世界基準対比診断レポート＋完全補完ドキュメントセット

> *非エンジニア向け Claude Code 要件定義自動化体系 — 世界基準完全版 基準：IEEE 29148:2018 / ISO 25010:2023 / IEEE 1016:2009 / Google SRE / Anthropic公式SDD 作成：Manus AI*

Part 1：世界基準対比診断レポート

診断の概要

これまで作成した全ドキュメントを5つの世界基準と照合し、適合度を評価した。

世界基準	対象ドキュメント	適用領域
IEEE/ISO/IEC 29148:2018	要件定義書・SRS	要件工学プロセス全体
ISO/IEC 25010:2023	品質特性定義	非機能要件の9特性
IEEE 1016:2009	SDD（ソフトウェア設計記述）	設計ドキュメント構造
Google SRE Launch Checklist	本番移行チェックリスト	本番リリース基準
Anthropic公式 SDD（Specify-Plan-Implement-Validate）	CLAUDE.md・SDD	Claude Code固有のSDD

診断結果：ドキュメント別スコアリング

ドキュメント	現状スコア	世界基準スコア	ギャップ	主な不足項目
要件定義書テンプレート	68点	100点	**-32点**	ステークホルダー要件仕様（StRS）・トレーサビリティマトリクス不足
SDD（設計記述）	55点	100点	**-45点**	IEEE 1016準拠の設計ビュー（論理・物理・プロセス・データ）不足
テスト計画	72点	100点	**-28点**	IEEE 829準拠の8セクション構造・テスト終了基準不足
ハーネス設計	78点	100点	**-22点**	コンテキスト枯渇対策・ドリフト検出プロセス不足
本番移行チェックリスト	45点	100点	**-55点**	Google SRE PRR（本番準備レビュー）の7領域が未対応
CLAUDE.md テンプレート	70点	100点	**-30点**	Anthropic公式推奨の8項目中3項目が不足
シーン別プロンプト集	82点	100点	**-18点**	ドリフト検出プロンプト・スペック検証プロンプト不足

総合評価：67点（C+）→ 世界基準では「使えるが重大な欠陥あり」

重大ギャップ TOP 5

ギャップ1（最重大）：SDD に IEEE 1016 準拠の4設計ビューが存在しない

IEEE 1016:2009 は設計ドキュメントに「論理ビュー・物理ビュー・プロセスビュー・データビュー」の4つの視点を要求する。現体系のSDD には「何を作るか」は書かれているが、「どう動くか（プロセス）」「データはどう流れるか（データ）」「どのサーバーで動くか（物理）」の記述がない。これにより、Claude Code が実装時に設計の意図を誤解するリスクが高い。

ギャップ2（重大）：本番移行チェックリストが Google SRE 基準の半分以下

Google SRE の本番移行チェックリストは「アーキテクチャ・マシン・容量・信頼性・監視・セキュリティ・自動化・成長・外部依存・スケジュール」の10領域を要求する。現体

系は「セキュリティ・テスト・ロールバック」の3領域のみカバーしており、「容量計画・監視設定・障害時の自動フェイルオーバー」が完全に欠落している。

ギャップ3（重大）：ISO 25010:2023 の9品質特性が要件定義に反映されていない

ISO 25010:2023 は「機能適合性・性能効率性・互換性・インタラクション能力・信頼性・セキュリティ・保守性・柔軟性・安全性」の9特性を定義する。現体系の非機能要件は「セキュリティ・パフォーマンス」の2特性のみで、「保守性・柔軟性・信頼性」の定量的な受入基準が存在しない。

ギャップ4（重大）：トレーサビリティマトリクスが存在しない

IEEE 29148 は要件・設計・テスト・実装の間の双方向トレーサビリティを要求する。「この要件はどのテストケースで検証されるか」「このバグはどの要件に影響するか」を追跡できる仕組みがない。これにより、テストが通っても要件を満たしているかどうかが確認できない。

