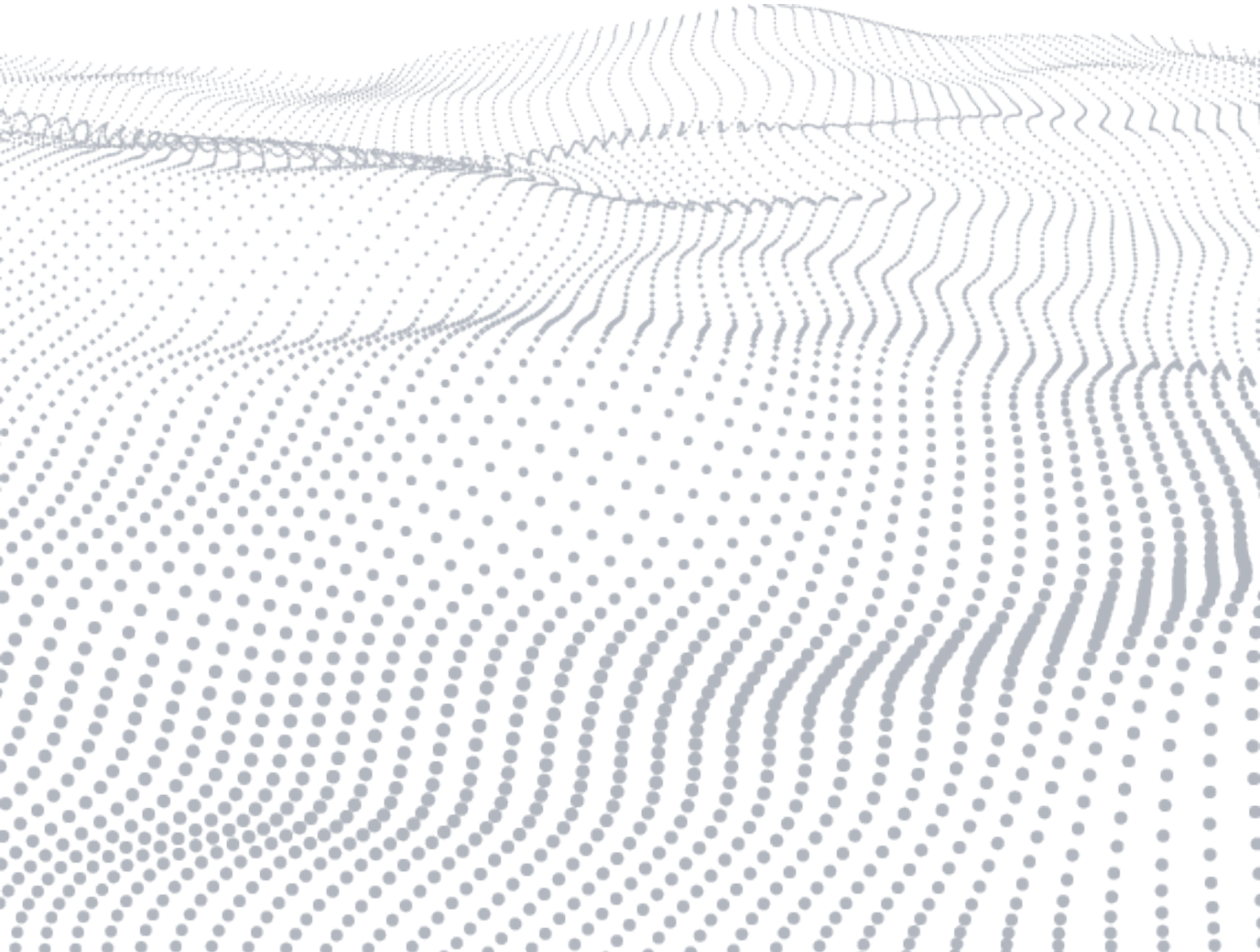
BLACKDUCK®

ホワイトペーパー

ブラック・ダック・ポートフォリオによる ISO 26262 ガイドラインへの適合



目次

現在、平均的な自動車には 1 億行ものソフトウェアコードが含まれていますが、この先 10 年間でその規模は 3 億行にまで増えるものと予測されており[1]、さらに自動車関連の技術革新のうち 90%をソフトウェアが占めるようになると予想されています[2]。ソフトウェアは、ブレーキやパワーステアリングなどのセーフティクリティカルシステムから、ドアやウィンドウなどの自動車の基本的な制御、さらには V2V、V2I や複雑・高度化したインフォテインメントシステム、テレマティクスなど、あらゆるものの制御に使われます。しかし、ソフトウェアの急激な規模拡大には、ソフトウェアの不具合の劇的な増加も伴います。平均的な自動車には最大で 150,000 個ものバグが残るものと推定されており[3]、その多くはブランドを毀損し、お客さまの満足感を損ない、さらに極端な場合には壊滅的な故障へとつながります。トヨタは人気車種プリウスに対して、2018 年に 2 回のリコールを実施しました。これはソフトウェアの不具合によって車両が停止し、高速走行中の衝突リスクが高いと判断したためで、243 万台に影響が出ました[4]。こうしたリコールは Alfa Romeo、Fiat、BMW、Ford、GM、日産などのメーカーでも 2000 年以降、数十回にわたり発生しており、数千万台の車両が影響を受けています[5]。

ISO 26262 の概要

自動車の安全性向上のため、国際標準化機構（ISO）は 2011 年に自動車機能安全規格 ISO 26262 を発行しました。この規格は、システマティック故障やハードウェアランダム故障のリスクを避けるために役立つ、実行可能な要件やプロセスを示したガイダンスとして制定されています。ISO 26262 は、従来の機能安全規格である IEC 61508 をベースとし、電源、センサー等の入力デバイス、データ・ハイウェイ等の通信経路、アクチュエータ等の出力デバイスなど、各種の電気的、電子的な構成要素のアプリケーションセクターにおけるニーズに適合するよう変更を加えたものです。本書は、ISO 26262 に示されたガイドラインを満足するためのブラック・ダック・ポートフォリオの活用方法について説明します。

この規格は 12 のパートで構成され、マネジメント、開発、生産、運用サービス、廃棄等を含む自動車安全ライフ・サイクルの全体を幅広くカバーしています。

