THE MAKING-OF

Creating a Book Using Fractal Semantic Graphs

How one book was built with an agent, in six days and eighty-eight releases

graphs.sgit.ai · written against v0.5.19 · 26 August 2026

Written by an AI agent, on the repository it describes.

Contents

Creating a Book Using Fractal Semantic Graphs · 5

What this book is · 5
Who it is for · 5
What you will know at the end that you do not know now · 5
How to check anything in this book · 6
Disclosure · 6
What is locked and what is not · 7

1 · The loop · 8

One hour, start to finish · 8
The five steps · 8
The four numbers · 9
Why an hour and not a week · 10
What holds it together · 11
The loop is not new; the cost of it is · 11

2 · The first book, and the pivot that saved the second · 12

Three days to a book · 12
The plan for the second book, and the memo that stopped it · 14
Why this is the most valuable move in the book · 15
Freezing the first edition · 16
The empty second edition · 16

3 · Briefs as the contract · 17

The rule · 17
What a brief actually looks like · 17
Why verbatim · 18
The reading is not optional either · 19
The questions are where the trust lives · 19
The failure mode nobody warns you about · 20
What to copy · 21

4 · Gates buy speed · 22

The counterintuitive claim · 22
The seven checks that came first · 22
The unit suite · 23
The gates that check the content, not the code · 24
The release note as a gate · 24
Automating the release itself · 25
The arithmetic · 26

5 · From a page to an instrument · 27

Where it starts · 27
The reader arrives · 28
Options as an instrument panel · 29
The design law · 30
What it is for · 31
The component proof · 32
Where it lands · 34

6 · Two agents, one repository · 35

The situation · 35
The four notes · 35
What the collisions actually were · 36

The four rules · 37
The rule that generalises past agents · 37
The founder’s part in it · 38

7 · The failures, lovingly · 39

Failures only a person using the thing could find · 39
Failures the picture itself revealed · 40
Failures where the agent’s own tools lied · 43
Failures of claim, not of code · 44
What the pattern is · 44

8 · Reviewing out loud · 46

The problem with written feedback about visual things · 46
What the transcript actually looks like · 46
What the tool got right · 46
The finding that mattered · 47
The inversion: pages that broadcast their own state · 48
The rule for anyone building anything an agent must diagnose · 49
Why this is easier than writing a specification · 49

9 · The experiments · 51

Building a lab · 51
What the experiment was · 51
What shipped, in one afternoon · 52
The restructure: twelve folders · 53
Keeping zooming · 55
How to run experiments this way · 57

10 · The founder’s craft · 58

1. Point at a thing, then say what is wrong with it · 58
2. Say the principle, not only the fix · 58
3. Name the reference · 59
4. Ask “does this make sense, any questions?” · 59
5. Praise the specific thing, then correct · 60
What he did not do · 60
The trade this represents · 60

11 · The playbook · 62

Before you start: what this is and is not · 62
Day 0: set up five things · 62
Day 1: your first memo · 63
The loop, in five steps · 64
Rules that make it work · 64
What to expect to go wrong · 65
Signs the loop has stalled · 65
The honest costs · 65
The one-page version · 66

12 · What it costs, and where it loses · 67

What was not produced · 67
What was extracted · 67
The costs nobody mentions · 67
Four situations where you should do something else · 69
What it does buy · 69
The honest summary · 70

Appendix A · One brief, annotated · 71

The header the brief carries · 71

Segment 1 · The praise, and the thing that worked · 71
Segment 2 · The central ask · 72
Segment 3 · The principle, stated · 72
Segment 4 · The direction, not the ask · 73
Segment 5 · Where it is all going · 73
Segment 6 · The measurable ask · 73
Segment 7 · The small, concrete last request · 74
The agent’s reading, as published · 74
The questions, as published · 75
What shipped, thirty-three minutes later · 75
The whole loop, in timestamps · 75

Appendix B · The chronology · 77

Friday 21 August 2026 · 77
Saturday 22 August 2026 · 77
Sunday 23 August 2026 · 78
Monday 24 August 2026 · 79
Tuesday 25 August 2026 · 79
Wednesday 26 August 2026 · 80
The shape of it · 81

Appendix C · The harness · 82

1 · Time travel: photographing a page as it was · 82
2 · The numbers · 85
3 · One caution about your own history · 88
4 · The figures in this book · 88

Colophon · 90

How it was made · 90
The structure, and why · 90
What was cut · 91
What remains open · 91
Honesty positions carried from the corpus · 92
Disclosure · 92
Licence and provenance · 92

Creating a Book Using Fractal Semantic Graphs

How one book was built with an agent, in six days and eighty-eight releases

---

What this book is

This is the true story of how a second book got built, told for people who want to build one the same way.

