Adapting SM-2 and SRS Algorithms to Incremental Writing and Task Management

Background: SM-2 and Incremental Practices

- **SM-2** is the original Spaced Repetition (SRS) algorithm used by SuperMemo and popularized in tools like Anki. It schedules items for review based on recall performance, dynamically extending intervals to optimize retention[1] [2].
- **Incremental writing** involves the gradual development and refinement of ideas and documents, typically returning to drafts at spaced intervals, rather than linear completion[3].
- **Task management with SRS** involves “submerging” tasks into a pool and resurfacing them for action or review at calculated intervals.

Prior Art

1. Incremental Reading (SuperMemo, Obsidian, Anki)

- SuperMemo pioneered incremental reading, where excerpts and notes are reviewed and expanded using SRS. This allows simultaneous progress on many texts, reprioritizing as knowledge/needs evolve[4] [3] [5].
- Incremental writing has been integrated with SRS in various note-taking platforms (Obsidian plugins, Anki extensions). Notes or writing fragments are queued and scheduled for review or further elaboration[6] [7] [8] [9].

2. Tasklists in SRS Context

- SuperMemo’s tasklist manager combines prioritization and spaced review. Tasks can be reviewed, postponed, updated, or promoted based on changing priorities, with features for introducing tasks into the SM-2 schedule (“memorizing” a task)[10] [11].
- Users have adapted SRS (Anki, RemNote, and others) to habit formation and recurring task reminders, treating each task as an “item” and using review intervals to refresh one's engagement with delayed or deferred tasks[12] [13] [14].

3. Academic and Workflow Use Cases

- Notes, research problems, and partially developed ideas are often added to an “inbox” or list. Review and work on these notes uses SRS to periodically surface items for attention, preventing them from stagnating or being forgotten[15] [16] [17].

Approaches for Adapting SM-2

Approach	Description	Pros	Cons
Item-level Incremental Writing	Each draft, idea, or paragraph is treated as an "item" in an SM-2 schedule. Review sessions involve editing, expanding, or refactoring the item.	Easy to implement; smooth context-switching; supports vast "idea gardens."	Item atomization is difficult; potential cognitive load if too granular.
Task Submersion & Surfacing	Every task/subtask is inserted into an SRS queue. Review responses (e.g., progress, blocked, not started) dictate next interval.	Tasks won't be forgotten; can batch similar tasks; adjustable to urgency.	SM-2's focus on memory, not task urgency, may lead to out-of-sync priorities; not optimized for dependencies.
Dynamic Prioritization	Use SM-2's "ease factor" and interval as a priority estimator: highly active or important tasks/items are reviewed frequently; others fade.	Hybridizes value-urgency; automates triage; fits both knowledge fragments and tasks.	May not suit deadlines; items could drop below awareness threshold ("eternal backlog").
Quality-based Feedback	On reviewing an item/task, users rate "progress" or "clarity" (from 0-5 as in SM-2). Low scores reset interval or trigger deeper review/edit; high scores delay resurfacing.	Flexible; lets user surface only what most needs attention; handles "blocked" or "deferred" states well.	Requires regular, honest feedback; more subjective than rote memory tasks.
Incremental Merging and Consolidation	As in SuperMemo's writing process, multiple "atomic" ideas are eventually merged during the consolidation phase, with SRS ensuring previously outlined points return for context before integration[3].	Prevents lost threads; supports synthesis for complex documents.	Requires additional tooling for merging; cognitive complexity increases with note granularity.

Novel and Experimental Directions

- **Serendipitous Idea Collision:** By leveraging SRS's pseudo-random resurfacing, you can prompt unexpected combinations of ideas or tasks during review, encouraging creativity or cross-pollination[17] [3].
- **Value/Time Priority Heuristics:** As in SuperMemo's tasklists, integrate time-to-completion and estimated value directly into SRS scheduling, "promoting" items that become more urgent or valuable over time (functionally overlaying Eisenhower matrix style prioritization) [10] [18].
- **Auto-archiving and Dormancy:** If an item repeatedly scores low (uninteresting, blocked), the system can auto-archive it, making room for active tasks or ideas, but allowing for periodic resurfacing to prevent permanent loss[16] [10].
- **Context-aware Scheduling:** Algorithms can factor in upcoming deadlines or dependencies, adjusting "ease factor" or overriding SM-2 intervals to resurface items/tasks as required—a blend of SRS and traditional task managers[19] [18].

Tradeoffs

- **SM-2 for Non-memorization Tasks:** The fit is strongest where maintenance and gradual improvement are valued over immediate completion. Adapting it for deadlines or multitasking with dependencies may require augmenting with prioritization frameworks or manual overrides.
- **Cognitive Load:** Overloading the system with too many items (fragments/tasks) can lead to "review fatigue" if intervals or granularity aren't well-regulated[15] [20].
- **Subjectivity in Ratings:** Unlike factual recall, evaluating progress on ideas or tasks is subjective, potentially reducing algorithmic reliability if not paired with discipline and good review hygiene[15] [16].
- **Discovery vs. Efficiency:** Random resurfacing can yield creative breakthroughs, but may feel inefficient compared to tightly managed, goal-driven workflows. Balancing serendipity and structure is key[17] [3] [15].

Summary

Adapting SM-2 (and similar SRS algorithms) to incremental writing and task management is already a part of advanced knowledge and productivity workflows, especially with tools inspired by SuperMemo. The core tradeoff is between maintaining a living backlog of ideas/tasks versus the precision needed for prioritization and completion. Using SM-2's exponential scheduling and feedback ratings fits best where gradual progress and retention matter more than strict deadlines. Blending SRS with additional prioritization, dormancy controls, and creative surfacing strategies extends its utility to the incremental development of both documents and to-do lists[15] [17] [16] [3] [10].

