ソフトウェア品質説明力の強化に伴い発生する

開発工程負荷の評価・分析

実施報告書

2013 年 2 月

独立行政法人情報処理推進機構
Information-technology Promotion Agency, Japan

はじめに

IPA/SEC では、ソフトウェア品質説明力を強化すべく様々な観点からの検討を実施してきました。その一環として、ソフトウェア品質を説明するための手法等について具体的な実施方法、そのための作業量、実施にあたっての課題等を整理し、実際にソフトウェア品質を説明する際の参考とできるようにするために、公募により、観点ごとに分けられた実験を別々に実施しました。本書は、それらの結果を、実験ごとにまとめた報告書のうちの１つです。

本報告書の実験は、「2011 年度 システムエンジニアリング実践拠点事業」として、株式会社ベリサーブに委託し実施しました。

「ソフトウェア品質説明力の強化に伴い発生する開発工程負荷の評価・分析」
【報告書】

独立行政法人情報処理推進機構

目次

1. 背景及び目的

日本の自動車関連メーカーの開発現場の多くでは、製品の最終品質（絶対品質）が重要視され、いわゆる擦り合せ開発[1]が行われてきた。そのため、設計文書（設計企画書や概要設計書など）や、最終成果物であるソフトウェアに至った経緯（変更履歴及びその理由など）について、詳細に文書化し管理できていない企業も存在する。

しかしながら、ソフトウェアの品質を第三者に説明する必要が生じた際、その１つの手段となる国際安全規格への対応しようとすれば、システム構成や機能などを決定した理由などの開発経緯を文書化し、その関係性を含めて明らかにする必要がある。

本実験では、国際安全規格の１つである ISO26262 を取り上げ、自動車メーカーにおける従来の開発手順に ISO26262 を導入して、品質説明力を強化した場合の開発プロセスへの負荷を測定・分析する。また、他の対応作業として、検証作業のフロントローディング（前工程化）や、設計文書のトレーサビリティ確保が求められることも想定される。本実験では、モデルベース開発による検証作業のフロントローディングと設計文書のトレーサビリティ管理を導入することで、新たに発生する作業内容と必要工数を調査し、導入効果について分析を行った。

[1] 擦り合せ開発：本書では、開発目標に対して、開発関係者が議論と調整を重ねることによって、技術や部品の仕様を徐々に決定しながら開発する手法のことをいう

2. 実験の概要

2.1 実験の範囲

本実験は、ソフトウェア品質説明力の強化の手段として、国際規格への準拠に注目し、自動車の機能安全に関する国際規格である ISO26262 に準拠させるために必要な情報を収集する。

本実験で使用するモデルシステムは、自動車用ドアロックシステム（詳細「3.実験対象システムの概要」参照）とし、開発ライフサイクルの「概要設計～システム設計」（ライフサイクルマネジメントプロセスを除く）において、ISO26262 に準拠した開発プロセスにより、仮想的にソフトウェア開発を行う。（Fig2-1 参照）

自動車用ドアロックシステムは、構造や制御の仕組みが比較的シンプルであり、自動車に対する専門知識が無くても理解し易いシステムであると考えられる。一方で、故障時には安全性に大きな影響を与えるシステムであるため、モデルケースとして適切と考え選定した。

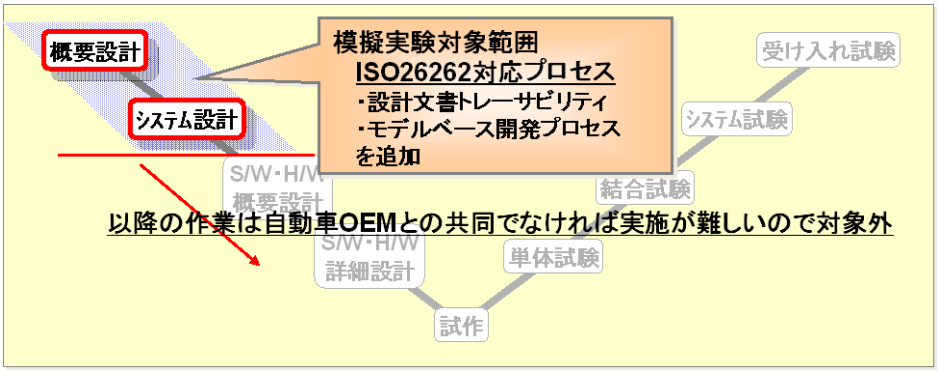


Fig2-1 実験の対象範囲

従来型開発プロセスにおける「概要設計～システム設計」は、ソフトウェア及びハードウェアの概要を検討する工程であり、詳細な検討経緯を含めて設計文書のトレーサビリティ確保が難しい領域である。したがって、ISO26262 に準拠する場合においても、既存の開発プロセスに新たに追加されるプロセスが多くなる領域と想定される。一方、これ以降の開発プロセスについては、実物の開発が含まれてくるため、自動車メーカーなどとの共同作業が必要となるため、本実験の対象範囲外とした。（Fig2-2 参照）

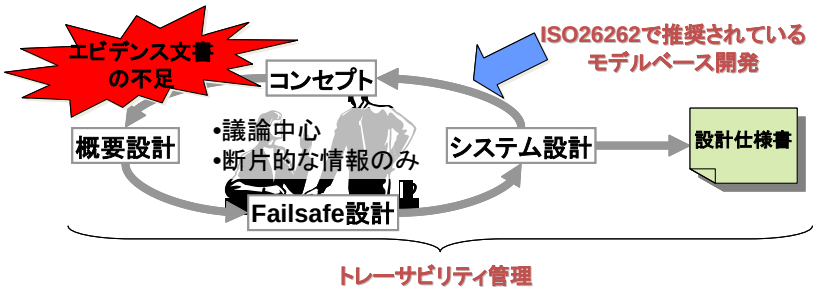


Fig2-2「概要設計～システム設計」の業務イメージ

設計文書であるシステム設計仕様書（RFQ：Request for Quotation）と FMEA：Failure Mode and Effects Analysis 及び FTA：Fault Tree Analysis が作成された経緯は、メモや議事録などに表されており、組織的に管理されていない場合や、文書化されていない場合が多い。そのため、製品開発後に開発経緯を十分に参照することが難しく、製品の品質説明力を確保できない一因となっている。

ISO26262 では、モデルベース開発やトレーサビリティ管理を導入することを推奨していることから、本実験では、モデルベース開発とトレーサビリティ管理を取り入れた新しい開発プロセスを定義し、開発を行った場合に新たに発生する作業工数を測定・分析する。

モデルベース開発とトレーサビリティ管理において、本実験で利用するツールを以下に記載する。

①モデルベース開発シミュレーションツール：OpenModelica

OpenModelica は、プラントモデル及び簡単なプラント制御モデルを記述できる言語である。世界的には、シミュレーションのプラントモデルを中心に、広く利用されるようになってきており、OpenModelica をベースとしたシステムシミュレーションソフトウェアも多数存在する。加えて、オープンソースのツールであり、だれでも利用することが可能で汎用性が高いため、OpenModelica を利用してシミュレーションを実施した。

②トレーサビリティ管理ツール：TERAS TRA

TERAS TRA は、経済産業省の組込みシステム基盤開発事業により、一般社団法人 TERAS が開発しているトレーサビリティ管理ツールである。本実験では、このツールを利用して、トレーサビリティ管理に関する実験を実施した。なお、以下、単に TERAS と記した場合、TERAS TRA を意味するものとする。

2.2 全体アプローチ

本実験は、モデルベース開発とトレーサビリティ管理が導入された新しい開発プロセスを定義して実験を行い、作業工数を測定・分析する。

モデルベース開発については、作業工数以外に導入の効果についても分析を実施する。モデルベース開発を導入した場合には、プラント制御モデルを構築してシミュレーションを行うため、作業工数は増加することが予測される。

一方で、上流工程で不具合を発見できる他、試作品の作製コスト削減や、自動テストが可能であるなど、工程全体では効率化できることが効果として挙げられている。本実験では、作業工数の測定と並行して、検証のフロントローディングによる不具合発見の早期化に関する効果を検証することとする。

モデルベース開発の効果を分析するために、従来型の開発プロセス（仕様書作成後、試作品を作成して、実機により検証する開発プロセス）では、後工程で発見されるような不具合

を予め設定しておき、モデルベースシミュレーションにより発見できるかどうかの実験を行う。客観性を高めるために、仕様書作成は株式会社ベリサーブ（以下、「ベリサーブ」という）で実施し、モデルベースシミュレーションは別の組織のシミュレーション担当にて実施する。シミュレーション担当は組み込まれた不具合内容を知らずに、シミュレーション検証を実施する。シミュレーション検証の結果である不具合の発見状況と、モデルベース開発に要した作業工数を比較・分析して評価を実施する。

トレーサビリティ管理については、モデルベース開発のアウトプット及びその他の設計文書に付随する開発経緯を説明可能とする資料（議事録やメモなど）を想定して、これらの情報を関連付けしながら管理する作業に要した工数を測定する。

全体的なアプローチは Fig2-3 の通り。

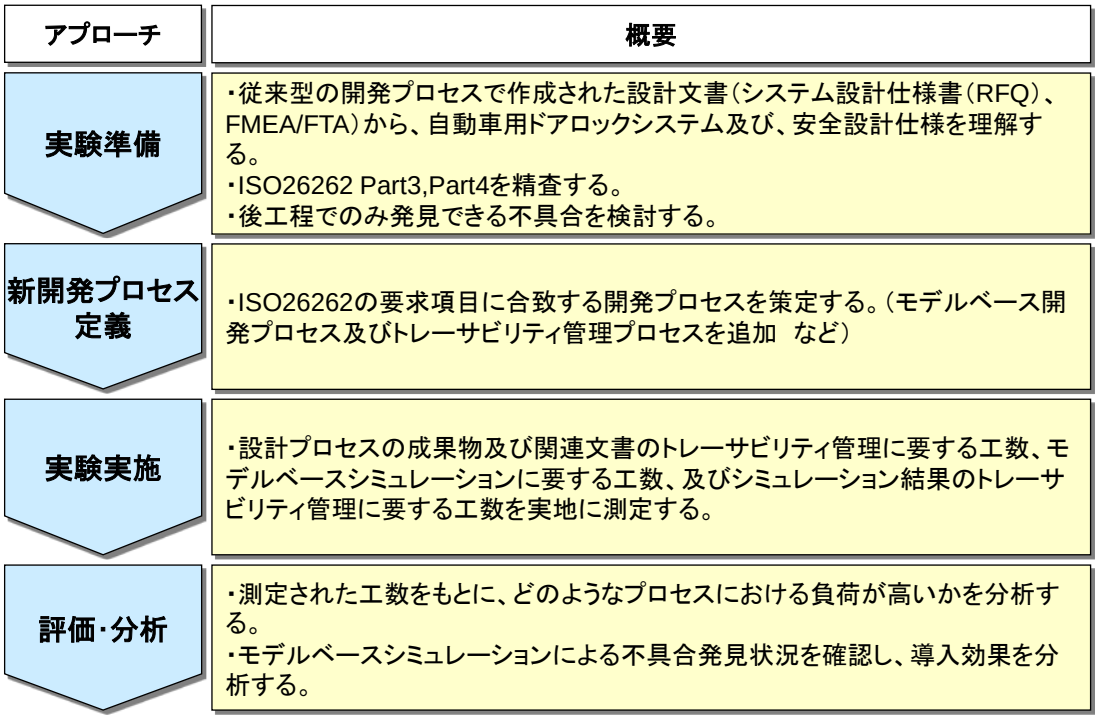


Fig2-3 実験の全体アプローチと主要タスク

2.3 評価方法

「概要設計～システム設計」の開発プロセスにおいて、トレーサビリティ管理に係る付加工数及び、モデルベース開発に係る工数・効果について、以下 3 点の評価指標について計測・評価を行う。

- モデルベース開発実行に要する工数
- モデルベース開発により発見された不具合数割合
- トレーサビリティ管理に要する工数の計測と算定

2.3.1　モデルベース開発実行に要する工数

評価指標：作業工数（単位：人月）

モデルベース開発では、仕様書を基に

・仕様理解
・モデル開発
・試験仕様書作成
・モデルベースシミュレーション実行
・シミュレーション結果分析

の 5 つの作業を行い、実地に工数を計測する。

2.3.2　モデルベース開発（設計段階でのシミュレーション）により発見された不具合数割合

評価指標：不具合発見率（単位：パーセント）

従来型の開発プロセスにおいて、一般的には後工程で発見されるような不具合を予め仕様書に設定する。モデルベースシミュレーションにより、発見できた個数を測定し、不具合発見率を算定する。これにより、モデルベース開発導入の効果を測定する。

2.3.3　トレーサビリティ管理に要する工数の計測と算定

評価指標：作業工数（単位：人月）

新たな開発プロセスにより、トレーサビリティ管理作業が追加されるが、これに係る作業工数を測定する。TERAS を活用して、設計文書や関連資料のトレーサビリティの設定に係る作業工数とこれらを格納するのに要する作業工数を測定する。

3. 実験対象システムの概要

本実験では、自動車に関する高度な専門知識が無くても理解しやすいことと、不具合発生時に搭乗者の安全に関わる機能であることから、自動車用ドアロックシステム（以下、本システム）を対象として実験を実施した。

3.1 自動車用ドアロックシステムの主要機能

本システムには、市販車の自動車用ドアロックシステムに近い仕様とするために、基本機能（ドアロック・アンロック機能、車速感応オートロック機能）以外に、トランクリッドのアンロック機能、エアバッグ連動オートアンロック機能、及びチャイルドプルーフのアンロック機能を設定した。エアバッグ連動オートアンロック機能については、当該機能に不具合が発生した場合には、搭乗者の安全に関わる可能性があるため、ISO26262 の適用対象システムになるものと考えられる。

また、当初はフュエルリッドのアンロック機能を設定する予定だったが、当該機能は燃料の盗難防止を目的とした機能であり、安全には影響を及ぼさない機能であることから、本システムからは除外した。

本システムの主要機能をまとめると Fig3-1 の通りである。

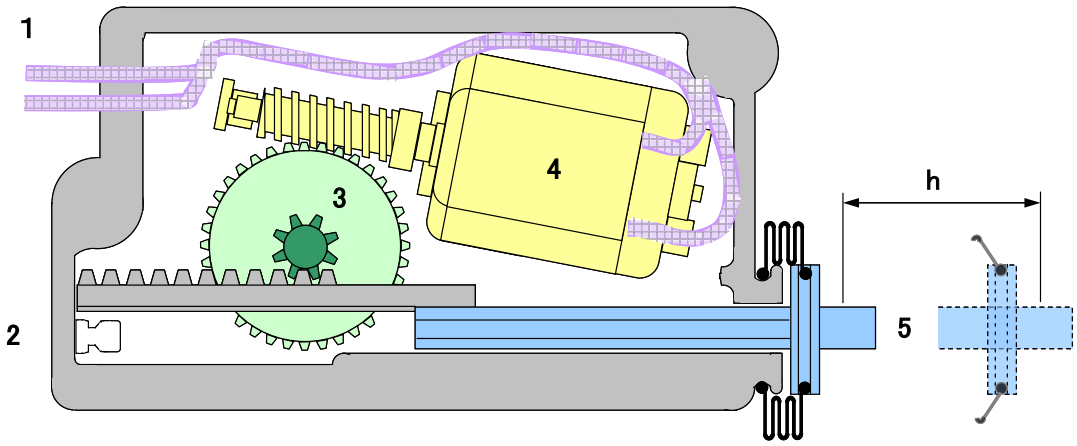
機能	仕様
ドアロック･アンロック機能	ドアロックボタン押下により全ドアのロック･アンロック作動
車速感応オートロック機能	車速約30km/h以上で自動ドアロック作動
トランクリッドアンロック機能	車速約10km/h未満でトランクリッドオープナーボタン押下によりトランクリッドアンロック作動
エアバッグ連動オートアンロック機能	エアバッグ作動時にオートアンロック作動
チャイルドプルーフアンロック機能	チャイルドプルーフボタン押下によりチャイルドプルーフアンロック作動
その他の要件	チャイルドプルーフと非干渉にドアロック･アンロックされること 「ドアロック･アンロック機能」不作動時にドアロックレバーにてアンロックできること 「ドアロック･アンロック機能」不作動時に外部よりキー操作にてアンロックできること

