Monte Carlo Simulation
MS Planner Field Alignment

Agentic Congress Planner (ACP) - Architecture Document
Prepared by Claudia | Sunrise Gen AI | Feb 2026

Monte Carlo Simulation - MS Planner Field Alignment

1. Duration Distributions (Core Random Variables)

MS Planner Fields -> Simulation Variables:

- startDateTime -> Task Start Time: Anchor point for scheduling
- dueDateTime -> Planned Deadline: Target completion constraint
- completedDateTime -> Actual Completion: Historical actual - due = delay distribution
- createdDateTime -> Task Creation: Lead time analysis start - created = planning buffer

Monte Carlo Mapping:

- For each task type, compute from historical data:
  - P(duration) = distribution fitted from completedDateTime - startDateTime across past congresses
  - P(delay) = completedDateTime - dueDateTime -> fit to log-normal or triangular distribution
  - These become the random variables sampled in each simulation run

2. Dependency Graph Construction

- orderHint -> Implicit sequencing within a bucket
- bucketId / Bucket name -> Task grouping -> parallel vs sequential inference
- checklist items -> Sub-task decomposition -> micro-dependencies
- references (links) -> Cross-task references -> explicit dependency edges

Monte Carlo Mapping:

- Build a DAG where edges = dependencies
- Each node's start is max(predecessor completion times)
- Simulation samples each node's duration from its fitted distribution
- Run N=10,000 iterations -> output completion time distribution for full congress

3. Resource & Assignment Modeling

- assignments (userId -> assignment) -> Resource allocation: who owns the task
- assignments.assignedDateTime -> Response time: assignedDateTime - createdDateTime
- assignments.orderHint -> Priority among assignee's tasks
- appliedCategories (labels) -> Task classification -> category-specific distributions

Monte Carlo Mapping:

- Resource contention modeling: If person X has 5 parallel tasks, simulate queuing delays
- From historical data: compute per-person P(throughput) -- how many tasks/day they actually complete
- Simulation enforces: a person can only work on K tasks concurrently (K derived from historical data)
- KOL-specific: Speakers assigned via assignments -> model their response/confirmation latency

4. Progress & Risk Signals

- percentComplete (0, 50, 100) -> Current state for live re-simulation
- priority (0-10) -> Task criticality weight
- taskDetail.description -> NLP extraction -> risk keywords, complexity signals
- taskDetail.previewType -> Attachment presence -> documentation completeness proxy

Monte Carlo Mapping:

- Priority -> weighting in objective function (high-priority task delays penalized more)
- percentComplete at time T vs historical pattern -> predict if task is ahead/behind
- Tasks at 50% longer than historical median at 50% -> inject additional delay probability

5. Bucket-Level Simulation (Workstreams)

Venue & Logistics: Physical setup -> Long-tail risk distribution (vendor delays)
Speaker Management: KOL coordination -> High variance (external dependency)
Content & Agenda: Program design -> Medium variance, internal dependency
Travel & Accommodation: Flight/hotel booking -> Event-driven (flight cancellation injection)
Registration: Attendee management -> Low variance, predictable

Monte Carlo Mapping:

- Each bucket gets its own category-specific distribution fitted from historical data
- Cross-bucket dependencies modeled as inter-workstream edges in the DAG
- Simulation can inject disruption events with configurable probability (e.g., P(flight_cancel) = 0.03 per speaker)

6. Simulation Output -> Decision Support

Each of the 10,000 runs produces a complete congress timeline. Aggregate to get:

- P95 completion date: 95% confident congress prep completes by date X
- Critical path probability: which tasks appear on critical path most frequently
- Bottleneck ranking: tasks/people that cause delays in most runs
- Risk heatmap: buckets with highest variance
- Buffer recommendation: optimal slack to add per task/workstream

7. Sample Simulation Pseudocode

```
for run in 1..10000:
    for task in topological_order(DAG):
        # Sample duration from historical distribution
        duration = sample(task.duration_distribution)

        # Resource contention delay
        assignee_load = current_tasks(task.assignee)
        if assignee_load > historical_K:
            duration += sample(queuing_delay_dist)

        # External event injection
        if task.bucket == "Travel":
            if random() < P_FLIGHT_CANCEL:
                duration += sample(rebooking_delay_dist)
        task.simulated_start = max(pred.simulated_end for pred in predecessors)
        task.simulated_end = task.simulated_start + duration
    record(congress_completion_time, critical_path_tasks)
output(percentiles, bottlenecks, risk_heatmap)
```