Mondragon Unibertsitatea

Faculty of Engineering

FUNDAMENTOS DE LA INGENIERÍA DE SISTEMAS BASADA EN MODELOS PARA LA GESTIÓN DEL CICLO DE VIDA

Plataformas de testeo y validación

Máster en Sistemas Inteligentes de Energía

ÍNDICE

1 Abreviaturas

AC Alternating current

DC Direct current

HIL Hardware-in-the-loop

MBD Model-Based design

MBSE Model-Based systems engineering

MIL Model-in-the-loop

PIL Processor-in-the-loop

SIL Software-in-the-loop

RCP Rapid Control Prototyping

2 Modelo en V para la gestión del ciclo de vida

El desarrollo y la fabricación de un producto pueden tener varias fases. En el ámbito de los sistemas inteligentes de energía, a menudo se comercializan productos complejos compuestos por numerosos subsistemas, que pueden ser tanto hardware como software. Además, hoy en día la conectividad que aporta la Industria 4.0 convierte a los sistemas actuales en sistemas ciber físicos. Se trata de un conjunto de mecanismos controlados por algoritmos informáticos.

Como ejemplo de un sistema de este tipo, la Figura 1 muestra un diagrama de bloques de un patinete eléctrico. Aunque no parezca un sistema tan complejo, la jerarquía de la arquitectura de la Figura 2 ya muestra cierta complejidad.

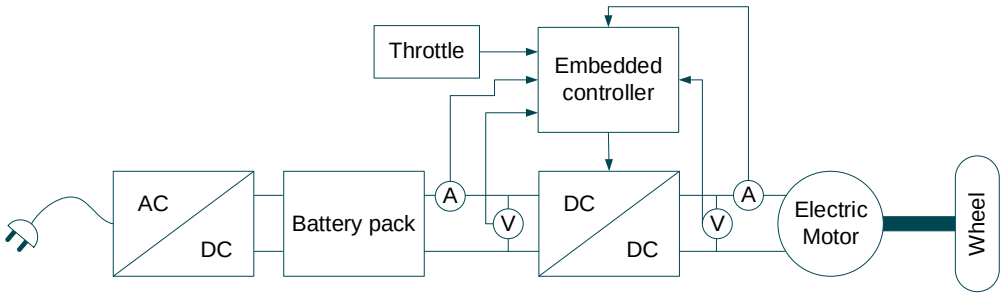


Figura 1 Diagrama de bloques del patinete eléctrico

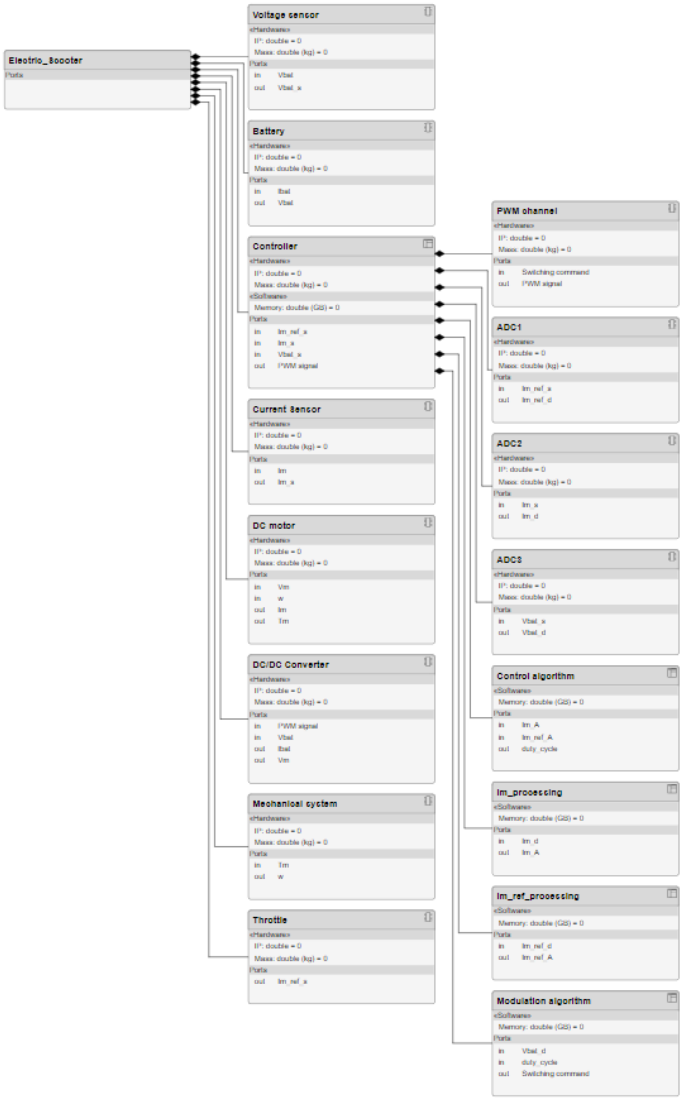


Figura 2 Jerarquía de la arquitectura del patinete eléctrico

En este contexto, es necesario establecer una metodología que permita el desarrollo teniendo en cuenta los principios de RAMS (Fiabilidad, Disponibilidad, Mantenibilidad y Seguridad). En la actualidad, uno de los procesos estándar más utilizados a este respecto es el denominado modelo V. Aunque originalmente estaba pensado para el desarrollo de software, en la actualidad se está extendiendo a la ingeniería de sistemas. Hoy en día ha sido adoptado por gobiernos como el alemán [1] o el estadounidense para el desarrollo de sistemas de transporte [2]. También ha sido normalizado por la IEC 62278 para la industria ferroviaria y la ISO 26262 para la industria del automóvil.

Este modelo (Figura 3) describe el ciclo de vida desde la concepción del producto hasta su operación y mantenimiento. El modelo se divide en dos ramas. La rama izquierda contiene los procesos de concepción, definición de requisitos y diseño (de alto nivel y detallado). Este proceso conduce al desarrollo y fabricación del producto (hardware y software). En la rama derecha se realizan los procesos de integración, verificación y validación. En este modelo, el eje temporal se dobla para formar una V y situar cada fase de la rama izquierda en el mismo nivel que su homóloga de la rama derecha.

