embeddedworld2021 Exhibition&Conference ... it's a smarter world DIGITAL

Designing Safety In by Extending System Modeling Languages
システムモデリングを機能安全規格で強化する

Juha-Pekka Tolvanen
MetaCase
Jyväskylä, Finland

Steven Kelly
MetaCase
Jyväskylä, Finland

概要 **— 安全なシステムの開発作業は適切な安全規格に従う必要がある。今日、システム開発にモデリング言語が採用されるが、これら言語のほとんどは安全上の懸念への十分な配慮が足りない。そこで我々は、システム要件に関連する安全面を第一級に扱うアプローチとして、システム設計の他の側面に加えて、安全性のコンセプトをサポートするモデリング言語を提示する。本論文では、安全面をシステム設計に応じて自然にモデリングできる仕組み、コラボレーション開発、および故障分析に沿った安全報告を可能にするために、車載システム向けの機能安全規格である ISO 26262 をシステムモデリング言語に追加して、メタモデルと関連ツールのサポートを拡張する方法について示す。この拡張された安全モデリングサポート機能の主な利点は、システム設計から安全設計モデルを自動生成させて、システム設計と安全設計間のトレーサビリティによりシステムエンジニアと安全エンジニア間のコラボレーションが強化されて、エラーを誘発し手間ひまのかかるマニュアル作業が、自動化された設計変換と分析に置き換えられることである。**

キーワード *— 安全性; ドメイン固有言語; ISO 26262; モデルベース開発*

I. はじめに

安全なシステムの開発は、ISO 26262[4] や SOTIF/ISO 21448[5]などの適切な安全規格に従う必要がある。今日、システム開発にモデリング言語が採用されるが、これらの言語のほとんどは安全上の懸念への十分な配慮が足りない。また安全なシステムの設計に向けてのガイドをしていない(たとえば SysML[10])。そこで我々は、関連する安全面を第一級に扱うアプローチとして、明確に安全性を目的としたコンセプトを含み、システム設計の他の側面と連動するモデリング言語を提示する。安全システムのモデリングは、システムブロック・信号・クライアント/サーバ接続だけでなく、危険・危険事象・安全目標についても含まれる。これにより、安全面が明確になり、全チーム内で可視化される。安全面はシステム開発言語の他の部分と連動するので、システムエンジニアと安全エンジニアのコラボレーションが可能になり、必要なトレーサビリティの基礎も提供する。トレーサビリティは、要件間の論理的な依存関係から、要件がシステム設計で満たされていることを評価するマッピング情報、ならびに要件から検証および検証ケースなどのマッピングまで支援される。そして安全情報をモデルに取り込むことで、フォールトツリー解析(FTA)、故障モードとその影響の解析(FMEA)、安全関連ドキュメント([1][8])の生成などが、ツールサポートと自動化で行える。

本論文では、安全性の側面をシステムモデリング言語に追加する概念とプロセスの両方を説明しながら、その方法を紹介する。このアプローチを実証するために、具体的な例として車載システム向けの ISO 26262 機能安全規格[4]を挙げ、システムモデリング言語に追加する。これと同じプロセスは他の安全規格にも適用することができる。我々は制御システムの安全関連部の規格である ISO 13849-1 にそれを適用した。

このプロセスは、まずモデリング言語のメタモデルを、安全コンセプト、命名規則や接続などのルール等で拡張することから始める。その後、新しく安全モデリングのコンセプトを表す視覚的な表記を追加する。表記を使用すると、言語を参照例でテストし、言語ユーザーに対して実証することができる。モデルの正しさを検証し、他のシステム設計とともに安全設計のトレーサビリティを提供するために、自動化ツールのサポートも追加される。ジェネレータはシステム設計から安全設計モデルを自動生成させ、また既存の解析ツールやシミュレータへの入力を生成して、FTA、FMEA、および安全性関連ドキュメントの自動作成を可能にする。

