Modelling, Simulation and Code Generation for Electronic Railway Interlocking Systems

Ramiro Ghignone, Cristian Falco, Facundo Larosa, Hernán Mendes, Leandro Chang, Martín Menéndez and Ariel Lutenberg

***Abstract*— Electronic railway interlockings are critical embedded systems which control the safe operation of train signals. Due to the broad variety of railway network topologies and the high functional safety level required, a flexible solution is needed, capable of taking formal requirements and implementing them accordingly to the required application. The scope of this work is to present an approach in which an automatic code generator transforms the control tables which describe the interlocking logic into functional units written in different programming languages like C or VHDL. The generated code allows its implementation in an embedded system based in a FPGA or a microcontroller. In addition, the project contains a graphical user interface to draw and simulate the behavior of the generated model for verification purposes. The developed tool comprises the entire design flow for interlocking systems and presents several advantages when compared with previous works.**

***Index Terms*— Railway interlocking, automatic code generation, object-oriented programming, functional safety, critical systems, FPGA.**

I. Introducción

Hoy en día, la red argentina de trenes moviliza a más de 400 millones de pasajeros por año y es uno de los medios de transporte más usados a nivel nacional [1]. Además, el país cuenta con una de las diez redes ferroviarias más extensas del mundo, con más de 36.000 km de vías férreas [2]. Sin embargo, en la mayor parte de estas vías se emplean sistemas de señalamiento mecánicos (operados mediante palancas) o electromecánicos (basados en relés y lógica cableada), los cuales son costosos de mantener o reponer debido a su obsolescencia. Además, presentan problemas de escalabilidad y de flexibilidad, ya que intervenir en su funcionamiento requiere una inversión económica importante y asumir la responsabilidad ante eventuales accidentes. En contraste, otros países con similar extensión ferroviaria han diseñado e implementado soluciones de enclavamiento electrónico [3-11]. Adicionalmente, tanto empresas como universidades han creado herramientas de software con el objetivo de reducir los tiempos de desarrollo y la probabilidad de fallas en el diseño de enclavamientos. Sin embargo, cuentan con ciertas limitaciones que se describirán en las Secciones III y IV.

Este trabajo se desarrolla en el marco del Grupo de Investigación en Calidad y Seguridad de Aplicaciones Ferroviarias (GICSAFe), perteneciente al Consejo Nacional de Investigaciones Científicas y Técnicas de la República Argentina (CONICET).

R. Ghignone, F. Larosa, C. Falco, H. Mendes y L. Chang pertenecen al grupo de I+D en Aplicaciones de Sistemas Embebidos de la Universidad Tecnológica Nacional, Facultad Regional Haedo.

(UTN-FRH-ASE; Haedo, Buenos Aires; e-mail: *embebidos@frh.utn.edu.ar*)

En este contexto, se propone una herramienta de software que integre el flujo completo de diseño, modelado e implementación de sistemas de enclavamiento electrónico. De esta forma, a partir de documentación de ingeniería se puede simular el sistema de enclavamiento diseñado mediante una interfaz gráfica y generar el código necesario para implementarlo en un sistema embebido. Cabe destacar que este proyecto es de interés para Trenes Argentinos, que participa activamente en los procesos de diseño, verificación y toma de decisiones. De esta forma se asegura que el resultado sea de utilidad para la red ferroviaria argentina.

El presente documento se organiza de la siguiente manera: la Sección II define los sistemas de enclavamiento, mientras que la Sección III introduce la solución propuesta para su implementación automatizada y sus ventajas respecto a enfoques previos. Finalmente, las Secciones IV y V presentan los resultados y conclusiones del trabajo, respectivamente.

II. Sistemas de Enclavamiento

A. Componentes del Señalamiento Ferroviario

Un sistema de enclavamiento ferroviario tiene por función garantizar el accionamiento seguro de los dispositivos de señalamiento y maniobra de los trenes. Su propósito es evitar situaciones que pongan en peligro la integridad física de las personas y bienes, tales como colisiones o descarrilamientos.

Los dispositivos de señalamiento pueden clasificarse según la función que cumplen:

- Detección de la posición del tren: se utilizan circuitos de vía, los cuales detectan la corriente de cortocircuito provocada por el eje del tren al unir dos rieles a diferente potencial [12].
- Direccionamiento del tránsito ferroviario: cumplen esta función las o máquinas de cambio cuya posición (normal o reverso) determina la dirección que tomará el tren en una bifurcación [12].

M. Menéndez y A. Lutenberg pertenecen al Laboratorio de Sistemas Embebidos de la Facultad de Ingeniería de la Universidad de Buenos Aires (LSE – FIUBA; Ciudad Autónoma de Buenos Aires; e-mail: *lse@fi.uba.ar*)

Actualmente este proyecto es parcialmente financiado a través de los subsidios UTN-PID 5253 y UBA-PDE 26/2019.

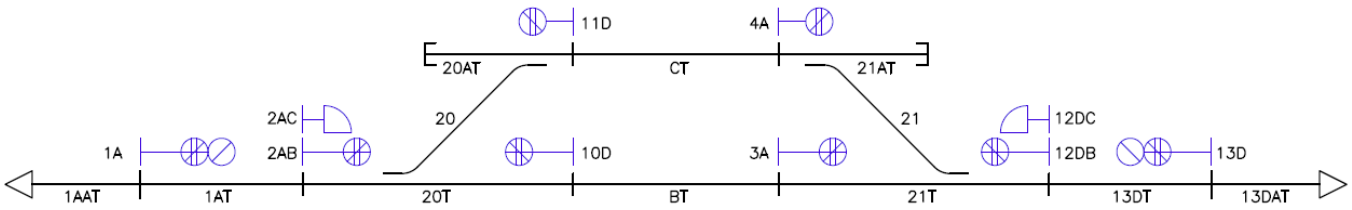


Fig 1. Diagrama de una estación de ejemplo, donde se aprecian los circuitos de vía (1AT, BT, etc.), cambios (20 y 21) y señales laterales (1A, 2AC,

- Señalización: el medio por el cual el enclavamiento habilita el avance del tren es mediante el uso de señales laterales, las cuales indican al maquinista si puede avanzar, y con qué velocidad debe hacerlo (marcha normal o con precaución) [13].
- Pasos a nivel: se emplean barreras automáticas para bloquear el paso de vehículos o peatones sobre una vía ferroviaria ante la proximidad de un tren.

