a medini™ solutions application report

Safety Analysis of C Code through Reverse Engineering

background

Software is increasingly making its way into embedded devices in mission- and safety-critical domains such as the automotive, avionics or railway industry. Assessing and managing the risk that is inherent in software realizing safety-critical control functions is a key requirement in engineering hybrid systems. This case study reports on a solution carried out to perform fault analysis and criticality analysis of existing source code (C code) for ECU’s in cars. Since software is extremely sensitive to errors and testing can never be exhaustive to detect and eliminate every harmful behavior of code, the motivation is to find failure causes by applying the quantitative risk method FTA on component configuration level. For this purpose, the manual written C code is reversely transformed into a data-flow model used for further safety analysis.

Note that the analyzed C code is typically already tested against the specification and basic quality measures have been applied such as MISRA coding style, HIL testing, and so on. Nevertheless, fault trees provide a way to manage potential hazards induced by hardware failures (bridging signals, electric radiations, unit breakdown, etc.) which can have a strong impact on failure propagation in the whole system. Therefore, such effects are indentified deductively using the FTA method.

process and approach

At the beginning of the solution development, a functional safety process was already established at the customer. Beside system level quality and safety procedures as prescribed in IEC 61508 and Automotive SPICE, an additional measure in the process is the assessment and analysis of (manually written) C code. This sub-process consists of several steps:

- Determination of the critical parts of the overall code based on an allocation and distribution model of the Software to the ECUs of the analyzed system. Main driver and input for this step are previously produced system-level safety requirements (safety specifications, SIL determinations of components, safety goals).

- In a next step, the C-code running on ECUs with higher SIL requirements is translated into a customer specific data flow graphical notation. Basically, this model helps understanding the implementation by visualizing the flow and behavior on a higher-level comparable to e.g. a MATLAB/Simulink model. The customer specific data flow models were produced manually with standard office tools.

we automate system creation – with medini™ solutions

we automate your processes with medini™

we analyze your development process activities and procedures jointly with your process experts.

we identify candidate activities which would profit from introducing automation.

we configure available components from our medini tool box and create automated tool chains, customized to your process.

you benefit from automation, consistency and artifact validation.
• For the risk analysis, the reversely derived data flow specification is analyzed using a standard FTA tool. In this step, safety experts deductively identify potential faults such as mal-functions, specification gaps, communication faults, hardware influences, etc. by creating fault trees for the scenarios described in the safety specification.

• Subsequent criticality measure during FTA analysis (Birnbaum, F-V, etc.) tries to minimize potential hazards that were detected and feed back measure proposals to the implementation and/or deployment.

Form the customers experiences, the process is sound and works out well to find hazardous parts of the implementation. However, the process is not well automated through integrated tooling and many steps have to be done manually by users, which makes it cost intensive and error prone. Moreover, during each step inconsistencies are likely to occur, especially when it comes to iterative changes and updates of the source code. The lack of automation and fine-grained traceability was the main motivation to realize a model-driven tooling using medini™ technology.

**automated solution with medini™ technology**

Model-driven engineering is centered on the notion of models. While it is intuitively clear that specification tools such as MATLAB/Simulink or SCADE support modeling, we have a broader notion of models and consider also C code as a model with textual syntax. In a nutshell, everything is a model that is defined by another model, its meta model, and we can define meta models for all artifacts used in a development process. Meta models provide a way to precisely describe and manage all artifacts in a structured and consistent way.

In the case of this solution, three types of models exist: C implementation models, data flow models and fault-trees. For each of them, we defined a meta model that clearly describes the model’s structure and relationships to other models which allowed us to build an integrated model-driven tool-chain – based on the medini™ technology – consisting of the following components:

• **C parser component** - the parser allows the import of C source code into a repository derived from a C language meta model. The resulting C implementation model makes up the bottom end of the tool-chain and is the enabler for applying modeling techniques directly onto the source code. For example, additional source validation can be performed to check for rules a compiler isn’t aware of, e.g. MISRA coding style (no pointer arithmetic, one return statement per function, etc.), company coding rules of functions/variables, and so on.
- **customized data-flow modeling tool** - the customer specific data-flow notation was captured by a meta model and a dedicated graphical editor. This integrated editor supports structured flow modeling by means of function blocks, inputs/outputs, flow branches, and so on. The editor was based on the existing medini™ flow modeler.

- **C to data-flow model transformation** – the mapping from C code to the data flow model has been a manual task previously. Now, with the model representation of C and the data flow, the task has been automated through model transformations to derive the data flow model in a fully automated manner from the C source code. For this purpose, we have applied the medini™ QVT engine that offers a high-level, declarative, rule-based language to express the bi-directional mapping from a model to a model, in this case from the C implementation model to the data-flow model. These rules are extensible and can be enhanced by users after deployment. Traces are created on-the-fly while transforming individual elements and hence full traceability between the models involved is achieved.

- **integrated FTA plug-in** – the solution integrates the medini™ fault tree plug-in. While standard analysis tools typically provide an isolated solution, the medini FTA plug-in integrates seamlessly with all other tooling components. Beside standard features for probabilistic evaluation and cut-set calculation, traceability and navigation between FTA’s and data-flow models is available. Hence, consistent management of fault trees together with design/code-model becomes reality.

- **documentation generation engine.** The tool-chain features a reporting facility to produce automated documentation from source code and analysis models. On the one hand, this generation guarantees a consistent documentation of the source code in conjunction with analysis artifacts for audits, reviews or certification. On the other hand, it supports analysts in following changes by means of change reports.

In summary, all components together constitute an integrated tool-chain to assess and analyze reverse engineered source code. Consistent management of all artifacts during iterations of the analysis process is supported, e.g. in the case source code or models change. While risk assessment and safety analysis are inherently creative human tasks, the case study shows how tedious and error-prone steps are automated by applying model-driven techniques.

*do you need more information?  
do you have questions?  
how can we automate your process?*  
**contact us at**  
[www.ikv.de](http://www.ikv.de)  
tel +49 (30) 3480 770  
email information@ikv.de