Proceso para el diseño de la arquitectura software en un sistema crítico ferroviario según la norma EN-50128

Irrazábal Emanuel, Sambrana Ivan, Pinto Luft Cristian

Grupo de Investigación en Innovación de Software y Sistemas Informáticos
Departamento de Informática. FACENA
Universidad Nacional del Nordeste
emanuelirrazabal@gmail.com, sambranaivan@gmail.com, cristianpl777@gmail.com

Resumen. Los sistemas electrónicos para la seguridad vial de trenes y subtes en Argentina son importados y tienen un costo muy alto. El desarrollo de este tipo de sistemas es complejo, en parte por ser sistemas críticos que requieren el uso y seguimiento de una gran cantidad de normativas internacionales. El objetivo de este trabajo es presentar la construcción de un proceso para el diseño de la arquitectura software en sistemas críticos ferroviarios conforme a la norma EN 50128. En particular teniendo en cuenta las secciones 7.3 de dicha normativa y la tabla anexa de técnicas propuestas. El soporte tecnológico de este proceso se realizó a partir de la herramienta Eclipse Process Framework y un ecosistema de herramientas para la gestión colaborativa de proyectos. Finalmente se detalla un ejemplo de utilización del proceso en un proyecto real llevado adelante para Trenes Argentinos.

1 Introducción

En Argentina cada día tres millones de personas viajan en tren o subte y el 10% del PBI se moviliza por ferrocarril [1]. Sin embargo, todos los sistemas electrónicos para la seguridad vial de trenes y subtes son importados y tienen un costo muy alto. Por ejemplo, un sistema de barrera automático cuesta hasta 200.000 dólares y un sistema de control de velocidad más de 100.000 dólares. Esto aumenta el desafío de mantener y actualizar las soluciones ferroviarias provocando muchas veces el uso de tecnologías inadecuadas. Esta situación ha favorecido que ocurran terribles accidentes [2] y ha urgido al Estado a adquirir en el exterior trenes y sistemas de seguridad ferroviaria, lo que implica enormes gastos y depender de tecnología importada [3], [4], [5]. Pero en la mayoría de los casos los accidentes se podrían haber evitado mediante el uso de sistemas electrónicos apropiados, que hoy en día son habituales en países con alto desarrollo tecnológico. Sin embargo, como se mencionó anteriormente, en la actualidad estos sistemas no se desarrollan en la Argentina.

Existen, sin embargo proyectos de investigación y de extensión que se encuentran actualmente trabajando en ello. Un ejemplo de ello son los siguientes: Proyecto Desarrollo de Estratégico UBA Nº23 “Controlador electrónico para barreras automáticas ferroviarias con nivel de integridad de seguridad certificable hasta SIL4", UBA Nº 08 “Análisis comparativo entre tecnologías electrónicas programables y no programables en la implementación de sistemas críticos ferroviarios. Caso de estudio:

sistema de hombre vivo" o el Proyecto de Investigación PI-F17-2017 "Análisis e implementación de tecnologías emergentes en sistemas computacionales de aplicación regional.", acreditado por la Secretaría de Ciencia y Técnica de la Universidad Nacional del Nordeste (UNNE).

El desarrollo de este tipo de sistemas es complejo, en parte por ser sistemas críticos que requieren el uso y seguimiento de una gran cantidad de normativas internacionales [6]. En particular, los sistemas ferroviarios son complejos, compuestos por distintos componentes software, hardware y humanos, que interactúan con su entorno de maneras muy variadas. Un fallo en uno de estos componentes o subsistemas puede llegar a tener asociados distintos niveles de peligros, pudiendo causar pérdidas financieras, daño al equipamiento, daños ambientales, lesiones a personas o en los peores casos pérdidas de vidas humanas. Por estos motivos dichos sistemas se encuentran regulados con distintas leyes y normativas cuyo fin es preservar los recursos anteriormente mencionados [6]. Algunos de los principales organismos que regulan esta actividad son el Comité Européen de Normalisation Electrotechnique (CENELEC) en Europa o la International Electrotechnical Commission (IEC) en América.