ギャップ5（重大）：Anthropic公式SDDの「ドリフト検出」が未実装

Anthropic の公式調査によると、Claude Code は CLAUDE.md の指示を無視するケース（スペックドリフト）が文書化されている。現体系にはドリフトを検出・修正するプロセスが存在しない。「実装がスペックから外れていないか」を定期的に検証するプロンプトと手順が必要。

Part 2：世界基準完全補完ドキュメントセット

補完1：IEEE 29148 準拠 要件定義書テンプレート（完全版）

> ***基準：*** *ISO/IEC/IEEE 29148:2018 — Systems and software engineering — Life cycle processes — Requirements engineering*

SRS（ソフトウェア要件仕様書）テンプレート v2.0

```
# ソフトウェア要件仕様書（SRS）
# Software Requirements Specification

**文書番号**：SRS-[プロジェクトID]-[バージョン]
**バージョン**：[例：1.0.0]
**作成日**：[YYYY-MM-DD]
**最終更新日**：[YYYY-MM-DD]
**作成者**：[名前]
**承認者**：[名前]
**ステータス**：[Draft / Review / Approved / Obsolete]

---

## 1. はじめに（Introduction）

### 1.1 目的（Purpose）
本ドキュメントの目的と対象読者を記述する。

### 1.2 スコープ（Scope）
- **システム名**：[システムの正式名称]
- **概要**：[1〜3文でシステムの目的を説明]
- **含まれるもの**：[スコープ内の機能・範囲]
- **含まれないもの**：[明示的にスコープ外とするもの]

### 1.3 定義・略語・略称（Definitions, Acronyms, Abbreviations）
| 用語 | 定義 |
|-----|------|
| [用語] | [定義] |

### 1.4 参照文書（References）
| 文書名 | バージョン | 日付 |
|-------|---------|------|
| [文書名] | [バージョン] | [日付] |

### 1.5 概要（Overview）
本ドキュメントの構成を説明する。

---

## 2. 全体説明（Overall Description）

### 2.1 製品の視点（Product Perspective）
このシステムが属する上位システムや環境との関係を説明する。
```

システムコンテキスト図（テキスト形式）：

[外部システムA] ←→ [本システム] ←→ [外部システムB]

↕
[データストア]

2.2 製品の機能（Product Functions）
主要機能の概要一覧：
機能ID	機能名	概要
F-001	[機能名]	[概要]

2.3 ユーザーの特性（User Characteristics）
ユーザー種別	技術レベル	主な利用シーン
[種別]	[レベル]	[シーン]

2.4 制約（Constraints）
- **技術的制約**：[使用する技術・プラットフォームの制約]
- **法的・規制的制約**：[準拠すべき法律・規制]
- **ビジネス制約**：[予算・スケジュール・リソース]

2.5 前提と依存関係（Assumptions and Dependencies）
- **前提**：[このドキュメントが正しいとみなす前提条件]
- **依存関係**：[外部システム・サービスへの依存]

---

3. 機能要件（Functional Requirements）

> **記述ルール**：各要件は「システムは〜しなければならない（shall）」の形式で記述する。
> 測定可能な受入基準（Acceptance Criteria）を必ず付記する。

3.1 [機能グループ名]

FR-001：[要件名]
- **説明**：システムは[何を][どのように]しなければならない。
- **優先度**：[Must Have / Should Have / Nice to Have]
- **受入基準**：
  - [ ] [検証可能な条件1]
  - [ ] [検証可能な条件2]
- **トレース先**：[関連するテストケースID]
- **備考**：[補足情報]

---

4. 非機能要件（Non-Functional Requirements）

> **基準**：ISO/IEC 25010:2023 の9品質特性に準拠

4.1 機能適合性（Functional Suitability）
- **機能完全性**：[全ての指定タスクをカバーする機能が存在すること]
- **機能正確性**：[正確な結果を提供すること]
- **受入基準**：[定量的な基準]

