成長型ハーネス設計 — 完全版

> *非エンジニア向け Claude Code 開発体系 — プロジェクトをまたいで進化するハーネス設計 1プロジェクト目から始めて、使うたびに自動的に強くなる仕組み。*

なぜ「成長型」が必要か

従来のハーネス設計は「1プロジェクト使い捨て」だった。プロジェクトが終わると、そこで得た知識・テンプレート・失敗パターンは消える。

成長型ハーネスは「プロジェクトをまたいで知識が蓄積・進化する」仕組みを持つ。3つのプロジェクトを経験すれば、4つ目のプロジェクトは最初から高品質なハーネスで始められる。

3層構造

Layer 1：プロジェクト固有層（Project Layer）

各プロジェクトに固有の情報。プロジェクト終了後に削除してよい。

ファイル	内容
CLAUDE.md	プロジェクト固有設定（概要・スタック・コマンド・完了基準）
progress.md	プロジェクト固有進捗（完了タスク・次のタスク・ブロッカー）
CONSTRAINTS.md	禁止事項・注意事項
e2e_scenarios.md	E2Eテストシナリオ（プロジェクト固有）

特徴：使い捨て可。プロジェクト完了後は Layer 2・3 に知識を移してから削除する。

Layer 2：知識蓄積層（Knowledge Layer）

プロジェクトをまたいで蓄積される知識。使うたびに充実していく。

ファイル	内容
lessons_learned.md	失敗・成功パターンの記録
boundary_cases.md	発見した境界ケースの蓄積
security_patterns.md	セキュリティ対策の蓄積
error_patterns.md	よく発生するエラーと対処法
prompt_library.md	効果的だったプロンプト集

特徴：永続・蓄積。削除しない。プロジェクトが増えるほど価値が上がる。

Layer 3：テンプレート進化層（Template Layer）

使うたびに改善されるテンプレート群。バージョン管理で進化を追跡する。

ファイル	内容
CLAUDE.md.template	バージョン管理付き CLAUDE.md テンプレート
CONSTRAINTS.md.template	汎用禁止事項テンプレート
requirements_template.md	要件定義書テンプレート
sdd_template.md	SDD テンプレート
test_plan_template.md	テスト計画テンプレート
harness_template.md	ハーネス設計テンプレート
progress.md.template	進捗管理テンプレート

特徴：バージョン管理。プロジェクト完了後に改善してバージョンを上げる。

ディレクトリ構造

```
~/claude-harness/                    ← 成長型ハーネスのルート
├── README.md                        ← 使い方ガイド
├── VERSION                          ← ハーネスのバージョン（例：v1.3.0）
│
├── templates/                       ← テンプレート進化層（Layer 3）
│   ├── CLAUDE.md.template
│   ├── CONSTRAINTS.md.template
│   ├── requirements_template.md
│   ├── sdd_template.md
│   ├── test_plan_template.md
│   ├── harness_template.md
│   └── progress.md.template
│
├── knowledge/                       ← 知識蓄積層（Layer 2）
│   ├── lessons_learned.md           ← 失敗・成功パターン集
│   ├── boundary_cases.md            ← 境界ケース集
│   ├── security_patterns.md         ← セキュリティ対策集
│   ├── error_patterns.md            ← エラーパターンと対処法
│   └── prompt_library.md            ← 効果的なプロンプト集
│
├── prompts/                         ← シーン別プロンプト集（12カテゴリ）
│   └── scene_prompts_complete.md    ← 本ドキュメントと対になるプロンプト集
│
├── projects/                        ← プロジェクト記録（Layer 1の保存先）
│   ├── project_001_[名前]/
│   │   ├── summary.md               ← 概要・成果・失敗
│   │   ├── lessons.md               ← このプロジェクトで学んだこと
│   │   └── templates_used.md        ← 使用したテンプレートのバージョン
│   └── project_002_[名前]/
│
└── changelog.md                     ← ハーネス自体の変更履歴
```

成長のサイクル：プロジェクト完了後に行う5つのアクション

プロジェクトが完了したら、必ず以下の5つのアクションを実行する。これがハーネスを成長させる核心。

Action 1：レッスン記録（lessons_learned.md に追記）

```
## [プロジェクト名] - [日付]

### 成功したこと
- [何がうまくいったか]
- [なぜうまくいったか]

### 失敗したこと・つまずいたこと
- [何が問題だったか]
- [なぜ問題が起きたか]
- [次回どう対処するか]

### 発見した境界ケース
- [発見した境界ケース]（→ boundary_cases.md にも追記）

### 効果的だったプロンプト
- [効果的だったプロンプトの概要]（→ prompt_library.md にも追記）
```