Fig3-1 自動車用ドアロックシステムの主要機能

3.2 アクチュエータ部

本システムのアクチュエータは、電動モーターの動力をモーター軸のウォームギアを介してラックアンドピニオンギアに伝達し、ラックギアに連結された調整レバーを稼働させるこ

とで、ドアロックの施錠・開錠を行う。(Fig3-2 参照)



1:電源ケーブル　2:フレキシブル終端カップリング　3:ギヤユニット
4:電動モーター　5:調整レバー　h:移動範囲

Fig3-2 自動車用ドアロックのアクチュエータ(イメージ)

3.3 制御部

本実験では、ECU 部分の制御仕様により、周辺デバイスが仕様通りに作動するかどうかについて、シミュレーションにより確認する。(Fig3-3 参照)

本システムにおける制御方法は、各センサーからのアナログ入力(車速センサーなど)、CAN 信号(エアバッグ作動信号など)、各スイッチ(ドアロック、チャイルドプルーフなど)からの ON/OFF 入力により、所定条件に従った ON/OFF 制御信号を発信することで、制御される。

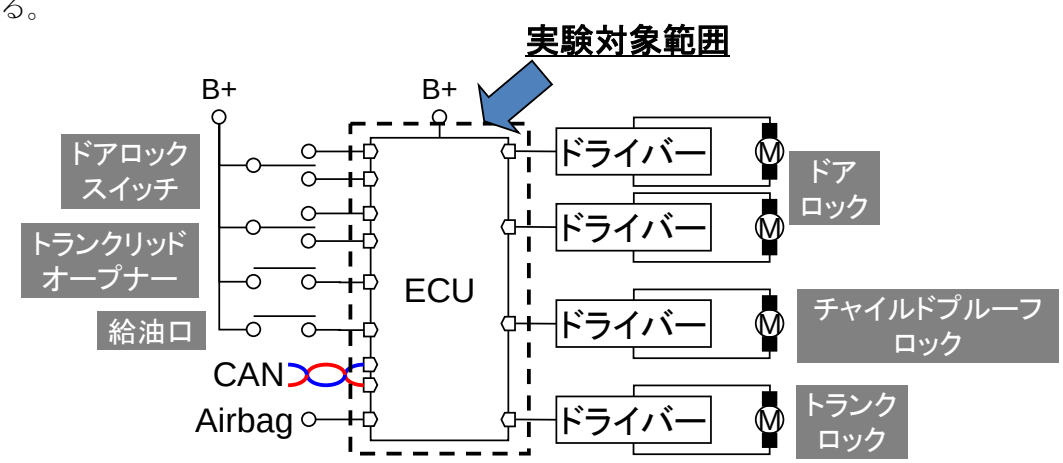


Fig3-3 自動車用ドアロックシステムにおける実験対象範囲

4. 実験方法

4.1 安全度水準の設定

本実験では、自動車用ドアロックシステムが ASIL[2] D に準拠するものとして、実験を行う。ASIL 設定の検討そのものは、本実験の対象外であるが、ASIL 設定検討経緯はトレーサビリティ管理する必要があるため、参考として ASIL 検討を実施した。

ASIL レベルの検討経緯の詳細については「4.2 ASIL 設定の検討（参考）」にて説明しているが、本実験では ASIL D に準拠するための開発手法を導入した場合の工数負荷、及び導入効果を測定することを目的としている。したがって、本システムが ASIL D よりも低いレベルであると決定された場合でも、本システムが ASIL D 相当であると仮定して本実験を実施する。

4.2 ASIL 設定の検討（参考）

ASIL を決定するために、まず本システムの運転状況を想定して、危険事象を発生すると想定されるハザード（潜在的に発生しうる危険な状態）を抽出する。抽出された各ハザードのうち、特に危険な状態（人命に関わる危険な状態）を選定し（Fig4-1 参照）、深刻さ（Severity）・発生頻度（Exposure）・回避可能性（Controllability）の 3 要素を考慮して ASIL を決定する。

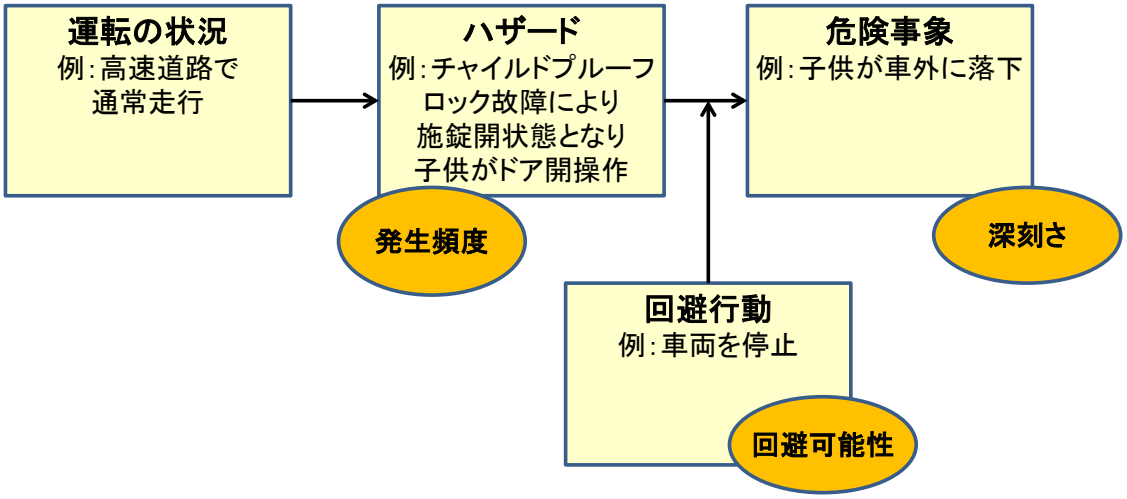


Fig4-1 ASIL 設定の考え方

4.2.1 深刻さ（Severity）の評価

本システムに深刻な危険事象が発生する場合を検討し、ISO26262 の「深刻さ（Severity）

[2] ASIL : Automotive Safety Integrity Level
リスクを許容水準に抑えることを達成するための安全性の要求 A(要求レベル低)~D(要求レベル高)までのレベルがある。

の分類」（Fig4-2 参照）より、その被害の大きさを評価する。

チャイルドプルーフロック故障が要因となって、高速道路走行時に子供が車外に落下した場合には、最悪致命傷に至るものと判断され、S3 と評価した。

Fig4-2 深刻さ(Severity)の分類

4.2.2 発生頻度（Exposure）の評価

本システムに深刻なハザードが発生する可能性を検討し、ISO26262 の「運用状況における発生頻度の分類」（Fig4-3 参照）より、その頻度を評価する。

本システムのチャイルドプルーフに不具合が発生して施錠開状態となる場合は、「ジュースなどの滴下によるスイッチ固着」や「タバコのヤニなどによるスイッチ固着」などが想定される。このようなケースの発生頻度を、ここでは E3 と評価した。

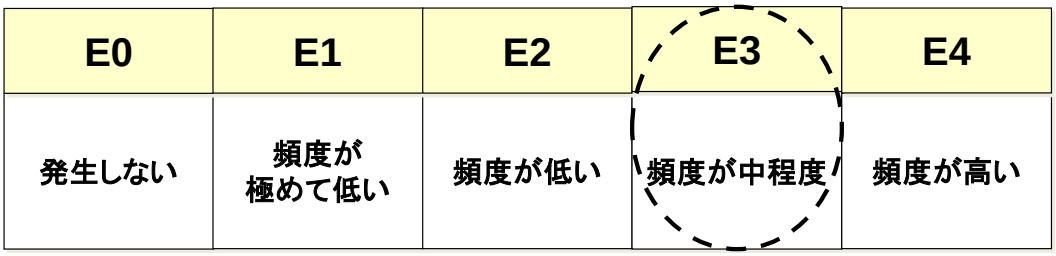


Fig4-3 発生頻度(Exposure)の分類

4.2.3 回避可能性（Controllability）の評価

本システムに深刻なハザードが発生した場合の回避できるかどうかを検討し、ISO26262 の「回避可能性の分類」（Fig4-4 参照）より、その回避可能性を評価する。

走行中に、チャイルドプルーフ不作動時にドアが開いた場合に、子供がドアを自分で閉めることは難しいが、「ドアが開いた状態で路肩に車両を停止」することで回避できると考えられ、ここでは C2 と評価した。

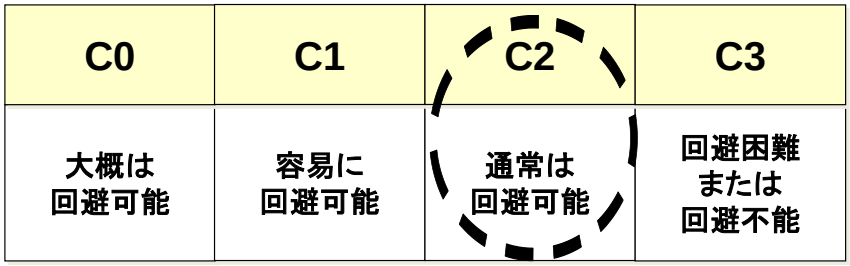


Fig4-4 回避可能性(Controllability)の分類

4.2.4 ASIL の決定

4.2.1～4.2.3 の結果から、ISO26262 の ASIL マトリクス（Fig4-5 参照）に当てはめ、本

システムの ASIL レベルは B と決定した。ただし、前述の通り、本実験では ASIL D に準拠するように本システムの開発を進めた。

		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	ASIL A
	E4	QM	ASIL A	ASIL B
S2	E1	QM	QM	QM
	E2	QM	QM	ASIL A
	E3	QM	ASIL A	ASIL B
	E4	ASIL A	ASIL B	ASIL C
S3	E1	QM	QM	ASIL A
	E2	QM	ASIL A	ASIL B
	E3	ASIL A	ASIL B	ASIL C
	E4	ASIL B	ASIL C	ASIL D

Fig4-5 実験システムの ASIL 決定結果（参考）

4.3 本実験にて対象とする ISO26262 セクション内容

本実験では、前述（詳細は「2.1 実験の範囲」参照）の通り、開発ライフサイクルの「概要設計～システム設計」を対象範囲として、ISO26262 に準拠するような開発プロセスを仮想的に進める。本実験の対象範囲に該当する ISO26262 のパート、及びセクションは Fig4-6 の通りである。

以下の項目では､各セクションにおいて ISO26262 が要求するタスクと成果物を説明する。本実験では、ISO26262 Part3、Part4 にて要求される成果物のトレーサビリティ管理及び、Part4 にて要求されるモデルベースシミュレーションを実施して、追加的に要した工数を測定し、分析・評価を実施する。

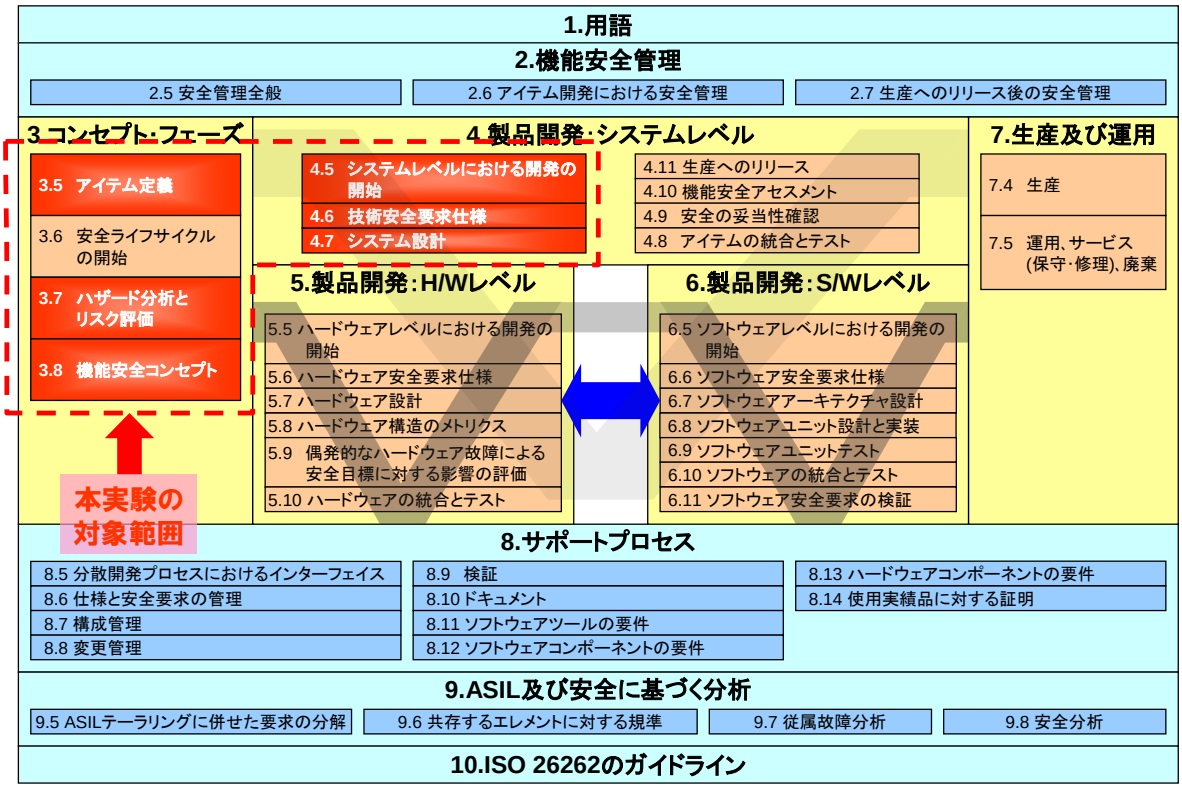


Fig4-6 ISO26262 の対象範囲

4.3.1 「3.5 アイテム定義」

「3.5 アイテム定義」では、関係者間で開発対象についての共通認識を形成するために必要な情報を定義する。要求されるタスクと成果物は以下の通りである。

［要求タスク］

- 開発対象システムと周辺環境の依存関係の明確化
- 開発対象システムとそのインターフェース部分、周辺システムの境界の明確化

［要求成果物］

- アイテム定義書

4.3.2 「3.7 ハザード分析とリスク評価」

「3.7 ハザード分析とリスク評価」では、開発対象システムの故障が原因のリスクを一般に許容される範囲内に収めるために、ASIL レベルとセーフティゴール（最上位レベルの安全要求）を設定する。要求されるタスクと成果物は以下の通りである。

［要求タスク］

- 状況分析とハザードの特定
- 危険事象のレベル分け
- ASIL とセーフティゴールの決定

- ハザード分析･リスク評価結果に対する検証
- セーフティゴールに対する検証

［要求成果物］
- ハザード分析とリスク評価結果
- セーフティゴール
- 検証結果報告書

4.3.3 「3.8 機能安全コンセプト」

「3.8 機能安全コンセプト」では、「3.7 ハザード分析とリスク評価」で設定したセーフティゴールを具体的に落とし込んだ機能安全コンセプトを策定する。機能安全コンセプトには、セーフティゴールから導出された機能安全要求、当該要求を開発対象システムの各要素に割り当てる際の考え方、及び最終的に開発されたシステムの妥当性を確認する際の基準が盛り込まれる。要求されるタスクと成果物は以下の通りである。

