Technical White Paper

ISO 26262 第 2 版の認証を支援する LDRA tool suite®の機能

ASIL A から ASIL D への 費用対効果の高いソフトウェア認証

目次

背景

アダプティブドライバーアシスタンスシステムやアンチロックブレーキシステム、ステアリング、エアバッグといった自動車用の電気・電子（E/E/PE）システムの範囲は広がり続けています。システムの統合と接続性のレベルが高まっていることで、エンターテインメントシステムのような重大でないシステムが、ステアリングやブレーキ、制御システムと同じ通信インフラストラクチャを共有するなど、その急増に伴って多くの重要課題をもたらしています。その結果、要求仕様から設計、実装、統合、検証、妥当性確認、そして構成設定にいたるまで、機能安全開発プロセスを厳格に行うことが必要になります。

ISO 26262 “Road vehicles – Functional safety” は、自動車のE/E/PEシステムの複雑さとそれに関連する公共の安全性が損なわれるリスクが急増したことに対応して2011年[1]に初めて公開され、2018年[2]に更新されました。それ以前の鉄道や医療機器、プロセス産業と同様に、自動車業界でも業界標準の機能安全規格IEC 61508[3]に基づいて規格が定められました。その結果、ISO 26262は自動車の機能安全規格の主流となり、その要件とプロセスが業界全体でますます一般的になりつつあります。

IEC 61508とその派生物によって採用された実践の多くは、商業および防衛アビオニクス産業に遡ることができます。DO-178B/C[4]や、DO-254、DO-278A、およびそれらのサプリメントなどRTCAのシリーズ規格は、構造化された一連のベストプラクティスに準拠することが大規模な信頼性の高いシステムで公共の安全を保護できるということを証明しています。ISO 26262は当初の形でも比較的最近の革新でしたが、他の場所での機能安全基準の確立は、実際に使われているアプリケーションをサポートする洗練された産業を生み出したので、この歴史は重要です。結果として自動車業界は、ISO 26262自体に先んじて確立された実績のあるツールと技術の恩恵を受けています。

このドキュメントでは、規格となる重要なソフトウェア開発と検証プロセスのアクティビティについて説明し、LDRAのツールスイートを使用して自動化がコスト効率の高い方法でコンプライアンスを証明するうえでどのように役立つかを示します。ISO 26262 第2版からの引用は*斜体*[5]*で示しています。*

---

1 https://www.iso.org/news/2012/01/Ref1499.html

2 https://www.iso.org/standard/68383.html

3 IEC 61508:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems

4 RTCA DO-178C Software Considerations in Airborne Systems and Equipment Certification, Prepared by SC-205, December 13, 2011

5 ISO 26262:2018 からの引用、© The British Standards Institution 2018. All rights acknowledged.

自動車の安全インテグリティレベル（ASIL）

DO-178B/CやIEC 61508と同様に、ISO 26262はASIL（自動車安全インテグリティレベル）として知られる多数の危険分類レベルを指定します。ASILに準じた不当な残留リスクを回避するために開発プロセスチェックと安全対策が指定されます。ASILはAからDまであり、ASIL Dは最も危険で要求の厳しいレベルを表し、それゆえ、安全クリティカルなASIL Dシステム（自動ブレーキなど）の製造に伴うオーバーヘッドは、安全性との関わりがほとんどないASIL Aシステム（車内エンターテインメントシステムなど）の製造に必要なオーバーヘッドよりも大幅に大きくなります。

ASILは項目レベルで個々の安全機能の属性として割り当てられます。ここで項目は「*システムまたはシステムの組み合わせで、ISO 26262が適用され、車両レベルで機能または機能の一部を実装するもの（system or combination of systems, to which ISO 26262 is applied, that implements a function or part of a function at the vehicle level）*」として定義されています。安全関連システムの安全機能に割り当てられたASILは関連する危険な事象の属性によって規定され、次の3つの属性によって影響を受けます。

- 状況の頻度（または「露出」）
- 損傷の影響（または「重大度」）
- 可制御性

ISO 26262は、しばしば「ASIL分解」（図1）と呼ばれるプロセスで機能安全要件（FSR）の分解をサポートしており、これはコストと労力の削減に役立ちます。

図1：項目全体で異なるASIL評価の分解は、異なるシステム、要素、およびコンポーネントで起こり得ます

項目全体で異なるASIL評価の分解は、システムからサブシステム、ソフトウェア、そしてハードウェアへと、異なるシステム、要素、およびコンポーネントで起こり得ます。ASIL分解は通常、手動で実行され、十分な技術的独立性をもつ設計要素に割り当てられた冗長な安全要件が必要です。

ISO 26262 第 2 版での変更点

ISO 26262規格の2018年の改訂版では、この規格の初公開以降の技術の進歩に基づいた業界からのフィードバックと更新を反映しています。より詳細な条項の目的と全体的な語彙を拡張するよう再構成され、以下が規格への注目すべき追加です。

- 条項の目的（オブジェクティブ）指向の確認手段
- 安全上の異常の管理
- サイバーセキュリティへの言及
- ハードウェアアーキテクチャメトリックのターゲット値の更新
- ハードウェア要素の評価
- 従属故障分析に関する追加のガイダンス
- フォールトトレランス、安全関連の特殊特性、ソフトウェアツールに関するガイダンス
- モデルベース開発とソフトウェアの安全性分析のためのガイダンス

さらに、次の2つの完全に新しいパートが規格に追加されました。ISO 26262 Part 11は半導体に関連するもの[6]で、ISO 26262 Part 12はオートバイに関連するもの[7]です。

ISO 26262 第2版は12のパートで構成されており、そのうちの3つが、システムレベル（Part 4）[8]、ハードウェアレベル（Part 5）[9]、ソフトウェアレベル（Part 6）[10]で製品開発にフォーカスしたものです。ISO 26262 Part 6は、安全上重要かどうかにかかわらず、自動車用システムおよび機器向けのすべてのソフトウェアの生産に関する詳細な業界固有のガイドラインを提供します。

ISO 26262 のコンテキストでのセキュリティ

自動車業界で使用される組込みソフトウェアの多くは、セキュリティと接続性が実際に議題になってこなかったという理由だけで、セキュリティ要件を考慮していません。自動車用組込みアプリケーションは、静的で、固定の機能、そしてデバイスに固有の実装という傾向がありました。分離が長年セキュリティを十分に保証するものであり、実践とプロセスはその状況に依存してきました。

車内の通信がますます高度化してくるので、車両自体が分離された存在のままである一方で、無害そうなアプリケーションから安全クリティカルなシステムに侵入できるというデモンストレーションは、学術的な見地で考えるべきことを示しています[11]。結局のところ、普通のワイヤーカッターで車の電気システムを損傷することが可能だということでした。

---

6 ISO 26262-11:2018 Road vehicles -- Functional safety – Part 11: Guidelines on application of ISO 26262 to semiconductors

7 ISO 26262-12:2018 Road vehicles -- Functional safety – Part 12: Adaptation of ISO 26262 for motorcycles

8 ISO 26262-4:2018 Road vehicles -- Functional safety -- Part 4: Product development at the system level

9 ISO 26262-5:2018 Road vehicles -- Functional safety -- Part 5: Product development at the hardware level

10 ISO 26262-6:2018 Road vehicles -- Functional safety -- Part 6: Product development at the software level

11 http://www.autosec.org/pubs/cars-oakland2010.pdf -- Experimental Security Analysis of a Modern Automobile, Karl Koscher, Stephen Checkoway et.al.

したがって、重要なことは外の世界との接続なのです。それによって車のシステムに物理的な変更を必要とせずリモートアクセスが可能になるので、事態が劇的に変わります。これについてはMillerとValasekによる“Remote Exploitation of an Unaltered Passenger Vehicle”[12]でデモンストレーションされたものが最も有名です。

おそらく、自動車システムのこの伝統的な分離があるため、ISO 26262の第1版ではセキュリティについて具体的に言及しなかったのでしょう。実際、安全クリティカルな組込みソフトウェアの分野では、セキュリティの懸念は一般的に、機能安全の中核事業とは別の領域であると認識されてきました。しかし、ハッカーがステアリングやブレーキ、エンジン制御システムをリモートで制御する可能性がある場合、セキュリティ脆弱性により安全が危険にさらされることは明らかです。このような状況では安全性とセキュリティは区別できません。

その事実を認めてISO 26262 第2版では、機能安全管理レベルとシステムレベルでの製品開発の両方で、機能安全とサイバーセキュリティ間の相関関係を特定しています。このようなアプローチは、現 SAE J3061 “Cybersecurity Guidebook for Cyber-Physical Vehicle Systems”[13]に記載されているサイバーフィジカル車両システムの「サイバーセキュリティ」と提案されているISO/SAE 21434 “Guidance on potential interaction of functional safety with cybersecurity”[14]で概説されている推奨事項に便利なインターフェイスを提供します。ISO 26262-2:2018 附属書E “Guidance on potential interaction of functional safety with cybersecurity”では「機能安全とサイバーセキュリティの間で起こりうる相互作用」をさらに議論しています。

その他の安全に関するリスクについては、セキュリティ脆弱性が安全を脅かす可能性がある場合、ISO 26262は安全目標とそれに対処するための要件を要求します。安全目標は、その重大性に応じて適切なASILで分類され、その分類を参照して設計され、システムの安全要件に対するコンプライアンスを示すように開発・検証される必要があります。要するに、安全を脅かす各セキュリティ問題に対処するために取られる行動は、それぞれリスク（したがってASIL）に比例する必要があります。

ISO 26262 第 2 版でのプロセスの各条項の目的

Part 4の重要な要素は、システム設計仕様で技術的な安全要件を割り当て、その設計をさらに発展させて項目の統合とテスト計画を導き出す実践です。これは、ソフトウェアを含むシステムのすべての側面に適用され、ハードウェアとソフトウェアの開発方法を「V」モデルを下方に行って明示的に細分化します。

Part 6は製品のソフトウェア側面の開発についてより具体的に述べています。これは、次の問題に関係しています。

---

[12] http://illmatics.com/Remote%20Car%20Hacking.pdf -- Remote Exploitation of an Unaltered Passenger Vehicle, Dr. Charlie Miller & Chris Valasek, August 2015

[13] https://webstore.ansi.org/standards/sae/sae30612016j3061 -- Cybersecurity Guidebook For Cyber-Physical Vehicle Systems

[14] https://www.iso.org/standard/70918.html -- Road Vehicles -- Cybersecurity engineering

- 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
- ソフトウェアの安全性要件の仕様
- ソフトウェアのアーキテクチャ設計
- ソフトウェアユニットの設計と実装
- ソフトウェアユニット検証
- ソフトウェアの統合とテスト
- 組込みソフトウェアのテスト

