Safety and Security in Code

AUTHOR

Dr. Darren Buttle is Senior Product Manager ASCET at **ETAS GmbH**.

ESDL as a basis for more secure software

In the field of embedded software, the C programming language still reigns supreme. Making sure that C code is safe and secure however, is not so easy. Increased vehicle autonomy will require an even greater reliance on vehicle software integrity than today. To meet this challenge, ETAS has developed the Embedded Software Development Language (ESDL). ESDL helps software engineers meet the challenge of building more software in less time while still satisfying the constraints of ISO 26262, IEC 61508, or related standards.

Over the last forty years, C has become the de facto language for developing embedded software. C is simple, small, fast, and portable and has extensive tool support.

But C has a dark side. It is too easy for errors to creep into the code that can be extremely difficult to find. Problems start with the syntax because it makes writing code vulnerable to error. For example, optional braces, assignment in expressions, and automatic switch/case fall through, etc. Then there are semantically dubious or complex features that are difficult to use correctly and encourage "programming on the edge of safety." For example, goto statements, pointers, and integral promotion. These aspects can also interact in dangerous ways.

Using C programming guidelines, for example MISRA-C or CERT-C, helps to avoid many of these risks. Even when following guidelines, C programming remains prone to errors.

Guidelines do not prevent runtime problems like the "buffer overflow" or numeric problems like underflow/overflow and division by zero. Nor can guidelines fix problems of program meaning such as increasing a speed past a limit, reducing a temperature below absolute zero, or accidentally adding distance to a pressure.
C is not expressive enough to capture this information so preventing these problems requires additional measures like static analysis and testing to identify and remove bugs from the code. This is inefficient: it would be more effective to stop bugs from being created in the first place.

A better language for development

ETAS is rising to all these challenges with a new language to engineer safe and secure software effectively: Embedded Software Development Language (ESDL). ESDL eliminates typical C pitfalls and, in addition, enables software reuse, simplifies maintenance, and supports product-line variant engineering. ESDL enables developers to spend time solving problems instead of programming around the inadequacies of C.

Using code generation to create C

Efficient use of ESDL in development is enabled with ETAS ASCET-DEVELOPER 7 (see page 15), an Eclipse-based Integrated Development Environment (IDE) and a C code generator.

The IDE provides modern editing features like language templates, content assistance proposals and quick fixes for problems. This makes ESDL easy to learn for beginners. ASCET-DEVELOPER 7 also continually checks for ESDL programming violations, calculates quality metrics, and offers best-practice recommendations. Feedback is provided to developers "on-the-fly" during edit time, therefore reducing the time between making a coding error and its detection to zero.

The C code generator translates ESDL to MISRA-conformant C. ASCET-DEVELOPER 7 automatically adds defensive coding checks where they are essential to ensure runtime safety so they do not need to be built and maintained by hand. The generated C easily integrates into any existing C-based development process.

