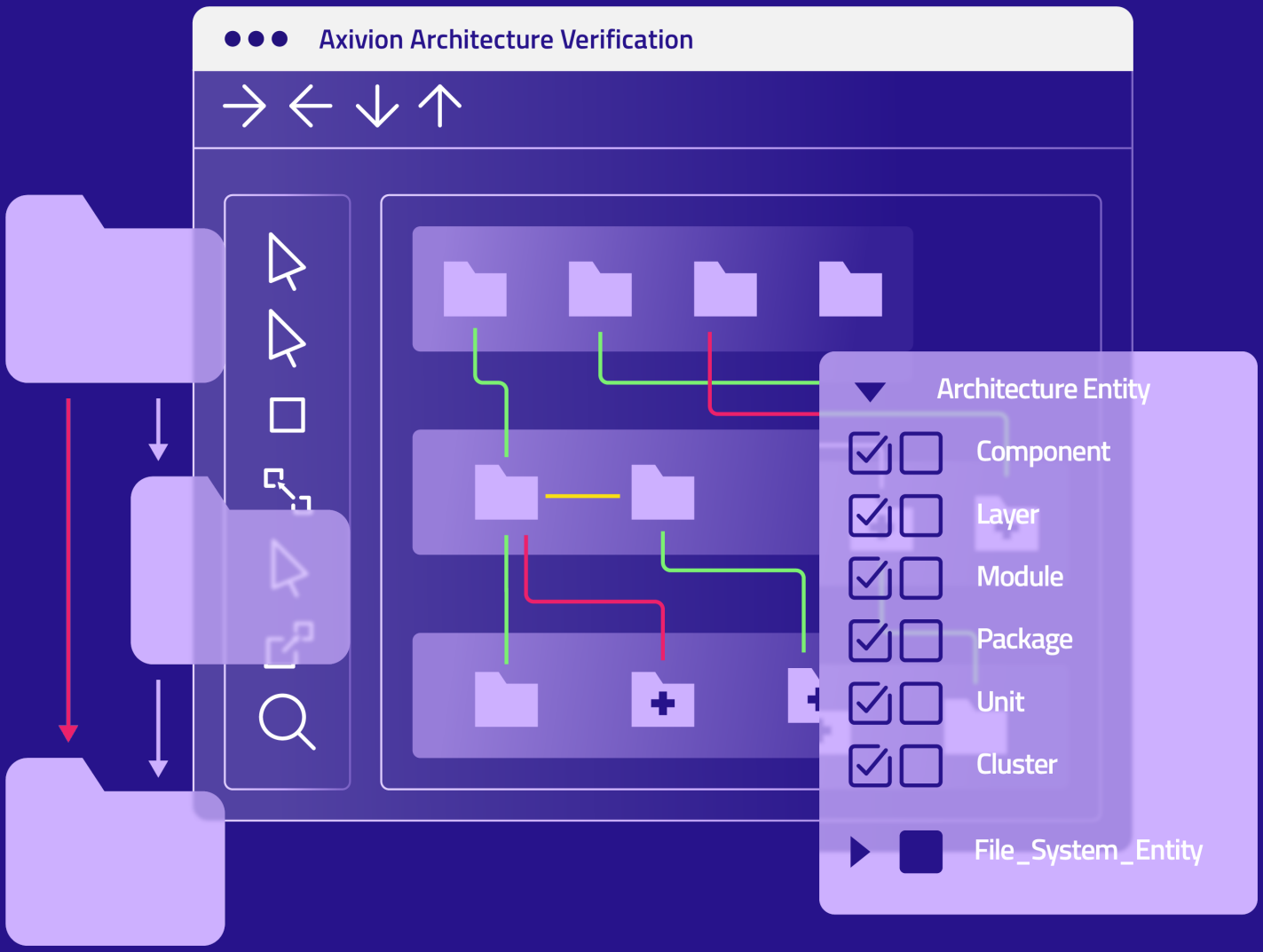
Qt Group

OVERVIEW

# Axivion アーキテクチャ検証

将来にわたって維持可能なソフトウェアのための理想的な基盤を構築



# Axivion アーキテクチャ検証

- コードが定義されたアーキテクチャに準拠していることを保証
- 逸脱を早期に検出し、アーキテクチャ違反がコードベース全体へ広がることを防止
- 既存モデルの利用も、モデルをコードからリバースエンジニアリングで生成することも可能
- 日常的な開発から大規模変更まで対応可能な、信頼性の高い影響分析を実現
- 長期的なソフトウェア保守性を支える、クリーンなソフトウェアアーキテクチャを維持
- 新規メンバーの迅速かつ効果的なオンボーディングを実現
- Freedom from Interference およびソフトウェア分離を証明