Action 2：テンプレートの改善

プロジェクト中に「テンプレートのここが使いにくかった」「この項目が足りなかった」と感じた箇所を修正する。バージョンを上げる（例：v1.2 → v1.3）。

Action 3：境界ケースの追加（boundary_cases.md に追記）

```
## [カテゴリ名]

### BC-[番号]: [境界ケースの名前]
- 発見プロジェクト：[プロジェクト名]
- 状況：[どんな状況で発生するか]
- 問題：[何が起きるか]
- 対処法：[どう対処するか]
- テストケース：[テストに追加すべき内容]
```

Action 4：セキュリティパターンの追加（security_patterns.md に追記）

```
## SP-[番号]: [セキュリティパターンの名前]
- 発見プロジェクト：[プロジェクト名]
- リスク：[何のリスクか]
- 対処法：[具体的な実装方法]
- 確認方法：[テスト・チェック方法]
```

Action 5：プロジェクト記録の保存

`projects/project_[番号]_[名前]/` ディレクトリを作成して今回のプロジェクト記録を保存する。

新プロジェクト開始時の手順

Step 1：ハーネスの最新版を確認する

```
~/claude-harness/VERSION を確認する
最新の changelog.md を読む
```

Step 2：テンプレートをコピーする

```
# 新プロジェクトのディレクトリを作成
mkdir ~/projects/新プロジェクト名
cd ~/projects/新プロジェクト名

# テンプレートをコピー
cp ~/claude-harness/templates/CLAUDE.md.template ./CLAUDE.md
cp ~/claude-harness/templates/CONSTRAINTS.md.template ./CONSTRAINTS.md
cp ~/claude-harness/templates/progress.md.template ./progress.md
```

Step 3：過去の知識を参照する

```
~/claude-harness/knowledge/lessons_learned.md を読む
→ 類似プロジェクトの失敗パターンを確認する

~/claude-harness/knowledge/boundary_cases.md を読む
→ E2Eシナリオに境界ケースを追加する

~/claude-harness/knowledge/security_patterns.md を読む
→ セキュリティチェックリストに追加する
```

Step 4：プロジェクト固有の情報でテンプレートを埋める

CLAUDE.md のプレースホルダーをプロジェクト固有の情報で埋める。

Step 5：Claude Code に最初の指示を出す

`scene_prompts_complete.md` の `P-01-A` プロンプトを使用する。

テンプレートのバージョン管理

バージョン番号の意味

バージョン	意味
v1.0.0	初期版
v1.1.0	マイナー改善（項目追加・文言修正）
v1.2.0	中規模改善（構造変更・新セクション追加）
v2.0.0	メジャー改善（大幅な再設計）

バージョンアップのタイミング

- プロジェクト完了後（必ず）
- 重大な問題が発見されたとき（緊急）

- 新しいユースケースに対応するとき（随時）

changelog.md の記録形式

```
## v1.3.0 - [日付]

### 追加
- CLAUDE.md: 「コンテキスト枯渇時の対処法」セクションを追加

### 改善
- test_plan_template.md: 境界ケースのセクションを拡充

### 修正
- sdd_template.md: エラー処理の記載例を修正

### 背景
- project_002 でコンテキスト枯渇が発生したため対策を追加
```

成長型ハーネスの効果

プロジェクト数	バージョン	ハーネスの状態	期待される効果
1プロジェクト目	v1.0	初期版	基本的な安全性の確保
3プロジェクト目	v1.3	改善版	境界ケース・セキュリティが充実
5プロジェクト目	v1.5	成熟版	類似プロジェクトの失敗を事前回避
10プロジェクト目	v2.0	高度版	業種・規模別テンプレートが揃う

ハーネス管理プロンプト（Claude Code に渡す）

H-INIT：成長型ハーネスの初期化

実行タイミング：最初の1回だけ実行する

成長型ハーネスを初期化してください。

以下のディレクトリ構造を作成してください：

```
~/claude-harness/
├── README.md（使い方ガイド）
├── VERSION（初期値：v1.0.0）
├── templates/
│   ├── CLAUDE.md.template
│   ├── CONSTRAINTS.md.template
│   ├── requirements_template.md
│   ├── sdd_template.md
│   └── test_plan_template.md
├── knowledge/
│   ├── lessons_learned.md
│   ├── boundary_cases.md
│   ├── security_patterns.md
│   ├── error_patterns.md
│   └── prompt_library.md
├── projects/
└── changelog.md
```

各ファイルの初期内容を作成してください。
作成後、ディレクトリ構造を確認して報告してください。

H-UPDATE：プロジェクト完了後のハーネス更新

実行タイミング：プロジェクトが完了するたびに実行する

プロジェクトが完了しました。成長型ハーネスを更新してください。

【今回のプロジェクト】
- プロジェクト名：[名前]
- 期間：[開始〜終了]
- 概要：[何を作ったか]