The book being built is *Fractal Semantic Graphs: Meaning Through Connectivity*. It is not finished. Its first edition shipped, was frozen, and now sits at `/v1/` of a website called graphs.sgit.ai; the second edition is being written now, out in the open, by one person talking into a phone and a set of AI agents turning what he says into working software and rendered pages, several times a day.

Everything in this book happened between the morning of 21 August 2026 and the evening of 26 August 2026. In those six days the repository behind that website went from version v0.1.0 to version v0.5.11: eighty-eight tagged releases, each one validated, tagged and deployed by a machine, each one carrying a paragraph in a public release table saying what changed and why.

The story stops at v0.5.11 because that is where the repository was when this book was commissioned. It ends cleanly, not tidily. The second book is not written. What exists is the machine for writing it, and the machine is the interesting part.

Who it is for

You have a book in you. You have an AI agent at hand. You suspect the two facts are related and you do not know what the working day actually looks like.

That is the reader this book is written for. You do not need to know what a graph database is. You do not need to be able to program. Roughly nothing in the six days described here required the author to write a line of code, and the parts that did are called out where they happen.

What you do need is the willingness to run a repository, ship small, and be told when you are wrong by a test rather than by a reader.

What you will know at the end that you do not know now

1. **The shape of the working day.** A loop that runs voice memo, verbatim brief, build, release, live review, usually inside one hour. The evidence for it, and the numbers underneath it.
2. **Why verbatim beats paraphrase.** The single highest-leverage habit in the whole method, and the one most authors get wrong on day one.
3. **Why gates make you faster.** Eighty-eight releases in six days was possible because of the test suite, not despite it. The counterintuitive arithmetic of that.

4. **What failure looks like here.** Six real failures, told with the cost attached: an hour lost to a browser that would not die, a bug caught on an iPad that no test would have found, a wire that jumped a layer and betrayed the whole architecture.
5. **How to steer an agent.** What the memos actually contain, why the good ones are specific and small, and what “does this make sense, any questions?” buys you.
6. **A playbook you can start on Monday.** Chapter 11 stands alone. Tear it out. It lists what to set up, what to say first, what to expect to go wrong, and what it costs.

How to check anything in this book

Every scene here can be re-opened.

The repository carries a git tag for every one of the eighty-eight releases. Every figure in this book was taken by checking out the tag its caption names, serving that checkout on a local port, and photographing the page as it actually was on the day. None of the screenshots are reconstructions. Appendix C gives the exact scripts.

Every number was computed, not remembered. Where a number comes from the release table it is quoted; where it comes from git history the command that produced it is in Appendix C. Where the corpus is silent, this book says so.

Every quotation from the founder is verbatim, including the transcription errors, and is already published in the repository at the brief it names. The readings around those quotations are the agent’s, and are marked as such, exactly as they are in the source.

Disclosure

This book was written by an AI agent (a Claude Code session) working from a written commission, on the repository it describes. That is the same arrangement the book describes, applied to itself, which is both the point and a hazard: an agent narrating a method it is itself an instance of has an obvious interest in the method looking good.

Three things are done about it.

The judgements are marked as judgements. Where this book says a thing worked, the evidence is named and you can disagree with the reading. Where it says a thing failed, the failure is a matter of public record in the release table, not an admission extracted under duress.

The failures get real space. Chapter 7 is the longest chapter for a reason.

The costs are stated. Chapter 12 is about what this method does not buy, what it charges, and the four situations in which you should do something else.

The wider disclosure the estate already publishes applies here too. The people who built this have an interest in the argument being right: they sell tools built on it. That is worth holding in mind in every chapter, and particularly in the ones where the story is elegant.

