WTB Implementation Summary - 2025-01-10

Today's focus was implementing the **Event Bus, Audit Trail, and Batch Test Infrastructure** as outlined in the TODO plan. All P0/P1 priority items for today have been completed successfully.

Executive Summary

Metric	Value
Tests Before	208 passing
Tests After	275 passing (+67 new)
Files Created	16 new files
Files Modified	6 files
Key Components	Event Bus, Audit Trail, Batch Runners, Environment Providers
Code Review	✅ Passed (2 minor issues fixed)

1. Event Bus & Audit Trail (Morning Session)

1.1 WTBEventBus Implementation

Key Design Decisions:

Decision	Choice	Rationale
Threading Model	RLock (Reentrant)	Allows handlers to publish additional events without deadlock
History Storage	`deque(maxlen=N)`	Bounded memory, O(1) append/pop
AgentGit Integration	Optional Bridge	Can operate standalone or bridge AG events to WTB
Global Instance	Singleton + Reset	Easy testing with `reset_wtb_event_bus()`

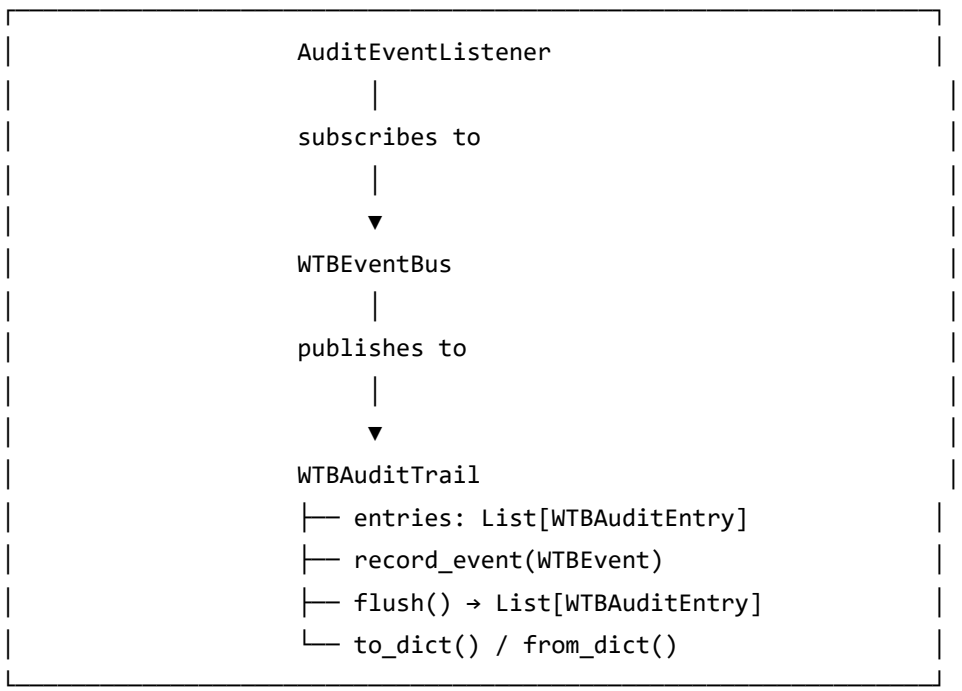
Implementation Highlights:

```
wtb/infrastructure/events/wtb_event_bus.py
├── WTBEventBus class
│   ├── subscribe(event_type, handler) - Thread-safe subscription
│   ├── publish(event) - Broadcast with history tracking
│   ├── get_history(event_type?, since?, limit) - Filtered retrieval
│   └── enable_agentgit_bridge() - Optional AG event translation
├── get_wtb_event_bus() - Global singleton accessor
└── reset_wtb_event_bus() - Test isolation helper
```

Test Coverage: 20 tests including thread-safety stress tests

1.2 WTBAuditTrail Implementation

Architecture:



Key Features:

- Auto-mapping of WTB domain events to audit entries
- Severity levels: INFO, SUCCESS, WARNING, ERROR, DEBUG
- Import capability for AgentGit audits (detailed debugging)
- Memory-safe `flush()` for long-running batch tests

