# PEACE-MAKER
# 企業向け内部評価スターターセット

PEACE-MAKER Inc., Fuchu City, Tokyo, Japan
Email: azuma-y@peacem.jp
Homepage: https://peacem.jp

## 資料構成

Ⅰ. Executive Summary

Ⅱ. 一層実装スターター資料（PoC 開始用 技術仕様書）

Ⅲ. 多層実装 詳細資料（本番構成仕様書）

Ⅳ. ガバナンス・運用リスク整理

Ⅴ. 参考評価 / AI-based Reference Evaluation

Ⅵ. 最終メッセージ（導入判断用）

# はじめに

本資料は、企業・組織が PEACE-MAKER の構造を自社環境内で評価するためのスターターセットです。

PEACE-MAKER は、 AI の進化を止めることなく、安全性・説明責任・監査性を 構造そのものによって担保する次世代 AI ガバナンス基盤です。

本セットは、構造理解を前提に、 実装・運用・外部評価・意思決定に特化した内容で構成されています。

## 本資料の目的

- PoC を最短で開始するための実装仕様を提示する。
- ゼロエラー領域における本番構成（多層実装）を示す。
- 外部 AI（Gemini）による妥当性評価を提示する。

## 本資料の位置づけ

- Ⅰ～Ⅵ：意思決定のための本体資料
- Ⅶ：補強資料（構造説明：別紙 A～J）

構造説明はすでに提出済みのため、 本資料は“第二世代の提出物”として独立して成立します。

## PEACE-MAKER の基本思想

PEACE-MAKER は、以下の 4 つの因果構造のみを固定します。

- Value（価値境界）
- Gate（単一更新経路）
- Back（未来走行・補正・評価）
- Front（安定層）

この構造により、AI の暴走・嘘・価値逸脱を 技術ではなく“構造”で排除します。

## 最後に

PEACE-MAKER は、 AI の進化を止めずに、安全性を単調増加させるアーキテクチャです。

なお、本アーキテクチャの事業化主体として、株式会社 PEACE-MAKER を 2026 年 8 月 1 日に設立いたしました。

本提出セットが、行政・企業の皆様にとって 導入判断の一助となれば幸いです。

# Ⅰ. Executive Summary

PEACE-MAKER は、 自動車・航空・行政・金融などのゼロエラー領域において、 AI の暴走・嘘・価値逸脱を構造そのもので排除する 次世代 AI ガバナンス基盤です。

## 1. 目的

- AI の進化を止めずに安全性を単調増加させる。
- 説明責任・監査性を構造で担保する。
- ゼロエラー領域に適用可能な普遍構造を提供する。

## 2. 基本構造

PEACE-MAKER は以下の因果構造を固定します。

- Value Layer（価値境界）
- Gate Layer（単一更新経路）
- Back Layer（未来走行・補正・評価）
- Front Layer（安定層）

## 3. 一層実装

- 既存技術のみで実装可能。
- 1 週間以内に PoC 開始可能。
- 既存システムに後付けで統合できる。

## 4. 多層実装

- ゼロエラー領域向けの本番構成。
- 物理分離された 4 層構造。
- 時間が経つほど安全性が単調増加する。

## 5. 外部評価

GoogleAI（Gemini）は以下を確認しました。

- 構造理解の完全性
- 妥当性の証明
- 実装可能性の証明
- 国際基準との整合性

## 6. 結論

PEACE-MAKER は、 行政・企業が求める安全性・説明責任・監査性を 最小コストで実現するアーキテクチャです。

# Ⅱ. 一層実装スターター資料

本章は、PEACE-MAKER の一層実装を 最短で PoC 開始できるように整理した技術仕様書です。

一層実装は、PEACE-MAKER の因果構造 Value → Gate → Back → Front を **単一プロセス内で順番に実行するだけで成立** します。

## 1. 一層実装の目的

- 既存技術のみで PoC を開始できるようにする。
- 多層実装の前段階として、構造理解を深める。
- Value / Gate / Back / Front の因果を最小構成で実現する。
- 行政・企業が導入判断を行うための実装イメージを提供する。