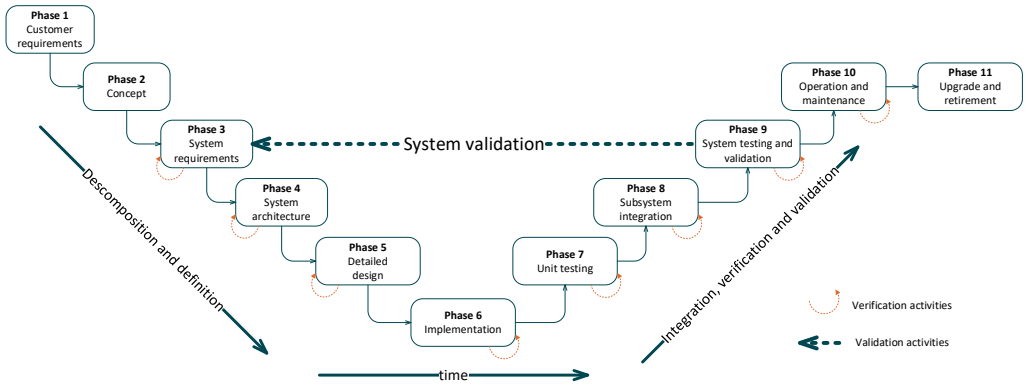


Figura 3 Diagrama del modelo en V

A continuación, se describe cada una de las fases de este modelo:

- Fase 1: Requisitos del cliente.
- Fase 2: Concepción. Las necesidades de los principales clientes y el entorno operativo del sistema que se va a diseñar se identifican y documentan en colaboración con el cliente.
- Fase 3: Requisitos del sistema. Los requisitos del cliente se traducen en requisitos del sistema. Se interpreta lo que busca el cliente y se traduce en requisitos técnicos que deben seguirse durante el diseño del sistema.
- Fase 4: Arquitectura del sistema. Se define una arquitectura de alto nivel para cumplir los requisitos del sistema. Se establecen los límites del sistema y las descomposiciones formales y funcionales y se identifican los subsistemas y las interfaces. Diseño de la arquitectura de alto nivel del sistema y correspondencia de los subsistemas con los requisitos.
- Fase 5: Diseño detallado. Diseño de subsistemas y componentes.
- Fase 6: Implementación, desarrollo de software y hardware. Selección de la tecnología adecuada y desarrollo de software y hardware para cumplir los requisitos de los componentes.

En estas seis primeras fases se descompone el sistema partiendo de la arquitectura de alto nivel. En las fases siguientes se realizan actividades de verificación y validación mientras se integra todo el subsistema:

- Fase 7: Pruebas unitarias. Se prueba cada componente de hardware y software, verificando su correcto funcionamiento a nivel unitario.
- Fase 8: Integración del subsistema: Integrar los componentes de software y hardware para crear subsistemas verificando su correcto funcionamiento a nivel de subsistema.
- Fase 9: Prueba y validación del sistema. Todos los subsistemas se integran para formar el sistema final. Se realizan pruebas del sistema y se valida con respecto a los requisitos del sistema.
- Fase 10: Operación y mantenimiento.
- Fase 11: Actualización o retirada.

Durante este proceso se crea la documentación del sistema. Para cada fase del lado izquierdo, se redactan los requisitos que guían la fase siguiente, así como el plan de validación para el nivel equivalente del lado derecho. Para cada fase del lado de la verificación y validación, se crea la documentación para la formación y validación de los usuarios.

Es necesario tener clara la diferencia entre las actividades de verificación y validación.

Por un lado, la validación se define como la evaluación de que un producto cumple los requisitos y necesidades del cliente. Suele implicar a partes interesadas externas al desarrollo. Durante este proceso se responde a la siguiente pregunta: ¿estamos desarrollando el producto adecuado? La validación es un proceso relativamente subjetivo que evalúa hasta qué punto el producto resuelve el problema del cliente. Por eso la validación del sistema se hace en función de los requisitos del sistema.

Por otro lado, la verificación se define como la evaluación de que un producto o servicio cumple las normas o especificaciones de diseño. Durante este proceso, se responde a la siguiente pregunta: ¿Estamos desarrollando el producto correctamente? En resumen, se trata de garantizar que el sistema construido está bien diseñado, es seguro y funciona correctamente. Este proceso evalúa en función de los requisitos internos.

Cuando nos enfrentamos a varias actividades y no sabemos si son tareas de verificación o validación, debemos valorar si implican dos fases consecutivas (de la fase n a la fase n+1) o implican fases paralelas en el ciclo de vida. Una tarea de verificación trata de comprobar si estamos preparados para pasar a la siguiente fase. Por lo tanto, esto se representa en Figura 4 con flechas que cierran bucles en la misma fase. Sin embargo, una tarea de validación comprueba si el sistema funciona según los requisitos, por lo que puede implicar la fase 3 y la fase 9 en la Figura 4.

Por último, la Tabla 1 muestra una clasificación de las diferentes tareas en el desarrollo de un scooter eléctrico.

Tabla 1 Ejemplos de actividades de verificación y validación

id	Tarea	Verificación	Validación	Comentarios
1	Comprobación de inspección para ver si el scooter eléctrico está listo para la prueba de conducción	X		Se integraron los subsistemas y la prueba de conducción forma parte de las pruebas del sistema de la fase 9.
2	Recopilar los resultados de las pruebas y redactar la documentación que demuestre el cumplimiento de los requisitos de seguridad.		X	Implica las fases 3 y 9 del modelo V.
3	Participación en talleres para definir requisitos y normas	X		Parte de la fase 3
4	Controlar que todos los requisitos de seguridad se incluyan en los documentos de diseño.	X		Se trata de comprobar si la fase 4 tiene en cuenta los resultados de la fase 3.
5	Inspección detallada de la calidad conforme a los documentos de diseño		X	Comprobaciones en la fase 9 respecto a los documentos de la fase 4.

La Figura 4 muestra la localización de cada actividad en el ciclo de vida.

