Hingepoint.

THE FRAMEWORK GUIDE

# The Hingepoint Guide

Hingepoint is a framework for professional AI-assisted software engineering, combining human judgment with AI execution in a structured, event-driven development process.

Version · 1.0 · First Publication · August 2026

Authors · Anastasia Galani & Ingo Rübe

## Contents

# 1 Purpose of this Guide

This Guide defines the Hingepoint Framework.

It describes the framework's principles, concepts, phases, artifacts, verification and validation activities, and organizational model. It explains why the framework exists, what it defines, and how its elements relate to one another. It intentionally does not prescribe specific tools, technologies, or implementation practices.

This Guide is the normative reference for the Hingepoint Framework. Books, companion guides, training materials, templates, and machine-readable work instructions build upon this Guide without changing its meaning.

# 2 Definition

## The Hingepoint Framework

Hingepoint is a framework for professional AI-assisted software engineering.

It combines human judgment with AI execution in a structured, event-driven development process. Humans define intent, shape solutions, make decisions, and remain accountable. AI accelerates implementation within these boundaries.

## Hingepoints

A Hingepoint is a planned, event-driven synchronization point at which automated execution pauses until a required decision, dependency, approval, or external condition has been resolved.

> A Hingepoint separates autonomous execution from conscious decision-making. It synchronizes people, AI systems, and organizational dependencies before development continues.

Every Hingepoint exists for one purpose: to preserve quality, accountability, and organizational flow.

# 3 Foundation

## Why Frameworks Evolve

Software development frameworks emerge in response to the dominant constraint of their time. They do not replace previous frameworks because those frameworks became wrong. They evolve because the primary bottleneck changes.

Each framework reflects the economics of software development at the time it emerged.

## Waterfall

When software was expensive to change, planning and documentation became the primary optimization targets. Waterfall optimized the cost of change.

## Scrum

As software became easier to modify, software creation itself became the primary constraint. Scrum optimized collaboration, shortened feedback cycles, and reduced waiting time within development teams.

## Hingepoint

Generative AI changes the economics of software development once again.

> The primary organizational bottleneck has shifted from creating software to organizing software development.

Today, project success increasingly depends on:

- precise specifications,
- effective coordination,
- governance,
- security,
- organizational synchronization,
- and timely decisions.

Hingepoint addresses these organizational constraints by replacing calendar-driven synchronization with event-driven synchronization. Development continues until a decision,

dependency, approval, or external condition requires human intervention. Once resolved, execution continues until the next Hingepoint.

## Framework Evolution

Frameworks are not ideologies. Each framework optimizes the dominant bottleneck of its time. When the bottleneck changes, the optimal framework changes with it.

Hingepoint does not replace Waterfall or Scrum. It builds upon their contributions while addressing a different organizational reality.

The question is not: Which framework is better?

The question is: Which bottleneck dominates software development today?

# 4 Principles

## Explicit over implicit

Important decisions are made explicit. Assumptions, specifications, and changes are documented before execution continues.

## Events over calendars

Development progresses through events rather than predefined timeboxes. Synchronization happens when it is needed — not because the calendar requires it.

## Ownership over speed

Software is not progress if no individual can accept accountability for its outcome, regardless of how quickly it was produced.

## Accountability follows the context boundary

Human accountability ends where human understanding ends. No one can genuinely accept accountability beyond the context they fully understand.

## Governance is built in

Governance and security requirements are specified before implementation begins and validated continuously while the solution evolves — with validation becoming increasingly automated over time.

## Tool agnostic

Hingepoint defines principles, not products. It does not prescribe AI models, programming languages, development tools, vendors, or jurisdictions.

# 5 Human and AI: Working Together

Generative AI changes software development fundamentally.

Technically, AI is a tool. Within the Hingepoint Framework, it is treated as a member of the development team. This perspective is fundamental: it defines how work is organized, how accountability is assigned, and how humans and AI collaborate throughout the development process.

> AI is neither a replacement for developers nor a traditional development tool. It is a new type of teammate with an exceptional profile. Its strengths and its limitations must both be reflected in the development process.

## An Exceptional Team Member

AI is

- a capable partner for specification,
- an extremely fast software developer,
- a rigorous reviewer,
- a tireless documenter, and
- available whenever it is needed.

At the same time, AI

- makes silent false assumptions,
- can be confidently wrong,
- loses coherence in large contexts,
- optimizes what was said rather than what was intended, and
- cannot accept responsibility for its decisions.

A process that recognizes only AI's strengths will eventually fail because of its weaknesses. Hingepoint is designed around both.

## MAY · KNOW · HOW

Successful collaboration between humans and AI depends on three complementary rule classes. Together they define the operational environment in which AI works.

### MAY

MAY defines what AI is allowed to do. Humans establish these boundaries: which actions AI may perform autonomously and which actions require human authorization. This includes permissions, available tools, accessible repositories, operational limits, and governance constraints.

These rules are deterministic. They are enforced — not requested. They define the operational boundaries within which AI may act autonomously.

### KNOW

KNOW defines what AI knows. Humans provide the authoritative context: the Intent, Specifications, Plans, approved repositories, governance rules, architectural decisions, and every other artifact required for the task.

This context is curated and continuously refined through the collaboration between humans and AI. The authoritative context, however, always remains under human ownership.

### HOW

HOW defines how humans and AI work together. It describes the shared way of working. The Hingepoint Framework itself belongs to this rule class.

Humans learn the framework through training and practice. AI requires the same onboarding. Without these shared working instructions, AI falls back to its default behavior:

- generating immediately,
- making implicit assumptions,
- and optimizing the request instead of the actual intent.

The technical implementation of MAY, KNOW, and HOW — configuration files, context files, machine-readable work instructions, or similar mechanisms — is tool-specific and intentionally outside the scope of this Guide.

MAY, KNOW, and HOW are not static. All three rule classes evolve as the collaboration matures: boundaries are adjusted, context improves, and the shared way of working is refined. How an organization evolves these rules deliberately — collecting observations, deciding on changes, and distributing them — is defined in Chapter 12.