最も重要なことは、言語の拡張はインクリメンタルなプロセスで行われて、言語ユーザーが進化するモデリングサポート機能を用いて実際のシステムをモデリングして、テストし検証することである。つまり、メタモデルに言語のコンセプトを追加して使用するプロセスはシームレスである。メタモデルに表記とルールとともに安全コンセプトを追加することはわずか数分で済み、ユーザーはこれを直ぐに利用して、言語をさらに定義するためのフィードバックを提供できる。システム設計と安全設計が連動することや、トレーサビリティによりシステムエンジニアと安全エンジニア間のコラボレーションが強化されることで、エラーを誘発し手間ひまのかかるマニュアル作業が自動化された設計変換と分析に置き換えられるという、安全モデリングサポートの利点を事例で示すことで本論文を締めくくる。

II. 言語設計

今日、多くの分野で、車載システム向けの ISO 26262 などの安全規格に従うことが求められている。これらの標準規格は、通常、安全設計のルールとプロセスとともに、関連する語彙とコンセプトを定義する。ISO 26262 を例にすると、主なコンセプトは次のとおりである。

- 安全目標(Safety Goal)は、1 つ以上の危険事象のリスクを許容可能なレベルに軽減することを目的として、システムに割り当てられるトップレベルの安全要件
- 危険事象(Hazardous Event)は、車両レベルの危険と、事故を引き起こす可能性のある車両の操作状況の関連する組み合わせ
- ハザード(Hazard)は、事故の原因となる、あるシステムの状態または状況を表す
- 安全コンセプト(Safety concept)は、安全目標を達成するための手段を定義する
- アイテム(Item)は、対象となる製品がどのようなシステムであるかのスコープと、その最上位のシステム機能要件を識別する

これらのコンセプトに加えて、安全のすべての側面を考慮するためのルールとプロセスガイドラインもある。たとえば、各安全目標に対して 1 つ以上の安全な状態を定義する必要がある。安全目標は、1 つ以上の危険事象に対処する必要があり、1 つ以上の要件にリンクする必要がある。システムエンジニアと安全エンジニアは、これらのコンセプトと関連するすべての制約を習得して安全設計を行って、それに応じて(認証を得るために)文書化する必要がある。安全規格は互換性がなく、異なる意味を持ち、反対の値に従うことさえある(たとえば、安全度水準(safety integrity level)は異なる安全規格で異なった値を取る)ので、残念ながら、一般的で普遍的な安全コンセプトのセットを持つことは不可能である。多くの安全規格は、機械・鉄道・工業プロセス・自動車などの異なる規格で、業界に固有のものとして調整される。

A. 抽象構文

モデリング言語に安全性を追加するには、まず関連する安全コンセプトをメタモデルとして定義する。図 1 は、信頼性モデリング(Dependability modeling)に定義された ISO 26262 に関連するコンセプトを示す。左上は、安全目標の定義と、Safety level や SafeState(Mode オブジェクト)などのプロパティであり、また要件や対処する危険事象など他のコンセプトへの接続もある。

これらの安全コンセプトを既存のシステムモデリング言語と統合する必要がある。この例では、プロダクトライン開発のバリアントや、車載ソフトウェアアーキテクチャである AUTOSAR[1]など、車載