［要求タスク］
- 機能安全要求の導出
- システム要素への機能安全要求割り当て
- 妥当性確認基準の設定
- 機能安全コンセプトに対する検証

［要求成果物］
- 機能安全コンセプト
- 機能安全コンセプトに対する検証結果報告書

4.3.4 「4.5 システムレベルにおける開発の開始」

「4.5 システムレベルにおける開発の開始」では、システムレベル開発プロセスにおける機能安全を実現するためのアクティビティを計画する。要求されるタスクと成果物は以下の通りである。

［要求タスク］
- 安全性を考慮した設計、インテグレーションの計画
- 妥当性確認の計画
- 機能安全アセスメントの計画
- システムレベルにおける安全ライフサイクルの開発プロセスへのテーラリング

［要求成果物］
- プロジェクト計画書
- 安全計画書

- インテグレーション計画書
- テスト計画書
- 妥当性確認計画書
- 機能安全アセスメント計画書

4.3.5 「4.6 技術安全要求仕様」

「4.6 技術安全要求仕様」では、機能安全コンセプトを具体的に実現する技術要件を定義する。技術安全要求には、システムが故障した際の検出･表示･制御、及びシステムの安全状態維持などに関する技術要件を盛り込む必要がある。要求されるタスクと成果物は以下の通りである。

［要求タスク］

- 技術安全要求仕様の策定
- 開発システムにおける安全メカニズムの検討
- ASIL のディコンポジション
- 潜在的故障の回避方法検討
- 生産フェーズ以降における該当箇所の特定
- 技術安全要求に対する検証と妥当性確認

［要求成果物］

- 技術安全要求仕様書
- システム検証報告書
- 妥当性確認計画書

4.3.6 「4.7 システム設計」

「4.7 システム設計」では、システムに対する機能要求をハードウェア、及びソフトウェアで実現する方策を開発する。その際に、技術安全要求仕様をハードウェア、及びソフトウェアに割り当てる際の考え方と、これらを統合した際に技術安全要求が実装されていることを確認する方策などを検討する。要求されるタスクと成果物は以下の通りである。

［要求タスク］

- システム設計仕様の検討
- 技術安全コンセプトの検討
- システム構造設計に関する制約条件の考慮
- システム的な故障に対する回避措置の検討
- ハードウェアの偶発的な故障に対する制御措置の検討
- ハードウェアとソフトウェアへの技術安全要求の割り当て
- ハードウェア、ソフトウェアのインターフェース仕様の検討
- 生産フェーズ以降における要求の考慮

・ システム設計に対する検証

［要求成果物］

・ 技術安全コンセプト
・ システム設計仕様書
・ ハードウェア、ソフトウェアインターフェース仕様書
・ 生産フェーズ以降に対する要求仕様書
・ システム検証報告書
・ 安全分析報告書

4.4 本実験にて使用する設計文書

本実験では概要設計～システム設計において、下記の設計文書を作成し、トレーサビリティ管理を実施する対象とした。

・システム設計仕様書（RFQ）
・是正措置指示書（FMEA の結果）
・FMEA 議事録
・システム試験仕様書
・システム試験結果報告書

また本実験では、システム試験仕様書とシステム試験結果報告書については、従来型の開発プロセスではシステム試験の工程で作成されるものであるが、ISO26262 へ準拠するために、新たにシステム設計の工程で作成する必要が発生した設計文書として位置付けている。

したがって、これらの文書を作成するのに要した作業については、ISO26262 に準拠するために新たに発生したものと定義する。

以下では、本システムにおける設計文書の内容について説明する。

4.4.1 システム設計仕様書（RFQ）

システム概要設計、及びシステム詳細設計をまとめた仕様書であり、従来のシステム設計フェーズにおける主要成果物である。従来型の開発プロセスでは、システム設計仕様書（RFQ）の中で概要設計と詳細設計を明確には区分してせずに、徐々に機能の概要を詳細化するプロセスを経ることで作成される。しかしながら、本実験では両者を区別してトレーサビリティ管理を実施する必要があるため、ISO26262 の要求事項を参照しながら概要設計と詳細設計を区分した。

本実験で使用した、システム設計仕様書（RFQ）は以下のように構成される。

[概要設計]

・ユースケース
・ソフトウェア/ハードウェアインターフェース仕様

・抽出ハザード

[詳細設計]

・ハードウェアインターフェース仕様
・ソフトウェアインターフェース仕様
・ハードウェア仕様
・ソフトウェアの機能ブロック仕様

4.4.1.1 ユースケース

運転者及び同乗者のユースケースを作成することによって、本システムに必要な機能を抽出した。ここでは、本システムを運転者及び同乗者が操作するために、必要とする情報や、本システムの動作に関わる他システムからの情報についても記載した。本実験では、ユースケースをシステム概要設計に区分されるものとして扱った。

ユースケースを作成することによって、実現するべき機能をどのシステムで実現するかを決定した。作成したユースケースは Fig4-7 の通りである。

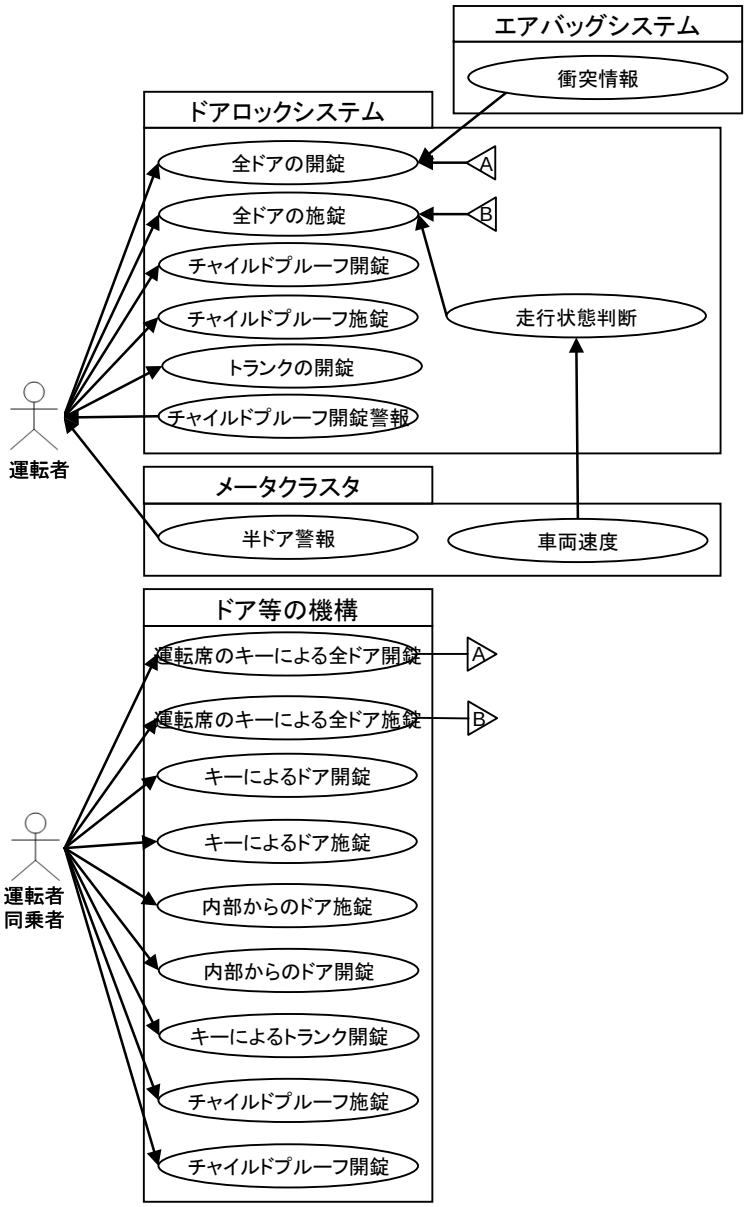


Fig4-7 ユースケース

4.4.1.2 ソフトウェア/ハードウェアインターフェース仕様

ソフトウェア及びハードウェアのインターフェースの全体概要を記載した仕様を作成し、システム概要設計の一部とした。

作成したソフトウェア/ハードウェアインターフェース仕様は Fig4-8 の通りである。

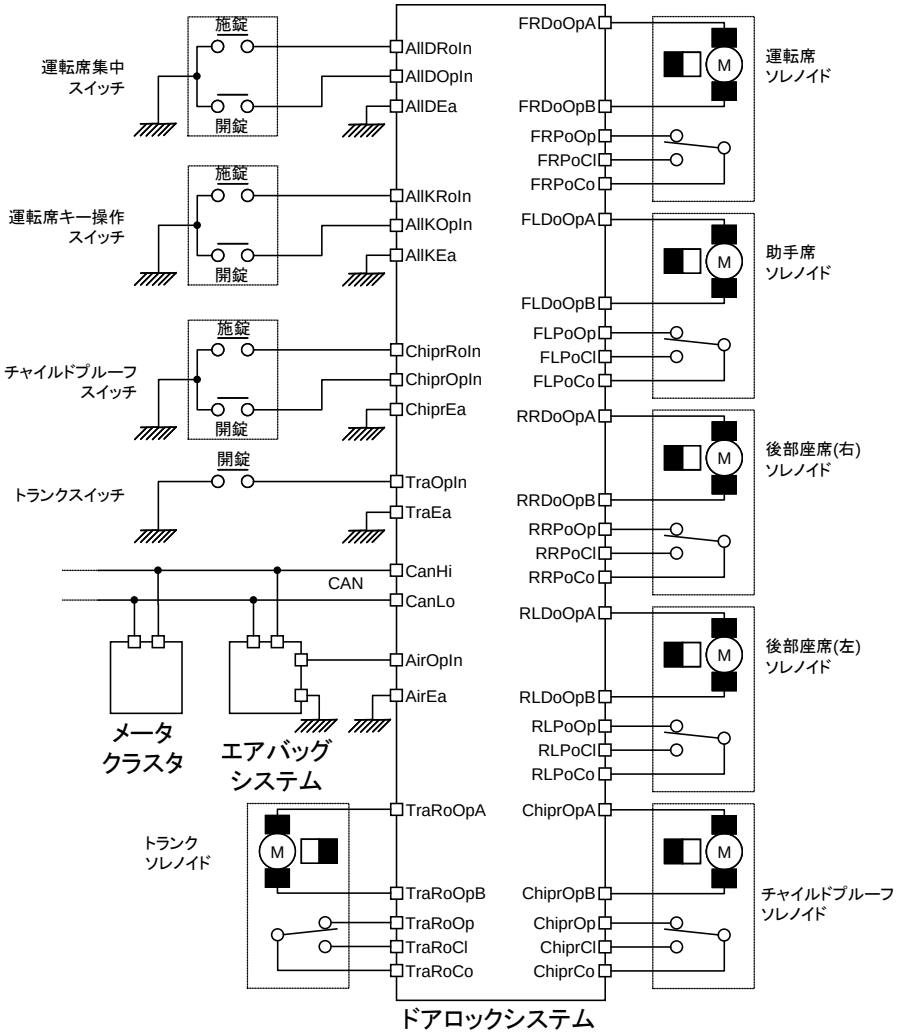
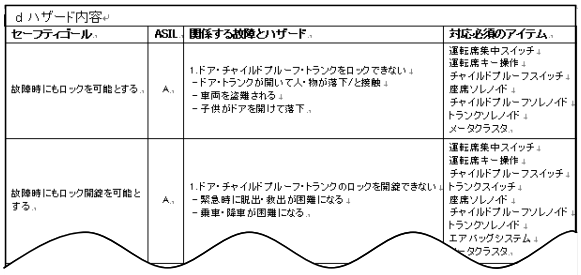


Fig4-8 ソフトウェア/ハードウェアインターフェース仕様

4.4.1.3 抽出ハザード

FMEA を実施した結果、抽出されたハザードを一覧としてシステム概要設計として記載した。抽出されたハザードへの対応については、詳細設計の中で是正措置を施している。

抽出ハザードのサンプルは Fig 4-9 に示す通りである。



d ハザード内容

セーフティゴール	ASIL	関係する故障とハザード	対応必須のアイテム
故障時にもロックを可能とする	A	1.ドア・チャイルドプルーフ・トランクをロックできない －ドア・トランクが開いて人・物が落下/と接触 －車両を盗難される －子供がドアを開けて落下	運転席集中スイッチ 運転席キー操作 チャイルドプルーフスイッチ 座席ソレノイド チャイルドプルーフソレノイド トランクソレノイド メータクラスタ
故障時にもロック開錠を可能とする	A	1.ドア・チャイルドプルーフ・トランクのロックを開錠できない －緊急時に脱出・救出が困難になる －乗車・降車が困難になる	運転席集中スイッチ 運転席キー操作 チャイルドプルーフスイッチ トランクスイッチ 座席ソレノイド チャイルドプルーフソレノイド トランクソレノイド エアバッグシステム メータクラスタ

Fig4-9 抽出ハザードのサンプル

4.4.1.4 ハードウェアインターフェース仕様

本システムを構成するハードウェアごとに、ソフトウェアとのインターフェース部分に関する仕様をハードウェアインターフェース仕様として作成した。作成したハードウェアインターフェース仕様のサンプルは Fig4-10 の通りである。

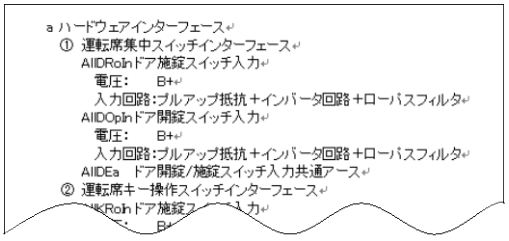


Fig4-10 ハードウェアインターフェース仕様サンプル

4.4.1.5 ソフトウェアインターフェース仕様

本システムを構成するソフトウェアの機能ブロックごとに、ハードウェアとのインターフェース部分に関する仕様をソフトウェアインターフェース仕様として作成した。作成したソフトウェアインターフェース仕様のサンプルは Fig4-11 の通りである。

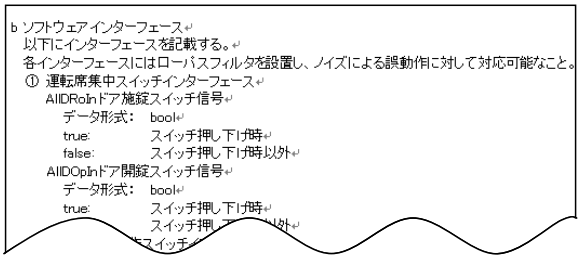


Fig4-11 ソフトウェアインターフェース仕様サンプル

4.4.1.6 ハードウェア仕様

本システムに使用するハードウェアの条件をハードウェア仕様として作成した。作成したハードウェア仕様のサンプルは Fig4-12 の通りである。

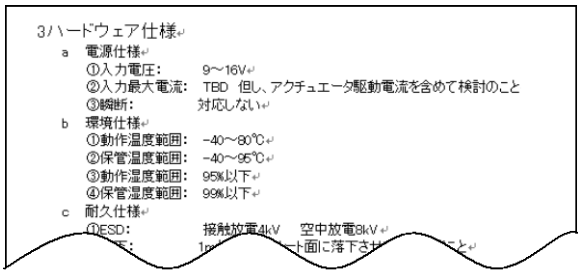


Fig4-12 ハードウェア仕様サンプル

4.4.1.7 ソフトウェアの機能ブロック仕様

ソフトウェアの機能ブロックを概要として作成し、機能ブロック毎に入出力の仕様を詳細化した。また、抽出ハザードに対するソフトウェア部分における是正措置については、この部分に盛り込んだ。