⁂

Mapping SM-2 Algorithm Parameters to Creation and Task Execution

The SM-2 algorithm, widely used for learning (memorization), can be adapted for creative work and task management by reinterpreting its core parameters: **easiness** (ease factor), **quality response**, **repetition counts**, and **failure**. Below are specific ways to map these parameters to suit incremental writing or task workflows:

SM-2 Parameters and Their Learning Roles

- **Easiness Factor**: Reflects how easy it is to recall an item; affects scheduling interval length.
- **Quality Response**: User grades each review attempt (0–5), influencing adjustments to intervals and ease factor.
- **Repetitions**: The count of successful reviews.
- **Failure**: Occurs when quality ≤2; resets repetition count and intervals[21] [22].

Mapping to Creation (Writing/Ideation)

SM-2 Parameter	Creation/Ideation Mapping	Practical Example
Easiness Factor	*Maturity or clarity of an idea/draft.* High easiness = idea is nearly complete/clear; low easiness = needs more work.	A paragraph that is nearly publish-ready gets a higher easiness, scheduled for less frequent revisits.
Quality Response	*Progress rating for an item/idea.* Score from 0 ("stuck"/needs overhaul) to 5 ("flowing"/nearly done).	Each review, give a 0–5 rating on how happy you are with a draft fragment.
Repetitions	*Number of development iterations.* Increments only when progress is made or the idea is advanced.	“Third round of revisions” on a section means 3 repetitions.
Failure	*Stalling, confusion, or creative block.* Restarts cycle; triggers more frequent attention.	If an idea is deemed fundamentally flawed or confusing (rated ≤2), it is resurfaced sooner.

Mapping to Task Execution

SM-2 Parameter	Task Management Mapping	Practical Example
Easiness Factor	*Task familiarity or straightforwardness.* High value = routine/minor edits; low value = complex/uncertain.	A recurring, easy task (filing a report) will be scheduled less often than a novel, complex one.
Quality Response	*Progress or blockage rating.* From 0 ("completely blocked") to 5 ("smooth/finished").	After attempted work session, rate progress; blocked tasks stay "in the queue."
Repetitions	*Number of returns or review cycles for the task.*	“Fourth attempt” at a stalled task = 4 repetitions.
Failure	*Failure to complete or make headway.* Triggers more frequent priority/focus, can mean need to break into subtasks or seek support.	If a task is not progressing after several tries (rated ≤2), raise urgency or change approach.

Tradeoffs and Implementation Notes

- **Subjectivity**: Rating “easiness” and progress is more subjective in creative work than in memory recall, introducing potential bias[22].
- **Granularity**: Tasks or creative fragments must be clearly defined to avoid overwhelming cognitive load or excessive fragmentation.
- **Failure as Guidance**: Frequent rating of “failure” should be leveraged as a signal to restructure, seek help, or reprioritize the item[21].

Recap Table: Learning vs. Creation/Task Mapping

Learning	Creation	Task Management
"Easiness" = recall ease	"Easiness" = item clarity/maturity	"Easiness" = task familiarity
"Quality" = recall grade	"Quality" = subjective progress score	"Quality" = progress/blockage
"Failure" = forgot	"Failure" = blocked/stuck	"Failure" = blocked/no progress

When adapting SM-2, map each review to a targeted check-in: Is this idea/task clear and progressing, or stalled and needing help? Use the ratings and calculated intervals to regulate when to revisit and rework items, ensuring neither ideas nor tasks languish for too long without progress[21] [22].

⁂

1. https://github.com/thyagoluciano/sm2
2. https://en.wikipedia.org/wiki/SuperMemo
3. https://supermemo.guru/wiki/Incremental_writing
4. https://help.supermemo.org/wiki/Incremental_reading
5. https://soki.ai/insights/what-is-incremental-reading
6. https://forum.obsidian.md/t/implementing-spaced-repetition-for-incremental-writing-workflow/92361
7. https://www.obsidianstats.com/plugins/obsidian-incremental-writing
8. https://forums.ankiweb.net/t/incremental-reading-in-long-term-future/596
9. https://github.com/bjsi/incremental-writing
10. https://help.supermemo.org/wiki/Tasklist_manager
11. https://super-memory.com/help/overviews.htm
12. https://andrewtmckenzie.com/spaced_repetition
13. https://www.reddit.com/r/todoist/comments/189osyn/spaced_repetition/
14. https://notes.andymatuschak.org/Spaced_repetition_systems_can_be_used_to_program_attention
15. https://notes.andymatuschak.org/Spaced_repetition_may_be_a_helpful_tool_to_incrementally_develop_inklings
16. https://notes.andymatuschak.org/Spaced_repetition_may_be_a_helpful_tool_to_incrementally_develop_inklings?stackedNotes=zUP4GuzPF33dWkZPiu9N6V5
17. https://cesarrodrig.github.io/garden/implementing-a-spaced-repetition-writing-system.html
18. https://help.supermemo.org/wiki/Features
19. https://en.wikipedia.org/wiki/Spaced_repetition
20. https://supermemo.guru/wiki/Advantages_of_incremental_reading
21. https://github.com/thyagoluciano/sm2
22. https://www.blueraja.com/blog/477/a-better-spaced-repetition-learning-algorithm-sm2