Deep Analysis: Cartedo Simulation Adaptation Framework

Date: 2026-01-16 **Total Lines of Code:** 23,300 **Total Files:** 42

Part 1: Alignment & Adaptation Logic (How It Actually Works)

The Complete Pipeline Flow

```
STEP 1: INPUT
   |
   v
STEP 2: FACTSHEET EXTRACTION (Gemini)
   - Extract company name, manager, industry
   - Generate poison list (terms to avoid)
   - Generate KLOs for TARGET scenario
   - Generate industry_context (KPIs, terminology)
   |
   v
STEP 3: SHARDER
   - Split JSON into 15 shards
   - Lock 2 shards (workspace_ids, scenario_options)
   - Hash each shard for change detection
   |
   v
STEP 4: PARALLEL SHARD ADAPTATION (Gemini)
   - Each unlocked shard sent to LLM
   - Factsheet injected into every prompt
   - RAG context added per shard type
   |
   v
STEP 5: MERGE SHARDS
   - Combine 15 shards back into single JSON
   - Run cleanup_merged_json()
   |
   v
STEP 6: ALIGNMENT CHECKER (GPT 5.2)
   - 9 parallel LLM checks
   - Calculate weighted score
   - If score < 95% -> trigger fixer
   |
   v
STEP 7: ALIGNMENT FIXER (GPT 5.2)
   - Phase 1: Fast fixes (string replacement)
   - Phase 2: Batched LLM fix (questions, content, resources)
   |
   v
STEP 8: VALIDATION AGENT (GPT 5.2)
   - 8 Critical checks (blocking)
   - 6 Flagged checks (non-blocking)
   |
   v
STEP 9: FINISHER
   - Compute weighted compliance score
   - Decision: PASS / FAIL_RETRY / FAIL_HUMAN
   |
   v
STEP 10: OUTPUT
```

Step-by-Step Breakdown

STEP 1-2: Factsheet Extraction (The Foundation) **File:** `src/utils/gemini_client.py` (846 LOC)

What happens:

```
# Line 234-237 in adaptation_engine.py
global_factsheet = await extract_global_factsheet(
    source_scenario=source_scenario,  # "HR Hiring at Summit Innovations"
    target_scenario=target_scenario,  # "Market Entry for EcoChic Threads"
)
```

Output structure:


```
{
  "company": {
    "name": "EcoChic Threads",
    "industry": "Sustainable Fashion Retail",
    "type": "DTC Startup"
  },
  "reporting_manager": {
    "name": "Maya Sharma",
    "role": "Director of Market Strategy",
    "email": "maya.sharma@ecochic.com",
    "gender": "female"
  },
  "poison_list": [
    "Elizabeth Carter", "Summit Innovations", "HR",
    "hiring", "candidate", "interview", "recruitment"
  ],
  "klos": [
    {"id": "klo1", "outcome": "Analyze market opportunity using TAM/SAM/SOM"},
    {"id": "klo2", "outcome": "Evaluate internal capabilities and resource fit"},
    {"id": "klo3", "outcome": "Develop go/no-go recommendation with risk assessment"}
  ],
  "industry_context": {
    "kpis": ["CAC", "LTV", "Gross Margin", "Market Share"],
    "terminology": ["market entry", "competitive analysis", "PESTEL"],
    "wrong_terms": ["HR metrics", "hiring KPIs", "recruitment funnel"]
  }
}
```


PROBLEM IDENTIFIED: The poison_list is generated from the TARGET scenario prompt, but the prompt may not contain explicit domain terms like "HR", "hiring", "candidate". These are domain-IMPLIED terms that the LLM doesn't automatically include.

