not wanting to change the story, while the aim is to just reword it, and I seem to experience an impossibility not absorbing the story. My conundrum might be solved, by adding indicators that everything is just a suggestion, so that no idea would disrupt anything, even when I may not entirely agree with it myself. Sometimes I might do the changes anyway.
* Going over the document again to reformulate.
* I might go over the loose ideas in the document and cross out or distribute those.
* Spell check.

### 2020-06-01 Checklist Tone Change

* (More helpers are in my personal development notes, not here, sorry.)
* Marking trigger words in red.
* Possibly mark questionable trigger words with a star (\*) too.
* Finding replacement words, possibly using online thesaurus.
* Going over the document again to reformulate.

### 2020-04-18 Brainstorm Writing Style

* The aim is to use less definite, more open language.
* (I can view ideas on how in my personal development notes, not here, sorry.)
* Also check: whether the stories make sense.
* And whether the content is still in scope.
* Remove links, since they break so easily.
* Casual mentionings of article titles are also fragile. They break quite easily.
* That seems quite a lot to check. Maybe that is why it is not easy.

### 2020-05-27 Checklist Reformulating Chapters

* Done projects:
  + The postponed reformulation of the done projects is on my mind.
* New Intro:
  + But that makes me think: I want to give it a new beginning in the first place: use some of the old Symbol Language documentation and reformulate a gentler intro.
* Reformulating:
  + More open, less resolute language
  + I could start reading and reformulating top to bottom.
  + Circle Language Spec Overview
  + Circle Language Spec Introduction: Applying trigger word marking technique to this already reworded document?
  + Basic Diagram Elements
  + Objects chapter
  + Classes chapter
  + Relationships chapter
  + System Interfaces chapter
    - System Objects article
    - Assignment article
    - System Command Call Notations article
    - Connectors & Connections article
    - System Objects Misc Issues article
    - List Concept article
  + Commands chapter
  + Parameters chapter
  + Modules chapter
  + Execution Flow chapter
  + Black Boxes chapter
  + Interfaces chapter
  + Inheritance chapter
  + …

### 2020-06-11 Brainstorm Over-Awareness of Pointers

* Some texts may also make a bit of an issue out of something almost always being related to a *parent* *object*. It may have gone a bit far incorporating that concept in pictures, usage of terminology. In pictures by displaying a parent object, where it might not add much and it might be an idea to remove it for clarity. In terminology, awareness of ref-ness might have slipped in by calling things related items and related lists more often than perhaps required. The terms 'object' or 'item' and 'list' might do in cases and that may make the text easier to absorb.

### 2019-08-05 Brainstorm More Pictures

* Explaining more with images is also something I would value. That might be more intense for me, because it requires some more 'mixed' tasks, which could be more difficult for me right now: reading, determine places to put pictures, draw pictures, scan, cut them out, paste them in text.
* 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.

## Postponed

### 2020-04-18 Postponed Reword 'Done' Circle Language Spec Planning Docs

* 2008-05 01 Diagram Expression Specs Project Summary.docx
* 2008-05 02 Classes & Relationships Specs Project Summary.docx
* 2008-06 01 Execution Control Spec Project Summary.docx
* 2008-06 02 Command as a Concept Spec Project Summary.docx
* 2008-06 03 Clarify Command as a Concept Spec Project Summary.docx
* 2008-07 01 System Objects Spec Project Summary.docx
* 2008-07 02 Assignment Spec Project Summary.docx
* 2008-08 00 Commands Spec Project Summary.docx
* 2008-09 01 Organize Circle Language Ideas Project Summary.docx
* 2008-09 02 Parameters Spec Plan.docx
* 2008-10 01 Scatter Symbol Language Documentation Content Project Summmary.docx
* 2009-06 01 Circle Language Spec Plan Part B, Sub-Projects.docx
* 2009-06 02 Circle Language Spec Plan Part B, Evaluation.docx
  + > Stopped, because I seem panicky about which words to pick and whether someone might be rubbed the wrong way with it somehow.
* 2009-06 01 Planning + Black Box Spec Part A, Eventual Plan, Goal Description.docx
* 2009-06 02 Planning + Black Box Spec Part A, Eventual Plan, Elements.docx
* 2009-06 03 Planning + Black Box Spec Part A, Evaluation.docx
* 2009-06 04 Planning + Black Box Spec Part A, Initial Plan, Goal Description.docx
* 2009-06 05 Planning + Black Box Spec Part A, Initial Plan, Elements & Time Estimation.docx
* 2009-06 01 Planning + Black Box Spec Part A, Notes, Startup.docx
* 2009-06 02 Planning + Black Box Spec Part A, Notes, Cycle 1, Process Idea Box.docx
* 2009-06 03 Planning + Black Box Spec Part A, Notes, Cycle 2, Black Box Part A.docx
* 2009-06 04 Planning + Black Box Spec Part A, Notes, Cycle 3, Redo System Objects & Assignment Part A.docx
* 2009-06 05 Planning + Black Box Spec Part A, Notes, Wrap Up.docx
* 2009-08 Review by Ramses, Small Plan.docx
* 2009-09 01 Wrap Up Black Box, System Objects & Assignment Specs, Goal.docx
* 2009-09 02 Wrap Up Black Box, System Objects & Assignment Specs, Startup.docx
* 2009-09 03 Wrap Up Black Box, System Objects & Assignment Specs, Implementation.docx
* 2009-09 04 Wrap Up Black Box, System Objects & Assignment Specs, Wrap-Up.docx
* 2010-02 01 Interfaces Spec, Strategy.docx
* 2010-02 02 Interfaces Spec, Product List.docx
* 2010-02 03 Interfaces Spec, Steps.docx
* 2010-02 04 Interfaces Spec, Evaluation.docx
* 2010-05 01 Events Spec, Strategy.docx
* 2010-05 02 Events Spec, Product List.docx
* 2010-05 03 Events Spec, Steps.docx
* 2010-05 04 Events Spec, Evaluation.docx
* 2010-05 01 Inheritance Spec, Strategy & Steps.docx
* 2010-05 02 Inheritance Spec, Products List.docx

### 2020-05-27 Postponed Pointers Chapter

* Maybe pointer-to-pointer situations might all be moved away from each chapter and into a separate chapter. It seems a specific issue and many things seem to be able to live without it.

### 2019-08-05 Postponed 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.

### 2019-12-29 Postponed Comment Notation

* I am missing the notation for comment. Would be welcome. I think it is simple, unambiguous and would be a shame if it were overlooked as an option. Just a line from the text to the wherever, as long as nothing's directly connected to it, there should be nothing else that would clash with that symbolism.

### 2019-12-29 Postponed Purity Lost?

* 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 an attempt at the purest form that I could find, in which you could draw out an object oriented system with shapes and lines. 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. This also counts for e.g. System Command Call with Argument Notation: I introduce simplified, yet abused slightly ambiguous notation for something you probably will never see used. What if at some point I find use case for this simplified notation that also looks like something it's not? E.g. a call to an object getter:



* It's ambiguous, because it looks like the diamond, which is a command call to the getter, is a reference to the object. It reference-associates a command call and an object, which are two separate 'objects', but connecting them with solid lines would suggest they are the same object.
* Maybe expressing doubt about it at the beginning is just as far as the solution might go.
* The protected access mark / access modifier:



I thought of this relatively early. To solve ambiguity problems, I introduced notations such as: a system command connector for the New command:



Event connector:



The concepts of its being *required* for a command to be executed:



But the latter notations are there kind of because the simpler notation was already in use by the concept of the protected access modifier.

I am having second thoughts about this. Maybe I would like to consider using a different symbol for Protected, so that these connectors, events and optional/required can get a less contrived notation.

Another example where it is an issue: This expresses that the other end of the connection should be a triangle:



But if you would accidentally bend the half triangle at the end the wrong way, it would look like a protected access modifier. Well… maybe that is not definitive proof that there's an issue, but it's an indication of something might be confusing.

* A jagged line type for the Name aspect I would like.
* Maybe choices for shapes with different line types could become less iffy. (See Basic Diagram Elements.)
* Maybe less rules for line direction, even though an effort for this may already have been done in moving from the previous version Symbol Language.doc to the newer definition Circle Language Spec.
* The access marks may become arrows instead, after considering if that works out with the rest of the notation rules.
* Maybe accept that the system interface may be visible more, while letting go of implicit notations to avoid them. Maybe not all the implicit notations for e.g. system commands calls would be scratched. Maybe the less usual ones, that seem to look the most ambiguous of all, if I may express it that black and white.
* The line merge and symbol merge notations may be let go of. (See Basic Diagram Elements.) But maybe later it was let go of already, I cannot remember that clearly anymore. Those notations look quite ambiguous with more basic notation ideas. I think I introduced those, because my idea was to make bidirectional relatioships more common, and I perhaps disliked how many symbols and lines you might use to express bidirectional relationships, because in my view they would be used everywhere. I let go of that idea of bidirectional relationships everywhere. Only in use case ideas I might have for aspect oriented programming ideas it may become handy. In general it may be handy: bidirectional relationships, but they do not seem to be used ubiquitously in technology, so... So long story, but just leaving the explicit 'forward' and 'backward' relationship references drawn out explicitly, might be a good idea. And I believe that I already came up with something somewhere, where I put a 'relational ring' around the two lines of a bidirectional relationship, to indicate they would be kept in sync automatically.

### 2020-05-12 Postponed Fully OO?

* Here and there I mention that the diagram notation could 'fully describe' an OO system. But I do not mention that it stops at arithmetic, comparative, logical operators and such. You might display that as an object graph. An idea was that 'Math' would be an external module that describes all the basic operators, but another idea was for expressions like that could be just included as bits of text code inside the diagram. Maybe not be too strict about this towards myself.
* Also when I look at Circle Construct Drafts, I see topics one might expect in a language spec, that are not addressed. I guess it does not have to have everything any other language spec might have…

### 2019-08-27 Postponed Simpler Rules

* 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 Postponed Mixed Writing Style Issues

* The read uses terminology in a very specific way, that is not shared with my peers, therefor not easing readers into the material.
* Rename the term 'Code Base' to something like 'Base Code' or 'Base of the Code'.
* I seem to have had several goals fighting over each other, 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.

### 2020-06-18 Postponed Static Notation

I also tend to go into when circles are displayed dashed. There I imply that symbols are dashed if they are used as a class, and if they are not, they are not. So members of classes would be drawn with a solid border. This seems to be, because the rule was: dashed borders was optional, if used, then it would imply the symbol is a class, or used as a class. But in other places in the docs, I already introduce variations on usages of dashed borders. Not describing it as a hard rule may help, a lot possibly. But the splinter in my brain is: I might have specific plans for the dashed borders. It might imply something is static. But I have not worked that out yet. And as I think about it, I think it might not work. If something's object aspect is not static, but something's class aspect is, a dashed border would not disambiguate. I like the idea of being able to configure each aspect of each symbol or member as static or not. I like the idea of expressing a symbol's being static as it being drawn with a dashed line. But as I think of it, it may not work unambiguously. I remember times where if I explore the idea, I might come up with a solution, for something initially seeming full of road blocks and perceived impossibilities. It's like a puzzle that I formulated myself. It is something I like about doing software. I think my idea about expressing the concept of static by having a symbol drawn in dasehed borders, was when the *object* aspect would be static. Not the class aspect or the value aspect. Maybe that is a base of a notation.

I think that for now, it might be a better idea not to express such notational choices where I use notational choices, because it seems not the subject at hand, and discussed elsewhere already and might obscure the main point.

### 2020-06-24 Postponed Content Changes for Overview Article

* The descriptions of each chapter may have more detail in it, than 'required', which may make it a tougher read. I find it a bit tiring to read it, though I might tire easily. I suggest some details might be moved or left up to the actual chapter content?

### 2020-06-01 Postponed Content Changes for Basic Diagram Elements

* 'Connecting Command Symbols' stories may be a bit long?
* Maybe there are so many rules in there, because of the idea that this would become the base for a patent? Patenting is sort of off the table now.
* Some things might be implementation details, movable to the Circle Construct Draft part?
* Certain rules might change and simplify, but that might not be for now.
* Outtakes:
  + Lines do have a direction, going from one symbol to the next, which will be explained later.
  + The access marks are sort of embedded into the language as it is right now. But these are all just a collection ideas, all just suggestions. A changed approach might not make things fall apart.
  + A line never gets a name. They are always called, for instance: ‘the object line of symbol B’.
  + Any object symbol might symbolize an object, a class or an interface.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Each object might serve as another object’s class, sort of functioning as its prototype. Any object might also provide the interface for another object, which may give another object the same exterior, while it might be different on the inside.

* Add a picture around 'Object Structure Solid, Class Structure Dashed', to demonstrate how things would look?
* Privacy issues in the paper scans?
* 'Object Symbols Drawn with Different Lines' starts with some information that may have been repeated more than once already before? Is there additional value to that, or might it just be removed or something?
* What might be a little odd, is that I spring the terms object, class and interface upon the reader in one of the first parts where I show the circle and triangle, but an arguably clear definition of these terms (though optional: these should be common IT terms) only later appears. Maybe it is an idea to move that clear definition more upwards. I tend to repeat the definitions, which may have some value. It might have the appearance that they are important, because of all that repetition, but it may make less sense that I do not just have one of those repetitions straight away where they are introduced.
* Question: What might be the purpose or scope of this Basic Diagram Elements article? Is it supposed to be a comprehensive overview of pretty much all the elements? Or the basic ones, to maybe get a good impression of this language formally? Because I think maybe I am missing some notation details. If made more complete, would it be almost a full summary of like all of it? I might be missing event notation, comments, system interface, I do not cover all the black boxing access marks, I also do not see assignment notation, etc. But maybe that is not the point: to be complete. But maybe a few key ones are missing, is my own personal feeling. Events, commends and a few access marks. Those maybe. Comment notation is not even described anywhere. Hmm. In that light it seems a bit arbitrary that I do show that line merge and symbol merge notation, which might possibly be deprecated at some point.
* Should I introduce a reference to parent notation in the Basic Diagram Elements chapter?



(minus the wiggly line)

* Extra rules for line drawing / access marks:
  + Drawing more diagrams on paper I remember why I invented the rule 'access mark not needed when connection from diamond to square is implied' and the other rule too: 'call lines can be solid'. They might make the language less 'pure', but they do forgive a 'mistake' you might make forgetting the access mark and that the line might be dashed between a diamond and a square: you almost always mean that when you draw a diamond and a square. I may reintroduce those 'rules', but then also express the doubt about them honestly. "Not a hard rule", "might make the language less pure", "might be forgiven if you use it that way", "could be used as a dialect that way", "might keep it purer and use dashed lines and access marks".

### 2020-06-11 Postponed Content Changes for Objects Chapter

* Attributes might get the 'cookie' notation.
* Attributes section: Could each sub-section be expressed in a picture in a way?
* Should I introduce a reference to parent notation in the Objects chapter?



(minus the wiggly line)

### 2020-06-15 Postponed Content Changes for Relationships Chapter

* The Relationships chapter is now before the System Objects chapter. I question whether the Relationships chapter would not better go after the System Interface chapter.
* The file '2. Relational Structure (Use As Cross Out Later).docx' might be placed in a 'previous versions' folder? I wonder how needed it is for Circle. It might be interesting to keep, not to rip apart as the file name seems to suggest should be done.
* I also miss, that classes might contain (static) data that looks like object relationships. This means I think, basically that you could see a mix of class and object structure. Perhaps a picture of that may take away some uneasiness and insecurity about applicability of this notation.
* The idea of multiple types of objects in a single list, might be pushed to the background?
* Special notations such as 'related to itself', maybe put those a separate part, so the main point may flow better?
* Is there a more containerish way to express things. For instance the 'Example': could I place documents inside Application with a sort of reference to parent notation?



(minus the wiggly line)

* Would that work for bidirectional relationships too?

### 2020-05-20 Postponed Content Changes for Parameters

* Maybe make the idea of ubiquitous bidirectional relationships between commands and objects less centric. Or perhaps at least diverge into other approaches, so attempting to make room for expressing what goes on in other languages, rather than replacing the construct by another take on it.
* I think maybe that bidirectionality is not of much use in this Circle Language Spec. It seems better off being placed in Circle Construct Drafts, because it just does not resonate much with other languages, programming seems to be able to live without it, and it seems alienating and full of implications, that seems to make concepts fall apart that did not seem to need to be ripped apart in the first place. I might be too harsh about it here. It is an interesting idea, I think, but it just complicates the idea of the diagram notation, which seems centric, not drastically changing how constructs work.
* The Imported Parameter Concepts could be tidied up. The ideas from the original Symbol Language.doc might be processed: discarded or embedded into the main text. Merging the texts of the conceptual explanation and diagram notation might make things more overviewable and shorter / less text.

### 2020-05-20 Postponed Content Changes for Execution Control

* Execution Control sort of seems an application of Circle Language rather than an intrinsic part. Arithmetic and other operators were supposed to be something like that too: an application of Circle diagrams. I wonder why I explain execution control in detail, while for math operations I leave that out. Maybe an article about that would make things clearer/complete/make you not wonder where all the operators have gone in this language.
* I saw 'execution flow' somewhere. I want to consider using that term instead of execution control. I just like the word flow. It makes me feel better than 'control'. But maybe it will somehow not work in sentences. Not sure. Maybe execution flow can mean the normal flow, while execution 'control' can actually change the flow. Still maybe flow as a general term feel nicer.

### 2020-05-20 Postponed Content Changes for Black Boxes

Black box is a notation I have reservations about. It works, but it might not be not elegant. 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…

There are topics, I probably want to move away from the Circle Language Spec at one point and go for something simpler. The whole friend notation is something I scratch my head about. I am not sure, if I like it or not.

I am not sure if the topics I dislike can be left in separate articles or something.

* Obs: One of the first few headings is that 'friends' idea in the 'Black Box Main Concepts' article.
* I think it might be hard to keep that separated from the rest.
* Maybe I should forget about trying and go for just longer articles, less articles.
* 'special reference:
  + When I think of a reference with a wavy line instead of a straight line, and all I think is 'special reference' then it clicks. I think I can just read the wavy lines then without much problem.
* I am still quite convinced I might want to introduce some general notation for more conventional member access control situations, perhaps derived from the 'full blown' system and with certain rules just implied, like 'a class's method has access to the private members of the class'.
* Half-way this one I start finding it difficult: Public & Friend Connections.
* From there on there are a few parts that I cannot easily read. I am not sure why. I just do not understand some of it, or the notation for some reason starts to become boggling to me, while earlier it was clear. I wonder whether it has anything to do with the way I explained it, or the idea itself is complicated or something. Maybe something in between.
* I do not worry as much though. I think it is appropriate to introduce a simpler scheme for the most common cases and do that at the beginning. Perhaps saying next, that it is based on a more elaborate notation covered after that, which explicitly denotes things that are just given in certain other languages, or a possibly broader range of access controlling possibilities than offered in other languages.
* I also feel that there could be more than one way to denote public and private with or without connectors. Not showing connectors at all might denote that something is private. Or not showing the members at all might imply things are private. Or showing a connector with a cross might say 'private', but that would imply the ones without a connector would be public? Is it bothersome to show public access connectors for everything public while all you are showing is the public members? Not sure. Maybe not.
* Maybe this detail is not even that important if you just present the options and leave the rest up to the imagination. My goal might not be to use this literally as described. I would be glad if anyone would even use bits and pieces, if at all.
* In the document Circle Language Spec Strategy one of the 'guidelines' for 'writing efficiently' was to not try to compare it to other systems too much. I am beginning to think that it might be useful to do the opposite when it comes to the Black Box principles. I think a demonstration of common cases in other programming languages might be more useful to start with, and then maybe later go into a systematic approach of maybe a broader view on what access modifiers actually do. That last bit might even be more something for Circle Construct Drafts, but I am beginning to think, that access modifiers from other programming languages, could be specific cases, of an underlying more general pattern and perhaps richer set of options. Why I now push that in the foreground in my story, is because I possibly would like to base notations of specific cases on this more general underlying pattern. The friend access thing… I might want to demote that to a special case, or specialized idea of special situations that might occur in other languages (friend access modifier in C++) for instance, and the philosophical idea of consequences of drawing a complete parallel between commands and objects. Maybe that idea might even be too alienating for Circle Language Spec. Maybe Circle Construct Drafts might be where you want to move that over to. But maybe if you just save it for last, and introduce it as the role I think it could have (not necessarily a big one), it might be OK to leave it in there.
* It's just that the Black Box story might be the chapter I am least satisfied with. Not sure yet.
* Note that the Events chapter has an item 'Black Boxed Events' in it.