【更新手順】
1. ~/claude-harness/knowledge/lessons_learned.md に学びを追記する
2. 発見した境界ケースを boundary_cases.md に追記する
3. 発見したセキュリティパターンを security_patterns.md に追記する
4. 効果的だったプロンプトを prompt_library.md に追記する
5. テンプレートの改善点があれば修正してバージョンを上げる
6. changelog.md に変更内容を記録する
7. ~/claude-harness/projects/[プロジェクト名]/ にプロジェクト記録を保存する

更新完了後、何が追加・改善されたかを報告してください。

H-NEW：新プロジェクト開始（ハーネスから）

実行タイミング：新しいプロジェクトを始めるとき

新プロジェクトを開始します。成長型ハーネスから設定を引き継いでください。

【手順】
1. ~/claude-harness/VERSION を確認する
2. ~/claude-harness/knowledge/lessons_learned.md を読む
   → 類似プロジェクトの失敗パターンを確認する
3. ~/claude-harness/knowledge/boundary_cases.md を読む
   → E2Eシナリオに境界ケースを追加する
4. テンプレートをコピーする：
   cp ~/claude-harness/templates/CLAUDE.md.template ./CLAUDE.md
   cp ~/claude-harness/templates/CONSTRAINTS.md.template ./CONSTRAINTS.md
5. CLAUDE.md のプレースホルダーをプロジェクト固有の情報で埋める

完了したら、過去のプロジェクトから引き継いだ知識の概要を報告してください。

H-REVIEW：ハーネスの定期レビュー

実行タイミング：月1回または5プロジェクトごとに実行する

> 成長型ハーネスの定期レビューを行います。
>
> 以下を確認してください：
>
> 1. テンプレートの品質確認
>   - 使いにくい箇所がないか
>   - 不足している項目がないか
>   - 古くなった情報がないか
>
> 2. 知識蓄積の確認
>   - `lessons_learned.md` の内容が活用されているか
>   - *同じ失敗が繰り返されていないか*
>
> 3. *プロンプト集の確認*
>   - *効果的でないプロンプトがないか*
>   - *新しいシーンに対応するプロンプトが必要か*
>
> 4. *改善提案*
>   - *次のバージョンで改善すべき点を3つ提案する*
>
> *レビュー結果を報告してください。*

CLAUDE.md テンプレート（v1.0）

新プロジェクト開始時に `CLAUDE.md.template` からコピーして使う。

```
# CLAUDE.md — [プロジェクト名]

## プロジェクト概要
[1〜3文で何を作るかを説明]

## 技術スタック
- 言語：[Python / Node.js / 等]
- 主要ライブラリ：[使用するライブラリ]
- 外部サービス：[連携するAPI・サービス]

## 重要なコマンド
```bash
# セットアップ
[セットアップコマンド]

# テスト実行
[テストコマンド]

# 本番実行
[本番実行コマンド]
```

完了基準

以下が全て満たされたときに「完了」とする：

- [ ] [完了条件1]
- [ ] [完了条件2]
- [ ] 全テストが通過している
- [ ] セキュリティチェックが完了している

作業ルール

1. 実装前に必ず計画を提示して私の承認を得ること
2. 修正は私の承認後に行うこと
3. 本番データには直接アクセスしないこと
4. 1つのコミットに複数の変更を混在させないこと
```

禁止事項

→ CONSTRAINTS.md を参照

引き継ぎ情報

→ progress.md を参照

```
---

## CONSTRAINTS.md テンプレート（v1.0）

```markdown
# CONSTRAINTS.md — [プロジェクト名]

## 絶対禁止（いかなる場合も実行しない）
- 本番データベースへの直接書き込み・削除
- APIキー・パスワードをコードにハードコード
- 私の承認なしに本番環境への変更を実施
- .envファイルをgitにコミット

## 要注意（実行前に必ず確認する）
- 外部APIへのリクエスト（コスト・レート制限に注意）
- ファイルの削除・上書き
- データベースのスキーマ変更

## プロジェクト固有の禁止事項
- [プロジェクト固有の禁止事項を記載]
```

```

progress.md テンプレート（v1.0）

```
# progress.md — [プロジェクト名]

## プロジェクト状況
- 開始日：[日付]
- 現在のフェーズ：[フェーズ名]
- 全体完了率：[0〜100%]

## 完了したタスク
- [完了したタスク]（[完了日]）

## 現在進行中のタスク
- [現在のタスク]

## 次のタスク（優先順）
1. [最優先タスク]
2. [次のタスク]

## ブロッカー・未解決の問題
- [問題があれば記載]

## 最新のセッションメモ
[最後のセッションの引き継ぎ情報]
```

作成：非エンジニア向け Claude Code 要件定義自動化体系