STEP 3: Sharder (Divide & Conquer) **File:** `src/stages/sharder.py` (407 LOC)

What happens:

```
# Splits JSON into 15 shards:
SHARD_DEFINITIONS = [
    "workspace_ids",         # LOCKED - never modified
    "scenario_options",      # LOCKED - never modified
    "lesson_information",    # Contains manager emails
    "overview",              # Scenario overview text
    "guidelines",            # Student guidelines
    "simulation_flow",       # Core activities + resources
    "submission_questions", # Assessment questions
    "rubric",                # Grading criteria
    "assessment_criterion", # KLO definitions
```

```
    "emails",               # Manager communications
    "characters",           # Personas
    "resources",            # Learning materials
    "workplace_scenario",   # Scenario narrative
    "activities",           # Task definitions
    "stage_definitions",    # Workflow stages
]
```

PROBLEM IDENTIFIED: KLOs are in `assessment_criterion` shard but questions are in `simulation_flow` shard. When processed independently, there's no cross-shard awareness of KLO-question alignment.

STEP 4-5: Parallel Adaptation & Merge **File:** `src/stages/adaptation_engine.py` (846 LOC)

What happens:

```
# Each shard gets this prompt structure:
prompt = f"""
## GLOBAL CONTEXT (MANDATORY):
{factsheet_json}

## POISON LIST (NEVER USE THESE):
{poison_list}

## RAG CONTEXT (Industry Examples):
{rag_context}

## TASK:
Adapt this {shard_name} content from {source} to {target}.

## INPUT SHARD:
{shard_json}
"""
```

PROBLEM IDENTIFIED: Each shard is adapted INDEPENDENTLY. Even though factsheet is injected, the LLM doesn't see the actual KLO definitions when adapting questions in `simulation_flow` shard.

STEP 6: Alignment Checker (Quality Gate) **File:** `src/stages/alignment_checker.py` (1,472 LOC)

What happens:

```
# 9 parallel checks run using asyncio:
check_coroutines = [
    self._check_reporting_manager_consistency(),  # R1
    self._check_company_consistency(),             # R2
    self._check_poison_terms(),                    # R3
    self._check_klo_to_questions(),                # R4 <- FAILING (90%)
    self._check_klo_to_resources(),                # R5 <- FAILING (88%)
    self._check_scenario_to_resources(),           # R6 <- FAILING (90%)
    self._check_role_to_tasks(),                   # R7 <- FAILING (90%)
    self._check_klo_alignment(),                   # R8 <- FAILING (90%)
    self._check_scenario_coherence(),              # R9 <- FAILING (90%)
]
```

Score Calculation:

```
# Line 322-324
if results:
    overall_score = sum(r.score for r in results) / len(results)
# Example: (96+97+100+90+88+90+90+90+90) / 9 = 92.33%
```

PROBLEM IDENTIFIED: The LLM checks run against the MERGED JSON, so they CAN see cross-shard relationships. But by this point, the content is already wrong - the checker can only DETECT issues, not prevent them.

STEP 7: Alignment Fixer (The Repair Agent) **File:** `src/stages/alignment_fixer.py` (1,782 LOC)

What happens:

```
# Phase 1: Fast fixes (no LLM)
if rule_id == "reporting_manager_consistency":
    # String replacement: find/replace manager names

# Phase 2: Batched LLM fix
if llm_fix_rules:
    batched_result = await self._fix_all_batched(
        json_data=fixed_json,
        failed_rules=llm_fix_rules,
        factsheet=global_factsheet,
    )
```

The Bug (Lines 391-399):

```
# Looks for questions HERE:
submission_questions = topic_data.get("submissionQuestions", [])

# But questions are actually in simulationFlow[].data.submissionQuestions!
# This is why fixes=0 - it can't find the questions to fix
```

PROBLEM IDENTIFIED: Alignment fixer searches for `submissionQuestions` at the TOP LEVEL, but they're nested inside `simulationFlow[].data.submissionQuestions`. The fixer returns `fixes=0` because it finds an empty list.

