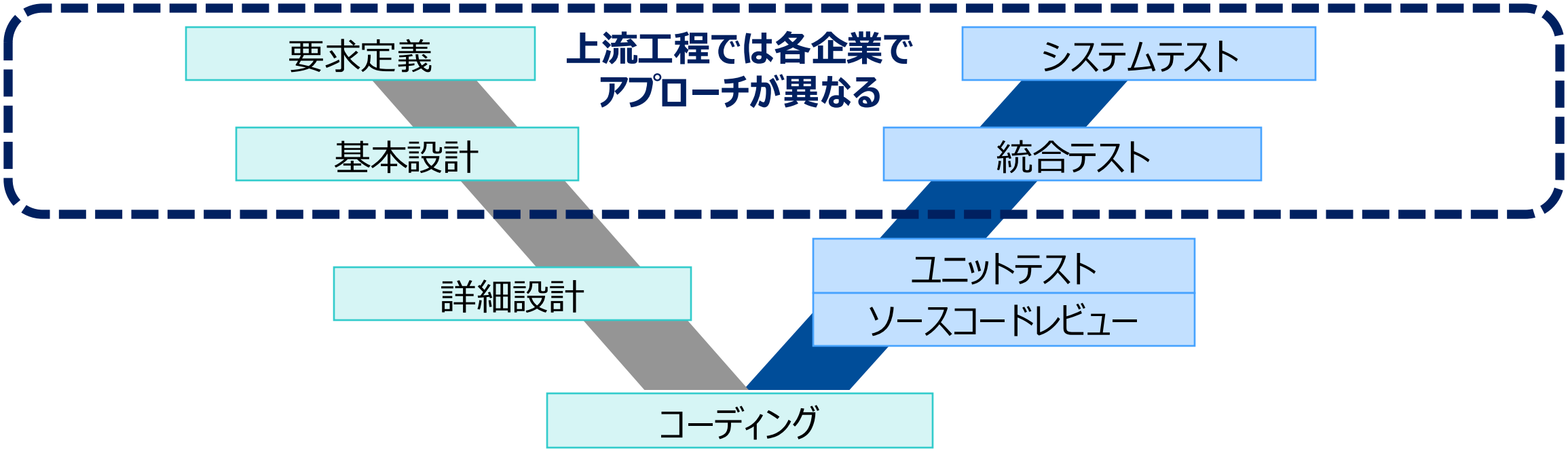
弊社が下流工程をターゲットとする理由（1）

- 開発プロセス全体を生成AIで効率化する場合、上流工程では各企業でアプローチが異なります。そのため各社共通の生成AIの仕組みを適応することは難しいと考えます



krugleプラットフォームが魅力に映ったこと（1）

第１．ローカルLLMとして利用可能なこと。

- モノづくりの現場では「生成AIを活用したい」というニーズは大きくありますが、**クラウド型LLMに対する懸念点**として以下があります。

データの機密性とプライバシーの懸念

バージョン管理の不透明性

サービス依存によるコスト変動リスク

API制限や利用制約の可能性

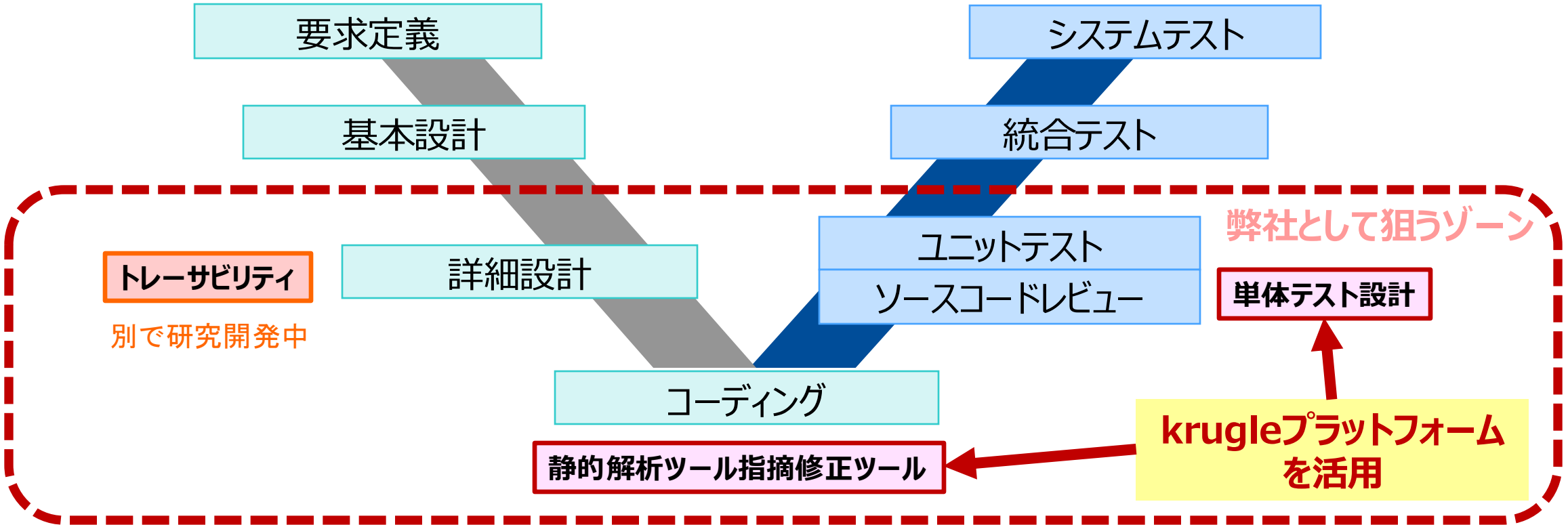
インターネット接続依存

- 上記の懸念を払拭するためのはローカルLLMが解決策であり、krugleはそれを満たしています。

krugleプラットフォームで実現したいこと（1）

- ソースコード生成に特化したkrugleプラットフォームを活用して、弊社では「ソフトウェア開発プロセスの下流工程に特化」して活用するための取り組みを実施しています。

下流工程に対する取り組み：

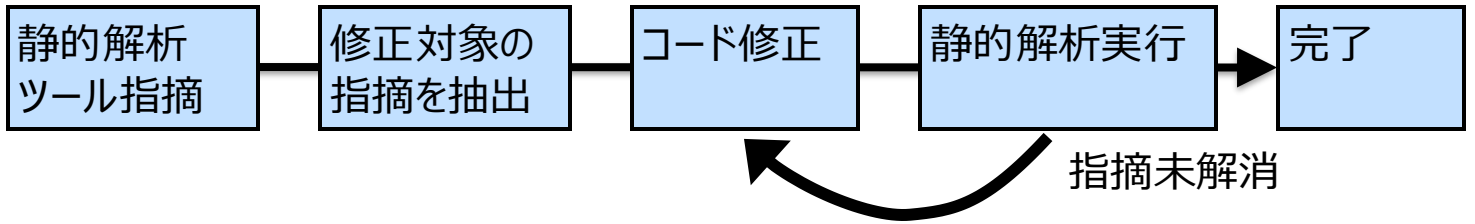


krugleプラットフォームで実現したいこと（2）