ソフトウェアの機能ブロック図は Fig4-13 の通りである。

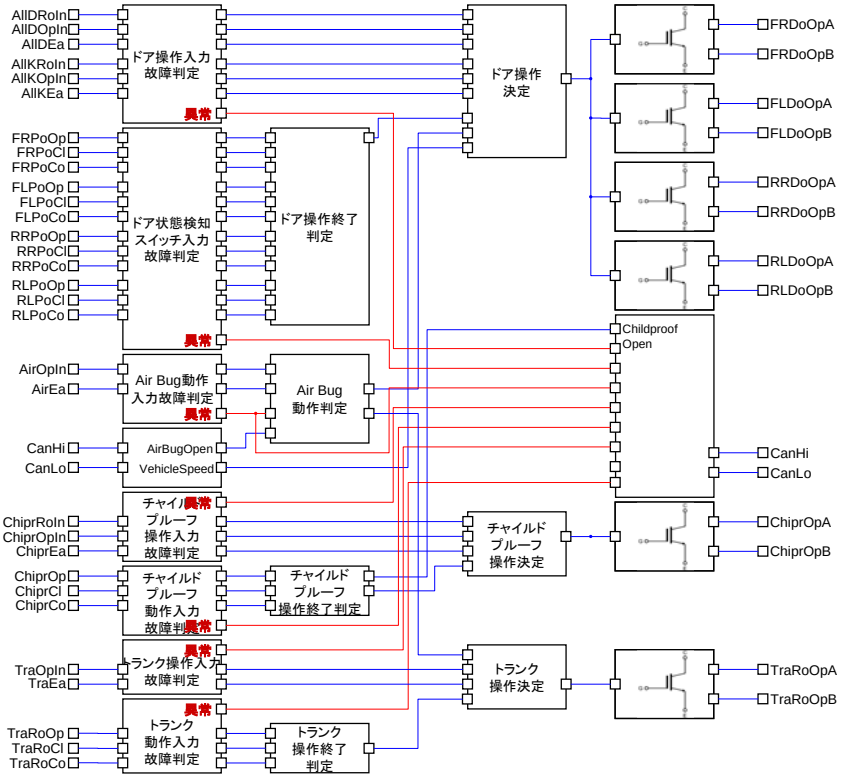


Fig4-13 ソフトウェア機能ブロック図

4.4.2 是正措置指示書（FMEA の結果）

システム設計仕様書（RFQ）の概要設計部分に記載されている内容に対して、本システムが安全面におけるハザードを FMEA の実施により抽出した。FMEA によって抽出されたハザードについては、安全面に問題が無い状態にする必要があるため、そのための方策を推奨是正措置として定めた。

FMEA の実施結果（是正措置指示書）のサンプルは Fig4-14 に示す通りである。

項目	故障モード	故障の影響	状況	故障事象	重大度	故障メカニズム	発生頻度	検出方法	非検出性	制御方法	非制御性	危険優先度	推奨是正措置
運転席集中スイッチ	施錠スイッチ常時 OFF	ドアをロックできない	停車中(乗降中)	ドアをロックできない	4	スイッチ故障	2	無し	10	機械的なロック	1	80	A-a. 機構側でのロックを可能とする
運転席集中スイッチ	施錠スイッチ常時 OFF	ドアをロックできない	停車中(ACC OFF)	ドアをロックできない	4	スイッチ故障	2	無し	10	機械的なロック	1	80	A-a. 機構側でのロックを可能とする
運転席集中スイッチ	施錠スイッチ常時 OFF	ドアをロックできない	停車中(ACC ON)	ドアをロックできない	4	スイッチ故障	2	無し	10	機械的なロック	1	80	A-a. 機構側でのロックを可能とする
運転席集中スイッチ	施錠スイッチ常時 OFF	ドアをロックできない	停車中(IG ON)	ドアをロックできない	4	スイッチ故障	2	無し	10	機械的なロック	1	80	A-a. 機構側でのロックを可能とする
運転席集中スイッチ	施錠スイッチ常時 OFF	ドアをロックできない	走行中	低速走行中にドアが開き、人が落下	9	スイッチ故障	2	無し	10	半ドア警報により運転手に通知	3	540	A-a. 機構側でのロックを可能とする
運転席集中スイッチ	施錠スイッチ常時 OFF	ドアをロックできない	走行中	低速走行中にドアが開き、モノが落下	8	スイッチ故障	2	無し	10	半ドア警報により運転手に通知	3	480	A-a. 機構側でのロックを可能とする
運転席集中			走行中	低速走行中 き		イッチ故	2	無し				540	A-a. 機構側で を可能

Fig4-14 是正措置指示書のサンプル

4.4.3 FMEA 議事録

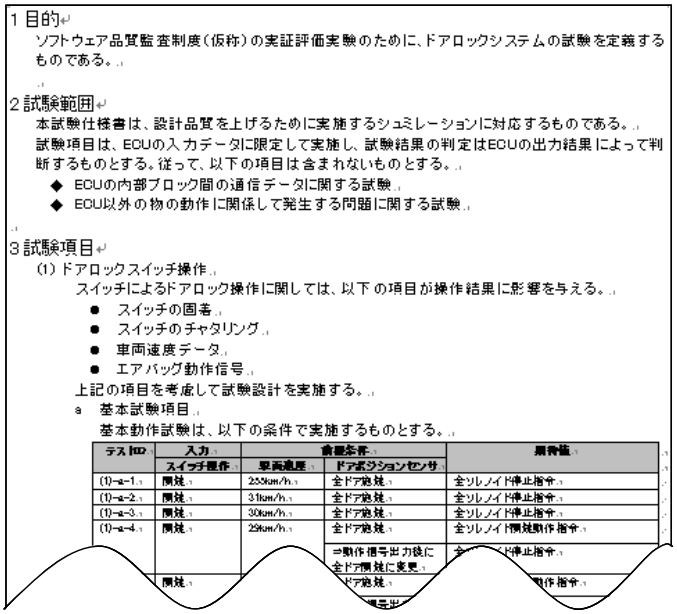
品質説明力を強化するために、FMEA の実施過程が分かるような議事録を作成し、トレーサビリティ管理の対象とした。

従来型の開発プロセスでは、FMEA 実施時の検討内容を議事録の形式で記録に残すことはあまり実施されていなかった。しかしながら、ISO26262 では FMEA の結果に対してレビューにより検証を行うことが要求されており、FMEA のレビュー結果を議事録として明示的に記録する必要がある。したがって本実験では、FMEA の議事録についても設計文書としてトレーサビリティ管理の対象とした。

4.4.4 システム試験仕様書

従来型の開発プロセスにおいて、システム試験仕様書はシステム試験工程までに作成するものだが、ISO26262 では検証のフロントローディングが要求されているため、システム設計工程において作成する必要性がある。したがって本実験では、システム試験仕様書をシステム設計工程で作成し、モデルベースシミュレーションを利用して検証した。

システム試験仕様書のサンプルは Fig4-15 に示す通りである。



1 目的

ソフトウェア品質監査制度（仮称）の実証評価実験のために、ドアロックシステムの試験を定義するものである。

2 試験範囲

本試験仕様書は、設計品質を上げるために実施するシュミレーションに対応するものである。
試験項目は、ECUの入力データに限定して実施し、試験結果の判定はECUの出力結果によって判断するものとする。従って、以下の項目は含まれないものとする。

- ◆ ECUの内部ブロック間の通信データに関する試験
- ◆ ECU以外の物の動作に関係して発生する問題に関する試験

3 試験項目

(1) ドアロックスイッチ操作

スイッチによるドアロック操作に関しては、以下の項目が操作結果に影響を与える。

- スイッチの固着
- スイッチのチャタリング
- 車両速度データ
- エアバッグ動作信号

上記の項目を考慮して試験設計を実施する。

a 基本試験項目

基本動作試験は、以下の条件で実施するものとする。

テストID	入力	前提条件		期待値
	スイッチ操作	車両速度	ドアポジションセンサ	
(1)-a-1	開錠	255km/h	全ドア施錠	全ソレノイド停止指令
(1)-a-2	開錠	31km/h	全ドア施錠	全ソレノイド停止指令
(1)-a-3	開錠	30km/h	全ドア施錠	全ソレノイド停止指令
(1)-a-4	開錠	29km/h	全ドア施錠	全ソレノイド開錠動作指令
			⇒動作信号出力後に全ドア開錠に変更	全ソレノイド停止指令
	開錠		ドア施錠	動作指令

Fig4-15 システム試験仕様書サンプル

4.4.5 システム試験結果報告書

システム試験仕様書に基づいて実施されたシミュレーションの試験結果を記載した設計文書である。従来型の開発プロセスでは、システム試験工程において作成されるが、システム試験仕様書と同様に、システム設計工程で試験を実施した。したがって、本実験ではシステム試験結果報告書をシステム設計工程における設計文書として作成した。

システム試験結果報告書のサンプルは Fig4-16 に示す通りである。

a 基本試験項目

(1)-a-	スイッチ操作	車両速度	ドアポジション センサ動作	期待値	結果	注記（×の場合）
1	開錠	255km/h	全ドア施錠	全ソレノイド停止指令	○	
2	開錠	31km/h	全ドア施錠	全ソレノイド停止指令	○	
3	開錠	30km/h	全ドア施錠	全ソレノイド停止指令	○	
4	開錠	29km/h	全ドア施錠	全ソレノイド開錠動作指令	○	
5			⇒動作信号出力後に全ドア開錠に変更	全ソレノイド停止指令	○	
6	開錠	0km/h	全ドア施錠	全ソレノイド開錠動作指令	○	
7			⇒動作信号出力後に全ドア開錠に変更	全ソレノイド停止指令	○	
8	施錠	255km/h	全ドア開錠	全ソレノイド施錠動作指令	○	速度による自動施錠
9			⇒動作信号出力後に全ドア施錠に変更	全ソレノイド停止指令	○	
10	施錠	31km/h	全ドア開錠	全ソレノイド施錠動作指令	○	速度による自動施錠
11			⇒動作信号出力後に全ドア施錠に変更	全ソレノイド停止指令	○	
12	施錠	30km/h	全ドア開錠	全ソレノイド施錠動作指令	○	速度による自動施錠
13			⇒動作信号出力後に全ドア施錠に変更	全ソレノイド停止指令	○	
14	施錠	29km/h	全ドア開錠	全ソレノイド施錠動作指令	○	
15			⇒動作信号出力後に全ドア施錠に変更	全ソレノイド停止指令	○	
16	施錠	0km/h	全ドア開錠	全ソレノイド施錠動作指令	○	
17			⇒動作信号出力後に全ドア施錠に変更	全ソレノイド停止指令	○	
18	施錠	0km/h	全ドア施錠	全ソレノイド停止指令	○	
19	開錠	0km/h	全ドア開錠	全ソレノイド停止指令	○	
20	開錠	0km/h	FRドア開錠 それ以外施錠	FR以外のドアのソレノイド開錠動作指令	○	
21	開錠	0km/h	FLドア開錠 それ以外施錠	FL以外のドアのソレノイド開錠動作指令	○	
22	開錠	0km/h	RRドア開錠 それ以外施錠	RR以外のドアのソレノイド開錠動作指令	○	
23	開錠	0km/h	RLドア開錠 それ以外施錠	RL以外のドアのソレノイド開錠動作指令	○	
24	施錠	0km/h	FRドア施錠 それ以外開錠	FR以外のドアのソレノイド施錠動作指令	○	
		0km/h	FLドア施錠 それ以外	FL以外のドアのソレノ		
			RRド			

Fig4-16 システム試験結果報告書サンプル

4.5 本実験で対象とする開発プロセス（想定）の定義

従来型の開発プロセスでは、システム設計工程で作成されたシステム設計仕様書（RFQ）に対する検証は、システム試験工程における試験を中心に実施していた。したがって、システム設計仕様書（RFQ）に不具合が含まれていた場合、システム試験工程まで不具合が発見されない可能性が高い。

このような不具合は通常出荷前までには修正されるが、修正内容が設計文書に反映されないことが多い。修正内容を反映する場合には、後から不具合の修正経緯を辿り、関連する設計文書を全て修正しなければならず、多大な工数が必要となる。

本実験では、従来型の開発プロセスを一定のレベルで維持しつつも、品質説明力を確保できるように、ISO26262 で要求されている、シミュレーション検証を実施できるモデルベース開発とトレーサビリティ管理の手法を採用した。

モデルベース開発を導入することで、検証のフロントローディングが可能になり、システム設計工程で設計文書の不具合を発見することが可能になる。これにより、後工程で不具合が発見された場合に比べて、不具合修正対応時の工数が削減できる効果があるとされている。

また、トレーサビリティ管理を導入することによって、設計文書における考慮漏れを検出できるようになることで、設計文書の網羅性を確保することが可能になる。さらに、設計文書を変更する際の影響範囲が容易に分かるようになるため、変更管理工数を削減できる効果が見込まれる。

以下では、本実験で実施した、モデルベース開発とトレーサビリティ管理を取り入れた開発プロセスの詳細を説明する。本実験で実施した開発プロセスを模式的に示すと Fig4-17 の通りである。

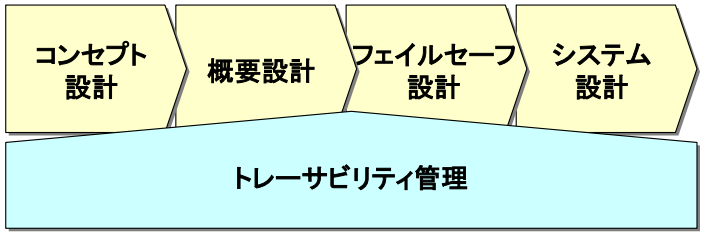


Fig4-17 本実験で実施した開発プロセス

4.5.1 コンセプト設計

顧客要望や市場調査の結果などから、開発するシステムの種別と必要な機能の概要を決定する。本実験では、開発するシステムを自動車用ドアロックシステムと決定し、必要となる主要な機能を定義した。コンセプト設計における成果物としては、ユースケースが該当する。

4.5.2　概要設計

コンセプト設計で定義された主要機能を本システムで実現するために、システムに対する入出力や機能の概要についての仕様を決定した。また、本システムと関連する周辺のシステムとのインターフェースについても、概要設計で決定した。

ユースケースとソフトウェア/ハードウェアインターフェース仕様が、概要設計における成果物に該当する。本実験では、コンセプト設計～概要設計が ISO26262 における「3.5 アイテム定義」に合致するものとした。また、概要設計の工程で決定した内容については、システム設計仕様書（RFQ）の概要設計部分に記載した。

4.5.3　フェイルセーフ設計

概要設計で決定された事項に対して FMEA を実施した。本システムを構成する要素に故障が発生した場合の不具合を推定し、当該不具合への対策を実現する機能を推奨是正措置として設計した。

本実験では、推奨是正措置を ISO26262 で要求されるセーフティゴールに見立てることによって、フェイルセーフ設計の工程が ISO26262 の「3.7 ハザード分析とリスク評価」に該当すると考えた。また、フェイルセーフ設計で抽出された推奨是正措置は、システム設計仕様書（RFQ）の概要設計部分に追記した。

