HELMARK STANDARD

Helmark STD 2.0

AI Delivery Governance for development teams

helmark.org

English translation of Helmark STD 2.0. In case of any discrepancy, the Polish version published at helmark.org prevails.

Contents

0.0 The standard in five sentences

1. Every task carries a label — NONE / ASSIST / CORE — describing how much AI contributed to producing the result.
2. AI involvement sets the minimum level of verification. Criticality is primarily a property of the component and is inherited by the task; it can raise verification requirements.
3. Responsibility for a change cannot be transferred to AI — every change has a human owner who knowingly accepts its result.
4. Understanding of the system is treated as an asset: as the adoption matures, the team identifies Anchors, monitors cognitive debt and checks whether it can independently verify solutions proposed by AI.
5. Adoption is incremental and fitted to the organisation: classification and verification first, then enforcement of the rules, and only later the mechanisms that protect understanding. Helmark does not impose an adoption timeline.

Opening part

0.1 About the standard and its scope

Helmark answers a single question: **how do you preserve human responsibility, verifiability and understanding of software when an ever larger share of the work is done by AI?**

Helmark is a standard for controlling software development carried out with AI, intended for development teams.

It defines:

— how a team classifies the involvement of AI in its work,

— how it adjusts verification to the level of AI involvement and the criticality of the component,

— how it preserves human responsibility for the result,

— how, as the adoption matures, it protects the team's ability to understand and maintain its own system,

— which minimum artefacts make it possible to demonstrate that control actually took place.

The standard is free and publicly available. Training, certification, verification and audits provided by Helmark may be paid.

Helmark works with any way of working, any tracker, and also in a PR-centric model without a classic tracker.

Helmark does not define:

— process roles,

— ceremonies,

— how work is planned or estimated,

— a specific AI vendor,

— a specific version control system.

Helmark does not replace security, privacy or AI risk management standards, nor legal regulations. In particular, the standard does not cover:

- the choice of AI tools or prompt-writing techniques,
- data protection and trade secrets,
- legal requirements concerning AI,
- model security and agent permissions,
- software supply chain security,
- measurement of individual productivity.

If an organisation has its own security, compliance or risk management requirements, those apply in addition to Helmark, not instead of it.

0.2 Glossary

TERM	DEFINITION
Task	A unit of work closed by the team: a task, ticket, issue, PR or equivalent.
Component	A part of the system or an area of responsibility, e.g. a service, module, integration or repository.
Significant component	A component with HIGH or CRITICAL criticality. The team may also treat another component as significant if losing knowledge of it constitutes a real risk.
Task owner	The human who knowingly accepts the result of a task and is responsible for its verification.
Reviewer	A human verifying someone else's change; not its owner.
Agent	An AI tool performing work autonomously or semi-autonomously, e.g. analysing a repository and preparing a change on its own.
NONE / ASSIST / CORE	The three levels of AI involvement in a task.
LOW / HIGH / CRITICAL	The three criticality levels, assigned primarily to components. A task inherits the criticality of its component.
ASSIST note	A short record on an ASSIST task of LOW criticality: where AI was used and how the owner checked the result.
Intent description	A record of what result was ordered from AI, under what constraints, and what was accepted from the output without significant change.
Cognitive debt	Code whose behaviour and consequences the team cannot explain well enough to maintain it safely.
Knowledge Anchor	A person who can explain and teach how a component works.
Orphaned component	A significant component for which the team cannot name any Anchor.
Handover	A short record, in your own words, when responsibility for a component is passed on.
Focus Block	Optional protected time set aside for building or rebuilding understanding.

TERM	DEFINITION
Cognitive Debt Signal	A recurring check of the state of understanding across components, and of the trend.
Intervention boundary	The state of a component in which the team can no longer independently judge whether a solution proposed by AI is correct.

0.3 The founding rule

One rule applies across the whole standard and takes precedence over every other provision.

R1. Responsibility for a change cannot be delegated to AI.