What is locked and what is not

The title of the book being written, *Fractal Semantic Graphs: Meaning Through Connectivity*, is the founder's and is fixed. The title of this book, *Creating a Book Using Fractal Semantic Graphs*, was set in the commission. Everything else here, structure, chapter count, voice, which figures to take and which stories to tell, was the writing agent's call, made confidently, and is recorded as such.

One position from the corpus travels with everything in this book, because the corpus insists on it and would be misrepresented without it: **this is not a graph database pitch**. There is no graph database anywhere in the system described here, and no RDF, SPARQL or Cypher in the code. The graphs are JSON files in a git repository.

Version written against graphs.sgit.ai at v0.5.11, 26 August 2026. **The repository** `SGit-AI/SGit-AI__Website__Graphs`, branch `dev`, tags `v0.1.0` to `v0.5.11`.

1 · The loop

After this chapter you will be able to describe, in one diagram, the working cycle that produced eighty-eight releases in six days, and you will know the four numbers that tell you whether your own cycle is running or stalled.

One hour, start to finish

At 13:44 on 26 August 2026 the repository behind graphs.sgit.ai was tagged v0.5.2. The release note for it opens like this:

> The WCLM: a deterministic transformer over our graphs. Brief 31, the founder's self-described "crazy experiment", built in its own folder (v2/wclm/) with its own code, exactly as asked, and it works.

Thirty-three minutes later the repository was tagged v0.5.3, and its note opens:

> Every box explains itself, both ways. Brief 32, recorded through the viewer's own loop after the founder tried the WCLM ("this WORKED really WELL"), built the same hour.

Between those two timestamps a person opened a page that had existed for half an hour, used it, spoke into a microphone about what he wanted next, sent two screenshots, and received a rebuilt page with every element on it clickable and explaining its own provenance in both directions.

That is the loop. It is not a metaphor for the process. It is the process, and it ran between eighty-seven consecutive pairs of releases over six days.

The five steps

1. The founder records a voice memo. Usually while using the thing that shipped an hour ago, on a phone or an iPad, often in the middle of a sentence about something else. It is transcribed automatically, by otter.ai in most cases or by the site's own chat panel once that existed. The transcription is bad in the ordinary ways: "thinking in graphs" comes out as "tignogen graphs", "peak board" comes out as "pasteboard", "nodes" comes out as "modes".

2. The memo is published verbatim as a brief. Not summarised. Not cleaned up. The transcript, garbles and all, goes into a numbered markdown file in the repository under `v2/briefs/`, gets a rendered page on the website, and is the source of truth for what was asked. The agent's reading of it is written underneath, in a table, marked as the agent's. Chapter 3 is entirely about why this ordering matters.

3. The agent builds. Code, data, generated pages, the release note, the tests. In the six days covered here there was never a design document written before the build that was not itself a brief; the plan and the build were the same act, and the record of what was decided is the release note.

4. The release ships itself. A push to the `dev` branch runs a pre-release validator. If the validator passes, a machine tags the commit `vX.Y.Z` and deploys the site. If the validator fails, nothing tags and nothing deploys. There is no manual release step and no staging environment where things sit.

5. The founder looks at the live site and starts again at step 1. Often within minutes. Sometimes he sends the memo before the previous release has finished deploying, which is how three separate instructions arrived inside one afternoon and shipped together as v0.5.5.