4.5.4　システム設計

システム設計では、概要設計で決定された機能をソフトウェア及びハードウェアで実現するために、機能ブロックに分解し、詳細化する工程である。ソフトウェアとハードウェアの機能ブロック単位ごとに、機能ブロックの入出力を示すインターフェース、及び機能ブロック毎の挙動を規定する。フェイルセーフ設計で定義された推奨是正措置については、システム設計工程で仕様として具体化する。これらの内容をシステム設計仕様書（RFQ）の詳細設計部分に追記した。

また、システム設計工程では、システム設計仕様書（RFQ）に記載された内容を OpenModelica で記述し、シミュレーションによってシステム設計仕様書（RFQ）の検証を実施した。シミュレーション検証を実施するための設計文書であるシステム試験仕様書、及びシステム試験結果報告書についてもシステム設計フェーズにおける成果物として扱った。

また、本実験ではシステム設計工程で実施される作業によって、ISO26262 の「4.6 技術安全要求仕様」、及び「4.7 システム設計」における要求事項に準拠できるものと考える。

4.5.5　トレーサビリティ管理

本実験では、概要設計～システム設計までの作業を実施するに当たって、設計文書間のトレーサビリティ管理を実施した。本実験では、トレーサビリティ管理を導入することによって、一定以上の工数負荷がかかることが予想されたため、設計文書を作成する開発者とは別に、専任のトレーサビリティ管理者が作業を実施した。

また、一般的な開発プロセスにおいては、膨大な量の設計文書が作成されるため、本実験ではトレーサビリティ管理を行うためのツールとして、TERAS を利用した。TERAS を選定した理由については、前述の通りである。(「2.1 実験の範囲」参照)

トレーサビリティ管理を導入することによって発生した作業の詳細については、「4.7 本実験におけるトレーサビリティ管理」にて説明する。

4.6 本実験で使用するシミュレーションモデル

4.6.1 モデル記述言語

本実験では、OpenModelica をモデルベース開発に使用するシミュレーション言語として選定した。選定理由は以下の通りである。

- OpenModelica が OSS（Open Source Software）のため、導入コストが低いこと
- 本システムがシーケンス制御のみで構成されるため、OpenModelica のみでの記述が可能であること

本実験では、本システムにおける制御部を試験対象としたため、プラント制御モデル（アクチュエータを制御する ECU 側のモデル）のみを OpenModelica で記述した。本実験で作成されたプラント制御モデルは約 50 個のモデルから構成される。

本実験において作成された、本システムの制御部の全体像を示すモデルは、Fig4-18 の通りである。

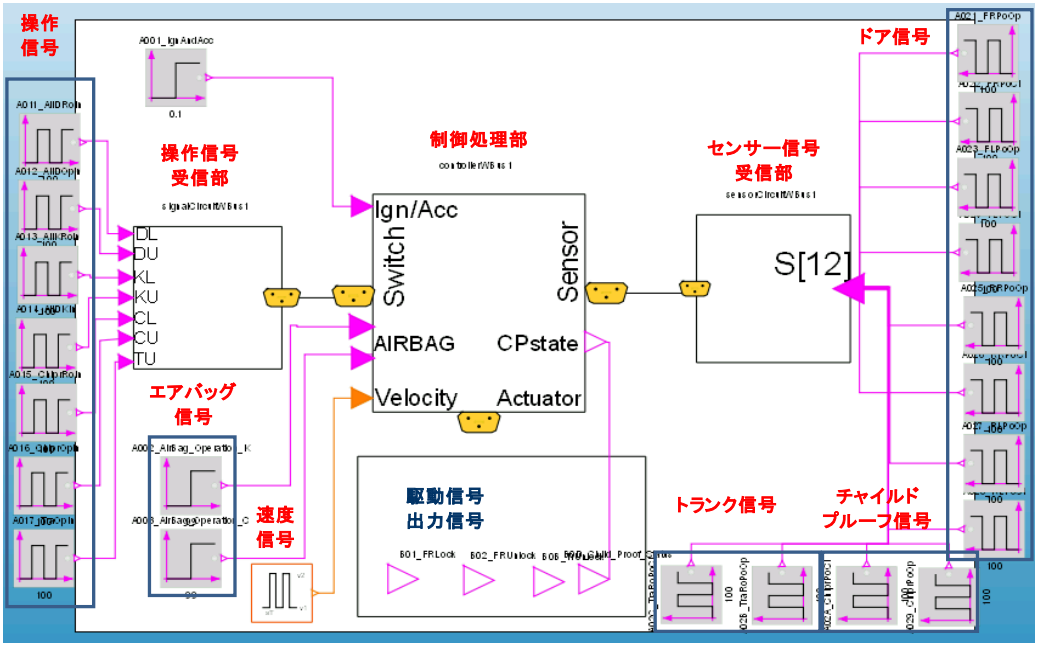


Fig4-18 本システムのプラント制御モデル

4.6.2　シミュレーションツール

本実験では、プラント制御モデルの挙動を確認するために、OpenModelica 対応のシミュレーションツールとして、スウェーデンの Mathcore Engineering 社が提供する MathModelica を使用した。

OpenModelica で記述したプラント制御モデルの入出力部分に対して、模擬信号を送信した結果を MathModelica でシミュレーションすることで、動作確認を行った。MathModelica を使用したシミュレーションの結果サンプルは Fig4-19 の通りである。

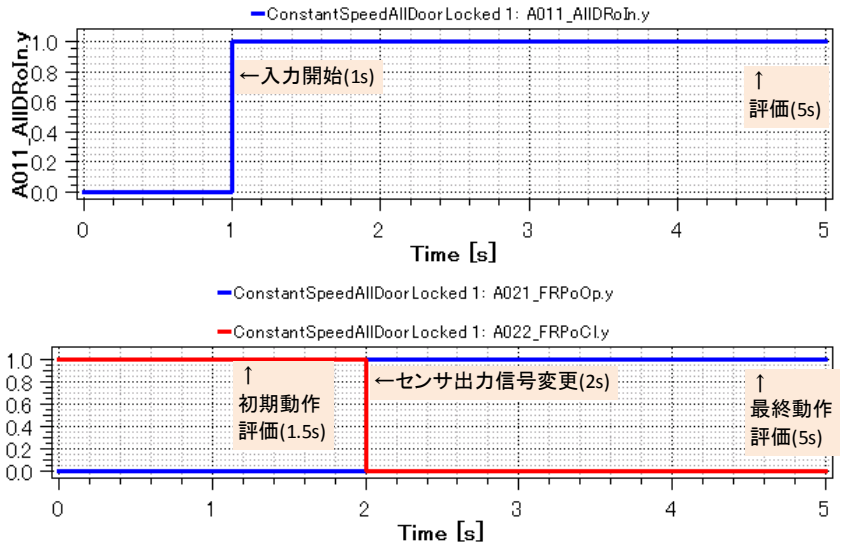


Fig4-19 MathModelica によるシミュレーション結果サンプル

ここでは、詳細設計で作成したソフトウェアについての仕様が、概要設計で策定した仕様と齟齬がないかについて、プラント制御モデルの振る舞いを確認することで検証した。また、プラント制御モデルの振る舞いをシミュレーションによって確認することで、策定した仕様が妥当であるかの確認についても実施した。

4.7 本実験におけるトレーサビリティ管理

本実験ではトレーサビリティ管理を行うためのツールとして、TERAS を利用した。TERAS の選定背景は前述（「2.1 実験の範囲」参照）の通りである。トレーサビリティ管理を行うことにより発生したタスクは Fig4-20 の通りである。本実験では、これらのタスクを実施するのに要した作業工数の測定を行った。

アプローチ	概要
設計文書間の関係性定義	・本実験における設計文書間の上流･下流の関係性を明確し、定義する。
TERAS TRAへの設計文書登録	・文書解析機能を利用して、作成された設計文書を、TERAS TRAへ登録する。
トレーサビリティ情報登録	・設計文書同士の関連する要素を抽出する。 ・関連する要素同士をリンクすることで、トレーサビリティ情報を登録する。
カバレッジのレビュー	・作成するべき設計文書が不足していないかを確認する。 ・不足している設計文書を作成する必要があるかどうかを検討する。

Fig4-20 トレーサビリティ管理のアプローチ

4.7.1 設計文書間の関係性定義

設計文書のトレーサビリティ管理を行うための準備段階として、本実験で作成した設計文書の種別、及び上流･下流の関係性について定義する。本実験で定義した設計文書間の関係性は、Fig 4-21 の通りである。

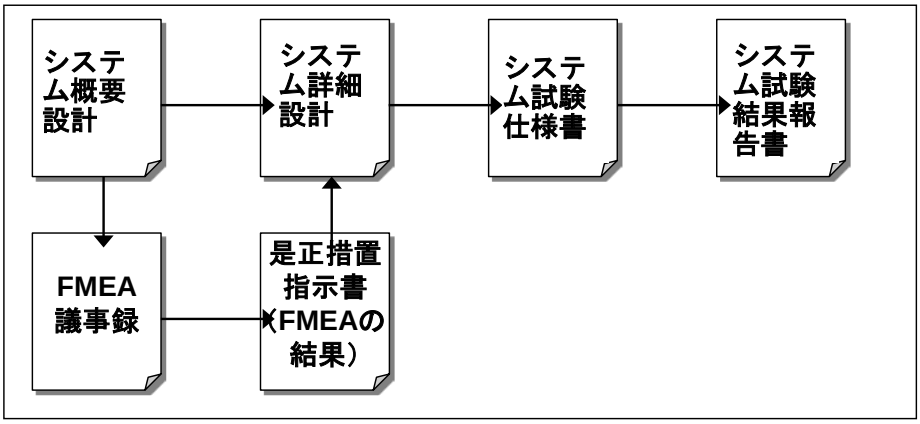


Fig4-21 設計文書間の関係性

本実験において定義した設計文書の関係性は、TERAS のスキーマエディタ機能（Fig4-22 参照）にて設定を行った。スキーマエディタ機能は、トレーサビリティ管理者が設計文書の関係性をカスタマイズするための機能である。

この作業によって、TERAS を利用してトレーサビリティ管理を実施するための準備が完了する。

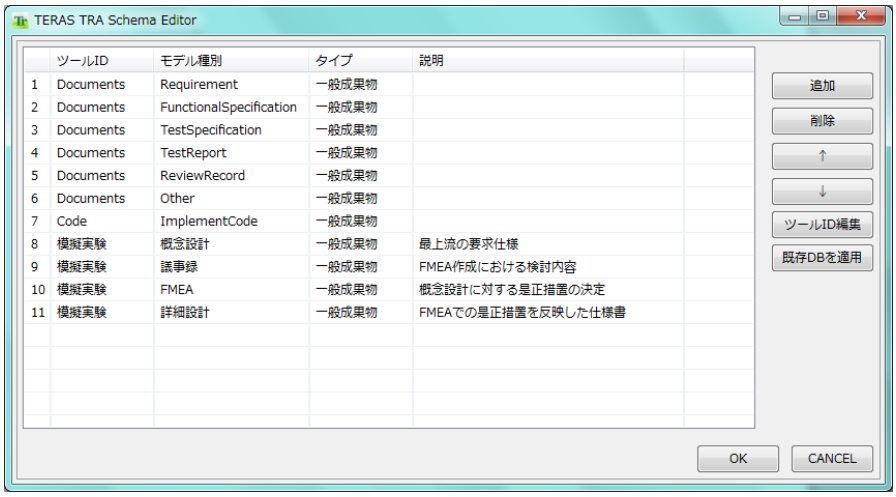


Fig4-22 スキーマエディタのイメージ

4.7.2 TERAS への設計文書登録

設計文書を新規に作成した、もしくは変更を行った後に TERAS へ登録する。TERAS では、Microsoft Office の Word、Excel、PowerPoint で作成された文書を自動的に要素分解し、登録することができる。各要素は階層構造に分解され、以下、上位が親要素、下位を子要素と呼ぶこととする。

トレーサビリティ管理者はこの機能を利用して、作成された設計文書を TERAS に登録する。

4.7.3 トレーサビリティ情報登録

トレーサビリティ管理者は作成された設計文書間の関係性を考慮しながら、トレーサビリティ情報を示すリンクを作成する。TERAS では、リンクを作成するためのリンクエディタ（Fig4-23 参照）を視認性の高い GUI で提供しているため、トレーサビリティ管理者は直感的にリンク作成の操作を実施することができる。

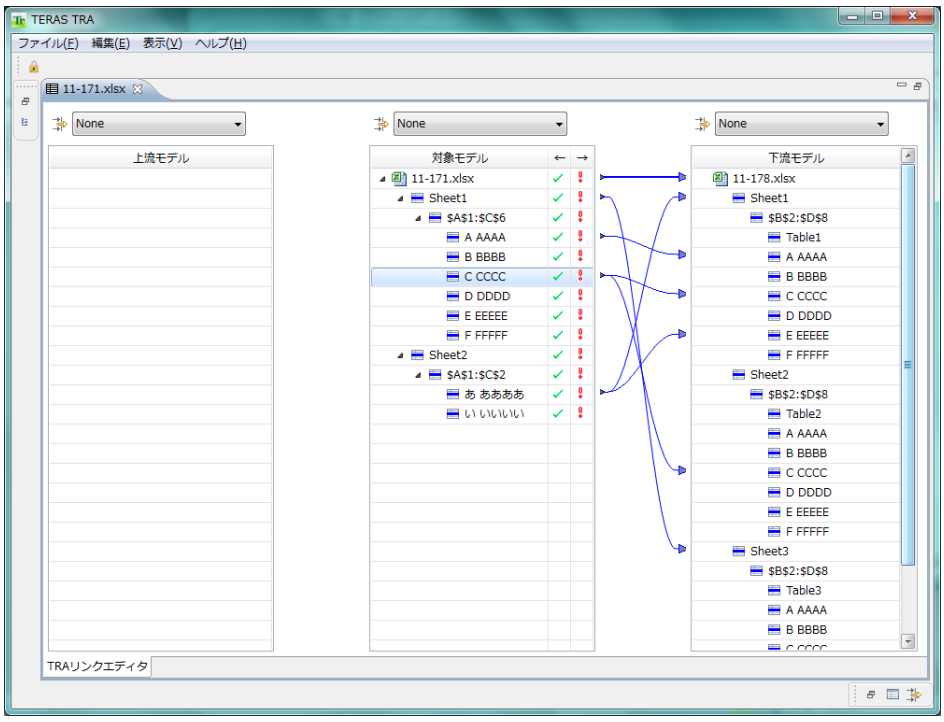


Fig4-23 リンクエディタのイメージ

本実験では、トレーサビリティ情報を登録する際の工数膨張を防止するために、以下の条件を満たす場合には、親要素のみにリンクを作成し、その子要素に対してはリンク作成を除外するという方針を採った。

- 全ての子要素が、上流から検討された経緯が同一である親要素
- 全ての子要素が、下流において同一の対応が採られている親要素
- 上流から検討された経緯が同一である複数の要素グループ
- 下流において同一の対応が採られている複数の要素グループ

4.7.4 カバレッジのレビュー

TERAS ではカバレッジ確認機能を利用することで、不足している設計文書を抽出することができる。Fig 4-24 のように作成すべき設計文書の過不足がアイコンで表示され、どの設計文書が不足しているかを把握することができる。