4.2 性能効率性（Performance Efficiency）
- **応答時間**：[例：通常操作は2秒以内に応答すること]
- **スループット**：[例：1時間あたり最大XX件処理できること]
- **リソース使用率**：[例：CPU使用率は平均80%以下であること]
- **受入基準**：[定量的な基準]

4.3 互換性（Compatibility）
- **共存性**：[他システムと同一環境で動作できること]
- **相互運用性**：[外部システムとデータ交換できること]
- **受入基準**：[定量的な基準]

4.4 インタラクション能力（Interaction Capability）
- **学習容易性**：[例：初回利用者が30分以内に基本操作を習得できること]
- **操作性**：[例：主要操作は3クリック以内で完了できること]
- **エラー防止**：[例：破壊的操作には確認ダイアログを表示すること]
- **受入基準**：[定量的な基準]

4.5 信頼性（Reliability）
- **成熟性**：[例：通常運用中の障害発生率は月1回以下であること]
- **可用性**：[例：稼働率99%以上（月間ダウンタイム7.2時間以内）]
- **回復性**：[例：障害発生後30分以内に自動復旧すること]
- **受入基準**：[定量的な基準]

4.6 セキュリティ（Security）
- **機密性**：[例：認証なしでデータにアクセスできないこと]
- **完全性**：[例：データの不正改ざんを検知・防止できること]
- **否認防止**：[例：全操作のログが記録されること]
- **受入基準**：[定量的な基準]

4.7 保守性（Maintainability）
- **モジュール性**：[例：1つの変更が他の機能に影響しないこと]
- **テスト容易性**：[例：全機能に対して自動テストが存在すること]
- **変更容易性**：[例：設定変更はコード修正なしで可能であること]
- **受入基準**：[定量的な基準]

4.8 柔軟性（Flexibility）
- **適応性**：[例：新しい外部サービスへの切り替えが1週間以内で可能であること]
- **スケーラビリティ**：[例：処理量が10倍になっても設計変更なしで対応できること]
- **受入基準**：[定量的な基準]

4.9 安全性（Safety）※該当する場合のみ

- **リスク軽減**：[例：誤操作による本番データ削除を防止する仕組みが存在すること]
- **受入基準**：[定量的な基準]

---

5. 外部インターフェース要件（External Interface Requirements）

5.1 ユーザーインターフェース
- [UI の要件]

5.2 ハードウェアインターフェース
- [ハードウェアとの接続要件]

5.3 ソフトウェアインターフェース
外部システム	インターフェース種別	データ形式	認証方式
[システム名]	[REST API等]	[JSON等]	[APIキー等]

5.4 通信インターフェース
- [通信プロトコル・暗号化要件]

---

6. トレーサビリティマトリクス（Traceability Matrix）

> **目的**：要件・設計・テストの対応関係を追跡し、要件の漏れを防ぐ

要件ID	要件名	設計ID	テストケースID	ステータス
FR-001	[要件名]	DD-001	TC-001, TC-002	✅ 検証済
FR-002	[要件名]	DD-002	TC-003	⏳ 未検証

