T-RDEと通常テストの違い

MDES駆動開発における意味監査と品質ゲート

論文とZYX MDES駆動標準コーディング手順マニフェストを接続した実務レポート

中心命題	Testは仕様を実行可能にし、CIは再現性を確かめる。T-RDEは意味差分を監査し、MDESはその結果を次のManifestへつなぐ。

作成日：2026年7月28日

状態：MDES再考版

要旨

AI支援開発では、生成されたコードがコンパイルでき、テストを通過し、要求された機能を備えているように見えても、人間が当初意図した意味が保存されているとは限らない。曖昧な要求が一つの解釈へ固定されたり、依頼していない機能が暗黙に追加されたり、暫定的な実装が確定仕様のように扱われたりするためである。これらは、必ずしも機能不良として現れない。

T-RDE（Test-with-Resonant Deviation Evaluator）は、この空白を扱うための意味監査フレームワークである。ただし、通常テストを置き換えるものではない。通常テスト、型検査、静的解析、セキュリティレビューなどが機能的・技術的な妥当性を確認する一方で、T-RDEは、人間の設計意図と生成成果物の間で意味がどのように保存、変換、逸脱、欠落したかを可視化し、人間の判断へ戻す。

ZYXのMDES駆動開発では、T-RDEをSustain段階だけの後付けレビューとして扱わない。Manifestで問題、非目標、受け入れ条件、撤退条件を定め、Designで仕様、ADR、テスト設計、T-RDE観点を準備し、EvolveでPrototype、Test First、TDD、local CIによって証拠を作り、SustainでPull Request、T-RDE、Release、既知制約、Follow-up Issueを統合する。品質ゲートは、この四相で作られた証拠と判断条件が揃っているかを確認する統合点となる。

重要な境界	T-RDEが担保するのは「意味の客観的な正しさ」ではない。意味の変化、不確実性、責任移動、判断根拠が可視化され、人間が説明可能な形で受容、修復、棄却、上申できる状態である。

1. なぜ通常テストだけでは足りないのか

通常テストは、定義された入力、状態、条件に対して、期待する出力や振る舞いが得られるかを確認する。単体テスト、統合テスト、回帰テスト、権限テスト、セキュリティテストなどは、mainを壊さず、変更を再現可能に検証するために不可欠である。

しかし、テストが通ることは、そのテストが表現している仕様に適合したことを示すにすぎない。テストの前提となった仕様自体が、人間の意図から変わっていれば、その変化はテストによって固定される可能性さえある。AI支援開発では、モデルが不足した文脈を補い、もっともらしい設計判断を追加できるため、この問題が顕在化しやすい。

1.1 機能不良として現れない四つの変化

- 曖昧さから単一実装への変化：複数の解釈があり得る要求をAIが一つに決め、未確定だったこと自体を見えなくする。
- 未依頼から暗黙の追加への変化：削除、永続化、認証、外部APIなどを追加し、プライバシーや責任の判断まで持ち込む。
- 意図から別の使い勝手への変化：「完了項目を下へ」という要求が自動並べ替えに変わり、手動順序という利用者の期待を失う。
- 要求から未実装への変化：動く部分だけでテストが通っても、警告表示など本来の意図要素が欠落したまま残る。

ここで重要なのは、変換や追加が常に誤りとは限らないという点である。自然言語からコードへの変換には解釈が伴い、有用な補完も起こり得る。問題は、変化が不可視のまま確定され、誰が何を理由に承認したかを追えなくなることである。

2. 通常テストとT-RDEの役割分担

評価層	主な問い	強みと限界
通常テスト	指定されたケースで期待どおり動くか	機能回帰を再現可能に検出する。設計意図そのものの変換は直接扱わない。
型検査・静的解析	構造的に妥当か、既知の危険パターンがないか	技術的な欠陥を検出する。意味や責任の移動は判断しない。
CI	同じ検証を再現可能に実行できるか	mainを壊しにくくする。検証項目に含まれない意味差分は保証しない。
T-RDE	設計意図から意味がどう変わったか	意味差分と修復可能性を追跡する。人間レビューと校正を必要とする。

関係	通常テストとT-RDEは競合しない。Testが動作を、CIが再現性を、T-RDEが意味差分を扱い、MDESがそれらを開発サイクルとして接続する。

3. MDES四相における品質形成

MDESは、Manifest、Design、Evolve、Sustainを連続的に回しながら、問題定義、設計、実装、検証、統合、運用、再定義を進める。品質は最後の検査で付与されるものではなく、四相を通じて形成される。

3.1 Manifest：問題と境界を定義する

何を作るかより前に、なぜ作るのか、何を解き、何を解かないのかを明確にする。価値仮説、ステークホルダ、影響範囲、KPI、非目標、制約、リスク、受け入れ条件、撤退条件が、後のT-RDEで比較する基準になる。