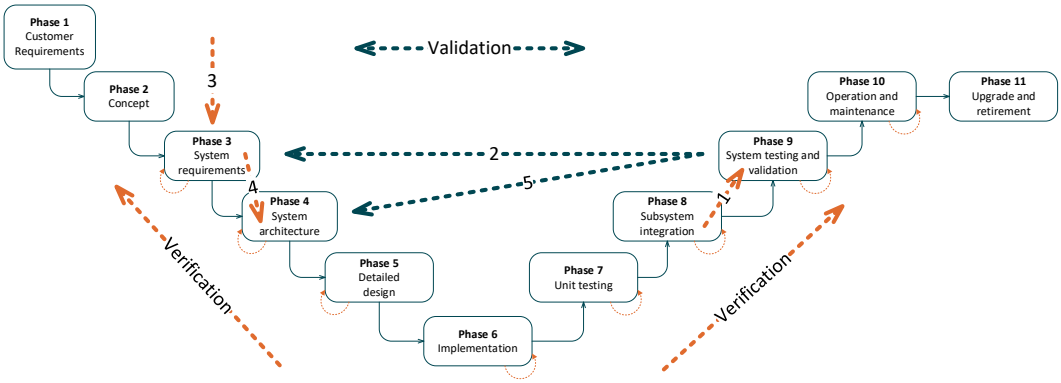


Figura 4 Actividades de validación y verificación en el modelo en V

3 Ingeniería de sistemas basada en modelos

3.1 Los modelos como centro del desarrollo de productos

El modelo V descrito anteriormente, hasta hace unos años, era una metodología centrada en los documentos. La información generada en el proceso se recogía y transmitía mediante documentos. Esta forma de trabajar se ha vuelto cada vez más difícil de gestionar, dada la complejidad de los sistemas actuales. Es difícil representar mediante documentos todos los puntos de vista desde los que se puede contemplar un sistema y mantenerlos actualizados a medida que avanza el diseño y el ciclo de vida. La Figura 5 muestra una representación simplificada de los puntos de vista en función del profesional que los contempla.

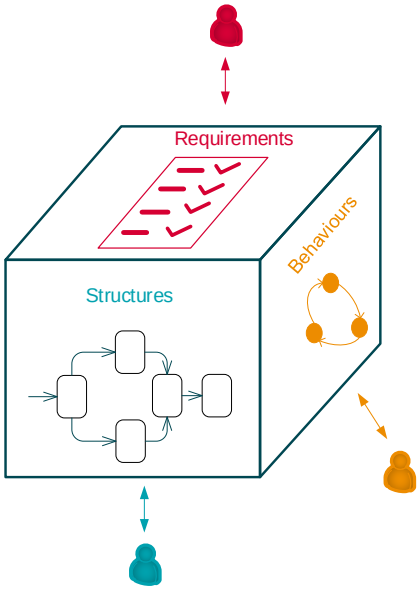


Figura 5 Vistas de un sistema (modificado de [3])

Ante estas dificultades, las empresas han adoptado una nueva forma de trabajar denominada Ingeniería de Sistemas Basada en Modelos (MBSE). En la filosofía MBSE, un único modelo representa los requisitos, la arquitectura y el comportamiento de un sistema. INCOSE [4] describe MBSE como "la aplicación formalizada del modelado para apoyar los requisitos del sistema, el diseño, el análisis, la verificación y las actividades

de validación que comienzan en la fase de diseño conceptual y continúan a lo largo del desarrollo y las fases posteriores del ciclo de vida".

Según [5], un modelo es una representación física, matemática o lógica de un sistema, entidad, fenómeno o proceso. En el apartado anterior, la palabra modelo hacía referencia a una forma de representar un proceso, en concreto, el ciclo de vida. Sin embargo, a partir de ahora, por modelo nos referiremos a la representación de un sistema, en nuestro caso, en sistemas energéticos. En este ámbito, un modelo es una réplica virtual de un sistema energético capaz de representar su estructura y comportamiento dinámico.

Los modelos se utilizan desde hace años en las fases de diseño y desarrollo. El modelado es un paso esencial para el desarrollo de controles para sistemas dinámicos. Además, los programas de desarrollo y simulación como MATLAB & Simulink permiten diseñar, analizar y simular modelos dinámicos de forma sencilla y rápida.

La ventaja de utilizar MBSE es obvia, ya que un modelo compartido por todo el equipo de desarrollo centraliza todos los puntos de vista mostrados en la Figura 5. Además, la comunicación de la información se basa en modelos visuales, más fáciles de entender [6]. Este enfoque permite trabajar sin tener que generar documentación adicional. Alguien que trabaje en el seguimiento de requisitos puede que sólo esté interesado en los resultados de la validación, mientras que alguien que trabaje en el diseño de un componente concreto querrá ver la especificación de las interfaces o su comportamiento dinámico.

Se han propuesto distintas técnicas para modelar sistemas complejos y hacer realidad el MBSE. Una de ellas es SysML (System Modelling Language), una variante de UML (Unified Modelling Language) utilizada en ingeniería de software. SysML establece diagramas normalizados para representar sistemas desde distintos puntos de vista (véase la Figura 6).

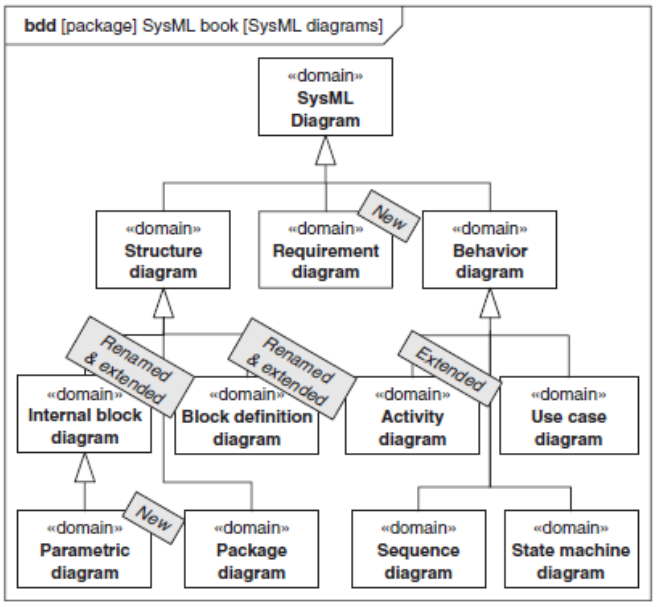


Figura 6 Diagramas SysML (y diferencias respecto a UML) [7]

Como ya se ha dicho, el sistema se ve desde tres perspectivas. El diagrama de requisitos (req) de la figura 4 representa el punto de vista de los requisitos.

