Trading Software Factory – Validated Architecture & Plan (v2)

Date: 2025-11-27 • Author: ChatGPT (with factory.ai & Claude reviews)

Table of Contents

1) Executive Summary

Dieses Dokument konsolidiert die geprüfte und aktualisierte Architektur für eine modulare Trading-Software-Factory, die den gesamten Lebenszyklus abdeckt: Research → Backtest → Paper Trading → Live Trading.

- Starke Verträge (Contracts) zwischen den Schichten: Typen, Validatoren, SLAs.
- Kanonische Schemata für Daten, Signale und Orders.
- Zentralisierte Risk- und Idempotency-Kontrollen.
- Schmale, testbare Engine-APIs und klare Artefakte pro Run.
- Dashboard-getriebene Iteration und automatisierte Promotion anhand von Schwellwerten.

2) Validation of External Reviews (factory.ai & Claude)

factory.ai — übernommene Empfehlungen

- Contracts in Code erzwingen (Pydantic für Config/Manifeste, optional Pandera für DataFrames).
- Kanonisches Order-Schema + deterministische Idempotency-Keys (run_id, strategy, version, symbol, side, oco_id).
- Schmale Engine-API; ein Manifest-Schema; erweiterte Metriken; robuste Dashboard-Loader und Artefakt-Links.

Claude Sonnet — übernommene Empfehlungen

- Daten-SLAs (M5-Vollständigkeit, Latenz, NaNs, Duplikate, Session-Korrektheit).
- Formale Go/No-Go-Kriterien Paper→Live; Promotion via Policies automatisiert.
- Runbooks und Disaster-Recovery-Plan ergänzt.
- Baseline-Performance-SLOs dokumentiert und verfolgt.

3) Architecture Overview

- **Data Layer:** fetch → assemble → resample → validate. Sessions via Markt-Kalender, UTC-Index, OHLCV (UPPERCASE) als Normalform.
- **Strategy Layer:** Plugin-basierte Signale mit einheitlichem, versioniertem Schema; Two-Stage (Daily-Filter → Intraday-Trigger).
- **Order/Sizing:** Kanonisches Order-Schema, zentrale Sizing-Regeln (%-Equity & Risk-based), Tick-Rundung, Idempotency.
- **Backtesting:** First-Touch-Entry, SL/TP/EOD-Exits; schmale API; einheitliche Artefakte/Metriken.
- **Dashboard & Observability:** Runs, KPIs, Equity/Drawdown, Trades/Orders, Artefakt-Links, SLA-Badge.
- **Promotion Pipeline:** Automatisierte Gates (SLAs + Metrik-Schwellwerte + Dry-Runs).
- **Risk & Limits:** Zentrale Guards vor Broker-Send; Kill-Switch.
- **Runbooks & DR:** Incident-Response und Rollback-Playbooks.

4) Data Layer – Contracts, Validators, Resampling, SLAs

Contracts

- *DailyFrameSpec* & *IntradayFrameSpec* (OHLCV vorhanden; UTC tz-aware DatetimeIndex; monoton; keine Duplikate; keine NaNs in O/H/L/C; Volume darf 0 sein).

Validatoren

- `assert_valid_daily(df)` und `assert_valid_intraday(df, cal, tz)` mit Session-Fences (Kalender) und DST-Handling.

Resampling

- `resample_m1_to_m5(df_m1, calendar, tz)` innerhalb der Session-Grenzen; saubere O/H/L/C/V; keine Cross-Session-Aggregation; DST-sicher.

SLAs

- m5_completeness ≥ 0.99; lateness_minutes ≤ 5; no_nan_ohlc; no_dupe_index. Verstöße stoppen Promotion.

5) Strategy Layer – Signal Schema & Two-Stage Contract

Signal-Schema

`["Symbol","long_entry","short_entry","sl_long","sl_short","tp_long","tp_short","setup","score","strategy","strategy_v`

Two-Stage Contract

- Stage 1 (Daily): begrenzte Kandidatenmenge (z. B. ≤ 50).
- Stage 2: Intraday nur für Stage-1-Symbole (erzwingen).

Konventionen

- OHLCV UPPERCASE; Index UTC; Symbol-Casing standardisiert; Sessions/DST via Kalender.

6) Order Export & Position Sizing – Canonical Schema, Idempotency, Invariants

Kanonisches Order-Schema

`order_id, idempotency_key, oco_id, run_id, symbol, side, qty, limit_price, stop_price, take_profit_price, valid_from, valid_to, session, time_in_force, source, strategy, strategy_version`

Idempotency (deterministisch)

`idempotency_key = sha1(f"{'{'}run_id|strategy|strategy_version|symbol|side|oco_id{'}'}")[:16]`

Sizing-Invarianten

- %-Equity: notional = equity * pos_pct/100; Modulo-Rundungsregel; Tick-Rundung; qty ≥ min_qty.
- Risk-based: risk_amount = equity * risk_pct/100; *stop_distance_ticks* gerundet; `qty = floor(risk_amount/(stop_distance_ticks*tick_value))`; Clip auf *max_pos_pct*.