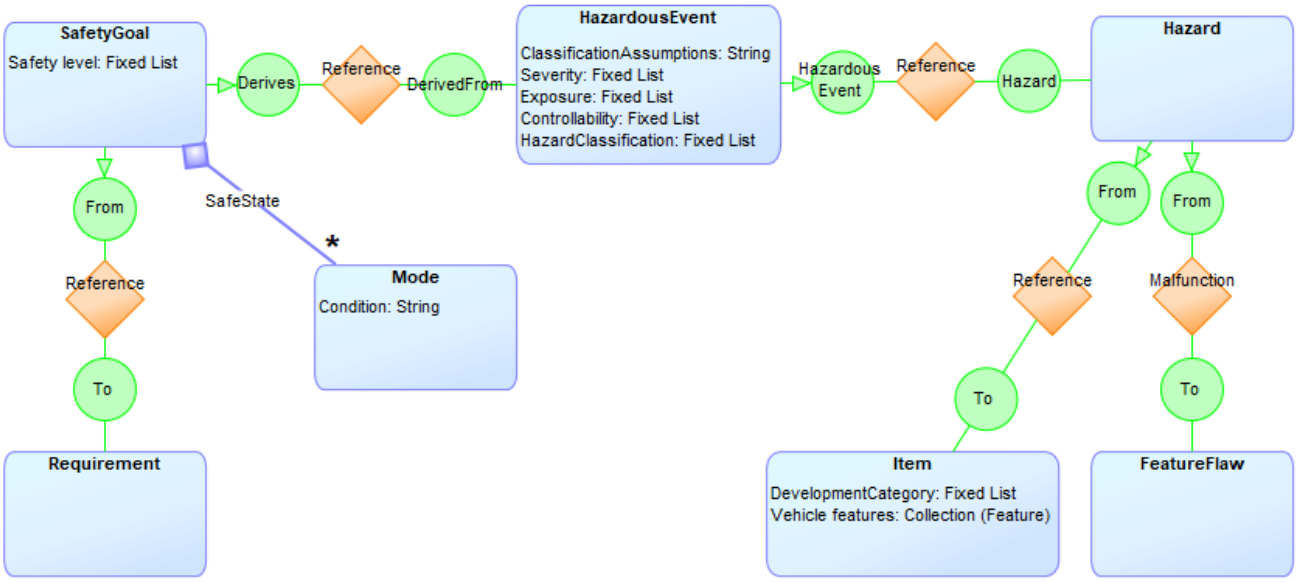

図 1.　信頼性モデリング（Dependability modeling）ISO 26262 のメタモデルとしての安全コンセプト(一部)

システムの側面をサポートするシステムモデリング言語である EAST-ADL を拡張した。ISO 26262 メタモデルに示されている要件のコンセプトは EAST-ADL の既存のモデリングコンセプトでもあり、これは再利用する。ISO 26262 のアイテムコンセプトには、フィーチャのコレクションを参照するプロパティ「Vehicle features」があり、EAST-ADL も同様のセマンティクスを持つフィーチャのコンセプトを定義しているので、それにリンクさせた。

言語拡張には MetaEdit+[6]を使用した。MetaEdit+は、直接変更・拡張しても既存のモデルを壊すことのない EAST-ADL(および UML、BPMN など)のメタモデルを既に提供しているからである。図1のような信頼性モデリング(安全コンセプト)に加えて、システムの誤った動作(たとえば、コンポーネントエラーとその伝播)を指定できるエラーモデリングのメタモデルも定義した。エラーモデリングの例を図 4 に示す。

B.　具象構文

モデルの各要素、それらの接続、または個々のプロパティなどの各モデリングコンセプトには、人手によるモデルの作成、読み取り、検証を支援する視覚的な表現がある。そのため、これらに表記記号を追加した。図 2 は MetaEdit+のシンボルエディタで定義された安全目標の表記例である。シンボルには、値を表示したり、値を計算したり、言語ユーザーにガイダンスを提供する条件付きまたは動的な部分を含むこともできる(たとえば、安全目標に対して安全状態がまだ定義されていないことを警告するなど)。安全性の様々な要素に固有の視覚化を用いると(たとえば、図 3 と図 4 を参照)、様々な種類のものをすべて同様の長方形のブロックで表すのとは対照的に、モデルが読みやすくなる[7]。

C.　ルールと制約

メタモデルには、安全エンジニアによるモデリングを支援する制約やルールも含まれる。たとえば、完全性を確保するために、安全目標がどの要件にも関連していないか、危険事象から派生していないかを、言語が自動でチェックする。また、整合性チェックとして、危険事象の名前が一意であること、その ASIL 分類が(規格にある)正当な値のみであること、が検査される。