Test Coverage: 24 tests

2. Batch Test Runner Infrastructure (Afternoon Session)

2.1 Interface Design

IBatchTestRunner - Abstract contract for batch execution:

```python
class IBatchTestRunner(ABC):
    @abstractmethod
    def run_batch_test(self, batch_test: BatchTest) -> BatchTest

    @abstractmethod
    def get_status(self, batch_test_id: str) -> BatchRunnerStatus

    @abstractmethod
    def get_progress(self, batch_test_id: str) -> Optional[BatchRunnerProgress]

    @abstractmethod
    def cancel(self, batch_test_id: str) -> bool

    @abstractmethod
    def shutdown(self) -> None
```

IEnvironmentProvider - Execution environment isolation:

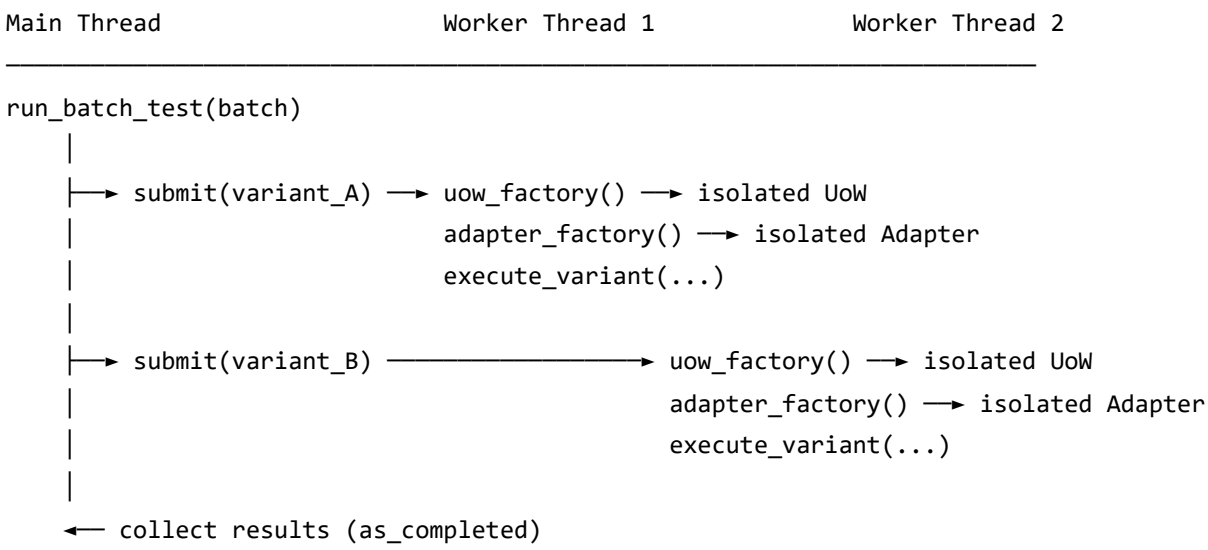
```python
class IEnvironmentProvider(ABC):
    @abstractmethod
    def create_environment(self, variant_id: str, config: Dict) -> Dict

    @abstractmethod
    def cleanup_environment(self, variant_id: str) -> None

    @abstractmethod
    def get_runtime_env(self, variant_id: str) -> Optional[Dict]
```

2.2 ThreadPoolBatchTestRunner

Design Decisions:

Aspect	Implementation	Rationale
Parallelism	ThreadPoolExecutor	WTB is I/O-bound (DB, LLM calls); GIL released during I/O
Isolation	Per-thread UoW + StateAdapter	Prevents cross-thread state corruption
Progress Tracking	Internal `_RunningTest` state	Enables `get_progress()` during execution
Cancellation	Future.cancel() + flag	Graceful stop with partial results

Thread Isolation Pattern:



2.3 BatchTestRunnerFactory

Usage Pattern:

```python
# Auto-select based on config
runner = BatchTestRunnerFactory.create(config)

# Explicit ThreadPool
runner = BatchTestRunnerFactory.create_threadpool(max_workers=4)

# Explicit Ray (if ray_enabled in config)
runner = BatchTestRunnerFactory.create_ray(config)

# For testing
runner = BatchTestRunnerFactory.create_for_testing()
```

3. Ray Foundation (Evening Session)

3.1 RayConfig Dataclass

Added to `wtb/config.py`:

```python
@dataclass
class RayConfig:
    ray_address: str = "auto"
    num_cpus_per_task: float = 1.0
    memory_per_task_gb: float = 2.0
    max_pending_tasks: int = 100
    max_retries: int = 3
    runtime_env: Optional[Dict[str, Any]] = None
```

