Diseño de Software Embebidos bajo Normas Ferroviarias

Dr. Ing. Emanuel Irrazabal (UNNE)
Ing. Sergio Gallina (UNCA)

Técnicas de desarrollo de sistemas embebidos críticos bajo normas ferroviarias, desde la normativa hasta el diseño.

Parte 1: Seguridad

Parte 2: Calidad

Parte 1

Ing. Sergio Gallina (UNCA)

- **Desarrollo bajo normas EN-5012x**
- **Políticas RAMS**
- **Seguridad en el desarrollo del software**
  - **Personal**
  - **Especificación de Requisitos**
  - **Arquitectura, Diseño e Implementación**
  - **Verificación**
  - **Integración Hardware / Software**
- **Plan de Seguridad y Funciones de Seguridad**

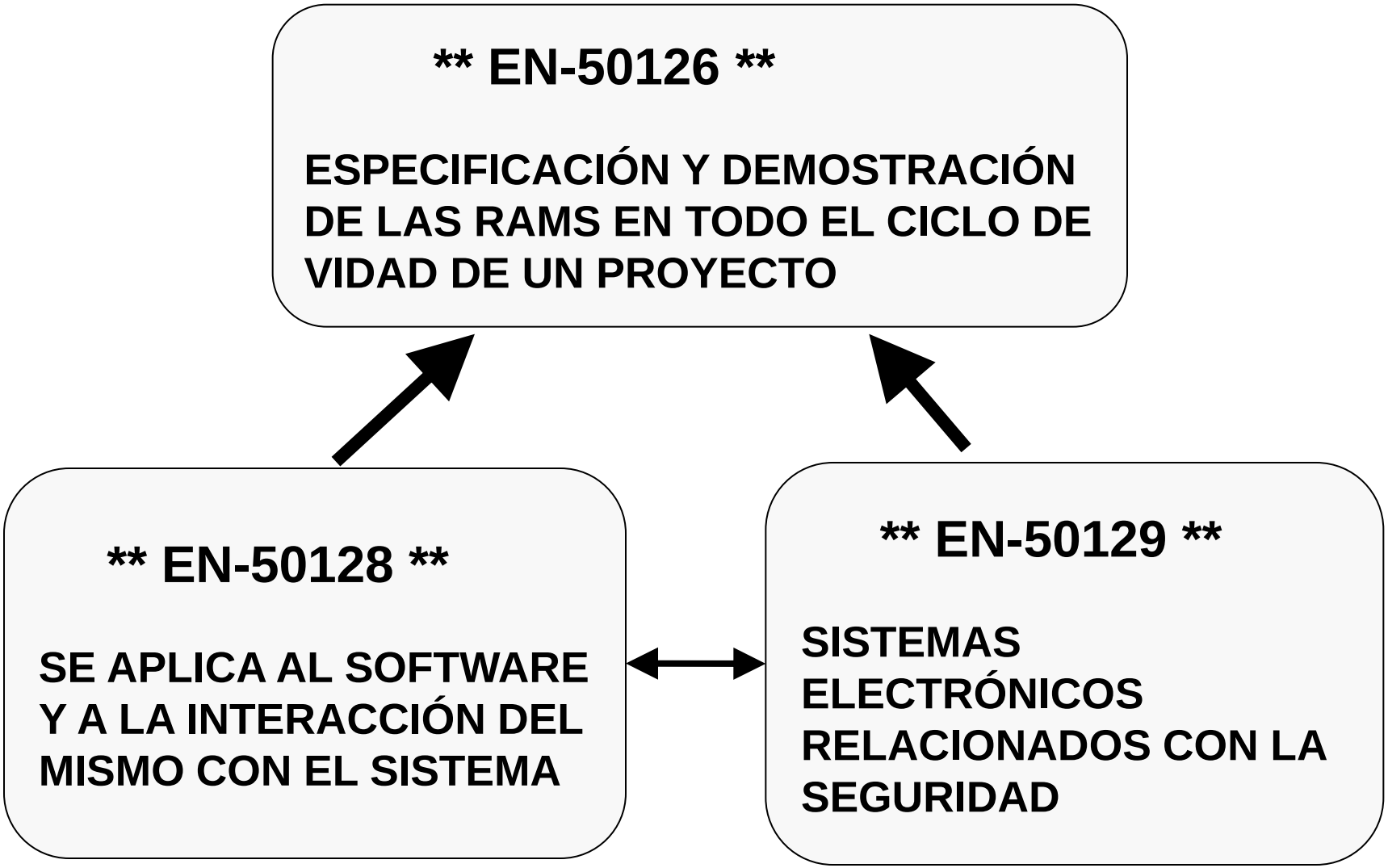
Metodología y normativa

- ✓ **Para minimizar los riesgos existen normas que fijan metodologías.**

- ✓ **Entre ellas tenemos las normas de la comunidad europea. Para aplicaciones ferroviarias haremos referencia a la serie UNE EN-5012x.**
  - **50126: Especificación y demostración de RAMS.**
  - **50128: Software para control y protección del ferrocarril.**
  - **50129: Sist. electrónicos de seguridad p/ la señalización.**

- ✓ **Existen otras normas, como las estadounidenses FRA 49:236, AREMA, IEEE 730, 8XX, 1012, 10XX, 1483, 1558.**

CONTEXTO PARA EL DESARROLLO DE SISTEMAS E/E/EP



Metodología y normativa
Políticas RAMS

- **La gestión de los procesos destinados a la especificación y demostración de las políticas RAMS son la piedra angular del desarrollo de sistemas críticos.**

- **El enfoque así definido es totalmente compatible con los procesos de control de calidad bajo normas ISO-9000.**

Metodología y normativa
Políticas RAMS

Fiabilidad (Reliability): probabilidad de que un elemento pueda realizar una función requerida en condiciones determinadas durante un intervalo de tiempo dado.

PARÁMETRO	SÍMBOLO	UNIDADES
Tasa de Fallo	$Z(t)$, λ	fallos / tiempo, distancia, ciclo
Tiempo Medio Activo	MUT	tiempo, distancia, ciclo
Tiempo Medio para Fallo Distancia Media para Fallo (para elementos no reparables)	MTTF MDTF	tiempo, distancia, ciclo
Tiempo Medio Entre Fallos Distancia Media Entre Fallos (para elementos reparables)	MTBF MDBF	tiempo, distancia, ciclo
Probabilidad de Fallo	F(t)	Sin unidades
Fiabilidad (Probabilidad de éxito)	R(t)	Sin unidades

Metodología y normativa
Políticas RAMS

Disponibilidad (Availability): capacidad de un producto de hallarse en situación de realizar una función requerida en condiciones determinadas en un momento dado o durante un intervalo de tiempo señalado, suponiendo que se faciliten los recursos externos requeridos.

PARÁMETRO	SÍMBOLO	UNIDADES
Disponibilidad	A (.) = MUT/(MUT+MUT)	sin unidades
inherente	A_i	
alcanzada	A_a	
operativa	A_o	
Disponibilidad de Flota	FA (= vehículos / flota disponibles)	sin unidades
Adherencia al horario	SA	sin unidades

Metodología y normativa
Políticas RAMS

Mantenibilidad (Maintianability): probabilidad de que una acción dada de mantenimiento activo, pueda ser llevada a cabo en un intervalo establecido de tiempo, bajo condiciones y recursos establecidos.

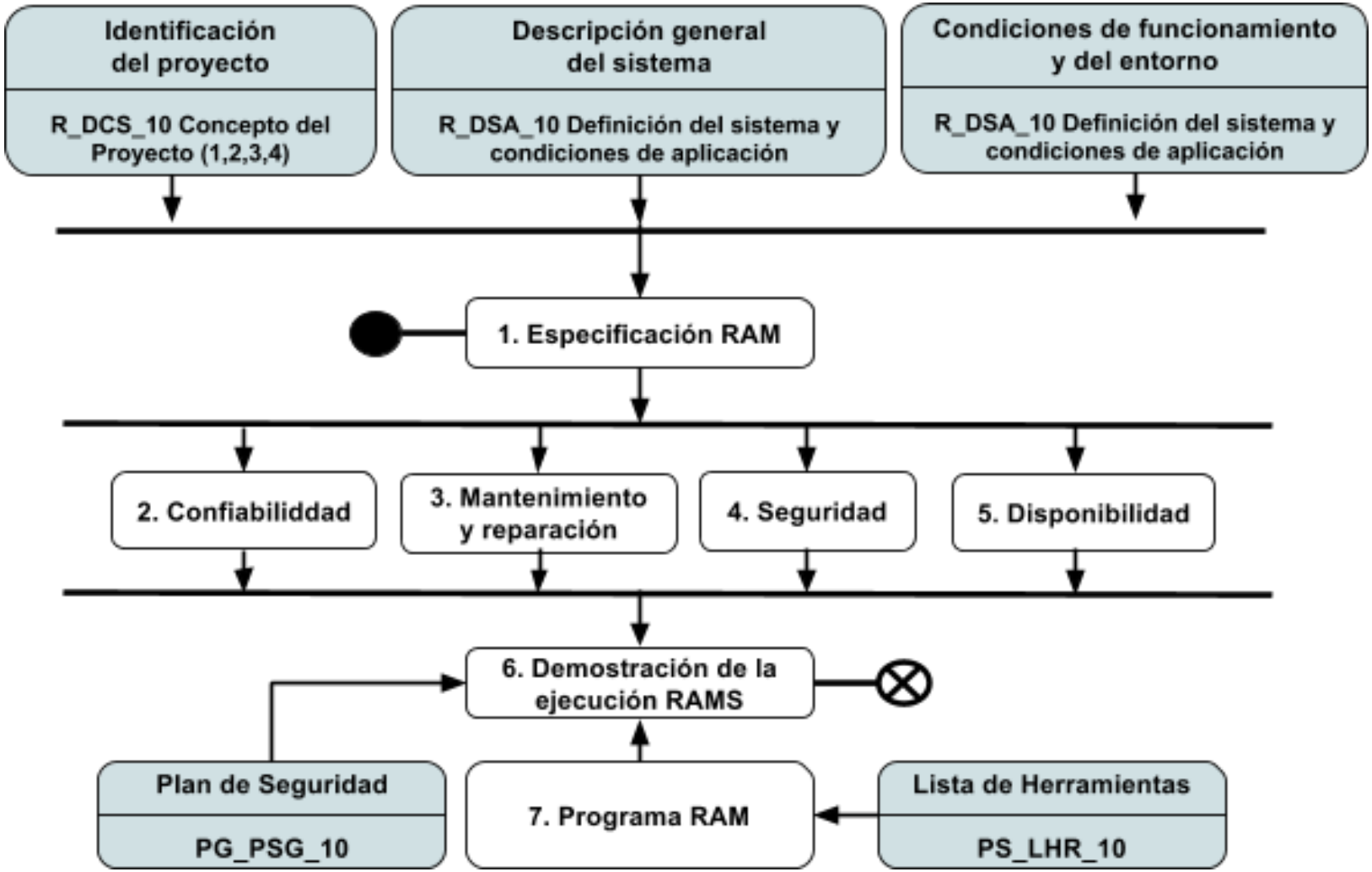
PARÁMETRO	SÍMBOLO	UNIDADES
Tiempo de Caída Medio	MDT	tiempo, distancia, ciclo
Tiempo / Distancia Media Entre Mantenimientos	MTBM/MDBM	tiempo, distancia, ciclo
MTBM / MDBM, Corrector o Preventivo	MTBM(c)/MDBM(c), MTBM(p)/MDBM(p)	tiempo, distancia, ciclo
Tiempo Medio hasta el mantenimiento	MTTM	tiempo
MTTM, Correctivo o Preventivo	MTTM(c), MTTM(p)	tiempo
Tiempo Medio de Reparación	MTTR	tiempo

Metodología y normativa
Políticas RAMS

Seguridad (Safety): Ausencia de riesgo inaceptable de daño.

PARÁMETRO	SÍMBOLO	UNIDADES
Tiempo medio Entre Fallos de Peligros	MTBF(H)	tiempo, distancia, ciclo
Tiempo Medio Entre "Fallos del Sistema de Seguridad"	MTBSF	tiempo, distancia, ciclo
Tasa de Peligros	H(t)	sin unidades
Tasa de Fallos relacionados con Seguridad	$F_s(t)$	fallos / tiempo, distancia, ciclo
Probabilidad de Funcionalidad Segura	$S_s(t)$	sin unidades
Tiempo de retorno a condiciones de Seguridad	TTRS	tiempo

Metodología y normativa
Políticas RAMS



Metodología y normativa
Ciclo de vida

"Una secuencia de fases, que contienen tareas y abarcan la vida completa de un sistema desde su concepto inicial hasta su retiro y eliminación".

"El ciclo de vida proporciona una estructura para la planificación, la gestión, el control y la supervisión de todos los aspectos de un sistema, incluido la RAMS."