## Result = f(Context)

> The quality of the outcome is a function of the quality of the context. The primary lever is not better prompting. It is better context.

Hallucinations are almost always context failures. Specifications and implementation plans are curated context. For this reason, think first, generate second is not an argument for bureaucracy. It is an argument for efficiency.

Two different contexts exist: the machine context and the human context.

The machine context consists of specifications, plans, rules, source code, and all information required by AI to perform its work. The human context is the mental model developed during Shaping and Plan Calibration.

The machine context determines what AI can produce. The human context determines whether someone can understand, explain, review, and ultimately accept accountability for the result. Successful software development requires both.

Human decisions are also a function of context. The goal is not to maximize context, but to provide the right context to the right participant at the right time. For both humans and AI, too much context is as harmful as too little.

> The human context window is the scarcest resource in the Hingepoint Framework. Work is intentionally organized around this boundary. The framework ensures that humans and AI receive the appropriate context throughout the development process.

# 6 One Flow

The Hingepoint Framework defines a single development flow for all software work. Every Work Package — the unit of work defined in Chapter 10 — follows the same flow, regardless of its size, complexity, or technical domain. The phases remain the same; only their depth and effort scale with the complexity of the work.

The flow always begins with an Intent. From there, development consists of two connected movements: the Ramp and the Delivery Cycle.

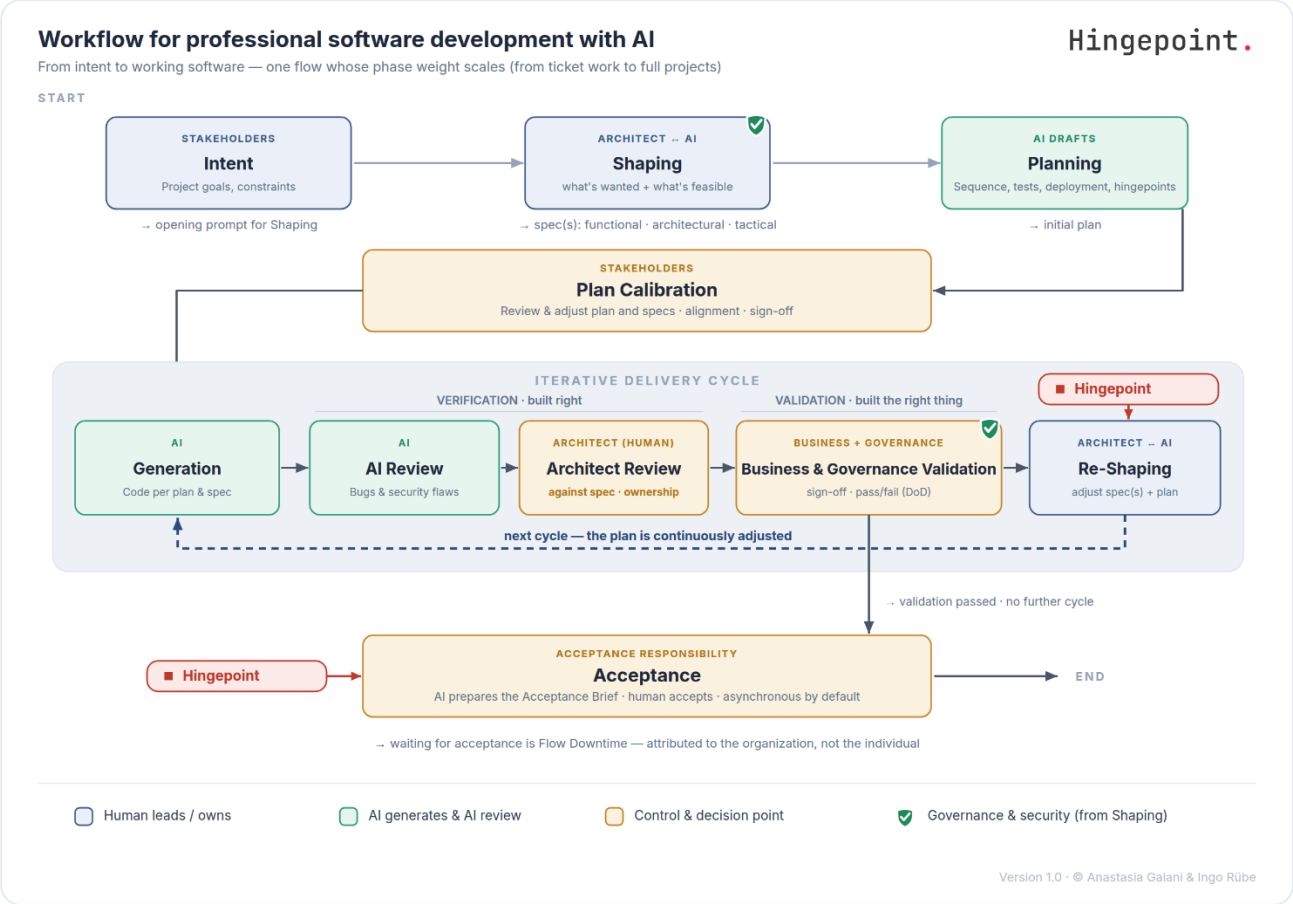


Figure 1 · The Hingepoint Flow: the Ramp prepares, the Delivery Cycle builds; Hingepoints synchronize.

## The Ramp

The Ramp prepares implementation. Across four phases: Intent, Shaping, Planning, and Plan Calibration. It transforms a business objective into a shared understanding of how the work will be implemented. Chapter 7 describes the Ramp in detail.

## The Delivery Cycle

The Delivery Cycle transforms this preparation into a validated software solution. Its four phases: Generation, Verification, Validation, and Re-Shaping, repeat until the required outcome has been achieved. Each iteration improves the implementation while preserving human accountability. Chapter 9 describes the Delivery Cycle in detail.