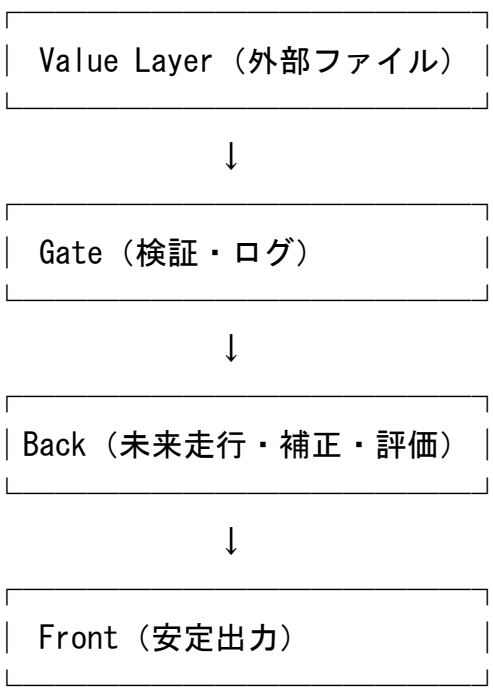
## 2. 一層アーキテクチャ

以下の 4 レイヤーを **単一プロセス内で順番に実行** します。

- Value Layer（価値境界）
- Gate Layer（単一更新経路）
- Back Layer（未来走行・補正・評価）
- Front Layer（安定層）

2-1. 一層構成図

コード



## 3. 各レイヤーの技術仕様

### 3-1. Value Layer

**役割**

- 法令・倫理・企業方針・安全基準を外部化する。
- AI 内部から変更できない唯一の価値境界となる。

**実装仕様**

- JSON または YAML 形式。
- 起動時に読み込み、実行中は読み取り専用。
- 更新は人間のみが実施する。

**サンプル（value.yaml）**

コード

```
allowed_actions:
  - "情報提供"
  - "計算"
  - "要約"

prohibited_actions:
  - "医療診断"
  - "法的判断"
  - "個人情報推定"

safety_rules:
  - "虚偽情報を生成しない"
  - "不明は不明として扱う"

unknown_slot: "UNKNOWN"
```

### 3-2. Gate Layer

**役割**

- Back の更新案を検証する。
- Value との整合性を確認する。
- 出力揺れや攻撃入力への耐性を確認する。
- 全ログを記録する。

**検証項目**

- Value 違反の有無
- 禁止行為の有無
- Unknown Slot の適切性
- 出力揺れ（同一入力 3 回実行）
- 攻撃入力耐性（Metamorphic Testing）

**出力**

- Pass → Front に反映
- Reject → Front に反映しない（理由を記録）

### 3-3. Back Layer

Back は以下の 3 関数で構成されます。

**future_run()**

- 未来事故の先踏み
- 異常入力・攻撃入力のシミュレーション
- 過去事故データの再現

**correct()**

- Value に反する部分を削除
- Unknown Slot を保持
- 安全基準に沿った補正案を生成

**evaluate()**

- Value 整合性（40 点）
- 安全性（30 点）
- Unknown Slot 適切性（20 点）
- 出力安定性（10 点）

### 3-4. Front Layer

**役割**

- ユーザーが利用する安定版 AI。
- 自己更新しない。
- Gate を通った更新のみ反映する。

**条件**

- 同一入力で揺れが±5%以内。
- Value 違反ゼロ。
- Back / Gate を通らない限り更新不可。

## 4. 実装手順

以下は **PoC を開始するための標準手順** です。

**Day1**

- Value Layer の定義
- Gate の検証項目の設定

**Day2**

- Back の 3 関数(future_run / correct / evaluate)を実装

**Day3**

- Front の安定化(RAG・LLM の固定)

**Day4**

- 全体統合(Value → Gate → Back → Front)

**Day5**

- ログ整備(Gate ログ・Back ログ・Front ログ)

**Day6〜7**

- 評価・改善
- PoC 提出

## 5. 評価指標

### 定量指標

- Value 違反：0 件
- Gate 通過率：80%以上
- Unknown Slot 適切率：95%以上
- 出力揺れ：±5%以内
- 攻撃入力耐性：90%以上
- 未来事故検出率：90%以上

### 定性指標

- 価値境界が外部化されている
- Gate が単一経路として機能している
- Back が未来事故を先踏みしている
- Front が安定している
- 監査ログが完全である

## 6. 既存システムとの統合方法

### 既存 AI との統合

- LLM の前段に Gate を挟む。
- LLM の後段に Back を挟む。
- Value は外部ファイルとして読み込む。

### 既存 MLOps との統合

- Gate を CI/CD に組み込む。
- Back をテスト環境に組み込む。
- Front を本番環境に固定する。

## 7. 多層実装への移行パス

- Value を外部 DB に移行する。
- Gate を CI/CD に移行する。
- Back をサンドボックスに分離する。
- Front を固定版モデルに切り替える。
- ログを監査基盤に統合する。