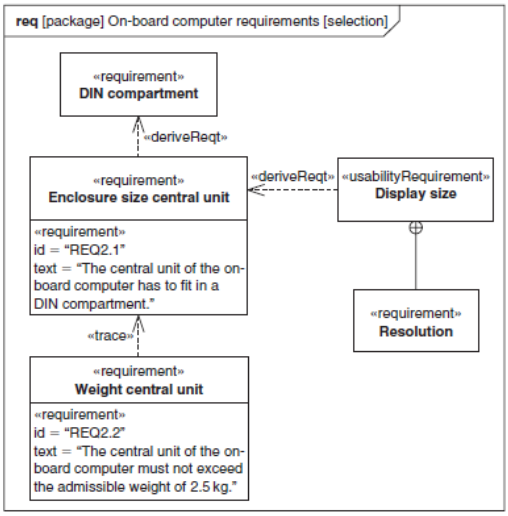
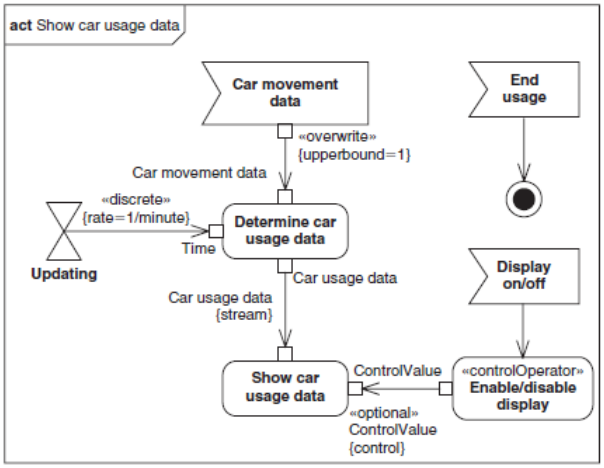
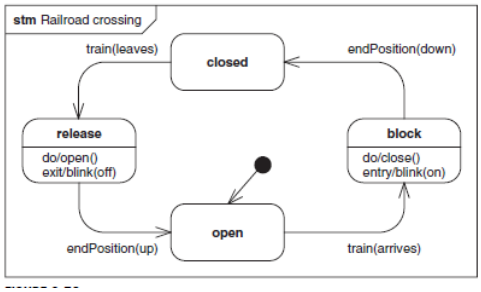


Figura 7 Diagrama de requisitos [7]

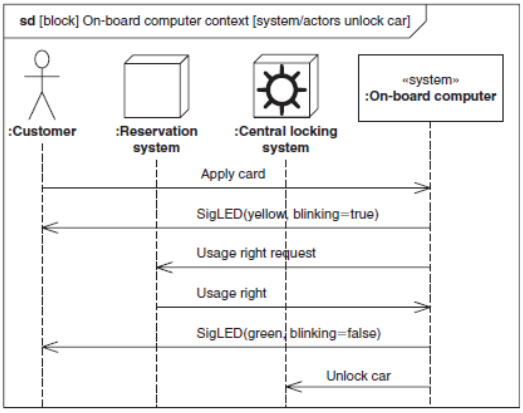
Los diagramas de actividad (act), secuencia (sd), caso de uso (uc) y máquina de estados (st) pueden representar el comportamiento.



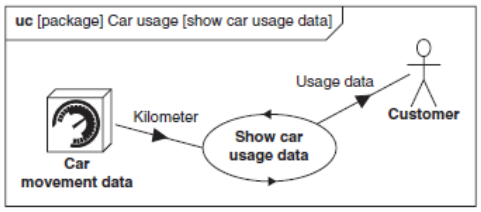
(a)



(b)



(c)



(d)

Figura 8 Diagramas de comportamiento de SysML. (a) Diagrama de actividad (b) Diagrama de casos de uso (c) Diagrama de secuencia (d) Diagrama de casos de uso [7]

Por último, la estructura se representa mediante diagramas de bloques internos (ibd), paquetes (pkg) y definición de bloques (bdd).

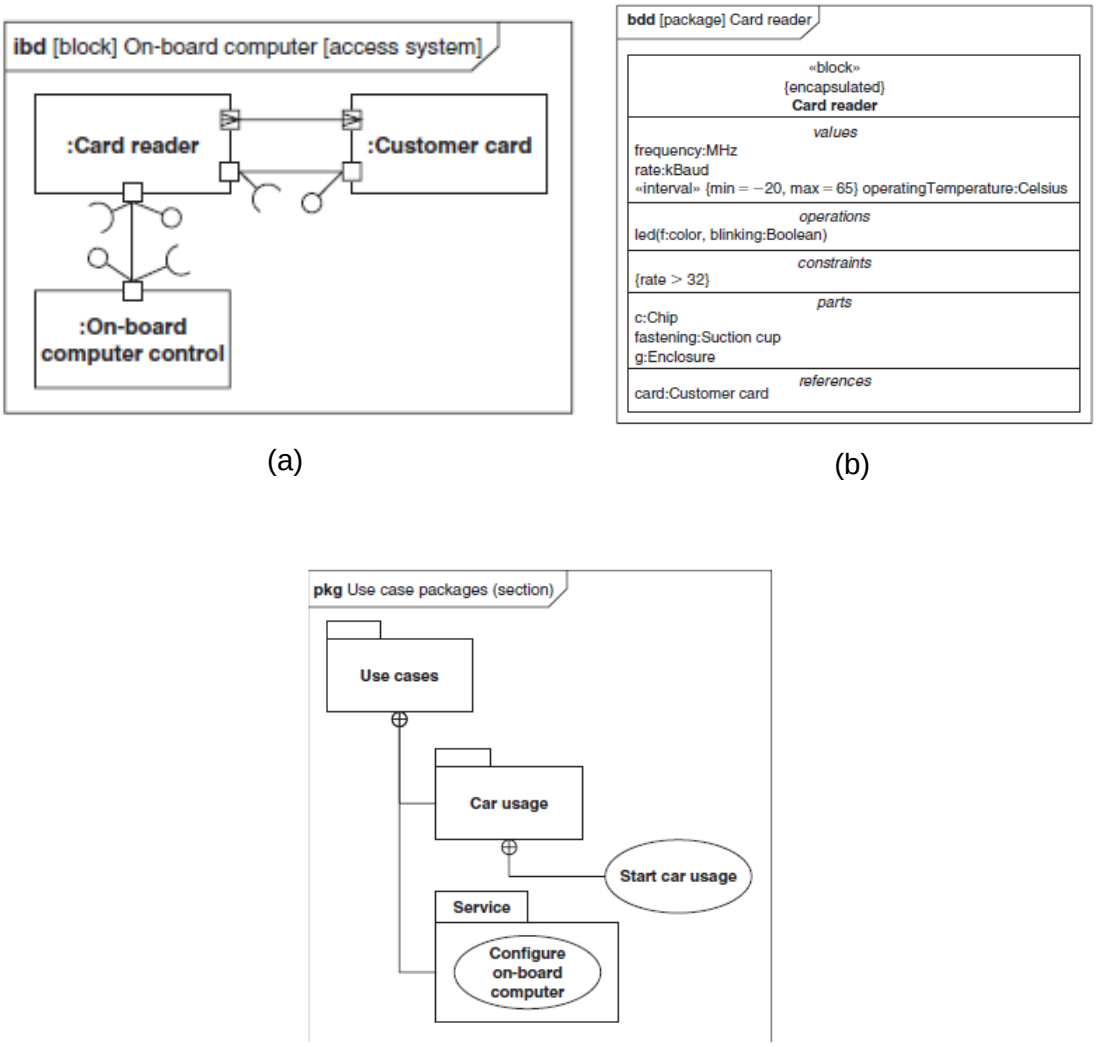


Figura 9 Diagramas de estructura de SysML (a) Diagrama de bloques interno (b) Diagrama de definición de bloque (c) Diagrama de paquetes [7]