車載ソフトウェア開発の課題

自動車分野における現在のソフトウェア開発環境およびプラクティスの大きな特徴として、俊敏性、機能の統合、素早い変化への要求の高まりがあります。また、自動運転車の出現やコネクテッド・カーの増加により、リスクと曝露の範囲も拡大しています。

これに関連し、ブラック・ダックの顧客の間では以下のような課題が見られます。

- ソフトウェアの大規模化と複雑化
  - 特に自社開発およびサードパーティのプラットフォームに含まれるオープンソース・コンポーネントについて、コード再利用のトレーサビリティと特定が複雑化
  - 複雑なコードベースにおけるコーディング規約（MISRA など）の大規模な導入と管理
- サプライ・チェーンおよびサプライヤー・マネジメント
  - 安全およびセキュリティ要件に関するサプライ・チェーン・マネジメント
  - ソフトウェア開発における認証済みツールの使用
- ネットワークに接続するコンポーネントで必要とされる通信の信頼性と堅牢性
- 設計中および製造開始後のサイバーセキュリティに関する脆弱性管理
- 開発に使用する認証済みツールの導入

ブラック・ダック・ポートフォリオの概要

ブラック・ダックのポートフォリオは、品質や安全上の問題を、開発期間やコスト、顧客満足に悪影響を与えることなくソフトウェア・ライフ・サイクルの早期（ソフトウェアプログラムの作成中）に容易に発見して修正できるよう、開発者や経営陣、組織に役立ててもらうためのものです。

ブラック・ダックのソリューションは、品質保証（QA）のための機能テストや性能テスト、セキュリティ監査といった従来のテストを補強するもので、開発チームは作成したプログラムコードに不具合がないかを素早く、容易に、かつ過度の負担がない方法でテストできるため、重要なコードを正しく確実にテストすることが可能です。その結果、開発者は常に新規開発の部分に集中でき、経営陣は可視化を通じて開発サイクル中の問題点を早期に発見し、よりよい意志決定を行うことができます。さらに組織は継続して高品質な製品を市場にいち早く供給できるので、競争上の優位性確保につながります。

ブラック・ダックは、各組織が ISO 26262 に準拠できるよう、開発チームやマネジメントチームのニーズに合わせたソリューションを組み込んだ、業界最先端のデベロップメントテスト・ポートフォリオを提供します。

さらに、Coverity 静的解析は、セーフティクリティカルソフトウェアの開発及びテストに関し、IEC 61508 及び ISO 26262 の該当する要件について TUV SUD Product Service GmbH の認証を受けています。