La ubicación de estos elementos se indica mediante diagramas como el presentado en la Fig. 1, que muestra un ejemplo de estructura tipo "*bypass*" o estación de vía simple.

B. *Arquitectura y Tecnologías Existentes*

La Fig. 2 muestra el modelo en capas propuesto para la implementación de sistemas de enclavamiento a partir del análisis del estado del arte en implementaciones comerciales [3-11]. Se pueden distinguir tres capas:

1. Logística: contiene la interfaz de usuario que permite al operador de señalamiento controlar y monitorear el sistema.
2. Lógica: contiene la unidad central que procesa la lógica del sistema. En general deben tener nivel de seguridad funcional SIL 4 (del inglés Safety Integrity Level) [14], especificado según las normas EN 50128 y 50129 [15].
3. Infraestructura: contiene la interfaz entre la unidad central y los dispositivos de señalamiento y maniobra en campo (señales, cambios, circuitos de vía, pasos a nivel).

Esta arquitectura existe en todo tipo de sistemas de enclavamiento electrónico, tanto los comerciales basados en microprocesadores [3-11], como aquellos diseños académicos que emplean FPGA (del inglés *Field Programmable Gate Array)* para la capa lógica [16-17].

C. *Lógica Vital y Requerimientos de Seguridad*

Las funciones lógicas que se implementan en la unidad central del enclavamiento componen lo que se denomina lógica vital del sistema. Se basan el concepto de "ruta", que representa cada movimiento que puede realizar un tren, circulando desde una señal hasta otra señal consecutiva. Cada una de estas rutas puede ser habilitada solo si se cumplen ciertas condiciones de seguridad relativas a la ocupación de la vía, la posición de los cambios y/o el estado de las barreras. El operador de señalamiento solicita la activación de rutas y, si el movimiento solicitado es válido, se acciona la señal correspondiente y se comandan las barreras y cambios involucrados.

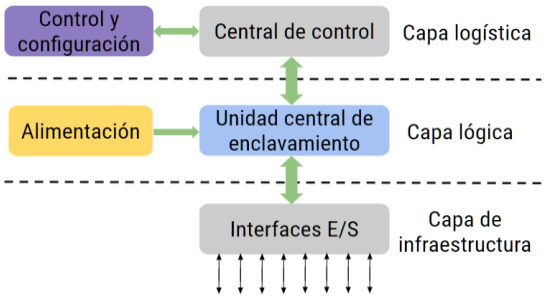


Fig 2. Modelo en capas de un sistema de enclavamiento completo

Las funciones lógicas de seguridad de un enclavamiento quedan especificadas a través de un documento formal denominado tabla de control o tabla de enclavamiento, que indica de forma explícita las condiciones que deben cumplirse para habilitar cada ruta [18]. A partir de la tabla se realiza el modelado e implementación del sistema, y se verifica el cumplimiento de los requisitos de seguridad [19]. La Tabla I muestra un ejemplo de tabla de control correspondiente al esquema de la Fig. 1.

TABLA I
TABLA DE CONTROL (SIMPLIFICADA) PARA EL EJEMPLO DE LA FIG. 1

Ruta	Señal de entrada	Señales de salida	Señales opuestas	Circuitos de vía en ruta	Cambio
1	1A	2AB,2AC	-	1AT	-
2	2AB	3A	10D	20T,BT	20(N)
3	2AC	4A	11D	20T,20AT, CT	20(R)
4	3A	-	12DB, 12DC,13D	21T,13DT, 13DAT	21(N)
5	4A	-	-	21AT,21T, 13DT,13DAT	21(R)

III. PLANTEO DEL PROBLEMA Y SOLUCIÓN PROPUESTA

A. *Software para Desarrollo de Sistemas de Enclavamiento*

Considerando la diversidad de topologías ferroviarias existentes (estaciones de vía simple y doble, playas de maniobra, empalmes, bloques automáticos, etc.), resulta claro que verificar cada uno de estos sistemas resulta muy costoso en términos de tiempo y esfuerzo. Por este motivo se han desarrollado varias herramientas de software destinadas a automatizar ciertas etapas del flujo de trabajo para el diseño, verificación e implementación de sistemas de enclavamiento, particularmente a través de sistemas embebidos [21-24].

La mayor parte de los trabajos académicos de desarrollo de software para sistemas de enclavamiento se enfocan en el estudio de métodos formales de verificación a través de técnicas de *model checking* [21]. Sin embargo, tal como se justifica en [22], la verificación de modelos con estas herramientas presenta poca flexibilidad para agregar nuevas funciones lógicas y poca escalabilidad para aplicar a casos reales. Frente a esto, se plantea la simulación aleatoria de eventos discretos [22] o la simulación interactiva [23] para verificar el cumplimiento de las reglas de seguridad.

Una decisión importante a tomar es la documentación a partir de la cual se inicia el flujo de diseño. No es esperable que el usuario de la herramienta reescriba el modelo del enclavamiento en un lenguaje formal para chequeo de modelos, ya que no es su especialidad. Algunos trabajos proponen la generación automática de las tablas de control a partir del diagrama de la estación [24]. Sin embargo, no siempre pueden definirse reglas generales de diseño que se adapten a todas las topologías. Resulta más conveniente partir de las tablas de control descritas en la sección II.C u otra documentación similar, tal como se discute en [25].