D.　ジェネレータ

システムエンジニアリングに MBSE を採用する大きなモチベーションは、自動化により生産性を向上させ、エラーを誘発し手間ひまのかかる作業を削減することである。我々はエンジニアリング作業を支援するために、次のようないくつかのジェネレータを定義した。

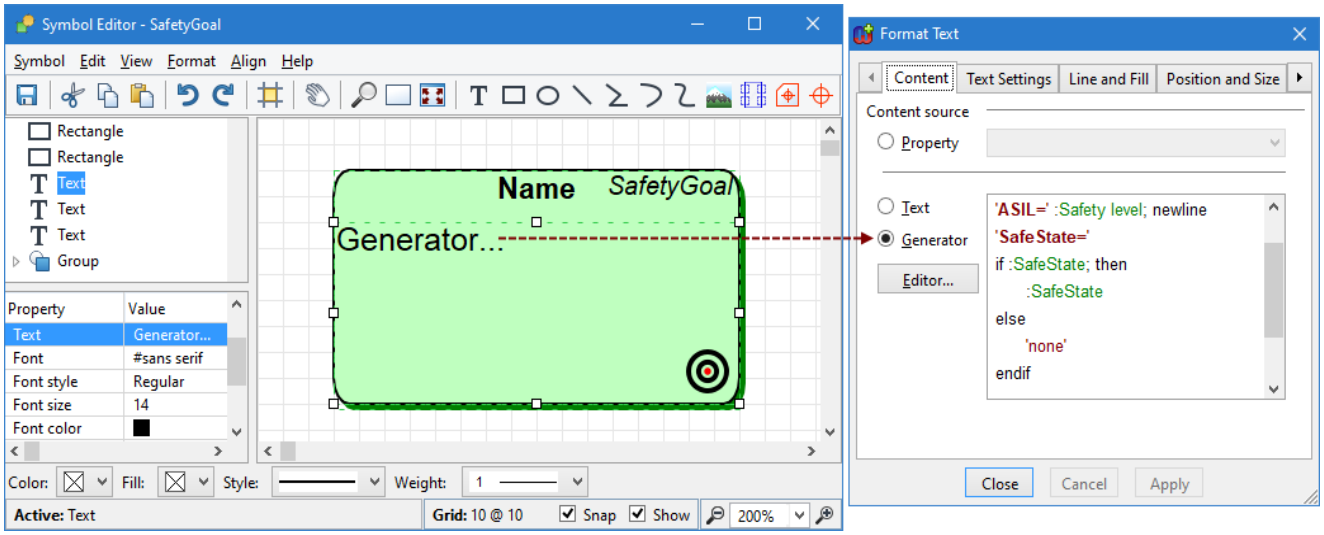

図 2.　安全目標の表記法

- まずシステムモデルから安全モデルを生成させる。事前にモデリングされたハードウェア仕様や機能仕様から直接生成させることで、安全エンジニアはより簡単に作業を開始できるし、安全設計がシステムの要件に沿うことを確実にできる。(安全設計がアーキテクチャ設計と個別に行われ、それらの同期を維持することと比較すると違いは明白である)
- システムエンジニアと安全エンジニアのコラボレーションを支援するトレーサビリティレポートの生成。MetaEdit+は、システムモデルと安全モデルのコラボレーションモデリングとバージョン管理を可能にし、トレーサビリティレポートはエンジニア専用のビューを提供する。このようなトレーサビリティデータは、作業が適正に行われたことを示すエビデンスにもなる
- FTA や FMEA などの分析ツール用のデータの生成。これにより、システム設計が変更された場合のフィードバックループが迅速化され、分析が容易になる
- 安全設計のレポート生成 – 認定用や企業固有のニーズに応じて必要とされる

E.　言語定義プロセス

言語定義プロジェクトでは、車載システム開発のエンジニアとともに、言語をインクリメンタルに実装した。その拡張は、企業内部で実施される(例：[8])か、あるいは外部のコンサルタントと一緒に行われた。産業界での実績から、MetaEdit+なら固有のニーズに対応する特定のモデリングサポートの開発工数は、平均2週間である(さまざまなケースについては[9])。

