GranSim ODD README

This collection describes GranSim, the Kirschner and Linderman groups' hybrid agent-based model of *Mycobacterium tuberculosis* granuloma formation. It follows the ODD protocol (Overview, Design concepts, Details) as updated by Grimm et al. 2020 (JASSS 23(2):7, doi:10.18564/jasss.4259).

Author: Christian T. Michael. Large language models were used to assist drafting these documents.

Contents

The description is split into one central docuent (the *Spine*, `ODD_spine.md`) and several companion documents.

- **The Spine** describes the high-level architecture of *GranSim*, explaining what each part of the model does and how they fit together. The Spine is arranged by the seven ODD elements described in Grimm 2020:
    1. Purpose and patterns
    2. Entities and scales
    3. Schedule
    4. Design concepts
    5. Initialization
    6. Input data
    7. Submodels.
- The Companion documents hold the detailed specifications. Submodels of *GranSim* Submodels are numbered `S#.md`, while modules (interchangeable or optional components) `M#.md`. We also provide `Appendix_B.md`, which indexes one of our more commonly-used configurations of *GranSim* (i.e., which modules are on and off, etc.). Element 7 of the spine describes the contents of each companion document in more detail. Note that we have endeavored to make sure that each model rule is stated in exactly one document, while others refer to it where appropriate.

How the GranSim ODD was drafted

Phase 1 - Generating the verbose ODD from the GranSim source code

Christian T. Michael (author), with U-M GPT as drafting assistant. U-M GPT is the university's secure AI service, and it received the C++ source directly. This phase produced the "verbose version" of each document that the revision below takes as input.

Ground rules for AI prompts

- **Standard:** Grimm et al. 2020, following the numbering of Figure 1 wherever the text disagrees.
- **Template:** the accepted TumorSim ODD, which is derived from GranSim. Its strong conventions were carried over: notation declared once, values stated once, explicit code availability, schedule pseudocode, and a reproducibility enumeration. Its errors were corrected rather than inherited.
- **Authority:** "describe what the program does, not what you think the model does." The ODD was derived

from the C++ source, not from the literature.

- **Self-sufficiency:** the text alone must be sufficiently detailed to permit reimplementation.

Flow

```
Grimm 2020 + TumorSim ODD (template)
|
v
[1] TRIAGE        file inventory -> out-of-scope files pruned;
|                 main loop read first (schedule is the spine)
v
[2] CONFIGURATION study command line + LHS ranges file fix the
|                 configuration: 2D, 300 x 300, 154 d, FFT (estimate
|                 plan), probabilistic recruitment, no drugs, no
|                 receptor ODEs, no TGF-beta -> no nested ODDs needed
v
[3] CHECKPOINTS   structured extraction after each source batch:
|                 schedule, entities, fields, equations, parameters,
|                 running ambiguity list
v
[4] BATCH INDEX   source grouped into numbered batches (1, 2a-2c, 3-10)
|                 with a minimum batch set per document
v
[5] SPINE         Elements 1-7 drafted; patterns and one flowchart
|                 left pending Results text
v
[6] COMPANIONS    one conversation per document, each opened with a
|                 standard handoff prompt:
|                 skeleton first -> approval -> sections in order
|                 S1-S9 (submodels), M1-M8 (modules), Appendix B (toggles)
v
[7] AUDIT         fresh thread per document, checking claims against
|                 source and classifying them (VERIFIED, INCORRECT,
|                 INERT HERE, DEAD CODE, ...)
v
Verbose versions -> revision (below)
```

Companion-document protocol

- **Skeleton first.** Each document opened with its full section list. Every heading carried a placeholder written as a work order, naming the mechanisms, identifiers, parameters and discrepancies to cover. The skeleton, not the prose, was the unit of continuity: a UM-GPT chat session that ended mid-document due to context limitations left a complete specification for the next session.
- **Structure from the code.** Sections followed the code: one per state machine, per effector and per coupling. There was no target section count.
- **Four fixed sections** in every document: scope and schedule (with a reads-and-writes table), parameters, module dependencies, and reimplementation notes.
- **Correctness before polish.** Drafts were written for accuracy and completeness. Compression and

register were deferred to the revision.

What made it work

- **Main loop before headers.** Reading the timestep schedule first gave the call graph, the coupling points and the list of files actually needed.
- **The command line as configuration.** Module toggles live on the command line, so a single invocation resolved scope, removed about 40% of the source from consideration, and eliminated the need for nested ODDs.
- **Checkpoint artefacts.** Distilled extractions outlived the raw code in context and seeded later sessions without re-pasting source.
- **Arithmetic and consistency checks while reading.** These surfaced candidate defects for the team. Examples: O_2 consumption applied for the full diffusion step in each Strang half; a neutrophil-injection off-by-one; an unguarded division in the MHC II statistics. The same checks corrected inherited TumorSim errors, such as the misstated timestep-synchronisation constraint.