Coverity 静的解析 – ブラック・ダックは、業界で最も正確かつ包括的な静的解析ソリューションを提供しています。これは C/C++、Java、C# など多くの言語のコードに潜む不具合をより早く見つけて修正するためのもので、世界中のデベロッパーに使われ、コード品質の向上と全体コストの低減に役立てられています。各組織は Coverity Extend SDK（ソフトウェア開発キット）を使い、その独自の要件に合うように解析ルールをカスタマイズできます。Coverity には、カスタム・コーディング・ポリシーを実装するための強力なフレームワークである Code XM も含まれます。静的解析は、コーディングガイドラインが遵守されていることを確認するための形式検証手法として ISO 26262 に規定されており、安全関連のデータが格納されているメモリーロケーションにアクセスする個々のコードを ISO 26262 付録 D（ソフトウェアパーティショニングによる干渉の排除）の規定に従って確認するものです。

Coverity Connect（オンプレミス向け）– この web ポータル・ソリューションで提供される集中型の不具合管理ワークフローにより、開発者と管理者はソース・コードに含まれる不具合の確認から修正までを迅速に実行できます。開発者と管理者は特定の ASIL（自動車安全水準）に関連する不具合を特定し、さまざまなコード・ブランチのどの部分で不具合が発生するかを見つけることができます。自動車業界では非常に多くのコードが再利用されているため、この機能は開発チームに大きな時短効果をもたらします。

Coverity Policy Manager – このソリューションを使うことで、各組織は ISO 26262 に規定された各 ASIL レベルの安全要件に対応する一貫性のあるポリシーを確立し、これを強制することができます。ユーザーは、この規格の重要な要件を満足するための、明確で分かりやすいポリシーを定義できます。このようにポリシーが設定されれば、Coverity 静的解析を利用して各ポリシーに照らしたテストを行い、プロジェクト中のリスク部分を、コンポーネントごと、ASIL レベルごとに素早く可視化できます。管理者や経営幹部はリスクの階層構造を見て、不具合への対処に要する労力を把握でき、具体的な問題点のピンポイントでの掘り下げや、特定の安全要件が満足されているかどうかの確認ができます。

Black Duck – ブラック・ダックの Black Duck ソリューションを使用してソフトウェア・コンポジション解析（SCA）を実行すると、ソフトウェア開発者は既存のコードベースに含まれるサードパーティ / オープンソース・コンポーネントを特定できます。一般的な HMI システムは数百ものソフトウェア・パッケージで構成されることもあり、その多くは Automotive Grade Linux（AGL）や Android などのオープンソース・プラットフォームが元になっています。Black Duck によるソフトウェア・コンポジション解析を利用すると、開発者はどのパッケージが再利用であるかを容易に見分け、場合によってはこれらを「使用実績あり」として安全の範囲から除外することができます。このような形で Black Duck を導入すると、オープンソース・ソフトウェア・パッケージに起因するソフトウェア・ライセンス義務やセキュリティ上の脆弱性についてアラートを受け取ることができるのも利点の 1 つです。

Defensics – ファジング・テストとは、通常の期待される入力を操作し、ターゲット・システム内に故障モードを引き起こすようなテスト・ケースを生成および送信する手法です。クラッシュを引き起こすようなソフトウェアの不具合はサイバーセキュリティ上の脆弱性や安全上の問題の根本原因となることがあり、ファジングによってこうした不具合を見つけることができます。Defensics は業界唯一のプロフェッショナル・グレードのファジング・テスト・ツールであり、2014 年に SSL の脆弱性 Heartbleed を引き起こした異常を特定したことでもよく知られています。

コンサルティング・サービス – ブラック・ダックは世界中で 200 名以上のコンサルタントを擁し、あらゆる業界の顧客に専門的なセキュリティ・サービスを提供しています。特に自動車業界では、脅威分析およびリスク評価（TARA）およびセキュリティ・バリデーション（ペネトレーション・テスト）のサービスを提供しています。また、ブラック・ダックは組織におけるチームの能力開発を支援する専門家によるストラテジー・コンサルティングも提供します。

ISO 26262 要件へのブラック・ダック・ポートフォリオの適用

ソフトウェアレベルでの製品開発に関する一般的なトピック（ISO 26262:2018 Part 6 Section 5）

継続的インテグレーションの使用、および自動ツールの統合

ISO 26262 規格では、開発活動および開発成果物の一貫性をサポートする手法および開発アプローチの例がいくつか指摘されています。特に Example 2 の Note 1 には、この目標を達成する上で自動ツールが果たす役割が示されています。