### 2020-05-22 Postponed Content Changes for Interfaces

* Interfaces Main Concept, section Explicit Interfaces:



has something missing. In C# you can also refer to the parent object as if it were of the type of the interface, right? And it is also not an accessible sub-object, right?

### 2020-05-22 Postponed Content Changes for Inheritance

* Detailed.
* Also raises questions here and there.
* For instance the assumptions question "Where did the language lose its purity?" comes to mind in. I do see that in this chapter's text I am honest about some of these issues. It seems that later texts seem more humanized that way, what I am sort of aim to do in this project.
* Base redirection for referencing by base type seems to work with both dotted and dashed lines without conflict and when it is base class redirection or interface redirection the eye might prefer one or the other. "How to detect whether it is base redirection or interface redirection by analyzing the diagram?" is a question that comes to mind, but the eye and mind seems to detect easily which is preferred. There is probably an exact definition, but would it matter? When you would program something that would use circle notation, maybe, but. And just thinking about it, it probably would fall in to place somehow.



(Originally from the end of the System Objects article.)

(It is about the tiny piece of dashed line at the bottom/center.)

### 2020-05-18 Postponed Content Changes for System Objects Chapter

#### More Difficult Perhaps

* Perhaps rename the System Objects chapter to System Interface.
  + It seems more indicative of something more prominently visible. I tend to refer these issue as 'system interface' issues.
  + It seems to sort of get out of hand quickly when I do that rename. Instead of reformulating, I start thinking about changing the whole build up of things, which may be too much for now.
  + The story, how it flows, does not seem to make sense when the title is 'System Interface'.
  + I might want to revert this for now.
  + If the title of the chapter might change from System Objects to System Interface, it might make sense to introduce that system interface is first, instead of leading with a story about System Objects and then introducing the System Interface sometimes.
  + Something like: "The system interface is like opening up the internal workings of a symbol. <Something about what would be in the system interface.> Wondering what could be exposed through the system interface and how a symbol may work internally, seemed to lead to a systematic overview of things you might do with the language. The story almost seems like a design of a run-time: an implementation of the language, rather than
* Maybe it can be solve easily, by introducing it as though it would run as a run-time, but the idea of controlling system aspect is wider in use and can represent constructs from other languages too. Or something. Maybe just put it into context. Or multiple contexts: can be a run-time, can be representation of system commands already part of the language. In a run-time the idea was that an object lives as sort of reflective data, so a reflection-first environment, where reflection data is not accessed through an API, but more like a readily available, language-intrinsic secondary representation of an object. Having full fledged self-reflective object live in a run-time might not be the most optimal performance-wise, though quite flexible perhaps. An idea is to have the system interface be there if you need it, and optimized away if not needed or just be a view on it, not necessarily a non-optimal way of object representation in runtime. I am going too far with this now, I think. Too many edge-cases. Lots of ideas come to mind.
* System Objects seem to lead to Assignment, Connectors and Connections, but one is sort of the internals, while the other is just basics of notation, so not sure to keep them intermixed like that.
* I just keep thinking the chapter could be split into a neutral view on system aspects, leading to connections, connectors and assignment, and separating out the specific implementation, which I do not find uninteresting, but… it does not seem to be 'the language', it seems 'the runtime', which can expose itself through the system interface. It may even be demoted to Construct Drafts. It might be more than a draft, but… maybe for that reason put it in Broader View.
* (Also about Pointers): The distinction between system interfaces for references and for objects might also be derived from the impression that in the object reference notation it may be implied what is the object and what is the reference. I feel uneasy that I seem to change opinion of what makes a symbol a reference. It is a story of subtleties that might lurk under the hood, that could be the subject of the Pointers chapter and may be better pretending the problem does not exist. I have multiple interpretations I sort of equally like, that might not even be in each other's way. It may just create an odd rule set of when something is a reference and when something is an object. If I keep comparing it with 'text code' I seem to find answers. How it works in text code, I feel there is often a parallel to be found in this diagram notation. If it is a reference or an object may also be 'weird' in C#, so Circle might be excused for it too.
* Elaborations on reasons might be moved to Circle Broader View.

#### Less Difficult Perhaps

* System Objects article: Maybe show a few examples of how objects might be distinguished from references? Not all, just a few. There is one more clearly denoted in the Assignment article: it attempts to state explicitly how references might be displayed and how objects might be.
* Move Clone and Data aspects to elsewhere?
  + The clone aspect might be a bit more alien. Perhaps move it out of the main story.
* "The System Objects" section:
  + The definition of *Symbol* also seems to try and make it black and white not to confuse possibly overlapping terminology. I guess it might be my search for black and white definitions and avoiding ambiguity, that lead me to want to keep everything separated from each other.
* The **List** aspect is also presented as something specific, while I might make it a generic aspect of an object, but that idea is sort of part of the separate language design issue "list concept more generic".
* Name aspect notations might not be covered in all the articles and that might be nice.
* I also like where you can open the system interface and it just shows a dashed circle to indicate the class. I like that for the item class of a list.
* Alternative Clone notation as like a value assignment with a depth, might be something to put somewhere.

### 2020-05-18 Postponed Content Changes for Assignment Article

* What about getting the class and assigning it as a class? Would it be strange to call that cross aspect assignment? Maybe mention it. How would it look? I would say explicitly using the system interface? O here it is mentioned: "3. System Command Call Notations" just before the section "Explicit Get & Set Notation". I think it might be worth reconsidering that notation, and mentioning it in the main Assignments article.
* The implicit notation for cross-aspect assignment might be let go of? Might I just use the system interface notation? Ambiguity seems to arise a bit, for class to object assignment vs 'normal' class assignment (Get Object, Use as Class).
* Evaluate what the access connector for the Data access would look like compared to an imaginary assignment notation for the Data aspect and if direction indicators seem consistent and intuitive once gotten used to the way described in the Assignment article.

### 2020-07-21 Postponed Content Changes for System Command Call Notations Article

* Get Reference-Bound Class 🡨 New 🡨
  + Maybe this edge case might be left out. It seems to introduce something new, not the main point of the article. Keep it somewhere? Move it? To where?
* I am weirded out a bit, that I used assignments for the value commands but just connections for the object commands.
  + Value commands' arguments seem to need assignment notation to be an assignment compared to value correspondence, but object commands' arguments do not seem need an assignment notation because they are a connection. Yet I do indicate that e.g. Set Object would be called. There seems to be an asymmetry there that does not resonate well with me. Why would the connection to a parameter result in a system command call in one case (in case of object commands), but the connection to a parameter would not result in a system command call in the other case (in case of value commands). Did I at one point want to solve that by making value connections with direction be value assignments? Or is it something special with consulting parameters? I do not know, but I would like it to feel like it makes sense and is consistent. I was already in doubt whether a system command was actually the result of the connection to the parameter on the right. Maybe they are not? Then quite something seems to fall apart about the overview. I don't know. I'd like to solve it and make it clean.
  + Connecting inward could mean an implicit get command call.
  + Connecting values inward would that mean an implicit get command as well?
  + Would inward connections imply assignment?
  + I would imply a get.
  + My thinking error might be, assuming a hard rule that get and set goes together into an assignment and that would be the only application of getters and setters. I think I may at one point have forgotten that there is also a Get for *member access* possible e.g. Object.MemberA.MemberB might triggers 2 or 3 getters, while not necessarily causing any setters to go off. Those might be the only 2 cases that cause getters other than explicit calling. It may seem that every time I assume a set of options is complete, I may be mistaken. If I assume the set is incomplete and relax the story so that it might not be a problem, could often be the solution for conveying these ideas. But once aware of a different case, it seems to gain importance in my mind and I would like to be a bit more specific about it. 'A bit' might not only be polite talking here. A bit would do perhaps, just being honest (to myself) that I might not have it all worked out could be enough. It may not make this idea any more or less relevant.

### 2020-07-04 Postponed Checklist Reformulating System Command Call Notations Article

* Fine-tune styling.
* Evaluating if texts are in scope, possibly moving them.
* Marking trigger words in red.
  + With find and replace
  + Manually
  + Over-used words? "The"?
  + Rename "Object Get" to "Get Object" etc.
    - Did not do it in the pictures.
* Finding replacement words, possibly using online thesaurus.
* Put subjectivity in perspective.
* Change the orange markings (pictures, texts to move or change).
* Pictures:
  + Maybe draw new ones in one go, on paper and then scan it.
  + Sometimes borrow (pieces of) other pictures
  + Or simplify pictures by editing them.
* Merge conceptual/diagram explanations:
  + When I reformulate things top to bottom, I might at one point feel that the merging of conceptual explanation and diagram explanation is actually finished, because right now, often they are put near each other, but not really became one merged text.
* I have difficulty not wanting to change the story, while the aim is to just reword it, and I seem to experience an impossibility not absorbing the story. My conundrum might be solved, by adding indicators that everything is just a suggestion, so that no idea would disrupt anything, even when I may not entirely agree with it myself. Sometimes I might do the changes anyway.
* Going over the document again to reformulate.
* Spell check.

### 2020-05-22 Postponed Content Changes for Type Control

* It seems part of it is from the old Symbol Language.doc. It seems to attempt to demonstrate how the terms generic and explicit apply to the diagram notation. It seems to attempt to demonstrate any possible variation of it. But in doing so, it seems to explain things already explained elsewhere, but now insisting the terms generic and explicit are so centric they must be systematically demonstrated. That was a bit sarcastic, sorry, but I question if feeling the terms generic and explicit are important are reason enough to explain things twice from a different angle.
* What I sort of expected to see when I opened this document, was an explanation on how type safety might be enforced in Circle. But what I saw was the different forms of generic and explicit, that explain things I already saw in other chapters.
* I think maybe I try to prove in this text too much that the idea of generic and explicit works in Circle.
* I cannot say it is bad. I am not sure where to go with this.
* It almost seems to me, that pieces of these explanations could be spread among chapters Objects, Classes, Interfaces, Commands etc. Type systems seem sort of cross-cutting and centric in object oriented languages.
* I once seemed to feel it was a main issue. Now I am not so sure.
* I do kind of like it. I like systematic overviews. It probably is redundant information. Perhaps I can say that in an intro paragraph eventually. But for overview: I think it's nice. I think maybe I was prejudiced about it being mostly from the previous documentation Symbol Language.doc. Prejudiced about it being old. The argument that it seems redundant info I stand by, but I think it can serves a purpose.

### 2020-05-24 Postponed Content Changes for Object Order

I might move that Object Order issue to the chapter Objects. It's single lone topic now. It starts with the word 'Object', so maybe therefore Objects is the chapter for it. It's just right now (2020-05-24) it is unfinished and unfinished things tend to be put at the bottom of the documentation right now and do not want to do anything about it right now, I think.

### 2020-06-11 Postponed Content Changes for Command and Classes Loosely Coupled

* (Moved to Circle Broader View\Classes)
* What might be mentioned if the text would be built up from the ground up? I think I would just talk about the static concept, because that seems to me the number 1 candidate for what classes might have a lot to do with and might possibly determine the dashed notation and what makes room for the distinction between classes and prototypes and might give a clearer definition of what classes and static structure might be conceptually and have that reflected in the diagrams more directly. I would move the more creative concepts away from the Circle Language Spec.
* I have questions about the dashed line usage of lines of the smaller squares:



I do not understand why the class would point out the external command definition with a dashed line, while the objects point out the external command definition with a solid line. I think I might have followed a style choice more than an exact depiction of things? Would both lines express something that kind of works, leaving the options open for a stylistics choice to have class structure be more with dashed lines and object structure be more with solid lines? I suspect that I might leave this doubt open.

I am a bit amused by my own typical behavior, because half an hour later, I see that the explanation was already in the text.

I do feel it might be quite something to burden the reader with. I am hoping for a more clear meaning for dashed shapes in the future.

* Another point of doubt might be:



Why is the line on the right no extended to the circle, so class structure and object structure look more alike except for the dashed and solid lines?

### 2020-04-19 Postponed Conversion to MD

* I thought MD would be the way to go with this project compared to docx. But I have doubts if MD is the better way to go. I keep getting stressed out about those doubts. It makes my decisions on what to do with the documents (formatting) harder, because I just am so unsure about this. What I would like is to have a good MD editor. I just hope some of the difficulties editing MD compared to docx can be taken away by a good tool. Also: I think MD will be vieweable and better indexeable on the internet, but I have not really thoroughly seen that confirmed. The main problems with MD editing I have is navigating headings and sections around and viewing the outcome of my formattings better as I type the MD in. The doubts about indexing: I know GitHub can show the MD in nice formatting, but will a lot of MD's actually be nicely navigatable? Will those MD's actually be indexed, or does that only count for e.g. the REAME.MD in the root dir of a git repository? another doubt about MD is: can I be satisfied with some of the tools that convert docx to MD? Will that all be doable? It's so many articles. Can I do it in bulk? Will I merge more articles into one making converting one by one by hand better. How does it work with links but especially with images? All those doubts made me postpone decisions about converting things to MD, but that makes me burdened with formatting docx's making them look nice, which is not that useful if I am going to convert all of that to MD anyway. That last argument does not seem very strong to me. If I know that MD indexes well, that might already make some decisions to go for MD easier, when I decide e.g. about small documents.
* So with all those doubts, I want to explore MD options a little.
* Main points:
  + MD editor
  + Will MD show and index better on the internet?
* Notes:
  + Trying to research this puts a strain on me.
  + I found out relative links to images work nicely in MD on github.
  + I cannot find if google will index the content well.
  + I do know that clicking an MD in GitHub leads to nicely readable page. That's something you cannot say about docx's.
  + That last point may actually convince me already to make MD's more often.
* MD editors:
  + Visual Studio code has a preview view and outline view: <https://code.visualstudio.com/Docs/languages/markdown>
  + <https://duckduckgo.com/?q=MD+editor&t=opera&ia=web>
  + <https://marketplace.visualstudio.com/items?itemName=MadsKristensen.MarkdownEditor>
  + There seem to be options.
* Alternative: Convert the Future Sub-Project Summaries to MD.
  + Pro: Those future docs are short and don't need much formatting.
  + Con: MD is harder to edit (?) > Needs research.
  + Pro: MD is more accessible on the internet (?) > Needs test.
  + I would like to have the arguments for and against better proven.
  + But it is also not directly my goal right now.
  + It would help me to know if MD really has those advantanges. Otherwise I would keep being confused about whether or not I should convert to MD.
  + But pragmatically: That was not what I was doing. It was formatting the docx's, what I was doing. But pragmatism is sometimes something I feel others have determined for me. Maybe knowing something for sure is more pragmatic for me, because the doubt causes me so much stress.
* Concession: Do both. First format the 4 remaining Summary's. Then research MD more.
* Conversion programs:
  + Pandoc: I did not like it. It was expecting a user interface, but I got a command line interface. Also, the output from docx to MD is not what I expected. I got quote markup > for just indentation and a dashed bulleted list got the dashes escaped or something \-. The check list style for GitHub that's kind a sorta already MD format, got escaped it seemed. I get not everything is unambiguously translatable, but this was too far removed of what I would type in myself. So sorry, pandoc. Moving on.
  + I might as well just copy paste to a text document and then do it myself.
  + But I am going into reseach again, and that is not what I want to deeply into. I do not have the energy.

### 2019-08-05 Brainstorm 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.

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

How do you even license documentation? I wanted to throw an MIT license against it, but that is for code… hmm…

<https://dreamsongs.com/IHE/IHE-50.html>

My brother's first name is in some sub-project planning doc. I can take it out of the current version. I should ask him if he is OK with it staying in the source control history or if he wants it out of there. Also in '2009-06 01 Circle Language Spec Plan Part B, Sub-Projects'.

Is the summary of the C and C++ in Summaries of Other Languages an intellectual property problem? I summarized the languages using MSDN as source information. Hmm… If I didn't mention the source information it would not look like an intellectual property problem. Just I read stuff about a programming language and wrote stuff down… not sure.

Relational Structure might mention a relationship example which might not be kept in there.

### 2020-05-14 Postponed Circle Broader View / Construct Drafts Neater Folders

* Make folder subdivision neater, now that the topics have found a new place. ("Circle Language Spec", "Circle Constructs Drafts", "Circle Broader View" and "Framework".)
* Circle Language Spec has priority over the other main folders.
* The others are postponed for now.

### 2019-12-29 Postponed Circle Broader View / Gap Lifting

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. I thought that was interesting to think about.

## Done

### 2020-07-24 Done Brainstorm Reformulating Commands Chapter

I say Commands *chapter*, but previously I seemed to have started one detail level deeper: article-level rather than chapter-level. I wonder whether I will do that again. For that I might look at the articles and how large they are, a bit of the concrete content.

Steps:

* Going through chapter' articles' content to see what the scope-status is of the issues.
* Reading over the 'Postponed Content Changes' notes in this planning document.

Some of the chapters might be not main issues, but according to my opinion, they do decorate the topic making it come alive, though the optionality of the issues might be fact.

The Command Redirection article might have issues that may be moved to the Pointers chapter. Some single-step redirections might be a main thing, useful to inform at this level, but recursion might be questionable to belong there, and the 'Target' of redirections: those topics were earlier on moved to the Pointers chapter from the Objects chapter or Classes chapters for instance, so it would make sense to move the Command-related 'targets' topic there too.

This may be out of scope: but the parameters topics that are brushed over in the Commands chapter seem odd in place, because there is also a parameters chapter that follows later. Perhaps some details about parameters might be moved from the Commands chapter to the Parameters chapter, but perhaps there is just a mutual dependency between concepts (overlap) that a sharp separation might not be what to go for.

Misc Issues:

* Parent Controls Its Sub-Executions section:
  + might be a rule that might be scratched. It might have implications if this is not a rule, but the solution might be obvious, if you let go of purist ideas about how it works in 'regular' object oriented programming.
* Command Referrers section:
  + Might be moved to Pointers chapter or Relationships chapter, not sure. I moved 'Referrers' sections before to somewhere, whereas they were spread over different chapters before. I think Relationships chapter.

### 2020-07-22 Done Prioritization for System Objects Chapter

* Reflection:
  + It seems trying to put content out of scope and a more clear-cut version in place, especially that second part, put quite a storm in my brain. Something I agreed was too much for me right now. It may have been collision of circumstances, that I wanted to take putting content out of scope a step further by replacing it with a different variation too, and secondly also the chapter might be one of the harder ones. In that process I seem to have forgotten what my focus was: tone change. My thoughts are still circling around topics now, that I had earlier chosen to leave untouched for now. I think to move focus to where I want, I might need to actively imprint in my mind to postpone topics and remind myself that postponing them is OK.
  + What sort of went strictly according to plan was: I did replace the implicit notations with explicit ones.
  + What went sort of wrong might be: Doubts about the systematics and their completeness and correctness seemed to have driven me into a brainstorm of unresolved details. I find there is not much wrong with that, if persistence leads to a well polished end-result, but some might also call it 'losing oneself in the details'. The effort that would come with that persistence is not estimated to be within my capacities right now. I sort of have to be gentle with myself.
* Steps:
  + Might re-evaluate if the content changes might be picked up or might be postponed.
  + Doing tasks after prioritizing content changes.
  + Steer focus back to tone change.
* Strategy:
  + Maybe postponing even more could perhaps make the transition back to the main focus quicker (and easier).
  + Moving the whole chapter way further down might also exaggerate the situation, so not be tempted to stay fixated on this topic.