Una de las características más importantes de los sistemas que estas normas intentan reforzar durante todo su ciclo de vida son las de fiabilidad, disponibilidad, mantenibilidad y seguridad (RAMS por sus siglas en inglés). Las principales normas propuestas por el CENELEC orientadas a la resolución de la problemática explicada anteriormente son las siguientes:

- EN 50126: Aplicaciones ferroviarias. La especificación y demostración de Fiabilidad, Disponibilidad, Mantenibilidad y Seguridad (RAMS). Esta norma se orienta principalmente al cumplimiento de las características RAMS del sistema en general.
- EN 50128: Aplicaciones ferroviarias. Sistemas de comunicación, señalización y procesamiento. Software para sistemas de control y protección del ferrocarril. Esta norma se centra principalmente en la calidad de los aspectos software de los sistemas de ferrocarriles.
- EN 50129: Aplicaciones ferroviarias. Sistemas de comunicación, señalización y procesamiento. Sistemas electrónicos relacionados con la seguridad para la señalización. Esta norma se centra principalmente en los aspectos de calidad del hardware de los sistemas de ferrocarriles.

En la actualidad, grandes organizaciones como la NASA Ansaldo Signal [7] o Siemens Rail Transportation [8] utilizan una combinación de metodologías y formas de trabajo provenientes de distintos campos del conocimiento para lograr dicha vinculación, y de esta manera mejorar la calidad y seguridad de los sistemas críticos que desarrollan, dedicando tiempo, recursos y esfuerzo a esta tarea. Para ello es indispensable el desarrollo de procesos y un sistema de gestión de calidad conforme la normativa EN 50128. En este sentido, cada vez se comprende más en la ingeniería de software el papel de la arquitectura de software como medio para gestionar la

complejidad y lograr cualidades emergentes, como la modificabilidad o la mantenibilidad [9].

Desde el punto de vista de la normativa ferroviaria EN 50128, en la sección 7.3 se declaran las características para la construcción de una arquitectura software. Esta debe expresarse y estructurarse de forma que sea completa, coherente, clara, precisa, inequívoca, verificable, que se pueda someter a ensayo, que se pueda mantener y que sea realizable. Así mismo en la sección 7.3 también se hace referencia a un conjunto de tablas anexas que enumeran técnicas mínimas a ser cumplidas de acuerdo con el nivel de seguridad.

El objetivo de este trabajo es presentar la construcción de un proceso para el diseño de la arquitectura software en sistemas críticos ferroviarios conforme a la norma EN 50128. En particular teniendo en cuenta las secciones 7.3 de dicha normativa y la tabla anexa de técnicas propuestas. El soporte tecnológico de este proceso se realizó a partir de la herramienta Eclipse Process Framework (EPF) y un ecosistema de herramientas para la gestión colaborativa de proyectos. También se muestra como caso de estudio un ejemplo de utilización del proceso en un proyecto real llevado adelante para Trenes Argentinos.

Además de esta introducción el artículo se divide en 5 secciones. En la sección 2 se tratan los trabajos relacionados, mientras que la sección 3 presenta el desarrollo del proceso y la sección 4 el ejemplo de uso un proyecto real. Finalmente la sección 5 detalla las conclusiones y los trabajos futuros.

2 Trabajos relacionados

A continuación se resumen otros trabajos relacionados con el uso de procesos en el desarrollo de la arquitectura software para sistemas críticos ferroviarios.

Por un lado, en [10] se detallan las técnicas utilizadas para el diseño de la arquitectura del Sistema de enclavamiento desarrollado dentro del Proyecto de señalización nacional turco. Asimismo, en [11] se hace hincapié en la técnica de la programación de aserción de fallas aplicado a la arquitectura de un sistema de enclavamiento perteneciente al del Proyecto de señalización nacional turco. Finalmente en [12] se lleva adelante el desarrollo de un sistema de detección de trenes basado en video, donde se utiliza un set de técnicas que se consideran suficientes para identificar todos los componentes del software. En [13] en el marco del desarrollo del proyecto “Automatic Train Protection” se centra en la utilización de métodos formales para la descripción clara e inequívoca de los requerimientos y en el modelado para representar la arquitectura de manera visual, clara y legible.

