Model-Based Development of Safety-Critical Software: Safe and Efficient

Translation of “Sicherheitskritische Software entwickeln”
Published at: MEDengineering, 06/2012
Software for safety-critical systems is subject to strict requirements, and so is the way it is developed. If development is model-based, it helps developers meet these requirements.

Modern medical devices contain an increasing amount of software – for sensor control, device networking, etc. Depending on the application area, the software must meet the requirements for safety-critical applications. The standards governing software for medical devices are IEC 62304 [1] and IEC 61508 [2], which define requirements and make recommendations from the very first development phases to the finished software. Two major aspects are the early, comprehensive validation of all functional requirements and the traceability of all development steps.

**Model-Based Software Development for Control Systems**

The control functions for pacemakers, insulin pumps, blood pumps and other devices involve large quantities of software. Because of the great complexity involved and the safety requirements, manufacturers must absolutely guarantee that the device software has exactly the functions that were specified and that the functions work as defined. Thus, the specifications themselves must contain unambiguous, comprehensive function descriptions. Model-based development, in which a function is described by graphical models in MATLAB®/Simulink®/Stateflow®, is a proven method of implementing these requirements. For one thing, unlike purely textual function requirements, models leave no room for different interpretations and avoid the risk of misinterpretation. For another, models provide specifications that are executable by simulation, which allows early verification [fig. 1].

Moreover, automatic code generators like dSPACE TargetLink® can directly convert the models into source code. Changes in a specification or model can therefore be simulated, verified, and then automatically implemented as high-quality code in fast iteration steps. The model and the code remain consistent at all times. The implementation can then be easily verified in various tests with tool support. For users, this means shorter development times, more efficient workflows and high-quality code. These three benefits are so convincing that the automotive and aviation industries are increasingly or even primarily using model-based development.

**Developing Safety-Critical Software**

Safety-critical software plays a key role in medical engineering, where it has to comply with standards such as IEC 62304 and IEC 61508. Safety-critical software is also developed in numerous other industries, where model-based development has become a proven and routine procedure, since it helps users meet the requirements defined in industry-specific standards.

Another major aspect of developing safety-relevant software, in addition to correct functionality, is the traceability of requirements. It must be possible to prove, at any time, that all the function requirements have been implemented exactly as described and have been tested comprehensively.

There must be a guaranteed ability to track from requirement to implementation to associated tests and back in all steps, and even long after the end of development when the software is already in use.

Ideally, all the requirements are defined from the very beginning, for example, in requirements management tools or in Microsoft® Word or Excel® documents. To ensure traceability between the requirements and the model parts that implement them, model-based development uses bidi-
In automatic production code generation, these links even reach into the code, guaranteeing complete traceability from the requirements to the model to the code [fig. 2]. For example, hyperlinks in an HTML representation of the code lead back to the linked model part.

In addition, users can generate up-to-date status reports at any time to monitor the status and maturity of individual functions.

**Guidelines for Greater Safety**

To ensure that functions are reliable, modeling and coding guidelines should be used, as recommended in IEC 62304. These guidelines contain rules and restrictions for using languages and the blocks available in MATLAB/Simulink. MISRA C [6] is an established standard for the widely used C programming language and implements these recommendations at code level. MISRA also publishes MISRA AC TL, special guidelines for the automatic production code generator dSPACE TargetLink [7]. The focus is on development support for safety-critical software, so that possible sources of error are avoided at both the model and the code level. The modeling guidelines for TargetLink also particularly address safety-critical aspects. Other objectives are enhanced readability and efficient code generation.

**Verifying Correctness and Completeness**

Different safety classes apply to medical software depending on the application. IEC 62304 defines acceptance criteria for each class: For example, the code must comply with the applicable coding standards (see IEC 62304, section 5.5.3). In model-based development, there are guideline checking tools that automatically test whether these criteria have been met and that document the results. These tools also efficiently check large models for guideline compliance, so they can and should be used even in early stages of development. The generated source code can also be checked, for example, by the MISRA C Checker.