本書で紹介するブラック・ダックのソフトウェア・インテグリティ・ツールはいずれも自動起動が容易で、Jenkins や Azure DevOps などの一般的な継続的インテグレーション（CI）および継続的デリバリ（CD）ツール用のプラグインが含まれています。

このような自動化の導入を検討している組織は、CI 環境で使用する静的解析ツールについて、以下の点を検討することを推奨します。

- 大量のコードを処理するのにかかる時間（変更を短時間で解析できること）
- 小規模な変更のみの場合は高速なインクリメンタル・モードで動作する機能
- コードベース全体ではなく、部分的なコード変更に対して実行できる機能
- 小規模な（インクリメンタルな）変更ごとに生成される出力の量
- 特定した課題を管理するための高度なワークフロー機能
- 過去の解析結果に新しい問題が見つかった場合はアラートを送信し、開発チームのバックログにアイテムを自動で生成する機能

サイバーセキュリティ

ISO 26262:2018 には、サイバーセキュリティを組み込みソフトウェアの開発時に検討できることが明記されています。ブラック・ダックはサイバーセキュリティ、品質、リスク、および安全に関するトピックを一元的なアプローチで検討することを強く推奨しています。業界のサイバーセキュリティ規格で取り上げられているツールと手法の多くは、安全、品質、信頼性向上のために組織が一般的に導入しているツールおよび活動と重複します。

ISO 26262:2018 は主に機能安全に関するものですが、ブラック・ダックは ISO 26262:2018 Part 2 Section 6.4.5 および付録 E で定義された新しい安全ハザードを生成するさまざまな種類のセキュリティ活動の実施を支援しています。

- 設計活動時に実施するリスク評価 / 脅威モデリング（TARA）（付録 E.3.2）
- 開発中にコーディング規約とセキュリティ上の脆弱性を特定する静的コード解析（付録 E.3.3）
- セキュリティ上の脆弱性を引き起こす可能性のある堅牢性の問題を特定する自動ファジング（付録 E.3.3）
- 検証およびバリデーション・フェーズにおけるユニット、コンポーネント、およびシステム・レベルでの自動 / 手動ペネトレーション・テスト（付録 E.3.3）
- オープンソース・パッケージの特定、およびサードパーティ製オープンソース・ソフトウェア・コンポーネントにおける新しい脆弱性の継続的監視（付録 E.3.4）

また、ブラック・ダックは自動車のサイバーセキュリティに関する包括的なストラテジーを定義した SAE J3061 および ISO 21434 規格にも策定メンバーとして関与しています。

分散開発 – 開発協働契約書（DIA）による要件の補足

ISO 26262:2018 Part 8 には、分散開発環境においてエンドツーエンドの協業を促進するための開発協働契約書（DIA：Development Interface Agreement）に関する条項が数多く規定されています。

通常、ブラック・ダックの顧客はツールの使用、範囲、レポート作成に関する要求事項を DIA に含めています。これには、以下のものが含まれます。

- 静的コード解析から生成されるメトリクス・レポート（HIS メトリクスなど）
- MISRA などのコーディング規約の適用、強制、レポート作成要件の範囲
- 各種セキュリティ・テスト（ペネトレーション・テストやリスク評価 /TARA など）の実施要件
- ファジング・テストのテスト・ケース数および実行時間
- ソフトウェア部品表（SBOM）におけるオープンソース・コンポーネントの宣言

これらの要求事項をサプライ・チェーン上流のソフトウェア・サプライヤーにも適用することにより、ソフトウェア成果物を生成する際に保証活動が確実に実行され、実施データが報告されるようにします。

ツール認証

Coverity 静的解析は、IEC 61508-3 で定められた支援ツールの要件を満たしていることが TÜV SÜD Product Service GmbH によって認証されています。このツールは ISO 26262、IEC 61508、EN 50128、および EN 50657 に従った安全関連ソフトウェア開発での使用条件を満たしています。このツールは T2 に分類され、ISO 26262:2011-8 に従って ASIL D まで使用できます。

Coverity ディストリビューションのドキュメント・パックには必要な機能安全マニュアルが含まれており、この中で、設定ミスのリスク、および偽陽性と偽陰性のリスクを含むツールの動作および故障モードを記述しています。