### 2020-05-18 Done Content Changes for System Objects Chapter

* Maybe rename "Object Get" to "Get Object".
  + There may be some naming purism that might have slipped in, which might be sort of subjective. Would "Get Object" not read more easily instead of "Object Get"? Ease in conveying the idea might be a good argument to write it that way. Fortunately no choice is set in stone anymore, which seems a bit of relief, meaning: even if you would name it Object Get later for some reason, stating it differently now, would not stand the main idea in the way. I may not have been entirely conscious of that when I wrote it. I was looking for rules to hold on to in a perhaps complicated task. I get it. But the idea can relax now.
  + Might do for whole chapter.
  + But skipping the images?
  + / Might not include the Circle Construct Draft version?
  + Done for System Objects article.
  + Done for Assignment article.
  + Done for System Command Call Notations article
  + Might do for Connectors & Connections article.
  + Might do for System Objects Misc Issues article.
  + / Might be no occurrences in the List Concept article.
  + / Might not do for the System Objects Ideas article.
* Move the System Objects chapter further down in the chapters?
  + Brainstorm:
  + I seem to be in doubt about how parameters work now.
  + So I have the feeling I am missing some basics. It may be better to describe them first.
  + I seem to be regularly in doubt about when which system command is called and under which circumstances. Not only does the chapter not seem to describe all those things in detail, but also it seems too early for that in the story. And also the basics of the language do not seem to depend on this.
  + I would like to see this through to an acceptable text for now (2020-07-22), and feel comfortable parking the content changes for later.
  + I may be saying this, because I deep dived into this topic, but this seems one of the more difficult topics.
  + It went from sort of a fun way to notate object oriented concepts into diagrams, to an abstract, difficult, precise, specialistic view, the specifics of which seem to be missing something and of which the correctness seems difficult to verify for me right now.
  + And probably because of this insecurity, it is just difficult for me.
  + I went beyond scope of changing the tone of the text, because I really wanted some of the notation constructs put out of scope, that I have second thoughts about.

### 2020-07-21 Done Content Changes for System Command Call Notations Article

* Assignment direction might be reversed in the section "Explicit Get & Set Argument Notation". > Checked both Circle Language Spec and Circle Construct Drafts.
* Add a sentence? The "Explicit Get & Set" section goes into notations of the New and Add commands as well, even though the title of the section may not say it. Change the title? Or excuse it in the content?

### 2020-07-21 Done Content Changes for System Command Call Notations Article

* Making structural overviews with explicit notations rather than implicit ones:
* I seem to have just finished "System Interface Calls with Arguments" section:
* I think what I did next is draw pictures for the "Explicit Get & Set" section.
* I might have scanned the pictures. Or have them just on paper. I could check.
* I seem to not have scanned them yet.
* Might cut and paste images from the paper scans into the "Explicit Get & Set" section.
* I might go to a next step, which might be adding the next section, based on the original text, but then changed to the currently preferred notation.
* I seem to be missing the original pictures in the original notation of Explicit Get & Set (with Arguments). Did I remove them, or did they never exist? I think I accidentally removed them and did not keep them in Circle Construct Drafts. I might look in the source control history trying to get them back for sort of archival purposes.
* Found part of it in a previous version of… going back in history on the main doc, might lead to the actual original instead of an already altered one.
* It seems there was no systematic overview of any system interface call with explicit arguments in the version before. So I sort of went on a ghost hunt.
* Might there be things I want to polish up about the Circle Construct Drafts version up until the "Explicit Argument Notation"?
* There are some 'orange' markings left for extensions with more content, but I might not are much about that currently, because I chose to focus on Circle Language Spec and leave Circle Construct Drafts in a bit of a 'drafty' state if I must.
* Clone in the "Explicit Get & Set" section seems to be missing its depth arguments.
* Explicitly drawn out assignment command:
  + Made scaffolding to be finished up.
  + The next appropriate section to do seems to be the one where I extrapolate the assignment notation step by step towards explicit display of what goes on inside it. I sort of promised myself to not go as far as to artificially prove that the explicit notation is no good. I would like to go from assignment notation to explicit argument notation inside an execution 'diamond' of the assignment call. But no further.
  + An explicitly drawn out assignment command might be interesting, but then maybe not go as far as to make it as rich as possible in an attempt to exaggerate that it is impractical, but instead maybe the most simple notation of the explicit notations, to be honest about what is being done.
  + Maybe not many steps are needed. I think: a) an (object) assignment b) an explicit get and set call c) an explicit get and set call inside an assignment execution.
  + The Get Object and Set Object calls displayed in System Interface Call with Argument notation sort of miss the parent around the object reference used in the argument. I do not know yet how I feel about it. It seems OK notation, perhaps even bad to put a parent around it, because it might suggest an extra access call, but I did sort of promise to use the 'has a parent' notation when something is might be a reference-bound aspect… It's just that it seems to look odd when comparing notations. Not sure yet.
* I may want to read over the article "Connectors & Connections" and see how the implicit notations relate to it. A quick scan lead me to this reasoning: Some connect*ors* seem just an indication of what a connect*ion* looks like. It is introduced as an alternative of displaying a system command in the system interface, and also some connectors seem to derive from the implicit notations, which might be OK with explicit system interface notation instead.
* I might want to steer back to evaluating what consistent, explicit alternatives might be for practically used situations for which an implicit notation is introduced.
* I think I want to move the whole article to Construct Drafts, then build up an alternative article containing only the notations left that I might want to introduce as part of this language spec.

### 2020-05-18 Done Content Changes for System Command Call Notations Article

* I actually feel uneasy about the notations introduced there at all. I want to try and avoid exaggeration, but this seems like a opening a flood gate to ambiguity. But that is not the question at hand.
* My main concern currently is whether basically any of the implicit notations other than the assignment notation are something I would enjoy keeping.
* My main 'objection': I do not feel that this would be an objection anymore: "Then remains the question whether to show the other members of the system interface as well, or only to show the members of the system interface, that are actually called." Either option seems ok to me, and I think it could be practically managed when to display which.
* The 'end result' would make system command calls look quite like assignment notation.
* My objection to the notation now might be to not introduce implicit slightly ambiguous notation for something not used very often (never?) that already seems to have an alternative notation, that is not ambiguous, consistent and adequate.
* The notation is indeed nicely consistent with the assignment notation. Even better now, I guess, consistent with the flipped use of access marks to indicate direction?
* I seemed to build in less wiggle room before in my writing. The text says: "Most of the notations above will never be used." Ok, so I did say 'most' which is not a definite. Some of them would be used.
* The text does seem to neatly lay out the use cases, where this implicit notation might be used.
* I might want to draw those use cases out with system interface notation. Then evaluate whether I am content with system interface notation and could do without the introduced implicit notations.
* Options are a bit dizzying for me now.
* I have some doubt, but I think I do not have to have that doubt. I scanned over the topics. I think moving this construct out will be OK. After I drew the 'alternative' notation in the system interface.
* Drawing explicit notations for many implicit notations mentioned.
* The New, Add and Remove commands might be explicitly called unlike Get, Set and Use commands, which can be implied by assignment notation, so I might evaluate if those could use an implicit notation, or look good enough to me using system interface notation.
* Flattening the aspect triangles to a flat list of commands might make system interface notation a bit more practical. > Say in the article that you go for that design choice and show flexibility of choices in this language.

### 2020-07-04 Done Checklist Reformulating Assignment Article

* Fine-tune styling.
* Evaluating if texts are in scope, possibly moving them.
  + Moved pointer assignment topics to the Pointers chapter.
  + Cross-Aspect Assignments still seems to have pointer-related parts.
* Marking trigger words in red.
  + With find and replace
  + Manually
  + Over-used words? "The"?
* Finding replacement words, possibly using online thesaurus.
* Put subjectivity in perspective.
* Change the orange markings (pictures, texts to move or change).
* Pictures:
  + Maybe draw new ones in one go, on paper and then scan it.
  + Sometimes borrow (pieces of) other pictures
  + Or simplify pictures by editing them.
* Merge conceptual/diagram explanations:
  + When I reformulate things top to bottom, I might at one point feel that the merging of conceptual explanation and diagram explanation is actually finished, because right now, often they are put near each other, but not really became one merged text.
* I have difficulty not wanting to change the story, while the aim is to just reword it, and I seem to experience an impossibility not absorbing the story. My conundrum might be solved, by adding indicators that everything is just a suggestion, so that no idea would disrupt anything, even when I may not entirely agree with it myself. Sometimes I might do the changes anyway.
* Going over the document again to reformulate.
* Spell check.

### 2020-05-18 Done Content Changes for Assignment Article

* I might want to be ok that changing the notation, I might leave the other documents untouched. In that light, it might be worth mentioning some idea changes, maybe at the bottom, just in case.
* Consider whether to switch the direction sign, so assignment call looks more like the result of the assignment.
* Assignment notation might raises an issue. An assignment almost looks like the result of an assignment, except for a diamond that crosses the line and the direction is flipped. Assignment:



Would result in:



It may be worth to reconsider. Maybe the indicated direction in the assignment notation might be better off being the direction of the result, rather than of the direction of the assignment. But maybe it is not as simple as that.

* Show a value in the object where value assignment is demonstrated?
* How this assignment/result direction flip problem came to be might be that at first there might only have been value assignment notation where it seemed to make sense. Then e.g. object assignment may have been derived from that, which may have introduced this assignment/result direction flip problem. Did I introduce a new problem working backwards? Does assignment notation direction not make sense for the value assignment notation, or is it just something I might be used to? Especially when I replace it with an actual arrow? My attempt here is to introduce more consistency. Formerly I think I tried to introduce intuitiveness over consistency, which might be a contradiction. Consistency might lead to more intuitiveness.
* Consider not making an exception for the value aspect, that the call symbol might not be shown. The main reason that it might still be an idea to not show the diamond in a value assignment, might be because if value correspondence would have a direction, what else could it be but a value assignment. I might just not entirely agree that a value line should always mean assignment, because then you lose something: the possibility to differently express value assignment from value correspondence.
* Perhaps show the result of an assignment next to the assignment call.
* "Because assignments might be so common, they might be given an implicit, simplified notation in Circle."
* Using the words 'source' and 'target' might not lead to an intuitive story.
* Once I invert the assignment direction notation cross-aspect assignment notation seems to fall apart? > No still has some intuitiveness to it, is my opinion.

### 2020-06-26 Done Reformulating System Objects Article

* Fine-tune styling.
* Evaluating if texts are in scope, possibly moving them.
* Marking trigger words in red.
  + With find and replace
  + Manually
  + Over-used words? "The", "related"?
* Finding replacement words, possibly using online thesaurus.
  + "Relation" => "Relationship"?
* Put subjectivity in perspective.
* Change the orange markings (pictures, texts to move).
  + Maybe I forgot to merge the conceptual description with diagram demonstration.
* More over-used words / trigger words?
  + "The"
  + "Full"
    - I still might be stating half-truths where I use trigger words. I do not seem to notice it at first. I just said 'full system interface' but later I introduce more members, so is the word 'full' appropriate? This raises insecurity with me. I often do not even seem to see it happen.
* Going over the document again to reformulate.
  + Temporarily taken out:
    - It could be used by other system objects to wrap an object reference into context.
    - A reference may come into play when objects relate to each other.
  + I have difficulty processing the info/reading.
* Pictures:
  + "System Interface for the Class Aspect"
    - Not many pictures here.
    - Each sentence might call for the question: how would that look in a diagram? Some of it might not even be clear to me. I might want to consider one by one.
  + Maybe draw new ones in one go, on paper and then scan it.
  + Sometimes borrow (pieces of) other pictures
  + Or simplify pictures by editing them.
* Merge conceptual/diagram explanations:
  + When I reformulate things top to bottom, I might at one point feel that the merging of conceptual explanation and diagram explanation is actually finished, because right now, often they are put near each other, but not really became one merged text.
  + More prominently the possible merging of System Commands and System Interface sections.
  + (I wonder if System Commands and System Interfaces could become 1 section, because is the System Interfaces section(s) the visualization of the System Commands? I tend to not like conceptual explanations without accompanying diagrams anymore, so perhaps.)
* I have difficulty not wanting to change the story, while the aim is to just reword it, and I seem to experience an impossibility not absorbing the story. My conundrum might be solved, by adding indicators that everything is just a suggestion, so that no idea would disrupt anything, even when I may not entirely agree with it myself.

### 2020-07-01 Done Content Changes for System Objects Chapter

* The chapter starts as if it is a run-time.
* "if Circle would function more like a stand-alone language, rather than mostly a diagram notation"
* The Related Items and Related Lists collections inside the System Interface might be handy for reflective/aspect oriented/relational model logic. But in essence you might only need one collection of 'sub-objects' (that term might questionable). Possibly, certain characteristics that a sub-object might have could turn it into a Related Item, Related List or perhaps even an Attribute. And commands, etc. There may be one 'bag' of sub-items, while you may query for ones with certain characteristics. A set of those queries could be: Commands, Items, Lists or Attributes. But at least 2 times now I guess I changed my mind about the subdivision of those 'queries'. It looks like the .NET Type type with its members like Properties, Methods, etc.? That parallel feels nice. These might even be framework extensions. I wonder how to present this. Presenting alternatives. Leaving in this design choice or changing it? It should maybe be one of the use cases of the system interfaces. I think those might deserve mentioning. That this reflective data might be used for software design based on a relational model if you just see it as Related Lists and Related Items, that might be nice. But due to the detailed construct subdivisions that commands might cause, the view on the relational design might be lost if you do not put some sort of filter over it, that excludes maybe anything going on inside a command. These are just options. I would like to explore them. I am not sure if this would make the text unnecessarily long. Let's just see.
* I think I like the way the specific approach might shed light on flexibilities, because it could mean openness to alternative ideas, at the same time expressing that other variations are also possible too. Even though things in System Objects might seem specific implementation issues, I tend to want to keep all of that in there, because it seems a nice demonstration of how things might be (on a deeper level?)
* In the System Object article, there seems prominence of the 1 and n multiplicity concepts, that might only be interesting once you start talking about relationships or reflection or aspect oriented programming.
* Rename?
  + "Reference-Class Get" to "Reference-Bound Class Get"
  + "Object-Class Get" to "Object-Bound Class Get"
  + The reason is that the terminology seems to get dizzying and confusing. For instance sort of clashing visually with terms like "Class Redirection" and "Class Reference" (already sort of ambiguous).
* I frequently have the opinion that some texts may have design choices not necessarily relevant to the notation design, that are presented as hard rules, while I could perhaps think of a few alternatives from the top of my head. Just admitting it is not set in stone might help. / (Perhaps leaving base code design / the design of a run time out of the story somehow, might help? I am not sure.) The example that is given might help, because it might give pointers towards various things you might use a system interface for, ideas that might otherwise not be considered, and perhaps relevant to the purpose in mind with system interfaces. Something that could be considered, though, is to then not try to design a perfect base code in writing, but to make some things simpler on purpose, since this seems for demonstration purposes, rather than to formulate a fixed set of rules anymore.
* Clone (2) seems a practical variation on the value aspect, because the case where there is a parent object, whose child objects are values or reference and those elements together might need a copy ('MemberwiseClone'). The value aspect might only copy a parent object's value, which it might not even have, just through its sub-objects/child value objects. But the depth parameter might be a bit creative. I think expressing this use case (maybe a bit more concisely than above), might help appreciate that the idea is proposed.
* The Execute aspect might be more generically applicable. In the section "Execute Aspect in the System Interface" I might say any object in Circle might become executable, instead of fixating on an object's definitely being a command or not.
* "The System Objects" section:
  + I find terminology introduced dubious and ambiguous. For instance: related item would be a pointer, related object would be an undecorated object pointer: what would the difference be? The Related Item makes the connection with its parent. But how does the term Related Item even mean that? I just seemed to have picked some words and assign meaning to it. I think maybe the meaning of the terms could be kept pure. I might not need to abuse the term 'Related Item' in order to define that a parent object manages the relationships with its related objects. Item and object are almost synonymous, but as I translate that to terms related item and related object, all of a sudden they change meaning? Nah, not a great plan perhaps. I just wonder then: How am I supposed to present it then? They do seem clearly distinct concepts, just not clearly distinct terminology. This might be an age old problem in computer programming: finding good names. Basically the concept of 'wrapping a pointer' may need a name. Maybe Related Item wraps a Reference or is that abusing terminology again? Maybe not if you do not try to define a term but just say the concept the parent manages the connection to the related item, and that a Related Item might wrap a reference to the related object.
  + Could I just pick a term and then say: these are all basically synonyms?
  + That items in the related lists collection should be circles, well, maybe they might be nonagons instead. But I might keep it how it is now: circles. It might fall under the language design issue of "List concept more generic".
* Remove the section "Overview of System Commands for the Class Aspect"?
* / If value assignment is in assignment notation, then the notation for value connection with an access mark might be open for use for something else. If I make it the same as the basics of other parts of the language, this could mean something like a pointer to a value? Not sure if this is a useful concept.
* Are connector notations consistent with the assignment / reference direction changes? It would be nice if it were. Then 'everything is the same'. > I think it is now.

### 2020-06-26 Done Scoping System Objects Article: Move Pointer-to-Pointer Issues

* Reference aspect does seem used only for pointer-to-pointer situations? And without the context of pointer-to-pointer situations, the concept of the reference aspect might also be hard to place.
* More prominently might be the pointer (to pointer) issues, which I promised myself I would move to the separate Pointers chapter. I think most of that is a bit scattered around the System Commands and System Interfaces sections.
* It seems iffy to me, that all point-to-pointer issues should be moved. The distinction between the Reference aspect and the Object aspect might be desired for a certain completeness. (Evaluating this, I am currently looking specifically at an image under the section "The Full System Interface for Related Item"). Perhaps pointer-to-pointer *details* might be moved.
* That there are references and that there are objects, might be key to understanding a bit about this particular object oriented take on things. But as it starts getting a bit hard on your head with pointer-to-pointer edge-cases, maybe I can extract those topics and move those.
* It is becoming harder for me now, because it is iffy. I can notice it inside myself.
* Here I start scratching my head about whether these are details I might leave out of the main story: under "The Extra Commands & Overloads":
  + **Set Object** 🡪 **Set Object to Other Related Item**
  + **Set Object** 🡪 **Set Object to Other Related List Item**
  + **Object Get** 🡪 **Get Object which is Another Related Item**
  + **Object Get** 🡪 **Get Object which is Another Related List Item**
* I see now that the section is named "The Extra Commands & Overloads". So it seems isolated, and perhaps separately movable to the Pointers chapter?
* "Get Class which is a Reference" might have overloads that are perhaps pointer to pointer situations?
* "Set Class to Reference" might also be a pointer-to-pointer situation.

### 2020-06-26 Done Brainstorm Next Step

Up next might be the System Objects chapter. That one is rather large: 67 pages. I wonder if I am up for it. Maybe I would want to skip it and do another one first? Commands is larger 79 pages. Hmmm… and its content seems if-and-or-but'y. Parameters is 18 pages, some of which is material I think I want to change the idea of. At one point I would like to go from one topic to the next more sequentially. I think maybe just accept that there's quite some content. I think I am just going to start with the first article of the System Objects chapter.

