

1. 重大な矛盾/欠落 (P0)

- **用語集が複数箇所に存在:** docs/Part02 と glossary/GLOSSARY.md で用語を別々に定義しており、SSOT（一元管理）原則に反するおそれがある ①。用語は一箇所に集約すべきで、複数ソースの不整合を避ける必要がある ①。
- **Part14 見出しの重複:** Part14 が2回登場しており（ファイル行4699行と5089行）、番号の整合性が失われている。設計書内で同一パート番号が重複すると真実順序が乱れ、参照混乱の原因となる。

2. 改善推奨 (P1/P2)

- **CI/CD連携の明記:** GitHub ActionsなどCIツールと連携し、Verify Gateを自動実行・ブランチ保護による必須ステータスチェックにすることを明示する ②。これにより手動ミスを防ぎ、Verify忘れを防止できる（必須チェックの設定例 ②）。
- **SBOMフォーマットの規格化:** SBOMの標準フォーマットとしてOWASP CycloneDX v1.5を推奨する記述を追加する。CycloneDXは業界標準のBOM形式であり、多くの企業や政府機関で採用されている ③。最新仕様（v1.5）を利用するよう明示すれば、工具・互換性の混乱を防止できる ③。
- **セマンティック・バージョンニング採用:** バージョン管理ルールとしてSemVer (MAJOR.MINOR.PATCH)を採用することを明記する ④。これにより互換性破壊の変更でメジャーバージョンを上げる規則が明確になり、CHANGELOGの有用性が向上する ④。
- **AIエージェントの最新モデル活用:** Core4やAI Packの説明に、「可能なら最新かつ高性能なモデル（例：GPT-5）を優先して使用する」旨を追記する ⑤。最新モデルは信頼性が高いため、AIによる誤作動・推論エラーを低減できる ⑤。
- **連鎖的思考（Chain-of-Thought）の促進:** 複雑なタスクではAIにステップごとの思考過程を明示させる手法（Chain-of-Thought）を運用手順に組み込むことを検討する ⑥。プロンプト設計に構造的推論を含めることで、AIの誤判断を減らし安全性・透明性を向上できる ⑥。

3. 具体的修正案 (Patch案)

- **file:** docs/Part14.md
change: 「1. 目的」セクションの箇条書き末尾に「6. バージョン番号: Semantic Versioning (X.Y.Z)を採用し、後方互換性を破る変更ではメジャーバージョンを上げる ④。」を追加。
reason: バージョン付けルールを明確化し、CHANGELOGの信頼性を高めるため ④。
- **file:** docs/Part14.md
change: 重複している「# Part 14：変更管理」ヘッダ（2箇所目）を削除。
reason: パート番号の一意性を確保し、参照混乱を防止するため。
- **file:** docs/Part17.md
change: ルールセクションに「### R-1707: CI/CD連携【SHOULD】 - Verify GateをGitHub Actions等で自動実行し、ブランチ保護でステータスチェックを必須とする。」を追加。
reason: CI環境での自動検証とブランチ保護で人為的ミス（Verify忘れ）を防ぎ、運用の信頼性を向上させるため ②。
- **file:** docs/Part13.md
change: 「1. 目的」セクションの内容にSBOM標準フォーマットの追記として「推奨SBOMフォー

マット: CycloneDX v1.5 を使用する。」を追加。

reason: 業界で広く採用されているCycloneDX最新仕様をSBOM標準とすることで、ツール・規格の不整合リスクを低減するため³。

• **file:** docs/Part03.md

change: AI Packの「適用範囲」セクションに次の箇条書きを追加：

- **最新モデル使用:** 利用可能な最新モデル（例：GPT-5）を優先して使用する⁵。
- **Chain-of-Thought:** 複雑なタスクには逐次的推論（Chain-of-Thought）を促す設計とする⁶。
reason: 最新の高性能モデルと構造化推論を活用することで、AIエージェントの誤作動を抑え、信頼性・透明性を高めるため⁵⁶。