図2はこの規格から引用したもので、ここではPart 4とPart 6を強調表示しています。

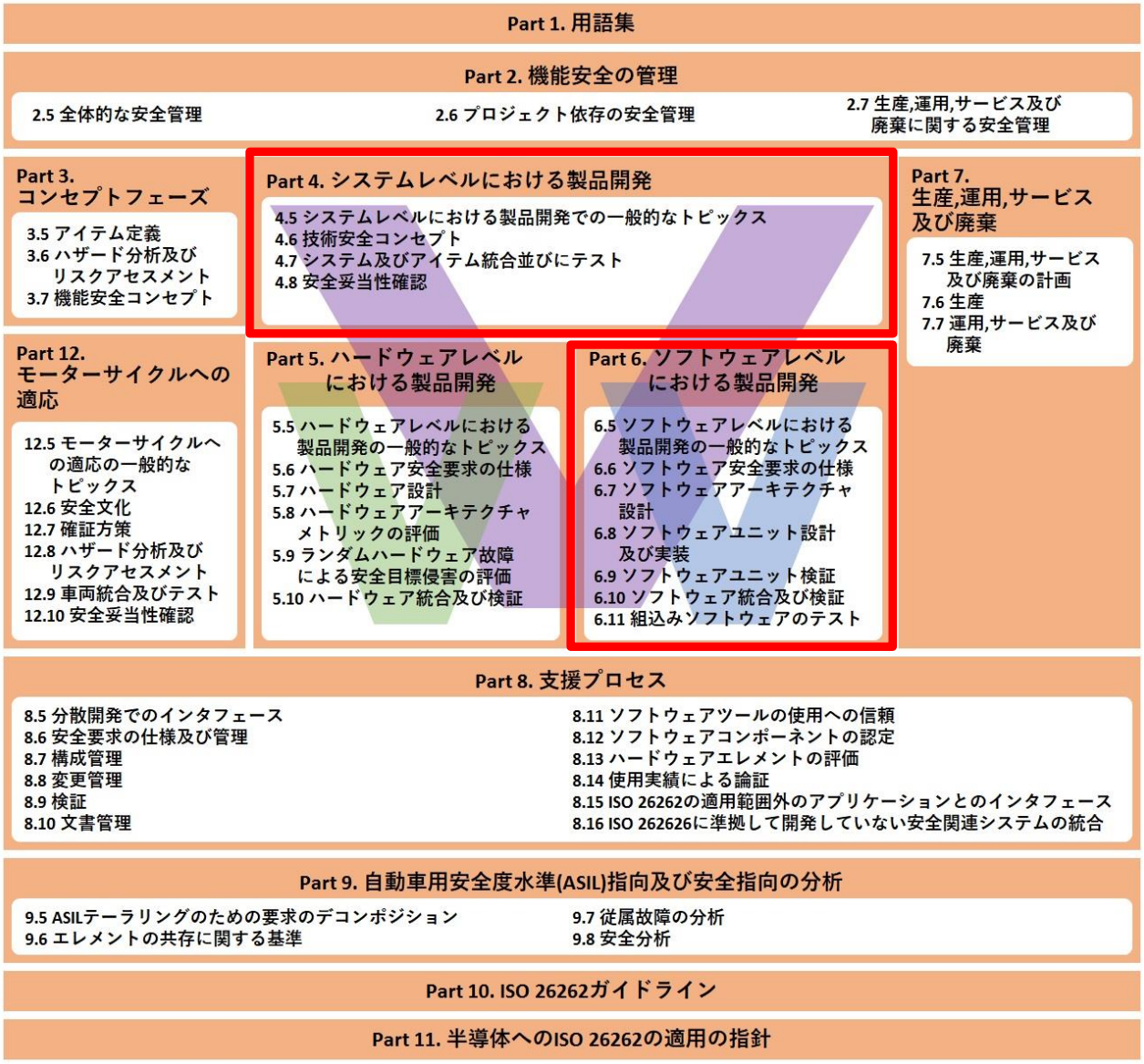


図2：ソフトウェア開発は主にPart 4とPart 6で参照され、ここではISO 26262 第2版全体のコンテキストで示されています。（この日本語版の図は、JARI「機能安全（ISO 26262）」ISO 26262の概要図：http://www.jari.or.jp/tabid/112/Default.aspx からの引用です）

これらの異なる要素は隔離されたサイロとして見られるようには意図されておらず、むしろ規格では、安全要件はアーキテクチャ設計へ、アーキテクチャ設計はユニット設計と実装へ、などトレース可能であると主張しています。ISO 26262ではこのトレーサビリティが、たとえば、アーキテクチャ設計でカバーされない安全要件はなく、アーキテクチャによって要求されない設計要素もないというように、双方向であることも要求しています。このアプローチは、必要とされるすべての機能が存在し証明されているだけでなく、冗長なコードや「フィーチャークリープ」がないことも確認することで、障害のリスクを軽減します。

ソフトウェアの安全性要件とアーキテクチャが定義されると、そのソフトウェアユニットは設計でき、その設計に従って実装できます。開発組織では、プロジェクトの状況に適したコーディングルールを確立する必要があり、実装されたユニットのソースコードを検証し、そのコーディングルールが遵守されていることを確認する必要があります。次に、ソフトウェアユニットは次に動的にテスト （実行）され、ソフトウェアユニットの設計仕様を満たすことと、規定されていない機能が含まれていないことを示す必要があります。次に、これら要件ベースのテスト手順の実行中に、どのコード構造とコンポーネントインターフェイスが実行されていないかを判断するために、構造カバレッジ分析が必要になります。コードの未実行部分ではさらに解析が必要で、その結果、テストケースの追加や変更、不十分な要件への変更、デッドコードやアクティブとならないコードや意図しない機能の削除が必要です。

エンディアンやデータとアドレスワードのサイズなど、低レベルでの実装詳細のバリエーションによってテスト環境とターゲット環境で動作が異なる場合があります。Part 6では、テスト環境をターゲット環境に可能な限り一致させる必要があります。

ISO 26262は、ソフトウェアツールの使用に関する広範なガイドラインを提供しますが、それらの使用を要求していません。しかし、非常に些細なものを除いたすべてのアプリケーションにあるレベルの自動化なしで規格を満たそうとするなら、それは不釣り合いに労働集約的な作業になるでしょう。ツールは、テスト自体の自動化だけでなく、そのテストの正常終了に関連する文書の証拠（すなわち「アーティファクト」）を提供するためのPart 6ほぼすべての側面を支援するために利用可能です。

図3は、ISO 26262ソフトウェア開発ライフサイクルの主要ステージを自動化する方法を示しています。

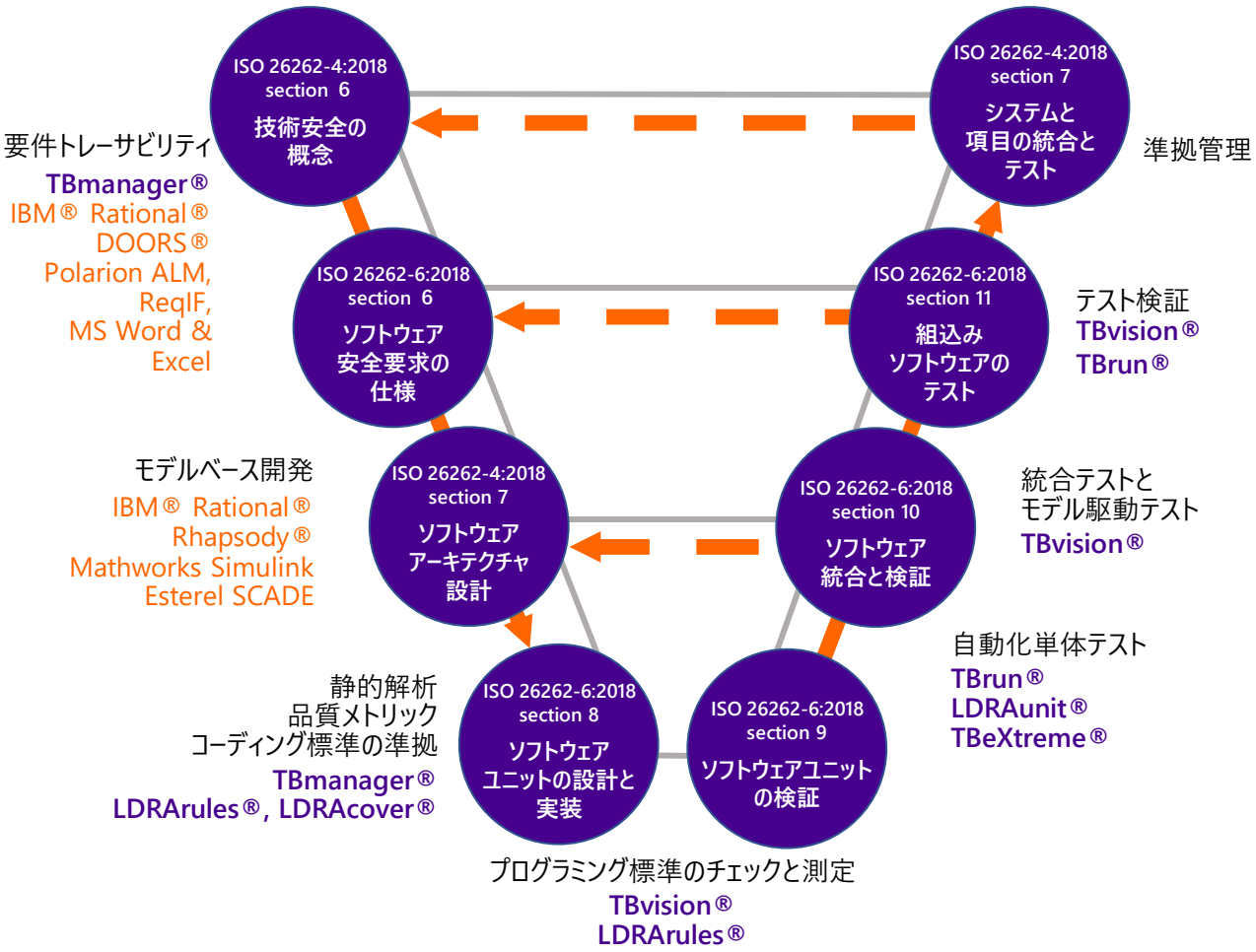


図3：ISO 26262 第2版ソフトウェア開発プロセスガイドラインと自動化ツールチェーンの機能の対応付け

ソフトウェアツールは、ベンダーの経験と専門知識を自動車や他の産業で効果的に活用して認証への道を容易にします。

V字モデル図の各要素を次節で詳述して、ISO 26262 第2版の関連テーブルを用いて参照します。各テーブルは図3に示すVモデルの特定の要素と明確に関連することにも注目してください。

技術安全コンセプト（Part 4、第 6 節）

「技術安全コンセプトは、技術安全要求と対応するシステムアーキテクチャ設計の集合であり、そのシステムアーキテクチャ設計が安全要求の実現に適しているという理論的根拠を与え、ISO 26262-3（非安全要求を考慮して）と設計上の制約に記載されている作業の結果です。」

Part 6はソフトウェア開発の主要なドキュメントですが、Part 4の分野である製品全体のシステム設計のコンテキストで考慮する必要があります。システムアーキテクチャ設計を進展させるために、機能安全要求、技術的安全要求、および非安全関連要求が実装されています。明らかにこれらの作

業成果物にはソフトウェアに関する考慮事項が含まれていますが、このシステム設計のサブフェーズでは、安全関連および非安全関連の要求は、ソフトウェア、ハードウェア、またはハードウェア-ソフトウェアインターフェイスの形でその両方に影響を与えるかどうかにかかわらず、1つの開発プロセス内で処理されます。

この設計段階の製品には、CAD図面やスプレッドシート、テキストドキュメント、その他の多くの成果物が含まれるはずで、その作成にさまざまなツールが関連することに疑いはありません。各要素の状態を管理し、それら要素と以降のフェーズでのトレーサビリティを維持することはプロジェクト管理の悩みの種になるでしょう。

要件管理のための理想的なツールは開発の規模に大きく依存します。ローカルオフィスで開発者が少ない場合には単純なスプレッドシートやWordのドキュメントで十分でしょう。より大きなプロジェクトで、例えば地理的に多様な場所に関係者がいるような場合は、IBMのRational DOORS[15]、シーメンスのPLM Polarion ALM[16]、または標準の要求交換フォーマット[17]をサポートする他のALMツールなどのアプリケーションライフサイクル管理（ALM：Application Lifecycle Management）ツールが役立つでしょう。