Why Alignment Score is Stuck at 92.33%

Rule	Score	Why It Fails
R4: KLO-Questions	90%	Questions are duplicated across KLOs, not KLO-specific
R5: KLO-Resources	88%	Resources truncated, missing financial model
R6: Scenario-Resources	90%	Resources still contain HR terminology
R7: Role-Tasks	90%	Tasks don't map to Junior Consultant role
R8: KLO-Task	90%	Activities not linked to specific KLOs
R9: Coherence	90%	"optimization" vs "go/no-go" mismatch

Weighted Average: (96+97+100+90+88+90+90+90+90) / 9 = **92.33%**

Part 2: Agent Clashes & Conflicts

CLASH 1: Alignment Fixer vs Semantic Fixer

Aspect	Alignment Fixer	Semantic Fixer
File	`alignment_fixer.py` (1,782 LOC)	`fixers.py` (1,612 LOC)
Triggered By	Alignment score < 95%	Validation failures
Fixes	KLO mapping, question alignment	Entity replacement, KPIs
Model	GPT 5.2	GPT 5.2
Input	alignment_report	validation_report

THE CLASH: Both agents try to fix "HR terminology" but neither succeeds because: - Alignment Fixer: Can't find questions (wrong path) - Semantic Fixer: Runs AFTER alignment fixer, by then alignment score is locked

FIX NEEDED: Either merge the two fixers OR ensure alignment fixer runs BEFORE semantic fixer with correct paths.

CLASH 2: Adaptation Engine vs Alignment Checker

Aspect	Adaptation Engine	Alignment Checker
File	`adaptation_engine.py` (846 LOC)	`alignment_checker.py` (1,472 LOC)
Model	Gemini	GPT 5.2
When	BEFORE shards merge	AFTER shards merge
Context	Per-shard only	Full JSON

THE CLASH: - Adaptation uses Gemini with per-shard context - Alignment uses GPT 5.2 with full JSON context - When Gemini makes a mistake, GPT detects it but can't fix it (different model, different context)

FIX NEEDED: Either use same model for both OR pass full KLO context to Gemini during adaptation.

CLASH 3: Check Definitions vs Alignment Rules

Source	Threshold	What It Checks
`check_definitions.py` C7	95%	KLO Preservation
`alignment_checker.py` R4	95%	KLO-Questions
`alignment_checker.py` R5	95%	KLO-Resources

THE CLASH: Three different places check "KLO alignment" with slightly different logic: - C7 checks if KLOs exist and are preserved - R4 checks if questions map to KLOs - R5 checks if resources support KLOs

A run can pass C7 but fail R4/R5, creating confusion about what "KLO alignment" means.

FIX NEEDED: Consolidate KLO checks into single source of truth.

CLASH 4: Finisher Thresholds vs Alignment Thresholds

Stage	Threshold	Pass Condition
Alignment Checker	95%	overall_score >= 0.95
Finisher	80% blocker, 70% overall	blocker_pass_rate >= 0.8 AND overall >= 0.7
Dashboard	95%	acceptance_threshold = 0.95

THE CLASH: - Alignment says 92.33% = FAIL - Finisher says 100% compliance = PASS - Dashboard says 95% = FAIL

A run can PASS finisher but FAIL alignment, creating confusion about release readiness.

FIX NEEDED: Align all thresholds to single standard (95%).

Part 3: Detailed Agent Analysis

STAGE 1: Adaptation Engine

File: `src/stages/adaptation_engine.py` **Lines of Code:** 846 **Model:** Gemini 2.0 Flash

One-Line Summary: Generates initial adapted JSON by extracting factsheet, splitting into shards, and running parallel LLM adaptation with RAG context.

Key Functions:

Function	Lines	What It Does
`extract_klos_from_json()`	50-91	Extracts KLOs from multiple possible locations in source JSON
`format_klos_for_prompt()`	93-124	Formats KLOs for injection into adaptation prompts
`AdaptationEngine.adapt()`	152-400	Main entry point - orchestrates full adaptation
`_adapt_shard()`	401-500	Adapts single shard with factsheet + RAG context
`_get_shard_rag_context()`	501-550	Retrieves shard-specific examples from ChromaDB

Data Flow:

```
Input JSON
    |
    v
extract_global_factsheet() --> factsheet
    |
    v
Sharder.shard() --> 15 shards
    |
    v
[Parallel] _adapt_shard() x 13 --> adapted shards
    |
    v
merge_shards() --> adapted JSON
```

Critical Dependencies: - `gemini_client.py` for LLM calls - `sharder.py` for JSON splitting - `prompts.py` for prompt templates

---

STAGE 2: Sharder

File: `src/stages/sharder.py` **Lines of Code:** 407 **Model:** None (pure Python)

One-Line Summary: Splits simulation JSON into 15 semantically independent shards with lock states and hash tracking for change detection.

