白皮书

使用基于模型的设计实现 IEC 62304 合规的最佳实践

MathWorks®

简介

对医疗设备工程师来说，遵循 IEC 62304 安全标准通常涉及基于文档的需求、手动编码以及在物理设备上构建原型。

基于模型的设计可以更快速、更经济高效地为医疗设备打造高完整性软件。系统模型是中心所在，涵盖需求开发、架构分析和设定、详细设计、实现和测试。您可以通过仿真和快速原型来优化设计，然后生成代码并在嵌入式设备上实现。

本文将探讨如何在符合 IEC 62304 标准（尤其是更高安全等级的 B 级和 C 级）的过程中通过 MATLAB® 和 Simulink® 使用基于模型的设计。文中内容按照 IEC 62304 标准中确定的主要活动进行组织。

术语

本文将使用以下术语定义：

- *验证*：通过客观证据确认软件开发活动满足输出要求
- *软件单元*：设计、实现和验证的最小软件组件
- *软件项*：一个软件组件，其中可能包含一个或多个软件单元；通常，它具有中等规模，但也用于泛指从单元到软件系统的任何项
- *软件系统*：一个软件包，其中包含一个或多个软件项；它表示一个完整的功能包，但也可能只是整个医疗设备的一个组件
- *子系统*：Simulink 模型的一个分组组件，通常包含多个基本模块或固有模块

软件开发过程

软件开发规划

IEC 62304 标准第 5.1 节所述的软件开发过程从规划如何在工程期间完成主要活动和任务入手。该标准建议制定若干不同子计划，涵盖开发过程的方方面面。采用基于模型的设计的组织应在这些规划文档中引用这些工具作为开发策略的一部分。此外，将该过程中生成的工件包含在软件过程的可交付成果列表中也很重要。

可创建的一些典型项包括 MATLAB® 脚本和函数、Simulink 模型、数据字典、生成的产品级代码、S-Function 和其他用户模块库、仿真输入数据（测试向量）和结果，以及生成的文档，例如设计文档和测试报告。应考虑行业标准（如 MathWorks 汽车咨询委员会 (MAAB) 社区的 Simulink 建模规范）是否适用于正在开发的产品。此外，IEC Certification Kit 提供了一系列模型设计标准和工具，用于检查 IEC 62304 等标准的合规性。

许多组织先从这样一组模型设计标准入手，根据个人或组织的需要对其进行扩展或定制。除了模型标准之外，IEC Certification Kit 还提供参考工作流和工具验证文档，这些文档已经过 TÜV 南德的评估。研究表明，按照 IEC 62304 标准，MathWorks 基于模型的设计工具经验证适用于安全相关的开发。最后，请注意，开发工具本身应作为软件配置的一部分进行管理和版本控制。