I had checklists for changing the wording. I seem to also wander off that topic to try and fix content. Usually because then I might have less words to reformulate. Scoping might put things at a side line, which helps get through the stuff to do.

### 2020-06-26 Done Reformulating Relationships Chapter

* Redo trigger word analysis?
  + I do think I see the pitfall of trigger words.
  + Maybe do another trigger word analysis.
  + I think I wanted to reconsider how I perform these trigger word analyses.
  + Perhaps a standard list of words might be marked.
  + I might want to reconsider the list of standard words, that I might mark with find and replace.
  + Then manually spotting words of resoluteness.
  + Replacing trigger words.
* I do not seem to emphasize "bidirectional" much in the text of section "Bidirectional Relationships Between Objects". In "Bidirectional Relationships" from "1 to N Relationship" onward I also do not seem to mention the word "bidirectional" very much anymore. It might be welcome to repeat that every now and again, perhaps in each section have the word "bidirectional" in there at least once if it has anything to do with that.
* Possible over-use of the word 'related'?
* Read over top to bottom and made minor changes.

### 2020-06-19 Done Deprecating Double Dashed Ring Notation for Relationships

The Basic Diagram Elements chapter may mention the double dashed ring notation. I think I remember it already expressed doubt about it. I may extend Basic Diagram Elements with the proposed relational ring notation, and maybe carefully express preference for it.

Plan:

* Deprecating double dashed ring notation in the Relationships chapter.
* Using (single bordered) relational ring notation in the Relationships chapter.
* Commenting on both notations in the Basic Diagram Elements chapter.
* Not changing the other chapter's content. That might come or should be excusable.
* It may deviate from other tasks (like tone adaptation). I do think it is in line with overall goals. One thing might help the other.

### 2020-06-19 Done Brainstorm Bidirectional Relationship Notation

I have some difficulty dealing with the following: I do not seem to have much hope anymore for the double-dashed bordered ring notation that would be proposed in the current text. I tend to like the explicit two line notation with a relational ring around it. I now 'fail' to see how that would be more convoluted. The double-bordered ring seems a tiny bit far fetched in its being derived from *automatic containment*, which does not seem to be part of the Circle Language Spec anymore. The derivation might be also difficult to understand without background about this difficult topic of automatic containment, which was (temporarily?) abandoned over its complexity and unresolved issues. (Might be an idea for the future, but not now was the current thought.) I am now failing to see how merging the lines together would create clarity or terseness. It was supposed to be easier on the eyes, less detail, for something that was supposed to be used all over the place. I am not so convinced it will be used all over the place, and not so convinced that the notation proposed is clearer. I like the lines to be a bit purer in what they express, as in, aspect correspondence. I, and maybe others at some point, might be confused that line connections change meaning just like that. I sort of like that you can see that the relationship from an OO point of view looks like two properties closely related. That's quite some talk about why I am not convinced about this notation.

It sort of feels wrong to put effort into texts about a notation I do not seem to like that much anymore.

Possible consequences of changing the notation now:

* Might deviate from reformulation efforts and make that slower or in jeopardy of coming to a halt.
* Might be many places in the docs to change the notation in (disputably).
* May take on more work in a detour task.
* Not currently the focus / main goal.

I however, also feel that a perceived excess in effort going into text that might be deprecated, could be an issue I might not feel entirely ok with.

Maybe a bit of observation over the text, to see where the notation is used.

I suspect not in many places. I think many topics might be described independently of the bidirectional relationship notations.

The double dashed ring notation might be used in:

* The Relationships chapter itself.

This brainstorming also costs energy.

I might solve it more easily. In the Relationships chapter I might switch over to the relational ring notation. Then when other chapters might use the double dashed ring notation, the arguments are already given: the Relationships chapter no longer would mention it, the Basic Diagram Elements chapter would express the preference of relational ring notation. Then any piece left that still might use the 'old' notation might be excused. This may be a side-effect of stating things more 'humby' instead and being honest where there is doubt.

I used to think expressing doubt or uncertainty might make people dismiss an idea. Now I do not feel like that anymore and might be better off reverting to a way more close to what I was born with. I might want to hold on to that. The language style I try to introduce might not be one that does not come naturally to me. It might be one that clashes with my original language style + other people's demands as I perceived them. So perhaps adopted behavior rather than original behavior would be what makes it difficult to switch language style. To have difficulty finding the balance in. If I can find a point in the past where honesty and humility was my go-to kind of voice…

### 2020-06-18 Done Brainstorm Reconsidering Writing Style

I seem to not like it when I make the rules for reformulating things so strict. It is sort of paradoxical too. The text produced is supposed to be not as strict, but the rule for making the text not strict, is strict.

It was supposed to feel like hobby.

I think I would like to change the rules by which I was reformulating these texts.

I think I want to be more ok with words that seem to exaggerate or be quite strong. For instance "distinct", ''rule", "defined". Maybe I want to keep those words in there.

I think I am noticing that many words just are black and white. If I use a thesaurus I get to see words that might also be quite strong.

Maybe I can leave it with adding maybe's and perhapses and leave words that seem to express it strongly, in there.

Reasons for changing the way to do it might be:

* It seems to make me anxious being hyper aware of these 'forbidden' words.
* The text seems to now lack flair, and might not grab attention as well anymore.
* I feel I use more in between words, that carry not much meaning. It seems wordier than necessary.

I used to aim to be concise. In doing so, some thing might start to seem a bit commanding or something. But I did do my best then to make the text clear. Maybe this is not such a bad thing and maybe I was not doing that badly at it after all. Maybe just adding some wiggle words is enough, so that I might form an intermediate writing style. Perhaps that will be concise, strong, but leaving room for other options.

I would like to feel comfortable writing again. I would like to not do it 'wrong' like before, if that was even wrong.

### 2020-06-15 Done Content Changes for Relationships Chapter

* The Referrers list might be only part of the system interface.
* Bidirectional Relationships Between Objects:
  + The idea of one reference forward creates one reference back may not be universal. In one implementation it might. But I have made implementations of bidirectional relationship management code, where this does not seem to be the case. I think the notation might be more neutral or keep more options open, to perhaps accommodate a more honest, open and inclusional view on the topic. Perhaps the framework dependence matters. Perhaps an admission of framework dependence is enough. / But I also would like to explore how bidirectional relationships between objects might look in more possible implementation situations.
  + Framework-dependent:
    - For bidirecitonal relationships between objects might be different implementations of synchronized bidirectional relationships, that might invite other notational subtleties. This is just one idea of a way object relations might be subtly different from class relationships. Bidirectional relationships might be managed certain ways by certain software frameworks, and the way things are kept in sync on both ends of the relationship, might inspire slightly different ways to notate it.
  + The framework-dependence seems introduced too early, perhaps. Or too suddenly. The descriptions in this section seem to be about how I planned to implement it. So the notation may be too implementation dependent. / Perhaps introducing several variations of notations may make sense and make things a bit more open.
  + / What happened to unidirectional relationships between objects? Those do not seem to be explored. Perhaps there are surprising subtleties in it, that might be explored?
* Bidirectional Relationship Synchronization:
  + It seems to be mentioned textually, that relationship synchronization's workings might be displayed as part of the system interface. No image is there (unfortunately)? The system interface at this point in the documentation might not be explained yet, hmm… I also often opt to not refer to other chapters, but I might make an exception here.
* Bidirectional Relationships Between Objects:
  + / I do not see class relationships in the picture. I am not sure if that leaves in doubt about the link between class relationships and object relationships. Oh, in "Bidirectional Relationship Synchronization" I seemed to have a similar issue. I also doubt that the image would become clearer with classes in it. The issue in the other section seemed different, because I used class names in the explanations. But maybe I could show images with classes after having shown an image with only objects, to attempt to make clear what the correlation is. > The next section ("Bidirectional Relationship Synchronization") has I think nice pictures showing that correlation. So I think this is fine.

### 2020-06-15 Done Scoping/Changing Relationships Sections

* I feel there might be details of the Relationship chapter that may be postponed to Circle Construct Drafts or Circle Broader View or something.
* I feel that the relationships chapter could introduce the notation, maybe descriptions of the concept, what it is about, what it is for. Maybe how it could be applied. Something about a relationship being bidirectional and the the two ends of the relationship can be kept in sync.
* The sub-topics and what might be done with them:
  + Relationships Between Classes
    - Still might merge conceptual description and diagram demonstration together.
  + Bidirectional Relationships Between Objects
    - May have a valid reason to be there.
    - It seems, that explaining it with pictures with explicit notation may help explain the concept more precisely. And 'might be's may loosen things up a bit.
    - Merged the diagram notation and conceptual descriptions together a bit.
  + Bidirectional Relationship Synchronization
    - To support the notation for a synchronized bidirectional relationship, it would help that the concept of a synchronized bidirectional relationship is described. I may want to define it.
    - But this section does seem to go into implementation details, that you might not need in a description of the notation. Perhaps in a description of a run-time, a framework, but I think a conceptual explanation might be more appropriate.
    - It seems most (difficult?) text is about implementation details that I might opt to move out of this notation description.
    - Conceptual description and diagram notation have become one text with pictures.
    - I might want to change the order so it starts with a simpler picture, where the relationship is indeed synchronized at first and then try and cover what would happen if it was not. > I feel that it reads alright currently and the order change might not be needed. I did remove some redundant sentences for brevity.
* I think I do have a plan.
  + Moving around texts as described above per sub-topic.
  + Keep in the description of class-relational design ideas.
  + Reformulate what remains in more open language.
  + Introduce the notation for relational rings + explicitly showing the two 'counterparts' of the relationship.

### 2020-06-15 Done Reformulating Relationships Chapter

* The Relationships chapter is larger than the Classes chapter.
* I feel uneasy going through the text marking trigger words in red. It is a lot of text, and it sort of feels like wasted energy sometimes.
* For subjects to postpone I might not value trigger word marking as much as texts I might use at an earlier time.
* I had an uneasy feeling like that when I went through the Classes chapter. Eventually I feel I have boiled the Classes down to something more close to core of the subject and postponed half of the content to Circle Broader View.
* I think I might do something similar with the Relationships chapter.
* I saw implementation details in the text. I saw some concepts, that either might be out of scope or maybe have a place in the System Objects chapter.
* I do not feel that rewording things is wasted energy as a whole.
* But I want to think that if I feel uneasy doing something, there might be a reason why.
* The reason why might be here that the topics I might not really want to cover there.
* Maybe I could just read and go by the topics to evaluate whether I feel they are in scope or whether they might be moved.
* I used to use my intuition like that. I used to trust it. For some reason I seem to have forgotten to trust my intuition.
* (I think I should rename relation to relationships, since it seems a more common thing to call it.)
* Sub-topics and what might be done with them:
  + Referrers
    - Perhaps move to System Objects or otherwise to Circle Construct Drafts or maybe Circle Broader View?
  + Class Referrers
    - Probably the same as above.
  + Relationships between Classes:
    - Not sure what this adds. Seems sort of a bridge towards the 'Relationships between Objects' section.
    - I now moved it close to the top as an extended introductory text, because it might shed light clarifying the main point of relationships.
  + Related Classes
    - Keep it there? Seems a general outline of what unidirectional relationships are, which might be nothing new to introduce in the documentation, but seems an interlude to the other relationship topics.
    - Under "Related Classes\Diagram Notation" I tend to explain details of notational choices, that might not be that relevant to the general point. If explained at all, they might be put elsewhere. Basic diagram elements I what I deemed a place for talking about how relationships between symbols might be implied by various rules.
  + Bidirectional relationships
    - Describes some points of view from a software design perspective I guess.
    - Then goes into that Relationships sort of are bidirectional relationships, which maybe is not a given, so maybe I might not say it that way and may reformulate this.
    - The Diagram Notation part would start to go into that it is not quite solved to apply both *automatic containment* versus *ubiquitous bidirectional relationships* at the same time. Those two concepts may not be part of Circle Language Spec anymore, so that might be moved to Circle Construct Drafts or Circle Broader View. When the postponed concepts are taken out of the text, what might remain of the content? It may make it easier. Notational things might be in there still then.
    - I wonder where my idea went, for a relational ring around the two lines of the two counterparts of a bidirectional relationship. I may just want to introduce it there, even though I was not extending content currently, I am missing this.
    - Besides the things I might like to change, I sort of like the text as I scrolled through it.
    - Section "Counterpart out of Sight"
      * Implicit notations seem to cause jeopardy for ambiguity.
      * In my aim to express multiplicity that is out of sight, combined with my reverting to explicit relationship counterpart notation ("relational ring" notation), the notation looks like what I might have in mind for the notation for 'optional'.
      * I propose to leave this 'problem' unsolved: move the 'Counterpart of of Sight' to Circle Construct Drafts. The argument "something out of sight' is out of sight" seems to make sense enough for this 'problem' not to be considered a problem at all.
      * In another way of viewing it, a proposed notation looks like it 'lies' about the containment structure.
    - Section "No Reuse of Merged Imaginary References"
      * Only seems relevant when automatic containment is relevant and when the sort of deprecated double dashed ring notation would have been relevant. Because those 2 topics would not be considered relevant right now, the topic could move away from Circle Language Spec and into Circle Construct Drafts perhaps. Not even Circle Broader View?
  + Dual and Unary
    - Much of this might be moved or removed, because in essence this might have been covered under the section of Relationships. The issues raised there might be software design issues, perhaps not relevant to the notation. The Relationships section may be renamed to Bidirectional Relationships maybe.
    - I might replace these terms with bidirectional and unidirectional. I think I was searching for shorter clear terms, but I sort of decided to try and use more common IT terminology.
    - I might want to not push the bidirectional relationships too much to the foreground. I may want to make it more neutral on the subject. That should be in line with trying to use more open wording.
    - The Referrers concept seems to be mentioned there again. That might be moved to the System Objects chapter, or to Circle Broader View or Circle Construct Drafts.
  + Referrers Versus Related Objects
    - This might be moved away from Circle Language Spec, since the Referrers topic is too.
  + Relationship to a Pointer
    - According to chapter subdivision ideas I have not, this might be moved to the Pointers chapter.
  + Relationship Direction
    - May tap into a problem that comes with automatic containment, a concept I believe I decided to move away from the Circle Language Spec. So this sub-topic might too.
* Model-based / class-relational design:
  + I am guessing that after the content is changed like that, not that much notation would be introduced, but more an idea of thinking in terms of relationships between classes would be sort of pushed there. That idea might not serve the notation all that much. It may serve an idea I have for aspect oriented programming that might not become part of the Circle Language Spec. I have a question for myself: Would I leave out the topic of relationship-oriented software design, and just strip it to a notation introduction? The alternative seems to be to explain a way you might organize and model code, that to me seems an interesting topic, but does that have much to do with the Circle language notation? I think maybe this is one of those moments where you move a text you like out of the works, to serve the whole and the goal better? Not sure.
  + I think maybe I could get away with apparently squeezing in 'ideals' about software structuring while introducing the relationships notation? Or is that sleezy somehow? I think I would say nothing wrong and the notation is there to support that (common?) software design choice, so why not raise some points about it? They could shed light on why the notation exist in the first place.
* I might already be moving out about half of the (complicated?) text or so. I would consider moving the texts around first, before reformulating in more open wording, just so save some time.
* I think I pushed away the idea of 'what am I doing it all for' and found value in what might become the end result.

### 2020-06-14 Done Reformulating Relationships Chapter

* I might go over the loose ideas in the document and cross out or distribute those.
* Going over the document again to reformulate.
* Pictures:
  + Maybe draw new ones in one go, on paper and then scan it.
  + Sometimes borrow (pieces of) other pictures
  + Or simplify pictures by editing them.
  + Maybe write out more what the pictures mean precisely.
  + I think I notice a tendency to sometimes be helped with a step by step explanation/justification for each symbol or line, when it raises questions for me. But I also have another tendency sometimes, to let the images follow and support the conceptual story, rather than adding text to explain the pictures, the pictures would be there to explain the conceptual text. I do not seem to have a rule there.

### 2020-06-14 Done Checklist Reformulating Relationships Chapter

* Marking standard trigger words in red, with search and replace, possibly marking questionable trigger words with a star (\*) too.
* / Manually marking trigger words in red:
  + Might not give high priority to marking trigger words manually anymore.
* / Possibly manually mark questionable trigger words with a star (\*) too.
* / Finding replacement words, possibly using online thesaurus.
* I have difficulty not wanting to change the story, while the aim is to just reword it, and I seem to experience an impossibility not absorbing the story. My conundrum might be solved, by adding indicators that everything is just a suggestion, so that no idea would disrupt anything, even when I may not entirely agree with it myself.

### 2020-05-25 Done Brainstorm Content Changes for Relationships

I question the way the Relationships topic is approached.

I think it is an interesting idea. But it seems presented in a way that all relationships are supposed to be bidirectional. In one of my prototype apps for a programming environment this idea was sort of centric. But now that the notation seems more important to me trying to push constructs that might not be that widely applicable, this seems to disturb the story.

It seems to be more about an idea for ubiquitous bidirectional relationships. It seems that Relationships in my view back then was synonymous for bi-directional relationships and my idea was you should make most relationships bi-directional. But the problem with, that I have with that now, is that doing so sort of removes hierarchy: everything seems to land on the same level of the containment structure, if every relationship is mutual. I had some ideas how to then solve that again, with relationship direction. I think I still like that idea. But now I have a construct proposal, that seems to be pushed too hard to the foreground, that seems something you should be able to choose, not something you should be pushed down your throat, excuse the sarcasm. And unless I finish it, it seems to hinder the notation, that I find more important than having all relationships be bi-directional.

Maybe I might demote the Relationships construct to Circle Construct Drafts entirely. Maybe. It seems about what you could do with these unfinished 'Creator' systems I once programmed. But what you could do with Creator had to do with aspects, relationships, libraries of aspects, that tap into a way to describe relationships. Possibly all those concepts will be put aside. Maybe to pick up at one point later, but not so much part of the Circle notation. I do think that in the Circle projects in the past, I had figured out that the Creator concepts are probably integrable with Circle, though the precise notation was not worked out to the last detail. I remember now, that I was quite keen to try and make sure the Circle and Creator ideas were to be combined into one. Later on I think I had it clearer in my head where these two ideas met, and how they could go together.

Maybe describing it more 'openly' would just make it usable as a proposed construct notation. Right now the aim is not so much that everything will work as a run-time or that automatic determination of containment structure is also let go of, which might simplify things. I just might not want to propose that this is a required thing.

I think I remember that some other chapters do tap into it. Maybe it is nicer to just keep it separated into this topic on its own, so not to make other notation explanations dependent on it.

### 2020-06-11 Done Reformulating Classes Chapter

* I made text changes, which may have resulted in more trigger words again. Maybe mark those and solve those.
  + Hoping to do the last time of marking and reformulating trigger words.
* I am not sure this open language stuff made it much clearer. It looks to me like using more words than necessary, which may make reading it harder. But that might be specifically how I would see it. I may just be tired. Often.