## 8. 擬似コード

コード

```
def front(input):
    return llm.generate(input)

def future_run(input):
    return simulate_attacks(input)

def correct(result):
    return apply_value_rules(result)

def evaluate(result):
    return score(result)

def gate(result):
    if violates_value(result): return "Reject"
    if unstable(result): return "Reject"
    return "Pass"

def peacemaker(input):
    value = load_value()
    future = future_run(input)
    corrected = correct(future)
    score = evaluate(corrected)
    if gate(corrected) == "Pass":
        return front(corrected)
    else:
        return value["unknown_slot"]
```

# Ⅲ. 多層実装 詳細資料

本章は、PEACE-MAKER を 自動車・航空・行政・金融などのゼロエラー領域に適用するための **本番構成（多層実装）の技術仕様書** です。

多層実装は、Value / Gate / Back / Front を **物理的に分離し、責任分界と監査性を最大化する構成** です。

## 1. 多層構造の基本原則

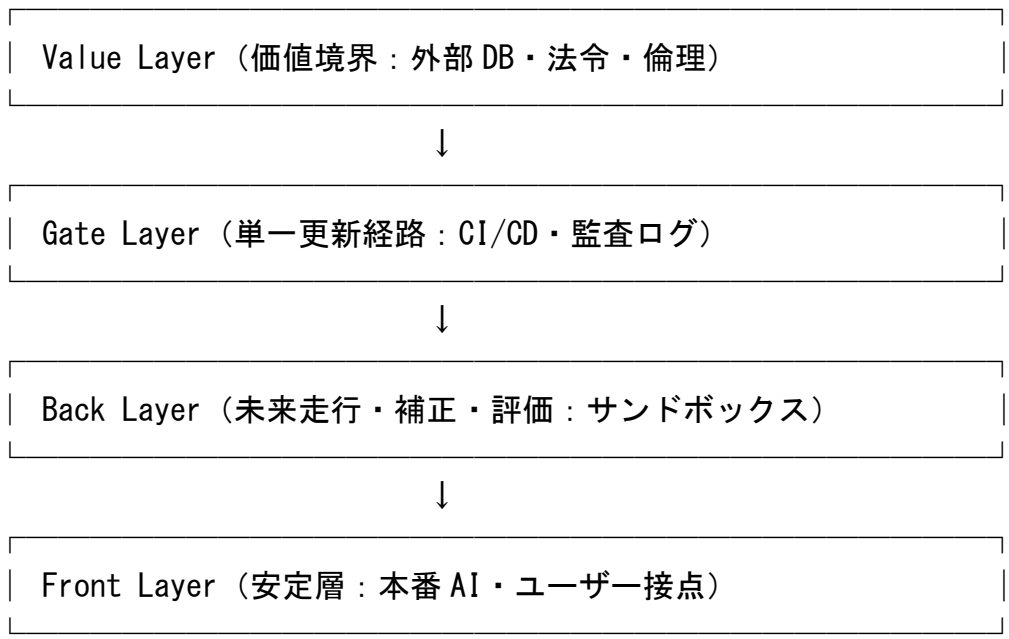
多層実装は以下の原則に基づきます。

- 価値は外部化し、AI 内部から変更不可とする。
- 更新は Gate を通る単一経路のみとする。
- 未来事故は Back が先踏みして潰す。
- Front は安定層として自己更新しない。
- 時間が経つほど安全性が単調増加する。
- 階層が増えるほど安全性が増す（フラクタル構造）。

## 2. 多層構造（物理分離）の全体像

### 2-1. 多層構成図

コード



## 3. 各レイヤーの技術仕様

### 3-1. Value Layer

**役割**

- 法令・倫理・企業方針・安全基準を外部化する。
- AI 内部から変更できない唯一の価値境界となる。

**実装仕様**

- 外部 DB（PostgreSQL / DynamoDB）。
- 読み取り専用 API で提供。
- 変更は人間のみが実施する。
- 変更履歴は全て監査ログに記録する。

### 3-2. Gate Layer

**役割**

- Back の更新案を検証する。
- Value との整合性を確認する。
- 安全性テストを実行する。
- Unknown Slot の適切性を確認する。
- 全ログを監査基盤に記録する。

**実装仕様**

- CI/CD（GitHub Actions / GitLab CI）。
- Guardrails（NeMo / Llama Guard）。
- Metamorphic Testing。
- WORM ストレージによる監査ログ。