- 現在、別の環境で、詳細設計書とソースコードのトレーサビリティの研究をスタートさせていたので、「静的解析ツールの指摘事項を生成AIで修正する」という対応を開始しています。

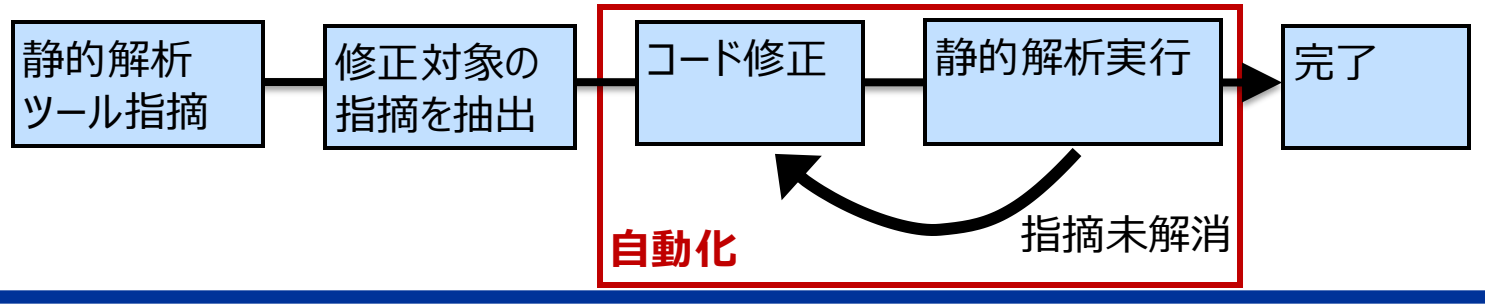
  - 現状、静的解析ツールの指摘修正は、以下の工程を「人力」で対応しています。

従来



  - 本ツールは、修正作業の効率化を実現するため、赤枠の工程を自動化します。

ツール導入



krugleプラットフォームで実現したいこと（4）

- 課題：

  静的解析ツールの指摘を修正しようとすると、全部が完全体で修正できる訳ではなく
  ４つのカテゴリに区別する必要が出てきました。

  - ※これはkrugleの問題ではなく、静的解析ツールの修正アプローチの問題です。

【ランクについて】

本ツールでは違反の対応方針によって、CERT Cの各ルールを以下のランクに分類しています。
本資料ではA、B、Cに該当する修正事例を各ランクひとつずつご紹介します。

ランク	説明	
A	自動で修正が完了し、結果が信頼できるもの	
B	自動で修正が完了するが、人の確認が必要なもの	
C	修正ガイドを提示し、人による補完が必要なもの	
D	修正しないもの	修正すべきでないもの
		修正できないもの
		仕様の環境*で指摘がされないもの

*言語規格: C99, OS: Linux, コンパイラ: gcc, 静的解析ツール: CodeQL, Krugle導入済