软件需求分析

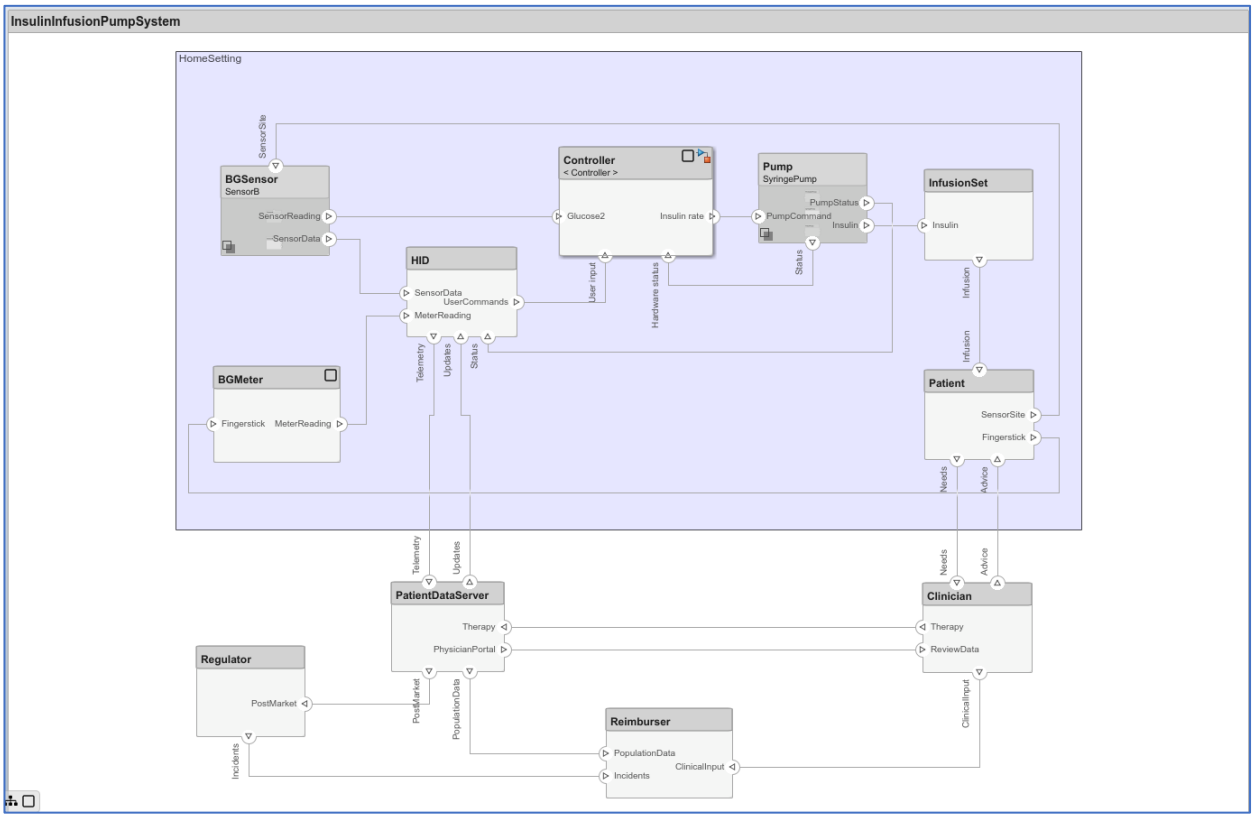
基于模型的设计具有显著的优势，它使设计者能够更好地了解系统和软件的功能需求。与分析传统的文本需求文档相比，将最初的设计理念表示为实时的可执行模型能够以更具体的方式来实现需求的一致性和正确性。除了能够对软件项的行为进行建模外，Simscape™ 系列中的 Simulink 和相关物理建模工具还可以对与项交互的环境进行建模，包括软件算法、机电组件和患者生理状况。仿真使软件系统模型能够以深入且有意义的方式与其相关操作环境进行交互，以便详述和帮助验证功能需求。此外，模型还可以基于实验、临床或分析定义的数据集进行仿真。用于推断 Simulink 模型的需求本质上是可通过仿真测试的。

Requirements Toolbox™ 提供链接功能，可用于建立需求与模型组件（如 Simulink 模块和 Stateflow® 图元素）之间的可追溯性。Simulink 生成的报告可以支持可追溯性分析。该标准要求对那些旨在保护患者以免受到伤害的软件项给予特别的关注。此类风险控制措施将具有关联的需求。如同所有需求一样，这些需求也可以链接到 Simulink 模型的元素。一种最佳做法是，使用唯一关键字标记风险控制需求，以便使用需求和报告筛选功能协助进行风险控制分析。

软件架构设计

根据该标准的第 5.3 节和附录 B.5.3，软件架构设计旨在将软件分为主要的结构组件，指示其外部属性，并展示组件之间的关系。通过 System Composer™，可以为基于模型的系统工程与软件设计定义、分析和设定架构和组合。System Composer 可用于创建系统和软件组件，指定其接口和依存关系，并将系统和安全需求分配和追溯至这些组件。