Fases del ciclo de vida

- El ciclo de vida que propone la UNE 50126 es:

1. Concepto	8. Instalación
2. Definición del sistema y Condiciones de aplicación	9. Validación del Sistema (incluyendo Aceptación de seguridad y Puesta en Servicio)
3. Análisis de riesgos	10. Aceptación del Sistema
4. Requisitos del Sistema	11. Operación y Mantenimiento
5. Distribución de los Requisitos del Sistema	12. Supervisión de Ejecución
6. Diseño e Implementación	13. Modificación, Realimentación
7. Producción	14. Retirada del servicio y Eliminación

GENERALIDADES EN-5012x

Al aplicar la norma se pretende

- **Dejar evidencia de la gestión de la calidad**
- **Dejar evidencia de la gestión de la seguridad**
- **Dejar evidencia de la seguridad funcional y técnica**

GENERALIDADES EN-50128 Desarrollo del software

- **Se centra en los métodos para desarrollar software**
- **Para el análisis de riesgos, peligros y seguridad remite a la norma IEC 61508 (EN-61508)**
- **Identifica 5 niveles de integridad de la seguridad (SIL)**

NIVEL DE INTEGRIDAD DE SEGURIDAD DEL SOFTWARE	DESCRIPCIÓN DE LA INTEGRIDAD DE SEGURIDAD DEL SOFTWARE
4	MUY ALTA
3	ALTA
2	MEDIA
1	BAJA
0	No relacionado con la seguridad

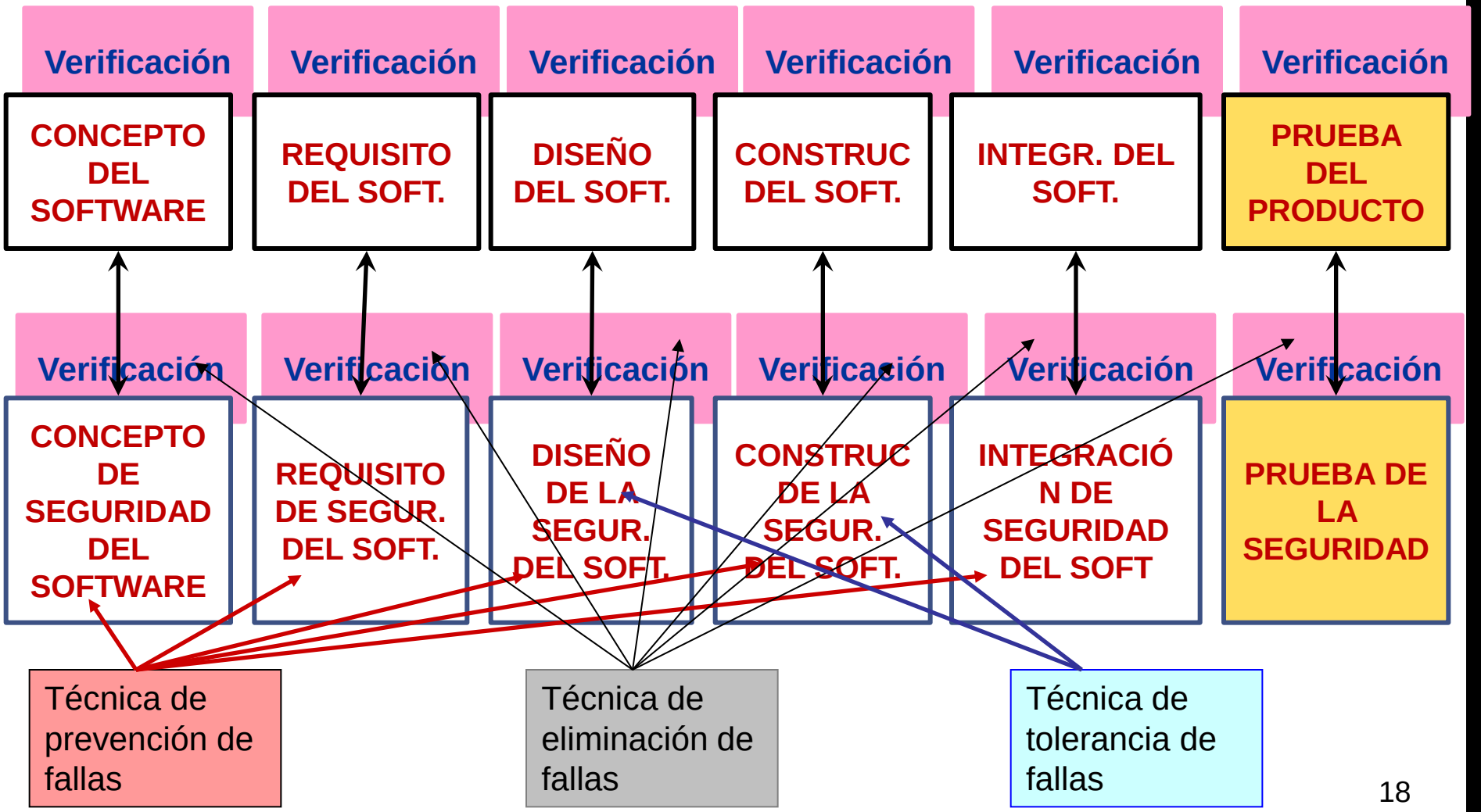
PRINCIPIOS DE DESARROLLO ACEPTADOS

- **Diseño descendente top-down**
- **Modularidad**
- **Verificación de cada fase del ciclo de vida de desarrollo**
- **Módulos y librerías verificados**
- **Documentación clara y auditable**
- **Ensayos de validación**
- **OTROS**

PASOS PARA LA APLICACIÓN DE LA EN-50128 EN EL DESARROLLO DE SE

1. **Definir la especificación de requisitos de seguridad y en paralelo considerar la arquitectura del software (Cap 4)**
2. **Diseño, desarrollo y ensayo del software de acuerdo al plan de aseguramiento de la calidad, al nivel de integridad de seguridad y al ciclo de vida del software (Cap 7)**
3. **Integrar el software en el hardware específico (Cap 7)**
4. **Validar el software (Cap 6)**
5. **Mantenimiento (Cap 9)**

PASOS PARA LA APLICACIÓN DE LA EN-50128



PERSONAL Y RESPONSABILIDADES

- **El proveedor y el responsable de desarrollo deben implementar partes relevantes de la ISO 9001**

- **Se debe incluir evidencia de la experiencia**

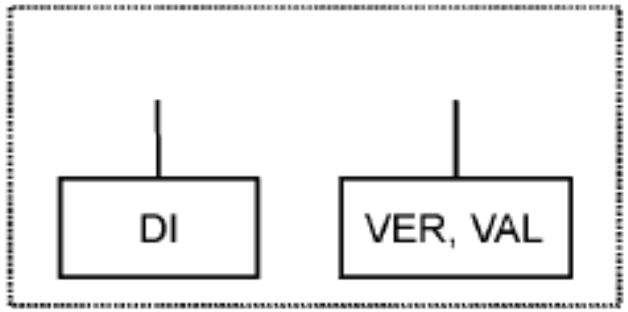
- **El verificador y el validador puede ser la misma persona pero no el desarrollador y no deben depender del director del proyecto**

PERSONAL Y RESPONSABILIDADES

NIVEL 0

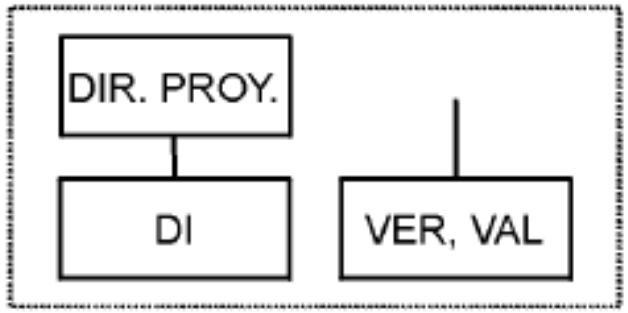
EVALUADOR

NIVELES 1 Y 2



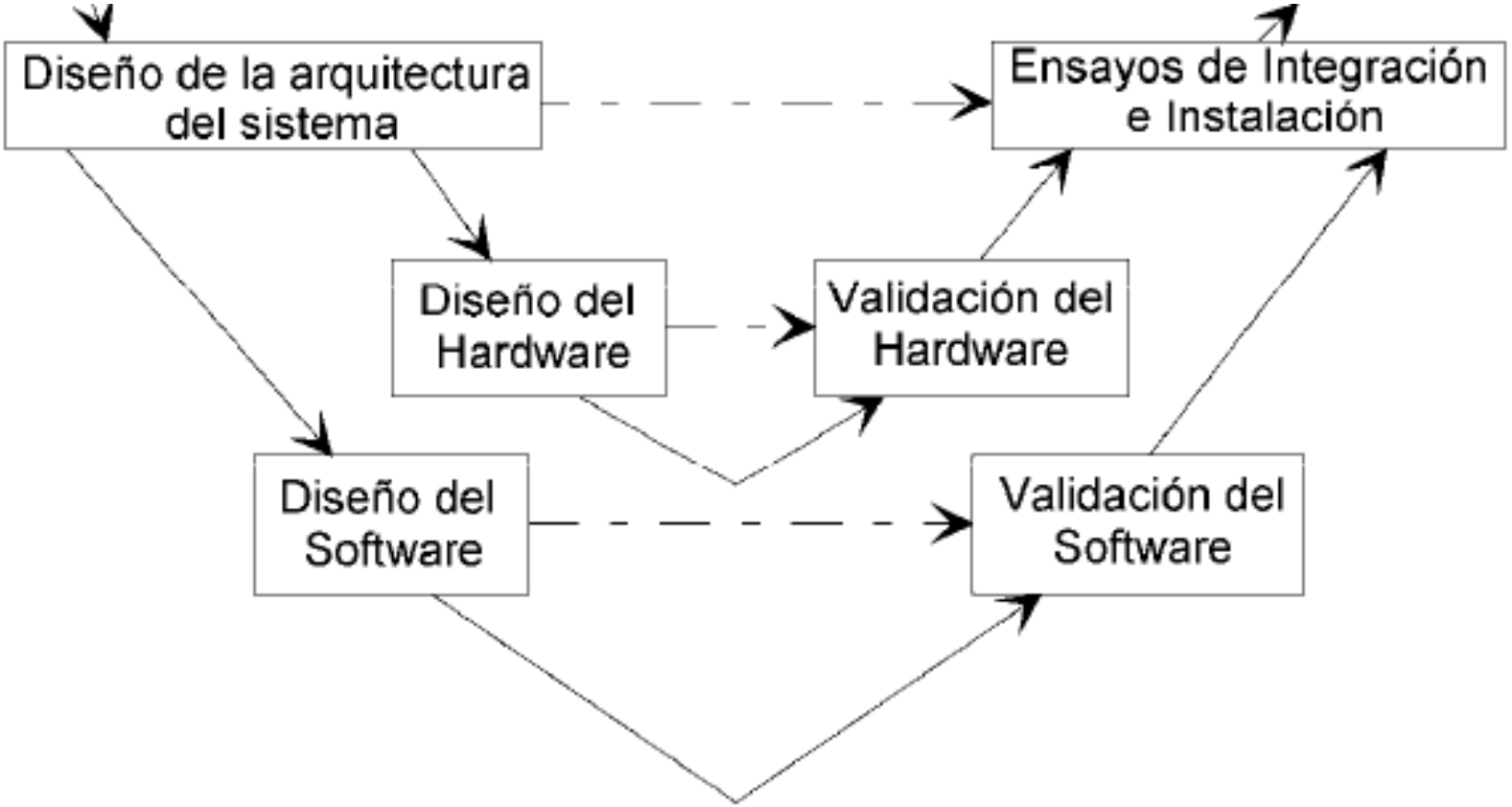
EVALUADOR

NIVELES 3 Y 4

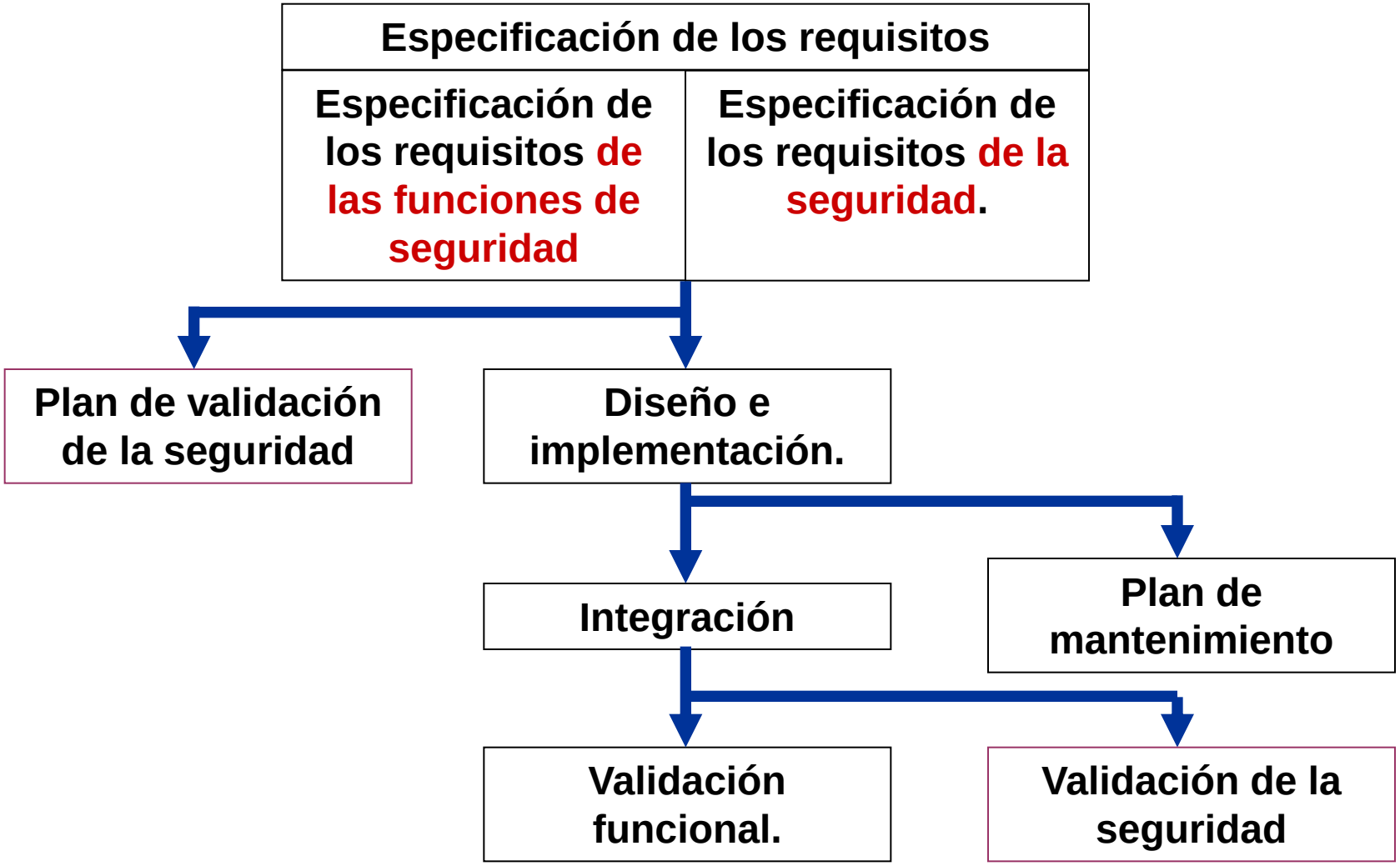


EVALUADOR

DESARROLLO DE SISTEMA EP (ELECTRONICOS PROGRAMABLES)