Every change has a human owner. The owner is responsible for the result having been knowingly accepted and verified in line with the requirements of this standard.

A model, an agent, a prompt or an AI vendor cannot be treated as the party responsible for accepting a change. “The AI generated it that way” is not a justification for skipping verification.

R1 applies regardless of whether the work was done by a human, an AI assistant or an agent.

R1 does not mean automatically blaming the owner of every change for every subsequent incident. Incidents may have systemic, process, architectural or organisational causes.

There is no software change for which, at the moment of acceptance, a tool alone is responsible.

Part I — classification

1.1 Three levels of AI involvement

Every task must carry exactly one classification of AI involvement — except for changes covered by a written class of trivial changes (1.2). **The classification is set by the task owner before review or before the change is closed.**

LEVEL	WHAT IT MEANS	EXAMPLE
NONE	AI played no part in producing the result.	A human wrote the code themselves.
ASSIST	AI supported the human, but the human set the main direction of the solution.	AI suggested a fragment, a name, a refactoring or a local solution.
CORE	AI produced the main result, and the human primarily ordered, integrated or verified it.	The human described the problem, AI delivered the implementation.

The levels do not rank the quality of the work. CORE is not worse than NONE. Helmark does not limit the use of AI; it requires an honest record of how the result came about, because the minimum verification depends on it.

1.2 Classification rules

- The classification is set by the task owner before review.
- The classification field must have no default value — a value has to be chosen deliberately.

- In case of genuine doubt, classify higher.
- The percentage of changed lines may support a classification, but it does not replace human judgement.

The deciding rule

If you removed the AI contribution, would a recognisable implementation made by a human remain?

If what remained were mainly a prompt, a description of the problem, a choice between variants, or fixes applied to a generated implementation, the task is usually CORE. If the substantive implementation was done by a human and AI helped locally, it is ASSIST.

Trivial changes

A team may define a narrow, written class of trivial changes — e.g. typos in text, automatic formatting, tidying imports — exempt from classification.

The exemption must be written down, must not cover HIGH and CRITICAL components, and is subject to the same sampling check as classification (4.4).

Classification disagreements

A reviewer may challenge the classification set by the owner. If the two cannot agree, the higher of the levels under consideration applies. The team records such cases as its own precedents (4.1, Stage 2).

Borderline examples

SITUATION	CLASSIFICATION	WHY
AI generated code that I then reworked heavily	CORE	The foundation of the solution came from AI.
AI generated several versions and I assembled one from them	CORE	Without the AI output the substantive implementation would not exist.
I wrote the code myself, AI fixed names and typos	ASSIST	The main implementation is human.
AI suggested the algorithm, but I wrote the implementation myself	ASSIST	AI supported the solution but did not deliver the main result.
An agent analysed the repository and prepared a PR on its own	CORE	The result was produced by an agent.

1.3 Agent work

The output of an agent is CORE by definition — regardless of the quality of the result, the number of files changed, the tests it ran, or whether the agent corrected its own mistakes.

Every task performed by an agent must have a human owner assigned before the agent starts work. The owner is responsible for knowingly accepting the result in line with R1.

If an agent runs on a schedule or initiates changes itself, the owner must be assigned in advance — for the repository, component or task queue the agent operates on. A change without an assigned owner cannot be closed.

Assigning an owner in advance does not remove the need for active acceptance. The owner must knowingly accept the result before deployment, and the change is subject to the CORE requirements in

2.1 — including a real review carried out by a human other than the owner. An automatic merge or a tool approval does not satisfy this requirement.

For an agent task, the following minimum must be preserved:

1. the intent — what result the agent was meant to achieve,
2. important constraints — what it was not allowed to change, or which conditions it had to meet,
3. the tool or type of agent that performed the work,
4. any significant configuration, instructions, permissions or integrations, if they could have affected the result.