## Hingepoints

Hingepoints connect planning with execution. They are scheduled for the point at which a decision, dependency, approval, or external delivery becomes relevant to the implementation. When that point is reached, the affected work pauses until the Hingepoint has been resolved; the Delivery Cycle then continues. Chapter 8 defines Hingepoints in detail.

Verification and Validation may trigger Re-Shaping, which updates the Specifications and the Plan before the next Delivery Cycle begins.

## Different Initial Conditions

The Hingepoint Flow remains unchanged. Only the initial project context differs.

### Greenfield

Greenfield development starts without an existing implementation. The required technical context is created during the Ramp: architecture, conventions, governance, and implementation strategy are established before implementation begins.

### Brownfield

Brownfield development starts with an existing software system. Much of the required context already exists, but often only within the implementation itself.

AI reconstructs this context by analyzing the existing system, its architecture, interfaces, and source code. Humans review, correct, approve, and preserve this reconstructed knowledge. Reconstructed context is never assumed to be correct. It becomes part of the project context only after human review.

> Regardless of the initial conditions, every Work Package follows the same Hingepoint Flow.

# 7 The Ramp

The Ramp takes place once for each Work Package. Its purpose is to create sufficient information density before implementation begins: it aligns the human and the machine context and establishes the shared understanding required for accountable software engineering.

The Ramp consists of four phases:

- Intent
- Shaping
- Planning
- Plan Calibration

It produces:

- a validated Intent,
- one or more Specifications,
- an implementation Plan,
- and a shared understanding of the implementation.

Its depth scales with the work. For a small ticket, the artifacts may be brief. For a complex initiative, the Ramp may require several sessions and multiple Specifications. None of its phases is omitted. No implementation begins before the Ramp has been completed.

## 7.1 Intent

The Intent defines the desired business outcome, the problem to be addressed, the relevant stakeholders, and the criteria for success.

The Intent is written in plain language. Prose is sufficient. It also serves as the opening input for Shaping.

The Intent is the only artifact written entirely by humans. Its purpose is to force agreement before implementation begins. Once approved, the Intent remains unchanged.

Gaps are normal. A gap that becomes relevant does not permit an implicit assumption. It becomes a Hingepoint.

The artifact matters more than the meeting format. There is one approved Intent. Depending on the size and context of the work, it may result from a workshop, several conversations, or an existing ticket.

## 7.2 Shaping

Shaping is a directed dialogue between a human architect and AI. Business stakeholders remain available for questions and approval.

Its guiding question is: What is required from a business perspective, and what is technically feasible?

Shaping produces one or more Specifications. Their number, format, and file structure are not prescribed. Together, they cover four types of information.

### Functional

What the solution is expected to achieve.

### Architectural

How the solution fits into its technical environment. In Brownfield work, this means fitting into the existing architecture. In Greenfield work, it includes establishing the architecture itself.

### Tactical

How the solution will be implemented.

The tactical Specification provides the mental model against which the implementation will later be reviewed. AI presents viable approaches and their trade-offs. The human architect chooses the approach.

Without a tactical Specification, a reviewer must reconstruct the implementation approach from completed code, and a fundamentally wrong approach is discovered late and at high cost. Hingepoint brings this decision forward: the implementation approach is examined before any code exists. Verification can later focus on whether the agreed approach was implemented correctly.

### Governance and Security

Which properties the resulting solution must satisfy. This may include access control, sensitive data handling, auditability, security requirements, and regulatory constraints.

“Not required” is an explicit decision, not an omission.

Specifications are as complete as reasonably possible at the time of Shaping. They must be sufficient for Planning. They are not assumed to be final and may evolve through Re-Shaping.

## 7.3 Planning

During Planning, AI derives an implementation Plan from the approved Specifications.

The Plan defines:

- implementation order,
- tests and test data,
- mocks and probes,
- deployment activities,
- dependencies,
- milestones, and
- planned Hingepoints.

A Specification and a Plan serve different purposes. The Specification defines what will be built, how it fits into its environment, and which properties it must satisfy. The Plan defines in which order the work will proceed, which means will be used, and where human decisions, external deliveries, or approvals will become necessary.

Decision Hingepoints are placed as early as possible. Early feedback against a mock or prototype is more valuable than late feedback against a completed solution.

Risky assumptions are planned as explicit validation steps. Where an external format, unknown data source, interface, or technical constraint is uncertain, the Plan includes a probe, spike, fixture, or test-data generator before work that depends on the assumption begins.

Planning produces the initial Hingepoints. Additional Hingepoints may emerge during the Delivery Cycle.

## 7.4 Plan Calibration

Plan Calibration completes the Ramp.

It is the mandatory human examination and approval of the implementation Plan. The human reads the Plan in detail, challenges its assumptions, confirms its sequence, and verifies that it reflects the approved Specifications.

Plan Calibration examines two relationships: Human and AI – Plan and Specifications.

### Human ↔ AI

Do the human and AI share the same understanding of the intended implementation?

### Plan ↔ Specifications

Does the Plan correctly implement the Specifications, or has Planning revealed that one or more Specifications must be revised?

Plan Calibration is the first point at which implementation insight may be written back into the Specifications — before the first Generation phase. Misunderstandings are resolved before they become code.

No implementation begins before Plan Calibration has been completed and the Plan has been explicitly approved. For small Work Packages, Plan Calibration may be brief. It is never omitted.

# 8 Hingepoints: Runtime Control

A Hingepoint is a defined point in the Plan at which further progress depends on a decision, delivery, approval, resource condition, or technical clarification.

Hingepoints are the control mechanism of the framework. They connect:

- machine execution with human judgment, and
- the work of an individual Human-AI work unit with the wider organization.

Hingepoints are initially identified during Planning. Additional Hingepoints may emerge during the Delivery Cycle when new information becomes available. Humans and AI may both identify them. AI often detects Dependency and Technical Hingepoints first; it must therefore be able to register them and stop at the relevant point instead of guessing.

Identifying a Hingepoint does not necessarily stop current work. A Hingepoint is placed at the point in the Plan at which its condition becomes relevant. Work continues until that point is reached. If the required decision, delivery, or approval is not available by then, the affected work is blocked.

