Uso del diseño basado en modelos para probar y verificar el software embebido de automoción

Por, Jim Tung. The MathWorks

The MathWorks
www.thematworks.es

El software embebido y la electrónica representan una parte cada vez más importante del contenido de ingeniería de un automóvil. Se prevé que en el año 2010, la electrónica suponga el 40% del coste total de los materiales de un coche, mientras que en 1970 sólo representaba el 10% [1]. El software embebido y la electrónica se utilizan en las áreas funcionales del coche para sustituir o simplificar los sistemas hidráulicos o mecánicos en funciones principales como el sistema de frenos o la dirección. También se usan para implementar funciones avanzadas, como sistemas de seguridad activa y de información al conductor que suponen un valor añadido para el cliente en cuanto a comodidad, seguridad y servicios.

Sin embargo, el software embebido y la electrónica son cada vez más complejos y esto, junto con la dificultad que conlleva del probar y verificar estos sistemas, se ha asociado a un aumento del número de llamadas a revisión y de problemas relacionados con la calidad. Según IBM Corporation, los fabricantes de coches emplean entre dos y tres mil millones de dólares al año en solucionar problemas de software y el 32% de las reclamaciones en garantía de los coches en Estados Unidos se deben a problemas relacionados con la electrónica o el software.

El diseño basado en modelos se ha convertido en la mejor opción para el desarrollo de software embebido de automoción, ya que mejora las fases de especificación, diseño e implementación. En la actualidad, existen nuevas herramientas y funciones para el diseño basado en modelos que ayudan a afrontar el reto que supone la realización de pruebas y verificaciones, y a la vez, mejoran la calidad del software embebido y acortan el ciclo de pruebas y verificaciones.

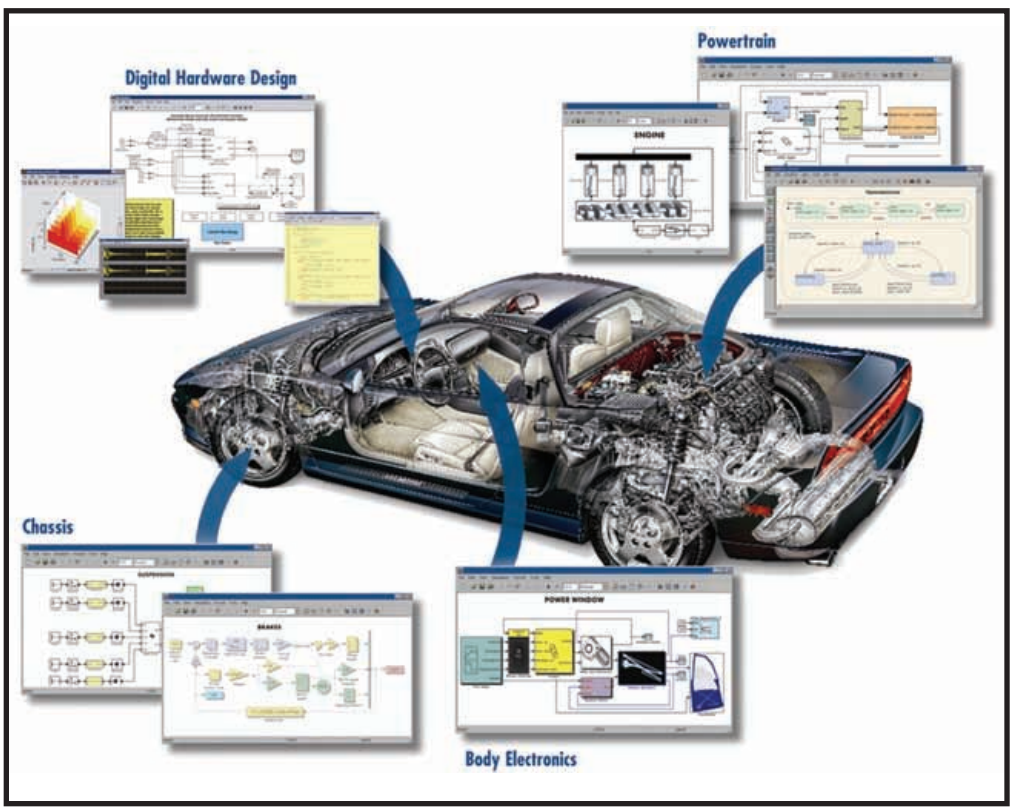

Figura 1. Con el crecimiento continuo de la electrónica y el software embebido en los automóviles, el diseño basado en modelos ayuda a reducir el tiempo de desarrollo y a mejorar la calidad en todos los dominios de la electrónica del automóvil.