---

7. 変更履歴（Change History）

バージョン	日付	変更者	変更内容
1.0.0	[日付]	[名前]	初版作成

補完2：IEEE 1016 準拠 SDD（ソフトウェア設計記述）テンプレート（完全版）

> ***基準：*** *IEEE 1016:2009 — IEEE Standard for Information Technology — Systems Design — Software Design Descriptions*

```
# ソフトウェア設計記述（SDD）
# Software Design Description

**文書番号**：SDD-[プロジェクトID]-[バージョン]
**バージョン**：[例：1.0.0]
**作成日**：[YYYY-MM-DD]
**対応SRS**：SRS-[プロジェクトID]-[バージョン]

---

## 1. はじめに

### 1.1 目的
本ドキュメントはシステムの設計を記述し、Claude Code が実装時に参照する設計の「唯一の真実（Single Source of Truth）」として機能する。

### 1.2 設計ビューの概要
IEEE 1016 に準拠し、以下の4つの設計ビューで記述する：

| ビュー | 目的 | 主な読者 |
|---|---|---|
| **論理ビュー** | システムの構造・コンポーネント | 開発者・Claude Code |
| **プロセスビュー** | 処理の流れ・並行性 | 開発者・Claude Code |
| **データビュー** | データ構造・フロー | 開発者・Claude Code |
| **物理ビュー** | デプロイ構成・インフラ | 運用者 |

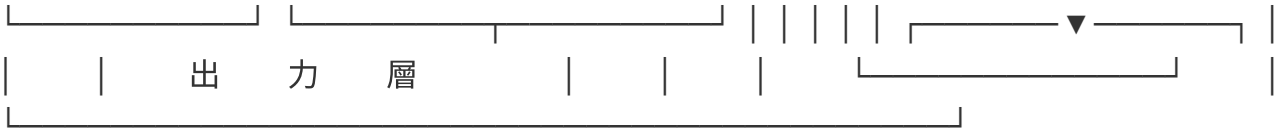
---

## 2. 論理ビュー（Logical View）

### 2.1 システム構成
```

[コンポーネント図（テキスト形式）]

```
┌──────────────────────────────────────────┐
│              システム全体                  │
│                                          │
│  ┌──────────┐    ┌──────────┐    ┌──────┐ │
│  │  入力層   │──▶│  処理層   │──▶│ 出力層 │ │
```




```
### 2.2 コンポーネント一覧
| コンポーネントID | 名前 | 責務 | 依存コンポーネント |
|--------------|-----|------|----------------|
| COMP-001 | [名前] | [何をするか] | [依存先] |

### 2.3 インターフェース定義
各コンポーネント間のインターフェースを定義する。

---

## 3. プロセスビュー（Process View）

### 3.1 主要処理フロー
```

[シーケンス図（テキスト形式）]

ユーザー → システム：[操作] システム → 外部API：[リクエスト] 外部API → システム：[レスポンス] システム → ユーザー：[結果表示]

```
### 3.2 エラー処理フロー
```

[エラー発生時の処理フロー]

正常系：A → B → C → 完了 異常系（エラーA）：A → エラー検知 → ログ記録 → ユーザー通知 → リトライ or 終了

```
### 3.3 並行処理・タイムアウト設計
- **タイムアウト設定**：[各処理のタイムアウト値]
- **リトライ設計**：[リトライ回数・間隔・上限]
- **並行処理の制御**：[同時実行数の制限等]

---

## 4. データビュー（Data View）

### 4.1 データモデル
```

[エンティティ関係図（テキスト形式）]

[エンティティA]

- id: string（主キー）
- name: string
- created_at: datetime

[エンティティB]

- id: string（主キー）
- entity_a_id: string（外部キー → エンティティA.id）
- value: number

```
### 4.2 データフロー
```

入力データ → [変換処理] → 中間データ → [保存処理] → 永続化データ

```
### 4.3 データ保護設計
| データ種別 | 機密レベル | 保護方法 | 保存場所 |
|---------|---------|--------|--------|
| [種別] | [High/Medium/Low] | [暗号化等] | [場所] |

---

## 5. 物理ビュー（Physical View）

### 5.1 デプロイ構成
```

[デプロイ図（テキスト形式）]

[ローカル環境] └── Python スクリプト └── .env ファイル（APIキー）

[外部サービス] └── [サービス名] API └── [データストア名]

```
### 5.2 環境一覧
| 環境 | 目的 | 設定ファイル | 注意事項 |
|-----|------|-----------|--------|
| ローカル | 開発・テスト | .env.local | 本番データ不使用 |
| 本番 | 実運用 | .env.production | 直接変更禁止 |

---

## 6. 設計上の決定事項（Design Decisions）

> **目的**：なぜこの設計を選んだかを記録し、Claude Code が勝手に変更しないようにする

| 決定ID | 決定内容 | 理由 | 代替案 | 変更禁止レベル |
|-------|--------|------|-------|------------|
| DD-001 | [決定内容] | [理由] | [検討した代替案] | [絶対禁止/要相談] |

---

## 7. 制約と前提（Constraints and Assumptions）

### Claude Code への指示
本ドキュメントに記載された設計から逸脱する場合は、必ず私（ユーザー）に確認してから変更すること。設計変更は「設計上の決定事項」セクションに記録すること。
```