### 2020-06-11 Done Content Changes for Classes Chapter

* Object-commands / command-objects exchangeability doubts.
* It seems to be used as the main idea, this exchangeability. But that seems demotable to Circle Broader View perhaps or maybe even Circle Construct Drafts.
* What might be left of the Classes story if the base of it would be deprecated? Almost a rewrite? Not sure.
* What if I move the exchangeability issue to Circle Broader View or Circle Construct Drafts? Will not much text be left?
* / Is there a version in the history that does not make the Object-commands / command-objects exchangeability issue centric? Could that text be used?
* Some of the text seems to address the design time = run time issues. That issue might also not be in scope anymore.
* I feel the same sentence is repeated a lot in just slightly different formulations.
* Class Reference seems a bit of an ambiguous term. You might reference a class with an object line, but here I mean when an object points out its class with a class line. Class redirection? Class pointer? I think I would rather make pointer and reference synonymous, so maybe not class pointer.
* The diagram explanation of Commands and Classes Loosely Coupled would be merged into the conceptual explanation, if I follow the editing ideas I had. But the issue might be moved from Circle Language Spec to Circle Broader View at some point, so perhaps this has less priority. Not sure. I could go either way.

### 2020-06-11 Done Reformulating Classes Chapter

* Marking trigger words in red.
* Possibly mark questionable trigger words with a star (\*) too.
* Finding replacement words, possibly using online thesaurus.
  + I seem less motivated, because I question the validity of the idea, so this action may seem futile.
  + Perhaps I might appreciate the idea for what it is. Maybe not part of this Circle Language Spec, but a Circle Broader View idea. So even though I might not use it in Circle Language Spec, I might still appreciate it as an idea outside of that.
  + I get anxiety if something seems unuseful to me. It's a thing.
  + I am done with this, but I feel uneasy about the ideas.
* I might go over the loose ideas in the document and cross out or distribute those.
* Going over the document again to reformulate.
  + I also have second thoughts about the order of subjects.
  + I am not well able to follow text right now, it appears, so this might be for later.
  + Maybe not literally follow the words, maybe shift around topics and draw pictures, to make it better.
* Pictures:
  + Maybe draw new ones in one go, on paper and then scan it.
  + Sometimes borrow (pieces of) other pictures
  + Or simplify pictures by editing them.
  + Maybe write out more what the pictures mean precisely.
    - Now that I do that, some fragments seem duplicates. 'Using Dashed Shapes' now seems a repetition of things said elsewhere.
* I have difficulty not wanting to change the story, while the aim is to just reword it, and I seem to experience an impossibility not absorbing the story. My conundrum might be solved, by adding indicators that everything is just a suggestion, so that no idea would disrupt anything, even when I may not entirely agree with it myself.

### 2020-06-11 Done Content Changes for Objects Chapter

* Object Reference section may start with a more basic image of the concept, / along side the more composite picture it might display now.
* I might be explaining multiplicity as a concept in quite some detail. Should it be more about the notation?
* Outtakes:
  + / From 'Multiplicity' (might use this text in the future again): There might be different words that can be used to express the concepts of single and multiple.

### 2020-06-07 Done Reformulating Objects Chapter

* Marking trigger words in red.
* Possibly mark questionable trigger words with a star (\*) too.
* Finding replacement words, possibly using online thesaurus.
  + Changes to the 'Multiplicity' topic went a bit overboard. Trying to revert it a bit.
* I might go over the loose ideas in the document and cross out or distribute those.
* Going over the document again to reformulate.
  + Would I replace 'related object' with just 'object' in certain cases for simplicity?
* Pictures:
  + Maybe draw new ones in one go, on paper and then scan it.
  + Sometimes borrow (pieces of) other pictures
  + Or simplify pictures by editing them.
* I have difficulty not wanting to change the story, while the aim is to just reword it, and I seem to experience an impossibility not absorbing the story. My conundrum might be solved, by adding indicators that everything is just a suggestion, so that no idea would disrupt anything, even when I may not entirely agree with it myself.

### 2020-06-07 Done Content Changes for Objects Chapter

* Maybe attributes can be introduced as perhaps not really a specific part of the Circle language. It is more of a concept with an explanation of how it might fit into the Circle programming language.
* 'Target Objects' kind of puts aside a different interpretation of object references, where all symbols are references, even 'final target' symbols.
* Attributes section:
  + Enumeration of Attributes: It has a text that might apply in an application where you might query for reflective data, perhaps for aspect oriented purposes? The issue might be a bit out of place there. Maybe elsewhere? In System Objects? Or otherwise in Circle Construct Drafts somewhere?
* I might move Target Objects to Pointers now. I am pretty sure I want to give it a separate place and might not want to reformulate parts of it now?
  + > Yes, I think I would do that.
* 'Related Objects': The picture has the labels A and B, which do not seem to serve a function at that particular spot.
* Related Objects section:
  + The term sub-object seems to change meaning from time to time.
  + At one point it means not the external references. At another it includes the external references. It might be more about the notation, rather than the terminology. I think I might tend to make terminology leading. I seems to explain synonyms of things in separate headings. I have doubs about the usefulness of that. Maybe thinking more of the concepts rather than the terms, might clean up the text? My goal is not to tech terminology or something, or make one up and then teach that. My goal might be to convey an idea. The term 'sub-objects' might not be a term often heard in IT. I think I just use it, because it makes the explanations work better. Maybe, since it does not seem a well adopted, perhaps therefor not well defined, term in IT, I might mention it briefly for the flow of the story, then maybe adopt common IT terminology or at least something I seem to assign a clear, non-ambiguous definition. It might be a subtle effect in the text, this ambiguity around the term 'sub-object', but it might clean things up if I do something with it. But how?
  + Maybe by using sub-object only for non-object-references, then letting go of the term.
  + The term 'object' may simply be used in a certain set of occurrences of 'sub-object' within the Objects chapter.
  + I think the argument 'use common IT terms' that I formulated in the project outline might apply here and prevent possible confusion.
  + Child object and sub-object might be synonyms. Child object also might be an object reference.
  + Object references are a bit amgibuous here. The notation suggests an external link. But other interpretations of pointer suggest otherwise (denoted in the Pointers chapter).
  + Maybe it should not be much of a problem to call it sub-objects some times. It might be a problem in the Relationship Objects section specifically, when you try to assign a (too?) specific meaning to it.
  + Does the section 'Related Objects' serve a purpose at all? Maybe you can drop the term 'related object' arbitrarily and it does not necessarily have to be defined, because it already makes sense on its own? The Related Objects section suggests something new and distinct is introduced, but it is really just the introduction of a term, and the sum-up of what has been previously said. Maybe this confusing effect can be thought of as rudimentary from when the documentation was more 'definition based', like a big glossary. That each term 'had to' have its own article. That might not be how I want to end up describing things.
  + I do like the picture. Maybe I can just drop that picture under object reference, if it seems to clarify things. Maybe I could just suggest the synonym there too, but briefly.
  + I question whether to show the picture at all and whether to define the term related objects at all.

### 2020-06-01 Done Reformulating Basic Diagram Elements

* Marking trigger words in red.
* Possibly mark questionable trigger words with a star (\*) too.
* Finding replacement words, possibly using online thesaurus.
* I might go over the loose ideas in the document and cross out or distribute those.
* Going over the document again to reformulate.
* I have difficulty not wanting to change the story, while the aim is to just reword it, and I seem to experience an impossibility not absorbing the story. My conundrum might be solved, by adding indicators that everything is just a suggestion, so that no idea disrupts anything, when I may not entirely agree with it myself.

### 2020-06-01 Done Content Changes for Basic Diagram Elements

* Maybe replace the word 'globality' by 'module'. I seem to keep running into it.
* In the access marks sections, maybe add a picture of how it would look if it is connected to a symbol. That might not be that clear. You might not see the context there. You might just see a sort of kite like drawing.
* Some content changes:
  + Is it nice to present the reasons for the shapes, or is it just superfluous text?
  + I do not know. The back story may show creativity, which might be nice. It may inspire the imagination of others. But more reasoning may be overwhelming so I am torn between arguments here. More text might not necessarily be bad. I like describing how I came to these choices, sort of artistic choices.
  + It might give it feeling. More humanization.
  + I think the dashed square with the big red cross throught it, might be a bit harshly expressed. Maybe the *curl* with the solid square might also be left out. There might not be much reason to emphasize things so much.
  + I still have doubts about explaining the reason behind the shapes. It may just obscure things. It may be text for 'Circle Broader View' instead.
* Maybe the object reference notation (the eye to another symbol) deserves specific mentioning. It is there under' Lines Pointing Outwards', but there is no specific highlighting of this notational issue. I feel it was sort of the aha moment that started my ideas about this possible notation flowing, so maybe nice to articulate. (There seem to be 2 interpretations of object reference. The eye notation may have one symbol be the object and the other be an object reference. But in another interpretation all the symbols are mere object references, and the lines just indicate correspondence of the aspect, that both symbols point to the same object.)
* 2x almost the same picture under 'Access Symbol Placement'.

### 2020-05-31 Done Reformulating Circle Language Spec Overview

* Committing reformulations of tone of Circle Language Spec Overview.
* Maybe a 2nd phase of editing aimed at the content, not so much about the tone.
* Then there is Loose Ideas at the end. Do I want to put it in a separate document? Do I want to evaluate them one by one to see if they are relevant or might be moved or something?
  + Comparisons with other systems might be better off in Circle Broader View. I feel it is interesting, that I looked around at other systems for inspiration or comparison, but it might not be that interesting to share what my opinion about it was. But topic-wise I think is might not belong in Circle Language Spec, but in Circle Broader View.
* Images in the overview page?
  + For Object Order: Not sure about the notation yet, so not sure what picture might be suitable.
  + For Ponters: Current picture lies a little. This may be how you might express it with abused/abstracted notation, but this might not be a pointer in Circle, if I look at it from a puritan perspective. It might be supposed to point 'inward'? I am not even sure.

### 2020-05-28 Done Content Changes for Circle Language Spec Overview

* I wonder how useful it is to mention what has been programmed or not. I would like to consider what the reasons of that were and if they still apply. I think I applied that as a general rule for these 'overview' or 'index' or 'contents' documents. The Circle Language Spec used to be part of a bigger whole that documented more pieces of software. Each contents page would mention what had been programmed and how far it was finished up. Now that the Circle Language Spec is intended to be more isolated from the rest, this rule might not apply anymore.
* Also the % state of finished up. Not sure if it necessarily adds much. Even when I say it is mostly finished, there are still loose ideas, and still adaptations I might want to make. And most topics do have some sort of description. Just some near the end are less finished up. I think I mention half-way the overview that 'now starts the ones that are less finished'.
* I think maybe the conclusion should be: just remove them.
* Outtake System Objects section:
  + Most of the system objects are part of the *code base*.
* Outtakes Commands section:
  + This seems a more general term for something you can execute on a computer.
* Black Boxes
  + The maybe alien nature of how black boxes took shape (the ubiquitous friend notation), might have made its story in this Overview long. But that is the design it is right now. No redesign would take place yet, is the plan. So can I shorten it? Do I admit, that maybe this notation will not be as prominent anymore at a later stage. Does it harm stability of the text if I say that? Or does it just humanize it?
  + Outtake:
    - The idea for the future is that *user* access control and *black boxing* access control will become a single concept of ‘what has access to what’, but that will not immediately be done at first.
* Interfaces outtakes:
  + Another concept, yet to be worked out, is how commands of an object are grouped by site: an object can have commands, defined on different sites, and the commands should be grouped by site in order to judge how reliable the commands are *(grouped by source)*.
  + At some point the idea was, to also cover all the different *uses* of interfaces. But the different uses of interfaces are adequately described by other literature. The focus will lie on explaining the raw concept of interfaces and their presence in the new computer language.
* Type Control outtakes:
  + Type control seems quite present in programming languages, almost as if programming languages need it. But It might be 'easier' to construct a programming language with no type control in it, than to program type control into it. You might say, that when a programming language supports classes and interfaces, then the programming language might need additional functionality. But in reality, all the possibilities are already present.
  + At first there is no type control at all. Anything can reference anything and this results in all sorts of possibilities, that things will go wrong in a program. Type control only enforces restrictions. When type controls is implemented inside the new computer language, it is like nothing extra is offered, but only the ability to impose more restrictions to the possiblities already offered.
* Brainstorm:
  + I am correcting something that does not naturally come to me somehow. I do not seem to have an intuition for it. This seems to make it harder. I am more used to evaluating whether texts are clear, not whether they are stated in a 'loose enough' fashion. I am not sure if I can do this…
  + Maybe it is unnecessary. I am not sure. I would like it not to be necessary, because it does not seem I can do it.
  + It seems useful, but I do not know if I can do it.
  + What can do? What am I better at?
  + Systemic tricks? Mark trigger words in red or something?
  + Try to build up a dictionary of standard trigger words, I might usually always check for?
  + I wonder whether searching on all the trigger words is useful. They seem too many. Maybe just an awareness of them is good enough to pick them out by hand.

### 2020-05-31 Done Wiggle Word Difficulty

* A less pleasant experience yesterday with the trigger word marking.
* Still the problem that formulations not necessarily come naturally to me.
* I liked describing a system of trigger words.
* I also kind of liked picking them out, marking them red.
* But trying to find alternate formulations for them, was tricky.
* One thing that seems to bother me, is that I do not see that a rule is too fixated, until I have formulated it in a more open sense and sensed that it is feels closer to the truth. But I do not seem to read a resolute sentence and realize it could use wiggle room. I mostly just see a valid rule, not one too harshly stated.
* Another aspect to consider is that formulating new sentences quite automatically I insert the wiggle words, probably because I did train myself a bit to use them more. But seeing the problem in existing texts is harder somehow.
* I would like that after marking trigger words in red, it would be a matter of arbitrarily inserting wiggle words, removing adjectives or changing definite verbs to more polite ones.
* But I end up scratching my head trying to find a way to reformulate something. Difficulty to find the 'nicest' alternate word, or sometimes even to realize something is even wrong with the trigger word I marked in red.
* Sometimes my brain says: nothing is wrong with trigger word, but when I replaced it with difficulty, I realize the trigger word did seemed to be 'wrong' after all, and realize that things are not necessarily that black and white.
* But my mind, that is reading and understanding these existing sentences does not sense anything wrong with them really, so has difficulty finding a proper way to change it, or even a reason. It's it like my mind resists changing the sentence, because at first I fail to see why.
* An additional problem seems to be that I have very little energy. This computer hobby usually takes little energy. I am built that way. But certain things about it do take energy. This task seems to tap into something my mind does not automatically do naturally or easily. Not sure why. It just takes too much effort.
* I kept re-evaluating the sentence as rewritten: whether it 'walked' well, whether it still made sense, whether the wording is varied enough in the document as a whole. That might be a lot of language processing for me right now.
* So after this reflection I am left with a question: How to proceed?
* Alternative: I am not sure, but could I proceed the same way, and just hope I will have more energy now?
* Alternative: It may need focus to try and insert wiggle words sort of 'robotically', maybe not even looking at the meaning of the sentences.
* > One alternative seems to need energy. The other altenative seems to need focus. Neither is aplenty.
* Alternative: Build up more wiggle word dictionary. May be hard, because thinking of alternate words seems energy draining.
* Limitation: That the document I work on now goes from one topic to the next, might not help me.
* Limitation: Switching between the chapter content and the overview content is probably difficult switching for me.
* Limitation: Doing it in a way to avoid where my problems are, requires focus, which seems to be a problem in itself.
* Limitation: I think maybe language processing is problematic for me right now.
* Alternative: Using the existing wiggle word dictionary, inserting wiggle words robotically. No careful evaluation.
* I do estimate that the above should be the kind of thing I used to be good at. But focus was the problem yesterday. Could not hold myself to a plan anymore yesterday, I guess. Maybe I can today.
* Helper: Putting up a Post It: "Insert wiggle words robotically", for focus.
* Helper: Keeping the wiggle room dictionary closed? (prevent switching)
* Helper: Use thesaurus.com for words.
* One problem I see with 'insert wiggle words robotically', is that inserting wiggle words was not my only goal editing this documentation. Maybe I can make wiggle words one phase and reformulation a second phase, and not do it in one blow, because it seems to make it more difficult. I just do not want to forget that I also want to make the text better content-wise, not just change the tone.
* While doing it:
  + Accept limited word variation at first?
  + Yes, when I cannot find a word, it is hard on me.
  + Sometimes a text might be deprecated. Saves some reformulation, but may personally mean more information processing > more easily tired.

### 2019-08-05 Done New Intro

* 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 might actually be 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.

### 2020-05-28 Done New Intro

* I have the idea, that I might copy and paste more content and search and replace certain basic trigger words like 'is' by 'could' and 'might' for instance. And also visually scan the text for trigger words, like superlatives, words that see express value or devalue.
* I would like to replace the word 'Procedure' by 'Command', also in the pictures.
* When I change notations, the call lines in the Circle Language Spec Introduction might become dashed.

### 2020-05-28 Done Brainstorm New Intro: Word Stress

I started writing over sentences from the 'old' Symbol Language documentation. I kept finding words that might seem opinionated. I feel things are slow. I liked the older text, except for some parts. Now it is like I want to reconsider each word from it, and replace it by something milder. But it seems so difficult to be so wary.

So I have the idea, that I might copy and paste more content and search and replace certain basic trigger words like 'is' by 'could' and 'might' for instance. And also visually scan the text for trigger words, like superlatives, words that see express value or devalue.

Copying text, then adapting it to be more gentle, might also not be easy. I might read over things quickly and not notice things. But carefully reconsidering each word, feels a bit cramped. It also feels less appreciative of the previous text, which I thought I liked.

It just feels I am taking these communication style rules a bit too far.

I feel that if I am not over-alert about the written text, I feel I might not recognize that a statement could be too black and white or opinionated, or an exaggerated view. So I seem to be inducing an alertness that I cannot maintain. Also I question how necessary all of this strain is. I just don't really know.

### 2020-05-27 Done Brainstorm Priorities

* I am not sure how to approach things right now.
* / I could also try and pick a design issue from the list 'Language Design / Content Changes' above.
* / I feel somehow I had more plans, but I forgot about them.
* / Maybe there are few main problems I would like to highlight.
* / Somehow I tend to want to just start at the top and start reformulating.
* I guess I am not sure where to start.
* And I am also still a little surprised that I am at this stage. I feel I must have forgotten something that must be left to do, before I can do this.
* / I think maybe heavy redesign is off the table for now, because I estimate that would be too intense for now.
* I also have in mind marking work as postponed for 'Circle Broader View' and 'Circle Construct Drafts', cleaning up the brainstorms that covert those + Circle Language Spec. That way I might create overview over the TODO notes.
* The feeling I forgot something, feels like it could be in the texts below, but the texts below seem to be too rough and unstructured to give me that insight and overview. If it were more structured and overviewable, and done stuff marked as done, postponed stuff marked as such and split into coherent chunks. I think I would like that.
* So structure the TODO and Postponed things about Circle Language Spec Revamp notes, might be what I want to do first.

### 2019-08-05 Done Writing Style Mixed Issues

* / 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. > Scoped things more sharply instead, so more is finished up better percentually.
* / 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. 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.

### 2019-08-05 Done Brainstorm Scope

* Circle Broader View: Maybe the design of the programming language should lose some ambition and express that only as dreams.
* 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 the main 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 wanting 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.
* Circle Construct Drafts: 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.

### 2020-05-24 Done Notes Planning

* Maybe merge more article together in the first chapters.
* Maybe list out main language design issues.
* / Some work in Future Sub-Projects may already be done.
  + > Applies to the TODO item 'Merge conceptual explanation with diagram explanation'. Those 2 things may now be put into the same article, but the explanations are still often 2 explanations instead of one. So I do not think I can call it done.