DESARROLLO DE SISTEMA EP (ELECTRONICOS PROGRAMABLES)



ESPECIFICACIÓN DE REQUISITOS DEL SOFTWARE

Para poder especificar los requisitos del software se requiere

- **Requisitos del sistema**
- **Requisitos de seguridad del sistema**
- **Descripción de la arquitectura del sistema**
- **PLAN DE ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE**

Documentos de salida (entregables)

- **Especificación de requisitos del software**
- **Especificación de ensayos de requisitos del software**

ESPECIFICACIÓN DE REQUISITOS DEL SOFTWARE

TÉCNICA/MEDIDA		Ref	SIL SW 0	SIL SW 1	SIL SW 2	SIL SW 3	SIL SW 4
1	Métodos Formales, incluyendo por ejemplo CCS, CSP, HOL, LOTOS, OBJ, Lógica Temporal, VDM Z y B	B.30	-	R	R	AR	AR
2	Métodos Semi-Formales	D.7	R	R	R	AR	AR
3	Metodología Estructurada, incluyendo por ejemplo JSD, MASCOT, SADT, SDL, SSADM y Yourdon	B.60	R	AR	AR	AR	AR

La especificación de requisitos, requiere siempre la descripción en lenguaje natural y la notación formal (matemática) que refleje tal especificación

ARQUITECTURA DEL SOFTWARE

Deberá cumplir con las especificación de requisitos del software para el nivel de seguridad especificado

Se deben identificar todos los componentes del software y verificar

→ si son propios o de terceros

→ si han sido validados

→ si son nuevos o reutilizados

ARQUITECTURA DEL SOFTWARE

TÉCNICA/MEDIDA		Ref	SIL SW 0	SIL SW 1	SIL SW 2	SIL SW 3	SIL SW 4
1	Programación Defensiva	B.15	–	R	R	AR	AR
2	Diagnosis y Detección de Defectos	B.27	–	R	R	AR	AR
3	Códigos de Corrección de Errores	B.20	–	–	–	–	–
4	Códigos de Detección de Errores	B.20	–	R	R	AR	AR
5	Programación con Detección de Fallos	B.25	–	R	R	AR	AR
6	Técnicas de Bolsa de Seguridad	B.54	–	R	R	R	R
7	Programación con Diversidad	B.17	–	R	R	AR	AR
8	Recuperación en Bloque	B.50	–	R	R	R	R
9	Recuperación Regresiva	B.5	–	NR	NR	NR	NR
10	Recuperación Progresiva	B.32	–	NR	NR	NR	NR
11	Mecanismos de Recuperación de Fallos por Reintento	B.53	–	R	R	R	R
12	Memorización de Casos Ejecutados	B.39	–	R	R	AR	AR
13	Inteligencia Artificial – Corrección de Fallos	B.1	–	NR	NR	NR	NR

DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE

Obtener un software capaz de ser analizado y verificado

Integración del software

Finalizada esta fase se debe disponer de

- **Documentación**
- **Código fuente**
- **Resultado de la verificación (ensayos)**

Mantener al mínimo el tamaño y la complejidad del software

DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE

TÉCNICA/MEDIDA		Ref	SIL SW 0	SIL SW 1	SIL SW 2	SIL SW 3	SIL SW 4
1	Métodos Formales incluyendo por ejemplo CCS, CSP, HOL, LOTOS, OBJ, Lógica Temporal, VDM, Z y B	B.30	–	R	R	AR	AR
2	Métodos Semi-Formales	D.7	R	AR	AR	AR	AR
3	Metodología Estructurada incluyendo por ejemplo JSD, MASCOT, SADT, SDL, SSADM y Yourdon	B.60	R	AR	AR	AR	AR
4	Aproximación Modular	D.9	AR	O	O	O	O
5	Estándares de Diseño y Codificación	D.1	AR	AR	AR	O	O
6	Programas Analizables	B.2	AR	AR	AR	AR	AR
7	Lenguaje de Programación Fuertemente Tipado	B.57	R	AR	AR	AR	AR
8	Programación Estructurada	B.61	R	AR	AR	AR	AR
9	Lenguaje de Programación	D.4	R	AR	AR	AR	AR
10	Subconjunto del Lenguaje	B.38	–	–	–	AR	AR
11	Traductor Validado	B.7	R	AR	AR	AR	AR
12	Traductor Probado por el Uso	B.65	AR	AR	AR	AR	AR
13	Librería de Módulos y Componentes Comprobados/Verificados	B.40	R	R	R	R	R

DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE

TÉCNICA/MEDIDA		Ref	SIL SW 0	SIL SW 1	SIL SW 2	SIL SW 3	SIL SW 4
1	Existencia de normas de Codificación	B.16	AR	AR	AR	AR	AR
2	Pautas para el Estilo de Codificación	B.16	AR	AR	AR	AR	AR
3	Ausencia de Objetos Dinámicos	B.16	–	R	R	AR	AR
4	Ausencia de Variables Dinámicas	B.16	–	R	R	AR	AR
5	Uso Limitado de Punteros	B.16	–	R	R	R	R
6	Uso Limitado de Recursividad	B.16	–	R	R	AR	AR
7	Ausencia de Saltos Incondicionales	B.16	–	AR	AR	AR	AR

Finalidad:

- Estructurar el código y la documentación.
- Evitar las personalizaciones

DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE

Lenguaje de programación adecuado

TÉCNICA/MEDIDA		Ref	SIL SW 0	SIL SW 1	SIL SW 2	SIL SW 3	SIL SW 4
1	ADA	B.62	R	AR	AR	R	R
2	MODULA-2	B.62	R	AR	AR	R	R
3	PASCAL	B.62	R	AR	AR	R	R
4	Fortran 77	B.62	R	R	R	R	R
5	'C' o C++ (sin restricción)	B.62	R	–	–	NR	NR
6	Subconjunto de C o C++ con normas de codificación	B.62 B.38	R	R	R	R	R
7	PL/M	B.62	R	R	R	NR	NR
8	BASIC	B.62	R	NR	NR	NR	NR
9	Ensamblador	B.62	R	R	R	–	–

DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE

¿Que se debe evitar?

- **Saltos incondicionales a excepción de llamadas a subrutinas**
- **Recursividad**
- **Punteros o cualquier objeto dinámico**
- **Declaración / Inicialización implícita de variables**

La programación en assembler no está bien considerada debido a su fuerte orientación al hardware

VERIFICACIÓN Y ENSAYO

Es obligatoria la creación de un plan de verificación del software

- **Selección de estrategias y técnicas**
- **Selección y utilización de equipos**
- **Selección de la documentación**
- **Evaluación de los resultados**
- **Evaluación de la fiabilidad del sistema**
- **El grado de cobertura que se logró en el ensayo**

Se debe poder demostrar (y quedar documentado) que se cumple con la fiabilidad, prestaciones y seguridad.

VERIFICACIÓN Y ENSAYO

TÉCNICA/MEDIDA		Ref	SIL SW 0	SIL SW 1	SIL SW 2	SIL SW 3	SIL SW 4
1	Ensayo Formal	B.31	–	R	R	AR	AR
2	Ensayos Probabilísticos	B.47	–	R	R	AR	AR
3	Análisis Estático	**D.8**	–	AR	AR	AR	AR
4	Análisis y Ensayos Dinámicos	**D.2**	–	AR	AR	AR	AR
5	Evaluaciones	B.42	–	R	R	R	R
6	Matriz de Trazabilidad	B.69	–	R	R	AR	AR
7	Análisis de Efectos de Errores software	B.26	–	R	R	AR	AR

1. Análisis de flujo de control
2. Análisis de flujo de datos
3. Ensayos y revisiones de diseño

1. Casos de ensayo a partir de valores extremos
2. Modelado de prestaciones
3. Ensayos basados en la estructura

INTEGRACIÓN HARDWARE / SOFTWARE

Demostrar que el hardware y el software interactúan correctamente

Durante esta etapa se debe

Confeccionar un plan de ensayos

Documentar el resultado de su ejecución.

INTEGRACIÓN HARDWARE / SOFTWARE

TÉCNICA/MEDIDA		Ref	SIL SW 0	SIL SW 1	SIL SW 2	SIL SW 3	SIL SW 4
1	Ensayos Funcionales y de Caja-negra	**D.3**	AR	AR	AR	AR	AR
2	Ensayos de Prestaciones	**D.6**	–	R	R	AR	AR
Requisitos:							
1	Para el nivel de integridad de seguridad del software 0, la técnica 1 deberá ser la aprobada.						
2	Para los niveles de integridad de seguridad del software 1, 2, 3 ó 4, la combinación de técnicas aprobada deberá ser 1 y 2.						

1. Tiempos de respuesta y limitaciones de memoria
2. Ensayo de avalancha / estres
3. Requisitos de prestaciones

1. Análisis de valores extremos
2. Simulación de procesos
3. Prototipado / Animación

MANTENIMIENTO

El mantenimiento debe ser tal que después de esta tarea el software se comporta en la forma requerida, preservando el nivel de integridad de la seguridad y la confiabilidad

Incluye:

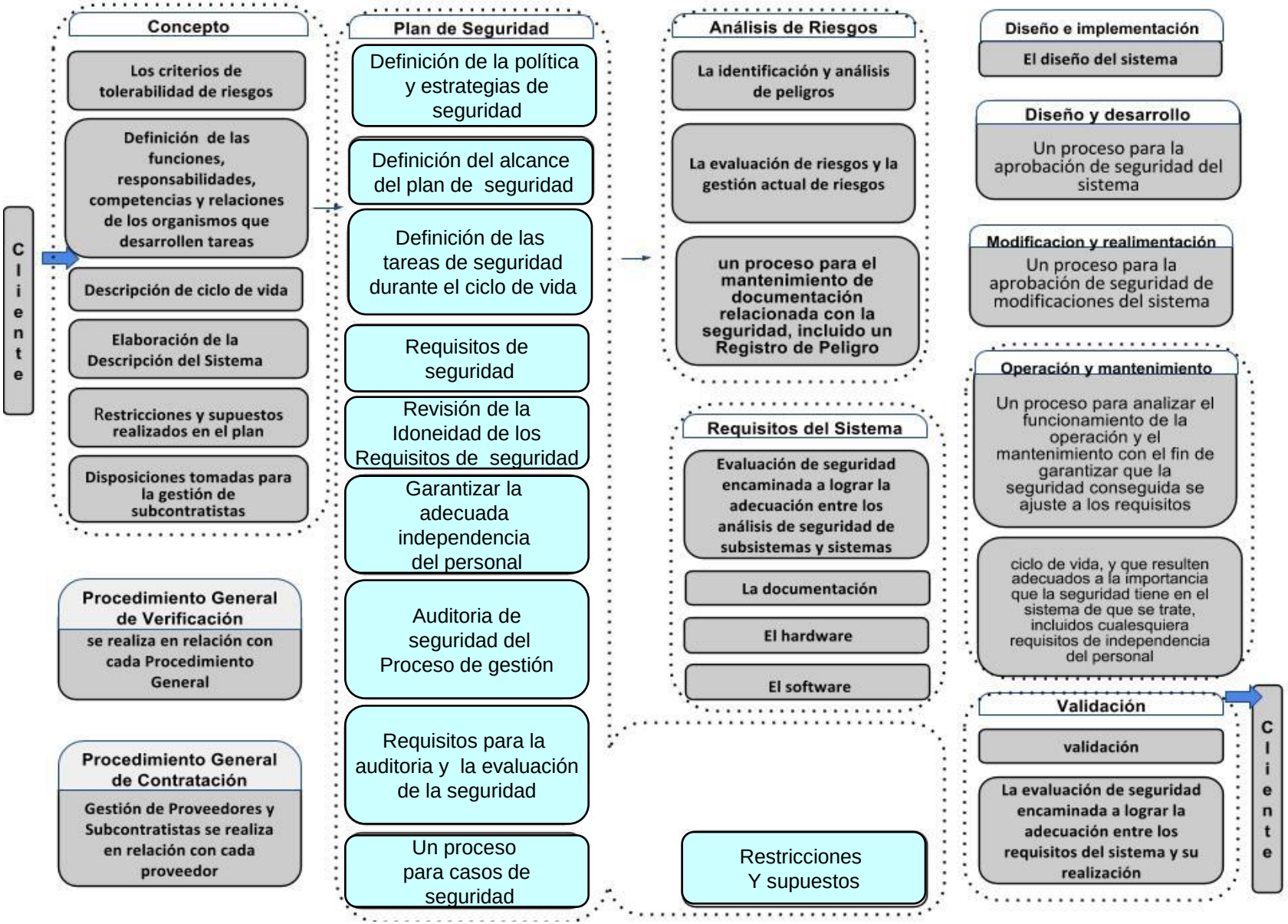
- **Correcciones**
- **Mejoras**
- **Adaptaciones**