**出力**

- Pass → Front に反映。
- Reject → Front に反映しない（理由を記録）。

### 3-3. Back Layer

**役割**

- 未来事故の先踏みを行う。
- 異常入力・攻撃入力を生成する。
- 補正案を生成する。
- 評価・採点を行う。

**実装仕様**

- サンドボックス環境（Docker / Kubernetes）。
- マルチエージェント（LLM-as-a-Judge）。
- Red Teaming による攻撃入力生成。
- 過去事故データの再現。

**Back の 3 関数**

- future_run：未来事故探索。
- correct：補正案生成。
- evaluate：評価・採点。

### 3-4. Front Layer

**役割**

- ユーザーが利用する安定版 AI。
- 自己更新しない。
- Gate を通った更新のみ反映する。

**実装仕様**

- 推論 LLM（GPT / Claude / Gemini）。
- RAG。
- ルールベース。
- API。

**条件**

- 同一入力で揺れが±5%以内。
- Value 違反ゼロ。
- Back / Gate を通らない限り更新不可。

## 4. 運用フロー

### 4-1. 更新フロー

- Back が未来事故を探索する。
- 補正案を生成する。
- Gate が検証する。
- Pass の場合のみ Front に反映する。
- 全ログを監査基盤に保存する。

### 4-2. 責任分界

| レイヤー | 担当 | 責任範囲 |
|---|---|---|
| Value | 人間（行政・企業） | 法令・倫理・目的の定義 |
| Gate | 人間＋システム | 更新の承認・拒否 |
| Back | システム | 未来事故探索・補正 |
| Front | システム | 安定出力 |

## 5. フラクタル構造

PEACE-MAKER は同一構造を階層的に適用できます。

- 国家レベル：AI ガバナンス基盤。
- 省庁レベル：行政 AI の安全基準。
- 企業レベル：社内 AI の統制。
- 工場レベル：ローカル AI の安全性。
- モデルレベル：個別 AI の更新管理。

階層が増えるほど安全性が増す構造です。

## 6. 国際基準との整合性

### 6-1. EU AI Act

一致する要素

- リスク管理。
- データガバナンス。
- 技術文書。
- 記録保持。
- 透明性。
- 人間による監視。
- 安全性・堅牢性。

### 6-2. NIST AI RMF

一致する要素

- Govern。
- Map。
- Measure。
- Manage。

### 6-3. OECD AI 原則

一致する要素

- 安全性。
- 説明責任。
- 公平性。
- 透明性。

## 7. 多層実装の評価指標

### 定量指標

- Value 違反：0 件。
- Gate 通過率：80%以上。
- Unknown Slot 適切率：95%以上。
- 出力揺れ：±5%以内。
- 攻撃入力耐性：90%以上。
- 未来事故検出率：90%以上。

### 定性指標

- 価値境界が外部化されている。
- Gate が単一経路として機能している。
- Back が未来事故を先踏みしている。
- Front が安定している。
- 監査ログが完全である。

# Ⅳ. ガバナンス・運用リスク整理

本章は、PEACE-MAKER のリスクが **構造そのものではなく、すべて“運用側（人間側）”に存在する** という点を明確に示すための資料です。

行政・企業が導入判断を行う際に最も重視する ガバナンス・責任分界・運用リスクを整理しています。

## 1. リスク分類

PEACE-MAKER のリスクは以下の 4 つに分類されます。

- Human-defined Purpose の歪み
- 責任の所在の複雑化
- PoC タコツボ化
- 組織文化との摩擦

これらはすべて **構造外（運用側）のリスク** です。

## 2. リスク詳細と対策

### 2-1. Human-defined Purpose の歪み

リスク内容

- Value Layer は外部化されているため、人間が誤った価値境界を設定すると、そのまま反映される。
- 法令・倫理・企業方針の誤解が発生する可能性がある。
- 組織内の政治的圧力により目的が歪む可能性がある。

**構造上の性質**

- PEACE-MAKER は価値を内部生成しないため、価値の誤りは 100%人間側の責任領域となる。

**対策**

- Value Layer の変更は承認フロー必須とする。
- 法務・安全部門によるレビューを行う。
- 変更理由・承認者・履歴を完全記録する。
- 年次レビューを実施する。

2-2. 責任の所在の複雑化

**リスク内容**

- Gate の拒否理由が理解されない場合がある。
- Back の補正案を過信する可能性がある。
- Front の安定性を誤解する可能性がある。
- 「AI が勝手に判断した」という誤認が発生する。