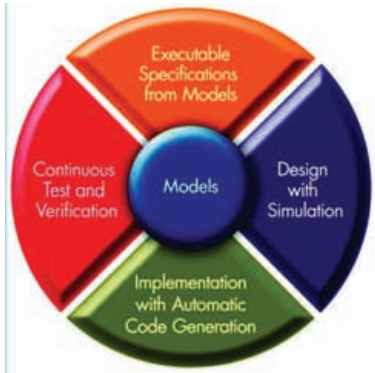

Figura 2. El sistema basado en modelos utiliza los modelos del sistema a través de todo el proceso de desarrollo, desde la captura de requisitos y el diseño a la implementación y verificación.

Diseño basado en modelos para la realización de pruebas y verificación

Los ingenieros de automoción suelen usar modelos para desarrollar funciones que se ejecutan en una unidad de control electrónica (ECU, del inglés Electronic Control Unit). Los modelos de este diseño basado en modelos se usan con diferentes fines: para proporcionar especificaciones ejecutables, para analizar el comportamiento dinámico del sistema, para simular los componentes y las condiciones medioambientales, reduciendo o eliminando la necesidad de costosos prototipos físicos, y para diseñar algoritmos. Además, la generación automática de código a partir de estos modelos se ha afianzado como una forma de implementación del software de producción de las ECU. Se espera que en los próximos años se convierta en la opción principal para la implementación de software de control embebido en muchas empresas de automoción.

Desde una perspectiva de calidad, la generación automática de código ya es de gran ayuda, porque optimiza el diseño mediante el análisis y la simulación y garantiza la repetibilidad de la calidad del código generado automáticamente. Además, el código se puede generar desde los modelos de componentes del sistema y de configuración para generar pruebas de hardware-in-the-loop (HIL).

En la actualidad, gracias a enfoques nuevos y mejorados, se puede sacar el máximo partido a estos mismos modelos con utilidades más potentes y variadas, entre las que se incluyen pruebas basadas en requisitos, análisis de cobertura y generación de pruebas, para acelerar y mejorar las actividades de prueba y verificación de los complejos sistemas embebidos actuales, tanto del software embebido como de los componentes electrónicos.

Traza y comprobación de requisitos

En la mayoría de los estándares de procesos y software, como CMM-I, es necesario que la trazabilidad entre el diseño y los requisitos sea bidireccional durante todo el desarrollo. El código de implementación también ha de ser trazable con respecto al diseño, para que pueda ser revisado y verificado. Las herramientas de diseño basado en modelos pueden vincular el texto de los requisitos, guardado en Excel, Word o en una herramienta de gestión de requisitos, con el modelo (el diseño). Cualquier requisito que no se cumpla en el diseño puede marcarse, de la misma manera que pueden marcarse los elementos del diseño que no se correspondan con ningún requisito. Si se cambia un requisito (o diseño), las herramientas indican los elementos del diseño correspondientes (o los requisitos) que pueden verse afectados. Además, el generador automático de código puede insertar enlaces HTML en el código C generado de manera que la implementación pueda ser trazada en el modelo, algo muy importante sobre todo en aquellos sistemas en los que la seguridad es vital. El conjunto de estas funciones ofrece una ruta de trazabilidad completa desde el código a los requisitos (figura 3).

Además de vincular los requisitos al diseño, las herramientas del diseño basado en modelos ofrecen funciones que sirven para confirmar que el diseño cumple ciertos requisitos. Estos requisitos se introducen en el modelo del diseño como propiedades (o asertos), y los algoritmos de métodos formales determinan matemáticamente si dichas propiedades se cumplen en todas las situaciones válidas. Si existe un escenario que pueda infringir las propiedades, la herramienta de métodos formales genera un contraejemplo que el ingeniero puede usar en simulaciones y añadir al plan de pruebas.

Comprobación del estilo de los modelos

Durante la fase de diseño, un modelo inicial de alto nivel se transforma en un modelo adecuado para su implementación, ya que se le incluyen las características necesarias para la implementación, como son datos de coma fija, y se eliminan las partes que no se implementarán. De la misma manera que los ingenieros de software establecen directrices sobre el estilo del código fuente para que sea más fácil su lectura, prueba e implementación, los ingenieros que trabajan con el diseño basado en modelos, establecen directrices sobre el estilo de los modelos para garantizar que el modelo pueda ser implementado y facilitar también su comprensión y prueba.