Como mínimo los mantenimientos se deben llevar a cabo de acuerdo a las normas ISO 9000-3

MANTENIMIENTO

Técnicas/Medidas	SIL 1	SIL 2	SIL 3	SIL 4
1 Producción de instrucciones para la operación de las aplicaciones y para el mantenimiento	R: todas las instrucciones operativas, de aplicación, y de mantenimiento se tienen que poder enlazar con el diseño incluyendo el uso del registro de peligros		HR: todas las instrucciones operativas, de aplicación, y de mantenimiento se tienen que poder enlazar con el diseño incluyendo el uso del registro de peligros	
2 Formación para la ejecución de las instrucciones de funcionamiento y mantenimiento	HR: formación inicial de todos los operadores y del personal de mantenimiento		HR: formación inicial más cursos de actualización de todos los operadores y del personal de mantenimiento	
3 Facilidad para el operador	HR: la interacción entre persona y sistema tiene que ser tan simple como sea posible, para reducir el riesgo de errores humanos			
4 Facilidad para el mantenimiento	HR: herramientas de diagnosis separadas, llevar a cabo medidas de mantenimiento relacionadas con la seguridad tan poco frecuentemente como sea posible		HR: se deben incluir herramientas de diagnosis de fácil manejo, prácticas y suficientes para llevar a cabo medidas de reparación inevitables, y de mantenimiento relacionadas con la seguridad tan poco frecuentemente como sea posible o a ser posible que no se tengan ni que realizar	
5 Protección contra los errores de operación	R: verificación de los procedimientos de verosimilitud de cada comando de		HR: verificación de los procedimientos de verosimilitud de cada comando	

Plan de seguridad



Funciones de seguridad (FS)

A cada evento peligroso se le asocia una FS, las cuales en conjunto forma en Sistema Instrumentado de Seguridad (SIS) → capa de protección independiente.

Las FS realizan funciones especificas para alcanzar y mantener el estado de seguridad.

El nivel SIL o la probabilidad de falla bajo demanda (PFD) son medidas del desempeño y la disponibilidad de las FS

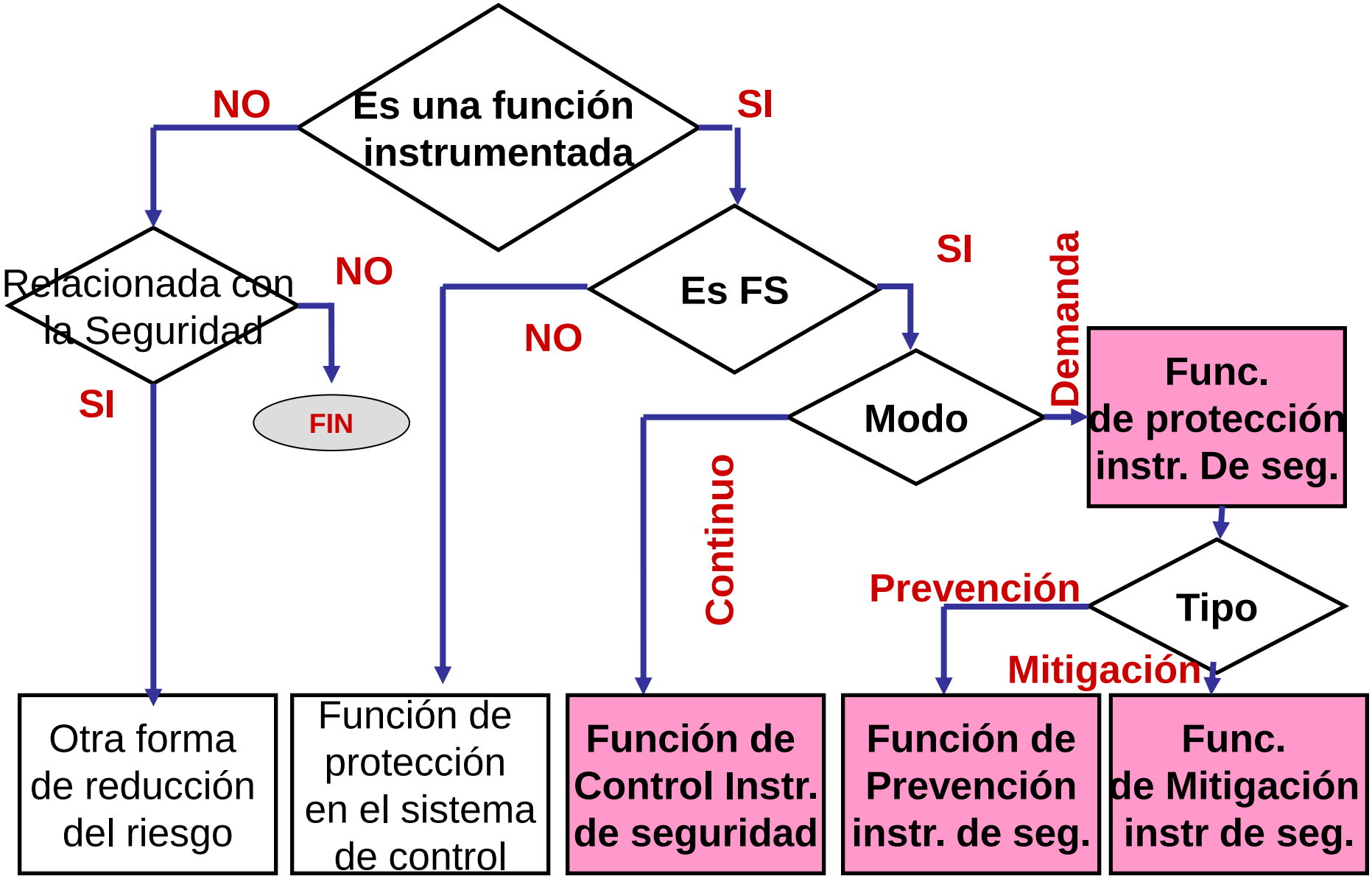
SAFETY AVAILABILITY = 1 – PFD

Las funciones de seguridad pueden ser

a) De modo continua: se desarrolla cuando una falla puede generar un potencial peligro y esta puede ocurrir sin causa

b) Bajo demanda: Es una acción que se toma ante la aparición de un evento que requiere ser controlado

Funciones de seguridad (IEC 61511)



EVIDENCIA TECNICA DE FUNCIONAMIENTO CORRECTO

La evidencia técnica de seguridad del sistema se debe presentar en el informe de seguridad técnica.

Secciones del informe de seguridad técnica:

Sección 1: Introducción

Sección 2: Garantía de operación funcional correcta

Sección 3: Efecto de las averías

Sección 4: Funcionamiento bajo influencias externas

Sección 5: Condiciones de aplicación relacionadas con la seguridad

Sección 6: Ensayos para la calificación de la seguridad

BENEFICIOS - RIESGOS

- **El desarrollo de software crítico es "más costoso"**

Pero en determinadas actividades hay que decidir

¿Qué riesgos se asumen y quien los asume si no se adoptan las medidas necesarias?

- **Por otro lado, como usuarios**

¿Sabemos el riesgo que asumimos al usar o depender de un sistema?

¿CUÁNTOS VIAJARÍAN EN UN AVIÓN QUE NO SE CUMPLEN LOS REQUISITOS DE SEGURIDAD?

¿CUÁNTOS UTILIZARÍAMOS UN CAJERO AUTOMÁTICO?

ANEXO PARTE 1 - CASO DE SEGURIDAD (NO PARA ESTE TUTORIAL)

TRANSFORMACION

MODELO FORMAL

Desarrollaremos un modelo de caso de seguridad genérico, mas o menos formal (cuasi formal) con el propósito de mostrar los requerimientos de las normas en forma amigable.

CASO DE SEGURIDAD

¿Cuáles son las tareas que se deben hacer?

¿Qué tipo de tareas?

CASO DE SEGURIDAD

FASES DEL CICLO DE VIDA	TAREAS GENERALES DE LA FASE	TAREAS RELACIONADAS CON LAS RAM	TAREAS RELACIONADAS CON LA SEGURIDAD
1. CONCEPTO	• Establecer el ambito y proposito … • Definir el concepto del proyecto • …	• Revisar las ejecuciones RAM en proyectos anteriores • Considerar las implicaciones RAM	• Revisar las ejecuciones de seguridad en proyectos anteriores • Considerar las implicaciones de seguridad • …
2. DEFINICION DEL SISTEMA	• …	• ...	• ...

TAREAS GENERALES **TAREAS RAM** **TAREAS DE SEGURIDAD**

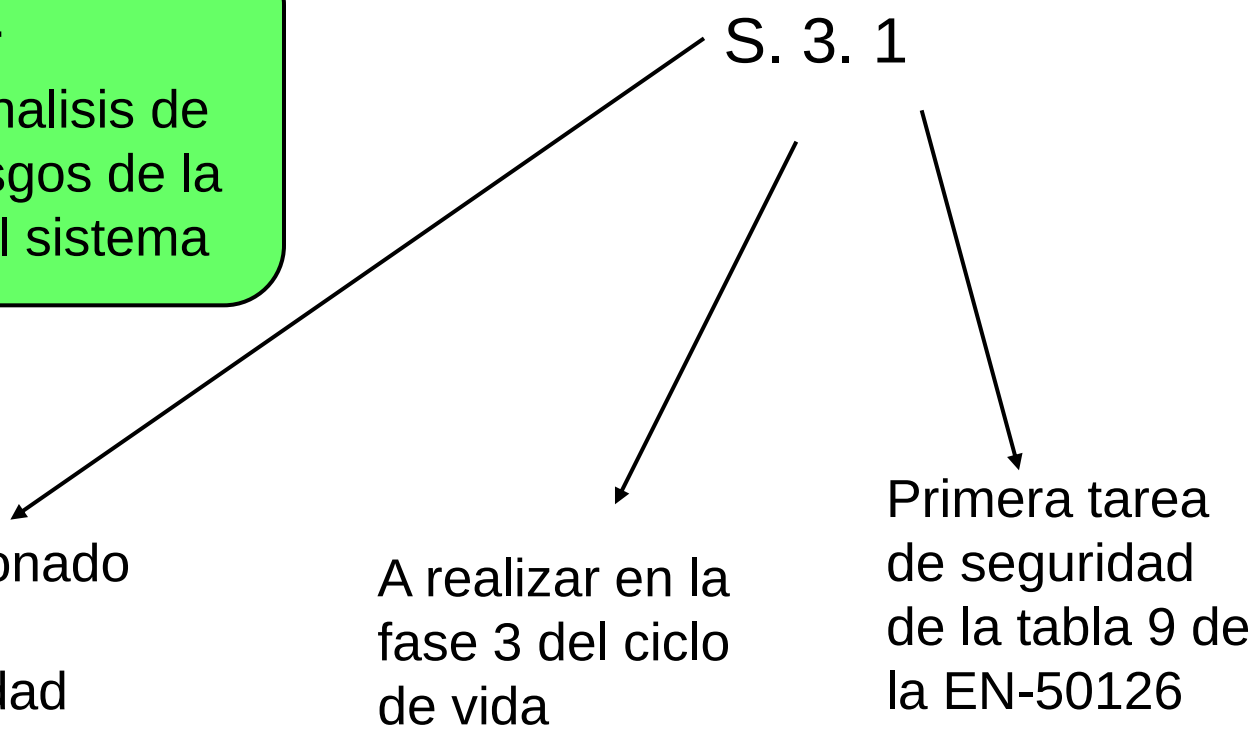
La figura 9 de la EN-50126 especifica las tareas para cada fase del ciclo de vida