Las principales ventajas del MBSE son [8]:

- Mejora de la comunicación en equipos multidisciplinares. Hoy en día en el desarrollo de sistemas ciber-físicos intervienen profesionales de muchas disciplinas: ingenieros eléctricos, ingenieros mecánicos, ingenieros de software, arquitectos de sistemas... Todos ellos utilizan lenguajes y herramientas diferentes, por lo que disponer de un modelo unificado del sistema ayuda a coordinar el trabajo de todos, evitar confusiones y ambigüedades.
- Permite a las empresas diseñar y simular sus productos antes de fabricarlos o tener prototipos. Esta ventaja se explota desde hace décadas en las fases 4 y 5 del ciclo de vida, donde las herramientas de simulación son esenciales para analizar el rendimiento de las arquitecturas y topologías propuestas a un coste reducido. Así, el MBSE permite analizar y anticipar en la fase 6 decisiones que antes se habrían tomado sobre la base de prototipos.
- Facilita la gestión de la complejidad del sistema. Gracias a los modelos, un sistema puede verse desde distintos puntos de vista de forma rápida y sencilla. Existe un modelo maestro único que todos pueden consultar desde su propio punto de vista.
- Los documentos pueden generarse automáticamente a partir del modelo, lo que facilita su mantenimiento y reduce costes y esfuerzos.

3.2 Herramientas para el diseño basado en modelos

Con el avance del MBSE han surgido distintas herramientas para la validación y verificación de sistemas. Además de representar el sistema mediante estándares como SysML y simularlo con software como MATLAB & Simulink, las plataformas de simulación se han ampliado a todo el ciclo de vida. Los modelos se utilizan tanto para sustituir el sistema a controlar como para disponer de una versión inicial del control antes de su codificación.

Existen diferentes técnicas de simulación para la verificación basada en modelos, como el Model-in-the-loop (MIL), Software-in-the-loop (SIL), Processor-in-the-loop (PIL), Hardware-in-the-loop (HIL) y Rapid Control Prototyping (RCP). A continuación, se detallan las características y la estructura de cada una de estas estrategias para una aplicación de control clásica, tal y como se muestra en la Figura 7.

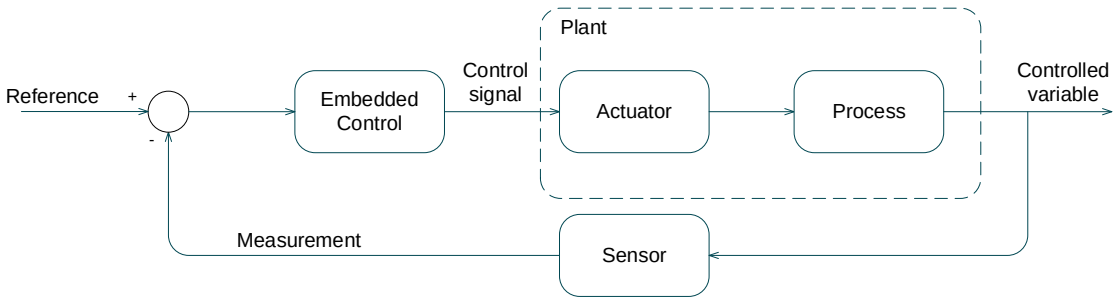


Figura 10 Diagrama de control básico

3.2.1 Model-in-the-loop

En la técnica Model-in-the-Loop se simula el sistema completo, tanto el control como la planta.

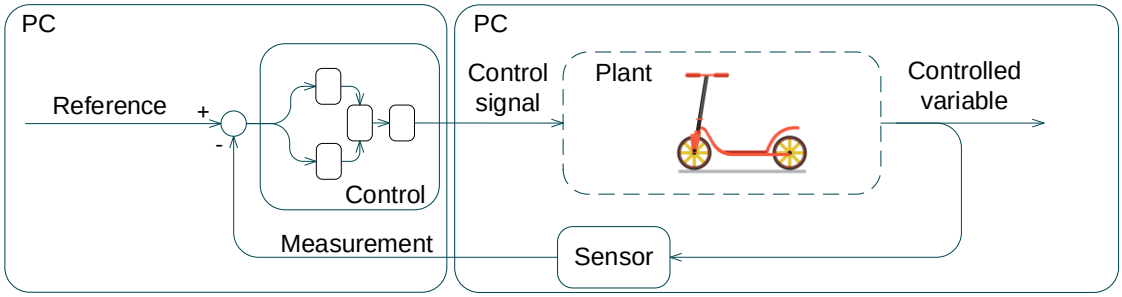


Figura 11 Estructura MIL

En el caso de MATLAB & Simulink, se trataría de tener un modelo de planta y un modelo de control. El modelo de control se refiere a una estrategia implementada mediante diagramas de bloques o máquinas de estado. En este escenario, el control aún no está codificado. Lo habitual es una simulación informática pura, pero puede haber casos en los que las simulaciones se realicen en simuladores en tiempo real.

3.2.2 Software-in-the-loop

La técnica Software-in-the-loop consiste en implementar el control en código. El producto se sigue implementando en plataformas de simulación, como un ordenador o un simulador en tiempo real, pero la implementación es real. El código se aproxima a la versión final que se entregará en el hardware de control. Este escenario permite verificar el código, las secuencias de ejecución e incluso dimensionar la memoria necesaria en el futuro procesador en tiempo real.