主な成果物：Problem Statement、非目標、受け入れ条件、撤退条件、Roadmap、Milestone。

3.2 Design：判断基準を設計する

Manifestを詳細仕様、ADR、API仕様、データ構造、状態遷移、権限モデル、エラー処理、テスト設計へ展開する。同時に、何を保存し、どの変換を許容し、どの変更を高リスクとみなすかというT-RDE観点を準備する。

主な成果物：Design Spec、ADR、Test Plan、Prototype Plan、T-RDE観点。

3.3 Evolve：実行可能な証拠を作る

未知性が高い部分はPrototype Firstで探索し、本実装とは分離する。本実装はTest First、TDD、リファクタリング、local CIを通じて進める。ここで生成されるテスト結果、CI結果、変更差分が、Sustainでの統合判断に使われる。

主な成果物：Issue Branch、Prototype結果、Test、実装、CI結果、Pull Request。

3.4 Sustain：統合し、次のManifestへ返す

Pull Request Review、T-RDE、Release Note、KPI確認、運用文書、既知制約、Follow-up Issueを統合する。Sustainは終点ではなく、運用で得た事実と未解決事項から次のManifestを生成する接続点である。

主な成果物：T-RDE記録、レビュー判断、Release Note、既知制約、次のIssueまたはManifest。

4. T-RDEの二層構造

論文上のT-RDEと、MDESマニフェストで使用する六項目の差異レポートは、同じ階層の定義ではない。前者は意味監査の分析モデルであり、後者はPull RequestやReleaseで差異を説明し、次のサイクルへ渡すための実務フォーマットである。

4.1 論文上の監査モデル

意味マップでは、設計意図の要素を「保存」「変換」「逸脱」「未実装」に分類し、依頼されていない機能、前提、データ構造、責任配置を「暗黙の追加」として別に記録する。変換は自動的な失敗ではなく、明示され、受容可能で、修復可能かどうかが問われる。

- 保存：関連する意味変更なしに意図が維持された。
- 変換：解釈または実装選択を通じて別の形になった。
- 逸脱：成果物が意図から離れ、修正または棄却の検討を要する。
- 未実装：意図要素が成果物に存在しない。
- 暗黙の追加：明示的に依頼されていない機能、前提、責任が導入された。

4.2 MDES実務の差異レポート

MDESでは、上記の分析結果を、開発と運用の判断へ接続するために次の六項目で記録する。この形式は、研究上の分類を置き換えるものではなく、差異を説明し、未解決事項と次回更新を管理するための実務的な要約である。

1. 保存された要素
2. 変換された要素
3. 補完された要素
4. 未解決の要素
5. 逸脱リスク
6. 次回更新方針

5. ΔMと不確実性の扱い

意味の変化を単一の欠陥スコアへ縮減すると、小さいが重要な責任変更や、曖昧さの消失を見逃しやすい。論文では、意味変化を五つのΔM成分として多面的に扱う。

成分	対象	例
ΔS	意味内容	概念や表現が別の意味へ変わる。
ΔP	実践的アフォーダンス	利用者ができること、できなくなることが変わる。
ΔR	関係構成	順序、強調、グループ、操作関係が変わる。
ΔI	制度的配置	責任、権限、永続化、データフローが変わる。
ΔU	不確実性の扱い	曖昧さ、条件、対立する解釈が消される。

ΔUの特別な役割	不確実性が隠されると、ΔS、ΔP、ΔR、ΔIの評価自体が歪む。したがってΔUの健全性は、品質ゲートを通過する前提条件として扱う。

実装上の数値、重み、閾値、方向評価は、意味の客観測定ではなく暫定的なレビュー支援である。高い総合点によって、責任の不明確化や利用者の自律性低下といった棄却条件を相殺してはならない。

6. 品質ゲートはどう担保されるか

品質ゲートは、AIや単一スコアが成果物の正しさを認定する装置ではない。MDES各相で作られた基準、証拠、記録をPull Requestで照合し、人間が統合可否を判断する条件束である。

6.1 通過条件を分離する

- 意味整合、不確実性の校正、価値調整、修復可能性を共同で確認する。
- ΔUを前提条件とし、曖昧さが隠れた状態では通過させない。
- 棄却条件を総合点で相殺せず、小さな変更でも責任や自律性を損なう場合は修正または明示的承認を求める。

6.2 独立した証拠を重ねる

T-RDE単体で機能、セキュリティ、アクセシビリティ、法令適合性を保証することはできない。必要な単体・統合・回帰・権限・セキュリティ・移行テスト、型検査、静的解析、Build、local actによるCI、専門家レビューを併用する。