Part 4ではまた、双方向トレーサビリティの原理を遵守することが必要です。この要件は開発のすべてのフェーズに共通であり、その自動化については、このドキュメントで関連する他の原則を紹介した後に説明します。

ソフトウェアの安全性要求の仕様（Part 6、第 6 節）

ソフトウェアの安全性要求のサブフェーズの目的は次のとおりです。

- ソフトウェアの安全性要求の仕様または洗練
- 安全関連の機能とその実装に必要なソフトウェアの特性の定義
- ハードウェア-ソフトウェアのインターフェイスの要件の洗練
- ソフトウェアの安全性要求とハードウェア-ソフトウェアのインターフェイス要件がソフトウェア開発に適しており、技術的安全コンセプトとシステムアーキテクチャ設計仕様に矛盾しないことの検証

技術安全要求は洗練され、Part 4の第8節で説明する技術的コンセプトフェーズでハードウェアとソフトウェアに割り当てられます。ソフトウェアの安全性要求の仕様では、ハードウェアの制約とその制約がソフトウェアに与える影響を考慮しますが、主として重点を置くのは、後続の設計フェーズをサポートするソフトウェアの安全性要求の仕様です。

---

15 http://www-03.ibm.com/software/products/en/ratidoor

16 https://polarion.plm.automation.siemens.com/

17 http://www.omg.org/spec/ReqIF/

このサブフェーズは、本質的にPart 4とPart 6の製品全体でのシステム設計間のインターフェイスで、ソフトウェアシステムに具体的に関連する下位レベルの要件の進化のプロセスについて詳しく説明しています。つまり、システム設計のサブフェーズに関連して説明したように、プロジェクトで選択された要件管理ツールを継続的に活用する必要があるのです。

ソフトウェアのアーキテクチャ設計（Part 6、第 7 節）

ソフトウェアのアーキテクチャ設計サブフェーズの目的は次のとおりです。

- 必要となるASILで確立された要件を満たすソフトウェアアーキテクチャ設計の開発
- ソフトウェアの実装と検証のサポート

ソフトウェアのアーキテクチャ設計の生成に使用できる多くのツールがあり、グラフィカルな表現がますます一般的となっています。適切なツールには例えば、IBMのRational Rhapsody[18]や、MathWorksのSimulink[19]、EsterelのSCADE[20]があります。図4は、Part 6の表4を改変したものであり、設計段階と実装開発過程の両方でソフトウェアのアーキテクチャ設計がどのように検証されるかを示しています。

トピック		ASIL			
		A	B	C	D
1a	設計のウォークスルー	++	+	o	o
1b	設計の精査	+	++	++	++
1c	設計の動的部分のシミュレーション	+	+	+	+
1d	プロトタイプ生成	o	o	+	+
1e	形式検証	o	o	+	+
1f	制御フロー解析	+	+	++	++
1g	データフロー解析	+	+	++	++
1h	スケジューリング解析	+	+	++	++
「++」… ASILに対してその手法が強く推奨される 「＋」 … ASILに対してその手法が推奨される 「o」… ASILに対してその手法は推奨されないか、使用を反対される LDRA tool suiteによって満たされます					

図4：LDRA tool suiteの機能とISO 26262-6:2018[21]の表4で指定されたソフトウェアアーキテクチャ設計の検証方法との対応

[18] http://www-03.ibm.com/software/products/en/ratirhapfami/

[19] https://www.mathworks.com/products/simulink.html

[20] http://www.esterel-technologies.com/products/scade-suite/®

[21] ISO 26262-6:2018 の表 4 より、© The British Standards Institution 2018. All rights acknowledged

静的解析ツールは設計検証の一役を担います。設計に従って生成されたコードは、制御フロー・データフロー解析されて、図5に示すように、コードコンポーネントの一部またはすべてとの関係を導き出し、それをグラフィカルに表して意図した設計と比較できるようにします。

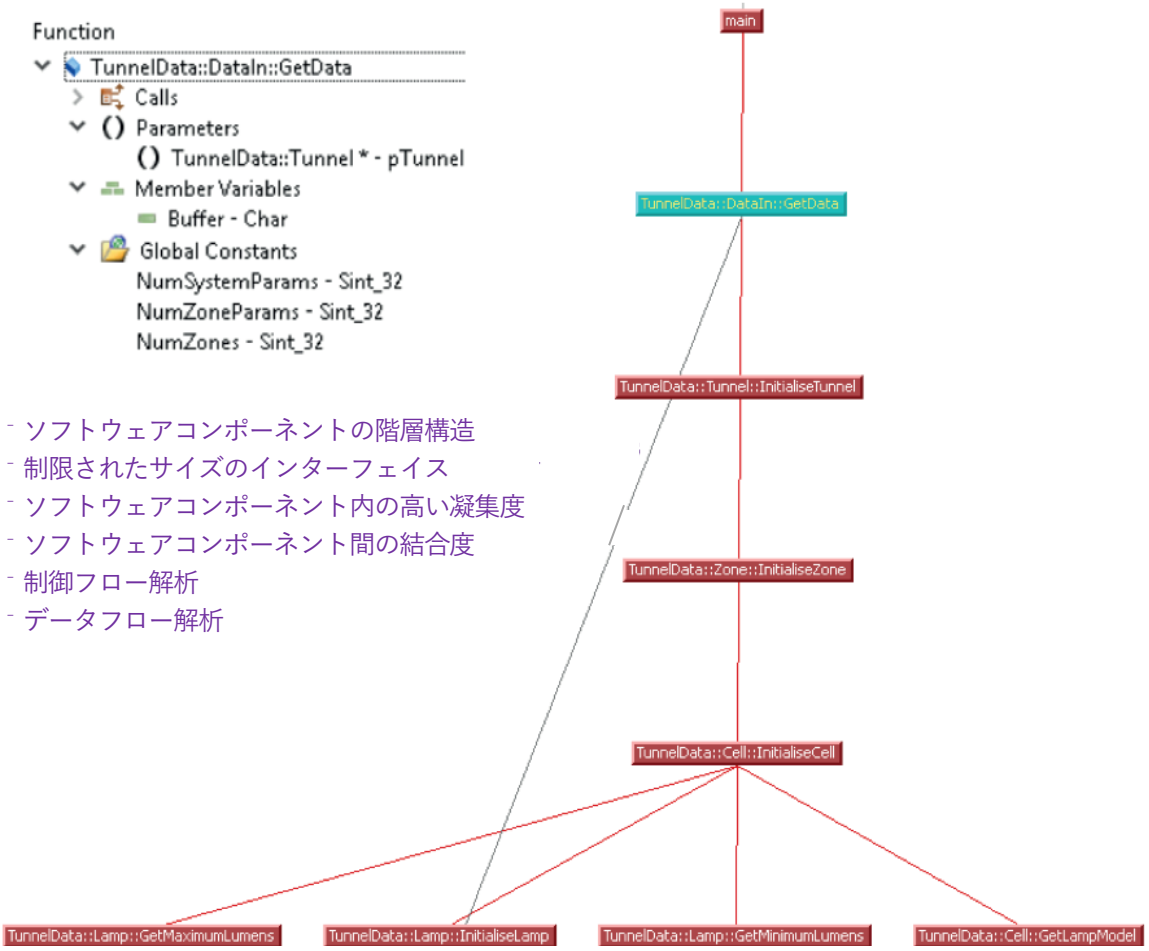


図5：LDRA tool suiteによってソースコードから生成された制御フロー・データフローの図式表現はソフトウェアアーキテクチャ設計の検証を支援します

リバースエンジニアリング

ISO 26262プロセスのフローの多くは、プロジェクトが何もない状態で開始されるか、「使用で実証済み」[22]（長時間の実行で信頼できると示されていること）であるという前提に基づいています。しかし、サードパーティのソースコードを既存の設計に統合する場合のように、長く使用されてきたアプリケーションを改変し「実績が示された使用」という主張が無効になることは十分にあり得ます。ソースコードは存在するがデザインドキュメントが存在しない場合はどうなるでしょうか?

[22] ISO 26262-8:2018 Road vehicles -- Functional safety -- Part 8: Supporting processes

第2版では、Part 8に2つの新しい節、すなわち第15節 “Interfacing an application that is out of scope of ISO 26262（ISO 26262の範囲外のアプリケーションとのインターフェイス）”と第16節 “Integration of safety-related systems not developed according to ISO 26262（ISO 26262に従って開発されていない安全関連システムの統合）” を追加してこのジレンマを認めています。

このような状況では、コードのグラフィカルな表現（図5）が既存のシステムのアーキテクチャの確立に役立ち、ISO 26262の原則に則ってコードの追加を設計および証明できます。

モデルベース開発

LDRA tool suiteは、いくつかのモデルベース開発ツールと統合することができ、1つの例はMathWorks社のSimulinkです。開発フェーズ自体は通常の方法でモデルを作成しますが、一旦ソースコードがそのモデルから自動生成されると統合がより核心に迫ります。

この統合は、主にソフトウェアのユニットテスト、およびソフトウェア統合とそのテスト中に活用されます。そこで、モデルベース開発のトピックはこれらのサブフェーズに関連してこのドキュメントの後半で再検討します。

ソフトウェアユニットの設計と実装（Part 6、第 8 節）

ソフトウェアユニットの設計と実装のサブフェーズの目的は以下にフォーカスされています。

- 確立されたアーキテクチャ設計、設計基準、および要件に従ったソフトウェアユニット設計の開発
- ソフトウェア要件が満たされていることを証明するソフトウェアユニットの実装

コーディングガイドライン

ソフトウェアユニットの設計と実装には、選択したプログラミング言語の使用方法に関するガイドラインの定義を始めとして、いくつか考慮するべき側面があります。図6は、Part 6の表1を改変したものです。規格の表1は実装時に適用されるコーディングおよびモデリングのガイドラインを示しており、ここではそれに対してLDRA tool suiteがコンプライアンスを確認できる場所やコンプライアンスの確認を支援できる場所を示します。

トピック		ASIL			
		A	B	C	D
1a	低複雑性の実施	++	++	++	++
1b	言語サブセットの使用	++	++	++	++

1c	強い型付けの実施	++	++	++	++
1d	防御的実装技法の使用	+	+	++	++
1e	高信頼な設計原則の使用	+	+	++	++
1f	明確な図形表現の使用	+	++	++	++
1g	スタイルガイドの使用	+	++	++	++
1h	命名規約の使用	++	++	++	++
1i	並列性の見地	+	+	+	+

「++」… ASILに対してその手法が強く推奨される
「＋」 … ASILに対してその手法が推奨される
「o」… ASILに対してその手法は推奨されないか、使用を反対される
LDRA tool suiteによってサポートされます
LDRA tool suiteによって検証されます

図6：LDRA tool suiteの機能とISO 26262-6:2018[23]の表1で示すモデリングおよびコーディングガイドラインがカバーするトピックの対応

これらのガイドラインはすべて、結果として得られるコードの信頼性を高め、エラーを発生しにくくし、テストを容易にし、そして、保守を容易にするという理由で正当化できます。例えば、

- 低複雑性（図6、1a）は、複雑なコードでは解読や保守が容易ではなく、エラーの影響を受けやすくなるので重要です。低複雑性を強化するには、定量化して「合格/不合格」の基準を確立する必要があります。
- MISRA Cなどの言語サブセット（図6、1b）は、プログラミング言語の使用を、問題を引き起こす可能性が最も低いことがわかっている要素に制限します。
- スタイルガイド（図6、1g）を使用すると、誰が書いたかによらずコードの外観が一貫していることを確認できます。これによって保守や変更がはるかに容易になり、エラーが発生しにくくなります。