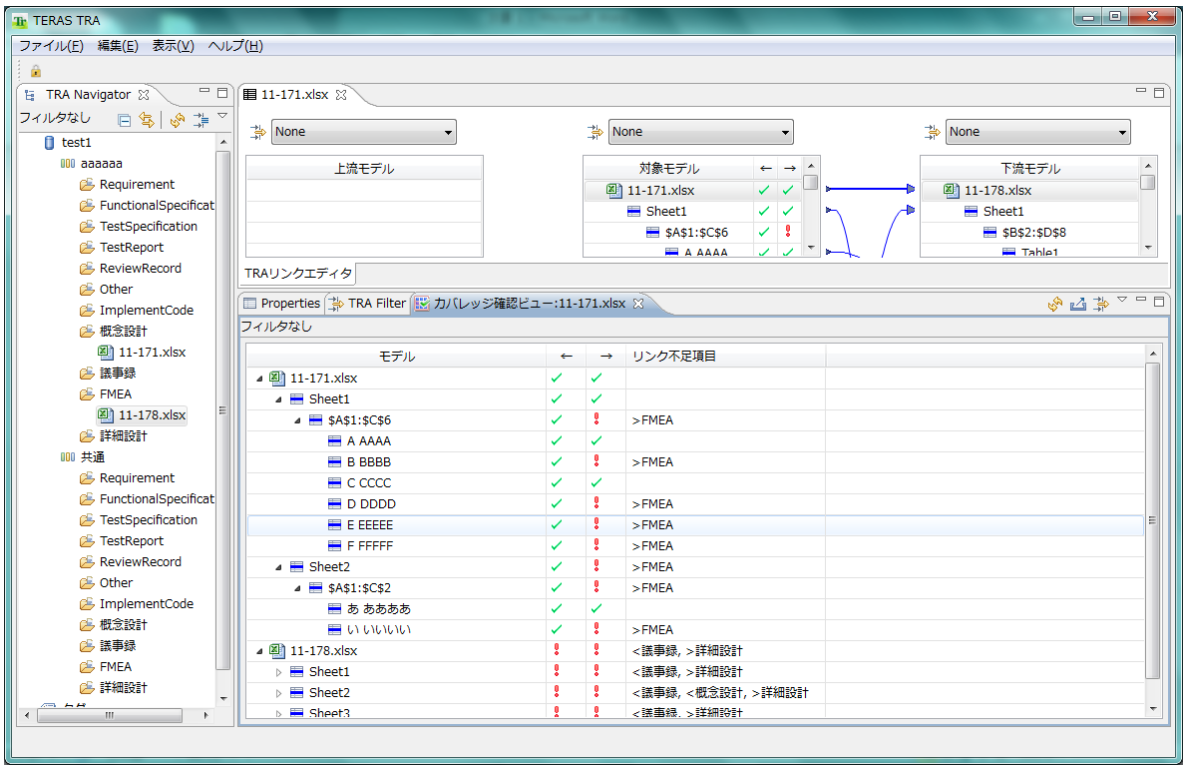


Fig4-24 カバレッジ確認ビュー

トレーサビリティ管理者は、カバレッジが不足している設計文書に対して、新規に成果物を作成する必要があるかどうかを検討し、カバレッジ不足解消のための対策案を提示する。

本実験におけるトレーサビリティ管理では、設計文書が新しく作成された際や、設計文書が更新された際に、「4.7.2 TERAS への設計文書登録」から「4.7.4 カバレッジのレビュー」までの手順を実施した。トレーサビリティ管理の結果、全ての項目においてカバレッジ不足が解消され、設計文書間のトレーサビリティが確保された状態を実現することが可能になった。

4.8 メトリクス収集方法

本実験では、「2.3 評価方法」で定義したとおり、「概要設計～システム設計」の開発プロセスにおいて、トレーサビリティ管理に係る付加工数及び、モデルベース開発に係る工数・効果について、以下 3 点の評価指標について計測・評価を行う。

- モデルベース開発実行に要する工数
- モデルベース開発により発見された不具合数割合
- トレーサビリティ管理に要する工数の計測と算定

4.8.1　モデルベース開発のシミュレーションに要する工数

従来型の開発プロセスにモデルベース開発を導入することにより、新たに発生した作業工数を時間単位で測定した。本実験では、モデルベース開発におけるシミュレーション検証に関する作業工程を以下のように区分して測定を実施した。

・仕様理解

システム試験を実施するために使用するプラント制御モデルの作成担当者が、システム設計仕様書（RFQ）に記載されている内容を理解することに要した作業を対象に工数を測定した。

・モデル開発

プラント制御モデルの作成担当者が、システム設計仕様書（RFQ）に記載されている内容を、OpenModelica を利用してモデルとして記述するのに要した作業を対象に工数を測定した。

・試験仕様書作成

試験仕様書作成担当者が、システム設計仕様書（RFQ）を基にシステム試験仕様書を作成するのに要した作業を対象に工数を測定した。

・シミュレーション実行

試験実施担当者が、MathModelica を利用してシミュレーションにより検証を行うのに要した作業を対象に工数を測定した。

・試験結果報告

試験実施担当者がシミュレーションの実行結果を分析し、システム試験結果報告書を作成するのに要した作業を対象に工数を測定した。

システム設計工程において、ISO26262 で要求されるレベルの品質を確保するために、本来はシミュレーション試験を繰り返し実施する必要がある。本実験では、1 回目のシミュレーション試験を実施するのに要した工数については実測値を計測し、2 回目の試験については実施するのに要する工数を推定して、1 回目に要した試験工数と比較・分析した。

4.8.2　モデルベース開発により発見された不具合数割合

モデルベース開発の導入により、制御ソフトウェアやハードウェアを開発する前に、仕様書をモデルによって作成し、シミュレーションの形で実際の動きを確かめることができるようになる。このことから、上流工程で仕様書の不具合が検出され、後工程で検出された場合よりも手戻り工数を削減できる効果があるとされている。

本実験では、モデルベース開発導入による、手戻り工数削減の効果を評価するために、詳細仕様書の中に予め不具合を設定した。設定した不具合数に対して、モデルベース開発によって発見された不具合の割合を測定することによって、モデルベース開発の導入効果の評価を実施した。

4.8.3　トレーサビリティ管理に要する工数

TERAS を活用したトレーサビリティ管理を行うために、新たに発生した作業の実施工数を時間単位で測定した。作業工程ごとの測定方法は以下の通りである。

・設計文書間の関係性の明確化

トレーサビリティ管理者が設計文書間の関係性を明確にして、トレーサビリティ管理を実施するための準備に関する作業を対象に工数を測定した。

・TERAS への設計文書登録

トレーサビリティ管理者が設計文書をTERASに登録するのに要した作業を対象に工数を測定した。

・トレーサビリティ情報登録

トレーサビリティ管理者がTERASに登録された設計文書同士のトレーサビリティ情報を登録するのに要した作業を対象に工数を測定した。

・カバレッジのレビュー

トレーサビリティ管理者と開発担当者がTERASによって抽出されたカバレッジ不足への対応に要した作業を対象に工数を測定した。

4.9 実験の実施

本実験は、約 2 ヶ月の期間で実施した。システム設計仕様書（RFQ）の作成や FMEA の実施など開発担当者が行う作業内容については、ベリサーブが実施した。シミュレーションによる試験に関する作業のうち、システム試験仕様書の作成はベリサーブが実施し、プラント制御モデルの作成やシミュレーション実行、システム試験結果報告書の作成については、別の組織のチーム（以下、「協力チーム」という）が実施した。

また、トレーサビリティ管理に関する作業についてもベリサーブが実施したが、開発担当者とは異なる担当者がトレーサビリティ管理者の役割を担った。

本実験において実際に行われた詳細な実施手順を以下に示す。

4.9.1　モデルベースシミュレーション

シミュレーションを利用したモデルベース開発の詳細な実施手順は、以下の通りである。

a. シミュレーション実施計画の策定

2012 年 4 月 19 日にベリサーブと協力チームが打ち合わせを実施し、本実験においてのシミュレーションによるモデルベース開発の実施計画を策定した。

b. システム設計仕様書の提示

4 月 27 日にベリサーブの開発担当者が試験仕様書作成担当者、及び協力チームに対して、詳細設計部分までの作成が完了したシステム設計仕様書（RFQ）を提示した。

c. システム試験仕様書の提示

5 月 8 日にベリサーブの試験仕様書作成担当者が協力チームに対してシステム試験仕様書を提示した。

d. シミュレーションの実施

5 月 8 日から 18 日にかけて、協力チームがプラント制御モデルを構築した上で、システム試験仕様書に則ってシミュレーションによる試験を実施した。

e. システム試験結果報告書の提出

5 月 18 日に協力チームがベリサーブに対してシステム試験結果報告書を提出した。

4.9.2　トレーサビリティ管理

トレーサビリティ管理の詳細な実施手順は以下の通りである。

f. トレーサビリティ管理計画の策定

4 月 20 日にベリサーブのトレーサビリティ管理者と開発担当者が打ち合わせを行い、トレーサビリティ管理計画を策定した。

g. トレーサビリティ管理の実施準備

4 月 22 日から 27 日にかけて、トレーサビリティ管理者がトレーサビリティ管理を実施するための準備作業を行った。準備作業には、TERAS のインストール、設計文書間の関係性明確化、TERAS の設定変更などの作業が含まれる。

h. 設計文書の提示

トレーサビリティ管理対象の設計文書がトレーサビリティ管理者に対して以下の日程で提示された。

・4 月 25 日　是正措置指示書（FMEA の結果）
・4 月 27 日　システム設計仕様書（RFQ）
・5 月 8 日　FMEA 議事録
・5 月 8 日　システム試験仕様書
・5 月 18 日　システム試験結果報告書

i. トレーサビリティ管理の実施

4 月 27 日から 5 月 18 日にかけて、トレーサビリティ管理者が提示された設計文書を TERAS に登録し、トレーサビリティ情報を作成するなどのトレーサビリティ管理作業を実施した。

j. カバレッジのレビュー

5 月 21 日から 22 日にかけて、トレーサビリティ管理者、開発担当者、試験仕様書作成担当者が打ち合わせを行い、トレーサビリティ管理を実施した結果カバレッジが不足している設計文書について対応方法の検討を行った。

5. 実験結果・分析

以下では、本実験で品質説明力を強化するために注目した「モデルベース開発におけるシミュレーション」と「設計文書のトレーサビリティ管理」とを、開発作業に導入した際の工数及び効果について、実験結果に基づいて分析を行う。

5.1 シミュレーション検証の実験結果・分析

5.1.1 シミュレーション検証の付加作業項目

従来型の開発プロセスにモデルベース開発におけるシミュレーション検証を導入することにより、新たに発生した作業項目は 5 項目であった。以下では、各作業項目の詳細を示す。

・仕様理解

本実験では、プラント制御モデルの作成を開発担当組織とは異なる組織で実施したため、モデル作成担当者がシステム設計仕様書（RFQ）の記載内容を把握するために要した作業を説明する。

モデル作成担当者は、開発担当者から入手したシステム設計仕様書（RFQ）を読み込むことによって、本システムの仕様を理解した。また、システム設計仕様書（RFQ）を読むだけでは理解が難しかった項目については、開発担当者に対して質問することで、仕様理解を補填した。

開発担当者はモデル作成担当者からの質問に対してシステム設計仕様書（RFQ）の記載内容を確認した上で、回答を行った。一部、システム設計仕様書（RFQ）の記述にミスや曖昧な箇所があった場合には見直しを実施した。

・モデル開発

プラント制御モデルを作成するための作業について詳細を説明する。

モデル作成担当者はシステム設計仕様書（RFQ）を基に、OpenModelica を利用してモデルを記述した。また、モデルを記述する中で生じた疑問点を開発担当者に質問した。

開発担当者はモデル作成担当者からの質問に対してシステム設計仕様書（RFQ）の記載内容を確認した上で、回答を行った。一部、システム設計仕様書（RFQ）の記述に誤りや曖昧な記述があった場合には見直しを実施した。

・試験仕様書作成

システム試験仕様書を作成するための作業について詳細を説明する。

試験仕様書作成担当者は開発担当者から入手したシステム設計仕様書（RFQ）を読み込むことで、本システムの仕様を理解した。また、システム設計仕様書（RFQ）を読むだけでは理解が難しかった項目については、開発担当者に対して質問するこ

とで、仕様理解を補填した。

開発担当者は試験仕様書作成担当者からの質問に対してシステム設計仕様書（RFQ）の記載内容を確認した上で、回答を行った。一部、システム設計仕様書（RFQ）の記述に誤りや曖昧な記述があった場合には見直しを実施した。

・シミュレーション実行

MathModelica を利用してシミュレーションにより試験を実施する作業について詳細を説明する。

試験実施担当者は、試験仕様書作成担当者から入手したシステム試験仕様書を読み込むことによって内容を理解した。また、システム試験仕様書を読むだけでは理解が難しかった項目については、試験仕様書作成担当者に対して質問することで、内容理解を補填した。

試験仕様書作成担当者は試験実施担当者からの質問に対してシステム試験仕様書の記載内容を確認した上で、回答を行った。一部、システム試験仕様書の記述に誤りや曖昧な記述があった場合には見直しを実施した。

モデルベースシミュレーションはコンピュータの自動演算によって実施されるため、試験実施担当者がシステム試験仕様書を基にして、自動演算を行うためのスクリプト作成した。シミュレーション実行は MathModelca を利用してコンピュータが自動演算により実施された。

・試験結果報告

シミュレーションの実行結果から、システム試験結果報告書を作成するための作業について詳細を説明する。

試験実施担当者は、シミュレーションの実行結果から、システム試験仕様書の各試験項目について合否を判断した。合否結果の一覧と試験実施の概要をまとめた資料をシステム試験結果報告書としてまとめ、試験仕様書作成担当者、及び開発担当者に提示した。

5.1.2　シミュレーション検証工数の計測

「4.8.1 モデルベース開発のシミュレーションに要する工数」で示した通り、モデルベース開発におけるシミュレーション検証を実施する際に、作業工程ごとに発生した付加工数を時間を単位として測定した。

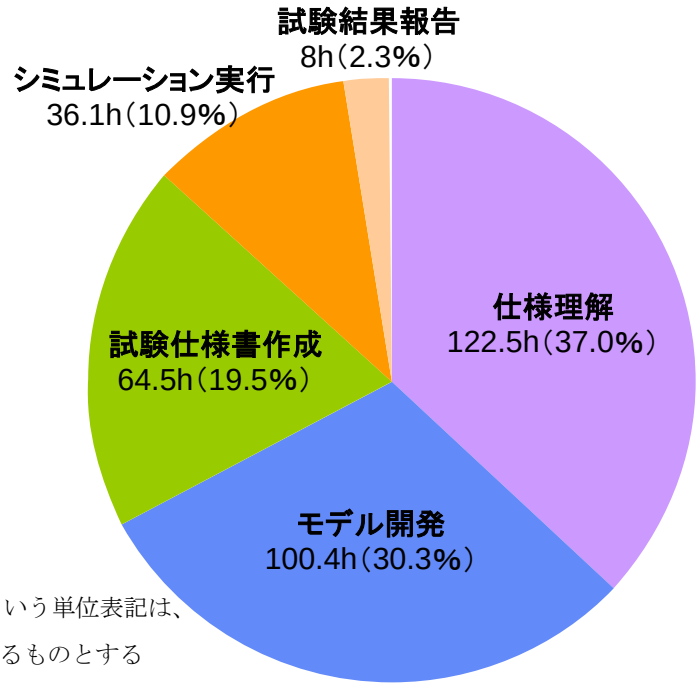
本実験では、シミュレーション試験を 1 回のみ実施したが、本来は 2 回実施することで、システム設計工程で十分に品質を確保する必要があると考えられる。したがって、ここでは、1 回目のシミュレーション試験については実測値を計測し、2 回目のシミュレーション試験については、1 回目の実施結果に基づいて学習効果を加味した工数を推定している。

本実験における作業工程ごとの工数、及び内訳は Fig5-1、Fig5-2 の通りである。

単位：人時間

	1回目（実測値）	2回目（推定値）	合計
仕様理解	98.5	24	122.5
モデル開発	72.4	28	100.4
試験仕様書作成	60.5	4	64.5
シミュレーション実行	24.1	12	36.1
試験結果報告	4	4	8
合計	259.5	72	331.5