一种应用是将风险控制措施纳入一个组件或子组件，以便更直观地跟踪风险控制措施，并可能降低软件架构的其他部分的软件安全等级。



Simulink 中的系统架构的示例。控制器链接到表示软件项的 Simulink 行为模型。

Requirements Toolbox 可以将功能需求、输入和输出和其在 System Composer 模型中代表的构造相关联。查看可追溯性报告有助于支持对照软件需求验证软件架构。

一种常见情形是，从 Simulink 模型到实现的过程，例如通过 Simulink Coder™ 和 Embedded Coder® 从 Simulink 模型生成产品级代码，将仅适用于软件系统的若干部分。这种情形可能有非常实际的原因，例如存在第三方组件制造商提供的低级驱动程序代码，或希望重用来自类似产品或产品线的现有源代码。基于模型的设计支持软件单元或软件项的模块化实现。也可以完全实现表示完整软件系统设计的 Simulink 模型。

软件架构的图形化表示是一种有效的验证和表达方式。此外，若要在组件之间共享数据，可以通过构造型（本质上是某种接口的重用）来加强接口一致性。架构可以在 System Composer 中以交互方式进行审核，也可以通过查看从模型生成的文档来审核。

虽然该标准推荐审核作为主要的架构分析方法，但基于模型的设计支持通过仿真来验证软件架构是否满足功能需求。您可以将 System Composer 中指定的架构组件链接到 Simulink 中的可执行行为模型，还可以创建系统级 Simulink 仿真来执行系统架构或其子部分。大家都喜欢在软件开发过程的早期阶段进行验证，因为众多事例表明，在软件开发生命周期中及早检测到缺陷可以避免后期解决缺陷带来的高昂成本。而且在功能验证期间对模型的某些部分进行覆盖率分析（例如，修正条件/决策覆盖率）还能揭示不完整的需求或不必要的设计元素，从而使得以较低成本在开发周期的早期阶段解决这两方面的问题变为可能。

软件详细设计

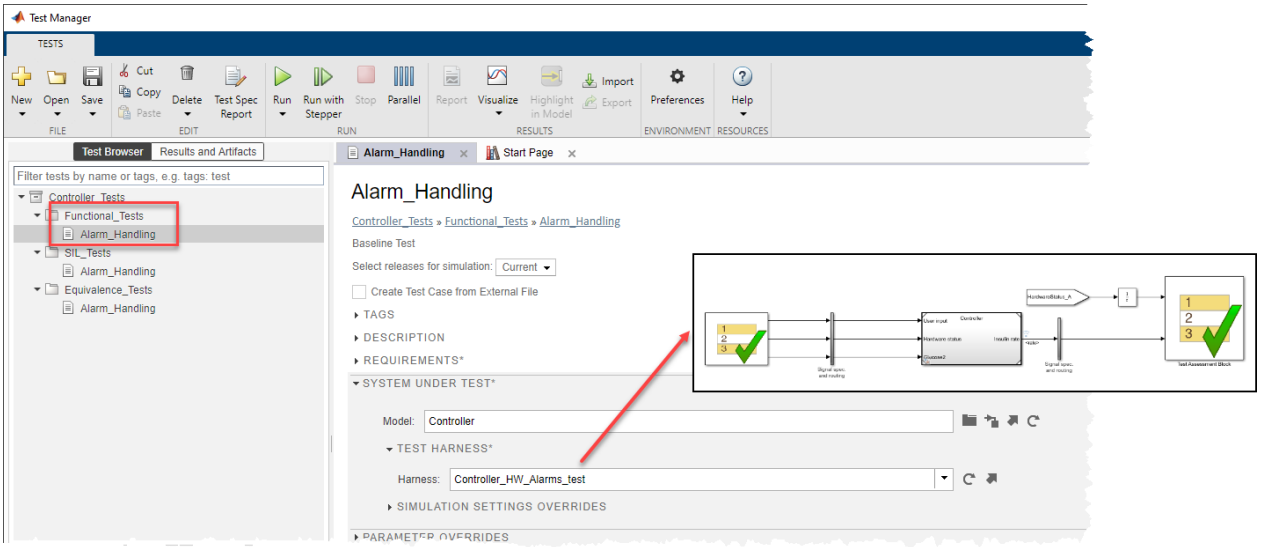
软件详细设计可以在 Simulink 中通过创建行为模型和子系统的层次结构来完成，用于详细说明软件架构。如果架构是用 System Composer 表示的，则可将这两个工具相集成，以便导入组件和接口设定，并链接架构和行为模型。这种链接支持软件系统设计验证活动，其中必须证明软件设计与架构的一致性。