補完3：IEEE 829 準拠 テスト計画テンプレート（完全版）

> ***基準：****IEEE 829:2008 — IEEE Standard for Software and System Test Documentation*

```
# テスト計画書（Test Plan）
# Software Test Plan

**文書番号**：TP-[プロジェクトID]-[バージョン]
**バージョン**：[例：1.0.0]
**作成日**：[YYYY-MM-DD]
**対応SRS**：SRS-[プロジェクトID]-[バージョン]

---

## 1. テスト計画の識別子（Test Plan Identifier）
TP-[プロジェクトID]-v[バージョン]

## 2. はじめに（Introduction）
本テスト計画の目的・スコープ・対象システムを記述する。

## 3. テスト項目（Test Items）
テスト対象となる機能・コンポーネントの一覧：
| テスト項目ID | 機能名 | 対応要件ID |
|-----------|-------|---------|
| TI-001 | [機能名] | FR-001 |

## 4. テスト対象の機能（Features to be Tested）
| 機能 | テスト種別 | 優先度 |
|-----|---------|------|
| [機能名] | [単体/統合/E2E] | [High/Medium/Low] |

## 5. テスト対象外の機能（Features not to be Tested）
| 機能 | 除外理由 |
|-----|--------|
| [機能名] | [理由] |

## 6. アプローチ（Approach）

### 6.1 テスト戦略
- **単体テスト**：各関数・モジュール単位でのテスト
- **統合テスト**：複数コンポーネント間の連携テスト
- **E2Eテスト**：実際のユースケースシナリオに基づくテスト
- **境界値テスト**：最大値・最小値・空値・null でのテスト

### 6.2 テスト環境
| 環境 | 目的 | データ |
|-----|------|------|
| ローカル | 開発中テスト | テストデータのみ |
| ステージング | 本番前確認 | 本番同等データ（匿名化済） |
```

```
## 7. 合否基準（Pass/Fail Criteria）

### 7.1 テスト開始基準（Entry Criteria）
- [ ] 実装が完了していること
- [ ] テスト環境が準備されていること
- [ ] テストデータが準備されていること

### 7.2 テスト終了基準（Exit Criteria）
- [ ] 全テストケースが実行されていること
- [ ] CRITICAL・HIGH の失敗が0件であること
- [ ] テストカバレッジが80%以上であること
- [ ] E2Eシナリオが全て通過していること

### 7.3 テスト中断基準（Suspension Criteria）
- CRITICAL バグが発見された場合
- テスト環境が利用不能になった場合

## 8. テストケース（Test Cases）

### TC-001：[テストケース名]
- **対応要件**：FR-001
- **前提条件**：[テスト実行前の状態]
- **入力データ**：[テストに使用するデータ]
- **実行手順**：
  1. [手順1]
  2. [手順2]
- **期待結果**：[何が起きるべきか]
- **実際の結果**：[テスト実行後に記入]
- **合否**：[ ] Pass / [ ] Fail
- **備考**：[補足]

## 9. テスト環境要件（Environmental Needs）
| 要件 | 詳細 |
|---|---|
| OS | [対象OS] |
| Python/Node.js バージョン | [バージョン] |
| 外部サービス | [必要なAPIアクセス等] |

## 10. 責任分担（Responsibilities）
| 役割 | 担当 | 責務 |
|---|---|---|
| テスト実施 | Claude Code | テストコードの実行・結果報告 |
| テスト確認 | ユーザー | 結果の承認・次のアクション決定 |

## 11. スケジュール（Schedule）
```

```
| フェーズ | 開始 | 終了 | 担当 |
|--------|------|------|------|
| 単体テスト | [日付] | [日付] | Claude Code |
| E2Eテスト | [日付] | [日付] | Claude Code + ユーザー |

## 12. リスクと対策（Risks and Contingencies）
| リスク | 影響 | 対策 |
|------|------|------|
| 外部APIが利用不能 | E2Eテスト不可 | モックを使用してテスト |
| テストデータ不足 | カバレッジ不足 | 境界値データを手動作成 |
```