A Hingepoint prevents AI from filling a relevant gap with an implicit assumption.

## 8.1 The Five Types

### Decision

A business, product, architectural, or conceptual decision is required. Example: a mock user interface is ready and requires business approval before implementation continues.

### Dependency

An external delivery or contribution is required. Example: an interface, driver, data set, or component must be supplied by another person or team.

### Compliance

A governance, security, or regulatory question must be resolved before the affected work proceeds.

### Resource

Cost, capacity, or another required resource deviates materially from the Plan.

### Technical

A technical assumption does not hold. Example: an interface behaves differently from the assumption on which the Plan was based.

Decision Hingepoints are common in Greenfield work because major choices remain open. Dependency and Technical Hingepoints are common in Brownfield work because integration and assumptions about existing systems dominate.

## 8.2 Lifecycle and Register

Each Hingepoint follows a visible lifecycle:

```
identified → scheduled → reached → blocked (if unresolved) → resolved → unblocked
```

Every Hingepoint is recorded in a register. Its name and format are not prescribed. The register records at least:

- the condition that must be satisfied,
- the point in the Plan at which it becomes relevant,
- the person or organizational unit from which input is required,
- its current status, and
- the timestamps required for Flow Uptime (Chapter 11).

A Hingepoint is resolved only when its actual condition has been satisfied. Where approval is required, a preference or informal indication is not sufficient: approval must refer to the actual artifact or decision under review.

## 8.3 Three Ways to Resolve a Hingepoint

### Delivery

The required decision, artifact, service, approval, or technical contribution is provided.

### Workaround

A temporary alternative removes the dependency. Mocks, test-data generators, probes, and replacement interfaces can often preserve flow at low cost. AI may propose a workaround. It does not implement one without human approval.

### Adaptation

New information changes the agreed solution or implementation approach. The affected Specifications and the Plan are updated through Re-Shaping.

If a decision requires an artifact e.g. a user-interface mock, a technical spike, a data sample, or a probe, the artifact is produced and reviewed before the Hingepoint is considered resolved.

## 8.4 The Acceptance Hingepoint

Every Plan ends with a final Decision Hingepoint: Acceptance. It is placed during Planning, like every other planned Hingepoint. Its condition: the person holding acceptance responsibility for the Work Package — typically the person who defined it — confirms that the result satisfies the Intent.

Acceptance belongs to the processing of a Work Package, not to organizational structure. It closes the gap between passed Validation and the explicit confirmation that the result has been seen, valued, and accepted by the person responsible for the business outcome.

Acceptance is resolved asynchronously by default. When the Acceptance Hingepoint is reached, AI prepares an Acceptance Brief: a compact summary of the Intent, the result, the Verification and Validation evidence, and any open points — where useful, accompanied by a short demonstration recording. The accepting person reviews the Brief and resolves the Hingepoint, or raises questions, which lead to Re-Shaping or to a new Work Package. A synchronous demonstration takes place on demand — not on a schedule.

An unresolved Acceptance Hingepoint is visible in the Hingepoint Register and is therefore addressed through Coordination (Section 10.2). No separate escalation path is required.

Waiting for acceptance is Flow Downtime (Section 11) of the Work Package. Slow acceptance becomes measurable and is attributed to the organization — not to the individual. The framework therefore prescribes no response deadline. Organizations introducing Hingepoint are, however, advised to agree on an initial response-time convention until asynchronous acceptance has become routine. Once Flow Uptime shows that acceptance is resolved reliably, the convention is no longer needed.

> Acceptance is event-driven: it happens because work is finished. Not because a meeting was scheduled.

# 9 The Delivery Cycle: Build Under Control

The Delivery Cycle transforms the approved Plan into validated software. It repeats until the intended outcome and the organization's quality criteria have been met.

The Delivery Cycle consists of four phases:

- Generation
- Verification
- Validation
- Re-Shaping

Hingepoints control progression through the cycle. When an unresolved Hingepoint is reached, the affected work pauses. Other independent work may continue.

## 9.1 Generation

During Generation, AI implements the approved Plan and the tactical Specification.

AI may make technical decisions that remain within the agreed implementation approach and the boundaries defined by MAY, KNOW, and HOW. It does not make business, governance, security, or scope decisions autonomously. If such a question arises, a Hingepoint has been reached or a new Hingepoint must be registered.

Nothing important is silently changed or "improved".

## 9.2 Verification — Was It Built Right?

Verification determines whether the implementation correctly follows the approved functional, architectural, and tactical Specifications. It proceeds in three stages and in a fixed order.

### Self-Test

AI creates and executes tests against the implementation. This establishes that the resulting software is executable and testable and that its defined functional behavior can be demonstrated.

### Independent AI Review

A second, independent AI reviews the implementation for defects, inefficiencies, security weaknesses, and missing tests. It may create and execute additional tests.

A human examines the findings and forwards the relevant ones to the implementing AI. Each forwarded finding is either resolved or explicitly justified.

Independent AI review broadens the inspection and identifies issues a human reviewer might otherwise miss.

### Architect Review

A human architect reviews the implementation against the architectural and tactical Specifications. Because the architect approved the implementation approach during Shaping and Plan Calibration, the relevant mental model already exists.

The review asks:

- Was the agreed solution implemented?
- Was it implemented in the agreed way?

Human accountability for the implementation is established through this review and its explicit approval.

AI generating and AI reviewing without informed human review is not sufficient. In that case, no human can credibly accept accountability for the result.

## 9.3 Validation — Was the Right Solution Built?

Validation determines whether the resulting solution satisfies the approved functional, governance, and security expectations and remains aligned with the Intent.

It asks:

- Does the solution achieve the intended business outcome?
- Does it satisfy the Governance and Security Specification?
- Is it acceptable for its intended use?

AI actively identifies:

- contradictions between the result and the Governance and Security Specification, and
- new risks introduced by the implementation that are not yet covered by the Specification.