Lessons

- **Context limitations:** The source code cannot be read in one thread. Batching, triage and checkpoints were necessary from the start.
- **Reply limits of UM-GPT.** Oversized replies failed outright rather than truncating. Large sections, especially parameter tables, had to be split into chunks, planned in advance, along subsection boundaries. Chunks left no trace in the text.
- **Copy-path formatting.** Content moved from the chat interface to Vim. That required line prefixes, a placeholder for backticks, ASCII-only output, and display-equation delimiters on separate lines.
- **Length and repetition.** First drafts were accurate but long and repetitive across documents. That motivated the DRY revision that follows.

Phase 2 - Revision

Christian T. Michael (author), with ClaudeAI as coordinator and editor. Review by Christian T. Michael and Paul Wolberg.

Ground rules

- **Standard:** Grimm et al. 2020, following the numbering of Figure 1.
- **Authority:** the code decides what the program does. Papers, comments and intent are supporting evidence only.
- **DRY:** every rule has one home. The spine explains, the S# documents specify submodels, the M# documents specify modules, and Appendix B indexes the toggles.
- **No silent changes:** any revision that alters a claim is tagged VERIFY. Open questions go to per-document audit files, linked from the body by anchors such as `<!-- audit: S5-A003 -->` that sed can strip.
- **Context limits:** no thread reads everything. Each step works from compact intermediate files: ledgers, maps and registers.

Flow

```
ODD_spine.md + S1-S9.md + M1-M8.md + Appendix_B.md          (Verbose Versions)
|
v
[1] LEDGER        one thread per document: every claim -> a row,
|                 with destination (keep / move / dup / FIX? / VERIFY)
v
[2] DECISIONS     author resolves each ledger's open findings
|
v
[3] HANDOFF  ---> secure model answers code questions (Q-###)
|            <--- secure_response.md (+ errata)
v
[4] APPLY         answers applied to each decisions file
|
v
[5] TRIM          one thread per document: rewrite to its ledger;
|                 TODO / VERIFY / dead code -> {DOC}_audit.md + anchors
v
[6] CHECK         fresh thread per trim (never the trimmer) ---+
|                                                              |
|                 "pass with issues" -> FIX -> narrow re-check -+
v
[7] RESTRUCTURE   optional renumbering; then a narrow re-check
|
v
[8] MAP           one thread per document -> checks/map/{DOC}_map.md
|                 (sections, links, interfaces, parameters, order, VERIFY)
v
[9] REDUCE        one thread reads only the maps ->
|                 validation_report, section_map, verify_register, spine_inputs
v
[10] SPINE        one owner; spine ledger first, then element by element
|                 (3+2.2 -> 5+2.1,2.3-2.6 -> 4 -> 1, 7, 6, 7.0)
v
[11] REDUCE x2    repeated after the spine -> CROSS-DOC FIXES (one thread per doc)
|                 -> REDUCE x3 -> author decisions (decisions_9nov.md)
v
[12] APPLY        one thread, owner-first; local copies, exact Edit,
|                 diff against the original, upload with local_path
v
Reviewable draft  (internal review by Christian T. Michael and Paul Wolberg)
|
v
[13] CODE CHECK   secure model, batch by batch, from verify_register.md
|                 (audit_packet_prompt.md)
v
[14] REVISION -> ASSEMBLY from a configuration manifest -> public ODD
```

What made it work

- **Independent checking.** A fresh thread checked every trim, never the thread that made it. Every "pass with issues" went through a fix and a re-check before the next step.
- **Map/reduce validation.** Validating the whole collection in one thread proved too large. Each document instead writes a summary under 8 KB, and one thread compares the summaries.
- **Owner-first edits.** When ownership moved, the owner was completed before the other copy was cut, so no claim fell between documents.
- **Removable audit layer.** Open questions sit in audit files that the team can review. The body strips clean with one `sed` command.

Observations of challenges found during drafting

- **Large files.** Any file over about 100 KB slows or breaks a thread. Staged reading and compact ledgers are essential.
- **Project storage.** The 2 MB store of ClaudeAI forced regular backups and deletions: original documents, check reports, decisions files, then ledgers.
- **Rewriting files.** Whole-file rewrites through the Projects tool risk silent drift. Edit local copies, diff them against the originals, then upload from disk.
- **Background sub-agents** often cannot reach project files. Run them in the foreground and have them report when they fail.
- **Cost.** The process is thorough but expensive: it used the entirety of a weekly ClaudeAI Max allowance.