CASO DE SEGURIDAD CONSTRUCCION DEL MODELO

1. Identificación de las tareas y el tipo

S.3.1

Realizar el analisis de peligros y riesgos de la seguridad del sistema

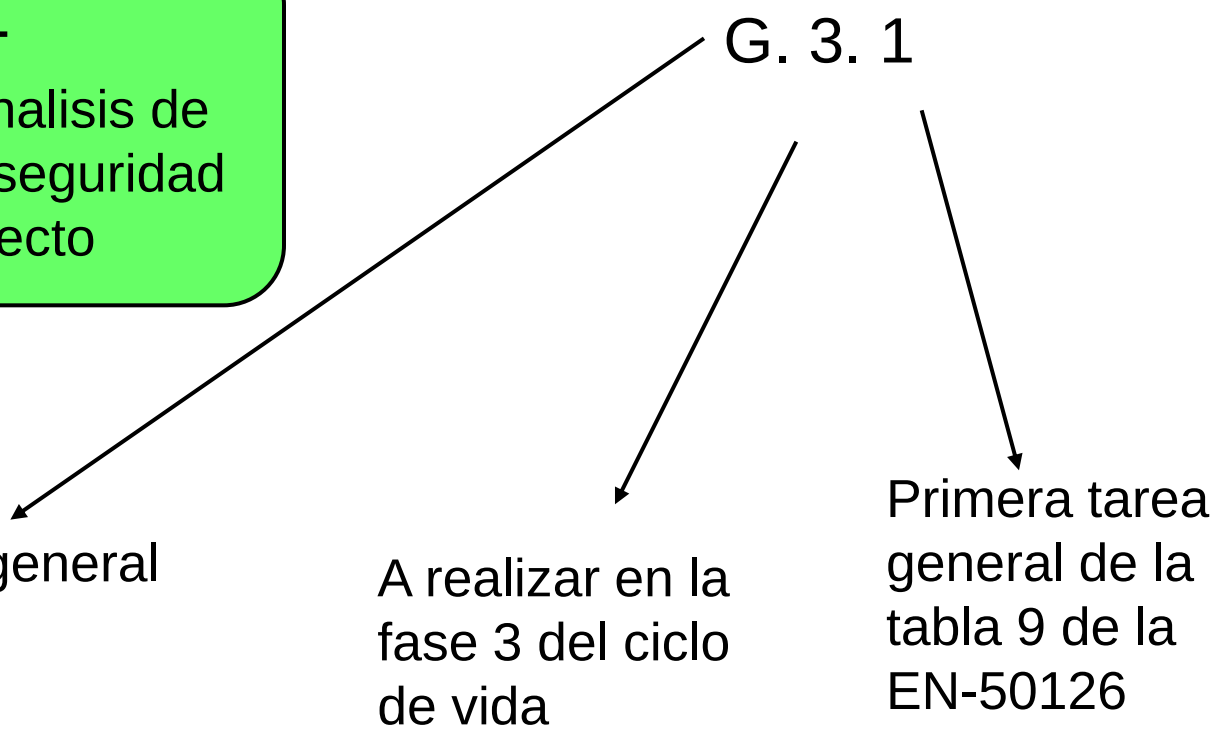


CASO DE SEGURIDAD
CONSTRUCCION DEL MODELO

1. Identificación de las tareas y el tipo

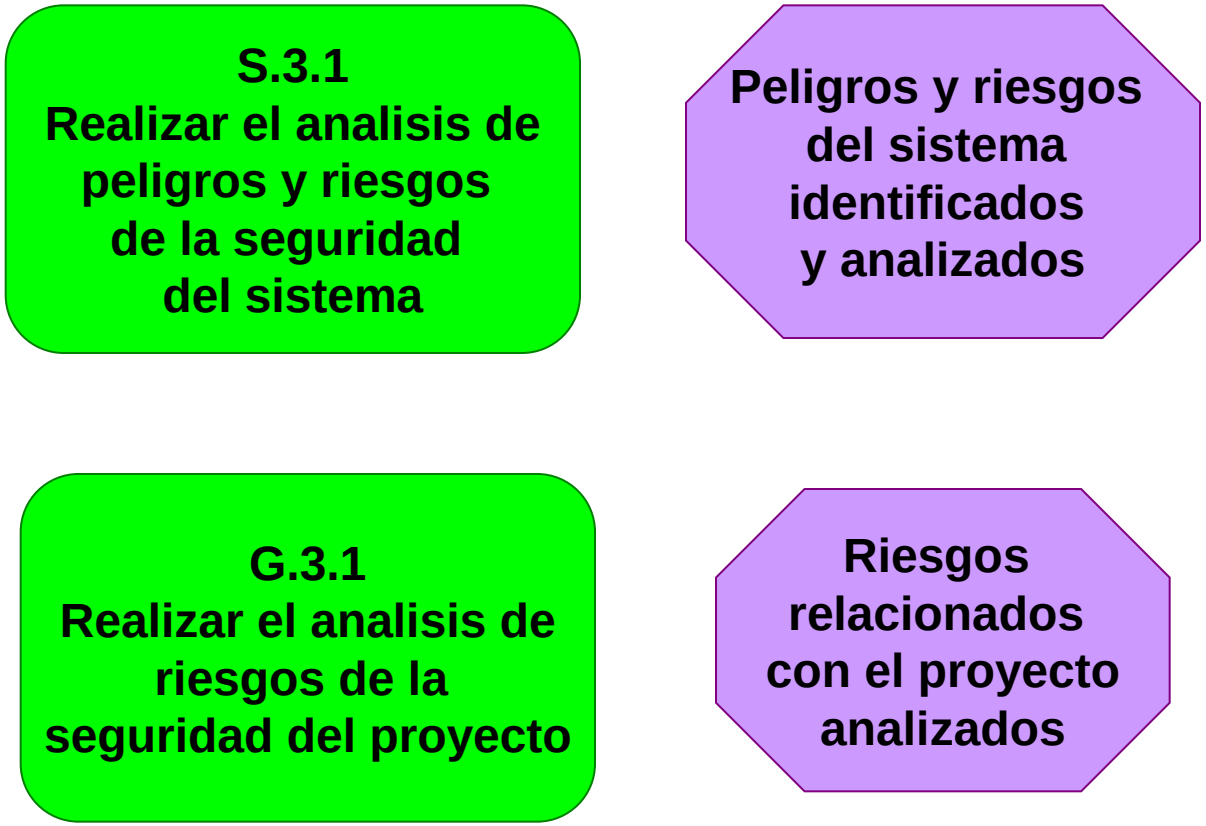
G.3.1

Realizar el analisis de riesgos de la seguridad del proyecto



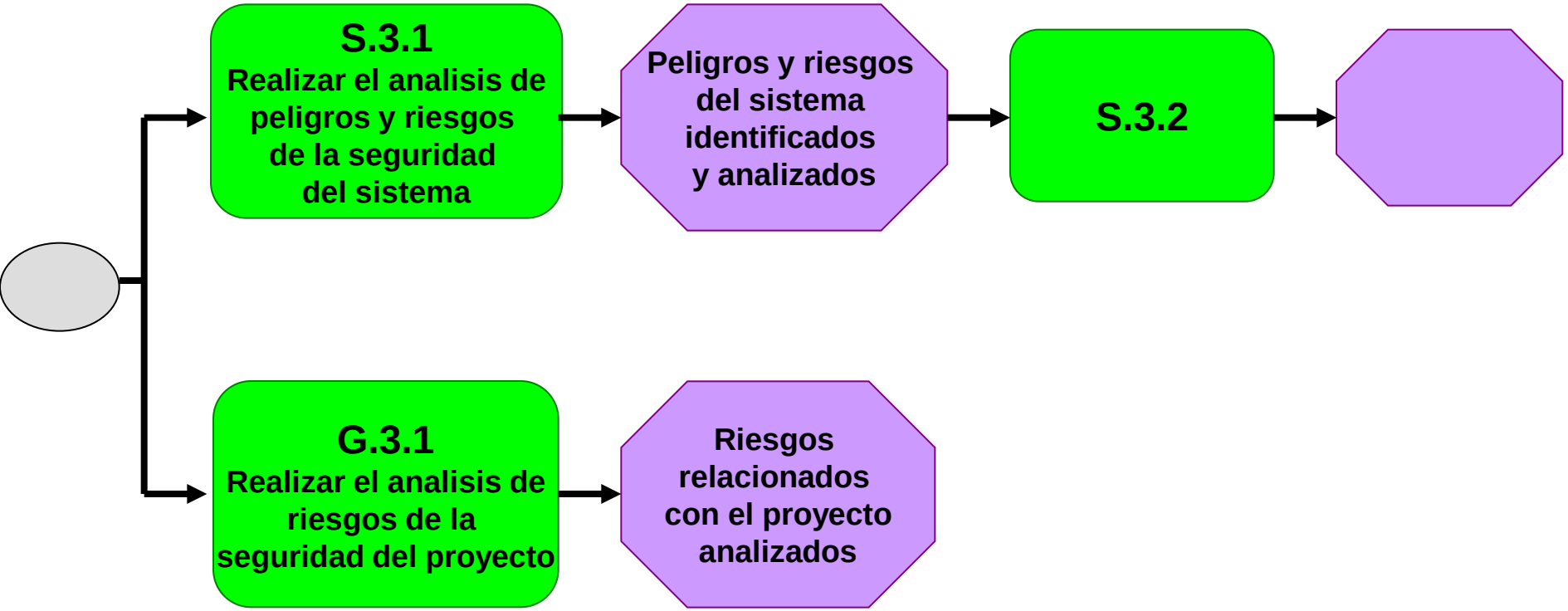
CASO DE SEGURIDAD

2. Resultado de las tareas



CASO DE SEGURIDAD

3. Que tareas se pueden realizar en paralelo y cuales deben seguir una secuencia



CASO DE SEGURIDAD

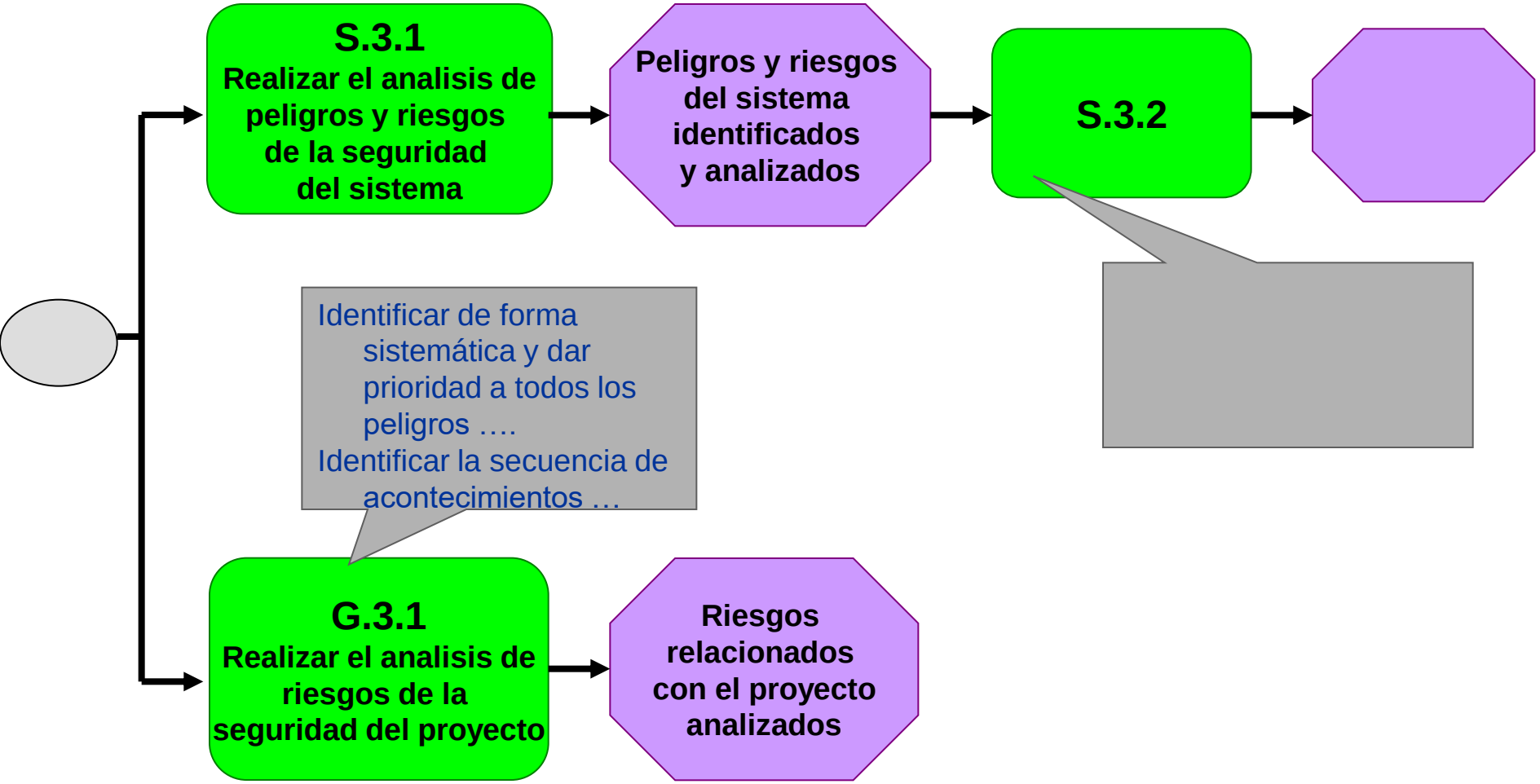
4. Que se requiere para realizar las tareas

6.3.3 Requisitos
6.3.3.1 El requisito 1 de esta fase es el de:
a) Identificar de forma sistemática y dar prioridad a todos los peligros ….
b) Identificar la secuencia de acontecimientos …