III.　結果

この安全モデリングサポートは、OEM やサプライヤーによって、モーター制御や ADAS システムなどの様々なシステムに採用されている。この事例として、ドライバーが制限速度を適切に遵守できるように設計された運転支援システムである Intelligent Speed Adaptation (ISA)の安全設計を通じて、その使用法を説明する(このシステムの詳細については[2]を参照)。図 3 は、ISA に誤った制限速度を与えて関連するハザード分析の結果を示している。この場合、2 つの危険事象が定義されている。1 つは中央分離帯がある道路での運転に関連し、もう 1 つは田舎道での運転に関連した事象である。どちらの事象でも、過酷度(Severity)、発生頻度(Exposure)、回避可能性(Controllability)、および ASIL の値は ISO 26262 に定義されている。信頼性モデルでは、ASIL レベルを許容レベル(QM)に下げる安全目標も定義されており、要件#14 (Req14)で、制限速度の変更をドライバーが確認する必要があると定義されている。図 2 で定義されている具体的な安全目標の記述が、モデリング時に使用されていることが図 3 の左下に示されている。

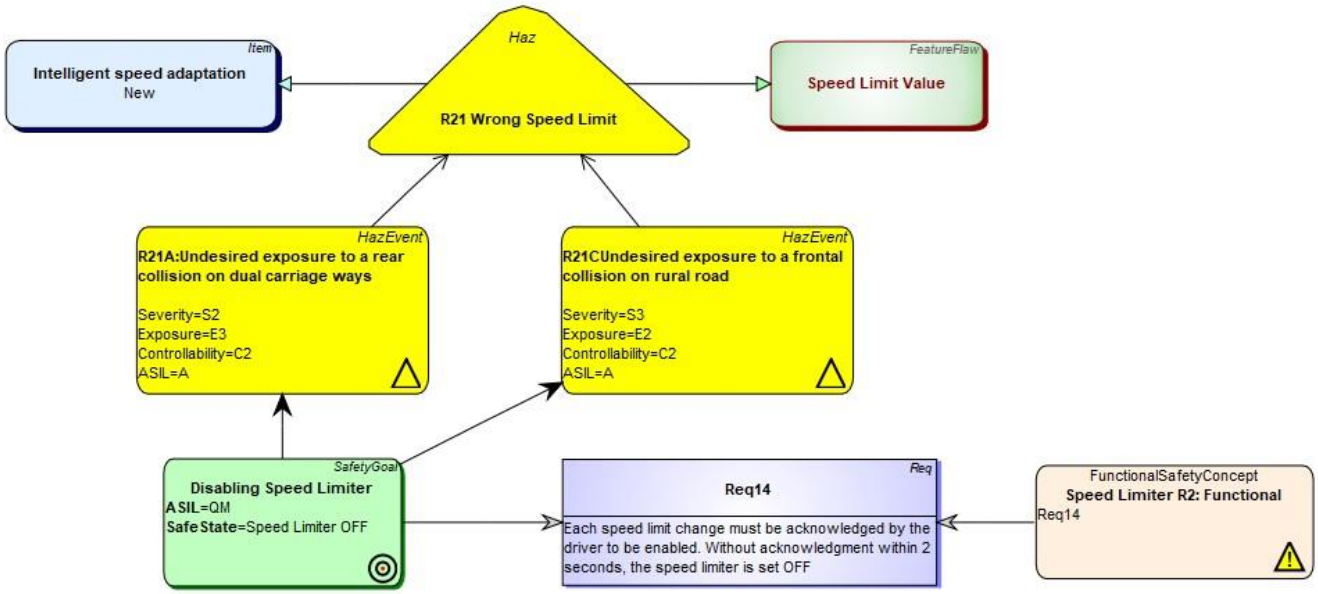

図 3.　安全性を指定する信頼性モデル

これらの信頼性モデルは他の設計と統合される。この例(図 3)では、要件(Req14)および ISA という車両フィーチャのアイテムである。このフィーチャは、プロダクトライン内の車両バリアントが選択可能な車両フィーチャモデルの一部である。このようにフィーチャにリンクすることで、特定の 1 つの車両モデルとその構成だけに限られるのではなく、安全関連フィーチャが適用されるすべての車両に与える影響をトレースすることができる。

