

**“人が読む”既存ソフトウェア資産を、“AI が読む”開発資産へ
～ドキュメントとコードの不整合や不十分なテストを自動で補完し、
AI-Native 開発^{※1} への転換を加速～**

発表のポイント:

- ◆ 長年運用されているシステムではドキュメント、コード、テストの内容が一致していないことがあり、生成 AI によるシステムの理解・活用が困難です。そこでこれら成果物間のトレーサビリティを自動構築することで、既存ソフトウェア資産を AI が理解・活用しやすい状態(AI-Ready)へ変換する技術確立しました。
- ◆ 本技術により、ドキュメントとコードの不整合などを自動的に検出・修正可能になるとともに、不足しているテスト観点やテストケースを自動補完可能となります。また、予備実験の結果から、不整合の検出率が従来手法の 27%から 80%に向上することを確認しました。
- ◆ これにより、既存システムの理解・改修・検証を生成 AI で一気通貫に実行可能とし、AI-Native 開発への転換を支援することで、ソフトウェア開発・保守の生産性向上に貢献します。

NTT 株式会社(本社:東京都千代田区、代表取締役社長:島田 明、以下「NTT」)は、既存のソフトウェア資産を AI が理解・活用できる状態に変換する技術確立しました。

生成 AI による開発効率化が進む一方、既存システムの追加開発ではドキュメントの陳腐化やコードとの不整合、テスト不足が課題となっています。本技術は、既存システムの仕様書や設計書などのドキュメント、およびコードやテストの依存関係を解析することで、不整合や記載漏れを検出・補完します。さらに、不足するテストの観点やケースを体系的に抽出し、自動で補完します。予備実験により、不整合を 80%検知するとともに、潜在的なバグを 3 件検出することを確認しました。これにより、既存のソフトウェア資産を AI が理解し活用できる状態へ変換し、既存システムの理解・改修・検証を生成 AI で一貫して支援できるようになります。今後、実際の開発・保守案件への適用を進め、AI-Native 開発への転換を支える技術の実用化をめざします。本研究成果は、2026 年 11 月 18 日～25 日に開催される NTT R&D FORUM 2026 ^{※2}に展示予定です。