En función del flujo de trabajo deseado y de si el diseño representa una nueva función o una modificación de una función existente, se puede proceder de dos maneras: limitar las opciones disponibles desde el principio al diseñador, o realizar comprobaciones en el diseño más adelante, a medida que se va transformando.

En el primer enfoque, usado con frecuencia en sistemas en los que la seguridad es vital o en diseños que son pequeñas modificaciones de una implementación ya existente, las restricciones se aplican al principio del proyecto. Generalmente, esta estrategia se basa en la definición de un subconjunto restringido de componentes del modelo (bloques o máquinas de estado), construcciones de modelado (la manera en que los bloques están conectados), y otros ajustes (p. ej. aspectos de coma fija) que no necesitan ser comprobados manualmente en cada diseño porque las herramientas pueden hacer dicha comprobación sistemáticamente.

En el segundo enfoque, se comprueba si el modelo cumple las directrices más adelante en el proceso de desarrollo. El hecho de que la comprobación sea posterior permite que el diseñador comience con un diseño sin restricciones y, luego, se vaya limitando y transformando gradualmente el diseño hasta conseguir una implementación sólida y lista para la realización de pruebas. El lugar ideal para la comprobación de este modelo es el propio entorno del modelado, para que el desarrollador pueda detectar rápidamente los problemas y editar el diseño de una manera eficiente e interactiva.

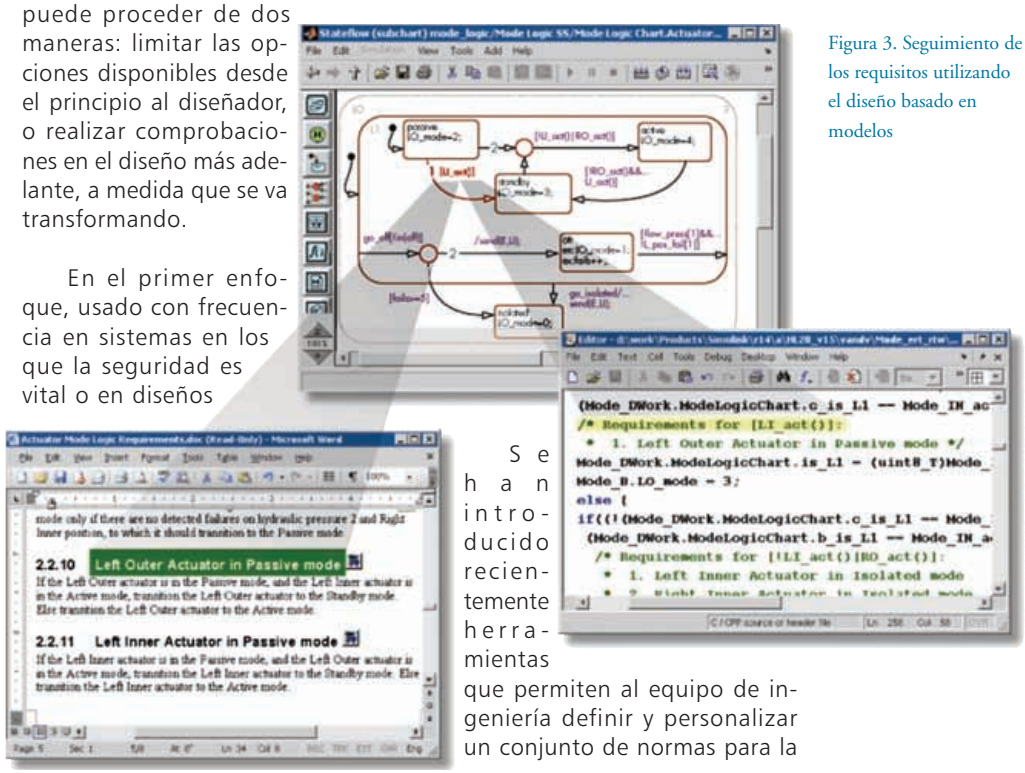

Figura 3. Seguimiento de los requisitos utilizando el diseño basado en modelos

Se han introducido recientemente herramientas que permiten al equipo de ingeniería definir y personalizar un conjunto de normas para la

Figura 4. Comprobación del modelo para verificar que se cumplen las directrices de los estándares y los definidos por el cliente.

comprobación de modelos, aplicarlas a los modelos y localizar inmediatamente las excepciones (Figura 4).