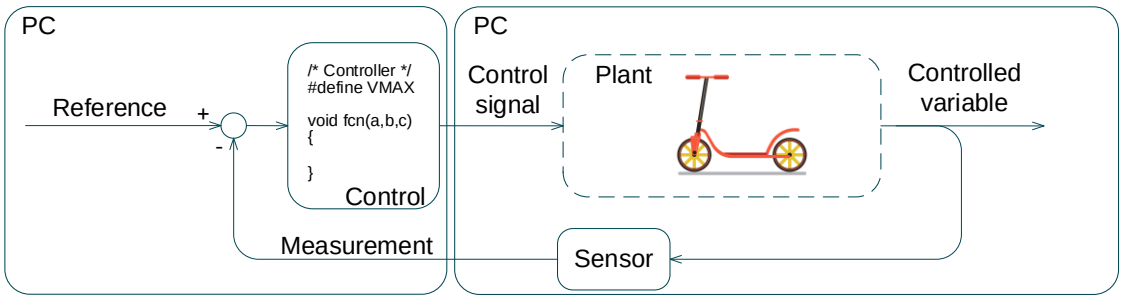


Figura 12 Estructura SIL

3.2.3 Processor-in-the-loop

En este escenario, el control ya está implementado en un controlador hardware, aunque sea como prototipo. De este modo, pueden medirse los tiempos de ejecución, el uso de memoria, etc. En las aplicaciones de control, como el controlador tendrá que leer o escribir señales analógicas o digitales, puede ser necesario que la planta se simule en un simulador en tiempo real. Para ello es necesario que la planta funcione y dé una respuesta inferior al tiempo mínimo de ejecución del control.

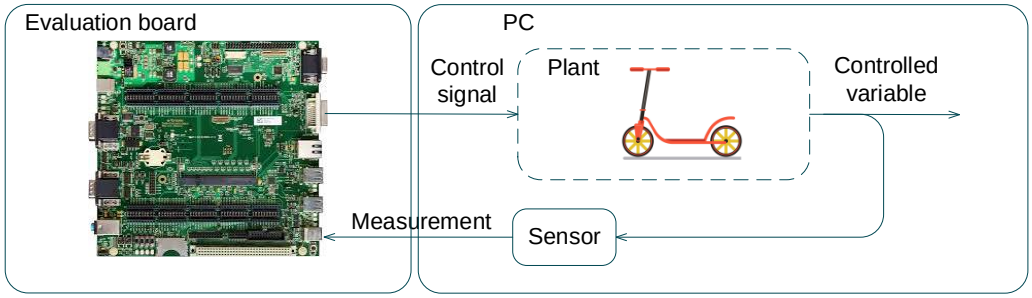


Figura 13 Estructura PIL

3.2.4 Hardware-in-the-loop

Esta herramienta permite validar el hardware y el software de control sin necesidad de disponer de la planta real. El simulador dispone de las entradas y salidas analógicas y digitales para comunicarse con el controlador, de forma que puede ejecutar el software de la misma forma que en la aplicación real. Dependiendo del controlador, el simulador ejecutará el modelo de la planta en tiempo real, incluso se pueden implementar comunicaciones de todo tipo para emular el sistema real controlado.

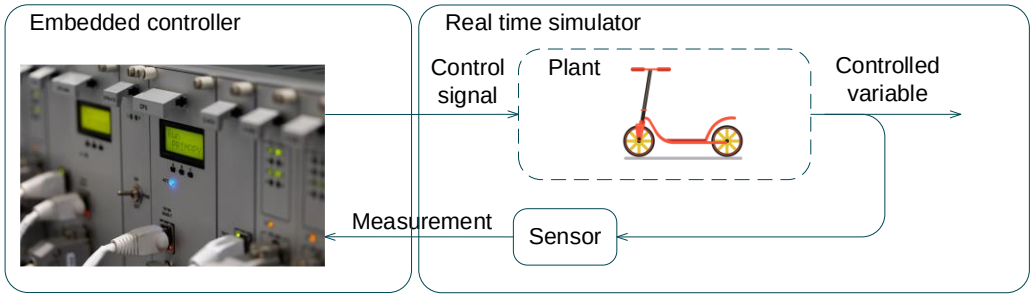


Figura 14 Estructura HIL

3.2.5 Rapid control prototyping (RCP)

En los casos en que el sistema que se desea controlar ya existe, esta técnica permite realizar implementaciones experimentales de la estrategia de control en entornos de prueba. Sin necesidad de disponer del hardware de control definitivo, la estrategia de control puede probarse en equipos de laboratorio para acelerar el proceso de verificación.

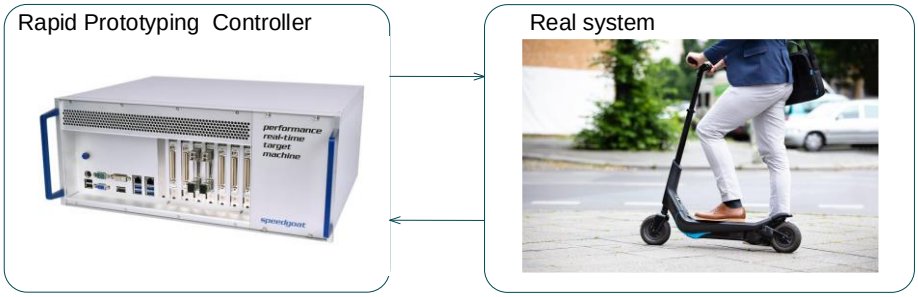


Figura 15 Estructura RCP

3.3 MATLAB & Simulink para la aplicación del modelo en V

Tradicionalmente, en el campo de los sistemas energéticos y el control de sistemas dinámicos, MATLAB & Simulink ha sido el software de referencia para su desarrollo. Hasta hace unos años, este software se utilizaba para analizar el comportamiento de los sistemas representándolos y simulándolos mediante diagramas de bloques. Pero últimamente, cada vez hay más complementos que permiten integrar todas las vistas mostradas en la Figura 5 en un único proyecto de simulación. Puede decirse que el software de simulación dinámica y el MBSE se están encontrando y uniendo sus fuerzas para mejorar y facilitar el desarrollo de sistemas complejos.

La Figura 16, la Figura 17y la Figura 18 muestran diagramas de bloques construidos con Simulink System Composer como alternativa a los diagramas SysML. La gran ventaja de este tipo de herramientas es que integran el modelo de comportamiento (dinámico) habitual con información sobre arquitectura y requisitos.