同じことが要件にも当てはまる。エンジニアは、すべての要件からトレースして、それらが安全目標を満たしているかどうかを確認し、関連する要件とともに安全目標を報告するドキュメントを作成することができる([8]で行われたように)。また、仕様を共同でバージョン管理することもできる[6]。たとえば、安全性に基づいて提案された変更は、安全性分析に基づいて変更される関連システム設計に自動的にさかのぼる。これにより、安全システムの開発中にフィードバックと変更の反復作業が容易になる。

またエラーモデリングの基礎はシステム設計から自動生成される。図 4 の左側は、ISA システムのエラーモデルとそのコンポーネントの 1 つのエラーロジックを示す。システムコンポーネントからの車間距離の欠落は、ホイールセンサーからデータを取得しない、あるいはコンポーネントに内部エラーがある場合に発生し得る。このようなエラーモデルから、図 4 の右側(HiP-HOPS ツール[3]によって解析されたフォールトツリーの断片)に示すように、FTA や FMEA の解析を自動実行できる。現実的なシステムでは規模が大きくなるので、このようなツールによる自動解析は非常に有用である。エラーモデルは他の FTA/FMEA ツールにも同様に変換できるため、使用するツールに応じて障害率と修復率に関する情報を提供することにより、分析をさらに拡張できる。

IV.　結論

システムモデリング言語を拡張することで、各機能安全規格に専用に誂えられた支援機能が得られる。この言語主導のアプローチは、システムエンジニアリングと安全エンジニアリングのコラボレーションを強化して、安全開発(信頼性モデルとエラーモデル)がより広範にシステム設計と連動できるため、生産性と品質が向上する。また、これにより安全モデルを開発して、システム設計との同期を維持することや、FTA および FMEA の解析を実行するといった、人手に頼った、エラーが発生しやすいタスクや手順も削除される。