Análisis y garantía de la cobertura del modelo

El modelo ofrece la oportunidad en las primeras etapas del proceso de diseño, antes de la implementación, de llevar a cabo los tipos de pruebas que se realizarían después con la comprobación basada en el código fuente. Los ingenieros realizan pruebas de esfuerzo del controlador para verificar la integridad del diseño y detectar problemas tales como secciones inalcanzables del diseño que aparecerían más tarde como código muerto. Los ingenieros realizan a menudo simulaciones muy exhaustivas de los modelos para confirmar que un diseño es sólido y funciona correctamente. No obstante, estas simulaciones serán tan útiles como los escenarios que ejecuten.

La realización de pruebas de esfuerzo mediante la ejecución de simulaciones con valores numéricos máximos y mínimos ayuda a evitar que se produzcan condiciones de desbordamiento. También es importante comprobar que durante las simulaciones se prueben todas las partes del diseño, y todos los modos y ramificaciones lógicas del comportamiento del diseño.

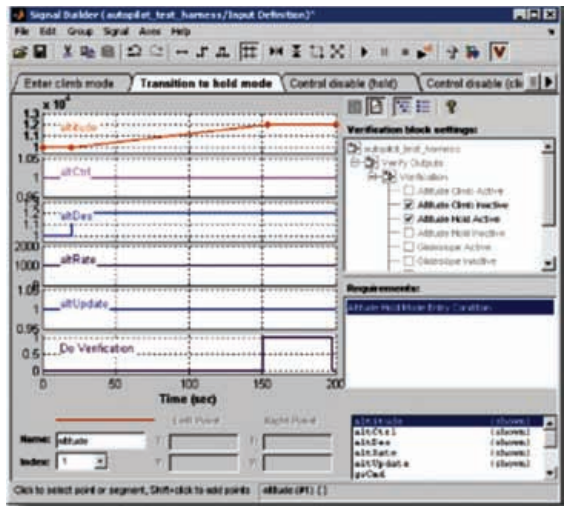

Figura 6: Patrones de prueba para uso en simulación y bancos de pruebas.

En el análisis de cobertura del modelo se evalúan los resultados acumulados de un conjunto de pruebas para determinar los bloques o estados que fueron ejecutados durante una simulación y los que no lo fueron. Algunos análisis de cobertura están bien establecidos en los lenguajes de código fuente, como C, C++ y Ada, pero estos tipos de análisis no han estado disponibles para los modelos hasta hace poco. [2]