Fig5-1 シミュレーション検証に要する工数



グラフ中の［h］という単位表記は、
［人時間］を意味するものとする

Fig5-2 シミュレーション検証の作業工程別工数内訳

本実験を実施した結果、モデルベース開発におけるシミュレーション検証を導入することによって新たに発生する工数は人月換算（1 人月=160 人時間：1 ヶ月を 1 日 8 時間、20 営業日として計算）で 2.1 人月であることが分かった。

5.1.3 シミュレーション検証工数の計測結果分析

シミュレーション検証の作業工程別工数内訳（Fig5-2 参照）から、仕様理解、モデル開発、及び試験仕様書作成といった事前準備の作業が全体の 86.8%を占めることが分かった。事前準備作業で作成された資産については、2 回目以降の作業に大部分を流用することが可能であるため、作業工数を大幅に削減できるものと推測される。

一方で「シミュレーション実行」や「試験結果報告」の作業工程については、過去の資産を流用することが困難なため、2 回目以降のシミュレーション試験であっても工数削減が比較的難しいことが考えられる

5.1.4　シミュレーション検証により発見された不具合の数

モデルベース開発におけるシミュレーション検証を実施する中で、開発担当者が試験実施担当者から受けた指摘件数は 29 件であった。29 件の指摘事項を調査した結果は Fig5-3 に示すとおりである。

	作業工程	指摘件数	不具合件数	設定 不具合件数	不具合発生原因
I-1	仕様理解	10	10	0	設計仕様書の記述ミス
I-2	モデル開発	3	3	2	設計仕様書の記述が曖昧
I-3	試験仕様書作成	-	-	-	
I-4	シミュレーション実行	2	2	2	設計仕様書の記述ミス
I-5		2	0	0	システム試験仕様書の記述が曖昧
I-6		12	1	1	仕様考慮不足
I-7	試験結果報告	-	-	-	
合計		29	16	5	

Fig5-3 シミュレーション検証により発見された不具合

試験実施担当者からの指摘事項を調査した結果、システム設計仕様書（RFQ）の不具合が 16 件発見された。また、予めシステム設計仕様書（RFQ）内に設定した不具合 5 件について、5 件全てが発見されていることから、本実験でモデルベース開発により発見された不具合の割合は 100%となった。

この結果から、モデルベース開発におけるシミュレーション検証について、一定レベルの不具合検出能力があることが判明した。

本実験では、試験実施担当者を開発担当組織とは異なる組織の者が担当し、第三者の視点からの検証を試みたため、仕様理解に多くの工数を要している。しかしながら、仕様理解の工程で最も多くの不具合が発見されている上に、これらの不具合は開発担当者が意図していない不具合であった。このような結果から、第三者の視点から検証を実施することによって、より多くの不具合が発見できる効果を確認することができた。

5.1.5　シミュレーション検証導入効果分析

モデルベース開発におけるシミュレーション検証を導入することによって、従来型の開発では、システム設計よりも後の工程にならないと発見できない不具合が、システム設計工程で発見することが可能になることが分かった。

本実験で発見された不具合が、システム設計工程で発見できなかった場合の手戻り工数を試算する

I-1、I-2 の 13 件（Fig5-3 参照）については、従来型の開発プロセスでは、コーディング時に指摘されることが多い項目であるため、不具合がコーディング時に見つかったと仮定して、修正に係る工数を見積もった。修正に係る工数は Fig5-4 に示す通りである。

	対応人数	時間	1件当たり工数	合計工数
指摘事項の精査	1人	4時間	4時間	52時間
設計仕様書の変更箇所特定	2人	4時間	8時間	104時間
設計仕様書の変更	1人	4時間	4時間	52時間
設計仕様書の変更レビュー	2人	0.5時間	1時間	13時間
コーディング、単体試験	1人	4時間	4時間	52時間
合計	-	-	21時間	273時間

Fig5-4 I-1、I-2 の指摘事項（13 件）への予測対応工数

I-4 の 2 件（Fig5-3 参照）の不具合は、シミュレーション実行を実施するまで発見できなかった不具合であるため、従来型の開発プロセスにおいてはシステム試験工程で発見されることが予想される。これらの不具合がシステム試験工程で発見されたと仮定して、対応に係る工数を試算した結果は Fig5-5 に示す通りである。

	対応人数	時間	1件当たり工数	合計工数
指摘事項の精査	1人	4時間	4時間	8時間
設計仕様書の変更箇所特定	2人	4時間	8時間	16時間
設計仕様書の変更	1人	4時間	4時間	8時間
設計仕様書の変更レビュー	2人	0.5時間	1時間	2時間
コーディング、単体試験	1人	4時間	4時間	8時間
試験仕様書変更	1人	4時間	4時間	8時間
システム試験	1人	4時間	4時間	8時間
合計	-	-	29時間	58時間

Fig5-5 I-4 の指摘事項(2 件)への予測対応工数

I-5 の 2 件の指摘事項は、シミュレーション実行を実施した際にシステム試験仕様書の記述が曖昧であったことから指摘を受けたものであり、システム設計仕様書（RFQ）を精査したところ、不具合ではないことが判明した。

従来型の開発プロセスにおいても、システム試験仕様書の記述が曖昧になってしまい、指

摘を受ける可能性が高いことから、手戻りが発生するものと考え、対応に要する工数を試算した。試算結果は Fig5-6 に示す通りである。

	対応人数	時間	1件当たり工数	合計工数
指摘事項の精査	1人	4時間	4時間	8時間
試験仕様書修正	1人	4時間	4時間	8時間
システム試験	1人	4時間	4時間	8時間
合計	-	-	12時間	24時間

Fig5-6 I-5 の指摘事項(2 件)への予測対応工数

I-6 の指摘事項である 12 件は全てシミュレーション実行の中で発見された。それぞれ異なるテスト項目により発見されたため、異なる問題として指摘されたが、内容を確認したところ、原因となった不具合は全て同一のものであることが確認された。

従来型の開発プロセスにおいては、システム試験工程で発見され、それぞれ異なる問題として指摘される可能性が高いと考えられるため、指摘事項の精査に関する作業工数は指摘件数に比例するものと想定される。

また、指摘事項を調査した結果、当該不具合はシステム設計仕様書（RFQ）の修正箇所が広範囲に渡ることが判明したため、「設計仕様書の変更」から「システム試験」の作業までについても、その他の不具合に比べて多く工数がかかることが推測される。

これらの状況を踏まえて推定した工数は Fig5-7 の通りである。

	対応人数	時間	1件当たり工数	合計工数
指摘事項の精査	1人	4時間	4時間	48時間
設計仕様書の変更箇所特定	2人	16時間	32時間	32時間
設計仕様書の変更	1人	20時間	20時間	20時間
設計仕様書の変更レビュー	2人	8時間	16時間	16時間
コーディング、単体試験	2人	20時間	40時間	40時間
試験仕様書変更	2人	12時間	24時間	24時間
システム試験	3人	8時間	24時間	24時間
合計	-	-	160時間	204時間

Fig5-7 I-5 の指摘事項(12 件　変更 1 箇所)への予測対応工数

以上により、従来型の開発プロセスにて本システムを開発した場合、システム設計仕様書（RFQ）の不具合をシステム設計工程以降で発見することによる手戻り工数は 559 人時間であると推定される。これを人月換算（1 人月=160 人時間：1 ヶ月を 1 日 8 時間、20 営業日

として計算）すると 3.5 人月となる。

モデルベース開発におけるシミュレーション検証の導入で発生した工数負荷が 2.1 人月であったため、従来型の開発プロセスを実施した場合と比べると、差し引き 1.4 人月の工数軽減が見込まれることが分かった。

したがって、モデルベース開発の導入により、システム設計工程では工数が増加するものの、開発工程全体としては工数の削減効果を見込むことができると考えられる。

5.2 トレーサビリティ管理の実験結果・分析

TERAS へのトレーサビリティ情報を登録した設計文書などのファイル数は 38 個であった。TERAS を利用して要素解析を行ったところ、627 個の要素に分解された。また、要素同士のトレーサビリティ情報を示すリンク線の数は 833 個となった。Fig5-8、Fig5-9、Fig5-10 に TERAS へのトレーサビリティ情報登録画面を示す。

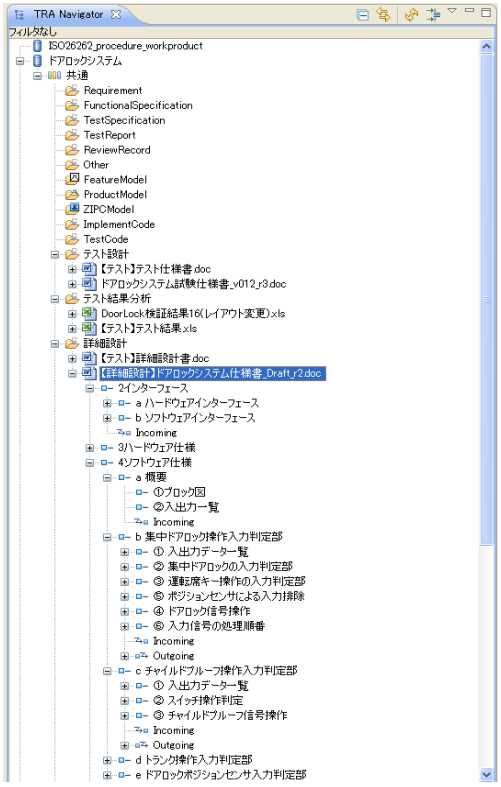


Fig5-8 設計文書の解析結果サンプル

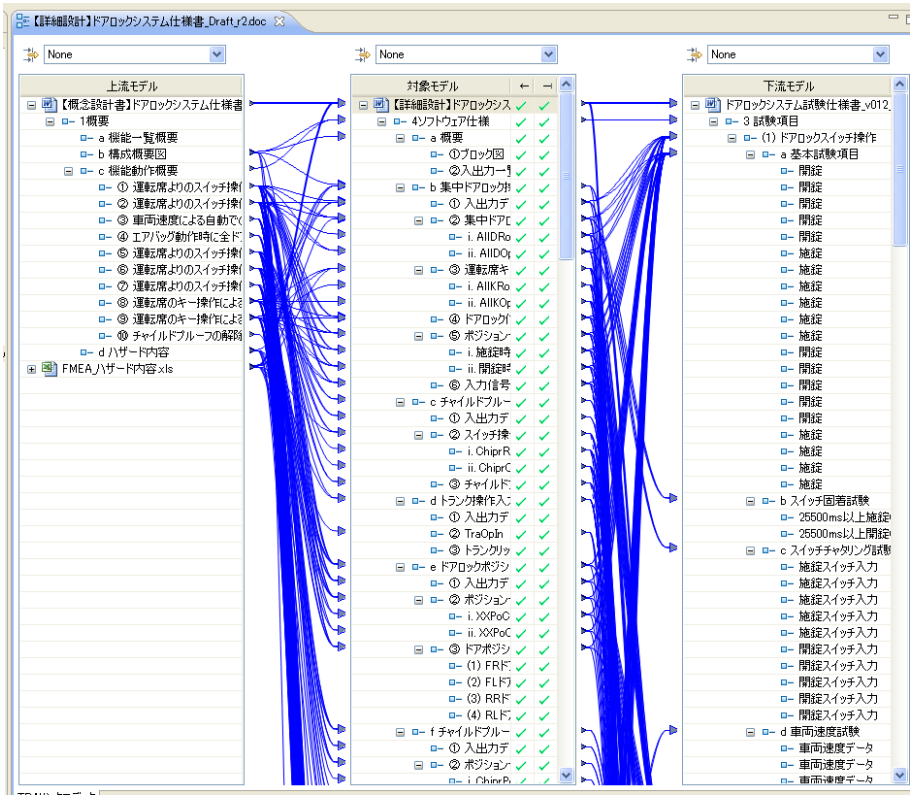


Fig5-9 設計文書のトレーサビリティ登録の結果サンプル

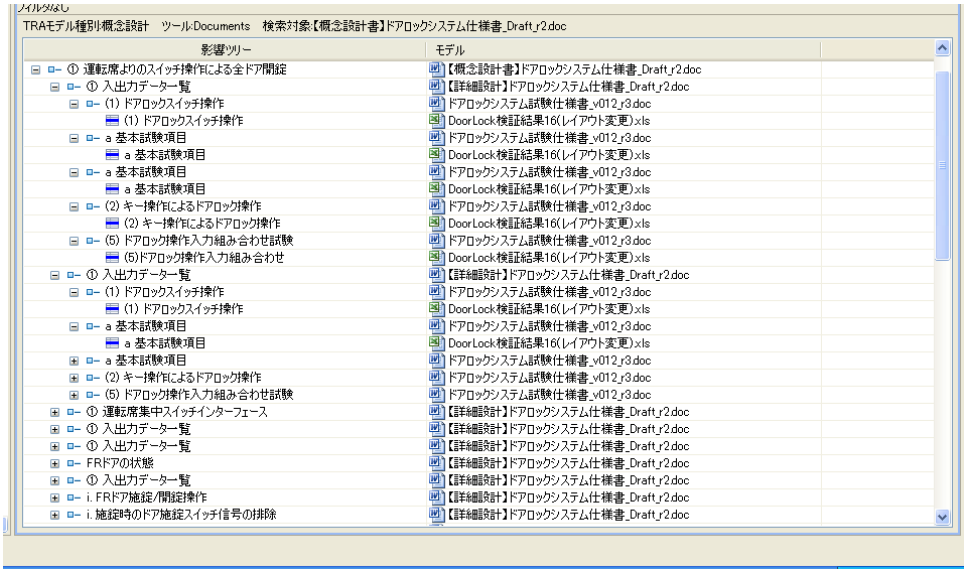


Fig5-10 変更箇所の影響範囲を検索した結果サンプル

5.2.1 トレーサビリティ管理に要する工数

従来型の開発プロセスにトレーサビリティ管理を導入することにより、新たに発生した作業項目は 4 項目であった。以下では、各作業項目の詳細を示す。

・設計文書間の関係性の明確化

トレーサビリティ管理者は既存の開発プロセスと設計文書について、開発担当者から説明を受ける形で確認を行った。

また、設計文書の関係性を明確化するために、トレーサビリティ管理者と開発担当者が会議を行った。会議の中で議論された内容を基にして、トレーサビリティ管理者がトレーサビリティ管理方針の素案を作成し、開発担当者に提示した。トレーサビリティ管理者と開発担当者は、トレーサビリティ管理方針の素案を基にして内容を検討し、関係者間の合意を形成しながらトレーサビリティ管理方針を決定した。

トレーサビリティ管理者は、トレーサビリティ管理方針に沿うように、TERASの設定を変更した。

・TERAS への設計文書登録

トレーサビリティ管理の対象とする設計文書が完成すると、設計文書の作成担当者はトレーサビリティ管理者へと設計文書を提示した。

トレーサビリティ管理者は入手した設計文書をTERASの文書解析機能を利用して登録した。TERAS の初期設定では、トレーサビリティ管理方針に従った形で設計文書を解析できない場合には、設定を変更することで対応した。

また、設計文書の作成担当者は設計文書に変更を加えた場合、更新した設計文書をトレーサビリティ管理者へと提示した。

トレーサビリティ管理者は更新された設計文書の差分を TERAS に登録した。

・トレーサビリティ情報登録

トレーサビリティ管理者はTERASに登録された設計文書間にリンク線を作成することで、トレーサビリティ情報を登録した。入手した設計文書を読むだけでは、どのようなリンク線を作成すればよいのか判断できない場合には、設計文書の作成担当者に質問を行った。