B. Solución Propuesta

A fin de reducir el tiempo y esfuerzo de modelado, verificación e implementación de la lógica vital en sistemas de enclavamiento, se desarrolló una solución denominada RAILIB (del inglés *RAilway Interlocking LIBrary*). Esta nueva herramienta presenta varias mejoras respecto a trabajos previos:

1. Parte de documentos de ingeniería de uso corriente.
2. Integra todo el flujo típico de diseño de enclavamientos.
3. Emplea modelos de mayor flexibilidad y escalabilidad.
4. Genera código para implementar en un sistema embebido
5. Permite la simulación y verificación gráfica del sistema.

En la Fig. 3 se puede ver el flujo de trabajo de la herramienta, la cual consta de cuatro partes que se indican a continuación y que se explican en las subsecciones III.C, III.D, III.E y III.F:

1. Bibliotecas de objetos de enclavamiento
2. Generador automático de código
3. Interfaz gráfica de diseño y simulación
4. Editor de tablas de control

C. Bibliotecas de Objetos de Enclavamiento

El primer paso para modelar los sistemas de enclavamiento fue estudiar y comprender su comportamiento. Inicialmente se se analizó un enfoque algebraico, representando mediante matrices las relaciones entre señales, circuitos de vía, rutas y pasos a nivel, de forma similar a las matrices utilizadas en la teoría de grafos [26]. Este análisis se realizó tomando como caso de estudio una de las estaciones ferroviarias ubicadas en la Ciudad Autónoma de Buenos Aires, y contrastando su funcionamiento con el comportamiento del modelo desarrollado.

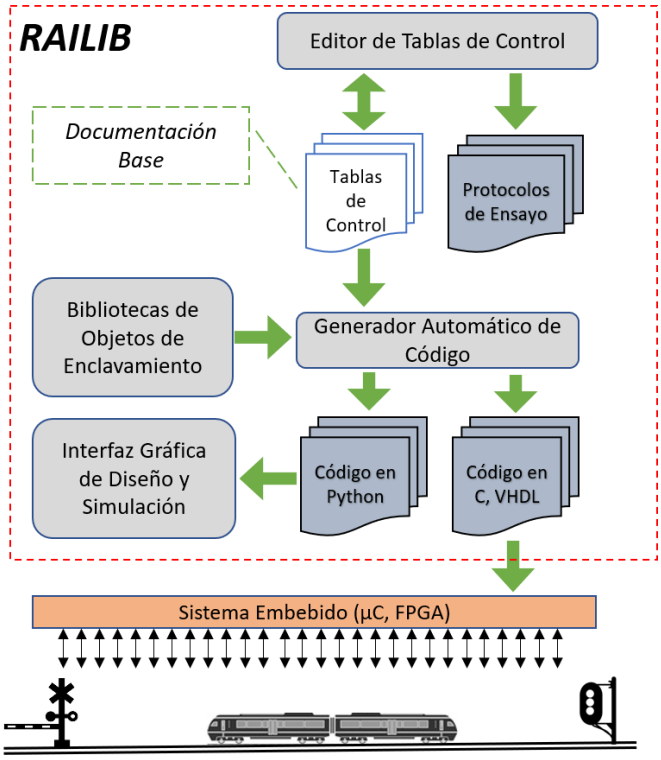


Fig 3. Flujo de trabajo para la herramienta desarrollada

Si bien el enfoque algebraico fue suficiente para representar las funciones más básicas, resultó ser poco flexible y difícil de escalar para agregar nuevas funcionalidades. Por lo tanto, se realizó un relevamiento del estado del arte en técnicas de modelado en sistemas de enclavamiento. Se encontraron lenguajes comerciales para modelado e implementación como EURIS [27] y ObjRail [28], donde se aprovecha el paradigma de la Programación Orientada a Objetos [29] para describir cada elemento.

Por otro lado, en [30] se presenta una metodología novedosa en la cual se combina el Desarrollo Basado en Componentes (CBD, por sus siglas en inglés) con el Desarrollo Basado en Modelos (MBD, por sus siglas en inglés), dando origen al Modelado Específico de Dominio Basado en Componentes (CBDSM, por sus siglas en inglés). Como se explica en [30], el enfoque CBDSM permite modelar, documentar o diseñar sistemas complejos a partir de componentes más sencillos, específicos del dominio de trabajo. Además, el comportamiento de cada componente puede ser verificado de manera individual, aportando flexibilidad, escalabilidad y confiabilidad al proceso de desarrollo.

Por lo tanto, se decidió desarrollar una biblioteca de objetos de enclavamiento, la cual define un conjunto de componentes de código desarrollados y verificados. Estos objetos facilitan el desarrollo de nuevos sistemas mediante su instanciación y conexión. La lógica global del sistema, indicada en la tabla de control del enclavamiento, define la interconexión de los componentes del modelo.

D. Generador Automático de Código

Como se mencionó en la sección II.C y se explica con detalle en [31], las funciones de seguridad que conforman la lógica vital de un sistema de enclavamiento se especifican mediante una o más tablas de control [18], similares a la ilustrada en la Fig. 3. Un *script* escrito en Python lee dicho documento, instancia los objetos a partir de la biblioteca de plantillas y los interconecta según indica la tabla.

Para que sea posible generar el código que implementa la lógica del sistema de enclavamiento en un determinado lenguaje (por ejemplo C o VHDL [32]), además del *script* de generación automática, debe existir la biblioteca de objetos de enclavamiento ya introducida en la sección III. A, donde cada componente del modelo está descrito a través de una plantilla en el lenguaje correspondiente. Como se explica en [33], el uso de plantillas garantiza que la flexibilidad y escalabilidad del modelo basado en componentes se mantenga para la generación automática de código. Además, las plantillas de código resuelven aspectos de implementación en bajo nivel necesarios para asegurar el funcionamiento del modelo en el sistema embebido final, ya sea basado en una FPGA o un procesador.

La Fig. 4 ilustra el flujo de generación de código. El *script* lee la tabla de control y escribe un nuevo archivo en el cual instancia los componentes de la biblioteca de plantillas en lenguaje ANSI C99. Por su compatibilidad con todos los compiladores comerciales existentes, el uso de este estándar permite la implementación del código generado en todo tipo de sistemas embebidos basados en microprocesadores.