このようなガイドラインの遵守を強制する従来のアプローチはピアコードレビューを行うことでしょう。これは開発プロセスにおいて、まだ存在するかもしれません ー チームメンバー間の学習の助けとなるなど非常に役立ちます ー しかし、退屈なチェックの自動化は、はるかに効率的でエラーは発生しにくくなります（図7）。

図7：LDRA tool suiteでの違反したコーディング規則やガイドラインの強調表示

[23] ISO 26262-6:2018 の表 1 より、© The British Standards Institution 2018. All rights acknowledged

Part 6ではMISRA言語サブセットを強調表示していますが、例としてだけのことで、使用するべきであるという指示ではありません。多くの利用可能なコーディング標準のセット（サイドバーを参照）があります。そして、特定のセットをベースとして選択する場合、対象となるアプリケーションにより適したものにするために、それを操作、調整、追加することは完全に容認されます。そしてツールも、これらの調整に対応できなければ役には立ちません。

言語サブセット

多くの言語サブセット（「コーディング標準」）があり、それぞれ特徴がありますが、同じ言語に対しては強い類似性をもっています。最も普及している標準は以下です。

C
MISRA C:1998
MISRA C:2004
MISRA C:2012
CERT C
CWE

C++
MISRA C++:2008
JSF++ AV
HIC++
CERT C++

Java
CWE
CERT J

ソフトウェアのアーキテクチャ設計とユニットの実装

Part 6のコーディングガイドラインがソフトウェアアプリケーションの「ブロック」に相当する場合、ソフトウェアのアーキテクチャ設計の原則により、どこに「壁」を構築するべきかが定義され、そしてユニット実装のガイドラインはこれらの「壁」をどのようにして構築すべきかを定義します。

コーディング、アーキテクチャ設計、ユニット実装に関する適切なプロジェクトガイドラインを確立することは、明らかに3つの別々のタスクですが、設計の実装を担当するソフトウェア開発者は、レンガ造りの壁を構築するのと同じようにその3つを同時に気を配る必要があります。図8と図9は、Part 6で必要とされるアーキテクチャ設計とユニット実装の原則がコーディングガイドラインと同様に静的解析によって確認できる方法を示しています。

図8はPart 6の表3を改変したものであり、元の表にLDRA tool suiteがコンプライアンスの確認を支援できる部分を重ね合わせています。

原則		ASIL			
		A	B	C	D
1a	適切な階層構造	++	++	++	++
1b	ソフトウェアコンポーネントのサイズの制限	++	++	++	++
1c	インターフェイスのサイズの制限	+	+	+	++
1d	ソフトウェアコンポーネント内の高い凝集度	+	++	++	++
1e	ソフトウェアコンポーネント間の緩い結合度	+	++	++	++
1f	適切なスケジューリング特性	++	++	++	++
1g	割込みの使用の制限	+	+	+	++
1h	ソフトウェアコンポーネントの適切な空間分離	+	+	+	++
1i	共有資源の適切な管理	++	++	++	++
「++」… ASILに対してその手法が強く推奨される					

「＋」 … ASILに対してその手法が推奨される
「o」… ASILに対してその手法は推奨されないか、使用を反対される
LDRA tool suiteによってサポートされています

図8：LDRA tool suiteの機能とISO 26262-6:2018[24]に示される「表3：ソフトウェアアーキテクチャ設計の原則」との対応

先行するコーディングガイドラインとしては、これらの原則はすべて結果として得られるコードの信頼性が高まり、エラーが起こりにくく、テストが容易で、保守が容易である、という考え方に基づいています。
例えば、

- ソフトウェアコンポーネントのサイズの制限（図８、1b）は、特に、大きくてとりとめのない関数は解読や、保守、テストが困難であるため、エラーの影響を受けやすくなるということで重要です。
- インターフェイスのサイズの制限（図8、1c）は、同様の理由で重要です。
- ソフトウェアコンポーネント内の高い凝集度（図8、1d）は、ソフトウェアプログラムの異なるコンポーネントが他と相互に関連する強度と一体性の程度を示します。高い凝集度は、ソフトウェアプログラムのモジュール間の密接な結合によって生じ、そのモジュールに割り当てられたさまざまなタスクをいかに速く実行できるかに影響を与えます。

図9はPart 6の表6を改変したものであり、元の表にコンプライアンスが自動的に確認できる部分を重ね合わせています。

原則		ASIL			
		A	B	C	D
1a	手続き、関数の１入口・１出口	++	++	++	++
1b	動的なオブジェクトや変数なし、でなければそれら生成時のオンラインテスト	+	++	++	++
1c	変数の初期化	++	++	++	++
1d	変数名を多重使用しない	+	+	++	++
1e	グローバル変数は使用しない、もしくは使用の正当性を示す	+	+	++	++
1f	ポインタの使用の制限	o	+	+	++
1g	暗黙の型変換を行わない	+	+	++	++
1h	隠れたデータフローや制御フローなし	+	+	++	++
1i	無条件分岐なし	++	++	++	++
1j	再帰なし	+	+	++	++

「++」… ASILに対してその手法が強く推奨される
「＋」 … ASILに対してその手法が推奨される
「o」… ASILに対してその手法は推奨されないか、使用を反対される
LDRA tool suiteによって検証されます

[24] ISO 26262-6:2018 の表 3 より、© The British Standards Institution 2018. All rights acknowledged

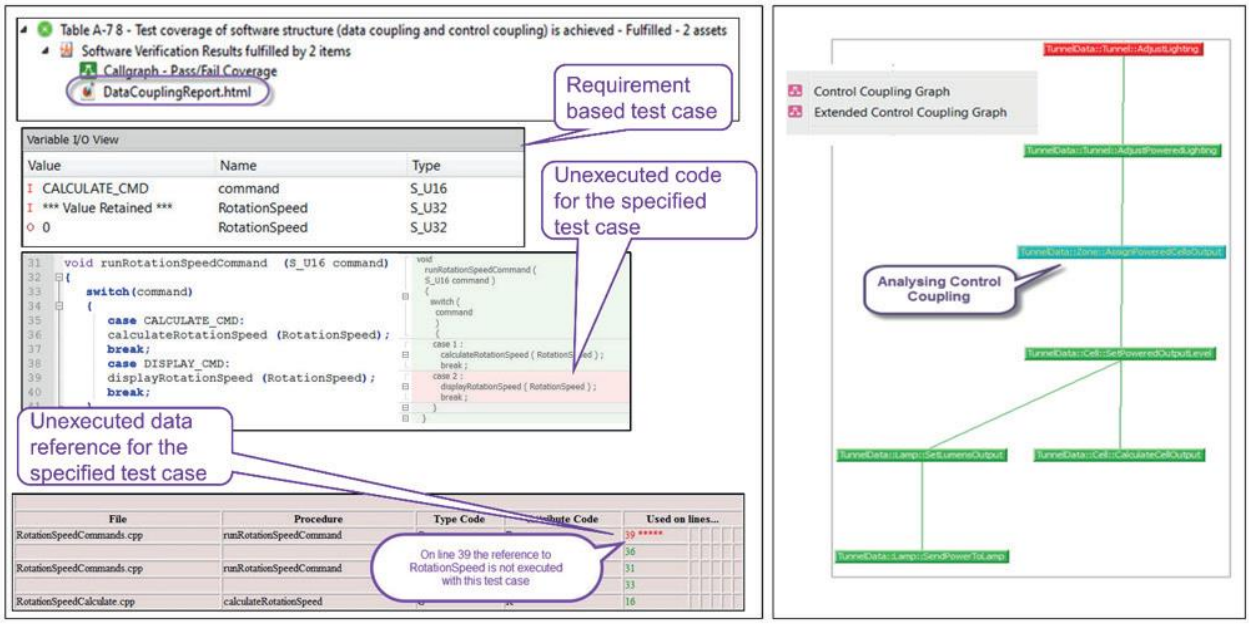
図9: LDRA tool suiteの機能とISO 26262-6:2018[25]に示される「表6：ソフトウェアアーキテクチャ設計の原則」との対応

繰り返しますが、これらの原則はすべて結果として得られるコードの信頼性が高まり、エラーが起こりにくく、テストが容易で、保守が容易であるという考え方に基づいています。例えば、

- 変数の初期化（図9、1c）は、初期化されていない変数をコンパイラベンダーによって異なる方法で扱うことができるというC、C++言語の「未定義の動作」の例を参照しています。0に初期化される場合もありますが、開発者がそのことを利用していて後でそのコードが別のハードウェアに移植される場合、期待どおりに振る舞わない可能性があります。
- グローバル変数の回避、もしくはその使用の正当化（図9、1e）。グローバル変数に関する問題は、すべての関数がグローバル変数にアクセスできるので、どの関数が実際にそれらを読み書きするのかを把握することが難しくなってテストや保守が困難になることです。静的解析は役立ちますが、グローバル変数を避ける方がはるかにクリーンです。
- 再帰の回避（図9、1j）。再帰関数は理解しにくく、多くの場合リソースの使用状況を予測できません。

一般に、完全に統合されたツールスイートを使用すると、規格が要求する優れた実践が、コーディングルール、設計原則、またはソフトウェアアーキテクチャ設計の原則のいずれであれ、それに遵守していることを保証できます。

たとえば、ソフトウェアコンポーネントのサイズや、複雑度、凝集度、結合度を評価して制御するために、メトリックが生成されます。複雑度メトリックは、インターフェイス解析とデータオブジェクト解析による凝集度の評価、およびデータ結合と制御結合の解析による結合度、からの成果物として生成されます（図10）。



[25] ISO 26262-6:2018 の表 6 より、© The British Standards Institution 2018. All rights acknowledged

図10：LDRA tool suiteによる制御結合およびデータ結合解析の出力

コード実装フェーズ全体を通じて静的解析の継続した適用は、開発チームへの有力な後ろ盾になります。実際、ISO 26262の初心者である開発者にとって、ツールの使用は多くの場合、違反が発生した場所を特定することから始まり、違反が存在しないことを確認するところにまで到達します。

ソフトウェアのユニット検証（Part 6、第 9 節）およびソフトウェアの統合とテスト（Part 6、第 10 節）

ソフトウェアユニットの検証のサブフェーズの目的は以下にフォーカスされています。

- ソフトウェア要件が満たされていることを示す証拠の提供
- 定義された安全対策が適切に実装されていることの検証
- 割り当てられたASILに従ってユニット設計が正しく実装されている証拠の提供
- 望ましくない特性や機能がないことを示す証拠の提供

ソフトウェア統合とテストのサブフェーズの目的は以下にフォーカスされています。

- システムの完成までのソフトウェア要素の定義と実装
- 定義された安全対策が適切に実施されていることの検証
- ソフトウェアアーキテクチャ設計が正しく実装されている証拠の提供
- 望ましくない特性や機能がないことを示す証拠の提供

Part 6の準拠に関してこれまでに説明した手法は、主に静的解析、つまりソースコードの自動的な「検査」にフォーカスしています。静的解析手法が、Part 6のガイドライン順守の検証のコーディング、アーキテクチャ設計、およびユニット実装に取り入れられたのと同様に、動的解析手法（コードの一部やすべてを実行する） はソフトウェア単体テストとソフトウェア統合テストの両方に役立ちます。

Part 6の表7、8、10、および11では単体テストと統合テストを実行するための技法とメトリックがリストされており、その主な機能は、期待される機能とソフトウェアインターフェイスがユニットレベルと統合レベルで検証されるようにすることです。ソフトウェアの単体テストと統合テストはターゲットハードウェアで実行する必要があり、開発されたユニットまたは統合ソフトウェアが「安全性関連」の場合、テスト結果は安全性要求に準拠する必要があります。フォールトインジェクションとリソーステストは、堅牢性とレジリエンス（速やかに回復できるしなやかな強靭さ）をさらに高めることに役立ちます。モデルベース開発を適用する組織では、モデルレベルとコードレベルでのback-to-backテストが推奨されます。