## なぜアーキテクチャ検証が必要か？

ソフトウェアアーキテクチャと設計はコードと一致している必要があります。これにより、新機能による影響を議論する際のガイドラインおよび基準としてソフトウェアアーキテクチャを利用できます。それによって初めて、長期的な目標に基づいた計画的な製品開発が可能になります。

Axivionアーキテクチャ検証は、コードがアーキテクチャに準拠していることを保証します。機能アーキテクチャに加え、安全性やセキュリティアーキテクチャの仕様（FFI - Freedom from Interference など）についてもレビューし、準拠性をチェックします。

## 必要なすべての機能を、単一のソリューションで

### アーキテクチャチェック

**ソフトウェアの長期的な基盤を構築・維持します。**

わずかな手順で、コードが意図したソフトウェアアーキテクチャに一致していることを確認できます。これは開発中だけでなく、製品ライフサイクル全体を通して継続的に実施可能です。

### アーキテクチャの再構築

**Axivionにアーキテクチャ検証により、 アーキテクチャ情報を回復・復元できます。**

Axivion　は、ソフトウェアの実際のアーキテクチャを明らかにし、正確なドキュメント作成を支援します。部分的な情報しか残っていない場合はもちろん、まったく情報が存在しない場合でもコードから生成できます。

### アーキテクチャのエキスポート

**コードからアーキテクチャを抽出し、Enterprise Architect へエクスポートできます。**

コードから抽出したアーキテクチャをファイルとして生成し、Enterprise Architectへの出力が可能。変更後の再インポートにも対応します。

# 既存開発環境へのシームレスな統合

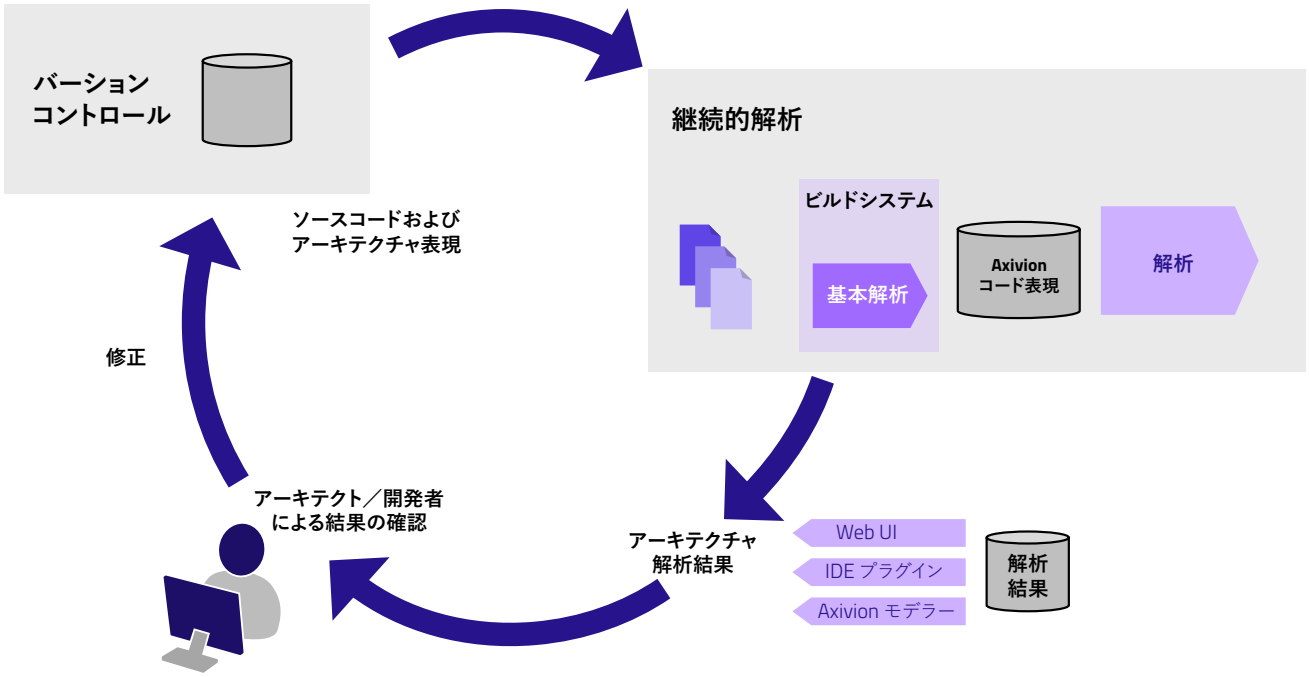
新しいツールを導入する際には、使いやすさが重要です。Axivionアーキテクチャ検証は、ニーズに応じてカスタマイズされ、既存の開発環境にスムーズに統合されます。