In addition to coding guidelines, the acceptance criteria also cover verification steps such as simulation, different kinds of tests, documentation and reviews:

- Different simulation modes for the model (model-in-the-loop (MIL) simulation), the host implementation (software-in-the-loop (SIL) simulation) and the target implementation (processor-in-the-loop (PIL) simulation). These methods [fig. 3] avoid work-intensive iterations in later development phases, and save time and money by:

  - Verifying at an early stage, by means of model simulation, that the model and requirement are correct
  - Verifying that the code and the mode are consistent, and that the code correctly represents the model’s functionality, by simulating the generated code on the host PC
  - Verifying seamless traceability for documenting the software development
  - Allowing resource requirements to be estimated at an early stage by simulating the code on the appropriate evaluation hardware

- Comparison tests, i.e., performing tests with the same test cases or test stimuli in different simulation modes and comparing them [fig. 4]. These methods reduce the time required for the necessary tests despite a high volume of tests and low development costs. These advantages result from:

  - Verifying that the code represents model behavior correctly and fulfills the requirements in terms of functionality
  - Comprehensively testing for errors with tools that generate test vectors partly automatically and that support the comparison of simulation results, automatically generated evaluation, and documentation
  - Verifying that system behavior is robust even with invalid preconditions or input values

- Code coverage tests at model and code level. These result in high-quality models and code by verifying that compared with all theoretically possible tests, the tests cases that were actually devel-
oped and executed are sufficient for covering the functional requirements.

Reviews at model or code level in which the model and the code are linked to provide quicker and more complete reviews and therefore enhanced model and code quality.

Automatically generated documentation with adjusted contents

In model-based software development, not only can the code be generated automatically from the models, the software documentation can also be generated at a click.

Specification and implementation changes are automatically included and do not have to be documented manually.

Template mechanisms and existing API interfaces enable users to adapt the form and contents of the documentation simply, quickly and individually, either as they see fit or according to formal documentation requirements. The model structure and other items such as graphical plots showing the simulation results can be added to the automatically generated documentation to represent the model’s dynamic behavior. The requirement information can also be included to show the consistency between the requirements and the model. Because the requirements are linked to the corresponding model parts and code sections, requirements management software can be used to integrate the verification results directly into a status report. The report shows whether tests were already performed and passed for each component.

Conclusion

The development of safety-critical systems imposes additional requirements and restrictions on the software development process. Experience from different industries has shown that complex, safety-critical applications can be developed successfully with model-based development and automatic production code generation. Some aspects of the model-based development processes were described as examples of how to support compliance with the requirements of safety standards such as IEC 62304 and how to automate software development steps.
© Copyright 2012, dSPACE GmbH.

All rights reserved. Written permission is required for reproduction of all or parts of this publication. The source must be stated in any such reproduction. dSPACE is continually improving its products and reserves the right to alter the specifications of the products contained within this publication at any time without notice.
dSPACE is a registered trademark of dSPACE GmbH in the United States or other countries, or both. See www.dspace.com/goto?trademarks for a list of further registered trademarks. Other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.

Company Headquarters in Germany

dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Tel.: +49 5251 1638-0
Fax: +49 5251 16198-0
info@dspace.de

China

dSPACE Mechatronic Control Technology (Shanghai) Co., Ltd.
Unit 1101-1104, 11F/L
Middle Xizang Rd. 18
Harbour Ring Plaza
200001 Shanghai
Tel.: +86 21 6391 7666
Fax: +86 21 6391 7445
infochina@dspace.com

United Kingdom

dSPACE Ltd.
Unit 87 · Beech House
Melbourn Science Park
Melbourn
Hertfordshire · SG8 6HB
Tel.: +44 1763 269 020
Fax: +44 1763 269 021
info@dspace.co.uk

France

dSPACE SARL
7 Parc Burospace
Route de Gisy
91573 Bièvres Cedex
Tel.: +33 169 355 060
Fax: +33 169 355 061
info@dspace.fr

Japan

dSPACE Japan K.K.
10F Gotenyama Trust Tower
4-7-35 Kitashinagawa
Shinagawa-ku
Tokyo 140-0001
Tel.: +81 3 5798 5460
Fax: +81 3 5798 5464
info@dspace.jp

USA and Canada

dSPACE Inc.
50131 Pontiac Trail
Wixom · MI 48393-2020
Tel.: +1 248 295 4700
Fax: +1 248 295 2950
info@dspaceinc.com

© Copyright 2012, dSPACE GmbH.

All rights reserved. Written permission is required for reproduction of all or parts of this publication. The source must be stated in any such reproduction. dSPACE is continually improving its products and reserves the right to alter the specifications of the products contained within this publication at any time without notice.
dSPACE is a registered trademark of dSPACE GmbH in the United States or other countries, or both. See www.dspace.com/goto?trademarks for a list of further registered trademarks. Other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.