Key Functions:

Function	Lines	What It Does
`Sharder.shard()`	50-150	Splits JSON into shards based on SHARD_DEFINITIONS
`Sharder.merge_shards()`	151-250	Combines shards back into single JSON
`cleanup_merged_json()`	251-350	Post-merge cleanup (URL fix, placeholder removal)
`compute_shard_hash()`	351-400	SHA256 hash for change detection

Shard Lock States:

```python
class LockState(Enum):
    UNLOCKED = "unlocked"      # Can be adapted
    LOCKED = "locked"          # Never sent to LLM
    LOCKED_AFTER_FIX = "locked_after_fix"  # Locked after structural fix
```

STAGE 3: Alignment Checker

File: `src/stages/alignment_checker.py` **Lines of Code:** 1,472 **Model:** GPT 5.2

One-Line Summary: Validates cross-shard semantic alignment using 9 parallel LLM checks with weighted scoring against 95% threshold.

Key Functions:

Function	Lines	What It Does
`AlignmentChecker.check()`	228-351	Main entry - runs all 9 checks in parallel
`_check_reporting_manager_consistency()`	408-485	Verifies manager name/email/role consistency
`_check_company_consistency()`	487-560	Verifies company name consistency
`_check_poison_terms()`	562-650	Scans for source scenario leakage
`_check_klo_to_questions()`	652-750	Maps questions to KLOs
`_check_klo_to_resources()`	752-850	Maps resources to KLOs
`_check_scenario_to_resources()`	852-950	Verifies scenario-resource alignment
`_check_role_to_tasks()`	952-1050	Verifies role-task alignment
`_check_klo_alignment()`	1052-1150	Verifies overall KLO structure
`_check_scenario_coherence()`	1152-1250	Verifies narrative coherence

Scoring Guidelines (Lines 358-379):

```
0.95-1.0: Perfect
0.85-0.94: Excellent
0.75-0.84: Good
0.65-0.74: Acceptable
0.50-0.64: Needs Work
Below 0.50: Poor
```

STAGE 3B: Alignment Fixer

File: `src/stages/alignment_fixer.py` **Lines of Code:** 1,782 **Model:** GPT 5.2

One-Line Summary: Fixes alignment-specific issues using fast string replacement for consistency and batched LLM calls for KLO/resource alignment.

Key Functions:

Function	Lines	What It Does
`AlignmentFixer.fix()`	147-250	Main entry - orchestrates fix phases
`_fix_manager_consistency()`	500-600	Fast string replacement for manager names
`_fix_company_consistency()`	601-700	Fast string replacement for company names
`_fix_all_batched()`	252-362	Batched LLM fix for all KLO/resource issues
`_fix_questions_batch()`	364-500	Fixes submission questions for KLO alignment
`_fix_content_batch()`	700-900	Fixes guidelines/overview/scenario text
`_fix_resources_batch()`	901-1100	Fixes resources for industry alignment

THE BUG (Line 396):

```python
# WRONG: Looks at top level
submission_questions = topic_data.get("submissionQuestions", [])

# CORRECT: Should look in simulationFlow
for stage in topic_data.get("simulationFlow", []):
    stage_questions = stage.get("data", {}).get("submissionQuestions", [])
```

STAGE 4: Fixers (Structural + Semantic)

File: `src/stages/fixers.py` **Lines of Code:** 1,612 **Model:** GPT 5.2

One-Line Summary: Applies surgical JSON patches for structural (shape) and semantic (meaning) fixes using JSON Pointer operations.