A full transcript of the conversation with the agent is not required, nor is a complete history of its actions. The purpose is to make it possible later to establish what was ordered, under what conditions, and what result was accepted.

1.4 Sources of classification

Classification may be set manually, automatically or in a hybrid way.

MODE	RULE
Manual	The owner sets the level themselves. The default approach, and it works everywhere.
Automatic	An attribution tool, an agent or metadata proposes or sets the level.
Hybrid	A tool proposes the level and a human confirms it. Recommended where the team has suitable tooling.

Automation may propose a classification, apply a label or detect agent work. It cannot by itself satisfy the requirement of human verification.

1.5 Component criticality and risk inheritance

AI involvement and the risk carried by a change are two different things. That is why Helmark uses three criticality levels: LOW, HIGH and CRITICAL.

Criticality is primarily a property of the component, not a decision taken afresh for each task.

- The team assigns a default criticality level to significant components; a task inherits the level of the component it concerns.
- If a task concerns several components, it inherits the highest of their levels.
- The owner may always raise the criticality of a task if that particular change carries more risk than is typical for the component.
- If a component has no criticality assigned yet, the owner determines it for the task and the team completes the component map.
- Lowering criticality below the component's level should be an exception and requires a written justification.

LEVEL	MEANING	EXAMPLE AREAS
LOW	An error has limited impact and is easy to detect or reverse.	Documentation, tests, harmless UI changes, internal tooling.
HIGH	An error may materially affect users, data, availability, integrations, security or a business process.	Significant business logic, integrations, important data processing.
CRITICAL	An error may lead to serious consequences for security, privacy, money, access control, data or critical processes.	Payments, authorisation, key security mechanisms, critical processes.

1.6 How AI involvement relates to criticality

The level of AI involvement sets the minimum verification. The criticality of the component, or of the particular change, may only raise it.

For example: a large generated change in tests may be CORE / LOW, while a few lines suggested by AI in an authorisation mechanism may be ASSIST / CRITICAL. The second change requires stronger control despite the smaller AI contribution.

For NONE tasks, Helmark adds no requirements on account of criticality. Work without AI involvement is governed by the rules the team applies independently of this standard; the escalation described in Part II applies to ASSIST and CORE tasks.

Part II — verification

2.1 Minimum requirements

Verification follows from the combination of AI involvement and component criticality. For ASSIST / LOW, Helmark does not require involving a second person merely because AI was used.

CASE	MINIMUM REQUIREMENT	VERIFICATION
NONE (any criticality)	Your team's normal definition of done; Helmark adds nothing.	Per the team's process.
ASSIST / LOW	ASSIST note: where AI was used and how the result was checked.	The owner may verify alone.
ASSIST / HIGH	Intent description + a test or scenario + real human review.	Owner + reviewer.
ASSIST / CRITICAL	ASSIST / HIGH requirements + an independent human verification criterion.	Owner + reviewer.
CORE / LOW	Intent description + a test or scenario + real human review.	Owner + reviewer.
CORE / HIGH	CORE / LOW requirements + a deliberate assessment of risk specific to the component.	Owner + reviewer.

CASE	MINIMUM REQUIREMENT	VERIFICATION
CORE / CRITICAL	CORE / HIGH requirements + an independent human verification criterion.	Owner + reviewer.

The ASSIST note is meant to be short. Its purpose is not to document AI for documentation's sake, but to show where AI influenced the result and what the owner checked.

2.2 HIGH and CRITICAL changes

For HIGH, the level of verification rises regardless of whether the task is ASSIST or CORE. For CRITICAL, at least one significant criterion or test must have an independent human source.

An independent source may be:

- a business requirement,
- a specification,
- a threat model,
- domain knowledge,
- a human decision resulting from risk analysis.

AI may help write a test. It cannot be the only source of the answer to the question: **what should actually be tested?**

2.3 Intent description

An intent description is required for all CORE tasks and for ASSIST tasks of HIGH and CRITICAL criticality.