豊富なカスタマイズオプションにより、独自に策定したルールやワークフローに合わせたアーキテクチャ検証が可能です。

セットアップ全体を通じて専門家がサポートし、期待以上の成果を実現します。

日々のレポートにより、新たな問題を迅速に特定し、影響を受けたコード部分に直接アクセスして必要な修正が行えます。差分解析により、コード内の新しい問題を即座に確認できます。

複数のスプリントやプロジェクト全体などの長期間レビューする際には、スマートなフィルタリングオプションが特定の問題に焦点を当てるのに役立ちます。




## 主要な機能

- アーキテクチャ検証・差異の特定
- 統合されたモデラー
- UMLツールとのインターフェイス
- AUTOSAR XML (ARXML)のインポート
- Enterprise Architect へのエクスポート
- アーキテクチャの再構築
- 安全およびセキュリティのための
  アーキテクチャビュー

### AxivionでAIワークフローを強化

Model Context Protocol (MCP)コネクターを有効にして、お好みの大規模言語モデル(LLM)をご使用ください。AIエージェントがAxivionのドキュメントや分析結果にアクセスできるようになり、開発者がバグをより迅速に理解・修正できるようサポートします。エージェントをその他のソース(インターネット、社内サーバーなど)に接続するかどうかは、お客様の判断で制御できます。

**注記:** Axivionアーキテクチャ検証は解析に AI を使用していないため、安全クリティカルな環境向けソフトウェアの開発にも適しています。

# わずか数ステップで簡単セットアップ

### 初期セットアップは想像以上に簡単です。

Axivionアーキテクチャ検証を使えば、信頼性の高いアーキテクチャの構築と維持が簡単に行えます。一度セットアップすれば、システムが自動的に早期に差異を検出し、アーキテクチャに影響が出る前に修正が可能です。

### アーキテクチャモデル作成

まず、アーキテクチャモデルを作成します。ツールで利用可能な既存のアーキテクチャモデルをインターフェース経由でインポートするか、 Axivion 同梱のモデラーを使用する、あるいはアーキテクチャを自動的にリバースエンジニアリングします。

### コードモデル作成

ソースコードからコードモデルを抽出します。このコードモデルは、エンティティ（例：ソースファイル）、クラスや関数、そしてそれらの依存関係を表します。

Axivionは、ソースコードプロジェクトを解析することで、コードモデルを自動的に構築できます。

### コードをアーキテクチャにマッピング

次に、コード要素をアーキテクチャコンポーネントに割り当てることで、両者の対応関係を定義します。製品構成やアーキテクチャモデルに応じて、以下の方法で実施できます：

- 手動（例：Gravisモデラーを使用）
- 自動（命名規則や階層情報を利用）
- モデル内の情報を使用（例：タグ付き値）

### 依存関係を解釈

このステップでは、アーキテクチャモデル内の依存関係を解釈し、それが何を意味するかを指定します。 つまり、アーキテクチャ上の依存関係をどのようにコード上の依存関係に対応付けるかを定義します。例えば、コンポーネント A はコンポーネント B 内の関数を呼び出すことが許可されている、という単純な例があります。もちろん、特に UML モデルでは、より高度な依存関係の解釈 （要求されるインターフェースや提供されるインターフェースなど）にも対応可能です。

# ソフトウェアアーキテクチャの再構築

アーキテクチャ検証の基盤となる正確なアーキテクチャモデルが存在しないことは珍しくありません。Axivion アーキテクチャ検証は、プロジェクトのアーキテクチャモデル作成を支援します。