ソフトウェア・ツールの妥当性確認

ASIL D の開発において、ISO 26262 Part 8-11（「ソフトウェアツールの使用に関する信頼性」）に従ってツールの妥当性確認を実施しなければならないチームは、ビルド環境内でツールの妥当性確認を完了する必要があります。これにより、インストールや設定の誤りによってセーフティ・クリティカルな不具合が見落とされることがないようにすることができます。

Coverity Qualification Kit は、Coverity がお客様のビルド環境内で適切に設定され、動作していることを確認します。この自己診断機能は、Coverity が適切に設定されていることを検証するために実行されたテストを行い、その結果を記載したツール認証報告書を作成します。この認証プロセスは、ISO 26262 Part 8-11.4.9 の勧告に準拠しています。

ソフトウェアのモデリングとコーディングのガイドライン（ISO 26262:2018 Part 6 Section 1）

ソフトウェアレベルでの製品開発フェーズの開始時の手続きの 1 つとして、ISO 26262 にはコーディングとモデリングのガイドラインが設定されており、ソフトウェア開発モジュールの Table 1 に公開されています。ブラック・ダックのポートフォリオはこれらのガイドラインに対応しており、その詳細は以下の通りです。

表の凡例

++：強く推奨される

+：推奨される

o：該当する ASIL に対して、その手法の使用が推奨も非推奨もされていない

ISO 26262:2018 Part 6 Table 1：モデリング / コーディングガイドラインの対象となる項目

項目	ブラック・ダック・ポートフォリオの対応	ASIL			
		A	B	C	D
複雑度低減の強制（1a）	Coverity はコードを解析し、循環複雑度とハルステッドのメトリクスを計算します。HIS メトリクスに従い、定義された複雑度のしきい値を超える関数を特定するポリシーを定義できます。	++	++	++	++
言語サブセットの利用（1b）	Coverity により、MISRA C/C++、AUTOSAR C++、CERT C/C++ をはじめとする一般的な言語サブセットおよびコーディング規約を強制できます。 Code XM 拡張フレームワークを使用して、特定の API または組織のコーディング規約に特化したカスタム・コーディング・ルールを作成できます。	++	++	++	++
強い型付けの強制（1c）	C 及び C++ は、暗黙の型変換や明示的な型変換に対応しているため、Java 等の言語に比較して型付けが弱いとされています。 Coverity は安全でない型付けを自動的に見つけ、その発生を不具合として伝えます。 Coverity Code XM フレームワークを使い、その他のチェック項目を生成できます。たとえば、型付けが許されない場合には、カスタムチェッカーを作成し、型付け動作の都度、不具合を通知することも可能です。	++	++	++	++

項目	ブラック・ダック・ポートフォリオの対応	ASIL			
		A	B	C	D
防御的な実装技法の使用（1d）	Coverity は、関数の返り値がチェックされていないとき、これをエラーとして明示することで防御的プログラミングを強制します。なお、これには単にヌルをチェックするだけでなく、返り値に間違いがないかの検証やテストも含まれます。 これは、Coverity エンジンの「CHECKED_RETURN」ルールです。	+	+	++	++
確立された設計方針の利用（1e）	Black Duck を使用すると、問題のないことが確認済みのコンポーネントのみが開発に使用されているかを検証でき、禁止された、または不明なコンポーネントが見つかった場合にはアラートを受け取ることができます。 Coverity SDK を利用してカスタム解析ルールを作成し、たとえばグローバル変数の使用など、特定の設計方針に対する具体的な違反がないかをテストできます。	+	+	++	++
明白なグラフィックの使用（1f）	該当なし	+	++	++	++
スタイルガイドの使用（1g）	Coverity には、カスタム・コーディング・ポリシーを実装するための強力なフレームワーク Code XM が含まれます。Coverity は、組織に固有のスタイルガイドに合わせてユーザーがカスタマイズできるほか、必要ならカスタム・ルールの作成支援をブラック・ダックのサービスに依頼することもできます。	+	++	++	++
命名規則の使用（1h）	Coverity Code XM 拡張フレームワークを使い、命名規則に対する違反を点検するためのカスタムチェッカーを作成できます。	++	++	++	++
並列処理に関する事項（1i）	利用可能な並列処理関数は MISRA などのコーディング規約で制限されます。一方、Coverity はデッドロック、リソース枯渇、ロックおよびスレッド管理ルーチン使用の矛盾など、並列処理に関するエラーを見つけるためのチェックを多数内蔵しています。	+	+	+	+