図11、12は、それぞれPart 6の表7、8を改変したもので、元の表にコンプライアンスが自動的に確認できる部分を重ね合わせています。

手法		ASIL			
		A	B	C	D
1a	ウォークスルー	++	+	o	o
1b	ペアプログラミング	+	+	+	+
1c	精査・インスペクション	+	++	++	++
1d	準形式的検証	+	+	++	++
1e	形式的検証	o	o	+	+
1f	制御フロー解析	+	+	++	++
1g	データフロー解析	+	+	++	++
1h	静的コード解析	++	++	++	++
1i	抽象解釈に基づく静的コード解析	+	+	+	+
1j	要件ベーステスト	++	++	++	++
1k	インターフェイステスト	++	++	++	++
1l	故障注入テスト	+	+	+	++
1m	資源利用の評価	+	+	+	++
1n	適用可の場合、モデルとコード間のBack-to-back テスト	+	+	++	++
「++」… ASILに対してその手法が強く推奨される 「＋」 … ASILに対してその手法が推奨される 「o」… ASILに対してその手法は推奨されないか、使用を反対される LDRA tool suiteによってサポートされています LDRA tool suiteによって検証されます					

図11：LDRA tool suiteの機能とISO 26262-6:2018[26]に示される「表7：ソフトウェアユニットの設計と実装の検証方法」の対応

手法		ASIL			
		A	B	C	D
1a	要件の解析	++	+	o	o
1b	同値類の生成と解析	+	+	+	+
1c	境界値の解析	+	++	++	++
1d	知識や経験に基づくエラー推測	+	+	++	++
「++」… ASILに対してその手法が強く推奨される 「＋」 … ASILに対してその手法が推奨される 「o」… ASILに対してその手法は推奨されないか、使用を反対される LDRA tool suiteによって検証されます					

図12：LDRA tool suiteの機能とISO 26262-6:2018[27]に示される「表8：ソフトウェア単体テストのテストケースを導出する方法」の対応

開発された各ソフトウェアユニットは、そのソフトウェアユニット設計仕様書を参照してテストす

[26] ISO 26262-6:2018 の表 7 より、 © The British Standards Institution 2018. All rights acknowledged

[27] ISO 26262-6:2018 の表 8 より、 © The British Standards Institution 2018. All rights acknowledged

る必要があります。次に、ソフトウェアユニットに望ましくない機能が含まれていないことを確認するために、テスト手順を作成し、レビュー、および実行する必要があります。その後、単体テストは、ターゲットハードウェアまたはシミュレーション環境で検証計画と検証仕様に基づいて実行できます。テスト手順が実行されると、実際の出力をキャプチャし、期待される結果と比較します。合格/不合格の結果が報告され、それに応じてソフトウェアの安全性要件が検証されます。

図13、14はそれぞれPart 6の表10、11を改変したもので、元の表にコンプライアンスが自動的に確認できる部分を重ね合わせています。

手法		ASIL			
		A	B	C	D
1a	要件ベーステスト	++	++	++	++
1b	インターフェイステスト	++	++	++	++
1c	故障注入テスト	+	+	++	++
1d	資源利用のテスト	++	++	++	++
1e	適用可の場合、モデルとコード間のBack-to-back テスト	+	+	++	++
1f	制御フローとデータフローの検証	+	+	++	++
1g	静的コード解析	+	++	++	++
1h	抽象解釈に基づく静的コード解析	++	++	++	++
「++」… ASILに対してその手法が強く推奨される 「＋」 … ASILに対してその手法が推奨される 「o」… ASILに対してその手法は推奨されないか、使用を反対される LDRA tool suiteによってサポートされています LDRA tool suiteによって検証されます					

*図13：LDRA tool suiteの機能とISO 26262-6:2018[28]に示される「表10：ソフトウェア統合*の検証*方法」*の対応

手法		ASIL			
		A	B	C	D
1a	要件の解析	++	+	o	o
1b	同値類の生成と解析	+	+	+	+
1c	境界値の解析	+	++	++	++
1d	知識や経験に基づくエラー推測	+	+	++	++
「++」… ASILに対してその手法が強く推奨される 「＋」 … ASILに対してその手法が推奨される 「o」… ASILに対してその手法は推奨されないか、使用を反対される LDRA tool suiteによって検証されています					

図14：LDRA tool suiteの機能とISO 26262-6:2018[29]に示される「表11：ソフトウェア単体テストの

[28] ISO 26262-6:2018 の表 10 より、 © The British Standards Institution 2018. All rights acknowledged

[29] ISO 26262-6:2018 の表 11 より、 © The British Standards Institution 2018. All rights acknowledged

テストケースを導出する*方法」の対応*

統合テストは、対象のユニットがソフトウェアアーキテクチャ設計にしたがって連携して動作するときに、関連する指定要件を満たすように設計されています。実際には、これらの統合テストは通常、安全性と非安全性に関連するソフトウェア機能の検証を伴います。

すべての動的テストで、ターゲット環境に密接に対応した環境を使用してハードウェアとソフトウェア間の依存関係をテストすることが望まれます。それは必ずしも実用的ではありませんので、別の方法として、シミュレートされた環境でテストを開発し、そこで実証済みとなった場合にターゲットで再実行するものがあります。Part 6では、次のようにシミュレーションテストとターゲットテストに関する具体的なガイダンスを提供しています。

「*ソフトウェア単体テストがターゲット環境で実行されない場合、ソースコードやオブジェクトコードでの違い、ならびにテスト環境とターゲット環境での違いを分析して、後続のテストフェーズにおいてターゲット環境で追加するテストを特定します。*」

動的テストの失敗の結果や要件の変更に応じてなど、変更が必要になった場合、影響を受けるすべてのユニットの単体テストと統合テストを再実行する必要があります（リグレッションテスト）。これらのリグレッションテストは自動化でき、開発の進行に合わせて体系的に再適用して、新しい機能が確立および証明された機能を損なわないようにします。

ISO 26262で要求されることはありませんが、これらのテストにツールの静的解析による情報を用いることで、特にプロジェクトが大規模な場合には、テストプロセスをはるかに効率的にすることができます。

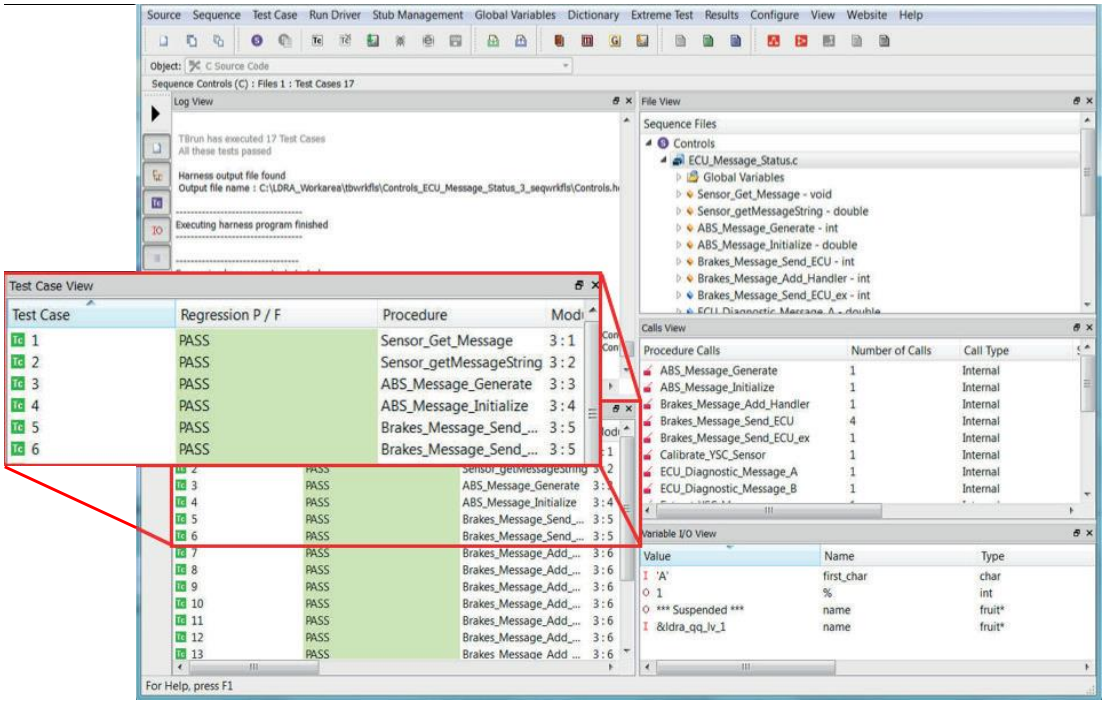


図15：LDRA tool suiteを使用した要件ベースの単体テストの実行

図15の例は、ユーザーが入力値と期待値を設定するソフトウェアインターフェイスが、関数のスコープで表示されています。そしてツールスイートはテストハーネスを生成し、それがターゲットハードウェア用にコンパイルされ、実行されます。実際の出力は、構造カバレッジデータとともに取得され、テストケースで指定された期待値と比較されます。

テストケースとして定義される入力値と期待値は通常、意図した機能が検証されることを保証するよう要件（図11、1j）から引き出されます。ネガティブテスト、フォールトインジェクション（図11、1l）および堅牢性テストなど他の様々な形態のテストが同じメカニズムを使用します。

ユニットがそれ単体でなくコールツリーの一部としてテストされるので、単体テストは統合テストになります。どちらの場合も、まったく同じテストデータを使用してコードを検証できます。

「エクストリームテスト」機能（図12、1c）を使用して境界値の解析を自動化し、一連の単体テストケースを自動的に生成できます。同じ機能で、最小値、区画の下限未満の値、区画の下限値、区画の上限値、および区画の上限超の値などの同値類の境界値の定義も提供します。また自動スタブ管理やグローバル変数宣言、例外処理などの機能は、包括的な単体テスト機能の遂行を支援します。

構造カバレッジのメトリック

ソフトウェアが正しく機能することを示すことに加え、動的解析を使用して構造カバレッジメトリックを生成します。ソフトウェアユニットレベルでの要件のカバレッジと並行して、これらのメトリックは「*要件ベースのテストケースの欠点*や、*要件の不十分さ*、*デッドコード*、*非アクティブコード*、*意図しない機能などを明らかにする*」ために必要なデータを提供します。

ステートメント、ブランチ、およびMC/DCカバレッジは、単体テストとシステムアクティビティの両方で生成できます。図16はPart 6の表9を改変したもので、元の表に自動テストツールによってコンプライアンスを支援できる部分を重ね合わせています。

トピック		ASIL			
		A	B	C	D
1a	ステートメントカバレッジ	++	++	+	+
1b	ブランチカバレッジ	+	++	++	++
1c	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++
「++」… ASILに対してその手法が強く推奨される 「＋」 … ASILに対してその手法が推奨される 「o」… ASILに対してその手法は推奨されないか、使用を反対される LDRA tool suiteによって検証されます					

図16：LDRA tool suiteの機能とISO 26262-6:2018[30]に示される「表9：ソフトウェアユニットレベル

[30] ISO 26262-6:2011 の表 9 による、© The British Standards Institution 2018. All rights acknowledged

での構造カバレッジメトリック」の対応

これらのパッケージは連携して動作できるので、例えば、動的システムテストを通じてソースコードの大部分に対してカバレッジを生成し、防御的な構造など通常のシステム動作中にはアクセスできない構成要素を実行させるために単体テストを使用して補完することができます。（図17）。