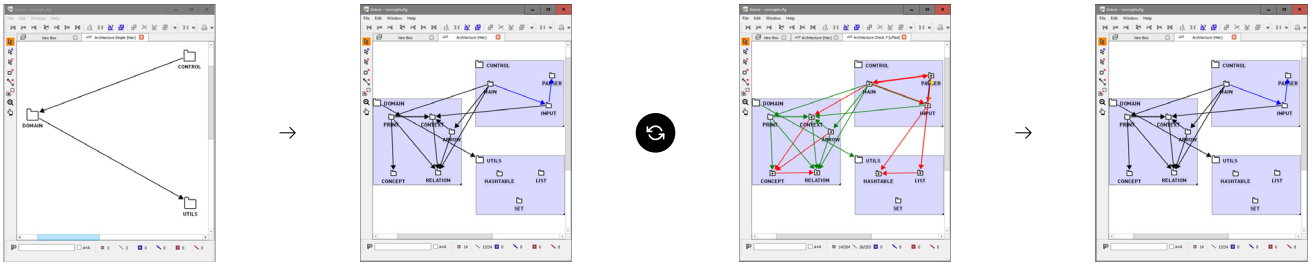
モデルの一部が存在する場合（ CASEツール、XMI、XML など）でも、ドキュメントしか残っていない場合（Word、PowerPoint など）でも、あるいはドキュメントがまったく存在しない場合（ 運用中プロジェクト・終了済みプロジェクトを問わず ）でも、ソフトウェアアーキテクチャ全体の正確で信頼性の高いモデルを構築できるよう支援します。これは特にサードパーティ製コードを組み込む必要がある場合に有効で、そのソフトウェアの構造や役割、変更がどのような影響を与えるかを容易に説明できるようになります。

統合されたAxivionモデラーを使用してアーキテクチャのスケッチを作成し、それを実際の実装へ直接かつ対話的にマッピングできます。さらに、その内容を検証し、得られた知見に基づいて調整を行うこともできます。これらのステップは繰り返し実施でき、最終的に、今後の開発の基盤となる「意図したアーキテクチャを」確立できます。

アーキテクチャの再ドキュメント化によって得られる結果は、アーキテクチャモデルをインポートした場合と同等です。つまり、プロセスの終了時には、再び検証済みのアーキテクチャが利用可能になります。

これにより、継続的なアーキテクチャ検証プロセスへ移行できるだけでなく、レガシーコードやサードパーティ製コードについても、新規開発コードと同等の高い品質基準を適用できるようになります。

### 仮説を記述し、検証・適応・洗練を繰り返してアーキテクチャを検証する



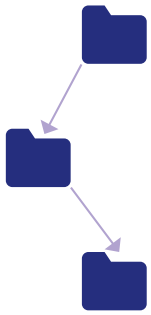
# 日々のアーキテクチャチェック

## アーキテクチャ検証

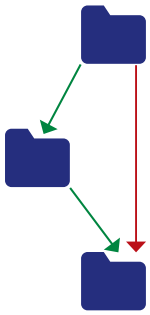
準備が整ったら、Axivionにアーキテクチャを検証させます。

これにより、以下の点を検証して特定します：

- 一致 (Convergences)
- 不一致 (Divergences)
- 欠如 (Absense)



仮説に基づくアーキテクチャモデル

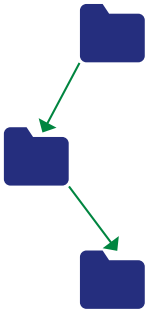


仮説と実装の一致を検証

## 結果の活用方法

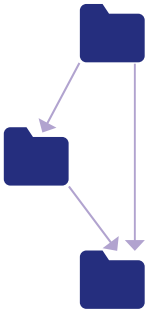
意図したソフトウェアアーキテクチャからの差異を検出するには、自動化プロセスが最も効果的で信頼性があります。しかし、これらの結果にどのように対応するかを決定するには人間の判断が必要です。選択肢は以下の通りです：

**ソースコードを修正する**



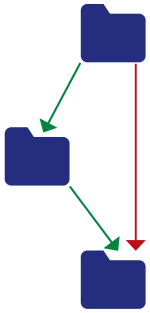
コードが誤りである場合は、コードを修正してアーキテクチャと一致させる必要があります。

**アーキテクチャ仕様を修正する**



コードが正当である場合は、アーキテクチャをそれに合わせて更新する必要があります。

**差異を一時的に受け入れる**



差異を一時的に受け入れ、将来の修正を計画します。

# コードの継続的解析

Axivionアーキテクチャ検証は、現在のアーキテクチャを単に一時点のスナップショットとして把握するためのツールではありません。継続的なソフトウェア解析および開発プロセスの一部として利用することを目的に設計されています。