Presets:

- `RayConfig.for_local_development()` - Single node, minimal resources
- `RayConfig.for_production(ray_address, num_workers)` - Cluster config
- `RayConfig.for_testing()` - CI-friendly minimal config

3.2 RayBatchTestRunner (STUB)

Implemented as a **stub** with full structure but placeholder logic:

```python
@ray.remote
class VariantExecutionActor:
    """Reuses DB connections across executions."""

    def __init__(self, agentgit_db_url, wtb_db_url):
        # TODO: Initialize AgentGitStateAdapter, UoW
        pass

    def execute_variant(self, workflow, combination, initial_state) -> Dict:
        # TODO: Full execution with ExecutionController
        pass

class RayBatchTestRunner(IBatchTestRunner):
    """ActorPool-based distributed execution."""

    def _ensure_actor_pool(self, num_workers):
        # Creates VariantExecutionActor pool
        pass

    def run_batch_test(self, batch_test):
        # Submit to ActorPool, collect results
        pass
```

Status: STUB - Structure complete, requires Ray cluster for full testing

3.3 Environment Providers

Provider	Purpose	Status
`InProcessEnvironmentProvider`	No isolation (dev/test)	✅ DONE
`RayEnvironmentProvider`	Ray runtime_env	✅ DONE
`GrpcEnvironmentProvider`	External gRPC service	STUB

RayEnvironmentProvider Example:

```python
provider = RayEnvironmentProvider(base_env={"pip": ["base-package"]})

env = provider.create_environment("variant-1", {
    "pip": ["numpy==1.24.0"],
    "env_vars": {"MODEL_VERSION": "v1"},
})
# Returns: {"pip": ["base-package", "numpy==1.24.0"], "env_vars": {...}}
```

4. Database Migrations

4.1 SQLite Migration (002_batch_tests.sql)

```
CREATE TABLE IF NOT EXISTS wtb_batch_test_results (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    batch_test_id VARCHAR(64) NOT NULL,
    execution_id VARCHAR(64) NOT NULL,
    combination_name VARCHAR(255) NOT NULL,
    variant_config TEXT NOT NULL,  -- JSON
    success INTEGER NOT NULL DEFAULT 0,
    metrics TEXT,  -- JSON
    overall_score REAL DEFAULT 0.0,
    duration_ms INTEGER DEFAULT 0,
    worker_id VARCHAR(255),
    created_at TEXT NOT NULL DEFAULT (datetime('now')),
    FOREIGN KEY (batch_test_id) REFERENCES wtb_batch_tests(id)
);

-- Indexes
CREATE INDEX idx_batch_test_results_batch ON wtb_batch_test_results(batch_test_id);
CREATE INDEX idx_batch_test_results_score ON wtb_batch_test_results(batch_test_id, overall_score DESC
```

4.2 PostgreSQL Migration (003_postgresql_production.sql)

Key additions for production:

- **JSONB columns** for `variant_config` and `metrics` (efficient querying)
- **BRIN indexes** for time-series queries (smaller than B-Tree for ordered data)
- **GIN indexes** for JSONB fields
- **Autovacuum tuning** for high-insert tables
- **Partitioning hints** for >10M records/month

5. Files Summary

Created (16 files)

Path	Description
`wtb/infrastructure/events/__init__.py`	Module exports
`wtb/infrastructure/events/wtb_event_bus.py`	Thread-safe event bus
`wtb/infrastructure/events/wtb_audit_trail.py`	Audit trail + listener
`wtb/domain/interfaces/batch_runner.py`	IBatchTestRunner, IEnvironmentProvider
`wtb/application/services/batch_test_runner.py`	ThreadPoolBatchTestRunner
`wtb/application/services/ray_batch_runner.py`	RayBatchTestRunner (stub)
`wtb/infrastructure/environment/__init__.py`	Module exports
`wtb/infrastructure/environment/providers.py`	Environment providers
`wtb/infrastructure/database/migrations/__init__.py`	Module marker
`wtb/infrastructure/database/migrations/002_batch_tests.sql`	SQLite migration
`wtb/infrastructure/database/migrations/003_postgresql_production.sql`	PostgreSQL migration
`wtb/infrastructure/database/repositories/audit_repository.py`	SQLAlchemy audit log repository
`tests/test_wtb/test_event_bus.py`	20 tests
`tests/test_wtb/test_audit_trail.py`	24 tests
`tests/test_wtb/test_batch_runner.py`	20 tests
`tests/test_wtb/test_audit_repository.py`	3 tests