Securing the language against potential errors

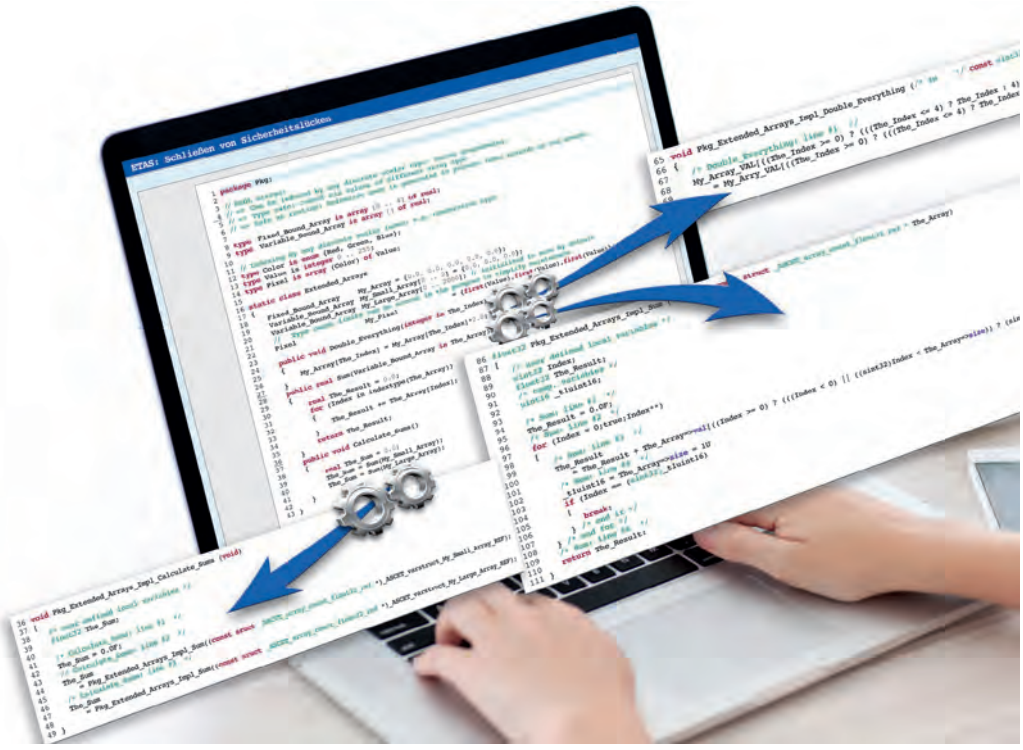
ESDL incorporates many of the aspects included in C programming guidelines into the language. Furthermore, ESDL's design includes features that make it easy to satisfy the requirements on language selection in standards like ISO 26262 and IEC 61508. Integrating these concepts into ESDL enables the ASCET-DEVELOPER 7 tools to check more error cases at editing time than is possible with classic C development.

ESDL has a similar syntax to C so that developers can feel comfortable immediately. However, ESDL removes the dangerous C features that guidelines typically restrict or forbid. ESDL has no optional braces, no use of statements as expressions, no assignment to loop variables, no automatic switch/case fall-through, no implicit integral promotion, no global variables, no pointers, no goto, no unions, etc. Removing all these pitfalls make ESDL intrinsically safer to use than C.

All calculations in ESDL are free from common numeric problems like underflow, overflow, division by zero, and signed overflow.

Out-of-bounds array access is not possible in ESDL: the common buffer overflow problem, seen in many security reports, cannot happen in ESDL.

ESDL has an extensible type system that assigns a name to a type (like a C typedef) but with additional information about what value range is allowed and (optionally) what resolution is needed. For example, in



Security gaps can be closed immediately while programming.

ESDL it is possible to define speed as a real number with a range 0.0 to 260.0 km/h and a resolution of 0.01 km/h. Types can use units, for example meters, degrees or time, etc.

Unit compatibility is checked automatically to prevent errors like adding a time to a distance. ASCET-DEVELOPER 7's code generator uses ESDL type information to select the optimal C type for storage and to generate runtime defensive coding checks to guarantee that values are always plausible.

Changes can easily be made at one location in the program and can be systematically applied by regenerating the C code. An added bonus is that review and inspection is easier. And ESDL programs are not polluted with hand written range checks that can make it difficult to understand what a program is really doing.

Controlling data access and enabling reuse

ESDL is object-based, using classes to manage and control access to data. Objects can be used safely, securely and have known memory bounds. Unlike C++ and Java, ESDL is free from memory leakage problems because there is no dynamic storage allocation.

Classes in ESDL also support product line variations without needing to "clone and own" functionality. Variation is possible for:

- Code
- Data initialization
- Memory allocation
- C storage representation (e.g., to switch between a floating-point and a fixed-point).

Data consistency in a real-time environment is provided in ESDL using a thread-safe communication mechanism called messages. Messages have a clear definition of readers and writers. This prevents undeclared access to data.

Conclusion

In the complex development environment of increasingly connected vehicles, the flexibility of the C programming language can become a disadvantage. It is too easy for errors to creep into the code unchecked and too time-consuming and inefficient to remove them later in the development process. When working with C, engineers often spend an inordinate amount of time working around the inadequacies of the C language.

With ESDL and the ASCET-DEVELOPER 7 tooling, ETAS enables to produce safe and secure C code in a more effective and efficient way. ESDL removes entire categories of potential error sources and ensures that it is easier to reuse software and the generated code between multiple projects.
With ESDL embedded software development has reached a new level of efficiency, safety, and security.