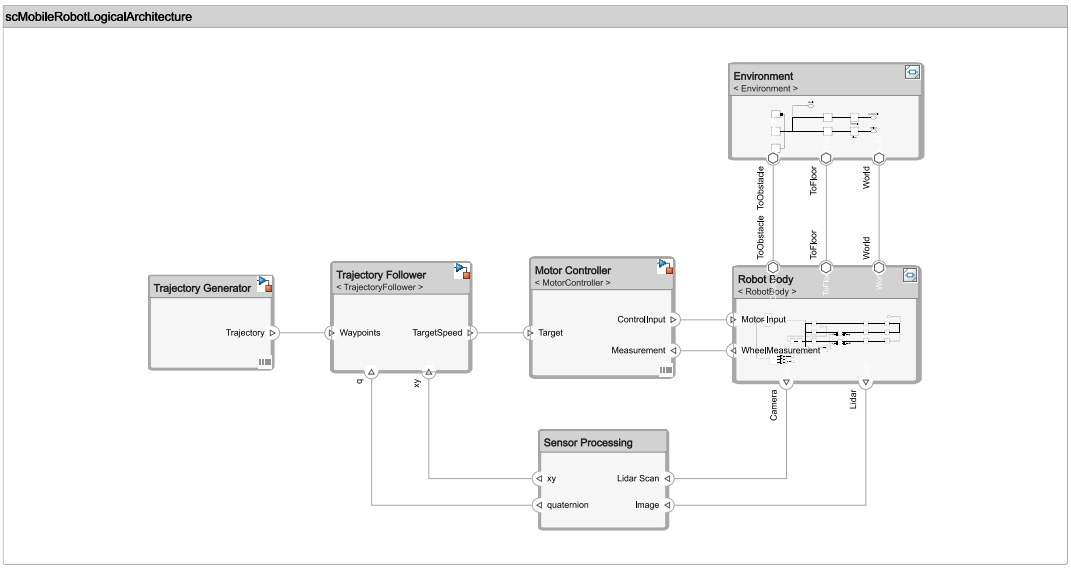


Figura 16 Diagrama de bloques y simulación dinámica diseñado con System Composer [9]

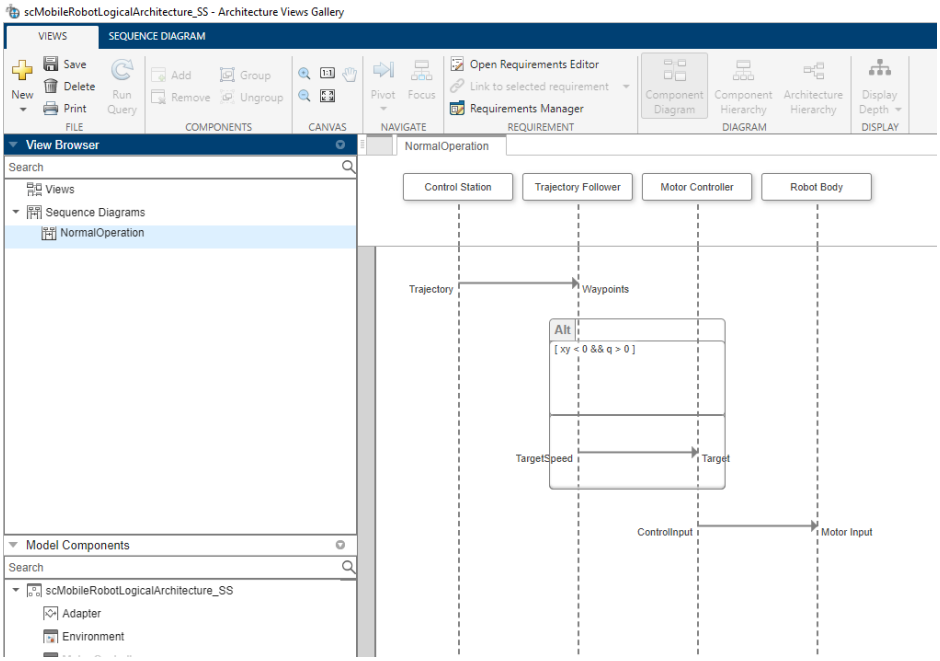


Figura 17 Diagrama de secuencia generado con System Composer [9]

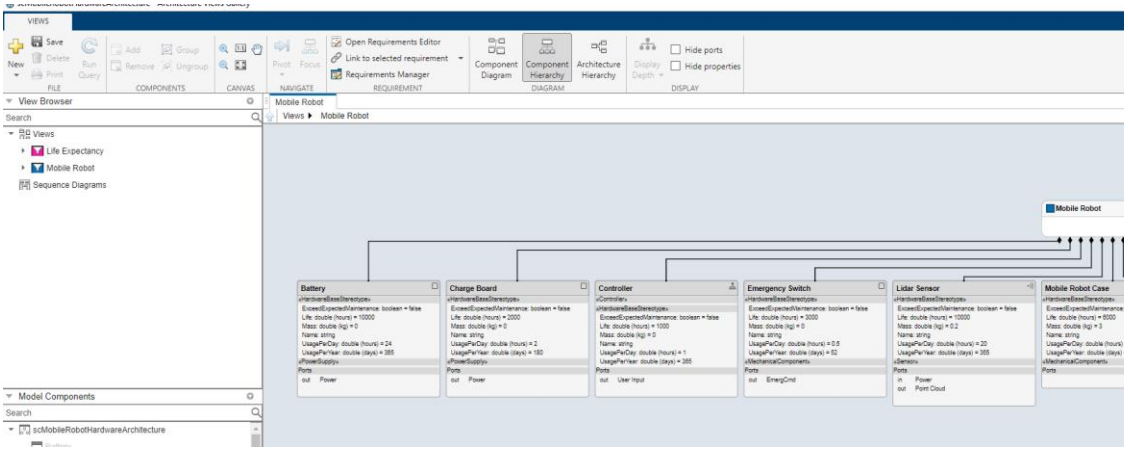


Figura 18 Diagrama de jerarquía generado con System Composer [9]

MathWorks proporciona múltiples herramientas que pueden utilizarse en diferentes fases del proceso de modelado en V. Con el uso de MATLAB & Simulink y sus toolboxes se pueden conseguir mejoras en el producto final, debido a las ventajas de este tipo de diseño.