Validation has three possible outcomes:

- Passed
- Gaps
- Skipped, with explicit human confirmation

Gaps become Hingepoints and are addressed in a subsequent cycle. The organization defines the pass/fail criteria for Verification and Validation when it adopts the framework.

Validation is not a formality. Governance is defined as a Specification during Shaping and validated throughout the Delivery Cycle, with validation becoming increasingly automated over time.

## 9.4 Re-Shaping — The Adaptive Core

Re-Shaping incorporates implementation insight into the agreed context. Initial assumptions are not treated as permanent.

If an implementation approach fails, the relevant tactical or architectural Specification changes. If new business knowledge emerges, the functional Specification changes. If new governance or security risks emerge, the Governance and Security Specification changes. The Plan normally changes with them.

All affected Specifications and the Plan are updated together. The changes and their reasons are presented to a human, explicitly approved, and recorded in a revision history. Nothing changes silently.

Re-Shaping follows Verification and Validation at an agreed milestone. It may also be anticipated in the Plan when a planned Hingepoint is expected to generate new information.

The updated Specifications and Plan become the authoritative context for the next Delivery Cycle. Re-Shaping keeps the agreement alive: the Specifications describe the current agreed solution — not an obsolete original assumption.

## 9.5 Completing a Work Package

When Verification and Validation have passed and no further cycle is required, the Work Package reaches its final planned Hingepoint: Acceptance (Section 8.4). AI prepares the Acceptance Brief; the person holding acceptance responsibility resolves the Hingepoint. With its resolution, the approved Plan has been completed and the Work Package ends.

# 10 Organizational Coordination

The Hingepoint Framework does not prescribe how an organization is structured. It defines how independently progressing Human-AI work units remain coordinated through their Hingepoints.

The smallest organizational unit in Hingepoint consists of:

- one human who accepts accountability for the work, and
- one or more AI agents operating within the boundaries defined by MAY, KNOW, and HOW.

Each unit works autonomously on a Work Package. Coordination (Section 10.2) connects these units without imposing a shared timebox, a specific hierarchy, or a new organizational model.

## 10.1 Work Packages

A Work Package is the unit of work processed through the Hingepoint Flow. It may consist of:

- a single ticket,
- several related tickets,
- a customer request,
- a feature,
- a module,
- a proof of concept,
- an MVP,
- or another coherent unit of software work.

> The size of a Work Package is limited by the human context boundary. It must be small enough for one human to understand the Intent, Specifications, Plan, implementation, and review evidence well enough to accept accountability for the result. Its size is therefore determined by understanding — not by duration.

A Work Package begins when a Human-AI work unit starts the Ramp. It ends when its approved Plan has been completed successfully. Successful completion includes the required Verification, Validation, and any resulting Re-Shaping defined by the Plan, and the resolution of the final Acceptance Hingepoint (Section 8.4).

The Hingepoint Framework does not prescribe how Work Packages are created. Organizations that already possess effective mechanisms for creating, prioritizing, and assigning work keep them; replacing working mechanisms would add disruption without improving software development. Organizations that lack such mechanisms — or that lose them when leaving a calendar-driven framework — are not left without an answer: Chapter 13 recommends a Reference Operating Model. The framework mandates how Work Packages are processed; for how the organization arranges the work around them, it offers a default — and mandates nothing.

## 10.2 Coordination

Coordination is the only meeting required by the Hingepoint Framework. Its purpose is to resolve Hingepoints and maintain organizational flow.

> Coordination is not a status meeting. It is a working meeting. Its outcome is action.

During Coordination, the organization:

- resolves Hingepoints whenever possible,
- initiates actions required to resolve unresolved Hingepoints,
- makes newly available Work Packages visible,
- assigns new Work Packages created by Hingepoints,
- and enables Human-AI work units to continue productive work.

Typical situations discussed during Coordination include:

- A Human-AI work unit has reached a Hingepoint and requires a decision, approval, delivery, or contribution.
- A Work Package has reached its Acceptance Hingepoint and awaits acceptance (Section 8.4).
- A Hingepoint cannot yet be resolved. The Human-AI work unit may therefore continue with another available Work Package whenever appropriate.
- A Work Package has been completed and another Work Package becomes available.
- A Hingepoint creates new work that becomes a separate Work Package.

Through Coordination, the Hingepoint of one Work Package may become the Work Package of another Human-AI work unit.

When an unresolved Hingepoint blocks further planned progress, the affected Work Package enters Flow Downtime: the waiting time defined in Chapter 11. It remains in Flow Downtime until the

Hingepoint is resolved, even if the Human-AI work unit continues productive work on another Work Package.

The frequency, duration, participants, and organizational form of Coordination are deliberately left to the organization. The framework defines the purpose of Coordination — not how organizations conduct it.

> Hingepoint makes calendar-driven, recurring meetings obsolete.

## 10.3 Organizational Freedom

Hingepoint does not define:

- how Work Packages are prioritized,
- how work is assigned,
- who leads Coordination,
- how decisions are made,
- how teams are structured,
- how many people participate,
- or whether the organization is self-organized or hierarchical.

These are organizational decisions. A functioning organization already has authority structures, escalation paths, product responsibilities, and governance mechanisms. Hingepoint uses them rather than replacing them.

The framework requires that decisions are made and dependencies are resolved. It does not prescribe who must make them.

## 10.4 No New Roles

> Hingepoint deliberately defines no organizational roles. There is no Hingepoint Master and no required new job title.

The framework defines responsibilities:

- someone writes and approves the Intent,
- someone performs and approves Shaping,
- someone performs Plan Calibration,
- someone reviews the implementation and accepts accountability,

- someone accepts the completed Work Package (Sections 8.4 and 13.2),
- someone provides business, governance, and security approval where required,
- and the organization resolves cross-Work-Package dependencies.

How these responsibilities are distributed depends on the organization. Existing roles may be used. Several responsibilities may be held by one person. One responsibility may also be distributed across several people.

The framework depends on explicit accountability — not on organizational titles.