The description must make it possible to compare what was ordered with what was delivered, later and without the author in the room. Minimum content:

1. what was ordered,
2. which important constraints were given,
3. what was accepted from the result without significant change.

A full transcript of prompts is not required. What is recorded is the essential intent, the constraints and the accepted result.

> *Ordered: an invoice export endpoint to CSV, consistent with the existing /orders/export. Constraints: no new dependencies; streaming above 10k records; date format as in the UI. Accepted: the generated implementation, except for field names changed by hand.*

Linking instead of duplicating

If the intent and the constraints are already recorded in a durable artefact — an issue, a task in a tracker or a requirements document — the intent description on the change may link to that artefact instead of repeating its content.

A link covers only those elements the artefact actually contains. An issue usually describes what the requester expected, not what was ordered from AI and under what constraints — the missing elements are supplied on the change. The third element, what was accepted from the result without significant change, arises after the work and is always recorded on the change.

Proportionality of the description

The length of an intent description is proportional to the change. For a small change, a single sentence covering the three required elements is enough. An intent description is not solution documentation.

> *Ordered: e-mail address validation compliant with RFC 5322, no new dependencies. Accepted: the generated implementation, unchanged.*

2.4 Review of changes

Review is carried out by a human other than the task owner. Helmark does not prescribe a review technique — it only requires the review to be real. A bare “LGTM” after an automated summary is not evidence of review.

In a CORE change, the main result comes from AI. The reviewer checks at least:

1. Delivery = order — does the implementation match the described intent and constraints?
2. Verification checks the goal — does the test or scenario actually check the behaviour that was meant to be achieved?
3. The change is understandable — can the reviewer explain why the solution is correct?

In an ASSIST change of HIGH or CRITICAL criticality, the reviewer checks at least points 2 and 3, and whether the intent description identifies the places where AI influenced the solution.

2.5 AI may support review

AI may analyse a diff, find defects, suggest fixes, search for vulnerabilities, generate questions and assess tests.

Acceptance of a change by an AI tool is not the human review required by Helmark.

A human may use AI while reviewing, but makes the acceptance decision themselves and must be able to explain the basis for it.

Part III — protecting understanding

Part III is not a precondition for reaching L1. It is the path towards L3 maturity, activated once basic classification and verification are already working.

AI can increase the pace at which software is produced faster than the pace at which human understanding of the system is built. Helmark treats understanding as an asset that can grow, decay, be passed on, monitored and rebuilt.

3.1 Cognitive debt

Cognitive debt is the situation in which code works, but the team does not have enough understanding to maintain it safely.

This is not the same as technical debt. Technical debt can make change harder. Cognitive debt means the team is changing a system whose behaviour it cannot adequately predict.

Code that works and has tests, but that nobody can explain — nobody can say why the solution behaves the way it does — is particularly risky.

3.2 Knowledge Anchor

Every significant component should have at least one Anchor. An Anchor is a person who can explain how the component works, point out its most important dependencies and risks, and pass that knowledge on to someone else.

An Anchor is always a human being. Documentation, a service catalogue and an AI agent can be tools for an Anchor — they shorten the work of building and recovering understanding — but they do not replace a person: they do not answer questions in a way for which someone is accountable, and they cannot take on responsibility under R1.

An Anchor is not automatically the author. A name in a table does not by itself mean the component has an Anchor — what counts is the real ability to explain it.

Gaps are allowed, but they must be visible. For CRITICAL components, the team should aim to have at least two people able to explain how they work.

The list of Anchors should live in an artefact the team already maintains, e.g. a service catalogue, a repository or an architecture document. If the team uses a code ownership file such as CODEOWNERS, Anchors should be marked separately within it — ownership of changes and the ability to explain a component are not the same thing.

3.3 Handover

A handover exists to pass on understanding, not to produce documentation. It is required when responsibility for a significant component or area of work is deliberately transferred to another person or team, when responsibilities rotate, or when someone leaves the project.

A team may also require a handover after large CORE changes.