软件项会进一步细分至单元级别，再次揭示了各单元的结构和属性及其之间的依存关系。软件安全等级 C 要求每个软件单元都有经过验证且有文档说明的设计。该设计必须指定支持软件实现所需的单元的各个方面。表示软件单元的 Simulink 模型或子系统可能是此详细设计的重要组成部分。该模型清楚地显示软件单元的接口以及与其他软件单元和软件项的关系，不仅通过模块图显示，而且还通过仿真功能展示。

上述软件架构的验证原则可应用于更详尽的 Simulink 模型，该模型包含满足安全等级 C 软件项要求的详细设计信息。软件单元设计验证必须证明该单元满足其需求。链接可以从模型追溯到需求文档（反之亦然），并在发生更改时将两者同步。报告和突出显示可以演示 Simulink 模型中需求链接的覆盖程度，以便为验证提供支持。

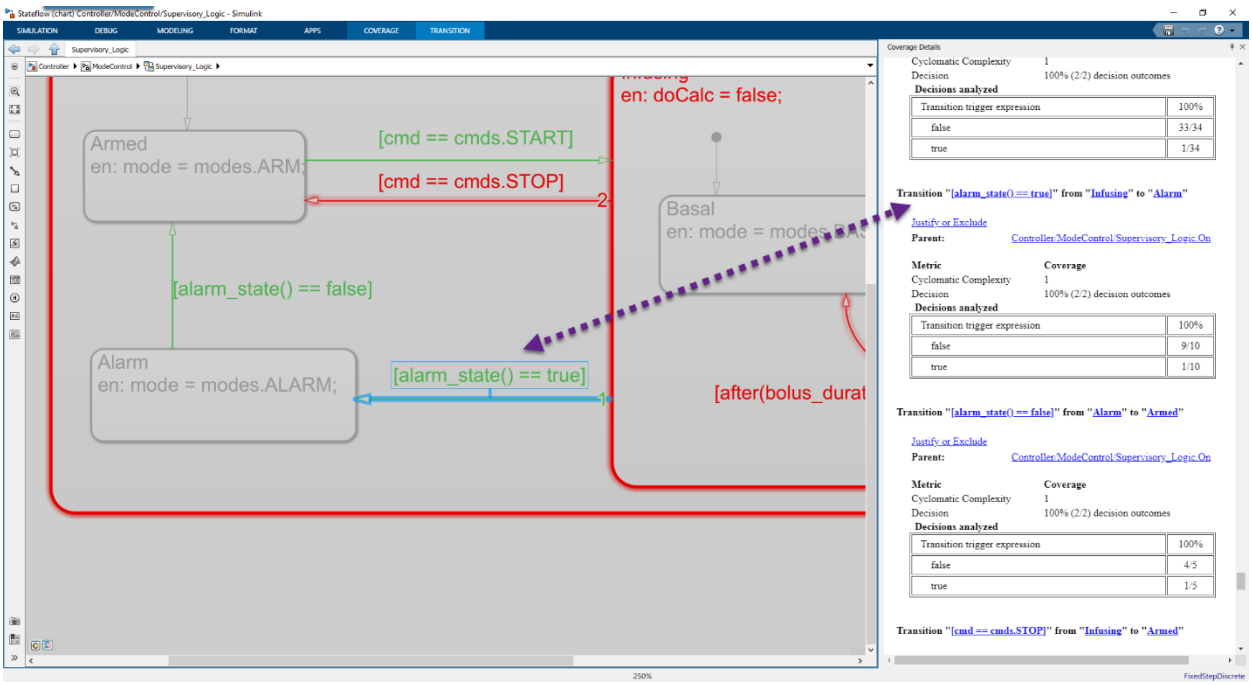
建模标准可以协助分析单元设计。Simulink Check™ 提供的选项之一是将模型与行业标准（如 IEC 62304 和 MISRA-C）进行比较。各组织可以自定义和扩充此自动检查列表。