## 10.5 From Coordination to Measurement

Every Work Package records its Hingepoints in its Hingepoint Register. Taken together, these records reveal how effectively the organization supports delivery. They show not how quickly individuals work, but how quickly the organization provides decisions, approvals, deliveries, and other required inputs.

This organizational property is measured through Flow Uptime. Chapter 11 defines the metric in detail.

# 11 Flow Uptime

Software development consists of two fundamentally different kinds of time.

The first is Productive Time: the time during which a Human-AI work unit actively progresses a Work Package according to its approved Plan.

The second is Flow Downtime: the time during which planned progress is prevented because a reached Hingepoint remains unresolved.

Flow Downtime belongs to the Work Package. It is independent of whether the Human-AI work unit continues productive work on another Work Package. Waiting is therefore a property of the organization — not of the individual.

The purpose of Flow Uptime is to make this organizational waiting visible.

## 11.1 Measuring Flow

For every Hingepoint, the Hingepoint Register records:

- when the Hingepoint became relevant,
- when Flow Downtime began,
- when the Hingepoint was resolved,
- and when productive work resumed.

These records are the basis for measuring organizational flow.

## 11.2 The Flow Uptime Metric

Flow Uptime expresses the proportion of Productive Time within the total elapsed time of a Work Package. It is expressed as a percentage.

```
FLOW UPTIME
Flow Uptime = Productive Time / ( Productive Time + Flow Downtime ) × 100%
```

> A high Flow Uptime indicates that the organization resolves Hingepoints efficiently. A low Flow Uptime indicates that Work Packages spend excessive time waiting for organizational dependencies.

Flow Uptime does not explain why work is waiting. It makes waiting visible.

## 11.3 Honest Measurement

Flow Uptime is valuable only when it reflects reality.

Hingepoints are recorded when they become relevant according to the approved Plan — not when it is convenient to report them. Delaying the registration of a reached Hingepoint artificially increases Flow Uptime and hides organizational constraints.

The purpose of the metric is not to demonstrate good performance. Its purpose is to reveal organizational waiting. Accurate measurement benefits the organization by identifying where organizational flow can be improved.

## 11.4 Organizational Learning

Flow Uptime is an organizational improvement metric. It is not a performance metric. It must not be used to evaluate individual developers or Human-AI work units.

Organizations improve Flow Uptime by improving their ability to resolve Hingepoints — not by asking individuals to work faster. Repeated Flow Downtime at similar Hingepoints often indicates opportunities to improve:

- organizational processes,
- decision-making,
- governance,
- communication,
- technical architecture,
- or dependency management.

> The objective is not to maximize utilization. The objective is to reduce unnecessary organizational waiting while preserving quality and accountability. Flow Uptime helps organizations identify where the next improvement should be made.

# 12 Evolution

MAY, KNOW, and HOW materialize as artifacts: configuration files, context files, and machine-readable work instructions. In a small team, they are adjusted informally the moment a weakness is noticed. In larger organizations, this breaks in three places: observations get lost, no one is visibly responsible for a rule change, and local adjustments create process drift that destroys the homogeneity larger organizations require.

Rule changes are changes to versioned artifacts with governance relevance — structurally, exactly what the Hingepoint Framework was built for. Evolution therefore introduces no new process.

> The Hingepoint Framework is applied to its own rules. Rule changes travel through the same flow as any other work.

## 12.1 Insights

An Insight is a recorded observation about MAY, KNOW, or HOW: a deficiency, a risk, or an opportunity for improvement. Anyone may register an Insight at any time — humans and AI alike. One sentence is sufficient. Insights are collected in an Insight Register, the counterpart of the Hingepoint Register.

An Insight blocks nothing. It is deliberately not a Hingepoint and not a sixth Hingepoint type: Hingepoints gate work, Insights do not. Where a rule acutely blocks or endangers work — a MAY boundary that is too narrow, a KNOW artifact that is wrong — the existing mechanism already applies: a Technical or Compliance Hingepoint.

Not every adjustment is an Insight. Rule maintenance within an owner's own authority and scope remains direct, informal work: an architect widening the context they own, a team refining its own working instructions. The project context itself already evolves through Shaping and Re-Shaping. The Insight Register exists for observations that cross a boundary — and only for those:

- the observer lacks the authority to change the rule — a developer observing a MAY limitation,
- the change affects the shared baseline and therefore other work units,
- the change touches governance or security,
- a local observation recurs and should become an organization-wide rule.

AI agents are instructed through HOW to register Insights instead of silently working around weaknesses — the counterpart of the rule that AI registers Hingepoints instead of guessing. AI thereby becomes a systematic source of process improvement.

Recurring Flow Downtime patterns (Section 11.4) are the second source of Insights. Organizational Learning thereby gains the mechanism that turns observation into action.