Minimum handover: what this element does, and where the most important risk lies.

> *"We cache sessions for 24 hours; the biggest risk is a duplicated key when the time zone changes."*

3.4 Focus Block

Helmark recommends giving the team real conditions in which to build and rebuild understanding. One such practice is the Focus Block — protected time without meetings and without the expectation of an immediate reply to messages.

The Focus Block is a recommended practice, not a mandatory condition of conformance. A team may use another mechanism that genuinely provides time for learning, analysis and recovering knowledge.

3.5 The Cognitive Debt Signal

At L3, the team checks the state of understanding of the system on a recurring basis.

The team records its own Signal cadence and ties it to a rhythm it already has — e.g. every second retrospective, an architecture review or the start of a quarter. The cadence should not be less frequent than once a quarter: below that, the trend stops being visible.

A minimum Signal answers three questions:

1. How many significant components have at least one Anchor, and how many are orphaned?
2. How many CRITICAL components have at least two people able to explain how they work?
3. How many components are past the intervention boundary?

The trend matters most. For every orphaned component, or component past the intervention boundary, the team makes a deliberate decision: rebuild the understanding, reduce the risk in another way, or accept the risk and write that decision down.

The Signal describes the state of the system, not an assessment of people. Using the number of Anchors to evaluate employees leads to people declaring understanding they do not have — and the Signal stops measuring anything at all.

3.6 The intervention boundary

The problem is not the use of AI itself. The problem appears when the team can no longer independently judge whether a solution proposed by AI is correct.

The intervention boundary has been crossed when the team cannot adequately explain the problem, identify its cause, or judge whether the proposed fix is safe and correct.

> **Can we understand a failure of this component and independently verify that the proposed solution is correct, even if AI helps us find it?**

An answer of “no” means the component is past the intervention boundary. AI may still help maintain it; the problem is the absence of deliberate, independent verification.

Once the boundary has been crossed, the team decides: rebuild the understanding, reduce the risk in another way, or knowingly accept the risk and write that decision down.

Part IV — adoption and conformance

4.1 Adoption step by step

Helmark does not prescribe a normative or guaranteed adoption timeline. The pace depends on, among other things, team size, the number of repositories and components, existing code review practices, and how heavily AI is used.

Adopting the basic rules is usually shorter than reaching full L3 maturity. The standard deliberately describes the order of steps instead of imposing a schedule.

Stage 1 — reach L1: classification and verification

1. Announce R1: responsibility for a change is not delegated to AI.
2. Add NONE / ASSIST / CORE to the existing workflow.
3. Assign a default criticality — LOW / HIGH / CRITICAL — to significant components or areas. If the organisation already has a risk classification, reuse it rather than creating a new one.
4. Turn on the minimum verification rules from Part II — in particular ASSIST / LOW without a mandatory second person, and human review for CORE.
5. Add the minimum fields to the task or PR, and the ability to find CORE, HIGH and CRITICAL changes.
6. Check on a sample of closed changes whether team members understand the classification in the same way.

Stage 2 — towards L2: enforcement

1. Resolve the most common classification ambiguities using the team's real examples, and record the outcomes as your own precedents.

2. Turn on technical or process safeguards that make it hard to merge CORE without human review.
3. Check on samples whether HIGH and CRITICAL changes receive the increased verification.
4. Automate labels and criticality inheritance where it reduces manual clicking, but do not let automation replace human responsibility.

Stage 3 — towards L3: protecting understanding

1. Choose the significant components for which losing knowledge would be a real risk.
2. Identify the existing Anchors and leave the gaps visible where knowledge is missing.
3. Start the Signal for those components and record its cadence.
4. Check the intervention boundary and act on components the team cannot independently control.
5. Use handovers when responsibility genuinely changes hands. Treat the Focus Block, or another mechanism providing time to rebuild knowledge, as a supporting practice.

