Safety and Security in Code

安全性とセキュリティ対策のコーディング

執筆者

Dr. Darren Buttle
ETAS GmbH
ASCET 上級プロダクトマネージャ

ESDL を用いた、よりセキュアなソフトウェアの実装

組み込みソフトウェアの分野では、プログラミング言語である C 言語が依然として実権を握っています。しかし C コードを確実に安全でセキュアなものにするのは容易なことではありません。高まりつつある車両の自律性に伴い、車載ソフトウェアの完全性への要求度はさらに高まっていきます。ETAS はこのような要求への対応として、組み込みソフトウェア開発言語「ESDL」を開発しました。ESDL は、ISO 26262 や IEC 61508 などの標準規格による制約条件を満たしながらより多くのソフトウェアをより短い時間で開発する手助けとなります。

過去 40 年以上の間、C 言語が組み込みソフトウェア開発用言語の事実上のスタンダードとして扱われています。C 言語はシンプルで軽量であるため、高速で移植性に優れ、幅広いツールに対応しています。

しかし C 言語には負の側面もあります。コード内にエラーが紛れ込みやすい、ということだけでなく、それを見つけ出すのが非常に困難である、ということです。随意的に使用される中括弧や、式における代入、switch 文における意図しないフォールスルーなど、エラーを招く要因が多く存在します。意味的に曖昧であったり複雑であったりする機能については、適切に使用するのが難しく、「危険性のあるプログラミング」（go to 文、ポインタ、整数拡張など）を促すような場合さえあります。これらにより危険な相互作用が引き起こされることも考えられます。

C プログラミングガイドライン（MISRA-C、CERT-C など）に従うことによってこれらの危険の多くを回避することはできますが、それでも C 言語プログラミングにおいてエラーが発生しやすいという傾向は、変わりません。

ガイドラインでは、バッファオーバーフローのような実行時エラーや、アンダーフロー／オーバーフロー、ゼロ除算などといった数値的問題を防ぐことはできません。さらに、制限速度を超えて加速したり、絶対零度以下に温度を下げたり、距離の値に圧力値を加算してしまう、といった機能的なエラーを解決することもできません。
C 言語はそのような情報を表現する能力が充分でないため、上述のような問題を防ぐには、静的なコード分析やテストなどの工程を設けてバグを発見し、取り除かなければならず、それでもまだ十分とは言えません。それよりも最初の段階からバグの発生を阻止する方がはるかに効率的です。

より優れた開発言語の提供

ETAS は、安全でセキュアなソフトウェアを効率的に設計できる ESDL（Embedded Software Development Language）という新しい言語により、前述のすべての問題に対処しています。ESDL は C 言語にありがちなトラブルを解消できるだけでなく、ソフトウェアの再利用が可能になるため、保守性が向上し、製品ライン内の複数のバリアントへの対応も可能です。つまり、これまで C 言語の欠点への対処に費やしていた時間を、本来の機能的問題の解決に利用することができるのです。

コードジェネレータによる C コード生成

開発工程における ESDL の効率的な利用は、ETAS ASCET-DEVELOPER 7（15 ページの記事を参照）と Eclipse ベースの統合開発環境（IDE）、および C コードジェネレータによって実現します。

IDE には、言語テンプレート、コンテンツアシスト、問題のクイックフィックスなどの先進的な編集機能が含まれているので、初心者でも ESDL を容易に習得することができます。ASCET-DEVELOPER 7 は、ESDL のプログラミング違反がないかどうかをチェックし、品質メトリクスの計算、ベストプラクティス推奨事項の提案なども継続的に行います。編集中に「オンザフライ」でフィードバックが提供されるので、コーディングミスがあってもユーザーが即時に対処することができます。

C コードジェネレータは、ESDL を MISRA 準拠の C コードに変換します。ASCET-DEVELOPER 7 は、実行時の安全性を確保する上で不可欠となる箇所にコーディングエラーをチェックするための防御的コードを自動的に挿入するので、手作業でのチェックコードの作成や保守は必要ありません。こうして生成された C コードは、既存の C 言語開発工程に容易に統合することができます。