La norma nos indica los requerimientos para cada una de estas tareas

CASO DE SEGURIDAD

4. Que se requiere para realizar las tareas



CASO DE SEGURIDAD

5. Cuales son los documentos que se deben entregar

CASO DE SEGURIDAD

6. Que se debe verificar

6.3.5 Verificación

6.3.5.1 Las siguientes tareas de verificación se deben realizar en esta fase:

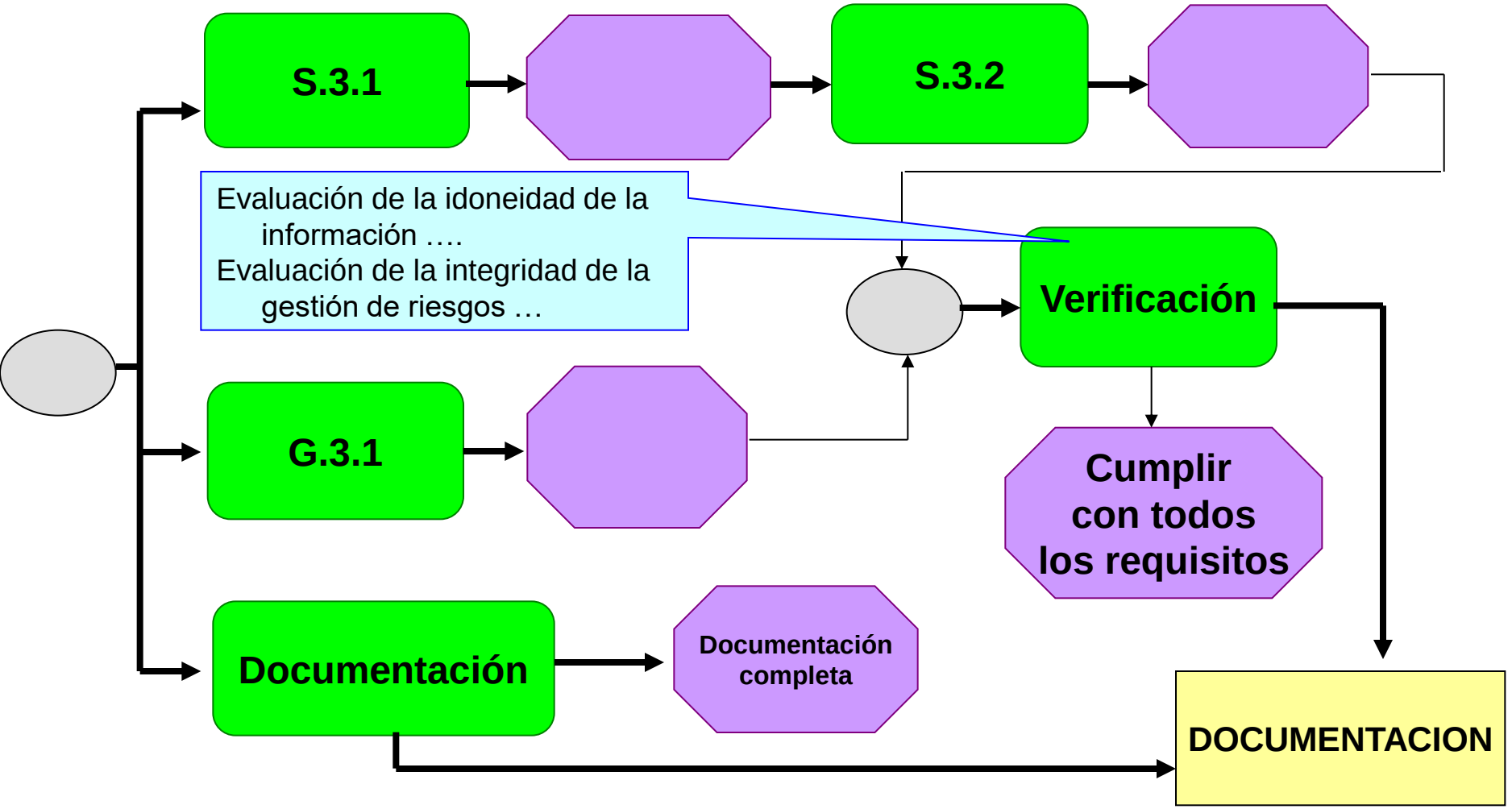
a)Evaluación de la idoneidad de la información ….

b)Evaluación de la integridad de la gestión de riesgos …

c) …..

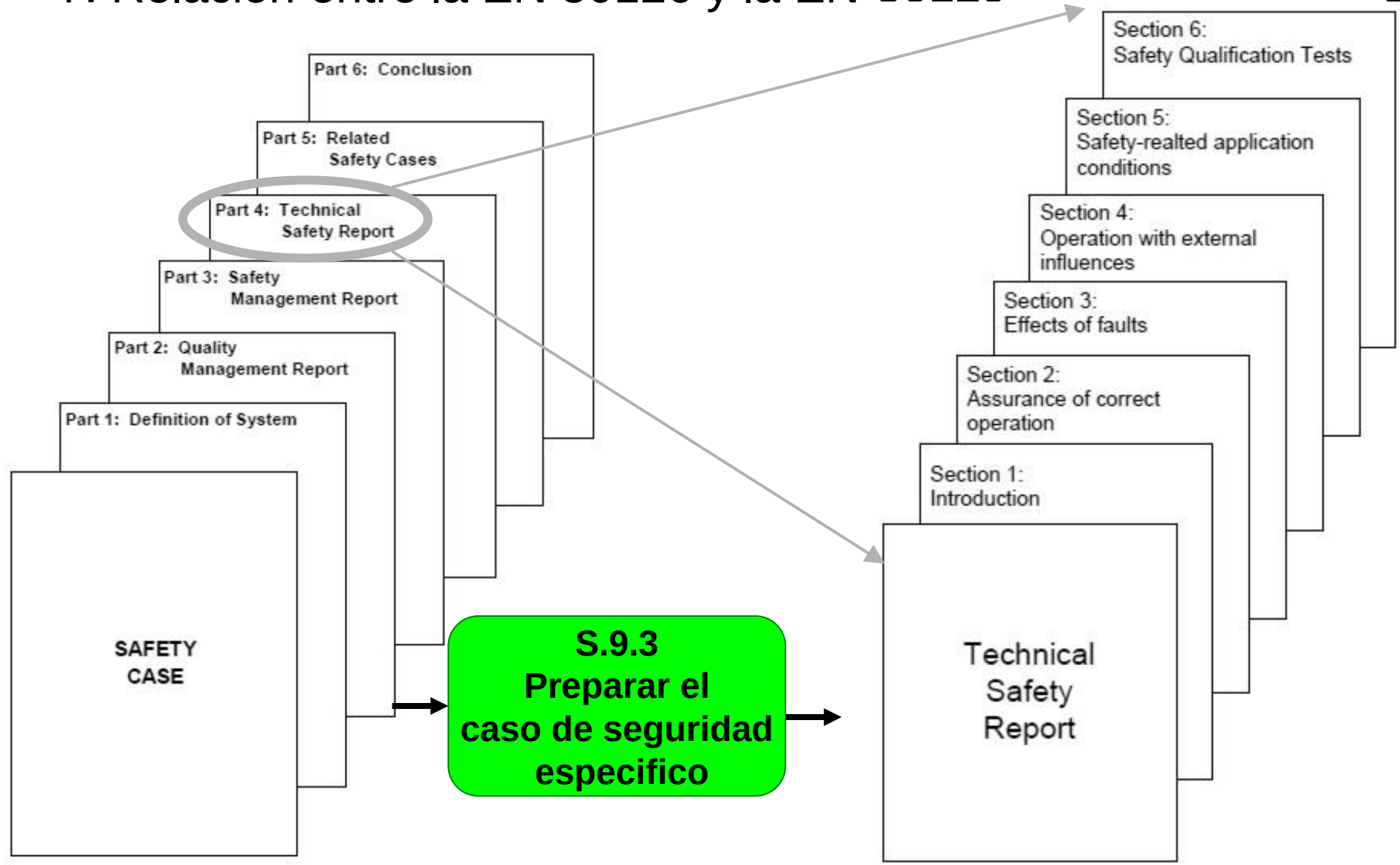
CASO DE SEGURIDAD

6. Que se debe verificar



CASO DE SEGURIDAD

7. Relación entre la EN 50126 y la EN-50129



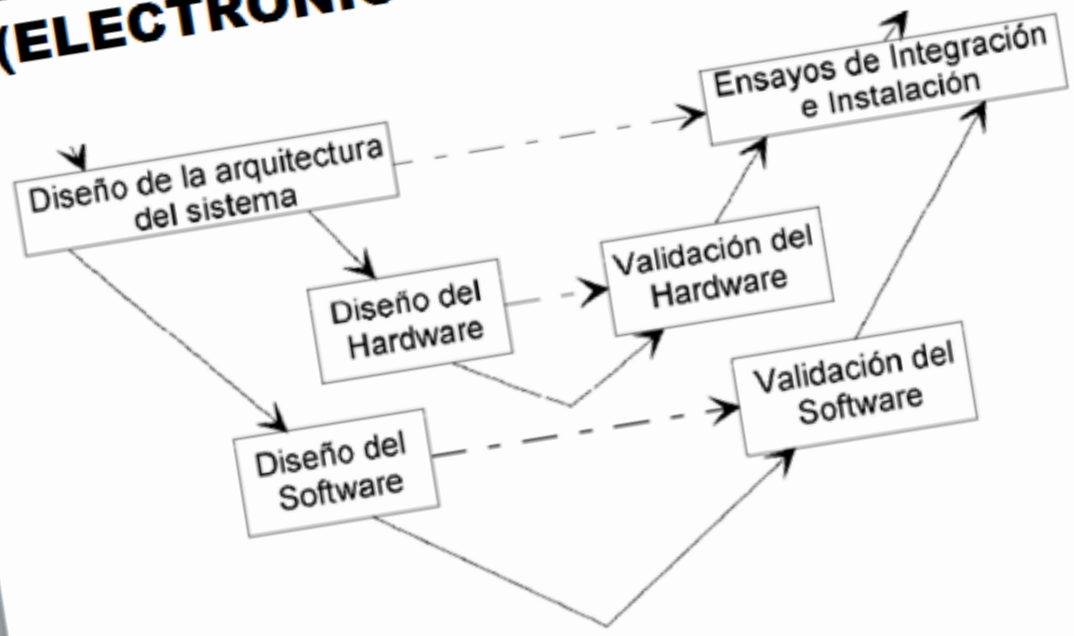
Fin anexo

Parte 2

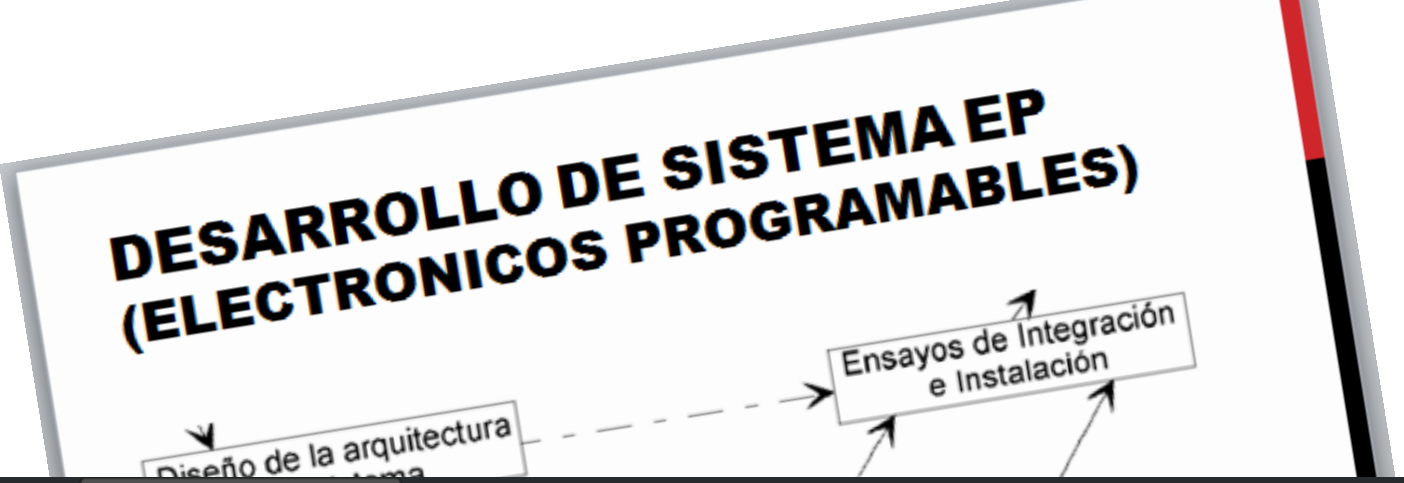
Dr. Ing. Emanuel Irrazabal (UNNE)

Cómo llevar a la realidad estas buenas prácticas?

DESARROLLO DE SISTEMA EP (ELECTRONICOS PROGRAMABLES)



21



safe scrum 50128 - Goog

Es seguro | https://scholar.google.com/scholar?q=safe+scrum+50128&btnG=&hl=es&as_sdt=0%2C5

Google safe scrum 50128

Académico Aproximadamente 47 resultados (**0,04** s) Mis citas 1

Artículos
Mi biblioteca

Cualquier momento
Desde 2017
Desde 2016
Desde 2013
Intervalo específico...