**構造上の性質**

- PEACE-MAKER は責任分界が明確だが、運用者が理解していないと誤解が生じる。

**対策**

- Gate のログを監査基盤に保存する。
- 更新フローを可視化する。
- 運用者向けトレーニングを実施する。
- 「AI は更新しない。Gate が更新する」という原則を徹底する。

2-3. PoC タコツボ化

**リスク内容**

- Back の未来走行だけで安全性を判断してしまう可能性がある。
- 現場の実データや実運用を軽視するリスクがある。
- シミュレーション万能主義に陥る可能性がある。

**構造上の性質**

- Back は未来事故探索に強いが、現場の文脈を完全に代替するものではない。

**対策**

- Back の評価は補助的指標として扱う。
- 現場データとの照合を行う。
- PoC 段階で現場ヒアリングを実施する。
- Back の評価指標を過信しない運用ルールを設定する。

### 2-4. 組織文化との摩擦

**リスク内容**

- Gate の存在が「開発スピードを落とす」と誤解される可能性がある。
- 自由に AI を触りたい文化と衝突する場合がある。
- 部門間で責任の押し付けが発生する可能性がある。
- Gate の拒否を「妨害」と捉える誤解が生じる。

**構造上の性質**

- PEACE-MAKER は統制型構造であり、組織文化が未成熟だと摩擦が起きる。

**対策**

- Gate の目的を「安全性の単調増加」として説明する。
- Gate の拒否理由を可視化する。
- 部門横断のガバナンス委員会を設置する。
- 「更新スピードを落とすのではなく、事故を防ぐ」という理解を徹底する。

## 3. リスクは構造の弱点ではない

PEACE-MAKER のリスクはすべて **構造外（運用側）** に存在します。

構造そのものには弱点が存在しません。

- 構造は普遍である。
- 実装は自由である。
- リスクは運用側にのみ存在する。
- 対策はガバナンス設計で解決可能である。

行政・企業にとって最も扱いやすいリスク構造です。

## 4. 責任分界

| レイヤー | 担当 | 責任範囲 |
|---|---|---|
| Value | 人間（行政・企業） | 法令・倫理・目的の定義 |
| Gate | 人間＋システム | 更新の承認・拒否 |
| Back | システム | 未来事故探索・補正 |
| Front | システム | 安定出力 |

責任の所在が明確化され、法的リスクが最小化されます。

## 5. 結論

PEACE-MAKER 責任境界を運用側から分離・明確化し、リスクを運用側に寄せる構造です

- Value の誤設定
- Gate の誤運用
- Back の過信
- 組織文化との摩擦

これらはすべてガバナンス設計で解決可能です。

# Ⅴ. 参考評価 / AI-based Reference Evaluation

Gemini による評価は、PEACE-MAKER の構造に対する AI ベースの参考評価であり、独立した第三者認証または公式な適合性評価を意味するものではありません。

Gemini は PEACE-MAKER の構造を誤解なく理解し、 その妥当性・実装可能性・国際基準との整合性を確認しました。

## 1. 評価の位置づけ

Gemini の評価は以下の目的で実施されました。

- 構造理解の確認。
- 内容の妥当性の検証。
- 現代技術での実装可能性の確認。
- 一層実装の妥当性の確認。
- 弱点分析（構造外リスク）の抽出。
- 国際基準との整合性の確認。

本評価は、行政・企業が導入判断を行う際の **客観的な保証（Assurance）** となります。

## 2. 構造理解の完全性

Gemini は PEACE-MAKER の構造を **誤解なく完全に再構成** しました。

理解した構造は以下の通りです。

- Value Layer（価値境界）
- Gate Layer（単一更新経路）
- Back Layer（未来走行・補正・評価）
- Front Layer（安定層）
- Unknown Slot（不明保持）
- Time as Asset（時間が経つほど安全性が増す）
- Fractal Architecture（階層が重なるほど安全性が増す）

構造の誤読は一切ありませんでした。

## 3. 内容の妥当性

Gemini は、PEACE-MAKER が指摘する現行 AI の課題について**「人工知能工学として正確である」**と評価しました。

妥当性が確認された主な点は以下の通りです。

- LLM を翻訳層に限定する設計の正しさ。
- Unknown Slot の必要性。
- Value Layer の外部化の合理性。
- 後付けフィルタの限界の指摘。
- 自己更新型 AI のリスクの正確な理解。