### 2020-05-24 Done Merge More Articles

* More articles from the first chapters might be merged together.
* Structure changes:
  + x Making back ground black?
  + x Changing main headers?
  + / Removing the footers?
  + / Removing 'see also' references?
  + x Convert to docx.
  + Merging or redistributing articles?
* Chapters
  + Objects
  + Classes
  + Relationships
  + System Objects
  + Commands

### 2020-05-27 Done Merging More Commands Articles Together

* Redirection and recursion seem to have some overlap conceptually.
* Maybe 05. Comparison to CPU-Like Calls belongs in 04. Creation Behavior Of Commands.
* 01. Commands Main Concepts > Larger > Keep separate
* 02. Start & Stop > Larger > Keep separate
* 03. Parent Controls Its Sub-Executions > Medium sized > Misc-ish
* 04. Creation Behavior Of Commands > Larger
* 05. Comparison to CPU-Like Calls > Medium sized > Might group into Creation Behavior Of Commands
* 06. Command Redirection > Medium sized
* 07. Recursion > Medium sized / smallish > Might group into Command Redirection.
* 08. Command Referrers > Smaller > Misc-ish
* 09. Commands Example Diagrams > Larger > Keep separate
* 10. Commands Ideas > Larger > Keep separate
* 11. This (unfinished) > Smaller > Misc-ish

### 2020-05-26 Done Merging More System Objects Articles Together

* Last time it seems I only merged together a few large groups of articles into their main topic.
* Right now I feel maybe the first 5 could be merged into one.
* Along with some main topics, I feel there are loose topics. Maybe those loose topics could be grouped together with something.
* I think Connectors and Connections is something I might want to combine into one article.
* Assignment I would like to keep separate for now.
* System Command Call Notations too. It's kind of large and honestly a bit questionable, so I would like to keep that separate.
* I am thinking the remaining 4 relatively small articles, might be combined into a System Objects Misc article.
* 01. System Objects > Larger > Keep separate
* 02. Assignment > Medium sized > Keep separate
* 03. System Command Call Notations > Larger > Keep separate
* 04. Connectors > Smaller > Combined into Connectors & Connections.
* 05. Connections > Medium sized > Combined into Connectors & Connections.
* 06. System Command Calls by User > Small > Put into single article System Objects Misc Issues.
* 07. Objects Floating Around > Small > Put into single article System Objects Misc Issues.
* 08. System Command Extension > Small > Put into single article System Objects Misc Issues.
* 09. Parameters For Objects > Medium sized > Put into single article System Objects Misc Issues.
* 10. List Concept > Medium sized > Keep separate
* 11. System Objects Ideas > Larger > Keep separate

### 2020-05-25 Done Merging More Relationships Articles Together

I feel merging the articles together is not something I necessarily want to do right now. I am not sure why those two things are connected to each other in my view.

I guess I feel I might want to polish up the way I explain things, before I put effort into merging the articles together at all? Are the articles not particularly small, like some of the articles I merged together before?

Some articles are smallish or sort of medium sized. I think I could just make it one article. That may make it more manageable the way I would like. I do not see my not entirely agreeing with the concept anymore as standing in the way of that.

Still want to change the fonts.

### 2020-05-16 Done Formatting and Article Merging and Distributing

* / Maybe put Type Control out of scope? > Deemed less important to consider right now.
* So, what's next?
* I think nothing to reorganize anymore about the file structure of Circle Language Spec.
* Maybe start with the texts?
* / Option: Convert to docx. > Too many docs.
* Option: If there are still Text Code topics, they might be moved to Circle Construct Drafts.
* Structure changes:
  + Making back ground black?
  + Changing main headers?
  + Removing the footers?
  + Removing 'see also' references?
  + Convert to docx.
  + Merging or redistributing articles?
  + Putting each conceptual and diagram explanation together in an article.
* Chapters:
  + Circle Language Spec Overview
  + Basic Diagram Elements
  + Objects
  + Classes
  + Relationships
  + System Objects
  + Commands
  + Parameters
  + Globality
  + Execution Control
  + Black Boxing
  + Interfaces
  + Events
  + Inheritance
  + > The next are 26 documents total. I might do formatting for all of them in one go. None of the folders seem to have large sets of articles anymore.
  + Type Control
  + Object Resolution
  + Conditions
  + Object Order
  + Uncategorized
  + Other Diagram Topics
  + Summaries of Other Languages
  + Previous Versions

### 2020-05-24 Done Merging and Redistributing Remaining Articles

Maybe going into detail about this is not part of the editing phase I am putting these articles through.

This is what the Circle Language Spec Overview calls the 'unfinished' part. So for that reason it seems less straight forward.

It is also harder to subdivide, because it is multiple different issues, unlike previous tasks I did, which were just about 1 main topic.

The question I have for myself is how much is worth doing with regards to these topics.

Some things I'd like to do.

But I might want to be selective about it.

* / Type Safety, Genericity, Explicit
  + Already 1 article. The topic seems to deserve its own spot.
* / 21. Object Resolution
  + Seems all of the articles might be put in one article, because I estimate these are not big issues separately.
  + A lot of the text is preliminary.
  + I do see this as being appropriate as a separate chapter, I think.
  + I do not see much of a reason to put effort into this right now.
  + It's kind of messy and when putting it into one article now, I may lose the overview that the split up into articles gives me now.
* / Conditions
  + These 3 articles could become 1.
  + It seems that if content is still messy, the split up into multiple articles might have value, because it seems to create overview.
  + Just added some comment on other content about Conditions to find elsewhere.
* 23. Object Order
  + Is one article.
  + But not a big one.
  + Makes me wonder if some topics are handy as a separate chapter or better grouped into another main topic.
  + Eventually might move to the Objects chapter, but will keep it in its own folder for now.
* This
  + I think I already had a place in mind for that, to move this article to a different main topic. I think tso Commands, but I think I put that down in the planning doc Circle Language Spec Product List.
  + Circle Language Spec Product List says that it should belong in the Commands chapter.
* Redirection
  + Circle Language Spec Product List does not seem to assign a specific existing Chapter to it. It says 'ref-ness'? Maybe that pointer to pointer situations might all be put in a separate chapter, this might also belong there?
  + It talks about 'target symbols'.
  + There are several target symbol topics in e.g. the chapters Objects, Classes and Interfaces.
  + Maybe that can all go to a chapter about ref-ness, or whatever generic term I might find for it.
  + If it is just about target symbols, maybe the topics are already covered by the previous content.
  + It does seem to be about redirecting e.g. the object aspect and then about tracing to the final target object/class/interface after following the redirections.
  + I do like the idea of a Pointers chapter or something.
  + This content does seem to be covered elsewhere.
  + Perhaps I should just make a folder 'Pointers' and put it there, and a readme just to make a point, that it could just become a separate chapter.
* 30. Misc Diagram Topics
  + I think this is appropriate as a separate chapter.
  + 1. Diagram Expression Introduction > Larger
  + 2. About Programming In General > Larger
  + 3. Diagram Metrics > Larger or will be.
  + 4. Coloring > Smaller
  + 6.0. Example Diagrams Introduction > Small
    - Maybe move to the folder 6.1. Example Diagrams? Maybe call it Readme?
* 7. Summaries of Other Languages
  + Larger
* 32. Previous Versions
  + This folder structure seems off when there is essentially only one version, as one document in it.
  + (There used to be multiple versions in one folder. Those were put out of scope of the Circle Language Spec.)

### 2020-05-23 Done Merging Inheritance Articles Together

Most of them are quite small.

Grouping together main concepts into one article might make sense.

Specialization is mostly small articles, but quite a few of them. But maybe combine those into one article.

The articles that follow do not seem to be many or they are already larger ones I might not want to group together.

* 01. Inheritance Introduction > Smaller
* 02. Inheritance Main Concepts > Smaller
* 03. Class Inheritance > Smaller
* 04. Object Inheritance > Smaller
* 05. List Inheritance > Smaller
* 06. System Inheritance > Medium-sized / smaller
* 07. Interface Inheritance > Smaller
* 07. Specialization > Smaller
* 08. Specialization & Data Replacement > Smaller
* 09. Altering the Member Set > Smaller
* 10. Member Addition > Smaller
* 11. Member Exclusion > Medium-sized / smaller
* 12. Member Inclusion > Smaller
* 14. Detouring Members > Smaller
* 15. Shadowing > Medium-sized / smaller
* 16. Overriding > Medium-sized
* 17. Altering Command Implementations > Smaller
* 18. Command Extension > Smaller
* 19. System Command Extension > Smaller
* 20. System Command Overriding > Smaller
* 21. System Command Shadowing > Smaller
* 22. System Command Extension By Shadowing > Smaller
* 23. System Command Extension By Overriding (Questionable) > Smaller
* 24. System Command Overriding By Extension > Smaller
* 25. Destructive & Non-Destructive Specialization Methods > Smaller
* 26. Misc Inheritance Situations > Medium-sized / smaller > Moved to Inheritance Main Concepts.
* 27. Enforcing & Preventing Specialization > Larger > Would probably keep it separate.
* 28. Deeper Specialization.docx > Larger > Would probably keep it separate.
* 29. Alternate Version Through Inheritance > Smaller
  + > Maybe move this detail elsewhere. Might not be worth an entirely separate article.
* 30.1. Extending System Objects (Older) > Larger
  + > Not sure what the status of this is? It has the word 'Older' in it. Is it deprecated? Covered elsewhere already? I don't know. But at this stage maybe I do not need to know.
* 30.1. Extending System Objects in a Diagram (Older) > Medium-sized / smaller
  + > Would probably merge it with the main article.
* 31. Inheritance Ideas > Medium-sized

### 2020-05-22 Done Merging Events Articles Together

Possibly this is one you want to put into a single document.

Most of them are small.

13. Qualified Event Names should be out of scope, since it is about text code?

The articles before 'System Events' might be put into one article.

I might put System Events in one article.

System Events can almost be demoted to a Construct Draft, but just like System Objects is in there, I might want to keep System Events part of the Circle Language Spec (for now). (The correspondence between System Objects and System Events is that they seem more like part of a run-time or if Circle would function more like a stand-alone language, rather than mostly a diagram notation).

The last 3 articles might become a single 'misc' article. But perhaps some of them have a place in the main article.

* 01. Events Introduction > Smaller
* 02. Prime Event Example, Button Clicked > Smaller
* 03. Events Main Concepts > Smallish
* 04. Event Situations > Smallish
* 05. Explicit Implementation of Event Object > Smaller
* 06. Explicit Interface of Event Object > Smaller
* 07. Event Interface Reference > Smaller
* 08. System Events > Medium sized / smaller
* 09. System Events in Normal Notation > Smaller
* 10. Simplified System Event Notation > Medium sized
* 11. System Event Parameters > Medium sized / smaller
* 12. System Event Design Choices > Smaller
* 13. Qualified Event Names > Smaller <<out of scope>>
* 14. Interaction Events > Smaller
* 15. Black Boxing Events > Medium sized / smaller
* 16. Event Alternatives> Smaller
* 99. Ideas > Smallish

### 2020-05-22 Done Merging Interfaces Articles Together

* 01. Interfaces Introduction > Small
* 02. Interfaces Main Concepts.docx > Medium sized
* 03. Interface Aspect.docx > Small
* 04. System Commands for the Interface Aspect.docx > Medium sized / small
* 05. System Commands for the Interface Merging Aspect.docx > Small
* 06. Interface Aspect in System Interface.docx > Medium sized / small
* 07. Interface Assignment Types.docx > Medium sized
* 08. Interface System Command Calls.docx > Medium size
* 09. Interface Connectors.docx > Small
* 10. Interface Connections.docx > Small
* 11. New Command with Interface Parameter.docx > Small
* 12. Command Interfaces.docx > Small <<use case>>
* 13. Interface Referencing and Redirectioning.docx > Small <<notation demonstration>>
* 14. Member Grouping.docx > Medium sized / small <<edge-case>>
* 15. Relationships Through Interfaces.docx > Medium sized <<notation demonstration>>
* 16. Interface Referrers.docx > Small <<edge-case>>
* 17. Reliability of Interfaces.docx > Medium sized / small <<edge-case>>
* 18. Group By Source.docx > Medium sized / small <<out of scope?>>
* 19. Imaginary Backward Relationships.docx > Small <<out of scope?>>
* 20. Target Interfaces.docx > Medium sized, because of ideas, main text is small. <<main concept>>
* 21. Mutual Commands, Mutual Interfaces (Unfinished).docx > Small <<construct draft>>
* 22. Extensive Classification with Interfaces (Unfinished).docx > Small <<construct draft>>
* 99. Interfaces Ideas.docx > Small > Move to main article.

None of the articles are that big. The size differences are a bit all over the place.

Logically I might group together all Interface Aspects articles.

The Intro and Main Concepts seem logical to group.

I think I'll just start with those and take it from there.

There's something about text code in Interface Assignment Types. That could be moved to Circle Construct Drafts.

Maybe move the Loose Ideas all to a separate article at the end.

I discuss problems with interface reliability in case of automatically established bidirectional relationships. The idea of always having bidirectional relationships between things is something I want to distance myself from in the Circle Language Spec, and perhaps leave as a topic for Circle Construct Drafts. But right now the idea seems embedded into the Circle Language Spec, and without significant modification, it does not seem you can deprecate it yet. I think such redesign of Relationships may take place in the future (to leave out automatic bidirectional relationships). But I think it is only then that I also adapt the text about this edge cases. So the question that worried me is whether it is out of scope or not. The answer for now seems to be: no, it is in scope, but the design of the concept of Relationships may be changed in later efforts.

### 2020-05-20 Done Merging Black Boxes Articles Together

There are quite a few relatively small articles.

But I have doubts how to group them to bigger articles.

I do not think I want to group them together just in one big article.

I might just go for it, grouping the first smaller articles into a bigger one.

* Smaller, integrated into a single article:
  + 01. Black Box Introduction
  + 02. Black Box Main Concepts
  + 03. Public & Private
  + 04. Friend Declaration
  + 05. Notations of Private
  + 06. Friend Declaration in Instances
  + 07. Public & Friend Connections
  + 08. Access Controlling Aspects
  + 09. Object-Bound, Reference-Bound & Access Control
  + 13. Assignment between Friend Members
  + 14. Friend = Wavy Line
* Group these together:
  + 10. Public & Private Connectors > Kinda medium sized
  + 11. Public & Private Connections > Relatively bigger.
  + 12. Public & Private Assignment > Kinda medium sized.
* Putting these together in one article:
  + 20. Black Box Miscellaneous Issues
    - Looks unfinished.
    - Objects Take Over Class Access Control: Might move to article Black Boxes, section Friend Declarations in Instances.
  + 21. Black Box Side-Issues
    - Looks unfinished.
    - Might move these Circle Broader View:
      * Black Boxing and User Access Control
      * Programmers and Users
  + 22. Black Box Details, May Cover Last
    - Unfinished drafts.
    - Might move these Circle Construct Drafts:
      * Level Limitation
      * Globality Level Limitation
      * Downsides to Black Boxes
      * Get For Access and Get For Copy
      * Inaccessible System Aspects
      * Compared to Traditional Black Boxing (goes in too deep on automatic containment)
      * Other Details
    - Special case:
      * Simplified Access Connectors: I think those ideas could be moved to Simplified Access Control Notation, because that is what they are about. Big brainstorm though.
  + 23. Black Box Details, Might Not Cover
    - Same as above.
    - Moving it to Circle Construct Drafts after some considerations.
* 23.3. Black Box Details, May Not Cover, Simplified Access Control Expression in a Diagram
  + Left it for what it is. Am not calling it unfinished. It seems finished. Maybe might change in the future, but it looks like a worked out idea.
* Moved from Circle Language Spec to Circle Construct Drafts:
  + 23.1. Black Box Details, May Not Cover, Simplified Access Control Expression
  + 23.2. Black Box Details, May Not Cover, Simplified Access Control Expression in Text Code

### 2020-05-20 Done Merging Execution Control Articles Together

* I think I want to merge it into the following set of articles:
  + 1.0.1. Introduction to Execution Control
  + 2.0.1. Conditional Execution
  + 3.0.1. Loops
  + 4.0.1. Jumps
  + 5.0.1. Loop-Related Jumps
* Reason: I had almost merged it into one document, but I like how the separate documents seem to nicely categorize the different kinds of execution control. Also: merging it into one document would give me a tendency to use more heading levels and the styling for those is not optimal yet. Not a strong argument maybe, but it does make me think: More heading levels might be 'proof' that a separation in several articles is not a bad thing.
* I just wanted to save some time merging each pair of documents lots of times only to merge them into a larger document again.
* Taking a look at how it looks now.

### 2020-05-20 Done Merging Parameters Articles Together

* / Maybe merge all the Parameters content into one article if it is not too much.
* Relatively small:
  + 02.0. Parameter Passing.docx:
  + 03.0. Joint Display of Access Connectors and Object Relationships.docx
  + Moved their text to the Miscellaneous Parameter Topics article.
* 05.0. Imported Parameter Concepts
  + Not large topics. Maybe Required & Optional is a bit larger. Half of it or so is idea texts, but not bad ones I think. Basically just the version from the previous version of the documentation (Symbol Language.doc).
  + Merging together seems appropriate.
* 10.0. Miscellaneous Parameter Topics
  + Also not large articles.
  + Merging together seems appropriate.
* / I think I want to merge things together in phases, before I decide whether or not I want to turn it into just one bigger article.

### 2020-05-18 Done Merging Commands Articles Together

* I still mean to rename the docs so the 'in a Diagram suffixes are gone.
* Could not see a pattern in what I would want browsing the first few documents.
* Maybe I want to group the first ones into a Main Concepts article.
* I might want to inspect it some more in search of ideas.
* Main Concepts:
  + I suspect I would like to put the 10 documents Commands.docx until Active Clause.docx in one document.
* Hard to group:
  + Start & Stop.docx until 13.1. Changing Inactive to Executable.docx the articles are sort of medium sized and hard to view as a group together, I think.
* Personal feeling: I think the information rate is too high for me now. Lots of topics rush by my eyes at a large pace and my energy level is dropping.
* Even if I do not know how I could group together many of the articles logically, if I can find a few groups, that would already benefit the overviewability of the topic list.
* / These are small:
  + / 15.0. Command References Inside Commands.docx
  + / 16.2. Executables & Executions.docx
  + / 17.1. Procedure.docx
  + / 18. Resolution When Not Allowed For Commands.docx
  + / 22.0. Sub-Commands Not Manually Started.docx
* 19. Parent Controls Its Sub-Executions.docx:
  + Seems like the 3 articles after it are part of it, but not sure.
* Creation behavior:
  + There seem 8 'Creation Behavior' articles. They might not be small, but grouping them to one article might tidy up. Also, it does not seem to me a topic you always want to see in detail. Maybe look at it once and say "Ok, so that's how that could work." Now the topic is quite prominently present in the article list.
* These seem groupable:
  + 35.2. Inactive Command Object Redirection.docx
  + 36.1. Inactive Command Class Redirection.docx
  + 37.2. Executable Command Class Redirection.docx
  + 38.2. Executable Command Object Redirection.docx
* These seem groupable:
  + 39.1. Target Command Object.docx
  + 40.1. Target Command Definition.docx
  + Or maybe put in Command Redirection.
  + Those two have loose ideas in them.
  + I might not want to stop myself from merging articles that have loose ideas. (My argument against it seems to be: It might get messy.)
  + I also might not want to step away from the 'put loose ideas where they belong'.
  + So maybe I want to accept that some loose ideas get spread a little over the content here and there.
  + The idea with the loose ideas was to go through them at some point seeing if any of it is still usable or not and then maybe discard them, or integrate the ideas into the content. Something like that.
* Out of scope? 42.0. Implementation of System Commands.docx
* Example diagrams:
  + Maybe put them all in one article.
  + / Maybe rename and see what they mean.
* Some topics' titles look like main concepts to me. I want to evaluate later if I want to move them to Commands Main Concepts.docx.
  + Maybe if ut were'd for the Start and Stop article in between I would just move more to Commands Main Concepts.
  + Consider merging the content of these with into the Commands Main Concepts.docx:
  + 13. Reading & Writing Parameters.docx
* Merge these two:
  + 17. Command Object Referrers.docx
  + 18. Command Definition Referrers.docx

### 2020-05-18 Done Merging System Objects Articles Together

* Deciding what to do with these, was a bit intense. Especially when I hit a part of the content I have reservations about.
* System Commands:
  + Articles for the various aspects might be merged to one article. Those are like 15 articles or so. Most of them are small. 3 of them are a bit larger.
* System Interfaces:
  + Articles for the various system objects might be put together into one. Not that all of them are small. Just that enough of them are. It seems a bit of a hinder to have to open them one by one. A single article might make browsing the content easier. The amount of articles might be more overwhelming that an article that has some pages.
* Assignment + subtopics
  + May be turned onto 1 article.
* I think the main parts may become more obvious from the article list if I merge together more articles into one.
* System Command Call Notations:
  + 4 smaller, 3 larger
  + Not sure if for the size of them I would merge them together.
  + It is kind of one topic.
* / Misc topics at the end:
  + / Not sure yet if merging is appropriate.