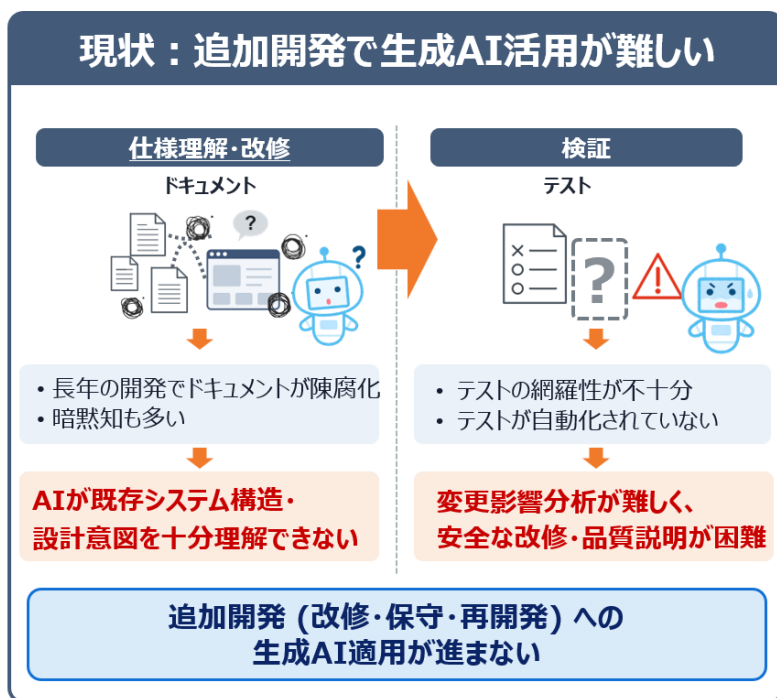


図1. 追加開発の課題

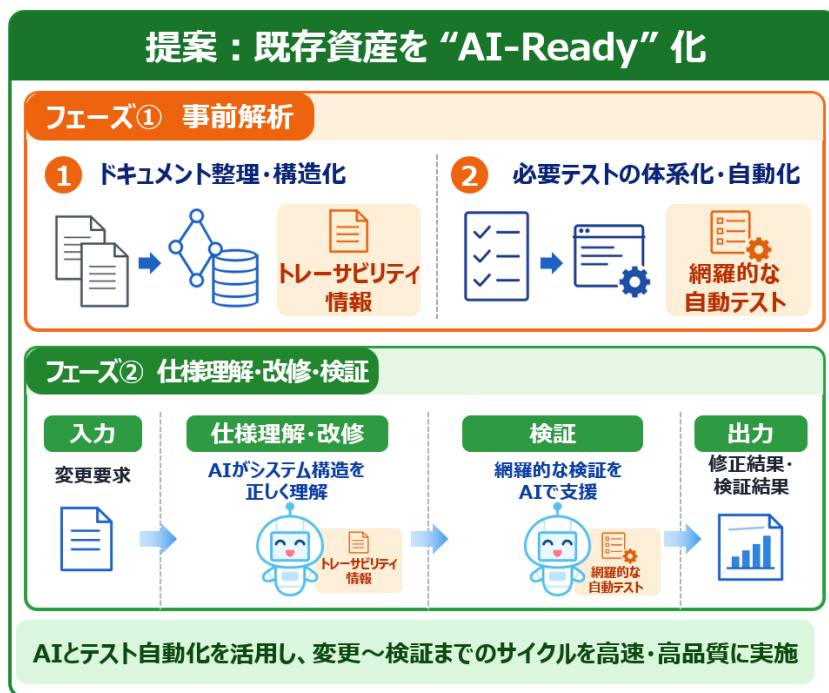


図2. 本取り組みの概要

1. 背景

生成 AI の活用によるソフトウェア開発の効率化が進む一方、その適用は、ソフトウェアを一から構築する新規開発が中心となっています。しかし、ソフトウェア開発の約 6 割^{※3}は、既存システムの改修・保守・再開発を行う追加開発が占めており、追加開発に対しても生成 AI を活用した開発支援による生産性向上が求められます。

一方、長年使われているシステムでは、改修を重ねるうちに、設計書に書かれた内容と実際のコードが一致なくなることがあります。その結果、どこを直せばよいか、修正によって別の機能に影響が出ないかを確認する作業に多くの手間がかかります。こうした状況では、生成 AI もシステムの構造や仕様を正確に把握することが難しくなります。さらに、既存機能を確認するためのテストが十分に整備・自動化されていない場合、変更による影響の検証や、その結果に基づく品質の説明にも多くの手間がかかります。こうした課題が、既存システムの追加開発を担う開発者の大きな負担となっています。

2. 技術のポイント

本取り組みでは、既存ソフトウェア資産（仕様書・設計書・コード・テスト等の成果物の集合体）を AI が理解・活用できる「AI-Ready^{※4}」な状態へ変換するため、以下の 2 つの技術を確立しました。

1 つ目は、仕様書・設計書・コードなどの対応関係を自動で整理し、工程間の不整合や記載漏れを検出・補完する「トレーサビリティ解析技術^{※5}」です。2 つ目は、既存のテストで確認できていない観点を明らかにし、不足するテストを補完する「網羅的なテスト自動生成技術」です。

これらの技術により、既存システムの構造や仕様を AI が理解し、変更による既存機能への影響を検証できる状態を整えることで、将来的な AI-Native 開発への転換を支援します。

① トレーサビリティ解析技術

本技術は、要求工学に関する ISO 等の国際標準・ガイドラインを基に、各成果物の構成・関係を自動的に構造化します。既存システムでは、会議ログなどの開発意図情報、仕様書、設計書、コード、テストケースなどに情報が分散しています。AI がこれらを正しく理解するには、各成果物を機能などの意味のある単位に整理し、そのつながりを明らかにすることが必要です。

各成果物は、ドキュメントごとに記載形式や情報量が異なるため、単純な分割では機能のまとまりを適切に捉えられません。そこで、各文章が特定の機能とどの程度関連するかを評価し、内容に応じて分割粒度を動的に調整することで、意味を保った機能単位への分割を可能にします。

さらに、たとえ同じ機能に対する仕様や設計の記載、およびコードであっても、それぞれ情報量や表現が異なるため、そのままでは正確な対応付けが困難です。そこで前工程の要点を基に後工程の情報を復元し、情報量・粒度をそろえて依存関係を判定することで、工程間の記載漏れや不整合まで高精度に把握します。