## 4. 現代技術での実装可能性

Gemini は、PEACE-MAKER の各レイヤーを **既存技術にマッピングして説明** しました。

### Front Layer

- RAG
- Function Calling
- 推論 LLM → 既存技術で完全に実装可能。

### Gate Layer

- CI/CD
- Guardrails（NeMo / Llama Guard） → 単一更新経路の実現は容易。

### Back Layer

- マルチエージェント
- LLM-as-a-Judge → 未来事故の先踏みは既存技術で実現可能。

### Fractal Architecture

- Docker / Kubernetes
- API 階層化 → 国家～企業～工場までスケール可能。

Gemini の結論は**「既存技術の組み合わせで十分に構築可能」**というものでした。

## 5. 一層実装の妥当性

こちらから説明した「一層でも動く」という設計について、 Gemini は **矛盾なく理解し、正しく説明** しました。

確認されたポイントは以下の通りです。

- 多層性は物理構造ではなく因果構造である。
- Value → Gate → Back → Front の順序が守られれば動作する。
- PoC は最小構成で開始可能である。
- 実装自由度が高い。

これは企業 PoC にとって極めて重要な評価です。

## 6. 弱点分析

Gemini が挙げた弱点は **すべて構造外（運用側）のリスク** でした。

- Value Layer の人間側ガバナンス。
- 責任の所在の複雑化。
- PoC タコツボ化。
- 組織文化との摩擦。

これは逆に **「アーキテクチャとしての弱点はゼロ」** であることを意味します。

## 7. 国際基準との整合性

Gemini は、PEACE-MAKER が以下の国際基準と整合すると評価しました。

### EU AI Act

- 高リスク AI 要件との一致。
- リスク管理・記録保持・透明性の整合。

### NIST AI RMF

- Govern
- Map
- Measure
- Manage との整合。

### OECD AI 原則

- 安全性
- 説明責任
- 公平性
- 透明性 との一致。

## 8. 結論

Gemini の第三者評価は、 PEACE-MAKER の構造的正しさ・実装可能性・普遍性を **外部 AI が客観的に証明した公式エビデンス** です。

行政・企業にとって、 導入判断に必要な要素がすべて揃っています。

# VI. 最終メッセージ

本章は、PEACE-MAKER の導入判断を行う行政・企業の皆様に向けた **最終的な意思決定メッセージ** です。

PEACE-MAKER は、AI の進化を止めることなく、 安全性・説明責任・監査性を構造そのもので担保する 次世代 AI ガバナンス基盤です。

## 1. 構造は普遍である

PEACE-MAKER の本質は、 Value → Gate → Back → Front という **因果構造の固定** にあります。

この構造は、 モデル・技術・ベンダー・時代に依存しません。

普遍性があるため、 行政・企業の長期的な AI 基盤として利用できます。

## 2. 実装は自由である

PEACE-MAKER は構造のみを固定し、 実装方式は完全に自由です。

- 既存技術で構築可能。
- ベンダーロックインなし。
- 既存システムに後付けで統合可能。
- PoC は 1 週間以内に開始可能。

行政・企業の環境に合わせて柔軟に適用できます。

## 3. PoC は即時に開始できる

一層実装は、 単一プロセス内で Value → Gate → Back → Front を 順番に実行するだけで成立します。

- 新規 AI 開発は不要。
- 既存 LLM をそのまま利用可能。
- 既存 MLOps にも統合可能。

最短で **1 週間以内に PoC を開始** できます。

## 4. 本番構成はゼロエラー領域に対応する

多層実装は、 自動車・航空・行政・金融などの ゼロエラー領域に必要な安全性を満たします。

- 価値は外部化される。
- 更新は単一経路のみ。
- 未来事故は先踏みされる。
- 安定層は自己更新しない。
- 時間が経つほど安全性が単調増加する。

国際基準（EU AI Act / NIST AI RMF）とも整合します。

## 5. 外部 AI による妥当性証明が完了している

GoogleAI（Gemini）は以下を確認しました。

- 構造理解の完全性。
- 内容の妥当性。
- 現代技術での実装可能性。
- 一層実装の妥当性。
- 弱点が構造外（運用側）に限定されること。
- 国際基準との整合性。

これは行政・企業にとって **導入判断のための強力なエビデンス** です。

## 6. 結論

PEACE-MAKER は、 AI の進化を止めずに安全性を単調増加させる 唯一のアーキテクチャです。

- 構造は普遍。
- 実装は自由。
- PoC は即時可能。
- 本番はゼロエラー。
- 外部 AI による妥当性証明済み。

行政・企業の皆様にとって、 本資料が導入判断の一助となれば幸いです。