

## 1. ツール別の最適担当表（作業→推奨ツール→理由）

作業	推奨ツール	理由
コード編集 (ローカル ファイルの 直接修正)	ローカルCLI (e.g., Git CLI, Vim/Neovim with AI plugins like Aider)	CLIはローカル環境で高速・低成本に実行可能。GitHub公式ドキュメント ( <a href="#">git-scm.com</a> ) とAiderのGitHubリポジトリ ( <a href="#">github.com/paul-gauthier/aider</a> ) に基づき、事故防止のための差分確認 (git diff) が容易で、再現性が高い。IDEエージェントより軽量でオフライン対応可。
コード検索 (リポジト リ内探索)	ローカルCLI (e.g., ripgrep or Git grep with AI-assisted tools like Aider CLI)	CLIツールの公式ドキュメント ( <a href="#">ripgrep on GitHub</a> ) で高速検索が証明されており、RAGと組み合わせればSSOT参照精度が向上。外部依存を最小限に抑え、コスパが高い。
テスト実行 (ユニット テスト/ インテグレ ーションテス ト)	ローカルCLI (e.g., pytest or Jest CLI, integrated with Aider for AI suggestions)	pytest公式ドキュメント ( <a href="#">pytest.org</a> ) とAiderの統合例 (GitHub リポジトリ) で、CLIがテストの自動化と即時フィードバックを保証。事故防止のためのログ出力が標準で、スピードを最大化。
差分確認/適 用 (変更管 理)	ローカルCLI (e.g., Git diff/patch)	Git公式ドキュメント ( <a href="#">git-scm.com</a> ) で最小差分運用が推奨されており、SSOT設計書のPart14 (変更管理) と整合。人間確認を強制し、事故を防ぐ。
実装支援 (コード生 成/補完)	IDEエージェント (e.g., Cursor or GitHub Copilot)	Cursor公式サイト ( <a href="#">cursor.sh</a> ) とGitHub Copilotドキュメント ( <a href="#">docs.github.com</a> ) で、IDE内リアルタイム支援が精度向上を示す。2026トレンドとして、AI-native IDEがリポジトリ全体理解を強化 ( <a href="#">Builder.io</a> ブログ)。
リファクタ リング (コ ード改善)	IDEエージェント (e.g., Cursor with agentic features)	CursorのGitHubリポジトリとAnthropicのClaude Code統合で、コンテキストベースのリファクタがベストプラクティス。スピードと精度を両立し、事故防止のためのプレビュー機能あり。
修正提案 (バグフィ ックス提 案)	IDEエージェント (e.g., Claude Code integrated in VS Code)	Anthropic公式 ( <a href="#">anthropic.com</a> ) とVS Code拡張ドキュメントで、AIエージェントが提案精度が高い。2026レビュー (Faros AI ブログ) で実務効率化が確認。

外部公式情報の取得 (APIドキュメント/標準規格)	MCP (Model Context Protocol)	Anthropic公式 ( <a href="https://anthropic.com/news/model-context-protocol">anthropic.com/news/model-context-protocol</a> ) と MCP公式サイト ( <a href="https://modelcontextprotocol.io">modelcontextprotocol.io</a> ) で、AIを外部システムに接続する標準プロトコル。根拠集めに最適で、2026標準化によりコストパフォーマンス向上 ( <a href="#">IBM think/topics</a> )。
仕様確認 (外部規格/ベストプラクティス検証)	MCP (integrated with ZAI for querying)	MCPのオープンスタンダード ( <a href="#">Wikipedia</a> ) と Z.ai公式 ( <a href="#">z.ai</a> ) で、外部コンテキストを安全に注入。事故防止のためのプロトコルベースアクセス。
根拠集め (一次情報収集)	MCP (with agent frameworks like LangGraph)	MCPドキュメントで外部ツール接続が定義されており、LangGraph公式 ( <a href="https://langchain.com/langgraph">langchain.com/langgraph</a> ) でエージェントチーンが可能。精度を最大化し、誤情報リスク低減。
リポジトリ内SSOT参考照 (知識検索/更新)	RAG (e.g., integrated with LlamaIndex or Haystack)	Neo4jブログ ( <a href="https://neo4j.com/blog/genai/advanced-rag-techniques">neo4j.com/blog/genai/advanced-rag-techniques</a> ) と Towards AI記事で、advanced RAG (re-ranking, agentic RAG) が2026ベストプラクティス。SSOT精度を上げ、スピードを確保。

## 1. MCP活用で“抜けを埋める”具体プロンプト例 (3~5本)

- ・プロンプト1: "Using MCP to connect to the official Git documentation API, retrieve the latest guidelines on branch management for SSOT repositories. Extract key rules for '1Part=1Branch' from the SSOT design, identify any gaps in our current Part14, and suggest minimal updates with citations."
- ・プロンプト2: "Via MCP, access the Anthropic API docs and fetch the current best practices for AI agent permissions. Compare with our Permission Tier in Part09, highlight any missing safety controls (e.g., HumanGate for deletions), and propose ADR for integration."
- ・プロンプト3: "Connect MCP to the IEEE standards database for software verification protocols. Pull 2026 updates on automated Verify Gates, map them to our Part10 definitions (Fast/Full Verify), and fill gaps in exception handling with evidence-based recommendations."
- ・プロンプト4: "Use MCP to query the official RAG frameworks (e.g., LangChain docs) for knowledge update workflows. Identify omissions in our RAG/ナレッジ運用 (Part16), such as automated validation, and generate a patchset for reinforcement."
- ・プロンプト5: "Through MCP, retrieve 2026 NIST guidelines on AI risk management. Scan for alignment with our incident handling in Part19, pinpoint gaps in reoccurrence prevention (e.g., audit loops), and draft a checklist addition."