Es importante remarcar que el código generado es independiente de los dispositivos físicos utilizados. Por lo tanto, puede utilizarse en sistemas embebidos tanto para el modelado como para la implementación final del sistema. Como se explica en la sección II.B y se muestra en la Fig. 2, la capa de infraestructura será la que provea la interfaz de comunicación con las etapas de potencia encargadas de manejar las señales, motores, relés y demás dispositivos en campo.

E. Interfaz Gráfica de Diseño y Simulación

Se han propuesto gran cantidad de algoritmos y métodos automáticos para la verificación formal de las reglas de seguridad en un enclavamiento [34], pero pocos trabajos [22-23] hacen hincapié en la posibilidad de simular el comportamiento del sistema de forma interactiva. Por esta razón se desarrolló una interfaz gráfica que permite, a partir del código generado para una tabla de control, construir su diagrama de señalamiento y simular su operación.

En la Fig. 5 se muestra la interfaz de simulación implementada. Fue desarrollada en Python 3 y emplea Qt5 [35] como librería gráfica. Posee dos modos de operación:

1. Modo Diseño: el usuario ubica los objetos sobre una grilla y edita sus propiedades (longitud de circuito de vía, tipo de cambio, señal o paso a nivel, etc.)

2. Modo Simulación: el usuario presiona los circuitos de vía para simular su ocupación y las señales para solicitar o cancelar rutas. La interfaz indica en todo momento el estado de los elementos.

La simbología empleada se construyó a partir de las interfaces de usuario utilizadas en enclavamientos comerciales [3-11] y las recomendaciones provistas por personal de Trenes Argentinos. Otras opciones del programa incluyen un manual de usuario construido mediante HTML (del inglés *Hyper Text Markup Language*) y la posibilidad de guardar y cargar diagramas previamente elaborados en formato JSON [36].

A nivel de la arquitectura de software, se buscó abstraer o independizar la interfaz gráfica de la complejidad y composición del enclavamiento, y de la naturaleza de cada capa del sistema. Para ello se creó una capa de abstracción o fachada [37] que se comunica con una biblioteca de objetos gráficos, los cuales implementan las funcionalidades de edición del diagrama (*drag and drop,* selección múltiple, etc.).

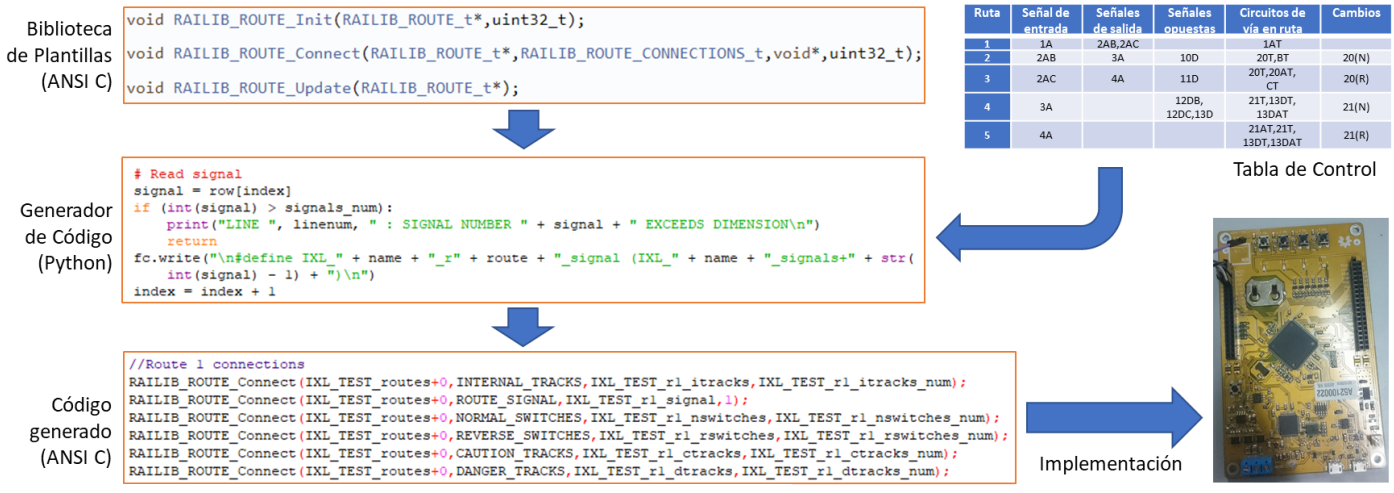


Fig. 4. Flujo de generación automática de código para un microcontrolador

F. Formato y Edición de Tablas de Control

A medida que se avanzó en este proyecto, fue necesario adoptar una estructura para las tablas de control que sirven como entrada al generador de código. Sin embargo, al comparar distintos trabajos [18], [19], [24] y bibliografía al respecto se encontró que no existe un único formato estándar para tablas de control, si bien todos los formatos detallan la misma información. Además, las tablas de control no son exhaustivas, ya que se centran en las rutas, pero dejan implícito lo referente a cambios de vía, pasos a nivel y otros elementos del sistema.

Por estas razones, se desarrolló un nuevo formato basado en documentación utilizada por Trenes Argentinos y en cuya elaboración participaron referentes de dicha empresa. Esto asegura que las tablas puedan ser elaboradas y/o comprendidas tanto por el equipo de desarrollo como por el usuario final. Este nuevo formato se basa en cinco tablas de control diferentes, cada una de las cuales especifica de forma exhaustiva ciertas funciones lógicas:

1. Tabla de rutas: es la tabla principal, y detalla las condiciones para activar cada ruta y sus señales asociadas.
2. Tabla de cambios de vía: aclara las condiciones de bloqueo de cambios de vía por ocupación.
3. Tabla de pasos a nivel: especifica el comportamiento de los pasos a nivel según ocupación de vía y rutas activas.
4. Tabla de despejes: detalla los circuitos de vía siguientes a una ruta según la posición de los cambios.
5. Tabla de aproximaciones: detalla los circuitos de vía anteriores a una ruta según la posición de los cambios.