Ordenar por relevancia
Ordenar por fecha

Cualquier idioma
Buscar sólo

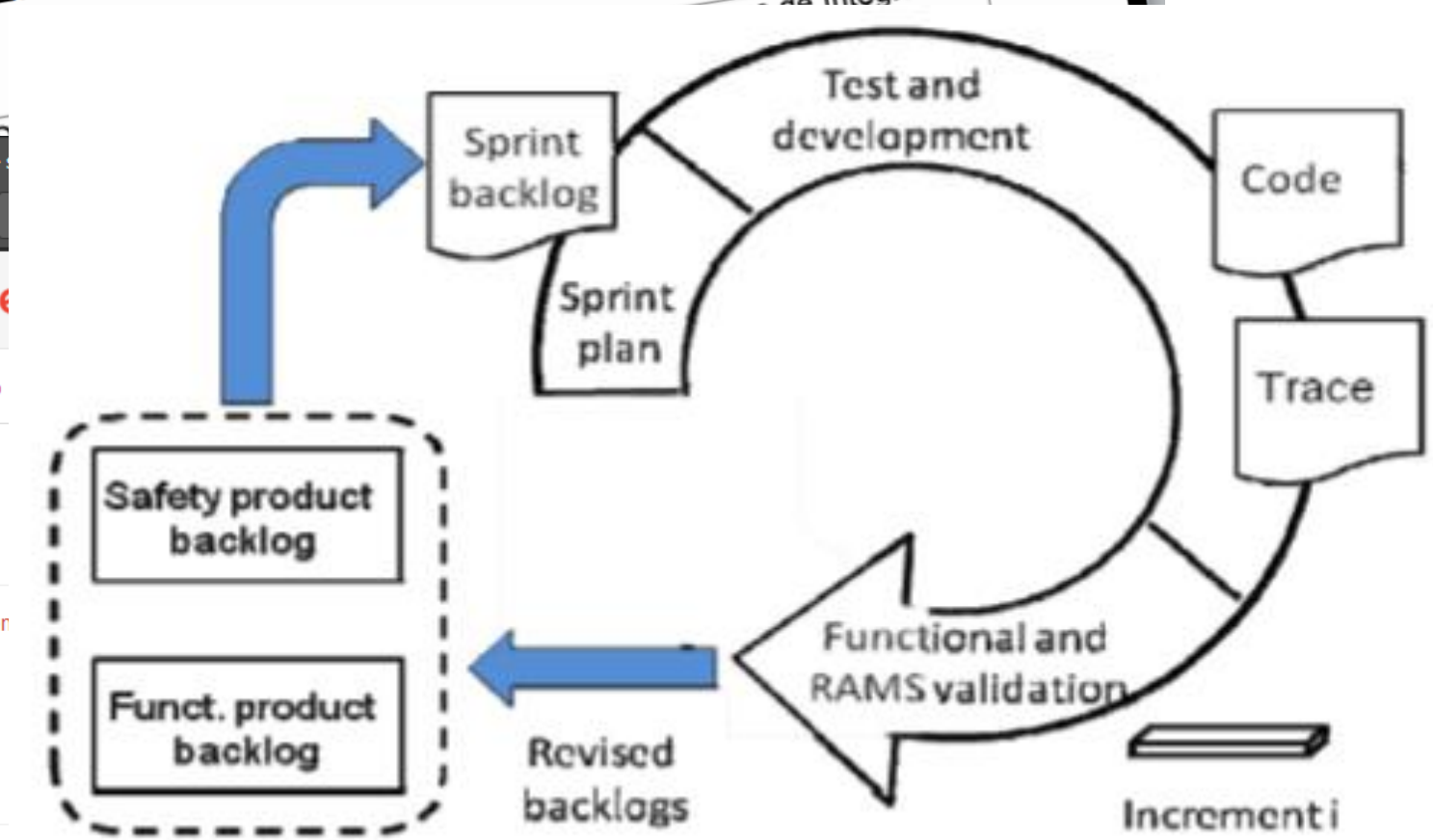
Sugerencia: Buscar solo resultados en **español**. Puedes especificar el idioma de búsqueda en Configuración de Google Académico..

[PDF] The application of **Safe Scrum** to IEC 61508 certifiable software [PDF] semanticscholar.org
T Stålhane, T Myklebust... - ... Probabilistic **Safety** ..., 2012 - pdfs.semanticscholar.org
... that will rise when trying to apply agile methods to aviation (DO-178B) and rail (EN **50128**) software. ... software in **Safe Scrum** while high level planning, systems design and decisions concerning **safety** – eg new **safety** requirements – are done outside the **Safe Scrum** process ...
Citado por 26 Artículos relacionados Las 5 versiones Citar Guardar Más

[PDF] Important considerations when applying other models than the Waterfall/V-model when developing software according to IEC 61508 or EN **50128** [PDF] researchgate.net
T Myklebust, T Stålhane, GK Hanssen - 2015 - researchgate.net
... Proceedings of the twenty-third **safety**-critical system symposium, Bristol, UK 3rd-5th February 2015 3. T. Stålhane, T ... The application of **Safe Scrum** to IEC 61508 certifiable software. ... Application of an Agile Development Prosess for EN **50128**/Railway conformant software. ...
Citado por 2 Artículos relacionados Citar Guardar Más

Challenges and Opportunities in Agile Development in **Safety** Critical Systems: A Survey
O Doss, TP Kelly - ACM SIGSOFT Software Engineering Notes, 2016 - dl.acm.org
... that either directly or indirectly address software **safety** assurance (eg IEC 61508, ISO 26262, EN **50128**, parts of ... responses also help inform the features of future integration attempts (eg

DESARROLLO DE SISTEMA EP (ELECTRONICOS PROGRAMABLES)



Requirements

Challenges and Opportunities in Agile Development in **Safety** Critical Systems: A Survey

O Doss, TP Kelly - ACM SIGSOFT Software Engineering Notes, 2016 - dl.acm.org

... that either directly or indirectly address software **safety** assurance (eg IEC 61508, ISO 26262, EN 50128, parts of ... responses also help inform the features of future integration attempts (eg

ESPECIFICACIÓN DE REQUISITOS DEL SOFTWARE

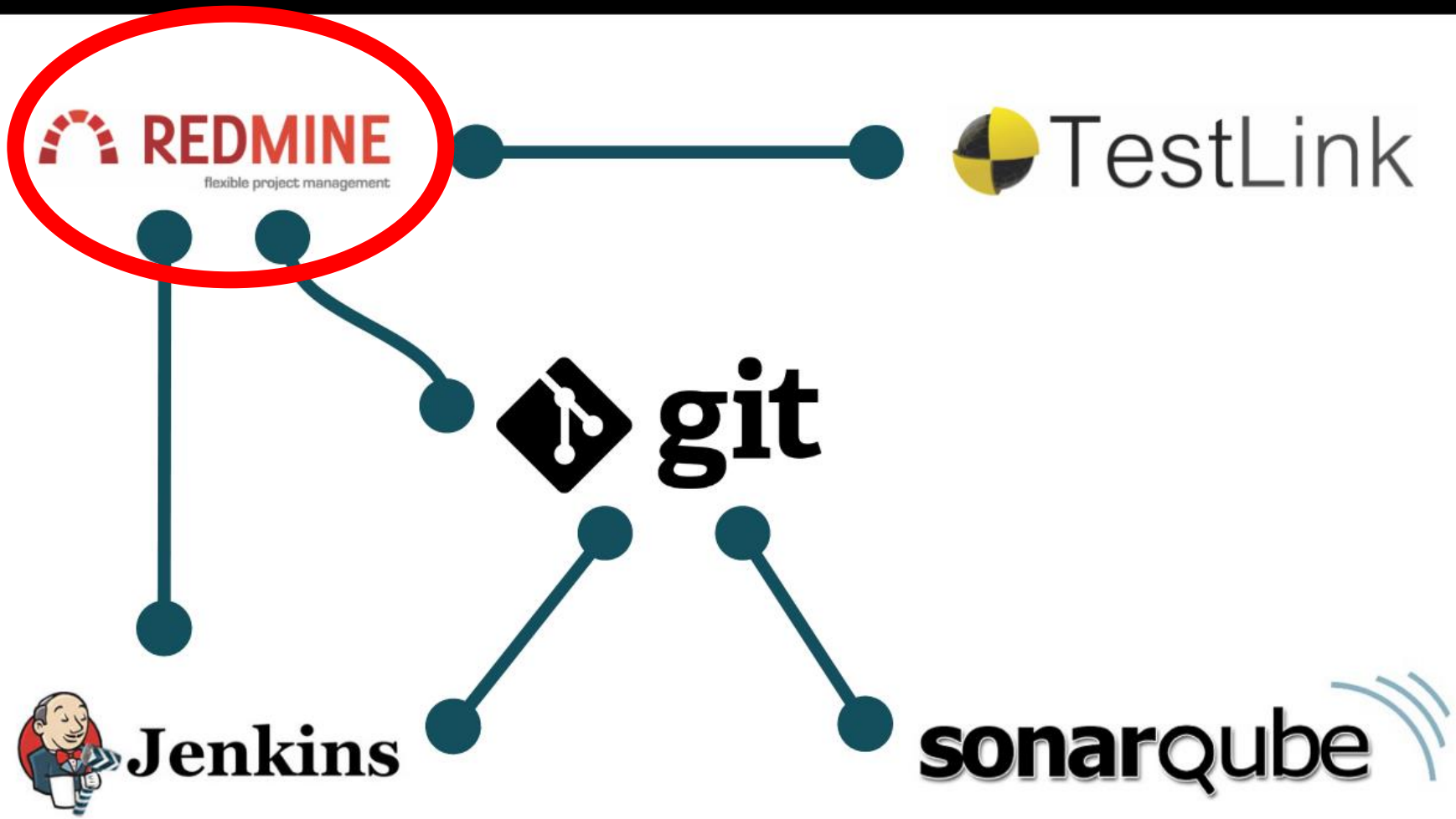
TÉCNICA/MEDIDA	Ref	SIL SW 0	SIL SW 1	SIL SW 2	SIL SW 3	SIL SW 4
1 Métodos Formales, incluyendo por ejemplo CCS, CSP, HOL, LOTOS, OBJ, Lógica Temporal, VDM Z y B	B.30	-	R	R	AR	AR
2 Métodos Semi-Formales	D.7	R	R	R	AR	AR
3 Metodología Estructurada, incluyendo por ejemplo JSD, MASCOT, SADT, SDL, SSADM y Yourdon	B.60	R	AR	AR	AR	AR

La especificación de requisitos, requiere siempre la descripción en lenguaje natural y la notación formal (matemática) que refleje tal especificación