一度セットアップすれば、Axivionアーキテクチャ検証を日々のレビュープロセスに簡単に統合できます。自動検出された差異を確認し、必要に応じて修正を行うだけで、コードが常にソフトウェアアーキテクチャに適合した状態を維持できます。

エラーが発生しやすい手作業による検証を排除し、信頼性と精度の高い診断情報へ置き換えることで、開発者は問題を早期段階で対処できるようになります。

これにより、後工程での高コストな修正を回避できるだけでなく、ソフトウェアへ加えようとしている変更がどのような影響を与えるかも把握しやすくなります。

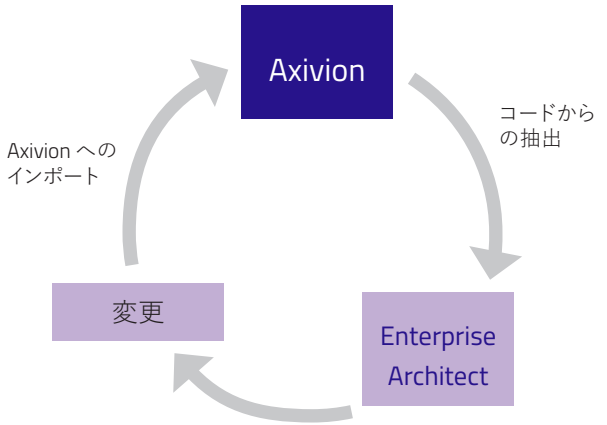
その結果、より長期にわたって維持可能な、信頼性の高いソフトウェアアーキテクチャを実現できます。

# Enterprise Architect へのアーキテクチャエクスポート

AxivionはUMLツール、AUTOSAR XML (ARXML)、その他データ形式化されたフォーマットからのインポートに対応しています。さらに、Enterprise Architect (EA) 向けアーキテクチャエクスポート機能を提供します。これは、Enterprise Architectの利用が求められている開発者や、アーキテクチャを手作業で再作成したくないユーザー、あるいはアーキテクチャ修正時にEAのユーザーインターフェースを利用したいユーザーにとって有用です。

コンポーネント、機能、属性、情報フロー、依存関係など、必要なすべての要素がコードから自動的に抽出され、現在のプロジェクトの状態を反映したアーキテクチャモデルが生成されます。このファイルは、Enterprise Architect で直接開くことができます。

必要に応じて、Enterprise Architect 上でアーキテクチャモデルを修正し、その後 Axivion に再インポートすることで、アーキテクチャ検証に引き続き利用できます。




## Architecture as Code

グラフィカルツールを使用してソフトウェアアーキテクチャを作成する方法に加えて、Axivion は「Architecture as Code」を提供しています。これは、アーキテクチャをPythonコードとして、より具体的にはPythonに組み込まれた内部ドメイン固有言語（DSL: Domain-Specific Language）として記述・マッピングできる仕組みです。

Axivionは、アーキテクチャおよびマッピングを宣言的に定義するためのPythonクラスとメソッド群を提供しています。これらのライブラリを利用することで、まず一般的なルールを定義し、その後で例外を追加する、といった構成を容易に実現できます。これは、名前空間の可視性制御やビルドシステムのフィルタ機能を扱った経験のある開発者にとって、自然に理解しやすいアプローチです。

開発者にとっての利点は、追加のツールを必要とせず、アーキテクチャモデルを既存のコードベースの一部として管理できる点にあります。

# Freedom from Interference: ISO 26262 準拠のアーキテクチャ

現在では、異なるASILやQM分類を持つ複数の安全関連機能を、共通ハードウェア上で共存させることが一般的になっています。しかし、そのためにはISO 26262に準拠した適切なソフトウェアアーキテクチャが必要です。安全アーキテクチャへの準拠は、Freedom from Interference（干渉からの自由）を実現する基盤となります。

Axivionアーキテクチャ検証は、定義されたインターフェースと選択された通信メカニズムの一貫した使用を保証します。アーキテクチャからの差異はソースコード内で即座に強調表示され、未定義の関数呼び出し、データの不正な上書き、またはインターフェースとして定義されていない宣言への参照などが検出されます。