La edición, carga y guardado de las tablas de control se realiza a través de una interfaz que también permite ejecutar la generación automática de código y realizar un chequeo sintáctico para buscar errores de confección. A través del editor también se puede generar un protocolo de ensayos de validación estandarizado por Trenes Argentinos para verificar el enclavamiento mediante pruebas de simulación o ensayos en campo.

IV. Resultados y Comparación con otros Trabajos

La solución desarrollada integra todo el flujo de trabajo desde la confección de la tabla de control que especifica al sistema hasta la generación de código para su implementación. El resultado es una herramienta de Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés) [38-39] de gran utilidad para el diseño e implementación de sistemas de enclavamiento sobre sistemas embebidos basados en FPGA o procesadores.

Si bien se relevaron numerosos trabajos académicos similares, no se encontraron métricas cuantitativas para la comparación de resultados. Sin embargo, se puede usar la Tabla II como un resumen en el cual se contrasta de forma clara el aporte de este trabajo, mediante una evaluación sobre cuatro ejes centrales: *Descripción de modelos, Generación de código, Verificación de modelos* y *Uso de interfaz gráfica.*

1. Descripción de modelos: la mayoría de los trabajos relevados emplean lenguajes ajenos al ámbito ferroviario para la descripción de los sistemas de enclavamiento, lo cual aumenta la complejidad de uso. Solo en dos casos se parte de documentación familiar para el usuario final [24-25]. Por otro lado, debido a la interacción continua con Trenes Argentinos, RAILIB implementa todas las funciones lógicas de uso corriente en sistemas de enclavamiento. En contraste, los demás trabajos se encuentran limitados por los lenguajes de chequeo de modelos empleados.

2. Generación de código: De los trabajos relevados que ofrecen esta posibilidad [17,30,31,33,38-39,44-45], solo [31] está orientado a sistemas ferroviarios, y ninguno genera documentación legible por el usuario.

3. Verificación de modelos: la mayor parte de los trabajos relevados emplean métodos formales de validación [19,21,31,33-34] o no integran este paso en el flujo de trabajo [24-25,30,38-19,44-45]. En cambio, la herramienta presentada permite combinar la simulación interactiva con protocolos estandarizados de trabajo.

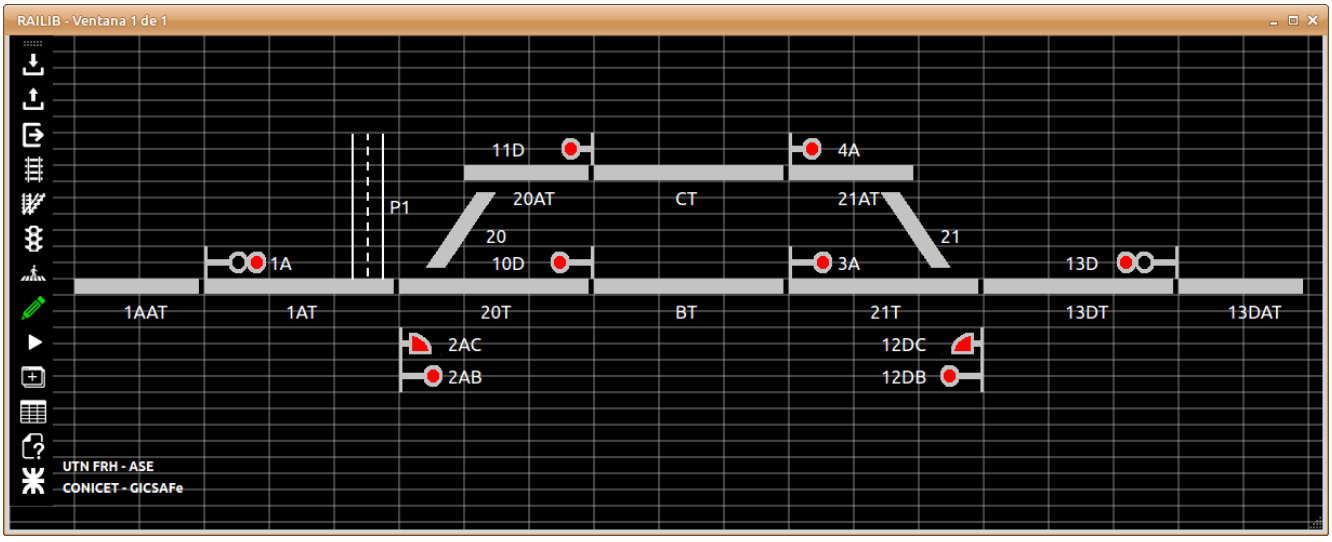


Fig. 5. Interfaz en Modo Diseño, mostrando el layout de la Fig. 1, al cual se ha agregado un paso a nivel

TABLA II
COMPARACIÓN DE TRABAJOS Y HERRAMIENTAS RELEVADAS

Trabajo relevado	Ámbito de aplicación	Interfaz Gráfica	Descripción de modelos	Verificación de modelos	Generación de código
[17]	Universal	Aldec Active-HDL (comercial)	Finite States Machine	Visualización de formas de onda	VHDL
[19]	Ferroviario	No	Abstract States Machine	Model Cheking	No
[21]	Ferroviario	Solo simulación	EVALSPN	Model Checking	No
[22]	Ferroviario	No	Scala	Simulación de eventos aleatorios	No
[23]	Ferroviario	Solo visualización de maqueta	Diagrama de flujo	Simulación mediante maqueta	No
[24]	Ferroviario	Solo para diagramar layout	Layout	No	No; genera tabla de control (.txt)
[25]	Ferroviario	MS Visio (comercial)	Documentación de ingeniería	No	No; genera reportes de ensayo (.docx)
[30]	Universal	Ad-hoc	ECSML	No	C
[31]	Ferroviario	I-Logix Statemate (comercial)	Statecharts	Simulación y Model Checking	C, ADA
[33]	Universal	Open Architecture Ware (comercial)	UML, XML, EMF	Model Checking	Según extensiones instaladas
[34]	Ferroviario	No	Ladder logic	Model Checking	No
[38-39]	Universal	Matlab – Artisan (comerciales)	UML/Simulink	No	C, C++, Java
[44-45]	Universal	GenERTiCA (propia)	UML	No	VHDL; extensible
RAILIB	Ferroviario	Propia, para todo el flujo de trabajo	Tablas de Control	Simulación en base a Protocolos de Ensayo estandarizados	C, Python y VHDL; extensible. Genera Protocolos de Ensayo (.pdf)