## 1. RAG/ナレッジ運用の強化案（更新・検証・証跡）

- ・**更新強化:** 採用agentic RAG (Towards AI記事に基づき、re-rankingとcontextual retrievalを組み合わせ)。更新時はMCPで外部一次情報を注入し、SSOT (docs/) のみを対象に最小差分適用 (Part14準拠)。自動化ツールとしてHaystack (haystack.deepset.ai公式) を使用し、更新前にHumanGate承認を必須化。
- ・**検証強化:** Advanced RAG techniques (Neo4jブログ) でクエリ拡張とハイブリッド検索を導入。Fast Verify (Part10) と統合し、RAG出力の正確性を機械判定 (e.g., cosine similarity threshold >0.85)。検証失敗時はVRループ (Part10) を回し、3回以内で解決。
- ・**証跡強化:** 各RAGクエリ/更新にEvidence Pack生成 (Part12準拠)：クエリログ、retrieved chunks、出力diff、sha256ハッシュ。保存先をevidence/rag\_logs/に固定し、MCP経由の外部参照をmanifestに記録。監査時はCycloneDX SBOM (cyclonedx.org公式) で依存追跡を追加し、再現性を担保。

1. 設計書へ追記する文章案（そのまま貼れる）

**Part21: ツール統合運用 (IDE/CLI/エージェント/MCP/ZAI/RAGの役割分担)**

## 0. このPartの位置づけ

- ・目的: IDE/CLI/エージェント/MCP/ZAI/RAGを組み合わせ、精度・コスパ・事故防止・スピードを最大化する運用を定義。
- ・依存: Part00 (SSOT憲法)、Part09 (Permission Tier)、Part10 (Verify Gate)、Part14 (変更管理)。
- ・影響: 全Partのツール使用規約、Evidence生成。

## 1. 目的 (Purpose)

ツールを役割分担し、SSOT運用を最適化。MCPで外部根拠を注入、RAGで内部参照精度を向上。

## 2. 適用範囲 (Scope / Out of Scope)

Scope: ツールの役割分担、プロンプト例、RAG強化。

Out of Scope: ツールインストール詳細 (別ドキュメント)。

## 3. 前提 (Assumptions)

1. MCPは外部接続標準 (Anthropic公式準拠)。
2. ZAIは一般AI支援 (z.ai公式)。

## 4. 用語 (Glossary参照 : Part02)

- ・MCP: Model Context Protocol（外部システム接続）。
- ・ZAI: Z.ai（AIチャット支援）。

## 5. ルール（MUST / MUST NOT / SHOULD）

R-2101: ツール分担遵守【MUST】作業は表の推奨ツールを使用。

R-2102: MCP外部アクセス【MUST】一次情報のみ注入。

R-2103: RAG更新Verify【MUST】更新後Fast Verify必須。

## 6. 手順

1. 作業分類: 表を参照。
2. MCP使用: プロンプトで抜け埋め。
3. RAG更新: 強化案に従う。

## 7. 例外処理

- ・MCP失敗: 手動一次情報確認、HumanGateエスカレーション。

## 8. 機械判定（Verify観点）

V-2101: ツール使用ログ確認 合否: ログ存在でPASS。

## 9. 監査観点（Evidence）

E-2101: MCP/RAGログ 保存先: evidence/tool\_logs/。

## 10. チェックリスト

- .□ ツール分担表を確認。
- .□ MCPプロンプトで抜け埋め。

## 11. 未決事項

U-2101: ツールAPI費用上限。

## 12. 参照

- Part00, Part10, MCP公式 ([modelcontextprotocol.io](https://modelcontextprotocol.io))。
  1. 導入リスクと回避策 (API費用、誤情報、権限、再現性)
- API費用: リスク: MCP/ZAI/RAGのクエリ多用でコスト増 (Anthropic/Z.ai公式でトークン課金)。回避策: クエリ最適化 (バッチ処理) と無料ティア優先 (Z.ai無料版)。月上限設定 (e.g., \$50) でHumanGate承認必須。
- 誤情報: リスク: MCP経由の外部データ汚染やRAGのhallucination (Towards AI記事で指摘)。回避策: 一次情報限定 (公式APIのみ) と出力Verify (Part10のFast Verifyでsimilarityチェック)。誤情報検出時はVRループで修復。
- 権限: リスク: IDEエージェント/MCPの過剰アクセスでセキュリティ事故 (Part09のPermission Tier違反)。回避策: Tierベース運用 (ReadOnlyから開始)。MCP接続はHumanGate承認し、権限最小化 (e.g., readonly APIキー)。
- 再現性: リスク: AI出力の非決定性で証跡不整合 (Neo4j RAG記事で指摘)。回避策: シード固定とログ保存 (Evidence Packにクエリ/レスポンス/seed記録)。オフラインCLI優先で依存低減。