ソフトウェアのユニット設計と実装（ISO 26262:2018 Part 6 Section 8）

ISO 26262 規格では、アーキテクチャ設計が完了したらソフトウェアのユニット設計と実装のステージに進みます。

この規格には、正しい実行順序、インターフェイスの整合性、データフローおよび制御フローの正しさ、シンプルさ、可読性、分かりやすさ、ロバスト性を確保するための、ソフトウェアの設計と実装に関するガイドラインが多数含まれます。

開発時には、ISO 26262 の設計原則に準拠したコードが作成されるようにブラック・ダックが開発者を支援します。これは主に、MISRA C/C++ など業界固有のコーディング規約を実装することによって達成されます。

ISO 26262:2012 Part 6 Table 6：ソフトウェアのユニット設計と実装の設計方針

項目	ブラック・ダック・ポートフォリオの対応	ルールマッピング	ASIL			
			A	B	C	D
サブプログラム及び関数には、入力点と出力点がそれぞれ 1 個ずつしかないこと（1a）	Coverity はリターン文を自動分析し、1 個のコンポーネントまたは関数に複数の入力点または出力点が存在していないかを判定します。	MISRA C 2004 ルール 14.4/14.7、MISRA C 2012 ルール 15.1/15.5	++	++	++	++
動的オブジェクトまたは動的変数がないこと、もしあればその作成時にオンラインでテストすること（1b）	Coverity はコードを自動分析し、動的オブジェクトの作成時にその使用が適切にテストされているかを特定します。たとえば、ユーザーがコードを分析し、malloc() が使われているかどうか、使われている場合はそのリターンが必ずチェックされているかを確認できます。	MISRA-C 2012 指針 4.12	+	++	++	++

項目	ブラック・ダック・ポートフォリオの対応	ルールマッピング	ASIL			
			A	B	C	D
変数の初期化（1c）	Coverity はコードの自動テストを行い、初期化されていない変数がないか確認します。	MISRA C 2004 ルール 9.1、MISRA C 2012 ルール 9.1/9.4	++	++	++	++
変数名が重複使用されていないこと（1d）	Coverity は、たとえばローカル変数によるローカル変数の隠蔽、ローカルに隠れたパラメータ、結合衝突などの問題について自動的にパース警告を生成し、開発者が処置すべき不具合として表示します。	MISRA C 2004 ルール 5.5、MISRA C 2012 ルール 5.8/5.9	++	++	++	++
グローバル変数の使用を避けること、また避けられない場合はその使用を正当化すること（1e）	C/C++ 言語では、これらの問題はコーディング規約チェッカーにより対処します。	MISRA-C 2012 ルール 5.1/5.2 とルール 5.8 を併用することで、この動作を効果的に抑制します。	+	+	++	++
ポインターの使用制限（1f）	C/C++ 言語では、これらの問題はコーディング規約チェッカーにより対処します。	MISRA-C 2012 セクション 8.18 により、ポインターの使用に起因するリスクの範囲を制限します。	+	++	++	++
暗黙の型変換がないこと（1g）	C/C++ 言語では、これらの問題はコーディング規約チェッカーにより対処します。	この要件に対する違反は、MISRA-C 2012 セクション 8.10 のルールで特定します。	+	++	++	++
隠されたデータフローや制御フローがないこと（1h）	C/C++ 言語では、これらの問題はコーディング規約チェッカーにより対処します。	制御フローの不一致は、MISRA-C 2012 セクション 8.15 のルールで特定します。	+	++	++	++
無条件ジャンプがないこと（1i）	C/C++ 言語では、これらの問題はコーディング規約チェッカーにより対処します。	無条件ジャンプなど制御フローの問題は、MISRA-C 2012 セクション 8.15 のルールで特定します。	++	++	++	++
再帰がないこと（1j）	Coverityは相当な深さの直接再帰と間接再帰を特定できます。	MISRA-C 2012 ルール 17.2 では再帰を禁止しています。	+	+	++	++

組み込みソフトウェアのテスト（ISO 26262:2018 Part 6 Section 9）

ISO 26262:2018 Part 6 Section 9 はソフトウェアのユニット検証に関する内容で、核となる機能要件をサポートするための数多くの要求事項が示されているほか、ライフ・サイクルのこのフェーズで実施すべき安全関連の活動も取り上げられています。