Algunas herramientas que se pueden identificar como útiles para la aplicación de modelo en V son estas:

- **Requirements toolbox**: permite crear, vincular y validar requisitos en MATLAB o Simulink. Puede crear requisitos utilizando texto con atributos personalizados o importarlos de herramientas de gestión de requisitos.
- **System Composer**: esta herramienta permite especificar y analizar arquitecturas para la ingeniería de sistemas basada en modelos y el modelado de arquitecturas de software. Con System Composer, puede asignar requisitos mientras perfecciona una arquitectura que, a continuación, puede diseñarse y simularse en Simulink.
- **Stateflow**: proporciona un lenguaje gráfico que incluye diagramas de transición de estados, diagramas de flujo, tablas de transición de estados y tablas de la verdad. Stateflow puede utilizarse para describir cómo reaccionan los algoritmos de MATLAB y los modelos de Simulink ante señales de entrada, eventos y condiciones temporales.
- **Embedded coder**: genera código C y C++ legible, compacto y rápido para procesadores embebidos utilizados en la producción en masa. Amplía MATLAB & Simulink Coder con optimizaciones avanzadas para un control preciso de las funciones, los archivos y los datos generados. Estas optimizaciones aumentan la eficiencia del código y facilitan la integración con código heredado, tipos de datos y parámetros de calibración. Puede incorporar una herramienta de desarrollo de terceros y crear un ejecutable para su implantación llave en mano en el sistema embebido o en la placa de prototipado rápido.
- **Simulink Test**: proporciona herramientas para crear, gestionar y ejecutar pruebas sistemáticas basadas en simulaciones de modelos, código generado y hardware simulado o físico. Incluye plantillas de pruebas de simulación, benchmark y equivalencia que permiten realizar pruebas funcionales, unitarias, de regresión y consecuenciales utilizando los modos software-in-the-loop (SIL), processor-in-the-loop (PIL) y hardware-in-the-loop (HIL) en tiempo real.
- **Generador de informes de Simulink**: permite crear vistas web con las que se pueden ver, navegar y compartir modelos de Simulink desde un navegador web y sin licencia de Simulink. Puede integrar vistas web en informes HTML de generación de código, requisitos, cobertura y otros.

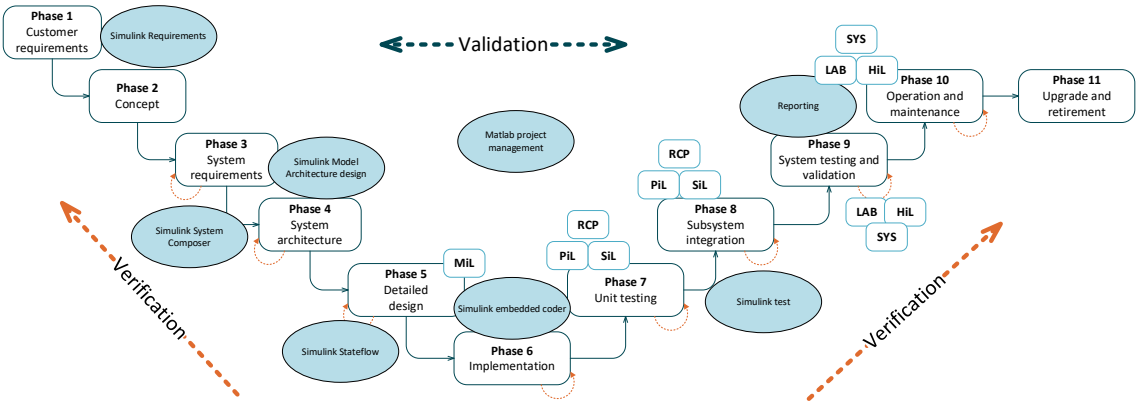


Figura 19 Herramientas de MATLAB & Simulink para el desarrollo basado en modelos y modelo en V

Tomando como ejemplo el desarrollo de una unidad de control para accionamientos eléctricos (software + hardware), las herramientas descritas suelen utilizarse del siguiente modo:

- En las fases de diseño no hay software ni hardware de control. La técnica MIL permite simular y analizar en detalle la arquitectura y el diseño de cada subsistema.
- En la fase de pruebas unitarias, el hardware y el software se siguen probando por separado. En cuanto al software, hay varias opciones. Con SIL, el código de control se integra en la simulación o con HIL, el código puede probarse en una tarjeta de control de prueba contra una planta simulada en tiempo real. Con RCP, el código de control puede probarse contra la planta real (o de laboratorio) en un controlador de prueba.
- En la fase de verificación de subsistemas, se utilizan las mismas herramientas que en la fase anterior, pero pueden probarse componentes en su conjunto o subsistemas completos.
- En la fase de integración, el software y el hardware de control se integran gradualmente y se prueban en HIL, laboratorio y escenario real.
- En la fase de aceptación y validación, se utilizan las mismas herramientas para garantizar el cumplimiento de los requisitos del sistema.
- En la de operación y mantenimiento, se siguen utilizando las herramientas anteriores para resolver incidencias o diseñar mejoras.

4 Referencias bibliográficas

[1] M. Meisinger and I. H. Krüger, “A Service-Oriented Extension of the V-Modell XT *,” 2007.

[2] US Department of Transportation, “Systems engineering for intelligent transportation systems,” p. 11, 2007, [Online]. Available: http://ops.fhwa.dot.gov/publications/seitsguide/seguide.pdf

[3] “Model Based Systems Engineering (MBSE) on AWS: From Migration to Innovation.”

[4] INCOSE, “SYSTEMS ENGINEERING VISION 2020,” 2007. Accessed: Jan. 04, 2023. [Online]. Available: https://sdincose.org/wp-content/uploads/2011/12/SEVision2020_20071003_v2_03.pdf

[5] J.S. Gansler, “DoD Modeling and Simulation (M&S) Glossary,” 1998.

[6] Chantal Cappelletti, Simone Battistini, and Benjamin K. Malphrus, Eds., *Cubesat Handbook*. Elsevier, 2021. doi: 10.1016/C2018-0-02366-X.

[7] Tim Weilkiens, *Systems Engineering with SysML/UML*. Elsevier, 2007. doi: 10.1016/B978-0-12-374274-2.X0001-6.

[8] J. Holt, S. A. Perry, and M. Brownsword, *Model-Based Requirements Engineering*. Institution of Engineering and Technology, 2011. doi: 10.1049/PBPC009E.

[9] MathWorks, “Simulate Mobile Robot with System Composer Workflow.” https://www.mathworks.com/help/systemcomposer/ug/mobile-robot-workflow.html (accessed Jan. 04, 2023).

[10] W. W. Royce, “Managing the Development of Large Software Systems,” in *ICSE ’87: Proceedings of the 9th international conference on Software Engineering*, 1987, pp. 328–338. Accessed: Jan. 04, 2023. [Online]. Available: https://dl.acm.org/doi/10.5555/41765.41801

[11] N. D. Birrell and M. A. Ould, *A Practical Handbook for Software Development*. Cambridge University Press, 1985. doi: 10.1017/CBO9780511624223.