软件单元设计验证可以包括通过模型仿真执行的功能验证。这通常是通过创建测试框架模型来实现的。该测试框架引用软件单元模型或包含软件单元子系统的库链接。测试框架模型可以配置为运行各种单元测试场景，以各种可能的方式表达。Simulink Test™ 是一种用于管理和执行测试场景并提供预期结果的工具。这些测试用例可与相关系统和软件需求关联。



Simulink Test 通过调用测试框架模型来执行设计模型，从而实现警报处理测试用例。在框架中。Test Sequence 模块提供测试向量输入，而 Test Assessment 模块检查输出。

Simulink Coverage™ 可用于为模型插桩，以在仿真期间收集执行、数据范围以及各类逻辑决策和分支覆盖率，从而评估单元测试场景的完整性。



模型，带有高亮显示的覆盖率结果。左侧的行为（Stateflow 图）显示已检测到警报，右侧的 HTML 报告是摘要工件。嵌入的超链接提供到受测模型的可追溯性。

虽然原则上基于需求的功能测试用例应在执行期间提供高模型覆盖率，以作为单元测试计划的一部分，但实则情况并非总是如此。一个示例是设计的防御性部分（例如输入范围检查）可能不会通过基于需求的功能测试来执行。Simulink Design Verifier™ 可以帮助基于模型结构派生额外的测试用例。其特点之一是使用形式化模型分析方法来生成测试用例，以实现模型的覆盖率目标，例如 100% 的修正条件和决策覆盖率 (MC/DC)，或表明由于设计的限制而无法实现这样的覆盖率目标。用户可以使用这些额外的测试用例了解和记录缺失的派生软件需求和/或修改设计，以提高其可测试性。

软件单元实现和验证

Simulink Coder 和 Embedded Coder 可用于将 Simulink 模型转换为使用 C 或 C++ 编程语言的软件实现。软件实现的细节是从精心设计的模型中表达的详细设计推知的，并且软件架构在代码接口以及打包为功能模块和基于文件的代码模块方面完全遵循设计。代码生成工具可以提供模型元素与代码之间的双向可追溯性，还可以包括链接到模型元素的需求描述作为注释。这为管理和跟踪此类功能特性和风险控制措施的实现奠定了基础。

向软件的转换可能会揭示额外的派生需求，例如单元运行时性能约束和现有代码接口，或组织编码标准合规性。这些项可通过修改详细设计模型和迭代生成代码来处理。当单元验证成功时，该过程即已完成。

单元验证的一个重要方面是功能验证。为对软件单元的 Simulink 模型执行功能详细设计验证而创建的模型测试场景和框架可用作执行实现功能单元验证的基线。生成的代码可以在主机（即开发人员用于设计和创建软件的计算机）上编译，并作为测试框架模型中的 S-Function 调用，以便使用相同的测试向量进行直接的模型与代码比较，也就是所谓的软件在环 (SIL) 测试。此外，还可以通过适用于 Embedded Coder 的目标支持包对目标编译代码执行此类测试。这种处理器在环 (PIL) 方法可用于帮助验证可执行软件是否具有与 Simulink 模型相同的行为，即使它是作为最终设备的一部分来执行的也是如此。代码覆盖率应作为测试过程的一部分捕获，以证明没有引入非预期的功能。Simulink Coverage 支持在 SIL 仿真中对生成的代码进行插桩。