これにより、膨大で形式の異なる各成果物を、AI がシステム構造や設計意図を理解できるトレーサビリティ情報へ変換します。

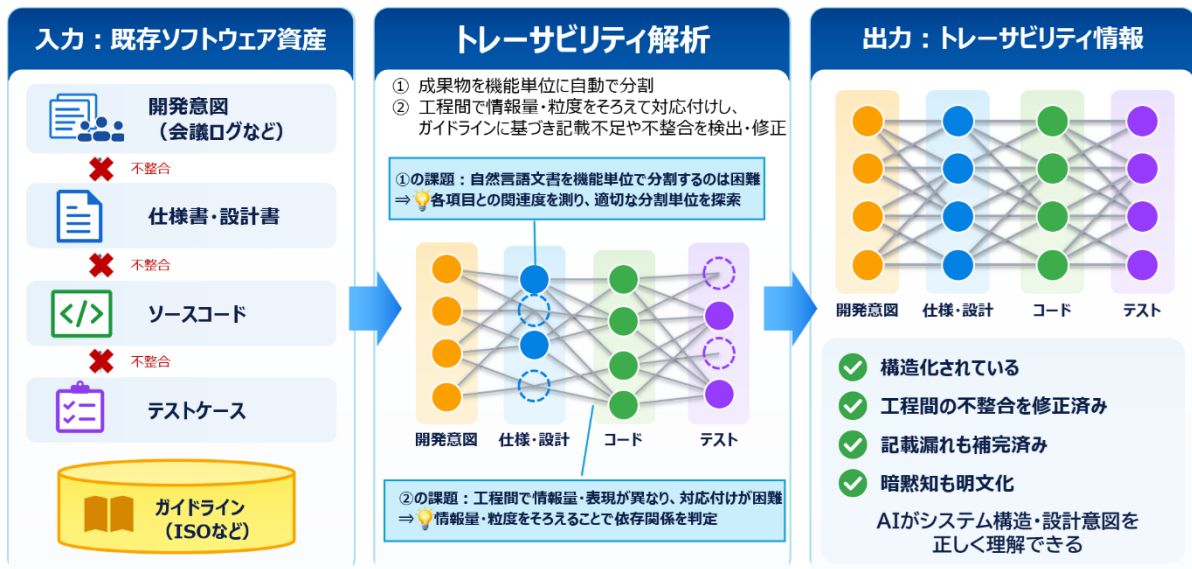


図3. トレーサビリティ解析技術の概要

② 網羅的なテスト自動生成技術

既存システムでは、長年の開発で多数のテストが蓄積されている一方、各テストがどの機能・条件を確認しているのか、重要な観点に抜けがないかを全体として把握することが困難です。また、機能間の依存関係が複雑なため、一部の変更が思わぬ既存機能に影響することもあり、既存機能をすべて手で再テストすると膨大な時間とコストがかかります。

本技術では、整合性を確保した仕様・設計・コードと、ISO 品質特性に基づき、システムのどの機能をどういった観点から確認する必要があるかを整理した「テスト観点空間」を構築します。長年の開発で蓄積されたテストをこの空間に対応づけることで、確認済みの領域と、テストが不足している領域を可視化します。これにより、正常系に比べて不足しやすい異常系や境界値、複数条件の組合せ、機能間連携時の振る舞いなど、十分に検証されていないテスト観点を具体的に特定できます。さらに、不足部分を決定表などのテスト技法^{※6}で体系的に補完することで、既存テスト資産を活かしながら、変更影響を網羅する自動実行可能なテストを生成します。

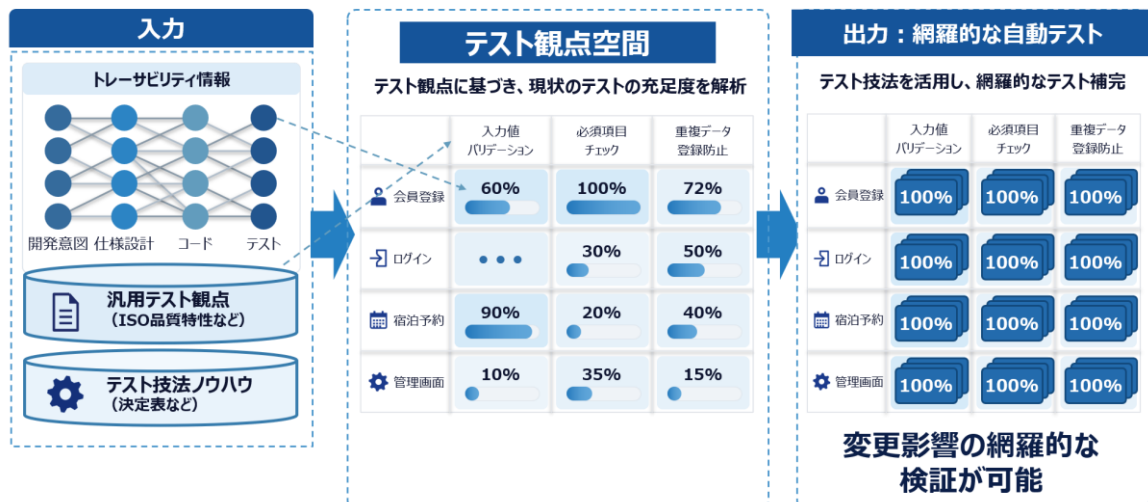


図4. 網羅的なテスト自動生成技術の概要

3. 実験の概要

Macchinetta Framework のサンプルアプリケーション「ATRS」^{※7}を用いて、ドキュメントとコードを準備し、本技術の有効性を確認するための予備実験として、2つの技術の評価実験を実施しました。