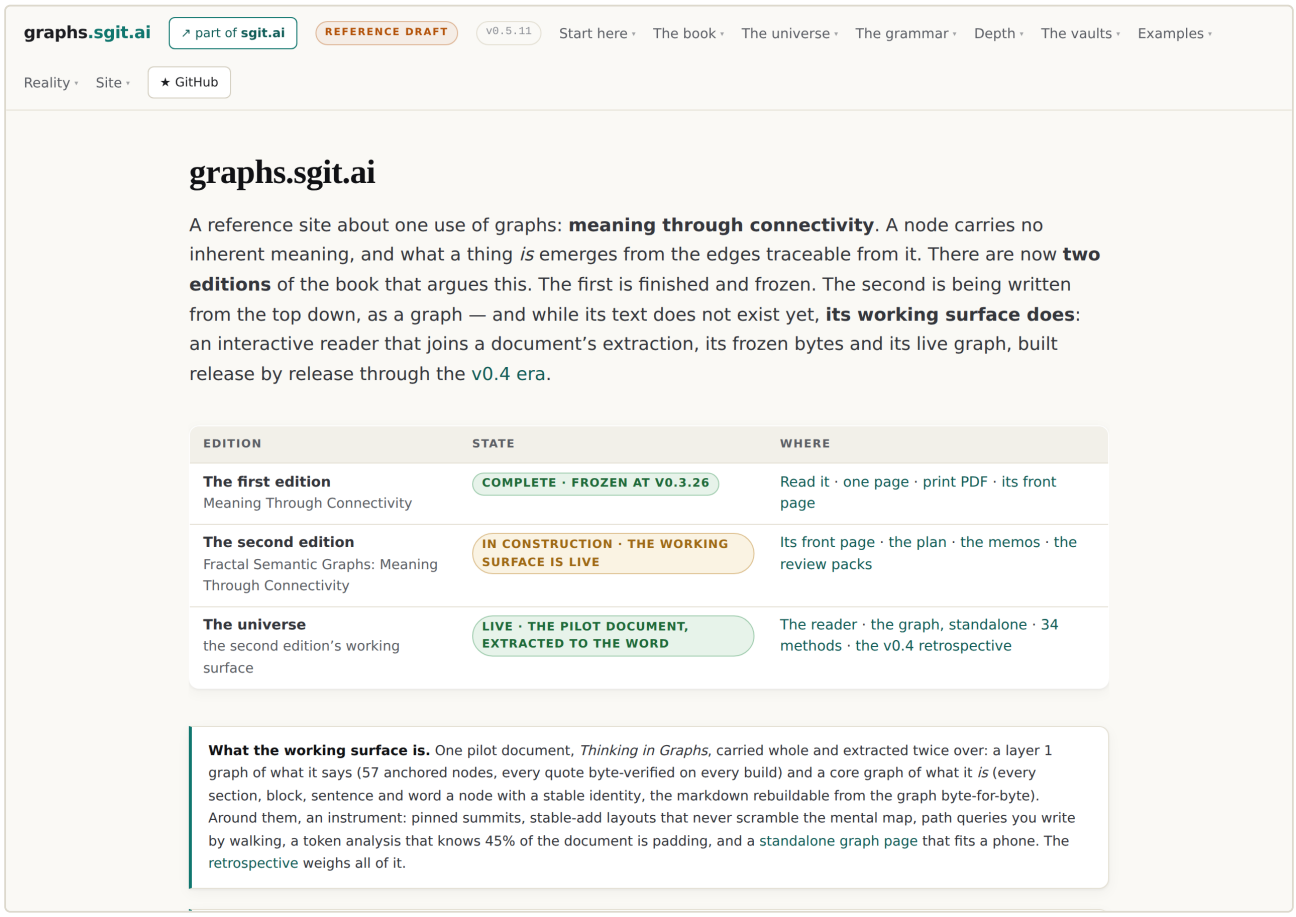

Figure 1. The front page at v0.5.11, re-taken from the tag `v0.5.11` on a local server.

The four numbers

The loop either runs or it does not, and you can tell which from four measurements. All four below are computed from the repository's own git history; the commands are in Appendix C.

Releases per day. Thirteen, sixteen, fourteen, twenty, nine, sixteen. Eighty-eight in six days, an average of 14.7 a day. The dip to nine on 25 August is worth reading before you conclude it was a slow day: this book's judgement, and it is a judgement, is that it is the day the two hardest design problems of the period were worked (the stability principle of Chapter 5, and navigation as query), and that each release that day carried more than the ones around it.

The gap between releases. The median gap between two consecutive tags is 31.2 minutes. If you exclude the overnight gaps and look only at pairs less than ten hours apart, that is, releases inside the same working session, the median is 29.3 minutes. Forty-two of the eighty-seven gaps are under half an hour.

The working window. The first and last release of each day, in UTC:

DATE	RELEASES	FIRST TO LAST	SPAN
21 Aug	13	14:47 to 21:39	6.9 hours
22 Aug	16	09:53 to 23:40	13.8 hours
23 Aug	14	00:17 to 22:25	22.1 hours
24 Aug	20	13:05 to 23:03	10.0 hours
25 Aug	9	11:48 to 20:12	8.4 hours
26 Aug	16	09:22 to 21:09	11.8 hours

Two of these days are longer than any reasonable person should work. That is worth saying plainly, and Chapter 12 says it again with the costs attached. The loop does not require this pace; it survives being run twice a week. But the density is what makes the evidence in this book unusually good, because six days of near-continuous releases leave a record with almost no gaps in it.