4.2 Minimum tooling requirements

Helmark requires only the ability to perform three kinds of operation:

1. Recording the NONE / ASSIST / CORE classification. Criticality may be read from a component map instead of being copied by hand into every task.
2. A place for a short piece of text: an ASSIST note, an intent description, agent information or a handover.
3. The ability to find changes that require control, e.g. CORE, HIGH or CRITICAL.

Helmark does not require a plugin, a dedicated platform, a separate application, or automatic detection of AI involvement.

4.3 Maturity levels

LEVEL	NAME	WHAT IS TRUE
L1	Classification and verification work	Tasks carry credible NONE / ASSIST / CORE labels and agent work is marked as CORE; criticality follows from the component or is determined by exception; R1 is known; minimum verification is carried out.
L2	Verification is enforced	ASSIST and CORE artefacts exist where they are required; HIGH and CRITICAL receive increased scrutiny; human review genuinely takes place; the rules are checked on samples.
L3	Understanding is protected and measured	Significant components have visible Anchors and visible gaps; the Signal runs on a recorded cadence; the intervention boundary is monitored; handovers support the transfer of responsibility; the data leads to action.

The levels are cumulative. L3 means L1 and L2 are also met.

4.4 Audit checklist

Every “yes” must be supportable by an artefact, an observation or a sample of changes. A declaration on its own is not enough.

1. **[L1]** Do the tasks examined carry a NONE / ASSIST / CORE classification?

2. [L1] Does the criticality of a task follow from the assigned level of its component, or was it determined deliberately where no mapping existed?
3. [L1] Does a random ASSIST / LOW task carry a note showing where AI was used and how the owner verified the result?
4. [L1] Does a random CORE task have an intent description, a test or scenario, and human review?
5. [L1] If the team uses agents or generates implementations — do any CORE tasks appear in the sample examined at all?
6. [L2] Is classification checked periodically on a sample of closed changes, and are the most common ambiguities recorded as team precedents?
7. [L2] Does a random ASSIST or CORE change of HIGH or CRITICAL criticality meet the increased verification requirements?
8. [L2] Does the review history show real human review rather than only the acceptance of an automated tool?
9. [L3] Is there a current list of Anchors and visible knowledge gaps for the significant components?
10. [L3] Is the Signal checked on the recorded cadence, and does it lead to action on orphaned components or knowledge gaps?
11. [L3] Does the team know which components are past the intervention boundary, and does it record decisions to rebuild knowledge, reduce the risk or accept it?

Audit questions are applied in proportion to the maturity level being claimed. A team at L1 does not yet have to meet the requirements concerning Anchors, the Signal and the intervention boundary.

Closing part

5.1 Formal matters

Versioning

The standard is numbered STD MAJOR.MINOR. MINOR denotes clarification without a substantive change to the model. MAJOR may change the requirements or the structure of the standard. A team claiming conformance states the version, e.g. “Helmark STD 2.0”.

Governance

Helmark is a publicly available standard whose development is governed by its authors. Change proposals may be submitted through helmark.org. The authors decide which changes are incorporated into the official version.

This model means openness to free use, reading, citation and the submission of change proposals, while the official specification remains centrally governed and versioned.

Certification

Helmark runs a certification programme based on this standard. The HDP and HDL titles are awarded solely within that programme; their scope, requirements and renewal rules are published at helmark.org.

Certification is not a condition of using the standard. A team may adopt Helmark and claim a maturity level without taking part in the certification programme.

Name and mark

Anyone may use the standard free of charge, implement its requirements and state which version they follow.

The name “Helmark Certified”, the HDP and HDL titles, and the official graphic mark may be used only in accordance with Helmark certification and licensing rules. A modified document must not be published as an official version of the Helmark Standard.

Authorship and licence

Authors: Piotr Sobiegała & Mateusz Pluta. The official version of the standard is published at helmark.org.

Permission for internal adaptations

In addition to the CC BY-ND 4.0 licence, the authors grant a further permission: an organisation may create internal translations and adaptations of the standard, and incorporate its requirements into its own procedures, policies and management systems — for internal use only.