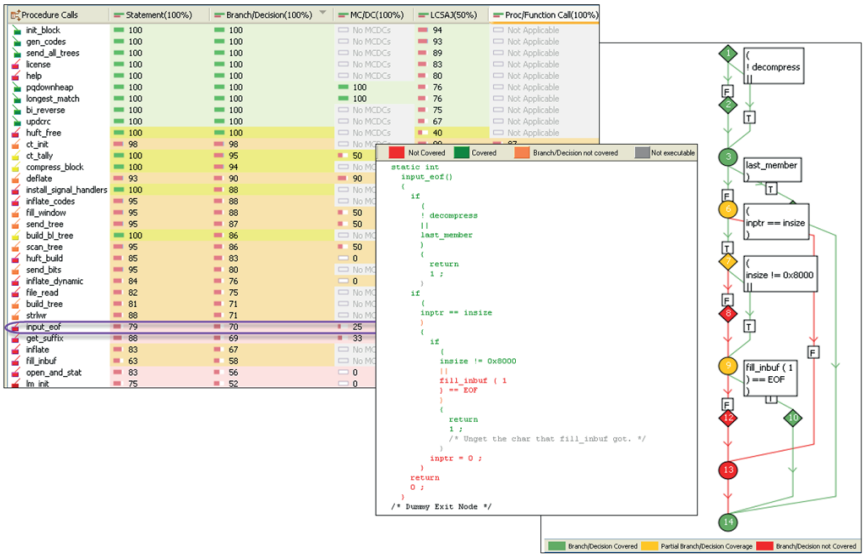


図17：LDRA tool suiteでの構造カバレッジの表示例

関数カバレッジデータとコールカバレッジデータの生成も、ターゲットでテストを実行することで自動化できます。図18は、統合テストに関連するPart 6の表12を変更したもので、元の表にコンプライアンスが自動的に確認できる部分を重ね合わせています。

カバレッジデータは、LDRA tool suiteの制御結合解析結果にも反映されます（図10で前述）。問題の可能性が静的解析中に強調表示され、実際のリンクが予測されたものと異なる場合には、動的解析中に警告メッセージが発生されます。その後、不具合を解決するためグラフィカルな表現を細かく調べます。

トピック		ASIL			
		A	B	C	D
1a	関数カバレッジ	+	+	++	++
1b	コールカバレッジ	+	+	++	++
「++」… ASILに対してその手法が強く推奨される 「＋」 … ASILに対してその手法が推奨される 「o」… ASILに対してその手法は推奨されないか、使用を反対される LDRA tool suiteによって検証されます					

図18：LDRA tool suiteの機能とISO 26262-6:2018[31]に示される「表12：ソフトウェアアーキテクチャレベルでの構造カバレッジメトリック」の対応

[31] ISO 26262-6:2011 の表 12 による、© The British Standards Institution 2018. All rights acknowledged

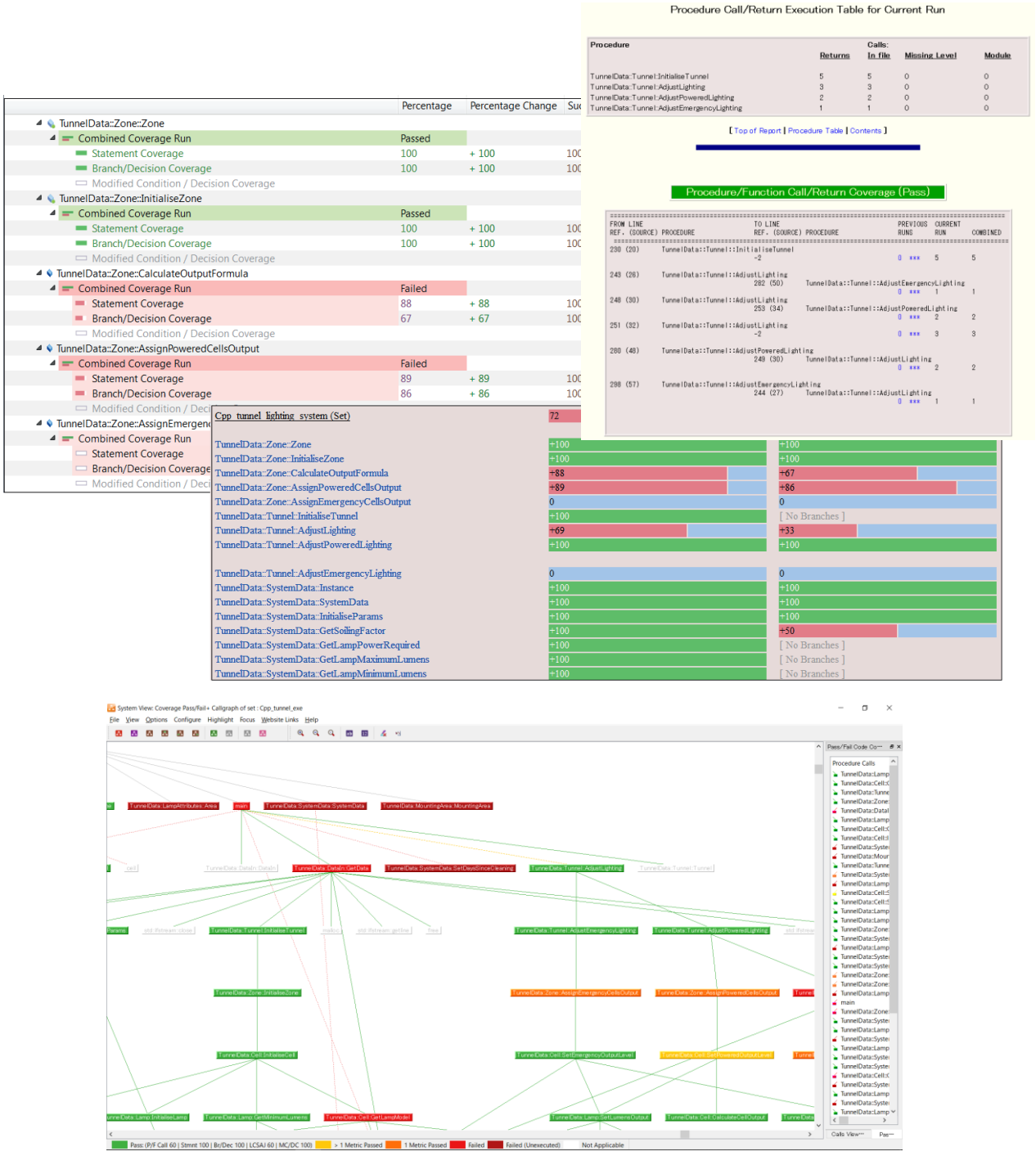


図19：LDRA tool suiteでの関数カバレッジとコールカバレッジの表現例

ソフトウェアテストとモデルベース開発

これら静的および動的機能は、IBMのRational Rhapsodyや、MathWorksのSimulink、ANSYSのScadeなどのモデルベース開発ツールと統合します。開発フェーズ自体は、通常の方法でモデルが作成されますが、そのモデルからソースコードが自動生成されることで統合がより適切になります。

モデルベース開発は、自動車ソフトウェアの開発者に多くの利点を提供し、多くのモデリングツールにはモデルと自動生成されたコードへのテスト機能が含まれます。しかし、ISO 26262には「*従来のライフサイクルデータが分離された開発プロセスと比較して、フェーズの強い合体 … が発生する可能性がある。このアプローチの潜在的な利点は … 魅力的だが、このアプローチには体系的な欠陥を引き起こす問題を導入する可能性もある。*」と示されています。モデリングツールと統合されながら、そこから独立したテストツールによる自動化は、これらの懸念を相殺するのに役立ちます。

図20は「Back-to-back」テストに適したアプローチを使用してSimulinkとの統合を展開する方法の一例を示しています（図11、1n および図13、1e）。設計モデルはSimulinkで開発され、Simulinkのテストで検証されます。次に、Simulinkから生成されるコードに対してLDRA tool suiteによってインスツルメントが行われた後、Software in the Loop（SILかホスト）またはProcessor in the Loop（PIL かターゲット）モードで実行されます。そして構造カバレッジが収集され、Simulinkとソースコードの動的解析が連携して構造カバレッジレポートが生成されます。

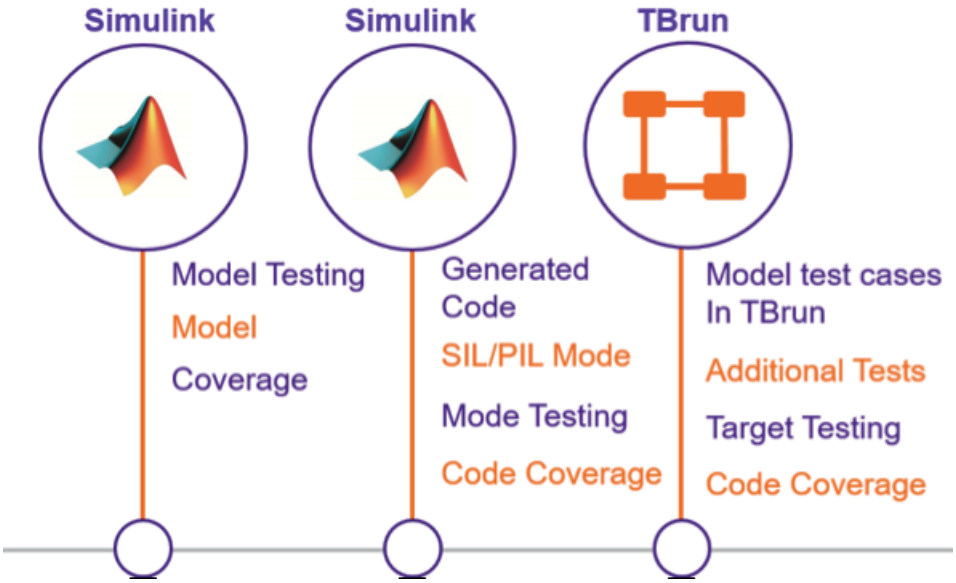


図20：MathWorksのSimulinkとLDRA tool suiteを使用して自動生成されたコードの構造カバレッジデータの生成

生成されたソースコードを静的に解析して、MISRA C:2012 付録E[32]などの適切なコーディング標準に準拠していることを確認できます。追加の動的テストはLDRA tool suiteからソースレベルで実行できます。要件ベースのテストを作成して、機能を検証し、構造カバレッジを照合できます。またテストデータをSimulinkからインポートして、LDRA tool suiteで実行することもできます。

自動生成されるコードがベースとなる場合でも、リアルタイム組込みシステムは従来のように人手で実装されたコードがある程度含まれています。ボードサポートパッケージ、割り込みハンドラー、ドライバ、およびその他の低レベルコード用のソフトウェアは、通常、手動でコーディングされます。ほとんどの場合、レガシーコードもシステムの一部に利用されます。システムのこれらの部分

[32] https://www.misra.org.uk/tabid/72/Default.aspx

は、自動生成されるコードとともに、LDRA tool suiteを従来どおりの方法で使用して検証できます。

双方向トレーサビリティ（Part 4 および 6）

双方向トレーサビリティはISO 26262全体で参照されており、Part 6 の段落7.4.2で次のように明示的に述べられています。

「*これは、ソフトウェアのアーキテクチャ設計と安全性要求との間の双方向のトレーサビリティを意味する。*」

しかし、より一般的には、その存在は、それより前の開発フェーズそれぞれを正確に反映する必要があり、例えばPart 6の段落9.1では以下のように、望ましくない特性が追加されずにすべての要件が満たされることを要求しています。

「*c）実装されたソフトウェアユニットがユニット設計に準拠し、割り当てられたソフトウェア要件を必要となるASILで満たしていることを証明する。そして、*
d）機能安全に関する望ましくない機能や望ましくない特性もソフトウェアユニットに含まれていないという十分な証拠を提供する。」

理論的には双方向トレーサビリティを達成するのは簡単です。Vモデルの厳格な順序が遵守されている場合、要件は決して変更されず、テストは問題を見過ごすことはありません。しかし、残念ながら大きな課題はプロジェクトの過程で発生します。

統合テストが失敗した場合に何が起こるか考えてみます。

- 要件に矛盾があるため失敗する可能性があり、その場合には要件を変更する必要があります。しかし、それによってソフトウェアの他の部分のどこが影響を受けますか?
- コーディングエラーが発生したため失敗する可能性があり、その場合には修正する必要があります。しかし変更されたコードに依存している、他のソフトウェアユニットはどれでしょうか? 誤った出力に依存している場合はどうなりますか?
- テストパラメータの基になる要件が正しくないために失敗する可能性もあります。これはつまり、要件の変更を意味します。しかし、これらのパラメータが間違っていたために上手くいかなかった単体テストはありましたか?