4. Uso de interfaz gráfica: varios trabajos utilizan herramientas comerciales no integradas a la aplicación [17,25,31,33,38-39] con las consecuentes desventajas de los costos adicionales y la pérdida de control sobre el flujo completo de trabajo. En otros casos no se dispone de interfaz gráfica [19,22,34], o solo abarca un subconjunto limitado del proceso [21,23-24]. Además, solo RAILIB permite visualizar sistemas de forma remota vía web [40].

V. CONCLUSIONES

El enfoque propuesto en este trabajo posee varias ventajas respecto al modelado *ad-hoc* de enclavamientos y a otras herramientas relevadas. En primer lugar, parte de un documento estándar de requisitos que asegura que la solución generada cumple las especificaciones del cliente. Además, el uso de una biblioteca de componentes verificados y certificados [41] reduce el tiempo de desarrollo y el esfuerzo de certificación. También facilita el agregado de nuevas funcionalidades, dado que solo se requiere extender la clase asociada a un objeto dado y, si es necesario, modificar el generador de código. Por otro lado, agiliza el proceso de desarrollo de enclavamientos, ya que unifica en una sola interfaz su diagramación, simulación e implementación.

Finalmente, el entorno diseñado para la generación automática de código fuente permite la implementación de sistemas de enclavamiento ferroviario sobre plataformas embebidas como una FPGA, un microcontrolador, etc.

VI. AGRADECIMIENTOS

Los autores agradecen a la empresa Trenes Argentinos por sus aportes que guiaron el desarrollo del proyecto. También se desea agradecer al Departamento de Ingeniería Electrónica de la Facultad Regional Haedo, Universidad Tecnológica Nacional y al Rectorado de la misma que a través de los fondos del proyecto UTN PID N° 5253 (“Sistema de enclavamiento electrónico para cambios de vía ferroviarios”) hicieron posible este proyecto. Finalmente, se agradece el financiamiento provisto por la Universidad de Buenos Aires a través del subsidio UBA PDE N°26/2019.

VII. REFERENCIAS

[1] Comisión Nacional de Regulación del Transporte (CNRT), *“Informe estadístico 2018 de la red ferroviaria argentina*” (2019) [Online]. Disponible: https://www.argentina.gob.ar/cnrt/estadisticas-ferroviarias

[2] U.S. Central Intelligence Agency (CIA), “*The World Factbook Archive – Country Comparison :: Railways* ” (2018) [Online] . Disponible: https://www.cia.gov/library/publications/the-world-factbook/fields/384rank.html

[3] Siemens, “*Trackguard Westrace Mk ll Flexible safety processor”*(2014).[Online]. Disponible: https://www.mobility.siemens.com/mobility/global/SiteCollectionDocuments/en/rail-solutions/rail-automation/electronic-interlockings/trackguard-westrace-mk2-en.pdf

[4] Siemens, “*Trackguard WESTLOCK. Cost-effective, highly available and reliable*“ (2013) [Online]. Disponible:

https://www.mobility.siemens.com/mobility/global/SiteCollectionDocuments/en/rail-solutions/rail-automation/electronic-interlockings/trackguard-westlock-en.pdf

[5] Alstom, "*Alstom Smartlock equipment at the heart of modular signalling project in the UK* "(2011).[Online]. Disponible: https://www.alstom.com/press-releases-news/2011/3/Alstom-Smartlock

[6] Hitachi Ltd., "*Computer based interlocking*"(2019).[Online]. Disponible: http://sts.hitachirail.com/en/products-services/business-segments/computer-based-interlocking

[7] AZD Praha S.R.O, "*Systémy pro kolejovou dopravu. Elektronické Stavedlo TYP ESA® 33*" (2019) [Online]. Disponible: https://www.azd.cz/admin-data/storage/get/198-

[8] Thales Group, "*LockTrac 6111 ESTW L90*" (2019) [Online]. Disponible: https://www.thalesgroup.com/en/route-control-systems

[9] Railway Technology, "*EBI Lock Computer-Based Interlocking Systems*" (2016) [Online]. Disponible:
https://www.railway-technology.com/products/ebi-lock-computer-based-interlocking-systems/

[10] Kyosan Electric Mfg. Co., "*Railway Signaling Solutions – Interlocking Equipment*" (2019) [Onlmine]. Disponible:
https://www.kyosan.co.jp/english/product/signal03.html

[11] Hima Paul Hindebrandt GmbH, "*COTS Rail Applications*" (2019) [Online]. Disponible:
https://www.hima.com/en/industries-solutions/cots-rail-references

[12] Institution of Railway Signal Engineers (IRSE), Chapters 19 : "*Points and other track features*" and 20 : "*Train Detection*" in "*Study Guide for Module 3 – Signalling Principles*", pp. 63-73.

[13] *Reglamento Interno Tecnico Operativo* (RITO), Ferrocarriles del Estado Argentino, 1993. Título IV: *Señales y Cambios*, pp. 57-91.

[14] *Functional safety of electrical/electronic/programmable electronic safety-related devices – part 1 : General requirements*, IEC 61508-1, International Electrotechnical Commission (IEC), section 7.6.2.9 (Safety Integrity Level), 2010

[15] *Railway applications – Communication, signalling and processing systems – Safety related electronic systems for signalling*, CENELEC – EN 50129, Comité Europeo de Normalización Electrotécnica (CENELEC), nov. 2018

[16] R. Dobias, H. Kubatova, "*FPGA Based Design of the Railway's Interlocking Equipments*", Department of Computer Science and Engineering, Czech Technical University, Prague, Czech Republic, 2004.