Materials created on that basis may not be publicly distributed, made available to third parties, or presented as an official version of the Helmark Standard. They should identify the source version, e.g. “based on Helmark STD 2.0”. Claims of conformance and maturity level always refer to the official version of the standard.

Annex A — the standard in practice

ELEMENT	WHEN	WHO	WHAT YOU ACTUALLY DO
AI classification	Every task	Owner	NONE / ASSIST / CORE.
Criticality	Usually once per component	Team	Assign LOW / HIGH / CRITICAL to the component. Tasks inherit the level.
NONE	Before closing	Owner	No additional Helmark requirements — the team's own rules apply.
ASSIST / LOW	Before closing	Owner	ASSIST note: where AI was used and how you verified the result yourself. Helmark does not require a second person.
ASSIST / HIGH	Before merge	Owner + reviewer	Intent description + a test or scenario + real human review.
ASSIST / CRITICAL	Before merge	Owner + reviewer	As HIGH + an independent human verification criterion.
CORE	Before merge	Owner + reviewer	Intent description + a test or scenario + real human review; for HIGH and CRITICAL the requirements rise per 2.1.
Agent	Before it starts	Owner	Assign a human; the result is CORE; record the intent and the significant conditions of the agent's work.
Anchors	On the path to L3	Team	For significant components, name the people who genuinely understand how they work. Leave the gaps visible.
Handover	Transfer of responsibility	Person handing over	Briefly: what the component or change does, and where the most important risk lies.
Signal	At L3, on a cadence	Team	Check Anchors, orphaned components and components past the intervention boundary.
Intervention boundary	At L3	Team	Check whether you can independently verify a solution proposed by AI.
Focus Block	Optional	Team	Provide protected time for building or rebuilding knowledge, or use another mechanism that works.

Annex B — adoption without a tracker (GitHub / GitLab)

A team working PR-centric can adopt Helmark without a classic tracker. There is no obligation to repeat criticality in every PR if it can be determined unambiguously from the component.

REQUIREMENT	HOW IT LOOKS IN GITHUB / GITLAB
AI classification	Labels ai-none / ai-assist / ai-core, or a field in the PR template.
Component criticality	A map of components, services or repositories with LOW / HIGH / CRITICAL. A label on the PR is optional if the level can be determined unambiguously.
ASSIST / LOW	An “AI used for / verified by owner” section in the PR template.
ASSIST / HIGH and CRITICAL	An intent description section + review; for CRITICAL also an independent criterion.
CORE	An intent description section + a test or scenario + required human approval.
Agent (CORE)	The name of the agent or tool, plus significant instructions, permissions and integrations.
Human review	Branch protection or approval by a human. A bot approval does not satisfy the requirement.
Filters	Search by ai-core and by HIGH and CRITICAL components.
Anchors (L3)	A service catalogue or another existing artefact. In CODEOWNERS, mark Anchors separately — ownership of changes is not the same as the ability to explain a component.
Signal (L3)	A list of components annotated Anchor / Orphaned / Beyond Boundary.

Annex C — minimal PR template

The template may be shortened or automated. Fields that do not apply to a given class of change can be hidden conditionally.

AI involvement

NONE / ASSIST / CORE

Component and criticality

Component: […]

Criticality: inherited from the component. Fill in by hand only when there is no mapping, or when the change requires raising the level.

If ASSIST / LOW

AI used for: […]

I verified the result by: […]

If ASSIST / HIGH, ASSIST / CRITICAL or CORE

Intent — what was ordered: […]

Constraints: […]

What was accepted without significant change: […]

Test or scenario: […]

If an agent was involved

Agent or tool: […]

Significant configuration, instructions, permissions: […]

If CRITICAL

Independent human verification criterion: […]

If you are handing over responsibility for a component

Handover — what it does and where the most important risk lies: […]