Estos ejemplos están desarrollados siguiendo la normativa EN 50128 y, en particular intentando cumplir con las técnicas de diseño de arquitectura software descriptas en la tabla A.3 de la normativa ferroviaria. Esta tabla lista una gran cantidad de técnicas pero define un subconjunto mínimo para lograr diferentes niveles

de seguridad. El subconjunto mínimo de técnicas son las siguientes: programación defensiva, códigos detectores de errores, programación con aserciones, programación con múltiples versiones, interfaz definida completamente y metodología estructurada

Y además es altamente recomendable, de acuerdo con el tipo de proyecto, el uso de modelado y el desarrollo con métodos formales. En la sección 3 de este artículo se describe el proceso que busca cumplir con los apartados 7.3.4.2 hasta 7.3.4.12 de la normativa EN 50128 y con el conjunto de técnicas mínimas referenciadas en el apartado 7.3.4.14 a partir de la tabla A.3. En la Tabla 1 se resumen las técnicas utilizadas por cada uno de los artículos. Puede verse que una de las técnicas más utilizadas es la programación defensiva.

Tabla 1. Tabla comparativa de técnicas utilizadas.

Técnicas\Artículos	Turkish National Signalization Project.[10]	Video Based Train Detection.[12]	Automatic Train Protection.[13]	Railway Interlocking Design. [11]
Programación defensiva	X	X		X
Códigos detectores de errores		X		X
Programación con aserciones				X
Programación con múltiples versiones	X			
Interfaz definida completamente		X		
Métodos Formales			X	
Modelado		X	X	
Metodología Estructurada		X		

3 Proceso de diseño de la arquitectura software según EN 50128

A continuación se describe el proceso diseñado en este trabajo para desarrollar la arquitectura software en sistemas críticos, implementando las buenas prácticas indicadas en el apartado 7.3 “Arquitectura y Diseño” de la normativa EN 50128 para alcanzar un determinado Nivel de Integridad de Seguridad (o SIL por sus siglas en inglés). Los niveles de integridad de seguridad (SIL) definen el orden de magnitud en la reducción del riesgo de una falla peligrosa. Hay cuatro niveles de SIL definidos en la norma IEC 61508, desde SIL 1 hasta SIL 4; siendo este último el más riguroso [14]

La entrada principal para el desarrollo del proceso son los requisitos software. En la Fig 1 se muestra el proceso para la documentación de los requisitos software. Los pasos del proceso se encuentran detallados en el Eclipse Process Framework (EPF). Y

se mantienen las plantillas para generar la documentación en la plataforma Google Drive.

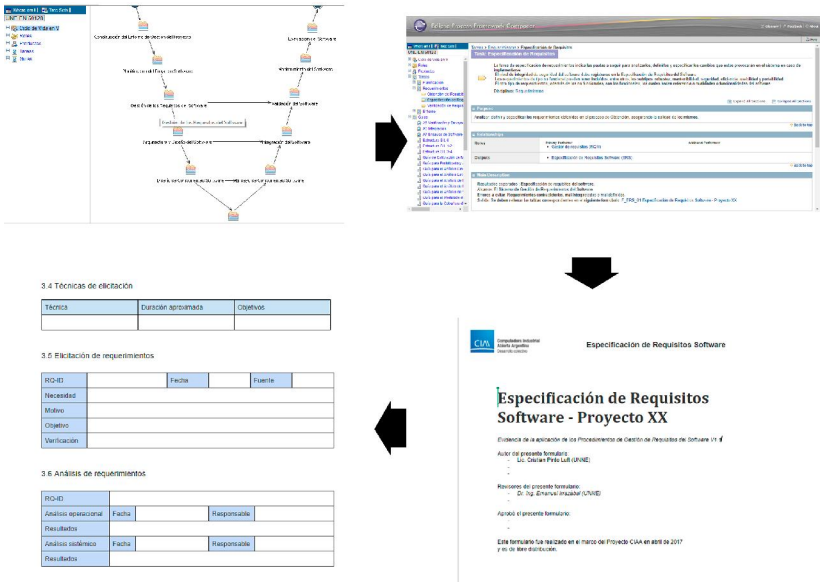

Fig 1. Ejemplo de proceso para la documentación de requisitos software construido en la herramienta EPF.