6.3 来歴と判断理由を保存する

Issue、Branch、Pull Requestを変更の来歴として接続し、意味マップ、テスト結果、CI結果、既知制約、未解決事項、レビュー判断を保存する。後から判断を再検討し、修復し、必要なら差し戻せることが品質の一部となる。

6.4 最終決定を人間へ戻す

T-RDEの出力は、受容、Follow-up付き受容、変更要求、棄却、上申のための監査情報である。AI、監査モデル、スコアへ責任を移さず、人間のレビュー主体が判断と結果への責任を持つ。

6.5 リスクに応じて監査深度を変える

通常Issueや小規模Pull Requestでは軽量T-RDEを行い、重要Issue、Architecture Change、Security Change、Breaking Change、Release前では完全T-RDEを行う。緊急hotfixでも最小のTestとT-RDEを行い、統合後に通常Issueとして再監査する。

7. mainへの統合条件

mainへの統合は、単一の品質点ではなく、次の四群が揃っているかで判断する。

変更の来歴

- Issueに対応している
- Issue単位のBranchで作業している
- Pull Requestがある
- 範囲外の発見を別Issueへ分離している

実行可能な証拠

- 必要なTestが通っている
- local actまたはCIが通っている

- Buildや必要なSecurity Checkが通っている
- Review指摘が解消されている

意味と責任

- 必要なT-RDEが実施されている
- 既知リスクと未解決事項が明示されている
- 人間の統合判断が記録されている

継続可能性

- 必要な仕様・ADR・README・運用文書が更新されている
- 破壊的変更には移行方針がある
- Follow-up Issueが作成されている
- 次のManifestまたはMilestoneへの接続がある

統合判断	条件が不足している場合、スコアで補うのではなく、証拠の追加、変更要求、Follow-upの明示、または統合延期を選ぶ。

8. 限界と継続的な校正

T-RDEには限界がある。意味マップはLLMの自己報告へ部分的に依存し得る。評価者間で分類が異なる可能性がある。重み、閾値、増幅係数、方向評価は暫定的であり、理論から一意に導かれる定数ではない。T-RDEは責任主体でも、客観的な真理判定器でもない。

そのため、品質ゲートは固定された規則として放置せず、Pull Requestの差し戻し、障害、Revert、利用者報告、専門家レビュー、運用上の逸脱などを使って校正する必要がある。ただし、運用データだけで理論上の判断を自動決定してはならない。データが少ないのか、棄却条件が狭すぎるのかといった複数の解釈を人間が再検討する。

T-RDEが意味差分の発見、修復可能性、レビュー品質、責任の明確化を改善しないなら、その実務的主張は弱められるべきである。T-RDE自身も、MDESのSustainから次のManifestへ戻され、監査、改訂、再検証される対象である。

9. 結論

AI支援開発の品質は、コードが動くかどうかだけでは決まらない。どの問題を解こうとしたのか、どの設計意図が保存されたのか、どの曖昧さが実装へ固定されたのか、誰がその変更を承認したのか、後から修復できるのかまで含めて評価する必要がある。

Testは仕様を実行可能にし、CIは変更の再現性を確かめる。T-RDEは、その過程で生じた意味差分、不確実性、責任移動を監査する。MDESは、それらをManifest、Design、Evolve、Sustainの連続へ配置し、監査結果と未解決事項を次の問題定義へ戻す。

したがって品質ゲートが担保すべきものは、絶対的な正しさではない。変化の来歴と判断根拠が見え、人間が受容、修復、棄却、上申を説明可能に選べること、そしてその判断を次の開発サイクルで再検討できることである。

付録A：軽量T-RDEチェック

- □ 元Issueの目的は保存されているか。

- □ 実装がIssueの範囲を超えていないか。
- □ テストで受け入れ条件が確認されているか。
- □ 暗黙の追加、未解決事項、既知リスクが明示されているか。
- □ 未解決事項がFollow-up Issueに分離されているか。
- □ mainに統合してよい既知リスクか。

付録B：完全T-RDE記録項目

- □ 対象Issue / Pull Request / Milestone
- □ 元Manifestの意図
- □ 元Designの意図
- □ 保存· 変換· 補完· 未解決
- □ 逸脱リスク
- □ 意味マップとΔM
- □ テストとの対応
- □ CI結果
- □ Sustainへの影響
- □ 次回更新方針
- □ 判定：Accept / Accept with Follow-up Issue / Request Changes / Reject

参考資料

1. Tomoyuki Kano, “T-RDE: A Semantic Auditing Framework for Repairable Human-AI Collaboration in AI-Assisted Software Engineering,” 2026. ユーザー提供PDF。

2. 「ZYX MDES駆動標準コーディング手順マニフェスト」ユーザー提供Markdown。

3. 「T-RDEと通常テストの違い Clear Chronicle MDES再考版」本レポートの基礎となったスライド。