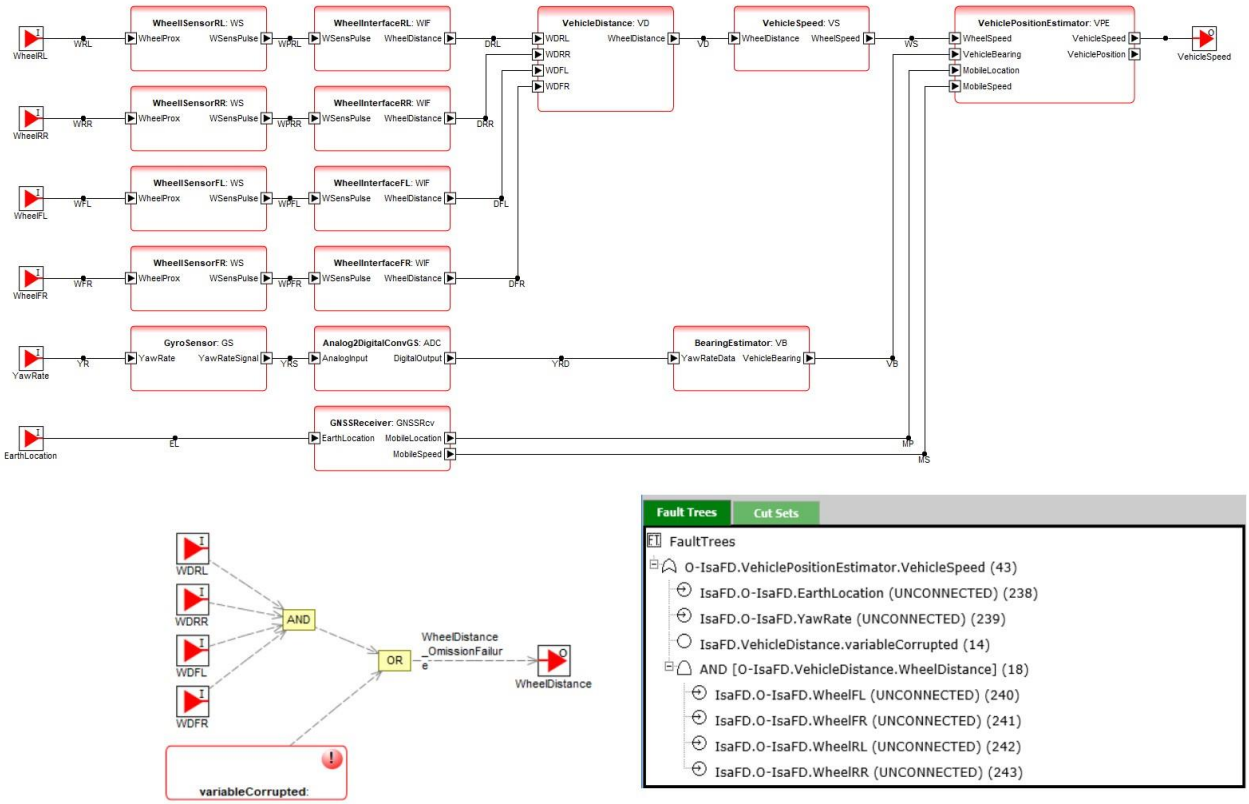

図 4.　ISA とその VehicleDistance コンポーネントのエラーモデル（関連する FTA とともに）

我々は、他の規格(ISO 13849-1 など)専用のモデリングサポート機能も作成し、また様々な分析ツールにも対応した。言語拡張の同じ原則(メタモデル、制約、表記、ジェネレータ)が同様に適用されたが、実装が異なる。たとえばジェネレータは、さまざまな分析ツールとその形式を対象とする。また、モデルには異なる安全コンセプト(SafetyMeasure など)が含まれ、加えて企業固有のレポート生成もサポートする。このようなモデリング機能や、分析ツールに対するジェネレータの開発は、MetaEdit+を使用することで通常、数週間で済むことを特筆したい。

参考文献

[1] H. Blom, D. Chen, K. Kaijser, H. Lönn, Y. Papadopoulos, M. Reiser, R.T. Kolagari, S. Tucci, “EAST-ADL: An Architecture Description Language for Automotive Software-intensive Systems in the Light of Recent use and Research”. In: International Journal of System Dynamics Applications, 2016.

[2] J. Ehrlich, J.-P. Tolvanen, “Modelling with EAST-ADL: Intelligent Speed Adaptation (ISA) as case study”, FISITA 2016 World Automotive Congress, Busan, Korea, Sept. 26-30, 2016.

[3] HiP-HOPS. http://hip-hops.co.uk/, [2020 年 9 月にアクセス]

[4] ISO Functional Safety, 26262-1, 2018.

[5] ISO, ISO/PAS 21448, Road vehicles — Safety of the intended functionality, 2019.

[6] MetaCase, MetaEdit+ User’s Guide. https://metacase.com/support/55/manuals/, 2018. [2020 年 12 月 4 日にアクセス]

[7] D. Moody, “The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering,” in IEEE Transactions on Software Engineering, vol. 35, no. 6, 2009.

[8] B. Sari, Fail-Operational Safety Architecture for ADAS/AD Systems and a Model-driven Approach for Dependent Failure Analysis. Springer, 2020.

[9] J.-P. Tolvanen, S. Kelly, “Effort Used to Create Domain-Specific Modeling Languages”. In: Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems. ACM, 2018.

[10] Omg.org, System Modeling Language, version 1.6. https://www.omg.org/spec/SysML/1.6/PDF, 2019. [2020 年 9 月 4 日にアクセス]