对于安全等级为 C 的软件单元，软件单元验收还包括其他标准。有些标准，如事件序列、数据和控制流以及变量的初始化，用详细设计的 Simulink 模型来阐述非常自然。这样的例子包括基于数据可用性与函数调用触发执行的自动模块排序、显式信号流与数据存储内存访问、以及模型中信号和状态的显式或默认初始化选项。由于使用模型覆盖率来衡量测试的充分性，软件中这些方面的测试将通过创建功能测试场景来执行。前面隐约提到过，实现的运行时性能或编码样式方面的要求在模型中的表示可能比较微妙，因此可能需要迭代的模型更新和代码生成来满足标准。Simulink Check 提供模型顾问检查，以确保符合 MISRA C:2012 等建模标准，而 Polyspace® 产品可用于分析生成的或手写的 C/C++ 代码的稳健性和关键运行时错误。其中两项主要功能是根据行业标准（例如 MISRA 和 JSF++）对代码进行静态分析，以及通过形式化软件分析方法检测潜在的运行时错误。

功能单元验证活动的结果可通过 Simulink Report Generator™ 捕获，并添加到整个单元验证文档包中，即法规合规性的技术文件。

软件集成和测试

在基于模型的设计中，软件集成过程可能有两个不同阶段。从单元到项或自下而上的角度讲，第一个是在 Simulink 中执行集成任务的阶段。模型层次结构可用于使用经过验证的单元或其他项组合精心设计的软件项，而代码可在所选层次结构中的任何级别生成。Simulink 语义可指定接口、执行速率和顺序以及调用树。从更大项或者整个架构模型中生成代码时，这些语义会被翻译。上面针对软件单元测试所述的所有验证、测试和文档功能都可以在模型层次结构的更高级别应用。另请注意，在增量集成期间，可以为回归测试创建和管理测试框架模型和 Simulink Test 过程。建议将这种迭代测试自动化，以提高其可重复性。

通常，从一个或多个软件项的模型生成的集成代码将进一步与其他软件集成，以完成软件系统。虽然基于模型的设计环境的好处不适用于此过程，但考虑到代码的来源，此集成阶段不需要采取特别的措施，可以遵循任何符合 IEC 62304 标准的集成计划。例如，可用持续集成等工具和方法形成通过使用基于模型的设计生成的软件来构建外部开发软件的机制。

对于使用基于模型的设计开发的项，Simulink 模型的历史记录功能可以成为解决软件问题过程的一部分。每次保存模型时，用户都可以提供更改的理由。这些条目可以引用软件问题解决系统中跟踪的项。

软件系统测试

到了开发过程的这一阶段，基于模型的设计工具已提供了解需求完整性和正确性的重要信息、根据这些需求进行设计和编码所需的测试用例，以及帮助演示软件系统中软件项的单元级和集成级验证的文档。不过，系统测试很可能将按照该标准第 5.7 节的要求，并遵循组织典型的做法，在基于模型的设计环境之外执行。

在系统测试期间，可以使用系统环境的 Simulink 模型来为软件提供激励。这种方法通常是在专用确定性硬件和软件系统的协助下进行的，又称为硬件在环 (HIL) 测试。根据所涉及的数学的复杂性，HIL 系统通常可以实现闭环控制性能评估系统环境的实时仿真。Simulink Real-Time™ 可用于实现 HIL 测试。

软件维护过程

基于模型的设计通过仿真受影响的软件项来帮助评估建议的软件更改所带来的影响。此外，模型是重要的设计工件，可以帮助评估建议的更改对各种软件项的耦合性。一旦做出实施更改的决定，就要按照上述软件开发过程的相关建议执行组织作为软件维护计划的一部分选择的适当活动。