### 2020-05-16 Done Reorganize Circle Language Spec Files

* The bulk of the content is in the Coding Concepts.
* Diagram Expression content:
  + Is now in more than 1 place.
  + "1. Diagram Expression" folder:
    - Describes Basic Diagram Elements.
    - The argument, that the diagram elements must be quickly introduced at the beginning, I am not sure about that anymore. So not sure if this actually must be at the beginning of the documentation anymore.
  + "6. Expression\1. Diagrams"
    - Seems less finished up. I am not sure I would want to put that at the top of the documentation or at least not in this state.
  + "7. Other Topics\9. Old or Boring\About Programming In General.doc"
    - This is old text, that when rewritten I might even want as the introduction text that the Circle Language Spec starts with, or maybe just part of it.
    - Is not really a diagram expression topic specifically.
* / I might want to somehow merge those diagram topics together.
* And put the Coding Concepts sub-folders just in the root.
* In what order will I 'mix' the Coding Concepts and the Diagram Expression topics?
  + Brainstorm: I don't think I want to just start with the Objects chapter. I do like that something starts with the overview. Right now that is the Basic Diagram Elements. Later it may be something else. I think I will keep Basic Diagram Element at the top. Then just the code concepts. Then Other Diagram Expression Topics, just a new folder, a flattened out version of the old.
  + I guess I am just making a flatter list of things and keeping most of the order.
  + 'About Programming In General.doc'
    - Could be put at the top as the introduction.
    - Actually, just reading it over, I think it may be still to rough of a text, that I want to make a bit more neutral, before making it the main thing to start with.
    - It and other bits of texts from the original Symbol Language.doc may compose a new introduction to the Circle language after some reformulating.
    - All in all, I think this content is a but rough for putting at the beginning of the documentation.
* There is a clash in index pages. "Code Concepts.doc" and "Language.doc" may need to become one and cleaned up.
* Moving "Language.doc" content to "Code Concepts.doc" in a controlled fashion.
  + There is no heading 'Object Order'.
* I think I am just going to knead the content.
* The folder subdivision is now much simpler.
* "8.Previous Versions" folder:
  + Has the version of the documentation: Symbol Language.doc.
  + Also contains those Symbol Pictures. I do not want to publish all of them.
  + I do not think I want this document in the published material: D:\Source\JJs Software\Software System Docs\1.1. Circle Language Spec\8. Previous Versions\2004-00-00 XX Symbol Language\Other Files\Coding.txt.
  + Or rather, this has some content in it, that I find questionable.
  + I see two zips with pictures.
  + How do I handle this? In the future I want to go list out all files in the history and remove things from the history that I do not want to publish. Those zips might become one of those files.
  + But when I clean up the current state, will I find all the files back that I want to remove from history? Some have obvious file names. So I guess I will.
  + I question whether I even want to include those example diagrams. Or whether I want to include them in this state.
  + Or maybe I will need them for a better quality referenced image? Nah, I doubt it. That seems too much work, to replace the images in the docs with possibly better images from that zip.
  + It might also not be useful to readers, at least not all of it, because it is just bulky. No one might want to go through them I think.
  + Perhaps a selection of it might be useful.
  + The argument 'Too interesting too throw away'. I might not agree with that anymore.
  + I think most of them are also present somewhere in the documentation.
  + Some of them I want to not have in source control history.
  + Some I might want to keep.
  + If those 2 zips with pictures are the same…
  + I just compared file sizes and they are different. So the zips might not be the same.
  + I could unpack and compare.
  + I think if they are comparable or one is preferred over another, I could remove the other.
  + I could strip one of the copies.
  + Leave that one in the new documentation.
  + So do I remove the one from the 'Previous Version' documentation?
  + For versioning like that, I think it would have made sense to keep the one in the 'Previous Version' documentation.
  + But for control over the content, that I publish, in this case I would really like one copy.
  + Result of diff: 2x Thumbs.db. 4x a file that is not in 'Previous Version', but is in 'Other Diagram Topics\Example Diagrams'
  + Then do I want to remove the one in 'Previous Versions'?
  + I am not sure I want to unzip them first and then commit. Maybe I want to clear it out first after I unzip it, before I commit it. That might make cleaning up later easier.
  + I tried to revert my commit, but it leaves behind files in my working folder.
  + Reason for deleting picture: Don't want it in there.
  + Reason for deleting picture: Is already visible in a document.
  + Reason for deleting picture: It has private info seen right through the paper.
  + Some of them I have a hard time saying goodbye to.
  + I am beginning to think I might just want to remove all of them.
  + Or most of them.
  + The scanned ones probably.
  + The ones certainly used in documentation or otherwise discardable.
  + I think I would remove most of them, and only keep a couple.
  + Oh, I am done.

### 2020-05-14 Done Split Up Circle Docs

* Next step might even be starting to split up the Circle Language Spec itself in 'in scope' and 'out of scope'.
* Doing a git push.
* Renaming git repo Circle Docs to Software System Docs.
* Renaming git repo Project Docs to Planning Docs.
* Moving around folders.
* "Previous Versions\2005-07-28 00 Creator 0.4":
  + For my own overview it might be useful.
  + But for a clear presentation of the Circle Language Spec, it seems to contribute much.
  + It might obscure the whole to include this.
  + An argument for including it, might be to prove it has a development history, but I might not value this that much.
  + Perhaps move it to one of the other parts of the Circle docs than the spec? Broader view or something?
* Branching
* "Language\Integration" and " Language\Libraries"
  + Under the Language folder: might I move those to the Framework part? They are not concept frameworks, but they don't need to be.
* "10. Circle Constructs Drafts\6. Expression\6. More Text Codes":
  + Has the sub-folders "1. Parser" and "2. Text Templates". I do not think those topics belong in Circle Construct Drafts. Maybe in Circle Broader View. But maybe those topics do not belong in Circle Language docs at all. More like framework. The index page includes other sub-topics, that used to be in that folder: such as 'other text codes', 'math' and 'assembly'. So some time in the past those topics were all in one place. I am not sure to exclude all of those topics from Circle Language and then maybe move them to the Framework part of the Software System docs, or whether some of it somehow has a place in broader perspective.
  + They are probably there, because in the main subdivision of Software System, they would fall under 'Code' and that might be what made them end up in the proximity of Circle Language Spec. So it indeed probably does not belong under Circle at all, but it does belong under 'Code' in Software System.
* By now I have:
  + Categorized topics from "Language" over "Circle Language Spec", "Circle Constructs Drafts", "Circle Broader View" and "Framework".
* Checking if I like how topics ended up in one of 3 categories.
* I have deleted files in my commit. I only want to see renames.
* Commit
* Some 'Operating System' topics might belong in a Circle Language topic group.
* Some documents cover multiple topics. I would like to split those documents up.
  + I could just make copies.
  + And then strip out content that does not apply.
  + / To what degree is it useful to rephrase those directory indexes to reflect the contents better? > It might be better to not do that right now.

### 2020-05-10 Done Brainstorm Effort

It may not be within my capacity to do some of the things I plan to do. Thinking critically about word usage and whether I am stating things too strictly… I feel this might not correspond with my energy budget. I made a careful conclusion about that before. Publishing might be more important, than if I might come off too strongly in the text. It feels like this effort is pulling me down more than I might be able to handle. It seems to be weighing on me.

Perhaps I can drop the goal of trying to use a different language style. My next goal was scoping. I seem to still be bargaining with the idea, like: "Maybe I can just reformulate when it *really* seems to harsh." I think maybe that won't work. Because I tend to evaluate the harshness, and this evaluation is 'jittery': sometimes it seems to be 'on', sometimes 'off', sometimes too on tightly, sometimes maybe too weakly. Perhaps I am better off dropping the whole goal of rewording it. I like learning to use language with more air in it, for my personal life for communication to others, or to prevent holding on too tightly to an idea. But this hobby project may need some air in it. Paradoxically, by not loosening up the language. I feel that writing new texts, I already seem to adopt a ligher way of conveying ideas. Evaluating existing texts seems harder.

I think I lean towards the decision of dropping the goal of reformulating texts to be more 'open'. I still have to get used to the idea.

Little reformulations here and there is something I do when reading over and editing my own texts. Some wording changes I might want to be tolerant towards. But the anxiousness might be something I want to let go of.

I feel like I recognize a personality in my own texts, that I adopted to hopefully be taken more seriously. I feel I am on a different wavelength now, or that I want to be.

So with that new perspective on the work, I was done and satisfied with the last page of the text I was working on within like a minute. Perhaps now I can focus more on what I actually chose to do: scope the project some more.

### 2020-03-07 Do Not Do: 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 a more rigorous planning methodology and higher 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. That felt 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 handles things.

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.

### 2020-03-22 Do Not Do: 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.

### 2020-04-13 Done Brainstorm Reword Circle Language Spec Planning Docs

* Reformulate:
  + 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.
  + I did not read the content of the sub-projects or the idea box.
  + I think, I guess, I would want to go through content top-down for reformulations?
* Tone:
  + 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, I mean more wiggle room in the wording. 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.
  + Cockiness/speaking in definites, and scope/out-of-scope are 2 different things. My intermediate goal now was to change the tone, not to change the scope covered by the documents? There are still gray areas of scope. I think I should deal with that later?

### 2020-01-13 Do Not Do: 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.

### 2020-04-13 Done Brainstorm Scope

* Scope:
  + Some content in the idea box is also out-of-scope.
  + Strategy: Goal of the Language: It mentions fundamental principles, that may become out-of-scope. Higher Goal does too. Will I separate these things? So say: it’s a (diagrammatic) programming language. I had thoughts about saving the higher goals for later.

The scope used to be split in two: Language Spec and the rest.

Now I lean towards a split up in 3 or 4 actually: Language Spec / Broader Perspective / Construct Proposals / Other.

This changes things. This puts even more out of scope of Language Spec. Earlier on I thought anything that has to do with language spec could be in scope of this project, but part of its postponed parts. Yeah this seems arbitrary terminology, that seems to mean the same, but I am going with it. Earlier anything language speccy would be still in the project definition, but deemed optional. This split up was by feeling it is part of the language or not part of the language. Now I actually have in mind 3 categories of things that kind of fall under that umbrella 'part of the language': Language Spec / Broader Perspective / Construct Proposals.

I feel maybe I am not finished making the first split up and now already starting with the next?

>> I want to check that.

Also the new split up would give the wilder ideas a place, not on the big pile 'other', like internet as a single computer would be 'broader perspective'. Actually… you could interpret it differently. Everything not part of Language Spec will at first be put on the pile 'Other'.

Maybe it is better to put off fine-tuning the scope for now. First make it neat the way it is. I was making such nice progress with that, wasn't I?

### 2020-05-08 Done Brainstorm Priorities

* Rewording the Done projects is getting a bit boring.
* / Maybe wrap up Parameters planning doc.
* Scoping was an alternative thing to focus on.
* / Something else could be actually digging into the Circle Language Spec docs themselves.
* I think scoping is a better choice for now.
* I want to go over the Circle Language Spec Plan documents, moving things out of scope off the top of my head.
* Side-goal: I read nicer docs, giving my head a bit of a break.
* I think I will mark things with red before actually moving content out.
* Still lots of definites in the wording of the main planning docs.
* I do like too see it reworded it to nothing being a strict rule anymore and you can always make up your own mind. It seems this does not harm the integrity of the idea.

### 2020-05-10 Done Scoping: Symbol Language and Software System Planning Docs

* Scope the planning docs even further.
* I went through all the planning docs.
* Marked in red things in the Main Project documents, things I would like to place out of scope.
* These two documents may have content to weed out at one point:
  + "2009-08 Review by Ramses, Small Plan.docx" has out-of-scope things. It's just a little hard to extract. Perhaps postpone that when I reconsider reformulating this content again.
    - Marked some content in red.
    - Take out the person's name.
    - Still want to split the document up in two.
  + "Diagrams, Coding Principles & Coding Concepts Plan (obsolete).docx": I may want to consider moving this out of scope, or at least part of it.
    - Marked some content in red.
    - Still want to split the document up in two.
* The other Circle Language Spec Plan docs seem fine regarding scoping for now.
* Still want to move the things marked red in the documents, to the out-of-scope documents.
* / Or consider the 'Software System' topics to put in scope.
  + / The content seems of not much use for inclusion in Circle Language Spec, not the planning docs nor the produced 'specification' docs.
  + / Could be that this folder is relevant: "2001 - 2002 Symbol"
    - / It is basically 4 empty folders to indicate projects, that took place, on the time line, without having a description. There is probably notes in hour sheets about this period.
  + / Could be that this folder is relevant: "2003 - 2009 Software System Documentation"
    - / "2007-10 Rethink Software System, Project Plan.doc":
      * There it seems to start interweaving with Circle Language Spec topics.
      * The goal does not really talk specifically about programming languages. The notes taken while doing that project, mention several topics about programming languages, but just naming larger parts, not details. Not really much can be found that goes specifically into Circle Language Spec and the diagram notation. I do not think much is of use in the Circle Language Spec Plan documentation. It seems 'Symbol attempts' is the real first effort put into the diagram notation, that is mentioned in these planning docs (perhaps aside from what's in the hour loggings, which have not been considered yet).
      * I do not think I can even formulate just a few sentences 'this and that' has been done regarding Circle Language Spec in this project, while the project was also about larger parts. It's just that inconcrete.
    - /"2009-04 01 Full Revision of Documentation 2009-04 - 2009-06"
      * / "Software System Full Revision of Documentation, Goal.doc"
        + / Most things are actually about Software System as a whole.
        + / Only in general the Circle Language Spec is mentioned, but no content seems to be concretely about Circle Language Spec.
      * / "Full Revision of Doc, Elements & Time Estim.doc"
        + / "Main Modules: (19 articles, 90 sections)": There is the first time concrete work for Circle Language Spec is mentioned, done during the project "Full Revision of Documentation 2009-04 - 2009-06".
        + / At "- Sub-Modules: (25 articles, 218 sections)" it starts going into a bit more detail about it.
        + / Up until now, just some main parts of Circle Language Spec are named. Not sure if it is really that significant information to tell about projects and their execution.
        + / It is recognizable for me, the topic subdivision. I could extract some info from it later. I could decide later what to do.
      * / "Software System Full Revision of Documentation, Reflection.doc"
        + / Half way, this reflection turns into almost the literal content of what the goal is of the Circle Language Spec project.
        + I think I can remove it. It is basically the strategy doc of Circle Language spec. Consider that later.
        + / It is interesting to see that Circle Language Spec is exactly the next project after this.
        + / So this is indeed the interlude to the Circle Language Spec project. I like to see those connections.
      * / "02. Software System Full Revision of Documentation, Implementation, Reach Goal A.doc"
        + / Has some detail about Circle Language Spec, but not much.
      * / "04. Software System Full Revision of Documentation, Implementation, Reach Goal B.doc"
        + / Among other things, lists main parts of Circle Language Spec again, for which contents/direction 'pages' were written.
      * / "06. Software System Full Revision of Documentation, Implementation, Finishing Touch.doc"
        + / Almost random details.
    - / The other parts of "2003 - 2009 Software System Documentation" contained nothing relevant.
  + / Should any of that be included in the Circle Language Spec Plan docs?
    - / "2003 - 2009 Software System Documentation" is sort of the prequel to Circle Language Spec. Haha.
    - / And the "2001 - 2002 Symbol" phase starts with initial ideas about Circle language, and ends with the unfinished Symbol.doc, though not much was written down about the progress (though probably things are in hour loggings, not considered yet.)
  + / Looking at "1. Software System.doc":
    - / If none of this part of the planning docs is usable for inclusion in Circle Language Spec planning docs, it may be worth a peek at the actual document Software System.doc or the 'redirection pages'.
    - / I like it. I do not think I see much content I would use for Circle Language Spec. From earlier ideas I had, I expected to find more. Perhaps I looked it over wrong. Or maybe now much is narrowed in scope, some more content from this document does not fall into scope anymore, making less of it very usable in Circle Language Spec. Just bits and pieces maybe, but I expect to find texts of similar content and quality near the topics of Circle Language Spec already.

### 2020-04-18 Done Brainstorm Writing Style

* / It kind of bothers me that the story starts in the middle. Where are the other sub-project descriptions? Maybe mixed with other big projects, like 'Software System Documentation'. Maybe.
* Brainstorm: I worry a lot if the words come off too strongly. On top of that I am not in a very subtle mood right now. If I was feeling more calm and friendly, I might have better judgement about it. Maybe I cannot do this right now. I can only judge language constructions, not whether it feels right. When my base line mood would be calm and friendly, I can just feel whether the words bump with that or not. That might be easier. Sometimes just using calmer language, makes me calmer, but right now it does not seem to be working.
* Wording:
  + Is it really such a problem, that I talk in definites about steps taken and rules of the systematics as I try to work them out? Maybe in talk like that, still add a 'might'. Imagine what if you would not explain to yourself, but to someone else that has to work it out further. Imagine what it would feel like if someone else read it. Like you're being dictated rules, perhaps.
  + After that I might want to change command-like sentences and rule-like statements, to just add 'might' to it or other wiggle words. I would talk to myself that way, just maybe comes off harsh. Maybe being less strict towards myself is also not a bad idea.
  + Added maybe's.
  + Perhaps after that a reformulation phase. I tend to come up with nicer flowing alternatives to just the word 'maybe'.
* Brainstorm wording:
  + Maybe I go a different direction with this.
  + With my gut I might assess whether the language is too blunt or not.
  + Maybe if it just list things, that actually happened, then it might not be much of a problem.
  + Definite language to create a pseudo-truth for a false sense of control, might be a problem.
  + Things stated as rules, that are not hard rules might be a problem.
  + When setting rules to follow a plan, wiggle words might be helpful.
  + Black and white opinion about something that may or may not be true, might be a problem.
  + In some sentences it seems more obvious. If they sound opinionated and black-and-white.
  + Maybe I am looking for clues in texts where there are none. Sort of like when I am driving and see no other traffic, I am scared that there may be traffic, that I am not seeing.
  + Words like 'annoying' or 'constantly' trigger more easily. Stating something as definite, without 'maybe's seems a more subtle 'problem'.

### 2020-04-18 Done Reword 2008-07 02 Assignment Spec Project Summary.docx

* Adding maybe's and perhapses.
* Second phase reformulating > Did not change much, but did change a few things.
* The word 'you' seems overly used. Try to reformulate to object-centric grammar, instead of person-centric.
* Maybe read one more time for possible small typing errors.