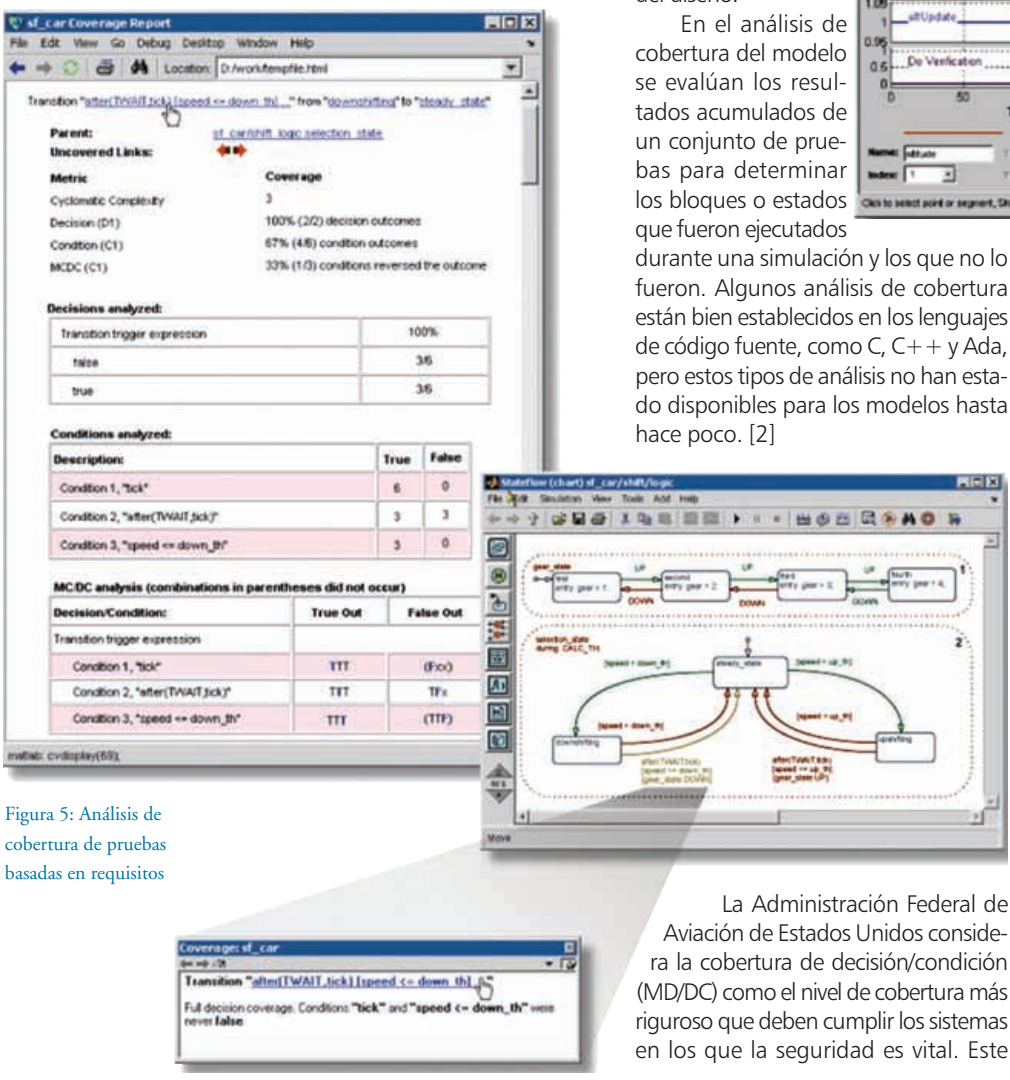

Figura 5: Análisis de cobertura de pruebas basadas en requisitos

La Administración Federal de Aviación de Estados Unidos considera la cobertura de decisión/condición (MD/DC) como el nivel de cobertura más riguroso que deben cumplir los sistemas en los que la seguridad es vital. Este análisis de la cobertura, entre otros, está ahora disponible en el diseño basado en modelos y se lleva a cabo basándose en ejecuciones de las simulaciones. Cuando se realiza dentro de las herramientas de simulación, MC/DC permite el registro automático y la creación de informes de los datos de cobertura del modelo (figura 5). Así, el ingeniero de pruebas puede evaluar si los escenarios de prueba están completos en lo referente a la estructura del diseño. El desafío se presenta entonces en la definición de un conjunto de pruebas que representen una cobertura total de manera eficiente.

Generación automática de pruebas

Un nuevo conjunto de herramientas para el diseño basado en modelos puede generar automáticamente patrones de prueba que satisfagan los objetivos de cobertura especificados, como MC/DC, analizando automáticamente la estructura del modelo y usando métodos formales para generar patrones. Este análisis estructural también identifica si hay partes del modelo que nunca se ejecutarán, lo que puede ser una indicación de que algo no se tuvo en cuenta durante la creación de la especificación, la implementación o las pruebas.

Se pueden combinar estos patrones de prueba con otros escenarios de prueba derivados de los requisitos, de los datos de los bancos de prueba, de escenarios de simulaciones de Monte Carlo y de modelos de planta/configuración para probar de manera exhaustiva el modelo durante la etapa de las simulaciones, así como durante su posterior implementación real (Figura 6).

Comprobación del cumplimiento de directrices en el código

La Asociación para la fiabilidad de software en la industria del motor (MISRA) ha publicado "Guidelines for the Use of the C Language in Vehicle Based Software" (Directrices sobre el uso del lenguaje C en software para vehículos). Cada vez más fabricantes y proveedores de automóviles adoptan este conjunto de directrices, conocido comúnmente como MISRA-C [5]. Gran parte de la comprobación del cumplimiento de las directrices de MISRA-C de un modelo se puede llevar a cabo mirando lo que la herramienta de generación automática de código genera, en lugar de comprobar todo el código generado, algo que sí es necesario si se crea manualmente.

Pero la comprobación de la herramienta de generación de código no incluye escenarios, como el código heredado y escrito a mano importado. Se pueden usar interfaces abiertas de modelo para comprobar automáticamente estos escenarios. De esta manera, las comprobaciones se realizan después de la creación del modelo y antes de la generación de código, garantizando así la verificación del código generado e importado. Otra opción es incluir un comprobador de código MISRA-C o una herramienta de análisis estático en el proceso de generación de código.