软件风险管理

基于模型的设计主要通过将需求链接到模型元素的功能来支持软件风险管理，如前所述。借助此链接和 Requirements Toolbox 的报告功能，工程师可以从记录的风险追溯到模型（设计）中的控制措施，再到生成的软件项中这些措施的实现，最后到用于评估其有效性的仿真测试。这种可追溯性通过注释可扩展到生成的软件代码行，并可使用代码生成 HTML 报告以及 Simulink 和 Embedded Coder 中的“导航到代码”功能导航。Embedded Coder 还可以生成详细的模块到代码可追溯性报告作为额外的证据。

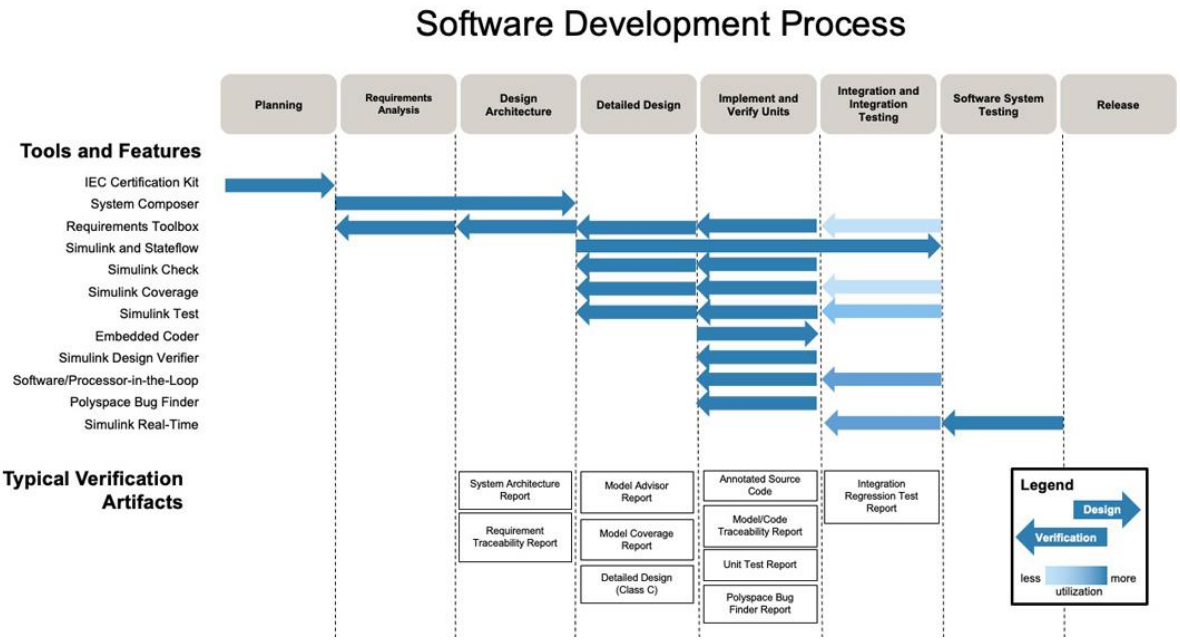
MathWorks 会发布其工具重要已知问题的列表。这些外部错误报告可以在工程期间查看，必要的项可以作为异常列表的一部分进行管理。

软件配置管理

在软件开发过程中创建的模型和数据必须与生成的软件项一起识别和管理。它们会成为软件设计历史记录的一部分，在重新创建或维护软件时需要用到。在开发活动期间创建的一些典型工件包括：Simulink 模型、MATLAB 脚本和函数、数据字典、生成的产品级代码、S-Function 和其他用户模块库、仿真输入数据（测试向量）和结果，以及生成的文档，如设计文档和测试结果。来自配置管理系统的信息，如版本号、保存日期和作者，可以作为信息性文本纳入 Simulink 模型，显示在 Model Information 模块中。

小结

您可以使用 MathWorks 基于模型的设计工具来遵循 IEC 62304 的软件生命周期过程要求。在软件开发和维护过程中，Simulink 和相关工具享有极高的价值，因为它们在完成需求分析、软件设计、单元实现和测试等关键活动方面的能力远远超过书面方法。过程工件（如需求文档）和模型之间的链接支持使得医疗设备制造商的风险分析和问题解决过程与系统和软件模型之间的紧密同步成为可能。借助基于模型的设计工具，可以对软件及其开发过程进行文档化。以下摘要图显示了这些工具的一些常见使用场景，以及它们可以生成的典型文档工件。