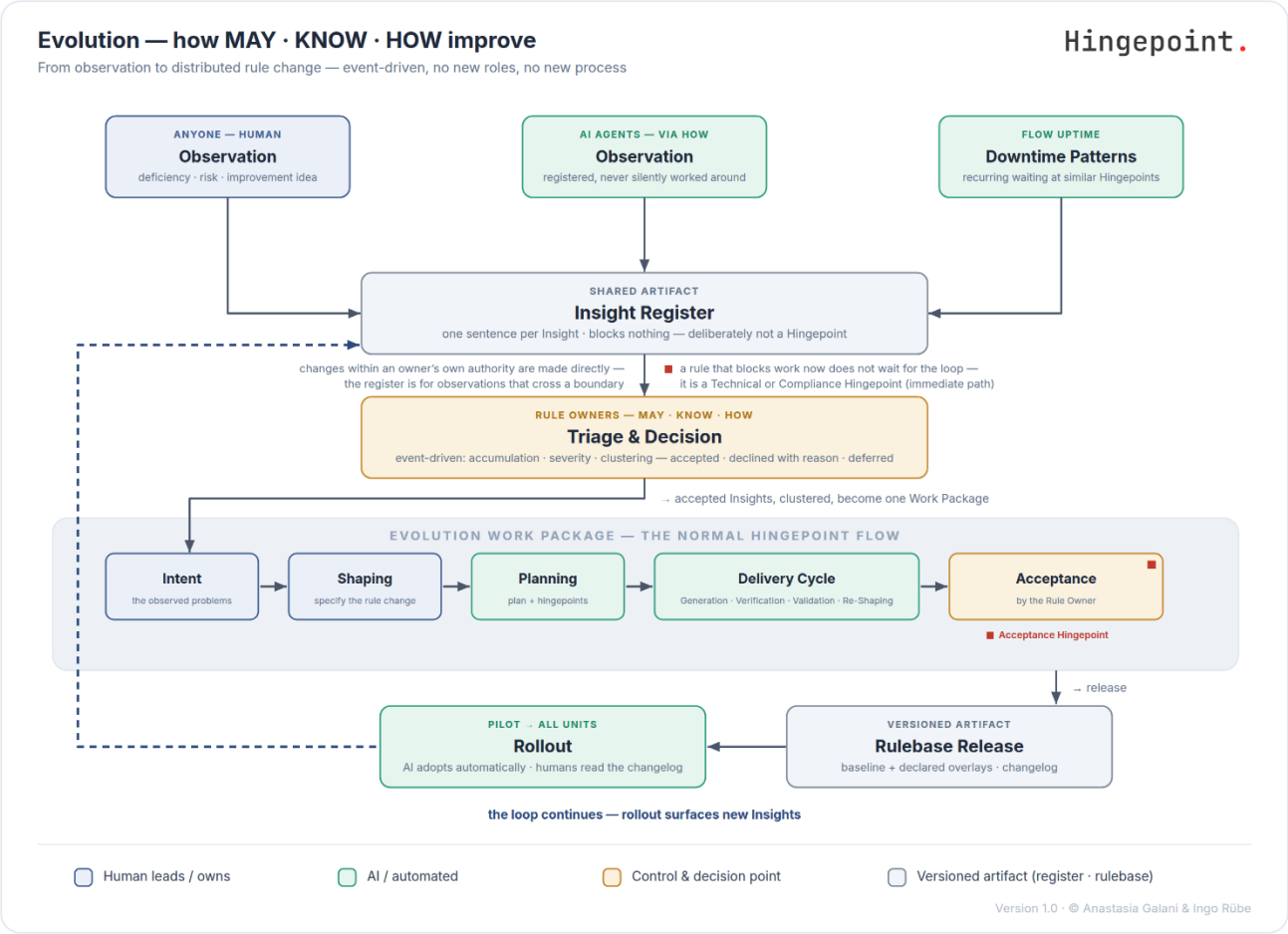


Figure 2 · The Hingepoint Evolution Flow: Onservations are registered and enter the normal Hingepoint Flow before Rollout.

## 12.2 Rule Ownership

Each rule class has an owner — a responsibility, not a role (Section 10.4):

- the MAY owner — typically security, platform, or governance,
- the KNOW owner — typically an architect,
- the HOW owner — typically engineering leadership or process ownership.

In small organizations, one person may own all three rule classes.

Owners triage the Insight Register asynchronously and event-driven — on accumulation, severity, or thematic clustering. Coordination serves only as a visibility backstop. Every decision is explicit:

accepted, declined with a reason, or deferred. A declined Insight is also an answer: the observation was seen.

## 12.3 Evolution Work Packages

Accepted Insights are bundled: a cluster of related Insights — or a single significant one — becomes a normal Work Package and travels through the normal Hingepoint Flow. The Intent describes the observed problems; Shaping specifies the rule change — including its Governance and Security Specification, because changes to MAY are governance; Generation edits the affected artifacts; Verification and Validation test the changed rules; the Rule Owner resolves the Acceptance Hingepoint.

Evolution Work Packages are created, prioritized, and assigned like any other work. Process improvement thereby becomes planned, accepted work — visible, prioritized, and accountable — rather than a side activity.

## 12.4 The Rulebase

The rule artifacts live in a versioned, central Rulebase with releases and a changelog. Larger organizations layer it:

- an organization-wide baseline, and
- declared team or domain overlays where local deviation is permitted.

Where deviation is permitted, it is explicit — explicit over implicit, applied to the process itself. Drift becomes visible because overlays are declared instead of accumulating in local files.

Rollout is incremental, following Chapter 14: a pilot unit adopts a release, its feedback — often new Insights — flows back, and the release then reaches the whole organization. AI agents adopt new versions automatically; they read the artifacts anyway. Humans receive the changelog.

# 13 The Reference Operating Model

The normative core of the Hingepoint Framework defines how Work Packages are processed. How Work Packages originate, how they are prioritized, and how they reach a Human-AI work unit remains an organizational decision (Chapter 10). This chapter is a recommendation, not a requirement: a default operating model for organizations that do not have — or no longer have — working mechanisms of their own. Organizations with functioning structures keep them.

> Hingepoint mandates the flow, not the organization. For the organization, it offers a default.

## 13.1 The Flow Board

Work Packages move across a Kanban-style board with five states:

> Intake → Defined → In Flow → In Acceptance → Done

Intake collects proposed Work Packages. Defined holds the Work Packages that are specified and prioritized — available for pull. In Flow covers the entire Hingepoint Flow — the Ramp and the Delivery Cycle. In Acceptance holds Work Packages waiting at their Acceptance Hingepoint. Done holds accepted Work Packages.

A blocked Work Package — one waiting at a reached, unresolved Hingepoint — remains In Flow and is marked as blocked. The Hingepoint Register remains the single source of truth; the board only makes its state visible.

## 13.2 Two Responsibilities

The Reference Operating Model requires two responsibilities — not two new roles (Section 10.4).

### Definition and Prioritization

An existing role — a product owner, a team lead, a head of development, a project lead — owns the Intake and the order of the Defined list. The model requires only that there is exactly one ordered list of Defined Work Packages, and exactly one person accountable for its order. The prioritization method remains free.

### Acceptance

The person who resolves the Acceptance Hingepoint (Section 8.4). Often the same person who defined the Work Package — but not necessarily.