Detección de los errores en tiempo de ejecución

Los errores en tiempo de ejecución son especialmente difíciles de detectar a nivel del modelo o durante la simulación y esto puede ocasionar problemas graves durante el desarrollo o las pruebas del software. Los errores en tiempo de ejecución son fallos latentes, que a menudo se producen con combinaciones específicas de valores de los datos, por lo que resulta caro localizarlos mediante la realización de pruebas dinámicas. De hecho, normalmente se descubren por las consecuencias que tienen en comportamientos funcionales, por ejemplo, en el envío de comandos inesperados a actuadores, la parada del coprocesador matemático y fallos inexplicables del software o difíciles de reproducir. En estos casos, es necesario un largo proceso de depuración para localizar el origen del problema. [3]

El análisis estático es un enfoque para la detección de los errores en tiempo de ejecución. En los últimos años, se han introducido herramientas de verificación estáticas que utilizan técnicas de análisis avanzado y han reducido el número de resultados "falso-positivos" que requieren una inspección o realización manual de pruebas .[4] Estas herramientas realizan un análisis estático, y también dinámico, del código C, independientemente de si el código se generó de manera automática o manual. La integración de estas herramientas de verificación con las herramientas de diseño basado en modelos supone una mejora importante para el flujo de trabajo. Al conectar el código analizado y el modelo a partir del cual se generó automáticamente, la herramienta de verificación estática puede presentar sus resultados tanto en el código fuente como en el modelo. El hecho de poder navegar del código al modelo, realizar el cambio y, después volver a generar y comprobar el código automáticamente, hace que ésta sea una manera muy eficaz para analizar, depurar y modificar algoritmos tanto desde una perspectiva de alto nivel como detallada. De este modo, se mejora el proceso de desarrollo ya que los cambios se realizan en el modelo en lugar de directamente en el código, lo que contribuye a la longevidad y reutilización de los modelos entre distintos proyectos.

Observaciones finales

La filosofía clave descrita en este artículo representa las tres mejores prácticas parade sacar el máximo partido del diseño basado en modelos para pruebas y verificación

En primer lugar, se aconseja la reutilización de los modelos como banco de pruebas para la implementación. A partir de las simulaciones de modelos, la ejecución de software implementeado vinculado a los modelos, la ejecución en el host o en el target y la ejecución del sistema completo embebido en un banco de pruebas, se pueden recopilar conocimientos, datos de pruebas y otra información útil que se podrá reutilizar más adelante en los procesos de desarrollo y realización de pruebas.

En segundo lugar, se aconseja realizar pruebas lo antes posible: pruebas simulación antes que en tiempo real, pruebas en tiempo real en el banco de ensayos antes de aplicarlo al mundo real y pruebas del modelo antes del código. La realización de pruebas al principio del proceso es generalmente más fácil, ya que el nivel de abstracción es más alto, y los beneficios económicos de la detección temprana de errores están bien documentados.

En tercer lugar, se aconseja aprovechar al máximo todas las técnicas disponibles. La simulación y los métodos formales deberían reforzarse mutuamente; las técnicas basadas en código y las basadas en modelos para la realización de pruebas y verificación se complementan, y todas ellas están disponibles para ser usadas en el diseño basado en modelos a partir de una serie de herramientas de distintos proveedores.

Muchos fabricantes y proveedores de equipamiento original de automoción utilizan el diseño basado en modelos para generar especificaciones ejecutables, simular su rendimiento y generar código automáticamente. Muchas de estas empresas comienzan a dar el siguiente paso, aprovechando los modelos existentes para realizar pruebas y verificación. Este artículo explora una amplia gama de métodos que ilustran los enfoques prácticos que pueden usarse para mejorar la calidad y reducir los costes del software embebido de automoción.

Referencias

1. Automotive Engineering International, marzo de 2005.

2. Aldrich, "Using model coverage analysis to improve the controls development process", AIAA 2002.

3. Hote, "Advanced Software Static Analysis Techniques that Provide New Opportunities for Reducing Debugging Costs and for Streamlining Functional Tests", prepublicación.

4. Ibid.

5. MISRA-C: 2004, "Guidelines for the use of the C language in critical systems", ISBN #0-9524156-2-3, www.misra-c2.com.