[17] P. Kawalek, M. Rzysko, "*Modern methods in railway interlocking algorithms design",* ACM Journal of Microprocessor & Microsystems, Volume 44, Issue C, 2016.

[18] Institution of Railway Signal Engineers (IRSE), "*Chapter 14 : Interlocking Control Tables*" in "*Study Guide for Module 3 – Signalling Principles*", pp. 36-44.

[19] K. Winter, N.J. Robinson, "*Modelling Large Railway Interlockings and Model Checking Smaller Ones*", Software Verification Research Centre, University of Queensland, Brisbane, Australia, 2003.

[20] *Railway Applications – The Specification and Demostration of Reliability, Availability, Maintainability and Safety (RAMS)- Part 1 : Generic RAMS Process* , CENELEC - EN 50126 - 1, Comité Europeo de Normalización Electrotécnica (CENELEC), oct. 2017

[21] K. Nakamatsu, Y. Kiuchi, W. Y. Chen and S. L. Chung, *"Intelligent railway interlocking safety verification based on annotated logic program and its simulator",* IEEE International Conference on Networking, Sensing and Control, 2004, Taipei, Taiwan, 2004, pp. 694-699 Vol.1

[22] Q. Cappart, C. Limbrée, P. Schaus, J. Quilbeuf, L. Traonouez and A. Legay, *"Verification of Interlocking Systems Using Statistical Model Checking"*, 2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE), Singapore, 2017, pp. 61-68

[23] P. Samootrut, M. Lertwatechakul, S. Tongkrairat, T. Anuwongpinit, A. Vijittanasan and V. Chutchavong, *"On development of train control and signaling simulation",* 2016 16th International Conference on Control, Automation and Systems (ICCAS), Gyeongju, 2016, pp. 671-674.

[24] U. Yıldırım, M.S. Durmuş, M.T. Söylemez,*"Automatic Interlocking Table Generation for Railway Stations using Symbolic Algebra",* IFAC Proceedings Volumes, Volume 45, Issue 24, 2012, pp. 171-176

[25] Z. Yong and S. Yaowei*, "A method for automatic generation of track layout graph based on visio secondary development*" 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP), Beijing, 2013, pp. 182-187

[26] R.J. Wilson, "*Matrix representations*" in "*Introduction to Graph Teory*", 4th ed., Addison-Wesley Longman Ltd., 1996, pp 14-15.

[27] F. van Dijk, W. Fokkink, G. Kolk, P. van de Ven, B. van Vlijmen, 1998, "*EURIS, a Specification Method for Distributed Interlockings*" presented at SAFECOMP 98, pp. 296-305

[28] ECM SPA, "*HMR9 Software Overview*" [Online]. Disponible: http://www.ecmre.com/en/sistemi/hmr9/hmr9_software

[29] M. Weisfeld, "*The Object-Oriented Thought Process*", 3rd ed., U.S., Addison-Wesley, 2009.

[30] Z. Shu, Di Li, Y. Hu, F. Ye, S. Xiao and J. Wan, *"From Models to Code: Automatic Development Process for Embedded Control System"*, 2008 IEEE International Conference on Networking, Sensing and Control, Sanya, 2008, pp. 660-665

[31] M. Banci, A. Fantechi, S. Gnesi, "*Some experiences on formal specification of Railway Interlocking Systems using Statecharts*", Formal Methods and Tools Group (ISTI-CNR, Pisa) and Systems and Informatics Department (Università degli Studi di Firenze), Italy, 2005.

[32] *IEEE Standard VHDL Language Reference Manual,* IEEE 1076-1993, IEEE Standards Board and American National Standards Institute (ANSI), abr. 1994.

[33] M. Regensburger, C. Buckl, A. Knoll and G. Schrott, *"Model Based Development of Safety-Critical Systems Using Template Based Code Generation",* 13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007), Melbourne, Qld., 2007, pp. 89-92.

[34] P. James, "*SAT-based Model Checking and its applications to Train Control Systems*", MRes. thesis, Department of Computer Science, Swansea University, 2010.

[35] The Qt Company, *Qt Project Homepage* [Online]. Disponible:
https://www.qt.io/

[36] *The JSON Data Interchange Syntax,* ECMA-404 Standard, ECMA International, 2nd edition, ene. 2017.

[37] E. Gamma, R. Helm, R. Johnson, J. Vlissides, "*Patrones de Diseño*", 1era edición, Addison-Wesley, 2003.

[38] Reichmann C., Kühl M., Müller-Glaser K.D. (2003) *"An Overall System Design Approach Doing Object-Oriented Modeling to Code-Generation for Embedded Electronic Systems".* In: Pezzè M. (eds) Fundamental Approaches to Software Engineering. FASE 2003. Lecture Notes in Computer Science, vol 2621. Springer, Berlin, Heidelberg

[39] M. Kuhl, C. Reichmann, I. Protel and K. D. Muller-Glaser, *"From object-oriented modeling to code generation for rapid prototyping of embedded electronic systems",* Proceedings 13th IEEE International Workshop on Rapid System Prototyping, Darmstadt, Germany, 2002, pp. 108-114.

[40] A. Laiuppa, B. Palacios, M. Amado, G. Ramoscelli, L. Dórdolo et al, "*Diseño de un sistema modular para el monitoreo y gestión de sistemas ferroviarios*", CONICET-GICSAFe, presentado en el X Congreso Argentino de Sistemas Embebidos (CASE), jul. 2019.