ブラック・ダックのソフトウェア・インテグリティ・ツールはこれら手法の多くを直接サポートしていますが、それに加え、ブラック・ダックは先行するツール実行からの出力をマニュアル・レビューの一部として活用し、これら出力をソフトウェア・サインオフ・プロセスに統合することを推奨しています。

ISO 26262:2012 Part 6 Table 7：ソフトウェアのユニット検証の方法

項目	ブラック・ダック・ポートフォリオの対応	ASIL			
		A	B	C	D
ウォークスルー（1a）	マニュアル・レビュー・プロセスの場合、レビュー・プロセスの一環として、関連する解析ツールからの出力を参照することを推奨します。 ・Coverity 静的解析の解析結果 ・Defensics のテスト・レポート ・Black Duck のポリシー・レポートおよび部品表（BOM）	++	+	o	o
ペアプログラミング（1b）		+	+	+	+
検査（1c）	不具合管理のベスト・プラクティスとして、検出結果を却下する前に二次レビューなどの機能を各ツールに実装することもできます。	+	++	++	++
準形式検証（1d）	Coverity は、たとえばコード内で異なる条件による遷移を評価し、解析の一部として準形式手法と形式手法の両方を使用し、この情報を利用して解析対象プログラムに対する理解を深めます。	+	+	++	++
形式検証（1e）		o	o	+	+
制御フロー解析（1f）	Coverity はプログラム制御フローの内部グラフを作成し、これを使用して到達不能コード、無限ループ、デッド・コードなど、制御フローに関する問題を検出します。	+	+	++	++
データフロー解析（1g）	Coverity は内部で値の追跡を実行し、データフロー汚染、バッファー・サイズの計算ミス、0 による除算のエラーなどいくつかの種類の不具合を特定します。	+	+	++	++
静的コード解析（1h）	Coverity は抽象表現に基づいて静的コード解析を実行し、コーディング規約違反を検出します。	++	++	++	++
抽象表現に基づく静的解析（1i）	Coverity は抽象表現に基づいて静的コード解析を実行し、単純なコーディング規約チェックよりも複雑な多くの種類の不具合（並列実行の問題、セキュリティの問題、メモリー管理、リソース管理など）を検出します。	+	+	+	+
要件に基づくテスト（1j）	ネットワーク・インターフェイスとファイル・フォーマットに関し、Defensics は要件としてのプロトコル仕様に基づいてテスト・ケースを生成します。	++	++	++	++
インターフェイス・テスト（1k）	ネットワーク・インターフェイスとファイル・フォーマットに関し、Defensics はサポートされるプロトコルおよび独自プロトコルで通信するインターフェイスのテストをサポートしています。	++	++	++	++
故障注入テスト（1l）	Defensics は、標準またはカスタム・プロトコル、あるいは指定したフォーマットに従ってメッセージ・ペイロード、シーケンス、メタ情報を操作することにより、故障注入で使用するテスト・ケースを生成します。	+	+	+	++
リソース使用量の評価（1m）	Coverity はプログラム・スタック・サイズの過剰な割り当て、割り当てられたリソースの解放エラー（リソース・リーク）など、リソース管理に関する多くの種類の問題を検出します。	+	+	+	++
モデルとコードの back-to-back 比較テスト（該当する場合）（1n）	ブラック・ダックのコンサルティング・サービスが提供するコード・レビューおよび脅威モデル作成により、実装後のコードとソフトウェア設計を比較できます。	+	+	++	++

組み込みソフトウェアのテスト（ISO 26262:2018 Part 6 Section 11）

ソフトウェアのユニット・テストは、ISO26262 規格の中でも重要な要求事項の 1 つです。ソフトウェアのユニット・テストを計画し、仕様を定義して実行する必要があります。

これら検証活動は、単にハードウェア - ソフトウェア・インターフェイス仕様への準拠を確認するだけでなく、意図しない機能や属性が存在しないことを確認することも目標にすべきであることがこの規格で規定されています。

規格のこの部分に対処するため、ブラック・ダックは組織に対してファジング・テストの導入を推奨しています。これは、既知の正常なデータを操作し、ターゲット・デバイスのコードに存在する境界条件を発見およびトリガーできるような新しいテスト・ケースを作成する手法です。

Defensics は主にネットワーク・プロトコルやファイル・フォーマットなど、外部システムとの間でデータを交換するインターフェイスに適用できます。独自の通信プロトコルの場合も、Defensics SDK を使用してカスタム・プロトコル実装を作成することにより、Defensics を構成可能なエンジンとして使用できます。