① トレーサビリティ解析技術の評価

設計書に意図的に埋め込んだ 41 件の誤り(仕様書、コードとの矛盾、抜け漏れなど)を対象に、誤りを検出できるかを評価しました。その結果、従来手法(RAG^{※8})では 11 件(27%)の検出にとどまったのに対し、提案手法では 33 件(80%)を検出しました。ソフトウェアレビューに関する既存研究では、レビューによる欠陥検出率が平均 68%であった事例が報告されています^{※9}。評価対象や条件が異なるため単純な比較はできませんが、今回の 80%という検出率は、この水準を上回る結果となりました。

この結果は、従来の RAG のように関連ドキュメントを検索・参照するだけでなく、仕様・設計・コードなどの各成果物間の関係をトレーサビリティとして構造化し、あるべき関係と照らし合わせて整合性を確認したことによるものと考えられます。これにより、個々のドキュメントを参照するだけでは把握しにくい工程間の不整合や、設計意図とのずれについても検出することができました。

表 1 設計書に埋め込んだ誤りの検出結果 (対象 41 件)

手法	検出件数	検出率
従来手法 (RAG)	11 / 41 件	27%
提案手法	33 / 41 件	80%

② 網羅的なテスト自動生成技術の評価

意図的に埋め込んだ 6 件^{※10}のバグを対象に評価した結果、4 件(67%)を検出しました。さらに、あらかじめ埋め込んだバグとは別に、既存の潜在バグを 3 件検出しました。これらの潜在バグは、入力検証やエラーメッセージに関するテストが十分に網羅されていなかった箇所を確認されました。一方、仕様書等から生成 AI で直接テストを生成した場合には、これらのバグはいずれも検出できませんでした。

この結果は、生成 AI にテスト生成を一任するのではなく、ISO 品質特性に基づいてテスト観点を体系化し、それぞれの観点到適したテスト技法を適用することで、網羅性の高いテストを生成したことによるものと考えられます。これにより、テスト観点の抜けを抑えながら、幅広い観点から検証することができました。

4. 今後の展開

今後は、さまざまな規模・特性の既存システムへの適用評価を進め、対象とする成果物やプログラミング言語、暗黙知の抽出・補完、品質観点・テスト技法を拡充します。さらに、実際のソフトウェア開発・保守案件に本技術を適用し、システム理解から影響範囲の特定、改修、テスト生成・実行ま



で生成 AI と連携して一気通貫で実行する有効性を検証します。これらを通じて、既存システムを継続的に「AI が理解し、活用できる状態」に維持し、ソフトウェア開発・保守全体の AI-Native 化に貢献していきます。

【用語解説】

- ※1 AI-Native 開発: AI を開発の主体に据え、AI が最大限の能力を発揮できるよう開発環境そのものを最適化するアプローチ。
- ※2 NTT R&D FORUM 2026 <https://www.rd.ntt/forum/2026/>
- ※3 追加開発の割合: 出典は『ソフトウェア開発分析データ集 2022』。本文の「約 6 割」は同データ集に基づく数値。
- ※4 AI-Ready: 既存のソフトウェア資産を整理し、AI が既存システムを理解・活用できるようにした状態。
- ※5 トレーサビリティ: 要件・設計・実装・テストの対応関係を明確にし、それらの一貫性や網羅性を確認したり、変更による影響を追跡したりできる状態にすること。
- ※6 テスト技法: テストする観点や条件を整理し、どのようなテストケースを作るかを体系的に導き出すための方法。
- ※7 Macchinetta Framework / ATRS: (<https://github.com/Macchinetta/atrs>)
- ※8 RAG: 生成 AI が回答を作る前に、外部のデータベースや社内ドキュメントから関連する情報を探し出し、その情報に基づいて正確な回答を生成する技術。
- ※9 What We Have Learned About Fighting Defects, Proceedings of the 8th IEEE International Symposium on Software Metrics (METRICS 2002)
- ※10 対象アプリケーションのコード規模(約 32 KSLOC)を踏まえ、ソフトウェア開発プロジェクトにおける不具合発生実績を参考に設定。

■ 本件に関する報道機関からのお問い合わせ先

NTT 株式会社

サービスイノベーション総合研究所

広報担当

[問い合わせフォームへ](#)