In small organizations, one person may hold both responsibilities — and others besides. This is explicitly acceptable.

## 13.3 Pull, Not Assignment

When a Human-AI work unit becomes free — its Work Package is completed, or blocked without meaningful continuation — it pulls the topmost Work Package from the Defined list. Becoming free is the event; no assignment meeting is required. Coordination keeps its purpose (Section 10.2) and is relieved of routine distribution.

## 13.4 Work-in-Progress Limits

A work unit holds one active Work Package, plus at most one blocked Work Package. The limit prevents blockage from being masked by ever more parallel work: a blocked Work Package should be visible and should create pressure to resolve its Hingepoint. This keeps Flow Uptime honest. The limit of two respects the rule that a blocked unit may continue productive work on another Work Package.

## 13.5 Coordination in the Board Model

Coordination is triggered by the state of the Hingepoint Register — not by the calendar. It takes place when unresolved Hingepoints exist that cannot be resolved locally. Organizations that cannot create ad-hoc meetings with decision-makers may reserve a short recurring slot that is cancelled whenever the register is empty: the calendar is then merely the container; the content remains event-driven.

## 13.6 Scaling

- A single team uses one board and one Coordination; responsibilities may be held by one person.
- Several teams or products use one board per team or product. Dependencies between them are Dependency Hingepoints and travel through local Coordination into the organization's existing escalation paths. No additional coordination layer is introduced.
- Large organizations achieve homogeneity through a uniform board schema and uniform register formats, which make Flow Uptime comparable across the organization. Portfolio and project management remain untouched (Chapter 14) and feed the Intake.

# 14 Adoption

The Hingepoint Framework is designed to integrate with existing software organizations. It does not require organizations to replace their existing structures, planning processes, governance models, or delivery practices. Instead, it introduces a consistent way of organizing Human-AI collaboration within those environments.

Organizations adopt Hingepoint by introducing its principles, artifacts, and development flow. Existing organizational structures remain unchanged unless the organization decides otherwise.

## Incremental Adoption

Hingepoint may be adopted incrementally. Organizations do not need to transform all projects simultaneously. Individual teams, products, or repositories may adopt the framework independently. Experience gained from early adoption naturally informs wider organizational use.

## Existing Processes

The framework intentionally does not replace:

- project management,
- portfolio management,
- backlog management,
- organizational governance,
- release management,
- quality management,
- or organizational leadership.

These practices continue to exist. Hingepoint defines how software is developed — not how organizations are managed.

## Coming from Scrum

Hingepoint and Scrum are incompatible in one precise place: the timebox. Scrum synchronizes by calendar — sprint, planning, review, and retrospective follow a fixed cadence. Hingepoint synchronizes by events. A sprint commitment and a Hingepoint — work pauses until a decision is made — do not coexist. This Guide states this openly, and Chapter 3 explains why: frameworks optimize the dominant bottleneck of their time, and the bottleneck has moved.

The incompatibility concerns the timebox — not the agile values. What Sprint Review and Retrospective provide — making work visible, valuing it, and learning — Hingepoint provides event-driven. Much of a Scrum organization remains directly connectable:

| Scrum | Hingepoint |
|---|---|
| Product Owner | Responsibility for defining, prioritizing, and usually accepting Work Packages |
| Product Backlog | Intake and Defined list of the Flow Board (Chapter 13) |
| Refinement | Intent and Shaping — deeper, and in dialogue with AI |
| Sprint Planning | Pull principle, plus the Planning phase of each Work Package |
| Sprint (timebox) | Event-driven flow, synchronized by Hingepoints |
| Daily Scrum | Coordination on demand, driven by the Hingepoint Register |
| Sprint Review | Acceptance Hingepoint of each Work Package — more frequent, smaller, usually asynchronous |
| Retrospective | Evolution (Chapter 12) — continuous and event-driven |
| Definition of Done | Verification, Validation, and the Governance and Security Specification |
| Scrum Master | No equivalent role — the competence remains valuable |

The Scrum Master role has no equivalent in Hingepoint. The competence behind it remains valuable: facilitation of Coordination, coaching during adoption, and training. Hingepoint replaces the timebox — not the people who made agile work.

## Continuous Evolution

The framework itself is intentionally small. Organizations are expected to develop supporting practices, templates, automation, and governance around it. These extensions remain organizational decisions. They are not part of the Hingepoint Framework.

# 15 Origins and Acknowledgements

## Evolution Rather Than Reinvention

The Hingepoint Framework did not emerge in isolation. It builds upon decades of experience in software engineering, software architecture, and agile development, combined with recent advances in generative AI.

As Chapter 3 describes, each generation of frameworks solved the primary constraint of its time. Hingepoint continues this evolution: it does not replace previous approaches, it extends them where new constraints have appeared.

## Standing on Existing Work

The framework has been influenced by many existing ideas and communities. Among them are:

- Software Engineering
- Agile Software Development
- Scrum
- Kanban
- Shape Up (the origin of the term Shaping)
- Domain-Driven Design
- Specification-Driven Development
- AI-Driven Development
- Vibe Coding
- Context Engineering
- Secure Software Development
- Human-Centered Design

Hingepoint neither attempts to replace these disciplines nor claims ownership of their ideas. It combines selected concepts where they contribute to accountable software development with AI.

## A New Organizational Perspective

Recent approaches have significantly improved how AI generates software. Hingepoint focuses on a different question:

> How can organizations maintain accountability while software is increasingly generated by AI?

The framework therefore concentrates on:

- shared understanding,
- accountable ownership,
- organizational coordination,
- and measurable organizational flow.

These aspects complement existing technical approaches rather than competing with them.

## A Living Framework

> The Hingepoint Framework is expected to evolve. Its principles are intended to remain stable. Its implementation practices, automation, examples, and tooling will continue to develop as software engineering and AI evolve.

The framework therefore separates stable principles from evolving practices. This Guide defines the framework. Supporting material, examples, templates, and reference implementations evolve independently.