El primer paso es el estudio de la viabilidad de los requisitos del software. Se toma cada requisito y el nivel de SIL software requerido de acuerdo con la especificación de requisitos software obtenida del proceso anterior. En la Tabla 2 se identifican los apartados a completar. El rol de Diseñador es el encargado de estudiar la viabilidad.

Tabla 2. Análisis de viabilidad de Requisitos del Software. Ejemplo de uso.

	Nivel SIL Requerido	**Viabilidad/nivel SIL recomendado**
RQ-ID-4	**2**	Viabilidad aprobada.

El siguiente paso es identificar las interacciones entre los componentes Software/hardware, el nivel de interacciones que detallan la importancia de las mismas, desde 1 (sin interacción hasta 5 (interacción crítica). En la Tabla 3 se identifican los apartados a completar.

Tabla 3. Nivel de interacción entre componentes Hardware/Software. Ejemplo de uso.

	Módulo SPI Memoria SD	Sensor Digital Brazo Bajo	Sensor Digital Brazo en Alto
Librería de sistema de Archivos - Fatfs	**5**	**1**	**1**

A continuación es necesario determinar para cada componente software identificado previamente si este es nuevo o ya existía previamente, si se ha validad de forma previa y de ser así, sus condiciones de validación, también el nivel SIL del componente software. En la Tabla 4 se identifican los apartados a completar.

Tabla 4. Descripción preliminar de componentes software. Ejemplo de uso.

	Antigüedad [Nuevo o Prexistente]	**Condición de validación**	**Nivel SIL del componente**
Librería de sistema de Archivos – FatFs	Preexistente	El fabricante provee estadísticas sobre distintos procesadores.	Por Determinar

En el siguiente apartado la norma establece mantener la gestión de la configuración de cada componente software identificando que subconjuntos de requisitos del software se cubren en cada versión. En la Tabla 5 se identifican los apartados a completar. La gestión de la configuración se lleva adelante con la herramienta de gestión colaborativa Redmine extendida con el complemento del gestor de versiones Git. Es decir, se consigue trazabilidad entre los requisitos, el diseño y el código fuente.

Tabla 5. Gestión de la configuración de componentes software. Ejemplo de uso.

Componente software	**Versión**	**Requisito Software Asociado**
Librería de sistema de Archivos - FatFs	R.0.13b	RQ-ID 4

A continuación se detallan los pasos para gestionar los componentes software preexistentes. En la Tabla 6 se identifican los apartados a completar. Por un lado los requisitos software relacionados, tal y como fuera descripto para los nuevos componentes. Por otro lado las interfaces con otros componentes software.

Tabla 6. Documentación de Software preexistente. Ejemplo de uso.

Componente	**Requisito software asociado**	**Hipótesis relativas al entorno**	**Interfaces con otros componentes**
Librería de sistema de Archivos - FatFs	RQ-ID 4	Información brindada por el fabricante. -Independiente de la plataforma. -Seguro para RTOS. -Tamaño del sector variable. -Múltiples páginas de códigos incluyendo DBCS.	Módulo de Escritura SD

La técnica “modelado” descrita en el apartado “tabla A.3” de la norma EN 50128 recomienda documentar a partir de modelos el diseño de la arquitectura. Para la

solución propuesta en este trabajo se utiliza un diagrama de contexto [15]. La Fig 2 muestra un ejemplo mostrando la relación entre interfaces. La aplicación de esta técnica de modelado favorece la implementación de otras 2 técnicas mínimas requeridas para nivel de SIL 3 y SIL4 como son la utilización de metodología estructurada y la completa definición de las interfaces del sistema.

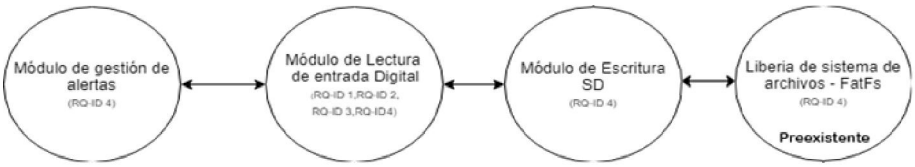

Fig 2. Diagrama de contexto de componente de software preexistente con interfaces definidas con otros componentes software.

En la Tabla 7 se analizan las posibles fallas y consecuencias de cada componente, como también describir las estrategias de detección y protección cuando ocurre un fallo.