基于模型的设计工具对符合 IEC 62304 标准的软件开发的适用性。

详细了解 MATLAB 和 Simulink 在医疗设备领域的应用：

mathworks.com/solutions/medical-devices.html

参考文献

1. “Caterpillar Automatic Code Generation,” Jeffrey M. Thate (Caterpillar), Larry E. Kendrick (Caterpillar), and Siva Nadarajah (MathWorks), Proceedings of the Society of Automotive Engineers World Congress 2004 (2004-01-0894)
2. “Rapid Deployment of Aerospace Flight Controls,” Edward L. Burnett (Lockheed-Martin Aeronautics), 2006, http://www.mathworks.com/programs/techkits/pcg_tech_kits.html
3. “GM Powertrain Automatic Code Generation Process,” 2005, http://www.mathworks.com/programs/techkits/pcg_tech_kits.html
4. “Medrad Ensures Safety of MRI Vascular Injection Pump Using MathWorks Tools,” 2004, http://www.mathworks.com/company/user_stories/userstory6313.html
5. “Alstom Generates Production Code for Safety-Critical Power Converter Control Systems,” 2005, http://www.mathworks.com/company/user_stories/userstory10591.html
6. “Achieving Six Sigma Software Quality Through the Use of Automatic Code Generation,” Bill Potter (Honeywell International), 2005, http://www.mathworks.com/programs/techkits/pcg_tech_kits.html
7. IEC 62304, “Medical Device Software – Software Life Cycle Processes,” International Electrotechnical Commission, Edition 1.0b, 2006
8. “Control Algorithm Modeling Guidelines Using MATLAB, Simulink, and Stateflow,” MathWorks Automotive Advisory Board - Version 2.1, 2007
9. “Weinmann Develops Life-Saving Transport Ventilator Using Model-Based Design,” https://www.mathworks.com/company/user_stories/weinmann-develops-life-saving-transport-ventilator-using-model-based-design.html, 2013
10. Solis-Lemus J A, Costar E, Doorly D, Kerrigan E C, Kennedy C H, Tait F, Niederer S, Vincent P, and Williams S, “A Simulated Single Ventilator / Dual Patient Ventilation Strategy for Acute Respiratory Distress Syndrome During the COVID-19 Pandemic,” Royal Society Open Science 7:200585, August 2020
11. *Software Requirements*, Karl Weigers, 2nd Edition, 2003, Microsoft Press
12. “Understanding and Controlling Software Costs,” Barry W. Boehm and Philip N. Papaccio, 1988, IEEE Transactions on Software Engineering 14(10), pages 1462-1476
13. MISRA C:2012.The Motor Industry Software Reliability Association:Guidelines for the use of the C language in critical systems, 2012
14. MISRA C Compliance for C code generated by Real-Time Workshop Embedded Coder can be found at http://www.mathworks.com/support/solutions/en/data/1-1IFP0W/?solution=1-1IFP0W
15. “Certify embedded systems developed using Simulink and Polyspace products to IEC 61508 and ISO 26262,” http://www.mathworks.com/products/iec-61508/
16. “Sound Verification Techniques for Developing High-Integrity Medical Device Software,” Jay Abraham (MathWorks), Paul Jones (FDA/CDRH), and Raoul Jetley (FDA/CDRH), Proceedings of the Embedded Systems Conference, San Jose, CA, 2009
17. The MathWorks external bug report database can be accessed at http://www.mathworks.com/support/bugreports
18. “Configuration Management of the Model-Based Design Process,” Gavin Walker (MathWorks), Jonathan Friedman (MathWorks), and Rob Aberg (MathWorks), Proceedings of the Society of Automotive Engineers World Congress 2007 (2007-01-1775)