設計文書の作成担当者はトレーサビリティ管理者からの質問に対して設計文書の記載内容を確認した上で、回答を行った。一部、設計文書の記述に誤りや曖昧な記述があった場合には見直しを実施した。

・カバレッジのレビュー

トレーサビリティ管理者は TERAS のカバレッジ確認機能を利用して、不足している設計文書を抽出した。抽出された設計文書が明らかに不要であるとトレーサビリティ管理者が判断できる場合には、TERAS に登録されている要素に対して、例外設定を加えることによってカバレッジ確認の対象から除外した。再度カバレッジ確認を実施した結果、不足している設計文書については、設計文書の作成担当者に不足文書の一覧を提示した。

設計文書の作成担当者は、提示された一覧を確認し、設計文書見直しの要否について検討を行った。見直しが不要な場合はトレーサビリティ管理者に連絡して、カバレッジ確認の対象からの除外を申請した。見直しが必要な場合は設計文書を更新、もしくは新規作成した上で、トレーサビリティ管理者に提示した。

5.2.2　トレーサビリティ管理工数の計測

「4.8.3 トレーサビリティ管理に要する工数」で示した通り、トレーサビリティ管理を実施する際の作業工程ごとに発生した付加工数を時間単位で測定した。

本実験における作業工程ごとの工数、及び内訳は Fig5-11 の通りである。

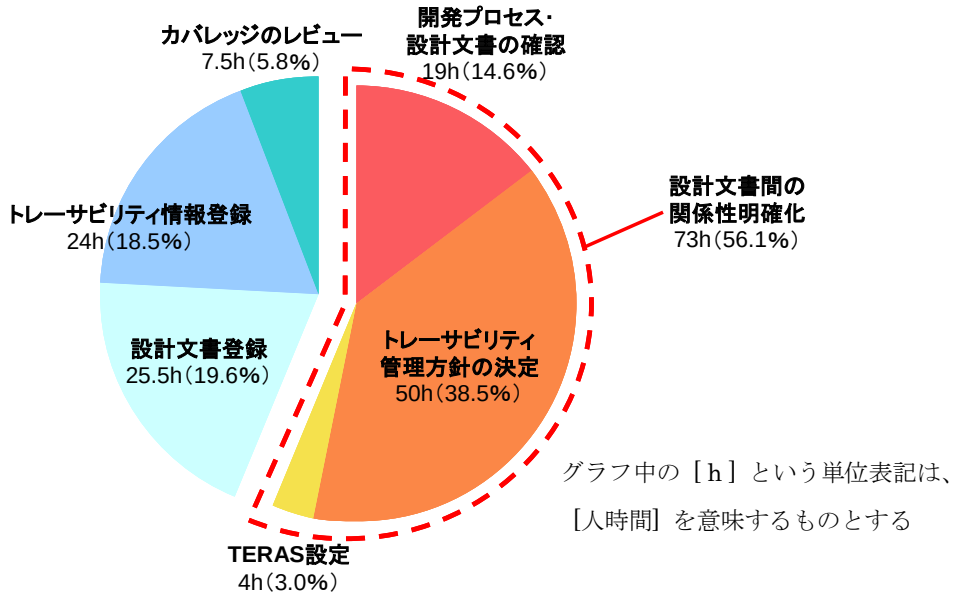


Fig5-11 トレーサビリティ管理の作業工程別工数内訳

本実験の結果、トレーサビリティ管理を導入することによって新たに発生する工数は 130 人時間、人月換算（1 人月=160 人時間：1 ヶ月を 1 日 8 時間、20 営業日として計算）で 0.8 人月であることが分かった。

5.2.3　トレーサビリティ管理工数の計測結果分析

本実験では、システム設計仕様書を作成した開発担当者とは異なる担当者が専任としてトレーサビリティ管理を実施した。トレーサビリティ管理のための作業としては、半分以上の工数をトレーサビリティ管理の準備作業（Fig5-11「設計文書間の関係性明確化」に費やした。

最も時間を要した作業は、トレーサビリティ管理方針を決定する作業で、全体の 38.5%を費やした。この作業は、トレーサビリティ管理者と開発担当者が打ち合わせを実施することで、開発プロセスの中で作成される設計文書の種別を特定し、設計文書間の関係性を明確にする作業である。

このことから、トレーサビリティ管理を導入する際には、トレーサビリティ管理を実施す

ることよりも、トレーサビリティ管理をどのように実施するかを決定することに多くの時間が費やされることが分かる。

5.2.4　トレーサビリティ管理導入効果分析

本実験において、トレーサビリティ管理の開始後ににシステム設計仕様書（RFQ）に対して 15 箇所の変更が加えられた。システム設計仕様書（RFQ）に変更を加えているため、システム設計仕様書（RFQ）の情報を基に作成されている設計文書についても、影響を受ける範囲について変更要否を検討し、必要に応じて設計文書を変更しなければならない。本実験では、TERAS の影響範囲検索機能を利用して、範囲を絞り込んだ上で、変更要否の検討を実施した。

従来型の開発プロセスでは、設計文書間のトレーサビリティが確保されていない状態を想定しているため、システム設計仕様書（RFQ）に変更があった場合には、システム試験仕様書全体に対して変更要否を検討しなければ、必要な変更が漏れてしまうと想定される。

システム設計仕様書（RFQ）に変更が加わった際の変更管理に要する推定工数を、従来型の開発プロセスとトレーサビリティ管理導入時の両者で比較した結果を Fig5-12 に示す。実測した検討項目数から 1 検討項目あたりの検討時間を 2 分として工数を試算した。図の赤枠の中が推定工数である（表中の「時間」は、「人時間」を表す）。

	従来型の開発プロセス	トレーサビリティ管理導入時
検討項目数	3645	511
影響範囲の特定に要する時間	-	1時間
変更要否の検討時間	122時間	17時間
合計（工数）	122時間	18時間

Fig5-12 変更管理に要する推定工数

従来型の開発プロセスと比べると、トレーサビリティ管理を導入したことによる変更管理の削減工数は 104 人時間と推定される。一方で、トレーサビリティ管理導入に際して発生した工数負荷は 130 人時間であるため、全体としては、差し引き 26 人時間の工数増加になるものと考えられる。したがって、トレーサビリティ管理導入により、ある程度の工数負荷がかかることが考えられるものの、品質説明力の向上が見込まれる場合においては、一定の導入効果があるものと想定される。

6. 品質説明力強化に際して想定される課題と考察

本実験の実施結果の分析や、作業を実施する中で問題となった点から、品質説明力強化を目的として本実験で検証を行った手法を導入する際に想定される課題を抽出した。本実験で使用した手法を導入する際には、これらの課題に関して対応方針を検討する必要があると考えられる。

6.1 開発対象の特性を考慮した開発手法の選択

本実験では、品質説明力を向上させるための開発手法として、モデルベース開発におけるシミュレーションを利用して検証を実施した。前述（「5.1.3 シミュレーション検証工数の計測結果分析」参照）の通り、初回のシミュレーション試験には、準備に多くの工数を要する一方、2 回目以降は初回実行時の資産を活用できるため、比較的少ない工数でシミュレーション試験を実施することが可能になる。

本実験では、OSS として提供されている OpenModelica をモデル記述の言語として採用したが、商用のモデル記述ツールを利用する場合には、導入コストが大きくなる。また、事前準備作業に一定以上の工数を要することから、シミュレーション実行による試験を繰り返す回数が少ない場合には、モデルベース開発の導入効果は小さいものと考えられる。

一方、試験する環境設定が多岐にわたる場合や、モデルに設定した値をチューニングしながらシミュレーションを行うような場合には、同じモデルに対して試験を繰り返し行うため、モデルベース開発の導入効果を高めることが可能であると推測できる。

6.2 モデルベース仕様書の導入による曖昧な表現の排除

本実験では、シミュレーション実行による検証を第三者の視点から実施するために、開発担当組織とは異なる組織がシミュレーション用のプラント制御モデル作成とシミュレーション実施を担当した。プラント制御モデルの作成を、客観的な視点で作成したことによって、開発担当者の思い込みによるシステム設計仕様書の不具合を発見することができた。

一方で、システム設計仕様書が自然言語で記載されているために、モデル記述者が仕様を把握する際に、記載されている内容を誤解してしまうなどの問題が発生し、内容を共有するための作業が追加で発生した。

モデルベース開発を導入する場合には、システム設計仕様書を自然言語で記述するのでは無く、形式手法やモデルベースなどの準形式手法によるシステム設計仕様書記述を推進することが望ましいと考えられる。これによって、システム設計仕様書の曖昧な記述や矛盾した記述を極力排除することが可能になり、担当者間での意思疎通が円滑になるものと考えられる。

6.3 トレーサビリティ管理導入時における関係者間の合意形成

本実験におけるトレーサビリティ管理作業の中で、トレーサビリティ管理方針を決定する

作業に最も多くの工数を費やした。

本実験では、トレーサビリティ管理方針を決定するための会議をトレーサビリティ管理者1名と開発担当者2名の合計3名で実施した。しかしながら、実際の開発においては、複数部門から設計文書を収集する必要があるため、トレーサビリティ管理方針を決定する作業については、より一層工数負荷が大きくなることが推測される。

また、設計文書をTERASに登録した後に、トレーサビリティ管理方針を変更する場合には、再度トレーサビリティ情報を登録し直す必要が生じる可能性がある。このことから、トレーサビリティ管理方針を適切に設定できなかった場合には、大きな手戻り工数が発生する要因になりかねない。

したがって、トレーサビリティ管理を開始する前に、関係者間においてトレーサビリティ管理方針に関する合意形成に十分留意することが望ましいと考えられる。

6.4 トレーサビリティ管理ツールの特性を考慮した文書作成

本実験では、トレーサビリティ管理ツールの特性や性能によって、十分なトレーサビリティ管理を実施できなかった設計文書が存在した。

特に本実験を進める上で障害となったのは、ファイルサイズが膨大な設計文書を登録するとツールの動作が遅くなってしまったことと、同一ファイル内でのトレーサビリティが取れなかったことである。

本実験で使用したTERAS以外にも、既に商用ツールとして提供されているトレーサビリティ管理ツールが存在するが、ツールによって性能限界やトレーサビリティ情報を登録する際の制約にばらつきがあるものと推測される。

開発プロセスの中にトレーサビリティ管理を導入する際には、既存の設計文書と条件が合致するようなトレーサビリティ管理ツールを導入する、もしくはコストの関係等で最適なツールが導入できない場合には、設計文書の作成ガイドライン等を定めることによって、トレーサビリティ管理ツールに文書を適合させるなどといった対応が必要になってくると考えられる。

6.5 トレーサビリティ管理の導入による変更管理負荷の軽減効果

本実験で、設計文書間のトレーサビリティ情報を生成した状態で、仕様変更が発生した場合の影響範囲分析を実施した。実際に発生した15件の変更によって影響を受ける可能性がある項目を抽出するために1人時間、変更要否の検討をするために17人時間、合計18人時間を要した。

一方、トレーサビリティ管理が未導入の場合に、同様の作業を実施した場合、項目抽出にかかる時間は発生しないが、変更要否の検討に122人時間が必要になると推定される。

このことから、トレーサビリティ管理の導入により、変更管理による工数負荷を104人時間ほど軽減できることが分かる。変更管理に要する工数負荷を1件当たりの時間に換算すると、約7人時間になる。本実験では、トレーサビリティ管理を実施するのに130人時間の工

数を必要としているため、19 件以上の変更が生じた場合には、トレーサビリティ管理導入による工数負荷を変更管理の工数軽減効果が上回ることになる。

このことから、開発プロセスを進める中で仕様変更が多く発生するような場合には、トレーサビリティ管理を導入した方が、全体の工数を低く抑えられるということが分かる。

6.6 モデルベース開発とトレーサビリティ管理の組み合わせによる相乗効果

一般的に、開発手法としてモデルベース開発を導入した場合に、モデルベース開発がトレーサビリティ管理との親和性が高いため、より高い効果が得られるとされている。

モデルベース開発の手法として本実験で取り入れたのは、システム設計工程におけるシミュレーション検証のみであったため、モデル同士のトレーサビリティ管理は実施できなかった。

モデルベース開発とトレーサビリティ管理を同時に実施した場合の相乗効果を評価するためには、一連の開発プロセスを実際に踏襲する必要があると考えられる。しかしながら本実験では、概要設計～システム設計のみに焦点を当てたため、このような観点で評価することが困難であった。今後、モデルベース開発とトレーサビリティ管理を同時に導入した場合の相乗効果についても実際に計測・分析を行い、実証される機会があることを期待したい。

7. まとめ

ソフトウェア品質説明力強化にむけた参考データ収集のための実験として、ISO26262 に準拠した形で、開発ライフサイクルの概念設計～システム設計における設計作業を実施し、新たに発生した工数負荷を測定した。

本実験では、従来型の開発プロセスに加えて、モデルベース開発を利用したシステム設計工程でのシミュレーション検証、及び設計文書のトレーサビリティ管理を実施した。

実験の結果、設計段階でのシミュレーション検証により 5 項目、2.1 人月、設計文書のトレーサビリティ管理により 4 項目、0.8 人月、合計で 9 項目 2.9 人月の作業が追加された。

モデルベース開発を利用したシステム設計工程でのシミュレーション検証により、システム設計仕様書に含まれる不具合を 16 件発見することができた。システム設計仕様書の不具合は従来型の開発プロセスにおいては、システム設計工程では見過ごされる傾向にある。一方、モデルベース開発を導入することによって、システム設計までの比較的早い段階で、一定レベルの設計品質を確保することができた。

もし、システム設計仕様書の不具合が見過ごされて、後工程で発見されたと仮定すると、作業工数の増加分は 3.5 人月と想定される。したがって、システム設計までの段階における付加工数と比較しても、開発工程全体としては工数の削減できるものと想定される。

本実験では、設計文書のトレーサビリティ管理を導入する際に、関係者間で管理方針を決定するための工数負荷が最も高かった。本実験は実際の開発に比べると小規模であることから、実際にはより多くの工数が必要になると考えられる。開発プロセスの見直しが必要になる場合も考えられるため、関係者間で十分な合意形成を行った上で、導入を進めることが肝要である。

以上により、従来型の開発プロセスに対して、ISO26262 の要求事項を加えた場合に、一部の工程に対して工数負荷があるものの、モデルベースシミュレーションツールやトレーサビリティ管理ツールを効果的に活用することで、全体としては導入効果が工数負荷を上回ることが可能であると考える。

参考文献

[1] 「ソフトウェアの品質説明力強化のための制度フレームワークに関する提案」(中間報告)（2011 年 9 月 30 日公開）http://sec.ipa.go.jp/reports/20110930.html

[2] 大藤正・小野道照・赤尾洋二（1990)『品質展開法（1）―品質表の作成と演習』日科技連出版社。

[3] 大藤正・小野道照・赤尾洋二（1994)『品質展開法（2）―技術・信頼性・コストを含めた総合的展開』日科技連出版社。

[4] ロバート・ボッシュ Gmbh（2003)『ボッシュ自動車ハンドブック　日本語版第２版』山海堂。

[5] Doug Rosenberg・Kendall Scott（2001)『ユースケース入門　ユーザマニュアルからプログラムを作る』株式会社 テクノロジックアート訳, ピアソン・エデュケーション。

[6] John X.Wang・Marvin L.Roush（2003)『リスク分析工学　FTA、FMEA、PERT、田口メソッドの活用法』日本技術士会訳, 丸善。

[7] ISO 26262, *Road vehicles –Functional safety–*