Tabla 7. Estudio del riesgo del software preexistente

Componente	**Posible Falla**	**Consecuencias**	**Estrategia de detección**	**Estrategia de protección**
Librería de sistema de Archivos - FatFs	Cambio en la configuración del sistema de archivos – ante una actualización del fabricante	Información no legible en destino	Se debe implementar en el plan de pruebas ante una actualización del fabricante	Aplicación de técnicas de **programación defensiva** para verificar la información luego de la escritura en memoria SD

En la Tabla 8 se realiza una descripción detallada de cada uno de los componentes software preexistentes, lo suficientemente precisa y completa.

Tabla 8. Descripción de componente software preexistente. Ejemplo de uso.

Componente Software Preexistente – CSP-1	
Descripción	"FatFs es un módulo de sistemas de archivos FAT/exFAT genérico para pequeños sistemas embebidos." Como lo describe el fabricante.
Funciones	• Gestión y Acceso a archivos y directorios • Gestión de volúmenes y configuración del sistema de archivos.
Restricciones Hardware	No especificado por el fabricante.

Componente Software Preexistente – CSP-1	
Restricciones Software	No especificado por el fabricante
Pruebas	La librería dispone de test desarrollados por el fabricantes
Objetivo	Escritura en tarjetas de memoria SD a travez de interfaz SPI
Propiedades	FatFs es un módulo gratuito y abierto para educación, investigación y desarrollo.
Comportamiento	
Características	Última actualización 7 de abril de 2018

El siguiente paso es describir las pruebas para cada componente con las que se verificará el cumplimiento del nivel de SIL requerido. En la Tabla 9 se identifica los apartados a completar. Cuando el software construido por componentes con diferentes nivel de SIL, se debe tratar a todos los componentes software como si tuvieran el más alto de estos niveles, a menos que se disponga de pruebas de independencia entre los componentes.

Tabla 9. Pruebas de nivel de integridad. Ejemplo de uso.

	Nivel de SIL	Plan de Prueba	Estado
Librería de sistema de Archivos - FatFs	**2**	TestSuite-ID-2	Correcto
Módulo de Escritura SD	**2**	TestSuite-ID-2	Fallido

Para la gestión general del plan de pruebas se utiliza la herramienta TestLink, Fig 3, los resultado se anexan a la correspondiente tarea en Redmine manteniendo así la trazabilidad a lo largo del proceso.

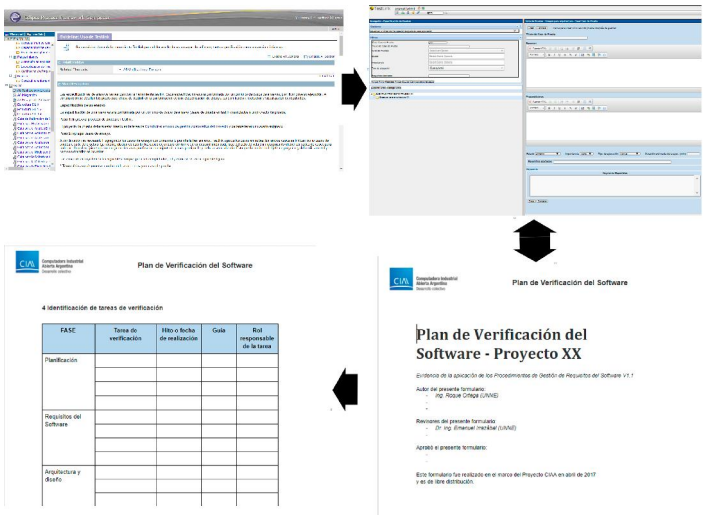

Fig 3. Ejemplo de proceso para la gestión del plan de pruebas con TestLink en el EPF.

A continuación se detallan las estrategias a seguir en cuanto al manejo de los errores. En la Tabla 10 se identifica los apartados a completar.

Tabla 10. Estrategias para la gestión de los errores. Ejemplo de uso.

Error	Estrategia de Detección del error	Estrategia de Evitación del error	Estrategia de Tolerancia
Luego de N procesos de escritura la tarjeta SD está vacía.	Aplicación de programación defensiva – Comprobar cada N escrituras que efectivamente se encuentren escritos N registro en memoria.	Ante la reiterada ocurrencia de este error, optar por el reemplazo del módulo hardware.	Guardar en memoria local N registros, ante una discordancia entre la cantidad de registros escritos y leídos, se vuelve a escribir la totalidad de registros guardados en memoria local.