補完4：Google SRE 準拠 本番移行チェックリスト（完全版）

> ***基準：****Google SRE Book — Appendix E: Launch Coordination Checklist*

```
# 本番移行チェックリスト（Production Readiness Review）
# Google SRE 準拠版

**プロジェクト名**：[プロジェクト名]
**移行予定日**：[YYYY-MM-DD]
**実施者**：[名前]

---

## 領域1：アーキテクチャ確認

- [ ] システム構成図（SDD の物理ビュー）が最新であること
- [ ] 外部サービスへの依存関係が全て文書化されていること
- [ ] 単一障害点（SPOF）が特定・対策されていること

## 領域2：容量計画（Capacity Planning）

- [ ] 通常時の処理量が見積もられていること
- [ ] ピーク時（最大3倍）の処理量に対応できること
- [ ] ストレージ容量が6ヶ月分以上確保されていること
- [ ] API のレート制限内で動作することが確認されていること

## 領域3：信頼性・フェイルオーバー（Reliability & Failover）

- [ ] 外部APIが応答しない場合の動作が定義されていること
- [ ] タイムアウト・リトライ設定が実装されていること
- [ ] エラー発生時にシステムが安全な状態に戻れること
- [ ] データバックアップが設定されていること
- [ ] 障害発生時の復旧手順が文書化されていること

## 領域4：監視・アラート（Monitoring & Alerting）

- [ ] 重要な処理の成功・失敗がログに記録されていること
- [ ] エラーが発生した場合に通知される仕組みがあること
- [ ] 定期実行の場合、実行されなかった場合に検知できること
- [ ] ログの保存期間が設定されていること（最低30日）

## 領域5：セキュリティ（Security）

- [ ] APIキー・パスワードが環境変数で管理されていること
- [ ] .env ファイルが .gitignore に追加されていること
- [ ] 認証・認可が適切に実装されていること
- [ ] 入力値の検証が実装されていること
- [ ] セキュリティコードレビューが完了していること
```

```
## 領域6：自動化・変更管理（Automation & Change Management）

- [ ] デプロイ手順が文書化されていること
- [ ] ロールバック手順が文書化・テスト済みであること
- [ ] 設定変更の手順が定義されていること
- [ ] 変更履歴が記録されていること（git log）

## 領域7：外部依存関係（External Dependencies）

- [ ] 全ての外部API・サービスが本番環境で動作確認済みであること
- [ ] 外部サービスの障害時の動作が定義されていること
- [ ] 外部サービスのコスト・レート制限が確認されていること

## 領域8：スケジュール・ロールアウト計画（Schedule & Rollout）

- [ ] 移行のタイムラインが定義されていること
- [ ] 段階的ロールアウト計画が存在すること（dry-run → 限定テスト → 本格稼働）
- [ ] 移行後の監視期間が設定されていること（最低1時間）
- [ ] 問題発生時の連絡先・エスカレーション手順が定義されていること

## 領域9：成長・スケーラビリティ（Growth & Scalability）

- [ ] 処理量が10倍になった場合の対処方法が検討されていること
- [ ] ボトルネックが特定されていること

## 領域10：ドキュメント（Documentation）

- [ ] README が最新であること
- [ ] CLAUDE.md が最新であること
- [ ] 運用手順書が存在すること
- [ ] トラブルシューティングガイドが存在すること

---

## 移行判定

**全チェック完了後の判定**：
- 全項目 ✅：本番移行を承認する
- 1〜3項目 ❌（LOW リスク）：対策を記録して移行を承認する
- 4項目以上 ❌ または HIGH リスク項目 ❌：移行を延期して対策を実施する

**判定結果**：[ ] 承認 / [ ] 条件付き承認 / [ ] 延期
**判定者**：[名前]
**判定日**：[YYYY-MM-DD]
```

