**BEST-PRACTICE GUIDE · TRANSFORM**

# Phase 05 — Build

From specification to production-ready increment

In the Build phase, multi-agent systems implement the target solution based on the quality-assured specifications from Analyse and Design. Consultants steer the agents, assure quality through reviews and policies, and keep the delivery track's DORA metrics at elite level.

## Goal of the phase

The phase delivers production-ready, tested increments with full traceability from requirement to code — shipped in short cycles over a trunk-based, fully automated integration track.

## Process model

01 **Delivery setup & definition of done**
Set up the agentic delivery track (repositories, CI, policies, environments); agree the definition of done, coding and review standards with the client team.

02 **Specification-driven agent implementation**
Decompose epics into agent-ready work packages with precise acceptance criteria; multi-agent code generation against the Design phase's contracts and ADRs.

03 **Continuous integration & review**
Trunk-based development with small batches; every change passes automated checks and a human review of functionally critical paths.

04 **Quality & security gates**
Static analysis, dependency and secret scanning, architecture conformance checking against the target architecture; policy gates block non-conforming changes.

05 **Measurement & learning loop**
Continuous collection of DORA metrics and agent effectiveness; retrospectives to improve prompts, contexts and work package cuts.

## Methodology mix

| METHOD | PURPOSE | REFERENCE |
|---|---|---|
| **Trunk-based development** | Small, short-lived branches and daily integration; precondition for agentic delivery speed without merge conflicts. | trunkbaseddevelopment.com |
| **Continuous delivery** | Deployment pipeline with automated quality stages; every change is potentially releasable. | Humble & Farley (2010) |
| **Test-driven development** | Test-driven specification of functionally critical logic; acceptance tests as executable requirements for the agents. | Beck (2002) |
| **DORA metrics** | Deployment frequency, lead time, change failure rate and MTTR as empirically validated steering measures of the delivery track. | Forsgren, Humble & Kim (2018) |

| METHOD | PURPOSE | REFERENCE |
|---|---|---|
| **Clean code & SWEBOK** | Binding coding and maintainability standards; SWEBOK as the reference frame of software engineering disciplines. | IEEE CS (SWEBOK v4) |
| **ReqPOOL Agentic Engineering Playbook** | In-house standard for work package cutting, context engineering, agent reviews and human-agent division of labour. | ReqPOOL (internal); cf. Anthropic (2024) |

## Platform support

ReqPOOL Suite: **reqCoder** (AI software development) · **Estimation Manager** (Estimate · Scope · Govern)

| | |
|---|---|
| **reqCoder — agentic implementation (coming soon)** | EU-based AI develops the software directly from the AI coding intent — no client data used for training, complete audit trail; human engineers review, harden and integrate the generated code. |
| **Specification as Code as working basis** | The AI-generated implementation order from the Estimation Manager (screens, use cases with acceptance criteria, business objects, interfaces, permissions matrix) serves as the direct working basis of agentic development. |
| **Estimation Manager — scope steering during build** | Scope changes during implementation are captured as versioned change requests; scope drift is continuously measured against the baseline. |
| **Traceability via IDs** | Requirement IDs from specification and coding intent link work packages, code and test cases — a complete evidence trail for audits and quality gates. |

## Best-practice checklist

- [ ] Cut work packages along the use cases of the AI coding intent — clear context, executable acceptance criteria, limited blast radius.
- [ ] Formulate acceptance criteria as tests before generation; the AI implements against executable requirements.
- [ ] Focus human reviews on functionally critical paths and architecture boundaries; never integrate generated code unchecked.
- [ ] Capture every scope change immediately as a change request in the Estimation Manager — scope drift is measured against the baseline (gate: code complete).
- [ ] Review DORA metrics weekly in the team; treat regressions as impediments, not as reporting figures.
- [ ] Version prompts and context packages and improve them in retrospectives — agent effectiveness is an engineering artifact.

## Artifacts & outcomes

— Production-ready, tested increments per release
— Full traceability requirement → code → test
— Automated integration and quality track
— Migrated and modernized legacy components
— DORA metric baseline and delivery report

## Quality gate criteria

- [ ] All increments meet the definition of done
- [ ] Policy gates passed without open exceptions
- [ ] Traceability chain proven without gaps
- [ ] DORA metrics within the agreed target corridor

## Scientific deep dives

**Forsgren, Humble & Kim: Accelerate (2018)** — Empirical foundation of DORA metrics and high-performance delivery.
https://itrevolution.com/product/accelerate/

**DORA research program** — Ongoing research on software delivery performance incl. AI impact.
https://dora.dev/

**Humble & Farley: Continuous Delivery (2010)** — Standard reference on deployment pipelines and release automation.
https://continuousdelivery.com/

**Beck: Test-Driven Development by Example (2002)** — Foundational work on test-driven development.
https://www.informit.com/store/test-driven-development-by-example-9780321146533

**IEEE CS: SWEBOK Guide v4** — Body of knowledge of software engineering.
https://www.computer.org/education/bodies-of-knowledge/software-engineering

**Anthropic: Building Effective Agents (2024)** — Design patterns for reliable agentic systems.
https://www.anthropic.com/research/building-effective-agents