7) Backtesting Engine – Narrow API, Manifest, Metrics

API

`simulate_insidebar_from_orders(orders_csv, data_path, tz, costs, initial_cash)`

Artefakte

`filled_orders.csv`, `trades.csv`, `equity_curve.csv/png`, `drawdown_curve.png`, `metrics.json`, `manifest.json`, `label.txt`

Manifest

```
{"
{"}"run_id","created_at","engine":"replay","mode":"insidebar_intraday","strategy":"insidebar","strategy_version":"1.0
{"}"}
```

Metriken

`initial_cash, final_cash, net_pnl, max_drawdown, num_trades, win_rate, avg_trade, pnl_per_trade, sharpe`

8) Dashboard & Observability – Drill-Downs, SLA Badge, Metrics

- Robuste Loader; Einheiten (EUR, Stk); Notional (qty*price); `label.txt` als Titel; Artefakt-Links; SLA-Badge aus `artifacts/quality/*.json`.
- JSON-Log-Schema; Prometheus-Starter: `orders_sent_total`, `fills_total`, `rejects_total`, `strategy_pnl{strategy=…}`, `data_fetch_latency_seconds`.

9) Promotion Pipeline (Experiment → Lab → Paper → Live)

Policy

- Schwellwerte: *min_num_trades*, *min_win_rate*, *max_drawdown*, *min_sharpe*, *data_sla_pass*.
- Gates: *data_quality_check* → *metrics_thresholds* → *dryrun_paper_broker_check*.

CLI

`cli_promote.py`; Make-Target `promote_if_green` automatisiert.

10) Central Risk & Limits – Guards & Kill Switch

- Guards: *MaxGrossExposure*, *PerSymbolMaxQty*, *MaxDailyLoss*, *SlippageSanity*, Markt-Halt-Proxy.
- Kill-Switch verpflichtend; gleiche Guards in Paper & Live.

11) Runbooks & Disaster Recovery (DR)

- `docs/runbooks/incident_response.md`, `docs/runbooks/promotion_checklist.md`.
- `docs/dr_plan.md` (RTO/RPO; Config-Rollback per `TR_MODE=paper|live`; Git-Tags; Backups, z. B. *restic*).

12) Performance Benchmarks (SLOs)

- Order-Latenz (Paper/Live): Median < 250 ms; p95 < 600 ms.
- Intraday-Signal-Pipeline: p95 < 200 ms pro Symbol-Update.
- Backtest-Durchsatz: Ziel > 5 Mio Bars/Min (Hardware-abhängig dokumentieren).
- Über Metriken tracken/alerten; optional in Promotion-Policy spiegeln.

13) Repository Layout & Make Targets

Neues/Erweitertes Layout

```
src/axiom_bt/
  contracts/{data_contracts.py, signal_schema.py, order_schema.py}
  validators/data_validators.py
  utils/resample.py
  risk/{sizing.py, guards.py}
  observability/log_schema.json
  cli_promote.py
configs/{policies/promotion_policy.yml, risk.yml, settings.yml}
docs/{runbooks/incident_response.md, runbooks/promotion_checklist.md, dr_plan.md}
artifacts/quality/
dashboards/run_dashboard.py (Links & SLA-Badge)
```

Make-Targets

- `make data:validate` – SLAs/Validatoren erzwingen
- `make bt:run CONFIG=…` – Backtest starten
- `make promote_if_green` – Policy-Promotion automatisieren
- `make risk:test` – Guards simulativ testen

14) Next Steps (Actionable Checklist)

1. Contracts & Validatoren implementieren (Pandera optional).
2. `cli_export_orders` erweitern: *idempotency_key* & Strategy-Metadaten (append-only CSV).
3. Engine-Metriken: *max_drawdown*, *avg_trade*, *pnl_per_trade* in `metrics.json`.
4. Dashboard: Artefakt-Links & SLA-Badge.
5. Policy & CLI: `promotion_policy.yml` + `cli_promote.py` + Make-Integration.
6. Risk-Guards im Broker-Adapter vor `send()` verdrahten; Konfig in `risk.yml`.
7. Runbooks & DR-Plan schreiben.

Bald darauf: Prometheus-Metriken/Grafana; Performance-Timer; Unit/Integration-Tests für Contracts, Validatoren, Guards, Promotion-Logic.

15) Appendix A – Key Learnings & Conventions

- OHLCV in UPPERCASE; Index in UTC; niemals lokale Zeit für Alignment.
- Two-Stage: Stage-1 Kandidaten deckeln; Stage-2 nutzt ausschließlich diese Symbole.
- Sessions & DST via Kalender; keine DIY-Hacks.
- Kanonische CSV-Schemata → Komponenten austauschbar.
- Deterministische Idempotency durchgängig für sichere Retries.
- Schmale Engine-APIs; reichhaltige Metriken aus einfachen Artefakten.
- Dashboard verlinkt Roh-Artefakte; zeigt Einheiten (EUR, Stk) & Notional.
- Sizing & Risk zentralisieren – kein Regel-Duplikat in CLIs/Adaptern.
- Promotion policy-getrieben & automatisiert; Human-Review bleibt.
- Observability klein starten; bei Bedarf skalieren.