4. “運用で事故る” 代表シナリオ3つ＋予防策

- **(1) Verify Gate抜けのミスコミット:** 開発者がVerifyチェックを実行せずコミットしてしまい、SSOT参照の破壊や欠陥コードが本流に入る事故。
予防策: GitHubのブランチ保護でステータスチェック（Fast Verifyの結果）を必須化し、プルリクエスト/マージ前に自動検証を行う²。また、checks/ スクリプトをCI連携して自動実行し、人為的な実行漏れを排除する。
- **(2) AIの権限逸脱:** AIエージェント（GPT等）が指示ミスや仕様未確認でsources/以下やCLAUDE.mdなどのSSOTファイルを変更し、根拠なく内容を書き換えてしまう事故。
予防策: Permission Tierを厳格に適用し、sources/ 改変など破壊的操作は必ずHumanGate承認の手続きを経る。また、AIエージェントにはエラー率の低い最新モデルを使用し、高リスク判断は人間にエスカレーションする仕組みとする⁷。プロンプトにも「推測禁止」を徹底し、不明点は必ず未解決事項として扱う。
- **(3) 用語/仕様の不明瞭による実装ミス:** ドキュメントの曖昧な記述や用語定義の欠落をAIや開発者が独自解釈して実装し、意図に反する動作となる事故。例：曖昧なプロセス定義を「たぶんこうだろう」と実装に反映する。
予防策: 「推測禁止・未決事項ルール」を厳守し、曖昧な点は必ず未解決事項として記録してから仕様を確定する。また、AIにはチェーン・オブ・ソート（逐次的推論）を促して設計意図を明確化させる（誤解釈を減らす）⁶。仕様凍結前にコードを書くことを禁止し、明文化された意思決定（ADRやFACTS_LEDGER）に基づいて実装する。

5. 未決事項リスト＋確認手順

- **U-0004: Verify実行タイミング** – 「コミット前」「プッシュ前」「CI内実行」のいずれが正しいか未定義。**確認方法:** CI/CD担当者と相談し、正式に「プッシュ時に自動実行＋ブランチ保護」を採用するか合意する。
- **U-0102: SBOM出力フォーマット** – CycloneDXかSPDXのいずれを標準とするか未確定。**確認方法:** CycloneDX v1.5仕様書やSBOM実装ツールを参照し、プロジェクト要件に合う方を選定・ドキュメント化する³。
- **U-0104: AI事故時の責任範囲** – AIが誤動作した場合の責任者・対応者が未定義。**確認方法:** ガバナンス文書またはADRで明文化する。具体的には、開発チームや運用チームに責任分界点を定め、AIによる変更には常にHumanGateが介在する運用にする。
- **U-1402: チェックスクリプトの自動化** – checks/verify_repo.ps1 の手動実行かCI連携か未定。**確認方法:** 現行運用実績をレビューし、初期は手動運用としつつ、安定後にCI連携を検討するADRを起案する（影響・コストを評価）。

- **U-1404: セマンティックバージョニング** – バージョン付けルールが未定義。**確認方法:** SemVer標準 (MAJOR.MINOR.PATCH) を参考にルール案を作成し、仕様凍結版に追記する ⁴。
- **その他:** 各未決項目は `FACTS_LEDGER` のU-XXXXセクションに整理し、必要に応じて関係者への問い合わせや公式仕様書での確認（例：SBOM標準、CIベストプラクティスなど）を実施する。

参考資料: CycloneDX v1.5リリース ³、GitHubブランチ保護設定 ²、AIエージェント設計ガイド ⁵ ⁶、Semantic Versioning 2.0.0仕様 ⁴。

¹ Single Source of Truth [+ How to Build One] | The Workstream

<https://www.atlassian.com/work-management/knowledge-sharing/documentation/building-a-single-source-of-truth-ssot-for-your-team>

² About protected branches - GitHub Docs

<https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/managing-protected-branches/about-protected-branches>

³ Introducing OWASP CycloneDX v1.5 - Advanced Bill of Materials Standard Empowering Transparency, Security, and Compliance | CycloneDX

<https://cyclonedx.org/news/cyclonedx-v1.5-released/>

⁴ Semantic Versioning 2.0.0 | Semantic Versioning

<https://semver.org/>

⁵ ⁶ ⁷ Technical Tuesday: 10 best practices for building reliable AI agents in 2025 | UiPath

<https://www.uipath.com/blog/ai/agent-builder-best-practices>