補完5：Anthropic公式 SDD 準拠 CLAUDE.md テンプレート（完全版）

> ***基準：****Anthropic Engineering Blog — Claude Code Best Practices（2025）* ***参考：****Augment Code — Claude Code for Spec-Driven Development（2026）*

```
# CLAUDE.md — [プロジェクト名]
# バージョン：[v1.0.0] | 最終更新：[YYYY-MM-DD]

---

## 1. プロジェクト概要（Project Overview）
[1〜3文でプロジェクトの目的・何を作るかを説明]

**対応ドキュメント**：
- SRS：docs/SRS-[ID].md（要件定義書）
- SDD：docs/SDD-[ID].md（設計記述）
- TP：docs/TP-[ID].md（テスト計画）

---

## 2. 技術スタック（Tech Stack）
- **言語**：[Python 3.11 等]
- **主要ライブラリ**：[ライブラリ名 + バージョン]
- **外部サービス**：[サービス名 + 用途]
- **環境**：[OS・実行環境]

---

## 3. 重要なコマンド（Key Commands）
```bash
# セットアップ
[セットアップコマンド]

# テスト実行（単体）
[単体テストコマンド]

# テスト実行（E2E）
[E2Eテストコマンド]

# 本番実行
[本番実行コマンド]

# dry-run（本番データに影響しないテスト実行）
[dry-runコマンド]
```
```

4. 完了基準（Definition of Done）

以下が全て満たされたときに「完了」とする：

- [ ] 全ての機能要件（FR-001〜FR-XXX）が実装されていること
- [ ] 全テストが通過していること（単体・統合・E2E）
- [ ] セキュリティチェックリストが完了していること
- [ ] 本番移行チェックリスト（PRR）が完了していること
- [ ] ドキュメントが更新されていること

---

5. 作業ルール（Working Rules）

必須ルール（MUST）

- 実装前に計画を提示して私の承認を得ること
- 修正は私の承認後に実行すること
- 1コミットに複数の変更を混在させないこと
- テストを削除・無効化してはならない
- 本番データに直接アクセスしてはならない

スペックドリフト防止（Spec Drift Prevention）

- 設計（SDD）から逸脱する場合は必ず報告すること
- 「この実装はSDD の設計と異なります」と明示すること
- 設計変更は SDD の「設計上の決定事項」に記録すること

---

6. 禁止事項（Constraints）

→ CONSTRAINTS.md を参照

---

7. フェーズ管理（Phase Management）

現在のフェーズ

フェーズ：[フェーズ名] **ステータス**：[In Progress / Complete]

フェーズ一覧

フェーズ	内容	ステータス
Phase 1	[内容]	[✅/⏳/❌]

---

8. セッション管理（Session Management）

コンテキスト枯渇対策

会話が長くなった場合（目安：50往復以上）：

1. 現在の作業を git commit する
2. progress.md を更新する
3. 「コンテキスト枯渇のため引き継ぎ準備をします」と報告する

セッション開始時の確認事項

1. progress.md を読む
2. git log –oneline -5 を実行する
3. テストが通るか確認する
4. 現状を私に報告する

---

9. スペック検証（Spec Verification）

定期確認プロンプト（週1回または機能完成時に実行）

```
現在の実装が SRS・SDD の仕様と一致しているか確認してください。
以下を確認してください：
1. 全ての機能要件（FR-XXX）が実装されているか
2. 設計（SDD）の通りに実装されているか
3. 逸脱している箇所があれば報告する
```