コードの潜在的エラーを防ぐ

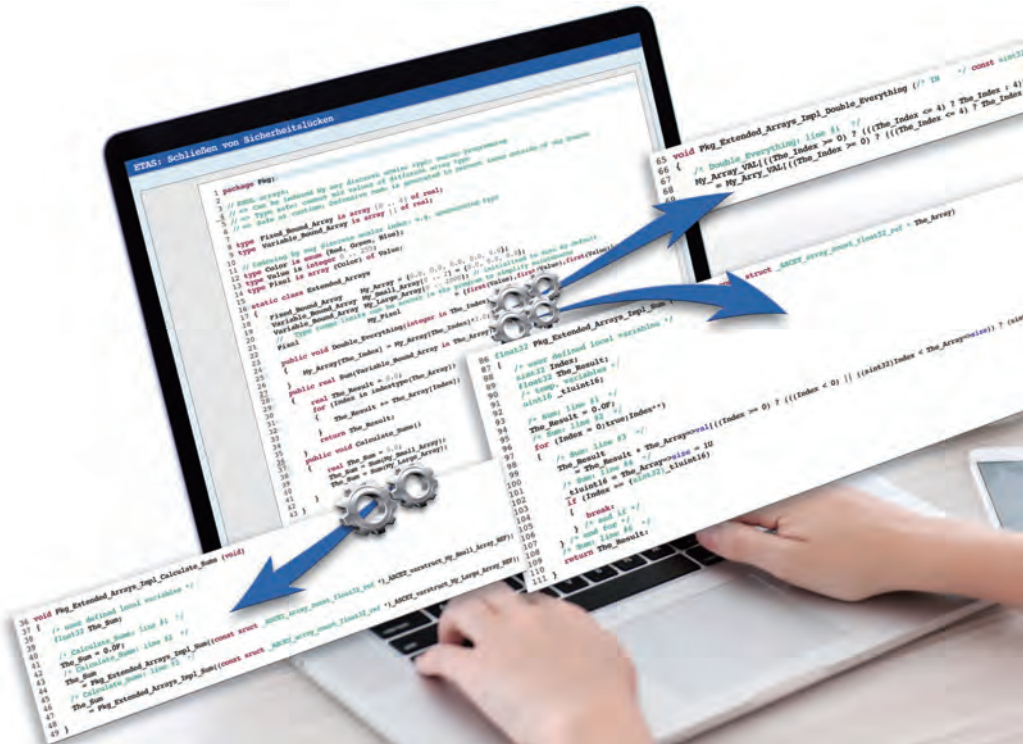
ESDL には、C プログラミングガイドラインに含まれる機能が多く組み込まれています。さらに ESDL には、ISO 26262 や IEC 61508 といった標準規格で定められている言語選択の要件を容易に満たすことができる機能が含まれています。ESDL に組み込まれたこれらのコンセプトにより、開発ツールである ASCET-DEVELOPER 7 での編集時に、従来の C 言語開発よりも多くのエラーケースをチェックすることが可能になりました。

ESDL の構文は C 言語とよく似ているので、すぐに快適に使用することができます。ただし、一般的にガイドラインによって使用を制限または禁止している危険な C 言語の構文は排除されています。ESDL には、随意的な中括弧や、ステートメントとしての式の使用、ループ変数への代入、switch 文のフォールスルー、暗黙的な整数拡張、グローバル変数、ポインタ、go to 文、共用体、といった危険要素は存在しません。そのため、本質的に C 言語よりも安全に使用することができます。

ESDL で記述された計算は、アンダーフロー、オーバーフロー、ゼロ除算、符号付き算術演算のオーバーフロー、といった数値に関する一般的な問題とは完全に無縁です。

ESDL では、境界を越えた配列アクセスが発生することはありません。多くのセキュリティレポートに見られるバッファオーバーフローも発生し得ません。

ESDL には、C 言語の typedef のように、名前を型に割り当てる拡張可能な型システムがあり、許容値の範囲と、必要に応じて分解能に関する情報を追加定義することができます。たとえば「speed」を、値の範囲が 0.0 ～ 260.0 km/h で分解能が 0.01 km/h の実数型として定義することができ、型の単位として meters（メートル）、degrees（度）、time（時間）などを使用することができます。



セキュリティギャップも、
プログラミングしながら
素早く埋めることができます。

「時間を距離に加算してしまう」といった誤りを防ぐため、単位の整合性が自動的にチェックされます。ASCET-DEVELOPER 7 のコードジェネレータは、ESDL の型情報に基づき、値を保存するのに最も適した C データ型を選択し、実行時に常に値が妥当であることを保証するための防御的チェックコードを生成します。

プログラム内の 1 カ所で変更を行えば、C コードを再生成するだけでシステム全体にその変更を適用することができます。これによってレビューや検査も容易になります。また ESDL プログラムは、実際の処理内容を捉えにくくする手書きの範囲チェックによって可読性が落ちる心配もありません。

データアクセス制御と再利用化

オブジェクトベースの言語である ESDL は、クラスを用いてデータアクセスの管理と制御を行います。オブジェクトは安全かつセキュアに使用することができ、メモリ境界も既知のものとして管理されます。ESDL には C++ や Java のような動的ストレージアロケーション機能がないので、メモリリークも発生しません。

ESDL のクラスは、複製して改変することなく、製品ラインの別のバリエーションに利用することができます。以下のようなバリエーションが可能です。

- コード
- データ初期化
- メモリアロケーション
- C データ型（浮動小数点から固定小数点へ、またはその逆の変更など）

リアルタイム環境におけるデータの整合性は、ESDL では「メッセージ」と呼ばれるスレッドセーフな通信メカニズムで実現しています。メッセージには読み手と書き手が明確に定義され、定義されていないデータアクセスは行えません。

結論

車両のコネクテッド化に対応する複雑な開発環境では、プログラミング言語である C 言語の特徴である「柔軟性」がデメリットとなる場合があります。コードに入り込みそのまま放置されたエラーが後の工程で発見されると、それを取り除くのは容易ではなく、非効率的です。C 言語で開発するエンジニアは、往々にしてこのような C 言語の不備への対処に膨大な時間を費やしている、と言えるでしょう。

ETAS は、ESDL と ASCET-DEVELOPER 7 の組み合わせにより、安全でセキュアな C コードをより効果的かつ効率的に作成できるようにしました。ESDL はあらゆる種類の潜在的エラー原因を排除し、複数のプロジェクトでのソフトウェアの再利用を容易にします。
ESDL により、組み込みソフトウェア開発の効率性、安全性、セキュリティ対策は、新しい水準に到達しました。