[41] TÜV-Rheinland, *"Certified Function Block (CFB) H-MO Library for the SILworX programming environment, according to the current version list"* (2015) [Online]. Disponible*:*
https://www.certipedia.com/fs-products/certificates?cert_id=2414

[42] Proyecto CIAA (Computadora Industrial Abierta Argentina), Homepage [Online]. Disponible*:* http://www.proyecto-ciaa.com.ar/

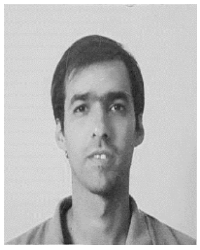
[43] M. Menéndez, F. Larosa, N. Álvarez, R. Ghignone, A. Lutenberg, "*Diseño de un sistema crítico de enclavamiento ferroviario implementado mediante FPGA*", CONICET-GICSAFe, presentado en el X Congreso Argentino de Sistemas Embebidos (CASE), jul. 2019.

[44] T. G. Moreira, M. A. Wehrmeister, C. E. Pereira, J. Pétin and E. Levrat, *"Automatic code generation for embedded systems: From UML specifications to VHDL code",* 2010 8th IEEE International Conference on Industrial Informatics, Osaka, 2010, pp. 1085-1090.

[45] M. A. Wehrmeister, E. P. Freitas, C. E. Pereira and F. Rammig, *"GenERTiCA: A Tool for Code Generation and Aspects Weaving",* 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), Orlando, FL, 2008, pp. 234-238.

Ramiro A. Ghignone es ingeniero electrónico egresado de la Universidad Tecnológica Nacional – Facultad Regional Haedo (UTN-FRH) a principios de 2019.

Entre 2016 y 2018 participó como becario del grupo de I+D en Aplicaciones de Sistemas Embebidos (ASE) de esa facultad. Allí trabajó en diversos proyectos sobre sistemas de navegación satelital e inercial, y sobre optimización de sistemas embebidos en bajo nivel. Actualmente se desempeña como integrante del mismo grupo de investigación, donde posee una Dedicación Simple como Ayudante de Trabajos Prácticos, y donde coordina el desarrollo del proyecto RAILIB. Desde fines de 2018 participa en el grupo CONICET-GICSAFe, donde se especializa en procesamiento digital de señales y en sistemas de enclavamiento ferroviario.

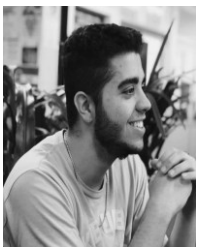


Cristian F. Falco es ingeniero electrónico graduado de la Universidad de Buenos Aires (UBA) en el año 2006, y desde 2011 es miembro asociado de IRSE (*Institution of Railway Signal Engineers*).

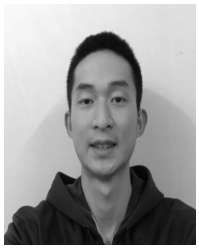
Entre 2008 y 2014 trabajó como Jefe de Obra a cargo de proyectos de electrificación y señalamiento ferroviario. Desde 2015 ejerce el rol de Jefe de Área en la Gerencia de Obras de Señalamiento de la empresa estatal ADIF (Administración de Infraestructuras Ferroviarias) y actualmente desempeña tareas como Especialista en Señalamiento Ferroviario en comisión en la Secretaría de Planificación del Ministerio de Transporte de la Nación. También es Jefe de Trabajos Prácticos en la asignatura "Señales y Sistemas de Cambios" de la carrera de Ingeniería Ferroviaria impartida en la UTN-FRH, y participa en el grupo ASE aportando su experiencia para la mejora continua del proyecto RAILIB.

Facundo S. Larosa es ingeniero electrónico, graduado en 2009 de la UTN-FRH En 2010 recibió una distinción de la Academia Nacional de Ingeniería por su promedio académico, y en 2017 obtuvo el título de Magister en Ingeniería en Sistemas Embebidos otorgado por la UBA. Actualmente se encuentra trabajando en su doctorado sobre el mismo campo de investigación.

Hoy en día se desempeña como Profesor Adjunto con dedicación exclusiva en la UTN-FRH y en la Maestría en Sistemas Embebidos (MSE) dictada por la UBA. También es Director del grupo ASE de la UTN-FRH y miembro del grupo CONICET-GICSAFe, donde trabaja en la investigación y desarrollo de sistemas críticos sobre FPGA.



Hernán P. Mendes Gouveia es estudiante de la carrera de Ingeniería Electrónica en la UTN-FRH. Desde el año 2018 colabora con el grupo ASE, donde actualmente es becario de I+D. Su trabajo en este equipo comprende el desarrollo de interfaces de usuario para el proyecto RAILIB.



Leandro A. Chang es estudiante de la carrera de Ingeniería Electrónica en la UTN-FRH. Desde el año 2018 colabora con el grupo ASE, donde actualmente es becario de I+D. Allí está a cargo del diseño gráfico de la interfaz para el proyecto RAILIB.

Martín N. Menéndez es ingeniero electrónico graduado de la UBA en 2017. En 2018 obtuvo el título de Especialista en Sistemas Embebidos otorgado por la misma institución, y actualmente se encuentra cursando la Maestría en la misma temática.

Desde fines del 2017 se desempeña como miembro del grupo CONICET-GICSAFe, en el cual desarrolló proyectos de monitoreo de número de pasajeros y de implementación de sistemas de enclavamiento sobre FPGA.

Ariel Lutenberg es ingeniero electrónico graduado de la UBA en 2006. En 2009 obtuvo su diploma de Doctor en Ingeniería de la misma universidad, con mención de honor "*Summa Cum Laude"*.

En 2018 recibió el Gran Premio INNOVAR por el desarrollo de un sistema de monitoreo remoto de barreras ferroviarias automáticas, y desde 2019 forma parte del jurado del Premio INNOVAR. Por ese proyecto también recibió el primer premio del Concurso "Desafío Eureka". Actualmente es Profesor Adjunto con dedicación exclusiva en la FI-UBA, Investigador Adjunto del CONICET, Director del Laboratorio de Sistemas Embebidos de la UBA y Director de la Carrera de Especialización en Sistemas Embebidos y la Maestría en Sistemas Embebidos de la misma Universidad. A su vez, es Presidente de la Asociación Civil para la Investigación, Promoción y Desarrollo de los Sistemas Electrónicos Embebidos.