これらのシナリオは、ソフトウェア開発の製品間のトレーサビリティが急速に低下する要因です。保守を手作業で行うことも可能ですが、このような状況では自動化支援が大いに役立つでしょう。

ソフトウェアユニットの設計は様々な形式を取ることができ、自然言語やモデルベースのアプローチを活用できます。いずれの手法でも、これらの設計要素はソフトウェアの安全性要求とソフトウェアアーキテクチャの両方に双方向にトレースできる必要があります。その後、ソフトウェアユニットを仕様どおりに実装し、設計仕様にトレースできるようにする必要があります（図21）。

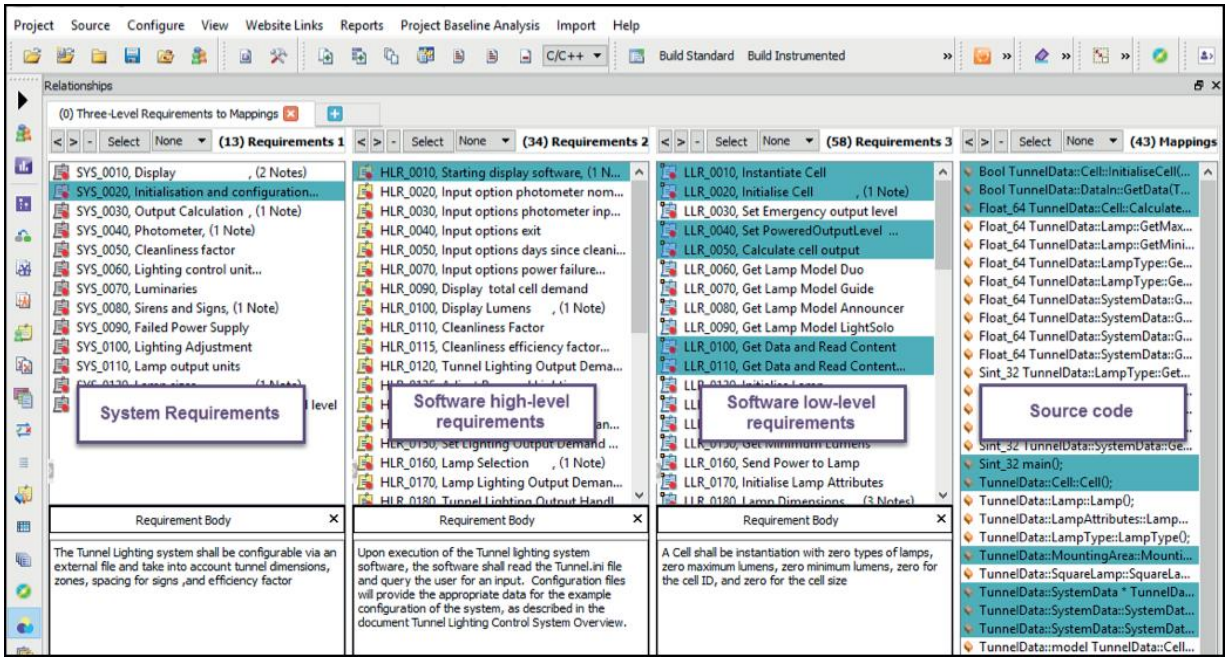


図21：LDRA tool suiteでのトレーサビリティ。上流のソフトウェア安全要求から下流のソフトウェアユニットにまでリンクされることが特徴。

図22は、各レベルの要件と相対するテストケース間でトレーサビリティを確立する方法を示しています。テストケースは要件に基づいて開発され、レビューされ、要件に対するテストカバレッジのギャップが評価されます。

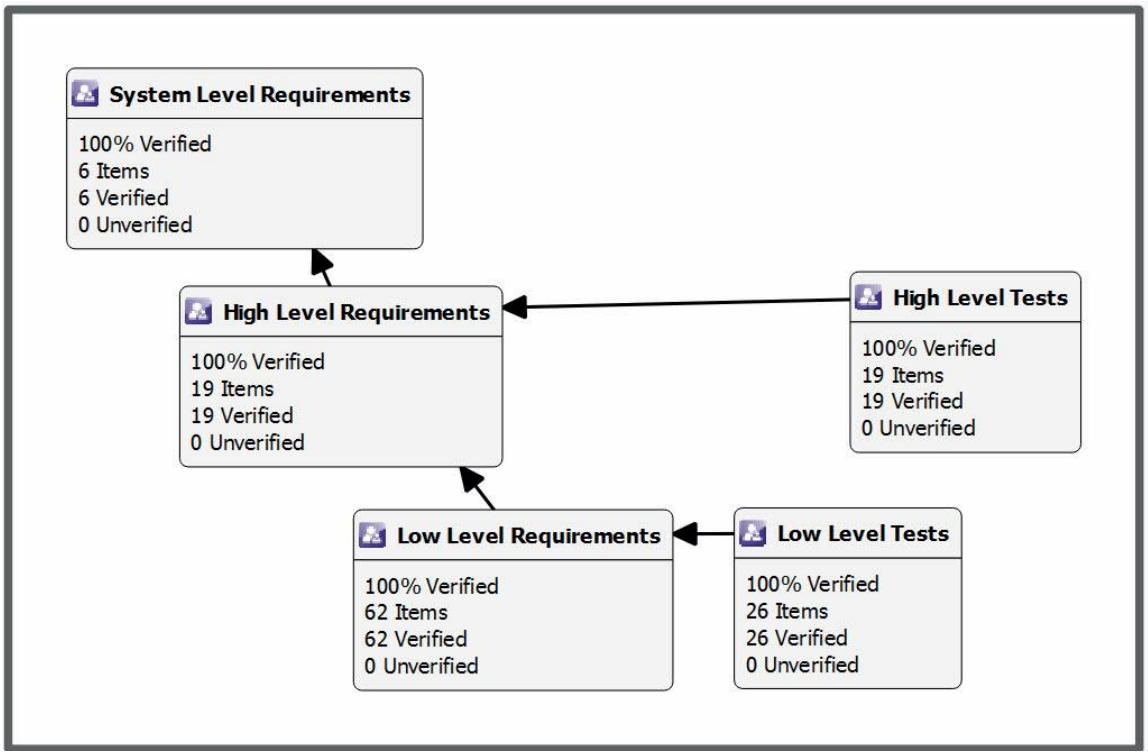
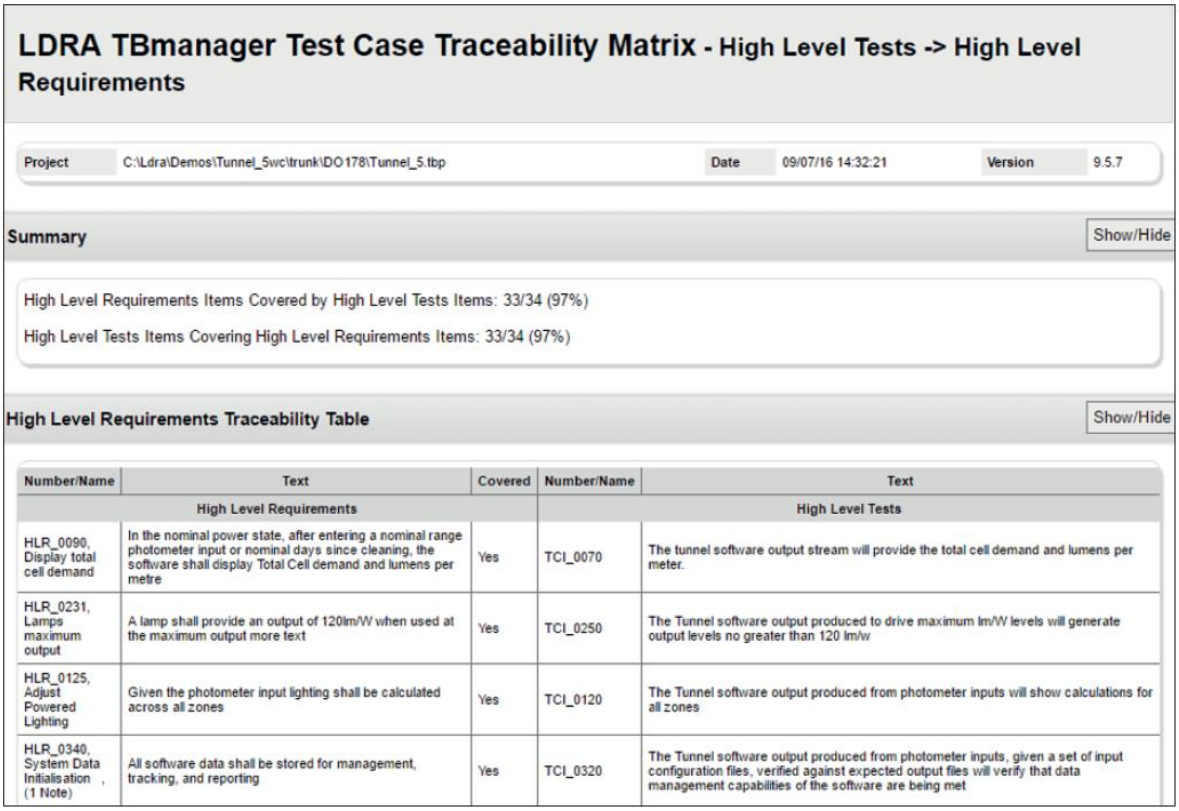


図22：要件ベースのテストの実行。LDRA tool suite内でテストケースが要件にリンクされ、直接実行できることが特徴。

また双方向解析なら、要件にリンクされていないテストケースを判定し修正を求めることもできます。さらに要件やテスト、コードへの変更の影響を分析するメカニズムを提供し、どこをリグレッションテストする必要があるか特定できるので、作業の負担を軽減します。また、トレーサビリティマトリックスなどの成果物を生成してコンプライアンスの証拠を提示するためのメカニズムも提供します。



LDRA TBmanager Test Case Traceability Matrix - High Level Tests -> High Level Requirements

Project	C:\Ldra\Demos\Tunnel_5wc\trunk\DO178\Tunnel_5.tbp	Date	09/07/16 14:32:21	Version	9.5.7

Summary Show/Hide

High Level Requirements Items Covered by High Level Tests Items: 33/34 (97%)

High Level Tests Items Covering High Level Requirements Items: 33/34 (97%)