### ツール認定キット

Axivionのツール認定キットには、アーキテクチャ検証を確認・検証するためのテストが含まれています。これにより、機能安全要求が存在する環境において、Axivionが適切に利用可能であることを確認できます。

# 基本仕様

Axivion Suite 7.12 の機能の一部を抜粋しています。技術仕様詳細についてはお問い合わせください。

### 対応言語とコンパイラ

| | |
|---|---|
| 言語 | C, C++, CUDA C++, C#, RUST[1)] |

### プラットフォームOS

| | |
|---|---|
| Host OS | Windows® 10/11 or Windows® Server® 2016/2019/2022 in 64bit<br>x86_64 GNU/Linux® (minimum requirement is glibc2.28 or later), Linux ARM64<br>macOS® (minimum requirement is macOS 14 in 64 bit), macOS® ARM64 |

### プラグイン

| | |
|---|---|
| IDE | Qt Creator, CLion, EclipseTM, Eclipse-based (e.g. e2 studio, Atollic TrueSTUDIO®, CodeWarrior®, DAVETM, STM32CubeIDE, TI Code Composer StudioTM), Microsoft® Visual Studio®, Microsoft® Visual Studio Code®, Generic plugins |

### 対応 UML® ツール

| | |
|---|---|
| UML® ツール | IBM Rational Rhapsody, Sparx Enterprise Architect (via XMI or .qea-files), PlantUML |

### その他

| | |
|---|---|
| 対応ブラウザ | Microsoft® Edge, Mozilla Firefox®, Google Chrome™ |
| 要件 | Java® Runtime (17,21,25) |
| アドオン | Axivionツール認定キット |

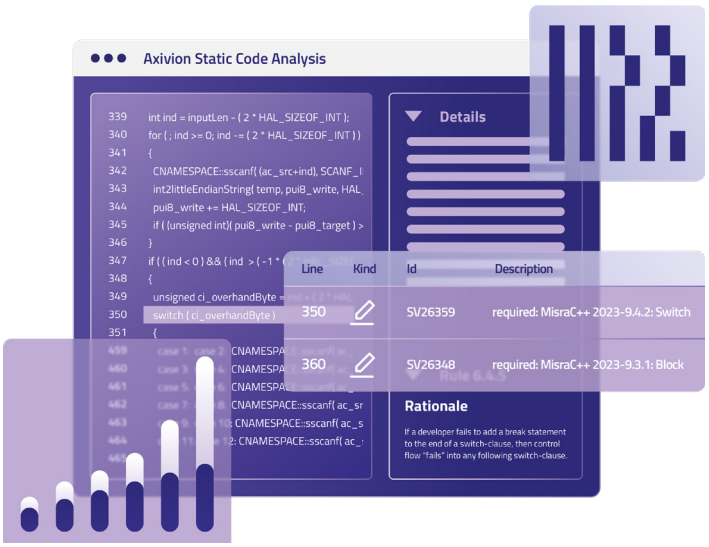
[1)] 一部サポート

技術データは予告なしに変更されることがあります。無断複写・転載を禁じます。すべての会社名および/または製品名は、各市場および/または各国における各メーカーの商標および/または登録商標です。
弊社は常に最新のデータ状況をパートナーにお届けするよう努めています。製品リリース時期と本ドキュメントの公開時期の間に、仕様が変更される可能性があります。

## Axivion Suite

Axivion アーキテクチャ検証は、ソフトウェアプロジェクトの堅実な基盤を提供します。これに加えて、Axivion 静的コード解析を利用することで、コードの詳細な分析が行え、最高水準の要件を満たすことが保証されます。

Axivion 静的コード解析の詳細情報やデモの予約は、当社のウェブサイトをご覧ください。
www.qt.io/ja-jp/axivion



Qt Group

Qt Group Nasdaq Helsinki: QTCOM は、世界のリーダー企業や150万人以上の開発者から信頼されるグローバルなソフトウェア企業として、ユーザーに愛されるアプリケーションやスマートデバイスの開発を進めるお客様をサポートしています。また、UIデザインからソフトウェア開発、品質管理・展開に至るまで、製品開発ライフサイクル全体を通して、お客様の生産性向上を支援しています。 Qt Groupのお客様は70以上の業界で180か国以上に広がっています。Qt Groupの従業員数は世界で約1,100名、2025年の売り上げは2億1,630万ユーロでした。

© Qt Group | 20260512

www.qt.io/axivion-av