ISO 26262:2018 Table 13：ソフトウェア・テスト実施のためのテスト環境

項目	ブラック・ダック・ポートフォリオの対応	ASIL			
		A	B	C	D
HIL（Hardware-in-loop）（1a）	Defensics は、インストルメンテーション・スクリプトのカスタマイズにより、任意のサードパーティ製 HIL システムと統合できます。この拡張機能はきわめて柔軟で、充実したマニュアルが付属しています。ブラック・ダックは、一般的な HIL テスト・スタンドとの統合を支援するサービスも提供しています。	++	++	++	++
電子制御ユニット（ECU）のネットワーク環境（1b）	Defensics は主にソフトウェア・コンポーネント間インターフェイスをテストするためのもので、CAN や CAN-FD などの一般的なバス・プロトコルを含む 200 種類以上のプロトコルをサポートしています。	++	++	++	++
車両（1c）	Defensics はテスト環境で車両への直接接続に適しており、その目的を達成するため、バス診断用の高度な機能を備えています。	+	+	++	++

ISO 26262:2018 Table 14：組み込みソフトウェアのテストの方法

項目	ブラック・ダック・ポートフォリオの対応	ASIL			
		A	B	C	D
要件に基づくテスト（1a）	Defensics のデータはプロトコル規格および RFC に基づきます。独自プロトコルについては、SDK を使用して Defensics を拡張できます。	++	++	++	++
故障注入テスト（1b）	Defensics は、プロトコル実装の極端なコーナー・ケースを操作することを目的としています。	+	+	+	++

ISO 26262:2018 Table 15：ソフトウェアのユニット・テストでテスト・ケースを作成する方法

項目	ブラック・ダック・ポートフォリオの対応	ASIL			
		A	B	C	D
要件の分析（1a）	Defensics のデータはプロトコル規格および RFC に基づきます。独自プロトコルについては、SDK を使用して Defensics を拡張できます。	++	++	++	++
同類クラスの生成と分析（1b）	Defensics は実際のプロトコルの挙動と対話に基づいてモデル化したテスト・ケースを生成します。	+	++	++	++
境界値の分析（1c）	Defensics は、プロトコル実装の極端なコーナー・ケースを操作することを目的としています。	+	++	++	++

項目	ブラック・ダック・ポートフォリオの対応	ASIL			
		A	B	C	D
知識または経験に基づくエラー推定（1d）	ファジング・テストは、アプリケーション内で予期しない条件（よく知られたソフトウェア脆弱性を含む）をトリガーできることが知られており、これがDefensics の前提となっています。	+	+	++	++
機能依存関係の分析（1e）	ブラック・ダックの Black Duck Binary Analysis を使用すると、依存関係のあるライブラリ（ネットワーク・サーバー、プロトコル・パーサー、ファイル・ハンドラーなど）を特定し、個々のソフトウェア・モジュールにどのような機能が存在するかを調べることができます。これに基づき、ファジング・テストの計画を作成します。	+	+	++	++
運用ユース・ケースの分析（1f）	TARA/HARA を実施する際に、外部インターフェイスに関連するリスクを特定する必要があります。これを使用して、ファジング・テストの範囲と要件についての計画を作成します。	+	++	++	++

参考文献

1. "The race for cybersecurity: Protecting the connected car in the era of new regulation," McKinsey & Company, Deichmann, Johannes; Klein, Benjamin; Scherf, Gundbert; Stuetzle, Rupert; October 10, 2019.
2. "Volkswagen CEO expects software to make up 90 percent of auto industry innovation," Reuters, March 12, 2019.
3. "Tech.View: Cars and software bugs," The Economist, May 16, 2010.
4. "Potentially deadly automotive software defects," Better Embedded System SW, September 25, 2018.
5. 同上。

ブラック・ダックについて

ブラック・ダックは、業界で最も包括的かつ強力で信頼できるアプリケーション・セキュリティ・ソリューション・ポートフォリオを提供します。ブラック・ダックには、世界中の組織がソフトウェアを迅速に保護し、開発環境にセキュリティを効率的に統合し、新しいテクノロジーで安全に革新できるよう支援してきた比類なき実績があります。ソフトウェア・セキュリティのリーダー、専門家、イノベーターとして認められているブラック・ダックは、ソフトウェアの信頼を築くために必要な要素をすべて備えています。

詳しくは www.blackduck.com/jp をご覧ください。

ブラック・ダック・ソフトウェア合同会社

www.blackduck.com/jp