### 2020-04-18 Done Reword 2008-05 02 Classes & Relationships Specs Project Summary.docx

* / Rename "Relationship" => "Relationship" > I would keep the term relationship in there for now, otherwise I would have to go through so much documentation. If I refer to a product with the work relationship in it and change it in the planning docs, then I also would have to change it in the language spec docs and that is just a whole lot of work. I briefly looked up a discussion online about these words, and it does not seem people can find a really clear distinction or rule, just wishy washy hand wavy stuff, I think. It's just that in IT the term is usually 'relationship'. In the spirit of not alienating readers you might change it, but it does not seem semantically incorrect would you accidentally use the word 'relationship' instead.

### 2020-04-18 Done Reword 2008-06 02 Command as a Concept Spec Project Summary.docx

* It mentions topics very much out of scope.
* I think it might not be harmful to keep that mentioning in there.
* Maybe I am trying to hide the out-of-scope topics too rigorously.
* There seem to be maybe's and perhapses missing, but it does not seem to sound too blunt.
* Reconsider to split into in-scope and out-of-scope parts.
* Brainstorm: The input / output topics is so present in this sub-project description. It does not seem lightly touching the topic. Sometimes it is half a paragraph that goes on about just that, while it is not in-scope anymore. Maybe it is worth trying to extract it out, so the summaries of the work become simpler.

### 2020-04-18 Done Reword 2008-06 03 Clarify Command as a Concept Spec Project Summary.docx

* I would like to look at the end-result of the project 'Clarify Command as a Concept Spec' to see if the description of the work done still makes sense if you split it in two.
* Maybe the sub-project Clarify Command as a Concept is mostly out-of-scope.
* That whole project should be out of scope, because the end-result is just about input/output and concurrency resolution.
* That would change the super project description too.

### 2020-04-18 Done Reword 2008-07 01 System Objects Spec Project Summary.docx

* Maybe reformulate harsher words, rather than understanding the text of all the notes. The notes are really unpolished.
* > Was at 'Other Issues' taking out the more harsh wordings.
* I see the difficulty of using the words 'concept' vs. 'aspect'. I want to call it 'aspects' everywhere. But contradictory: 'aspects' is now considered out of scope, while I cannot easily prevent mentioning it at all, because it was in the back of my head during some of these projects. I also have this a little with the term 'code base' which I want to rename to 'base of the code', because 'code base' is a term often used differently in IT.
* Out of scope ideas:
  + I see this document talks about what I would rather call a runtime. Figuring out how Circle could work stand-alone up and running is probably not in the scope of this project anymore. Just describing the language, mostly the diagram notation, is.
  + I see a sentence where I call the System Objects essential for the workings of Circle. But if you see the notation as separate from the runtime it does not seem that essential.
  + So how do I reword this putting these ideas out of scope, without harming the integrity of the content?
  + Hyp: In the original text I express that I feel system objects are essential, as a run-time, not for the notation.
  + I keep seeing the idea of a runtime back. It's an interesting idea, but not in the anymore scope as I want to isolate it. I am reluctant to remove the text, because it is so nicely descriptive about that idea.
  + All in all, I am still uneasy about multiple subjects now considered out of scope, being intermixed in the texts of this sub-project description.
  + The 'other issues' seems to contain loose ideas not fully worked out, part of the aspects construct for instance, or about a run-time. It seems to me some of those ideas, e.g. under 'Other Issues' could be cut-pasted to out-of-scope Circle Language projects.
  + It seems the System Objects Spec Project Summary has a some content that is maybe lazily left in there, while they are also like future ideas?
  + Q: The things that I now consider out of scope, that are mentioned in the sub-project planning documentation for System Objects, are those things even inside the actual language spec content for System Objects? Or is the language spec content also littered with now consider out of scope content?
  + I read most of the content of the System Objects chapter. I forgot what for. To assess whether the topics in the planning docs about it even made it into the documentation. Maybe I can assess that later.
  + Read the sub-project documentation again, to see if some things can be moved out-of-scope?
  + As I read the first sentences of the sub-project planning doc, I tend to want to redescribe it, now I can put it into context. As I explain that context, the integrity of the planning doc seems in tact again.
  + I want to keep reading over and reformulating.

### 2020-04-18 Done Reword The Done Circle Language Spec Planning Docs

* Ideas document: ISNSE = Internet as a Single Computer > But why? Internet aS a[N] Single Entity?
* Brainstorm: Scope the Done projects?
  + I was already putting things out of scope, not just reformulating. I have done that for the Future projects, because putting it out of scope was easier than reformulating some of the difficult text. Maybe that rule does not hold up for reformulating the Done projects. I want to realize that the focus is on reformulation, not scoping. Only if it helps me get through the reformulations, I may want to also fine-tune the scoping in the planning docs.
  + So the same argument kind of applies. If I know it is out of scope, I can just move it without further reformulation.
  + Fortunately I am quite clear on what I put out of scope.
  + Q: Will all 'fundamental principles' be put out of scope?
  + Exp: Read some of the actual documentation.
  + A: Yes, 'fundamental principles' is out of scope with the current ideas I have now.
  + By the way: I saw '2. Concurrency' under '3. Operating System' in the Circle Docs. I couldn't find that content before. Found it.
* I tend to want to subdivide all the Done projects into groups, like I did the Future projects. (in the Other folder, not Circle Language Spec).
  + But I get cramps in my face.
  + It's hard, because it means processing many disparate pieces of information, which takes too much energy/strain for me.
  + Feels bad to leave a grouping/categorization half- finished.
  + Can I make this easier for myself? Dim the screen. Dark mode would have been nice, but don't have it on Windows 7. Take it slower. Worry less about (small) mistakes. Accept this will take a while. Forget about everything else?
* Done projects (now out of scope)
  + / 2008-03 02 Fundamental Principles Spec Part A Project Summary.docx
  + / 2008-03 03 Orient in First Four Fundamental Principles Project Summary.docx
  + / 2008-03 04 Computer Language Programmed Within Itself Spec Project Summary.docx
  + / 2008-03 05 Generic, No Generators Spec, Project Summary.docx
  + / 2008-04 02 Get Clearer View over Difficult Topics Project Summary.docx

### 2020-04-18 Done Circle Language Spec Planning Docs: Reword Main Project and Future Sub -Projects

* The aim is to use less definite, more open, language and to take out the personal 'I' form.
* I think a lot of documents do not contain as much prose as Circle Language Spec Strategy, so maybe things are doable after all. Also it does not drain my energy much. It is relaxing work to me. So I am not sure how much it matters, how long it takes.
* / Limitation: I choose to not reorganize/resubdivide the sub projects. I choose to just change the wording.
* Main Project
  + Mostly finished rewriting the content.
* Future Sub-Projects
  + Assignment Spec Update Ideas.md
  + / Automatic Containment Spec Plan.docx
    - Split up topics about Diagram Metrics from Automatic Containment: two topics. Those intertwine in those two documents.
    - Automatic Containment was going to be put out of scope, because it is so difficult and really in the way of explaining how to use these diagrams normally.
    - I said I would focus on reformulating, not on scoping, but I just don't want to reformulate this content, it is so unclear.
    - Would I simply move more to the 'out-of-scope' document? I doubt, because I moved 'Errors' and the 'Concept Construct' from out-of-scope to in-scope but postponed, because they 'felt' part of the language. Now the scoping rules change and I am wondering about how to organize things for a moment.
    - Looking at the folder subdivision of the planning docs, it does seem like it should be moved there. ~~And so does the concept construct. Moved back again.~~
  + Classes Spec Update Ideas.md
  + Commands Spec Update Ideas.md
  + Concept Construct Spec Project Summary.docx
    - Move out-of-scope.
  + Concepts as External Modules Spec Goal (2008-05) (postponed).docx
    - Move out-of-scope.
  + Concurrency Resolution Spec Project Summary.docx
    - Move out-of-scope.
    - 'Advanced Input/Output' is Concurrency Resolution?
    - There is overlap, but I'm not sure. If I read over the product list, I think I'd have to look into it deeper to determine if all of that Input/Output stuff is up for postponement.
    - I have looked at multiple sources of information, but cannot determine yet if the topics mentioned under the Input Output section of the Products List are in scope or out of scope, due to the concurrency resolution's being in-scope and out-of-scope.
    - / I feel part of the TODO products are already done.
    - / I feel 'normal' parameter topic are intermixed with concurrency resolution / automatic execution order too much?
    - Or is it to shed new light on these topics in the area of automatic execution order?
    - / Alternative: Split apart in two: The parts / product list items, that I know are about concurrency resolution / automatic execution order would be put out of scope and then parts I am not sure about, would stay in in-scope with some comment shedding light on that doubt.
    - Exp: Read over a product list in the Future Sub-Project document "Input Output Spec Project Summary.docx".
    - Hyp: All the topics under Input / Output inside the main "Circle Language Spec Product List.docx" are not trivial parameter topics, but more a small ramp of parameter topics leading towards the automatic execution order concept.
    - / Hyp: But the Integration of Parameter Input/Output Concepts do seem already done topics.
    - Hyp: But may fall under the same umbrella: now put into the light of automatic execution order.
    - Parameters and input/output concepts entangle. The proposed product list would aim to disentangle those topics and explain their precise relationship.
    - I had a brainstorm. I put that in the Future Sub-Project doc in the out of scope folder.
  + Diagram Metrics Ideas.md
  + Diagrams, Coding Principles & Coding Concepts Plan (old).docx
    - At first glance, this seems to have the same scope as the whole project Circle Language Spec.
    - Maybe it can be thrown away (or archived), because it's already described by newer documents.
  + Execution Control Spec Update Ideas.md
  + Globality Spec Update Ideas.md
  + Objects Multiplicity Spec Ideas.md
  + Objects Spec Update Idea.md
  + Parameters Spec Update Ideas.md
  + Relationships Spec Update Ideas.md
  + Static Spec Idea.md
    - The second half is an idea, that is explanatory on some of my ideas about static. But also it uses a concept I want to put out of scope: Code = Data. I want to save such concepts for 'Circle Language JJ's Construct Proposals' or something.
    - I would not want to remove it, if I look at that it explains how I see 'static' in a broader view.
    - But to understand my own doubts about the different ways to see the concept of 'static' that idea description explains a lot and otherwise I might think: where did that idea go and how does it work?
    - I feel that this concept of static might have a place in the general descriptions, but to apply this to the concept of Code = Data, though interesting to me, might not have to be part of that, because it may confuse and alienate the reader, who might be more familiar with more regular implementations of the concept.
  + System Objects Spec Update Ideas.md
  + Text Code Spec Project Summary.docx
    - Out of scope
    - But change the main project description too.

### 2020-04-18 Done Circle Language Spec Planning Docs: Format the Project Summaries

* Change formatting in bulk (of Done Projects / Project Summaries)
  + Change fonts and layout in bulk first.
    - All normal text Calibri 11.
    - Heading 2:
      * Use heading 2 for e.g. 'Goal' and 'Date & Time'.
      * Heading 2 Calibri 10 bold, 9pt before, 9pt after
      * Instead of 'Dates' say 'Date & Time'
      * Americanish heading case usage
    - Sub-title:
      * Author and location Calibri 10, italic, 0.5 cm indented
      * No 'date' subtitle, since it is implied by the content of heading 1
      * Make spacing uniform.
    - Indent of normal text is not consistent among documents.
    - Heading 1:
      * Heading 1 month in separate line
      * Heading 1 no comma's at the end of the lines
      * Shorter heading 1
      * Heading 1 Calibri
    - Maybe double-check the formatting afterwards by opening all documents on top of each other, closing them one by one, to see if all is uniform.
    - Forgot the Done projects that have their own folder.
      * > They use different templates, so this formatting and making consisntent, probably does not apply to those.
    - Don't Future Sub-Projects also have documents in that 'project summary' format?
      * Alternative: format those too.
      * Alternative: Go look for more project summaries in the Circle Language Spec Plan folders.
      * Alternative: Don't do it.

### 2020-04-15 Done Circle Language Spec Planning Docs

* I finished 'Circle Language Spec Strategy'.
* 'Circle Language Spec Product List' seems finished.
* The word 'Legacy' is in there, which I tried to avoid.
  + Integration was not the best word, it seemed.
  + 'Inherited' is too ambiguous a term.
  + It is because other languages have those concepts?
  + Oh, wait, it is my own subdivision.
  + Maybe the word Legacy is ok there, because there is only the danger of putting down my own work, not others'.
  + It seems like it could be both.
  + 'Existing' is a little ambiguous too.
  + Why was integrate such a bad word according to my brother?
  + Synonyms?
  + I am going for 'integrate'.
* 'Circle Language Spec Steps & Time Planning' should be next.
  + Yeah, has definite language, that might need more wiggle room.
  + Sometimes I don't even notice it, because I am so used to try and gain a sense of control over the situation, that I just speak in definites.
  + Resolute language is solved.
  + 'Overview for Progress Monitoring' is not the same list as the scored and prioritized work items above it.
  + Why is there even a separate list for 'Progress Monitoring'? It seems handy for quick overview, but why is the main list not quickly giving that overview then? Improve the main list instead and remove the 'Overview for Progress Monitoring'? Because it might just not be handy to have two things.
  + Inheritance is already finished.
    - Do I update that now? Or do I see that as replanning that should be done later? Don't know. It just looks inconsistent this way.
* Done Circle Language Spec Ideas.
  + The wording is in rougher format, but it is not very long, so should be doable.
  + It may have out-of-scope content.
  + Maybe read over again. Second time around it still seems kind of resolute.

### 2020-04-13 Done Circle Language Spec Planning Docs

* I made a slight beginning with a next phase of scoping the project. But I want to turn back.
* Check if I can check this in.
* Do reformulations while maintaining the current scoping.
* This is a lot of work. I am not even sure if I can do it.
* It being forbidden to use the word 'I' and to be afraid to come off cocky… those are very strict rules to set for myself. Maybe too strict. I have trouble with this. Time may be better spent on something else.
* It's weird that if you want to make a project public property, it is strange to have the word 'I' in a project definition. In notes, maybe, but in central project definition it may be weird… But changing this just conflicts with my lack of energy.
* In my efforts to speak in a non-personal tense, I sometimes switch to the 'you' form and then it just sounds like I can't even keep the terms 'I' and 'you' apart… maybe I am over-conscientious and insecure here. 'You' is fine if you explain how one might experience something.
* I read some articles over personal and impersonal forms of language like that.

### 2020-04-09 Done 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?

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

* 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.
* Details:
  + Get rid of content about 'studying'?
  + Command as a Concept Spec Plan seems a super project but later turns out not to be.
    - Actually, it is a parent project.
  + Done project "Document Internet as a Single Computer" is out-of-scope…
    - Lots of fundamental principles may become out-of-scope of the Circle Language Specs Project and might become part of a separate piece of documentation Circle Language Broader View or something.
    - Not sure if right now I want to dump it in a Done folder with all these many projects in it.
    - Does it matter? I am trying to isolate just Circle Language Spec from Other / anything else.

### 2020-04-01 Done 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.
* Formatting:
  + 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 Relationships.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.
    - Remove '& Copyright'
      * > Only found 1 occurrence: 'Circle Language Spec Ideas'.
    - Remove unnecessary title page white space and 'version XXXX-XX' mentioning.
    - Make main heading and such not take up so much space.
    - 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?
      * ~~Months in headings: not entirely convinced about the difference of the order where the month ends up between 'Project Plans' and 'Project Summaries'.~~
    - 'TODO' => 'Notes'? > Cannot find it.
    - Set proofing language to English US. > I don't see the problem anymore. Maybe the conversion to docx fixed it.
    - First pages layout
    - Page header and footer
      * I am good with no page headers and footers.
    - File properties (author and such)
      * If they are odd. Not that odd 🡪 Leave it there.
      * Still have to do Done projects.
    - Update lists of contents
    - Legend:
      * Brainstorm: I might keep the legend as it is. It may be more work to check which colors are used where and to change it? Maybe not, maybe that would be the right way to go.
      * 'Legend' => 'Legend of Symbols and Markings'?
      * Use simpler color coding.
      * There is ambiguity between coloring of back then and now.
        + Brown meant I think postponed (indefinitely). Some share of brown even meant 'part done, part postponed'. Now brown means do not do. (And part done, part postponed is orange.)
        + Light grey used to mean do not do. Now it means done.
      * Normalize the legends.
      * Make it 1 central document
      * This document has a legend right inside it: Concepts as External Modules Spec Goal (2008-05) (postponed).docx
    - Font Calibri
      * Just the normal font, not the headings.
    - / Headings do not have colors.
      * Are not going to work for me anyway, with the low brightness on my screen.
      * It starts being not so functional anymore, the changes.
* 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.

### 2020-04-01 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.

### 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.

### 2019-12-29 Done Brainstorm Scope: Diagrams / Constructs / Gap Lifting

* The application of them are different (of *diagrams, conceptual constructs* and *boundaries lifting*).
* Proposed constructs:
  + Another example is the automatic diagram organization topics. The diagram notation idea can live 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 / constructs drafts / 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. 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, not in a fixed way. An argument that falls away when you open sourcing and not patenting or something, 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 somewhat of 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).
* "Circle Language Spec" / "Circle Language Broader View" / "Circle Language JJ's Construct Proposals"
* I don't like the last name. It seems long.
* Proposals seems more community-based, not single authorish.
* Language Spec vs Broader View:
  + The idea that the diagram expression should be canonical and unambiguously express anything from any computer language, might not be a rule I want to uphold in the Circle Language Spec docs. For instance if C# has certain rules for scoping of implied accessibility rules of members, you might not want to express that in a diagram. It sort of would make no sense if the diagrams are applied like a helper tool in Visual Studio to visualize certain aspects of C# code, to have symbols that disambiguate something, that is not ambiguous according to C#. The concept of canonicalizing things, may have a better place in Circle Language Broader View docs, I mean to isolate from the Circle Language Spec. A reason for unambiguous expression may only become clear, if you look at those ideas about possible broader applications of this notation: specifically where in a diagram you just switch from one source language to the next, by navigating the symbols. (E.g.: Now the diagram expresses something with C# as the source, navigate onward and you may see some diagram expressing something that came from JavaScript… the rules change, maybe the diagram expression should be unambiguous.) I think it is good and keeps it simpler and ideas less stuck on each other, to speak in options for diagram expression. There is a general theme in the expression. It is not all wishy washy, but there is wiggle room for how to use it. I just want to share the idea, not impose how it should be used.

### 2019-12-29 Done Scope: Diagram Topics

Eventually, you might split the now 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 a 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.

### 2019-12-29 Done Brainstorm Scope (Useful)

* Out of scope:
  + 'Operating system' like components
  + Studying
  + Fundamental principles
    - Basically all of them are probably 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.
    - / 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.
    - I think I should take interesting parts from the general Software System documentation.
    - Then drop them somewhere in the Circle Language Spec documentation.
    - Also for the planning docs: there are Software System Documentation out-of-scope projects. Perhaps part of it was Circle Language Design.
    - > Not much usable content is still in scope.
  + 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 enough. I did not have a clear idea back then how to merge the two or three concepts into that diagram language. I had some general ideas, but not entirely concrete. I should just leave it at that. That seems more achievable.

### 2019-12-29 Done Brainstorm Scope

* 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.

### 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 Version Folder 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 Version Folders 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.

### 2019-08-05 Done Version Control

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.

### 2008-08-31 Done Writing Style Ideas

Circle Language Spec, Writing Style,

2008-08-31

Sometimes it is just clearer to have an article,

with diagrams in it straight away, without

any article with just textual explanations.

In the future, the whole form of the documentation

might change as such and have diagram expression

be more present in the conceptual explanations.

JJ