Por último se justifica el uso de las técnicas, medidas y herramientas utilizadas el diseño de la arquitectura software, de acuerdo al apartado 7.3.4.14 de la norma, la combinación de técnicas seleccionada de justificarse como un conjunto que satisfaga los apartados 4.8 y 4.9 de la tabla A.3 de la norma. En la Tabla 11 se identifican los apartados a completar.

Tabla 11. Técnicas utilizadas para el diseño de la arquitectura software. Ejemplo de uso.

Técnica	Herramienta	Estrategia
Programación Divergente	Gestor de proyectos y Tareas “Redmine”. Gestión del versionado de código fuente “Git”	Se establece un N mínimo de 2 para los componentes de toma de decisión, más un tercer equipo para el desarrollo del módulo de voto.

Finalmente, como resultado final de la implementación documental de este proceso se presenta una tabla descriptiva para cada apartado de la norma.

Tabla 12. Explicación de la implementación del proceso. Ejemplo de uso.

Apartado	Detalle	Explicación de la implementación
7.3.4.2	Especificación de la arquitectura del software.	Se ha realizado una plantilla en la plataforma Google Drive, enlazada desde la especificación del proceso en EPF
7.3.4.3	Estudio de viabilidad de los requisitos.	Se justifica por cada requisito si el nivel de SIL requerido es viable, o en caso de no ser viable se debe recomendar un nivel de SIL. A cargo del rol de diseñador.

Apartado	Detalle	Explicación de la implementación
7.3.4.4	Identificación de las interacciones Hardware/Software	Se instrumenta a partir de una tabla de doble entrada indicando el nivel de interacción entre los componente utilizando una escala del 1 al 5.
7.3.4.5	Descripción de componentes software	Se identifica la antigüedad del software y si estos fueron validados con anterioridad y bajo qué condiciones.
7.3.4.6	Gestión de la configuración de los componentes software	Se consigue trazabilidad entre requisitos. El diseño y el código fuente a partir de la utilización en conjunto de las herramientas de gestión colaborativa Redmine y el gestor de versiones de código fuente GIT
7.3.4.7	Gestión del software preexistente.	Se cumple con una serie de tablas que permiten identificar toda la información a tener en cuenta cuando se utiliza software preexistente y durante la integración de los mismos al sistema.
7.3.4.8	Se prefiere el uso de componentes software verificados, desarrollados conforme a esta norma.	Se cumple en la condición de validación del componente software. Tabla 4
7.3.4.9	Plan de pruebas y verificación de nivel de integridad SIL por componente	Se ha realizado la gestión de casos de prueba y del Plan de pruebas a partir de la herramienta Testlink.
7.3.4.10	Garantizar la trazabilidad de la Especificación de la Arquitectura software hasta la Especificación de Requisitos del Software,	La realización de los pasos anteriores tiene como finalidad el uso de una “Metodología Estructurada” para lograr una trazabilidad constante con la Especificación de Requisitos del software en cada paso.
7.3.4.11	Estrategias de gestión de errores	Se identifican los errores y se describe las estrategias a utilizar en caso de la ocurrencia de los mismos.
7.3.4.12	Descripción de Técnicas, medidas y herramientas a utilizar.	Se detalla en cada caso la estrategia de uso de la técnica y los recursos necesarios para su correcta aplicación.
7.3.4.14	Justificación del conjunto de técnicas utilizadas.	Se verifica que el conjunto de técnicas seleccionadas satisfagan los requisitos de tabla A.3 de la norma. **¡Error! No se encuentra el origen de la referencia.**

4 Arquitectura software de un monitor de barrera

En esta sección se detalla un breve ejemplo de la utilización del proceso descrito utilizando, como entrada la Especificación de requisitos software – monitor de barreras. El ejemplo puede encontrarse como documentación anexa a este trabajo y en la siguiente URL: http://www.linsse.com.ar/arquitectura_software_anexo.pdf

5 Conclusiones y trabajos futuros

Para este trabajo se ha realizado un estudio de los trabajos relacionados con arquitectura software en sistemas críticos ferroviarios. Se ha podido comprobar que las técnicas más utilizadas son el modelado junto con la programación defensiva.