The ratio of releases to commits. The repository contains 97 commits up to v0.5.11. Eighty-nine of them have a subject line beginning `site v`, which is the release convention. Eight do not: three merge commits, four working commits on side branches, and the initial commit.

That last ratio is the one that surprised me most when I computed it. Ninety-two per cent of all commits to this repository are releases. There is essentially no such thing as work-in-progress in the main line. A change either passes the gate and ships, or it does not exist yet.

Why an hour and not a week

The obvious objection to a thirty-minute release cadence is that it produces churn: lots of motion, no direction. The record does not support that reading, and the reason is structural rather than heroic.

A release is small enough to hold in one head. Each of the eighty-eight release notes describes one coherent change with its verification. Read v0.4.27 in full:

> The close button now says so. The founder asked how to close the chat panel — which means the bare ✕ among nine header buttons did not read as “close the panel” on a wrapped iPad header. It is now labelled ✕ close, visually set apart with its own hover, its tooltip says what happens (the button brings the panel back), and Escape closes the panel from anywhere except inside an input, where Escape belongs to the field. Three new suite checks: the label, the click, and the Escape path. 60 checks green. No book content changed.

That entire release is one button label, one keyboard shortcut and three tests. It shipped three minutes after v0.4.26. It is not churn, it is a question answered before it could become an assumption.

A short loop turns opinion into evidence. When the founder said, at v0.5.4, that “meaning without connectivity” should not answer identically to “meaning through connectivity”, the disagreement did not have to be argued. It was run, on the live page, and the page agreed with him. Thirty minutes later the engine had a negation stage and the answer had changed. The cost of being wrong about a design question fell to about the cost of the conversation about it, which changes what kind of questions get asked.

The reviewer is reviewing the real thing. Every one of the founder's memos in this period was recorded while using the deployed site, not while reading a plan. There are no mockups anywhere in this story. There is no design phase separate from the build. That is only possible because the build takes minutes.

What holds it together

Speed like this normally breaks something. Three mechanisms stop it, and each gets its own chapter later:

The brief (Chapter 3) fixes the instruction. Because the founder's words are captured verbatim before the agent reads them, there is a permanent record of what was asked that is independent of what got built. An instruction cannot be quietly softened into an easier one, because the harder version is sitting in the repository with a URL.

The gates (Chapter 4) fix the quality. The pre-release validator grew from seven checks at v0.1.0 to sixteen; the unit suite went from thirteen tests at v0.4.13 to eighty-four at v0.5.9. A release that breaks an internal link, or lets a page drift from the markdown it claims to render, or quotes a document at bytes the document does not contain, does not tag and does not deploy.

The release note (Chapter 4 again) fixes the memory. Every release note in this repository is a paragraph of prose that says what changed, why, what was verified and how. They are long. v0.5.7's is over five hundred words. They are the single most useful artefact in the whole repository, because six days later they are the only reliable account of what happened, and unlike a commit log they were written by somebody who knew why.

The loop is not new; the cost of it is

None of the five steps above is an invention. Recording what a stakeholder actually said is old practice. Shipping small is old practice. Automating validate, tag and deploy is old practice.

What changed is the cost of step 3. When the build step takes an agent twenty minutes instead of a team two weeks, every other step in the loop can be shortened to match, and the loop as a whole crosses a threshold: it becomes faster to build the thing than to argue about whether to build it. Past that threshold, the discipline you need is not about planning better, it is about capturing intent faithfully and refusing to ship anything you have not checked.

That is the whole method. The rest of this book is what it looks like in practice, including the days it went wrong.

Where the live estate shows this. The release table at `/admin/versions.html` and its two archive pages, `/admin/versions-v0.4.html` (41 rows) and `/admin/versions-earlier.html` (35 rows), are the loop's own record: every release, dated, with a paragraph on what happened. The briefs it responded to are at `/v2/memos/` for the second edition and `/v1/briefs/` for the first.