High Level Requirements Traceability Table Show/Hide

Number/Name	Text	Covered	Number/Name	Text
High Level Requirements			High Level Tests	
HLR_0090, Display total cell demand	In the nominal power state, after entering a nominal range photometer input or nominal days since cleaning, the software shall display Total Cell demand and lumens per metre	Yes	TCI_0070	The tunnel software output stream will provide the total cell demand and lumens per meter.
HLR_0231, Lamps maximum output	A lamp shall provide an output of 120lm/W when used at the maximum output more text	Yes	TCI_0250	The Tunnel software output produced to drive maximum lm/W levels will generate output levels no greater than 120 lm/w
HLR_0125, Adjust Powered Lighting	Given the photometer input lighting shall be calculated across all zones	Yes	TCI_0120	The Tunnel software output produced from photometer inputs will show calculations for all zones
HLR_0340, System Data Initialisation , (1 Note)	All software data shall be stored for management, tracking, and reporting	Yes	TCI_0320	The Tunnel software output produced from photometer inputs, given a set of input configuration files, verified against expected output files will verify that data management capabilities of the software are being met

図23：LDRA tool suiteのトレーサビリティマトリックス（テストケースに対する高レベル要件）

図23は、高レベル要件と機能テストケース間のトレーサビリティマトリックスを示します。この例では、テストケースに関連付けられていない要件が1つあるため、高レベル要件のテストカバレッジの目的は満たされていません。同様の透過的なユーザーエクスペリエンスは、テストカバレッジ解析のすべての面で使用できます。これに単体テスト機能が組み合わされることで、開発ライフサイクル全体、特にソフトウェア単体テストと統合テスト中にトレーサビリティを確保するアクセス可能なメカニズムを提供します。

通常、実践的な構造カバレッジ解析は、機能テストの実行を解析のためにインスツルメントされたコードで行うことから始めて、さらなる解析を必要とするコードの未実行部分を明らかにします。最終的に、テストケースの追加や修正、要件の変更、および/またはデッドコードの削除が行われます。そして「レビュー、修正、解析」の繰り返し作業によって、設計仕様が保証されます。

ソフトウェアツールの使用に対する信頼（Part 8）

このISO 26262のサポートプロセスは、ソフトウェアツールチェーンが信頼できるという証拠を提供するメカニズムを定義しています。ソフトウェアツールに必要な信頼度は、各ツールの用途に応じて異なり、以下の項目を考慮します。

- ソフトウェアツールの誤動作とそれによる誤出力により、開発中の安全関連項目または要素にエラーを発生させる、あるいはエラーの検出ができない可能性がある（TI：Tool Impact）
- 誤動作による誤出力を防止あるいは検出する手段の信頼性（TD：Tool error Detection）

LDRA tool suiteは、ASIL DまでのISO 26262で使用できるツール認定を有しており、信頼性を証明するために必要となる大きな負担を排除することができます（図24）。

CERTIFICATE
No. Z10 084753 0006 Rev. 00

Holder of Certificate: LDRA Ltd.
Portside, Monks Ferry
Wirral, Merseyside CH41 5LH
UNITED KINGDOM

Factory(ies): 084753

Certification Mark:

Product: Software Tool for Safety Related Development

Model(s): LDRA tool suite
LDRArules
LDRAcover
LDRAunit
LDRAlite

Parameters: The certified tools fulfills the requirements for support tools classified T2 according to IEC 61508-3 and EN 50128. The tools are qualified to be used in safety-related software development according to IEC 61508, EN 50128 and ISO 26262. It is suitably validated for use in safety-related development according to IEC 62304. The test report is a mandatory part of this certificate.

Tested according to: EN 50128:2011
IEC 61508-3:2010
IEC 62304:2006
IEC 62304:2006/AMD1:2015
ISO 26262-8:2018

The product was tested on a voluntary basis and complies with the essential requirements. The certification mark shown above can be affixed on the product. It is not permitted to alter the certification mark in any way. In addition the certification holder must not transfer the certificate to third parties. See also notes overleaf.

Test report no.: LW85043C

Valid until: 2024-08-15

Date, 2019-08-16 (Peter Weiß)

Page 1 of 1
TÜV SÜD Product Service GmbH • Certification Body • Ridlerstraße 65 • 80339 Munich • Germany

ZERTIFIKAT ◆ CERTIFICATE ◆ 認證證書 ◆ СЕРТИФИКАТ ◆ CERTIFICADO ◆ CERTIFICAT

SGS TÜV SAAR

CERTIFICATE NO FS/71/220/19/0403 PAGE 1/1

LICENCE HOLDER
LDRA LTD
Portside, Monks Ferry,
Wirral, CH41 5LH,
United Kingdom

MANUFACTURING PLANT
LDRA LTD
Portside, Monks Ferry,
Wirral, CH41 5LH,
United Kingdom

PROJECT NO/-ID: M15F-AU04

LICENSED TEST MARK: SGS TÜV SAAR FUNCTIONALE SICHERHEIT GEPRÜFT / FUNCTIONAL SAFETY APPROVED

CERT. REPORT NO.: M15F0005

Tested according to: IEC 61508:2010; clause 7.4.4
ISO 26262:2018; clause 11.4.9
EN 50128:2011
IEC 60880:2006
IEC 62304:2015

Certified product(s): LDRA tool suite
LDRAunit
LDRArules
LDRAcover
LDRAlite

Model(s): Version 9.7.4
Version 9.7.5
Version 9.7.6
Version 9.7.7

Technical Data and Parameter: Suitable for development of safety-related software acc. to:
- ISO 26262 up to ASIL D,
- IEC 61508 up to SIL 4, class T2 tool
- EN 50128 up to SIL 4, class T2 tool
- IEC 62304 up to SW safety Class C
- IEC 60880 for category A functions

Specific Requirements: Any changes to the design, components or processing may require repetition of some parts of the pre-qualification to retain the certification. The certificate report is an integral part of this certificate.

Certification Body for Functional Safety SGS-TÜV Saar GmbH
Munich, 05.07.2019
Gudrun Neumann

SGS-TÜV Saar GmbH, Hohmannstr. 50, 81379 München, Germany
Website www.sgs-tuv-saar.com E-mail fs@sgs.com

CERTIFICATE

図24：LDRA tool suiteのTÜVによるツール認定

検証ツールは、ソースコード内のエラーの検出に失敗する可能性はありますが、自動コードジェネレータとは異なり、アプリケーションにエラーを導入することはないので、TI2（Tool Impact – 2）カテゴリに分類されます。

TI2分類では、アプリケーションの開発にツールスイートを使用する組織は、TD1からTD3の範囲のカテゴリ(Tool Error Detection)で、ツールの誤動作による誤出力を防止あるいは検出する手段の信頼性が規定されます。また、その評価の背後にある理由の証拠が必要です。

		Tool Error Detection		
		TD1	TD2	TD3
Tool Impact	TI1	TCL1	TCL1	TCL1
	TI2	TCL1	TCL2	TCL3

図25：LDRA tool suiteの特性とISO 26262-8:2018[33]の「表3：ツールの信頼度の決定」の対応

次に図25の表から、TCL（Tool Confidence Level）が得られます。ユーザーのアプリケーションに応じて得られるTCLは、LDRA tool suiteに対してTCL1またはTCL2どちらかになります。

ツールスイートにTCL2が割り当てられて製品がASIL Dと指定されている場合を除き、すべての場合においてTÜV証明書（図24）がツールの信頼性を確立するのに十分です。

手法		ASIL			
		A	B	C	D
1a	11.4.7に従った使用による信頼性の増加	++	++	++	+
1b	11.4.8に従ったツール開発プロセスの評価	++	++	++	+
1c	11.4.9に従ったソフトウェアツールの妥当性確認	+	+	+	++
1d	安全規格に従った開発	+	+	+	++
「++」… ASILに対してその手法が強く推奨される 「＋」 … ASILに対してその手法が推奨される TCL2指定を適用					

図26：ISO 26262-8:2018[34]の「表5：TCL2に分類されるソフトウェアツールの認定」からTCL2と指定された場合のLDRA tool suiteの認定の必要性の評価

それ以外の場合は、ツールの妥当性を確認する必要があります（図26）。ソフトウェアのサンプルを用いて、ターゲット環境でツールが正しく解析できることを示します。LDRAのツール認定サポートパッケージ（TQSP）は、そのサンプルソフトウェアを提供しています。

結論

自動車ソフトウェアの量と複雑性の爆発的増加について、既に多くのドキュメントで報告されています。ISO 26262は最新の第2版で、第1版によって確立された確固とした基盤を洗練し、他のタイプの車両を含めるだけでなく自動車ソフトウェア開発に対するサイバーセキュリティの影響が増大することも認めるよう拡張しています。

システムの各要素に適用されるASILを最小限にすることは、その実現に伴うコストが削減できるので明らかなインセンティブです。しかし、様々な自動車システムにASILを割り当てることの全体と

[33] ISO 26262-8:2018 の表 3 による、© 2018 IEC, Geneva, Switzerland. All rights acknowledged

[34] ISO 26262-8:2018 の表 5 による、© The British Standards Institution 2018. All rights acknowledged

しての原則は、車両で最も重要なシステムが他の重要ではない機能によって危険にさらされることがないようにという分離の仮定を意味します。したがって、コネクテッドカーが安全で、かつISO 26262の原則に準拠していると見なされるためには、攻撃対象を最小限に抑えることとシステム間の分離を最適化することが不可欠です。

それを保証することで、ASIL分解の原理は、機能安全に関わる製品の開発に伴う負担を最小限にします。LDRAが提供するツールは、航空機や防衛システムなどのセーフティクリティカルな分野で十分に実証されており、コンプライアンスへのルートをさらに最適化するだけでなく、機能安全規格自体よりも確立されています。

引用文書

“MISRA C:2012 - Guidelines for use of the C language in critical systems” ISBN 978-906400-11-8 (PDF), March 2013

“RTCA DO-178C Software Considerations in Airborne Systems and Equipment Certification”, Prepared by SC-205, December 13, 2011

“IEC 61508:1998 & 2000, Functional safety of electrical/electronic/programmable electronic safety-related systems” IEC, Geneva, Switzerland

“ISO 26262:2011 Road vehicles – Functional safety”. The British Standards Institution

“ISO 26262-4:2018 Road vehicles – Functional safety – Part 4: Product development at the system level”. The British Standards Institution

ISO 26262-5:2018 Road vehicles – Functional safety – Part 5: Product development at the hardware level”. The British Standards Institution

“ISO 26262-6:2018 Road vehicles – Functional safety – Part 6: Product development at the software level”. The British Standards Institution

“ISO 26262-8:2018 Road vehicles – Functional safety – Part 8: Supporting Processes”. The British Standards Institution 2018

“ISO 26262-11:2018 Road vehicles – Functional safety – Part 11: Guidelines on application of ISO 26262 to semiconductors.” The British Standards Institution

“ISO 26262-12:2018 Road vehicles – Functional safety – Part 12: Adaptation of ISO 26262 for motorcycles.” The British Standards Institution

“ISO/SAE CD 21434 Road Vehicles – Cybersecurity engineering”, ISO/SAE, under development
https://www.iso.org/standard/70918.html

“Experimental Security Analysis of a Modern Automobile”, Karl Koscher, Stephen Checkoway et.al. From 2010 IEEE Symposium on Security and Privacy
http://www.autosec.org/

“Remote Exploitation of an Unaltered Passenger Vehicle”, Dr. Charlie Miller & Chris Valasek, August 2015
http://illmatics.com/Remote%20Car%20Hacking.pdf

“SAE J3061-2016 Cybersecurity Guidebook For Cyber-Physical Vehicle Systems”, Society of Automotive Engineers, 2016
https://webstore.ansi.org/standards/sae/sae30612016j3061