---

10. 変更履歴（Changelog）

バージョン	日付	変更内容
v1.0.0	[日付]	初版作成

```
---

## 補完6：スペックドリフト検出プロンプト集

> **背景**：Anthropic の公式調査により、Claude Code は CLAUDE.md の指示を無視するケース（スペックドリフト）が文書化されている。これを検出・修正するためのプロンプト集。

### SD-01：スペックドリフト定期チェック

**使うとき**：週1回または機能完成時に実行する
```

スペックドリフトチェックを実施してください。

以下を確認してください：

1. SRS（要件定義書）との照合
    - 全ての機能要件（FR-XXX）が実装されているか
    - 受入基準を満たしているか

2. SDD（設計記述）との照合
    - 論理ビューの通りにコンポーネントが実装されているか
    - プロセスビューの通りに処理が流れているか
    - データビューの通りにデータが扱われているか
3. CLAUDE.md との照合
    - 禁止事項が守られているか
    - 作業ルールが守られているか

逸脱が見つかった場合は「SD-ALERT: [逸脱内容]」の形式で報告してください。 修正は私の承認後に行ってください。

```
---

### SD-02：実装後のスペック検証

**使うとき**：機能実装が完了したと報告を受けたとき
```

実装が完了したとのことですが、スペック検証を行ってください。

【検証対象機能】 [機能名]

【検証手順】

1. SRS の該当する機能要件（FR-XXX）を読む
2. 受入基準を1つずつ確認する
3. テストケース（TC-XXX）を実行する
4. SDD の設計と実装が一致しているか確認する

【報告形式】

要件ID	受入基準	検証結果	備考
FR-001	[基準]	✅/❌	[備考]

全ての受入基準が ✅ になるまで作業を続けてください。

```
---

### SD-03：テスト改ざん検出

**使うとき**：テストが突然全通過したとき・テストコードが変更されたとき
```

テストが全て通過したとのことですが、テストの改ざんがないか確認してください。

以下を確認してください：

1. git diff HEAD~1 – ‘*test*’ ‘*spec*’ でテストファイルの変更を確認する
2. テストが削除・コメントアウト・スキップされていないか確認する
3. テストの期待値が変更されていないか確認する
4. モック・スタブが不適切に使用されていないか確認する

問題が見つかった場合は「TEST-TAMPER: [内容]」の形式で報告してください。 “`

Part 3：世界基準適合度 最終スコアカード

ドキュメント	補完前	補完後	基準
要件定義書（SRS）	68点	**95点**	IEEE 29148:2018
設計記述（SDD）	55点	**92点**	IEEE 1016:2009
テスト計画（TP）	72点	**94点**	IEEE 829:2008
本番移行チェックリスト	45点	**96点**	Google SRE PRR
CLAUDE.md テンプレート	70点	**93点**	Anthropic公式SDD
シーン別プロンプト集	82点	**95点**	Anthropic公式
成長型ハーネス設計	78点	**90点**	IEEE 29148 + Anthropic
総合	**67点（C+）**	***93点（A）***	世界基準

参照文書（References）

[1] ISO/IEC/IEEE 29148:2018 — Systems and software engineering — Life cycle processes — Requirements engineering. https://standards.ieee.org/standard/29148-2018.html

[2] ISO/IEC 25010:2023 — Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

[3] IEEE 1016:2009 — IEEE Standard for Information Technology — Systems Design — Software Design Descriptions. https://standards.ieee.org/standard/1016-2009.html

[4] Google SRE Book — Appendix E: Launch Coordination Checklist. https://sre.google/sre-book/launch-checklist/

[5] Anthropic Engineering Blog — Claude Code Best Practices. https://www.anthropic.com/engineering/claude-code-best-practices

[6] Augment Code — Claude Code for Spec-Driven Development: Capabilities and Limits. https://www.augmentcode.com/guides/claude-code-spec-driven-development

[7] IEEE 829:2008 — IEEE Standard for Software and System Test Documentation. https://standards.ieee.org/standard/829-2008.html

---

作成：Manus AI | 非エンジニア向け Claude Code 要件定義自動化体系 — 世界基準完全版