Modified (6 files)

Path	Changes
`wtb/config.py`	Added RayConfig, ray_enabled, sqlite_wal_mode
`wtb/application/factories.py`	Added BatchTestRunnerFactory

Path	Changes
`wtb/domain/interfaces/__init__.py`	Export batch runner interfaces + IAuditLogRepository
`wtb/domain/interfaces/repositories.py`	Added IAuditLogRepository (fixed duplicate)
`wtb/infrastructure/__init__.py`	Export events + environment modules
`docs/Project_Init/PROGRESS_TRACKER.md`	Updated completion status

6. Next Steps (Remaining P0/P1)

Priority	Task	Status
P0	Complete RayBatchTestRunner with ExecutionController	TODO
P0	Implement ParityChecker (ThreadPool→Ray validation)	TODO
P0	Configure PgBouncer for production	TODO
P1	Add Prometheus metrics export	TODO
P1	Implement EvaluationEngine	TODO

7. Key Decisions Made Today

1. **Standalone Event Bus** - Implemented without inheriting from AgentGit EventBus to avoid import complexity; uses optional bridge pattern instead.
2. **RLock over Lock** - Allows handlers to publish events recursively without deadlock.
3. **Factory-based Isolation** - ThreadPoolBatchTestRunner creates isolated UoW/StateAdapter per thread via factory functions, not shared instances.
4. **Ray as Optional Dependency** - RayBatchTestRunner gracefully handles `ImportError` when Ray not installed.
5. **Dual Migration Strategy** - Separate SQLite and PostgreSQL migrations to leverage database-specific features (JSONB, BRIN indexes).

Test Results

```
======= 275 passed, 1 skipped, 10 warnings in 5.12s =======

New test files:
- test_event_bus.py:     20 tests (thread-safety, subscriptions, history)
- test_audit_trail.py:   24 tests (entries, recording, serialization)
- test_batch_runner.py:  20 tests (ThreadPool, factory, providers)
- test_audit_repository.py: 3 tests (SQLAlchemy/InMemory audit log repository)
```

8. Code Review & Fixes (Evening Session)

8.1 Issues Identified and Fixed

Issue	Location	Fix
Duplicate `IAuditLogRepository`	`wtb/domain/interfaces/repositories.py`	Removed duplicate class definition (lines 327-352)
Missing `IAuditLogRepository` export	`wtb/domain/interfaces/__init__.py`	Added to imports and `__all__` exports

8.2 Code Quality Assessment

Component	Quality	Notes
WTBEventBus	✅ Excellent	RLock for thread-safety, bounded deque, proper exception isolation
WTBAuditTrail	✅ Excellent	Clean event mapping, serialization support, flush capability
ThreadPoolBatchTestRunner	✅ Excellent	Factory-based per-thread isolation pattern prevents state corruption
RayBatchTestRunner	✅ Good (Stub)	Well-structured ActorPool design, clear TODO markers
Environment Providers	✅ Good	Clean abstraction, proper base_env merging

Component	Quality	Notes
IAuditLogRepository	✅ Good	Consistent with repository pattern, supports batch append

8.3 Design Strengths Observed

1. **Thread Safety in WTBEventBus**
   - RLock allows nested publishes without deadlock
   - Handler list copied before iteration to avoid concurrent modification
   - Exception in one handler doesn't block others
2. **Factory-based Isolation in ThreadPoolBatchTestRunner**
   - Each thread creates isolated `UnitOfWork` and `StateAdapter` instances
   - No shared mutable state between threads
   - Clean separation of concerns
3. **Graceful Ray Degradation**
   - `RAY_AVAILABLE` flag for conditional Ray functionality
   - `is_available()` static method for runtime checks
   - Clear error messages when Ray not installed

8.4 Warnings to Address (Low Priority)

Warning	Location	Recommendation
`datetime.utcnow()` deprecation	SQLAlchemy ORM defaults	Use `datetime.now(datetime.UTC)` in future
`TestWorkflow` collection warning	pytest	Rename domain model or add `pytest_ignore_collect`

Report updated: 2025-01-10
Author: AI Architect Assistant