Key Functions:

Function	Lines	What It Does
`StructuralFixer.fix()`	100-300	Fixes schema violations (missing keys, wrong types)
`SemanticFixer.fix()`	301-600	Fixes content issues (entities, KPIs, terminology)
`_identify_fixes()`	601-800	Uses LLM to identify exact JSON paths to fix
`_apply_patches()`	801-1000	Applies JSON Pointer patches to JSON

Fix Types:

```python
class FixType(Enum):
    STRUCTURAL = "structural"  # Shape only
    SEMANTIC = "semantic"      # Meaning only
```

STAGE 5: Validation Agent

File: `src/validation/validation_agent.py` **Lines of Code:** 360 **Model:** GPT 5.2

One-Line Summary: Produces human-readable validation reports for PM/QA decision-making with 8 critical (blocking) and 6 flagged (non-blocking) checks.

Key Functions:

Function	Lines	What It Does
`ValidationAgent.validate()`	50-150	Main entry - runs all checks
`_run_critical_checks()`	151-220	Runs 8 blocking checks
`_run_flagged_checks()`	221-280	Runs 6 non-blocking checks
`_generate_report()`	281-350	Generates structured report

STAGE 6: Finisher

File: `src/stages/finisher.py` **Lines of Code:** 392 **Model:** None (pure Python)

One-Line Summary: Computes weighted compliance score from validation results and makes final pass/fail decision.

Key Functions:

Function	Lines	What It Does
`Finisher.run()`	74-150	Main entry - computes compliance
`_compute_compliance_score()`	151-250	Calculates weighted score
`_decide_status()`	251-300	Returns PASS/FAIL_RETRY/FAIL_HUMAN

Pass Conditions:

```python
# Line 41-43
@property
def passed(self) -> bool:
    return self.blocker_pass_rate >= 0.8 and self.overall_score >= 0.70
```

Part 4: 10 Deep Questions for Seniors

Question 1: Alignment Fixer Path Bug

"The Alignment Fixer searches for questions at `topic_data.submissionQuestions`, but they're actually in `simulationFlow[].data.submissionQuestions`. This bug has been present for 4+ runs. Should we add schema documentation that maps where each content type lives to prevent similar issues?"

Question 2: Cross-Model Repair Strategy

"Gemini adapts the content, GPT detects issues, then GPT tries to fix Gemini's mistakes. Have we measured if this cross-model repair actually works, or does GPT just introduce different errors?"

Question 3: Threshold Inconsistency

"We have three different pass thresholds: Alignment (95%), Finisher (70-80%), Dashboard (95%). Which is the actual release gate? A run can PASS finisher but FAIL alignment."

Question 4: Poison List Generation

"The poison list misses domain-implied terms like 'HR', 'hiring', 'candidate' (185 occurrences). Should we maintain a static domain-to-terms mapping, or ask the LLM to infer domain vocabulary?"

Question 5: Shard Isolation vs Cross-Shard Alignment

"KLOs and questions are in different shards, processed independently. How do we maintain KLO-question alignment when each shard is adapted without seeing the other?"

Question 6: Duplicate Check Systems

"Both Alignment Checker and Validation Agent check 'KLO alignment' with different logic. Should we consolidate into single source of truth?"

Question 7: Resource Truncation Root Cause

"Resources are truncated mid-content. Is this (a) LLM output limit, (b) JSON parsing error, (c) context window exceeded, or (d) prompt doesn't specify minimum length?"

Question 8: Human Escalation Threshold

"After 4 consecutive failures (FAILED14-20) with same issues, no human escalation triggered. What should be the threshold for automatic human review?"

Question 9: Retry Strategy

"MAX_RETRIES=1 for speed, but we're stuck at 92.33%. Should we increase retries specifically for alignment-critical fixes that are close to threshold?"

Question 10: Validation-Fixer Loop

"Validation agent produces detailed failure report, but fixers don't consume it directly - they re-run their own checks. Why aren't we passing validation failures to the specific fixer agent?"

Part 5: One-Line Summary

"The 23,300 LOC pipeline achieves 92.33% alignment but has 4 critical clashes: (1) alignment fixer can't find questions due to wrong path, (2) cross-model repair (Gemini->GPT) may not be effective, (3) threshold inconsistency between stages (95% vs 70%), and (4) shard isolation breaks cross-shard KLO alignment."

Analysis completed: 2026-01-16 Files analyzed: 42 Total LOC: 23,300