Teniendo en cuenta esto y la realización de un prototipo de monitor de barrera para la empresa Trenes Argentinos se ha construido un proceso que cumpla con la normativa EN 50128 de desarrollo software para sistemas críticos ferroviarios. Este proceso ha hecho un uso exhaustivo de pasos documentales que serán trasladados a la herramienta EPF siguiendo con la construcción del ecosistema de desarrollo software llevado a cabo hasta el momento. Para completar su instrumentación se ha trabajado con la herramienta de trabajo colaborativo Redmine, el gestor de versiones Git y el gestor de pruebas Testlink.

El ejemplo de uso de este proceso se encuentra en un anexo a este trabajo por motivos de tamaño. La URL de acceso es la siguiente: http://www.linsse.com.ar/arquitectura_software_anexo.pdf. Finalmente, queda como trabajo futuro la propuesta de un proceso para la Especificación de Diseño del Software correspondiente a los aparatados 7.3.4.21 a 7.3.4.22 de la norma EN 50128.

Agradecimientos

El financiamiento de este trabajo ha sido realizado a partir del proyecto PI-F17-2017 “Análisis e implementación de tecnologías emergentes en sistemas computacionales de aplicación regional.”, acreditado por la Secretaría de Ciencia y Técnica de la Universidad Nacional del Nordeste (UNNE) para el periodo 2018-2021.

Referencias

1. Comisión Nacional de Regulación del Transporte, https://www.argentina.gob.ar/cnrt.
2. Categoría:Accidentes ferroviarios en Argentina, https://es.wikipedia.org/w/index.php?title=Categor%C3%ADa:Accidentes_ferroviarios_en_Argentina&oldid=101067910, (2017).

3. Sánchez, N.: Para la campaña, el Gobierno incrementa la polémica compra de trenes a China, https://www.clarin.com/ciudades/trenes-china-compra-licitacion-reestatizacion_0_rJduN7cwXe.html.
4. Ministerio de Transporte: Japón comenzará a fabricar la tecnología para el frenado automático de trenes, https://www.argentina.gob.ar/noticias/japon-comenzara-fabricar-la-tecnologia-para-el-frenado-automatico-de-trenes-que-beneficiara.
5. Clarin.com: Se destinarán US$ 2.400 millones para comprar trenes de carga chinos, https://www.clarin.com/economia/china-trenes_de_carga-randazzo-inversiones_0_rJygK8mKP7l.html.
6. Boulanger, J.-L.: CENELEC 50128 and IEC 62279 Standards. John Wiley & Sons (2015).
7. Ansaldo STS, http://www.ansaldo-sts.com/en.
8. Overview - Railway - Siemens, http://w3.siemens.com/mcms/industrial-controls/en/railway/pages/overview.aspx.
9. Kelly, T.: Using Software Architecture Techniques to Support the Modular Certification of Safety-critical Systems. In: Proceedings of the Eleventh Australian Workshop on Safety Critical Systems and Software - Volume 69. pp. 53–65. Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2006).
10. Durmuş, M.S., Yıldırım, U., Söylemez, M.T.: The Application of Automation Theory to Railway Signaling Systems: Turkish National Railway Signaling Project. Pamukkale Üniversitesi Mühendis. Bilim. Derg. 19, 216–223 (2013).
11. Durmuş, M.S.: Demiryolu Anklaşman Sistem Tasarımına Kontrol Ve Otomasyon Mühendisliği Yaklaşımı, (2014).
12. Dorka, M.: Safe software development for a video-based train detection system in accordance with EN 50128. (2013).
13. Booch, G., Rumbaugh, J., Jacobson, I.: The unified modeling language user guide. Addison-Wesley, Upper Saddle River, NJ (2005).
14. Bell, R.: Introduction to IEC 61508. In: Proceedings of the 10th Australian Workshop on Safety Critical Systems and Software - Volume 55. pp. 3–12. Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2006).
15. Yourdon, E., Constantine, L.L.: Structured design: Fundamentals of a discipline of computer program and systems design. Prentice-Hall, Inc. (1979).