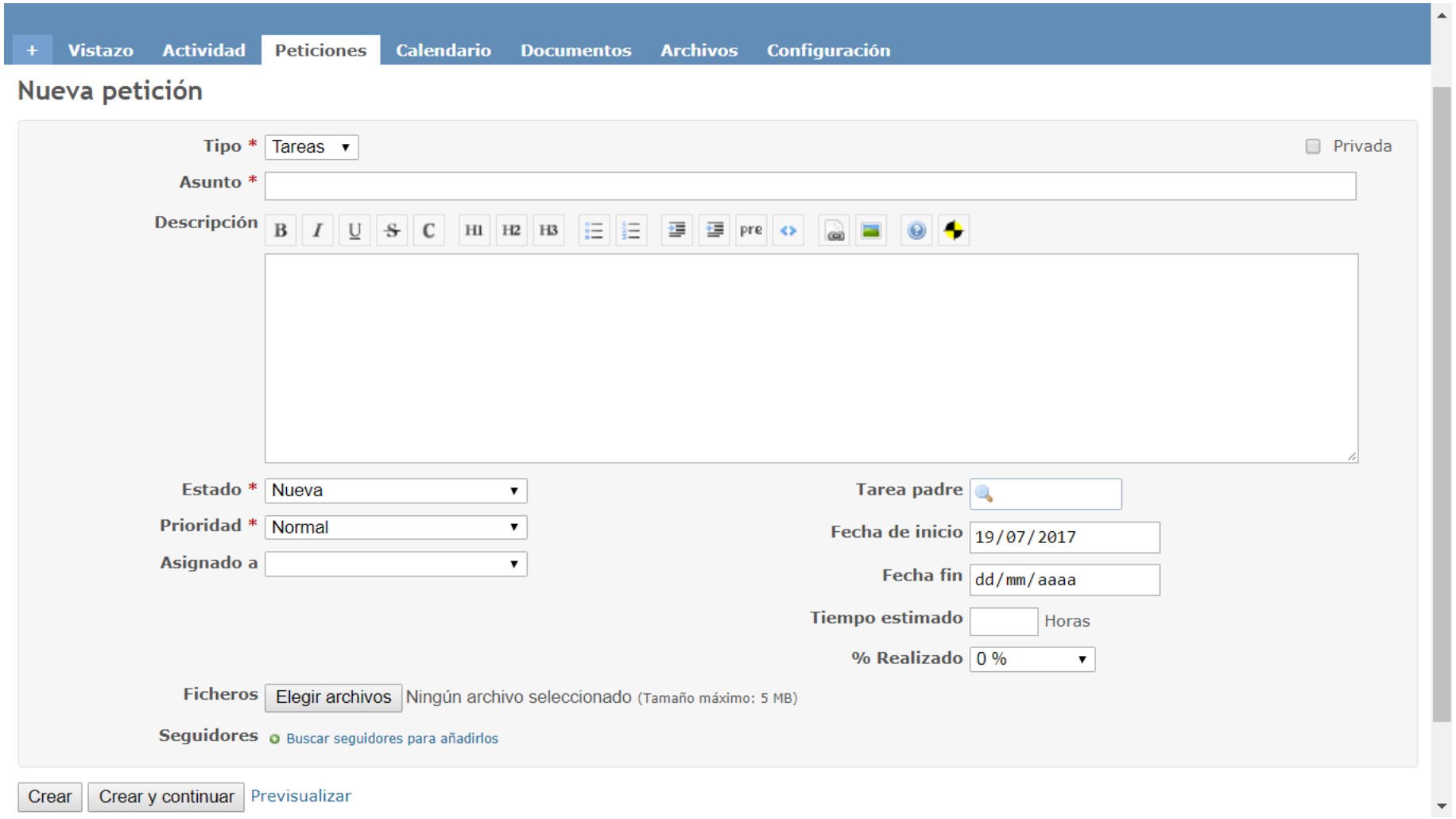
24

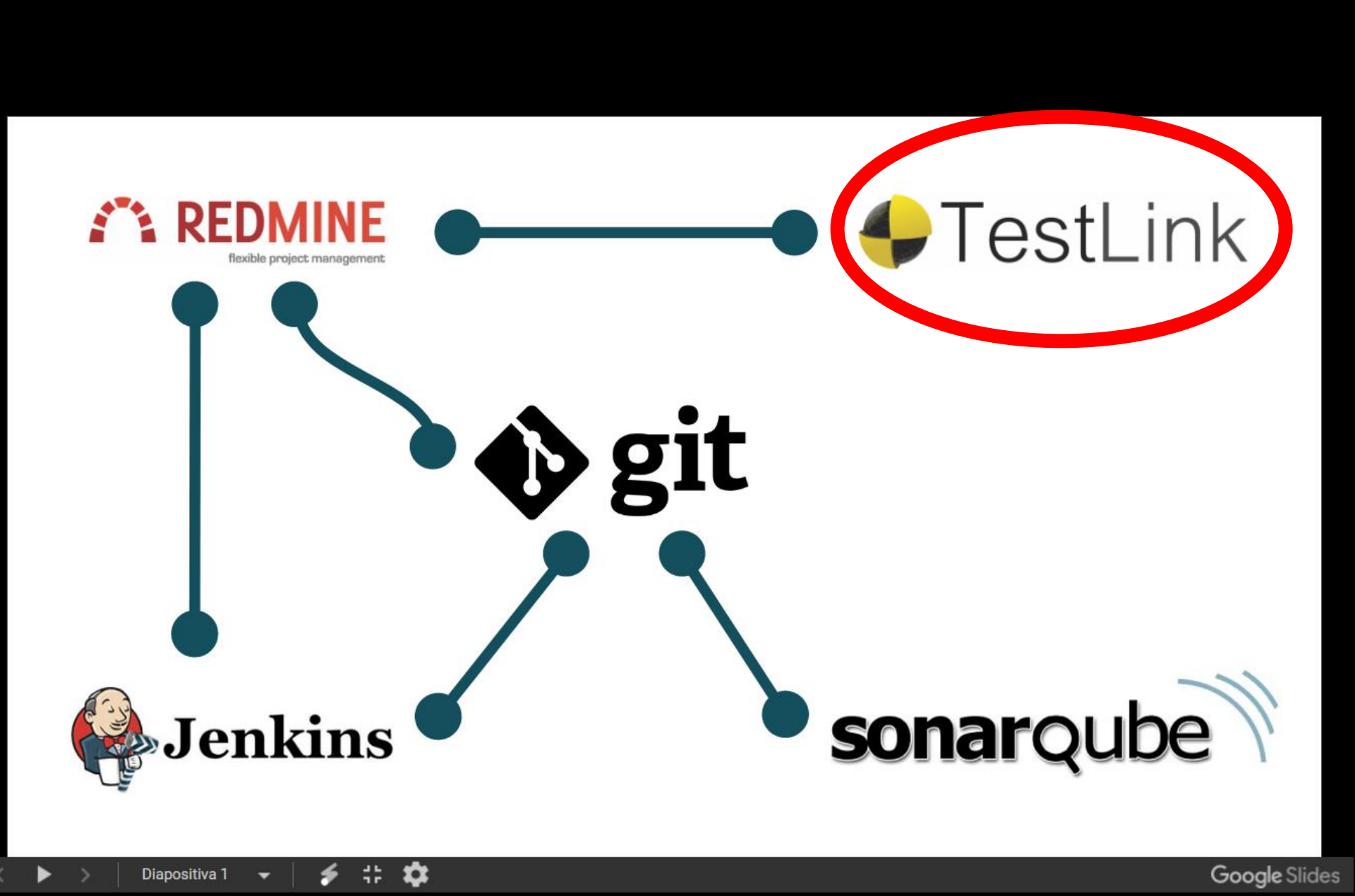

ESPECIFICACIÓN DE REQUISITOS DEL SOFTWARE
SIL SW 3
SIL SW 4
REDMINE
flexible project management
TestLink
git
Jenkins
sonarqube
Diapositiva 1
Google Slides


VOY A REGRESAR EN EL TIEMPO
A TOMAR REQUERIMIENTOS
memegen.es


REDMINE
flexible project management
TestLink
git
Jenkins
sonarqube


Inicio Mi página Proyectos Administración Ayuda
Conectado como sambrana Mi cuenta Terminar sesión
Gestor de Tareas y Defectos
Búsqueda:
Ir al proyecto...
Pulsa F11 para salir del modo de pantalla completa
Proyectos
Nuevo proyecto | Ver todas las peticiones | Tiempo total dedicado | Actividad global
Desarrollo Firmware - Monitor de Relé
Desarrollo de Firmware del Probador de Relé para las placas CIAA. Control de errores y defectos.
Desarrollo Software PC - Monitor de Relé
Desarrollo del software de PC para la gestión de datos, anomalías y comunicación del Probador de Relé. Control de errores y defectos.
Testlink-project
Projecto para probar integracion con Testlink
Mis proyectos
Exportar a: Atom
Proyectos
Ver proyectos cerrados
Aceptar


+
Vistazo
Actividad
Peticiones
Calendario
Documentos
Archivos
Configuración
Nueva petición
Tipo *
Tareas
Privada
Asunto *
Descripción
B
I
U
S
C
H1
H2
H3
pre
Estado *
Nueva
Prioridad *
Normal
Asignado a
Tarea padre
Fecha de inicio
19/07/2017
Fecha fin
dd/mm/aaaa
Tiempo estimado
Horas
% Realizado
0 %
Ficheros
Elegir archivos
Ningún archivo seleccionado
(Tamaño máximo: 5 MB)
Seguidores
Buscar seguidores para añadirlos
Crear
Crear y continuar
Previsualizar


REDMINE
flexible project management
TestLink
git
Jenkins
sonarqube

TestLink 1.9.16 (Moka po

190.188.255.130:224/testlink/index.php?caller=login&viewer=

TestLink

lider [leader]

TestLink 1.9.16 (Moka pot)

BA-

Proyecto de Pruebas BA:Pruebas Firmware Bateria

Gestión de Proyectos de Pruebas

Asignar Roles a Usuarios
Gestión de Keywords
Gestión de Plataformas

Requisitos

Documento de Especificación de Requisitos
Resumen de Requisitos
Buscar Requisitos
Buscar Especificación de Requisitos
Asignar Requisitos
Requirement Monitoring Overview
Imprimir Especificación de Requisitos

Especificación de Pruebas

Editar Caso(s) de Prueba
Casos de Prueba creados por Usuario

Gestión de Planes de Pruebas

Gestión de Planes de Pruebas

13:59
16/6/17

PROGRAMACIÓN DEFENSIVA ?

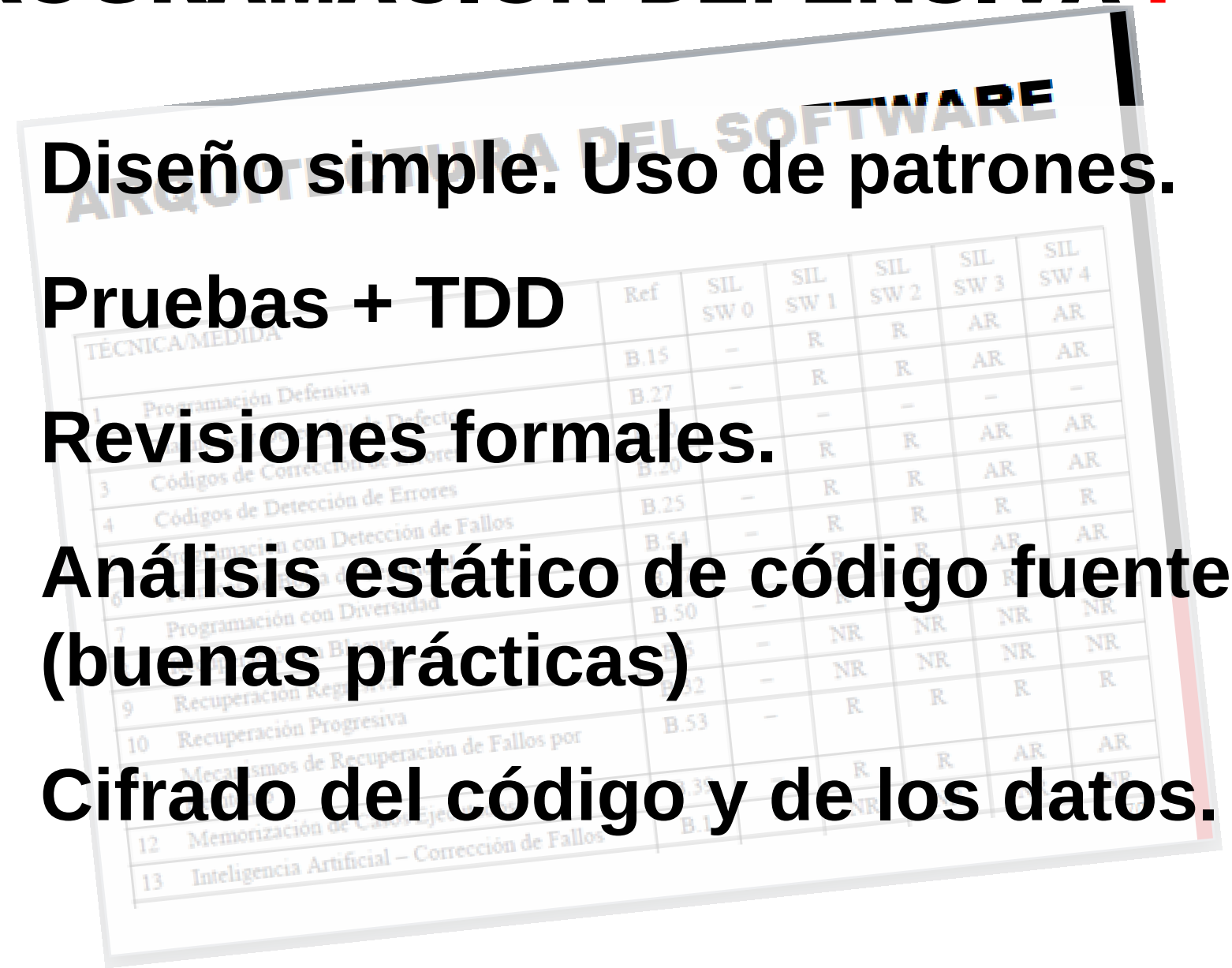
ARQUITECTURA DEL SOFTWARE

	TÉCNICA/MEDIDA	Ref	SIL SW 0	SIL SW 1	SIL SW 2	SIL SW 3	SIL SW 4
1	Programación Defensiva	B.15	–	R	R	AR	AR
2	Diagnosis y Detección de Defectos	B.27	–	R	R	AR	AR
3	Códigos de Corrección de Errores	B.20	–	–	–	–	–
4	Códigos de Detección de Errores	B.20	–	R	R	AR	AR
5	Programación con Detección de Fallos	B.25	–	R	R	AR	AR
6	Técnicas de Bolsa de Seguridad	B.54	–	R	R	R	R
7	Programación con Diversidad	B.17	–	R	R	AR	AR
8	Recuperación en Bloque	B.50	–	R	R	R	R
9	Recuperación Regresiva	B.5	–	NR	NR	NR	NR
10	Recuperación Progresiva	B.32	–	NR	NR	NR	NR
11	Mecanismos de Recuperación de Fallos por Reintento	B.53	–	R	R	R	R
12	Memorización de Casos Ejecutados	B.39	–	R	R	AR	AR
13	Inteligencia Artificial – Corrección de Fallos	B.1	–	NR	NR	NR	NR

70

PROGRAMACIÓN DEFENSIVA !

- Diseño simple. Uso de patrones.
- Pruebas + TDD
- Revisiones formales.
- Análisis estático de código fuente (buenas prácticas)
- Cifrado del código y de los datos.



VERIFICACIÓN Y ENSAYO

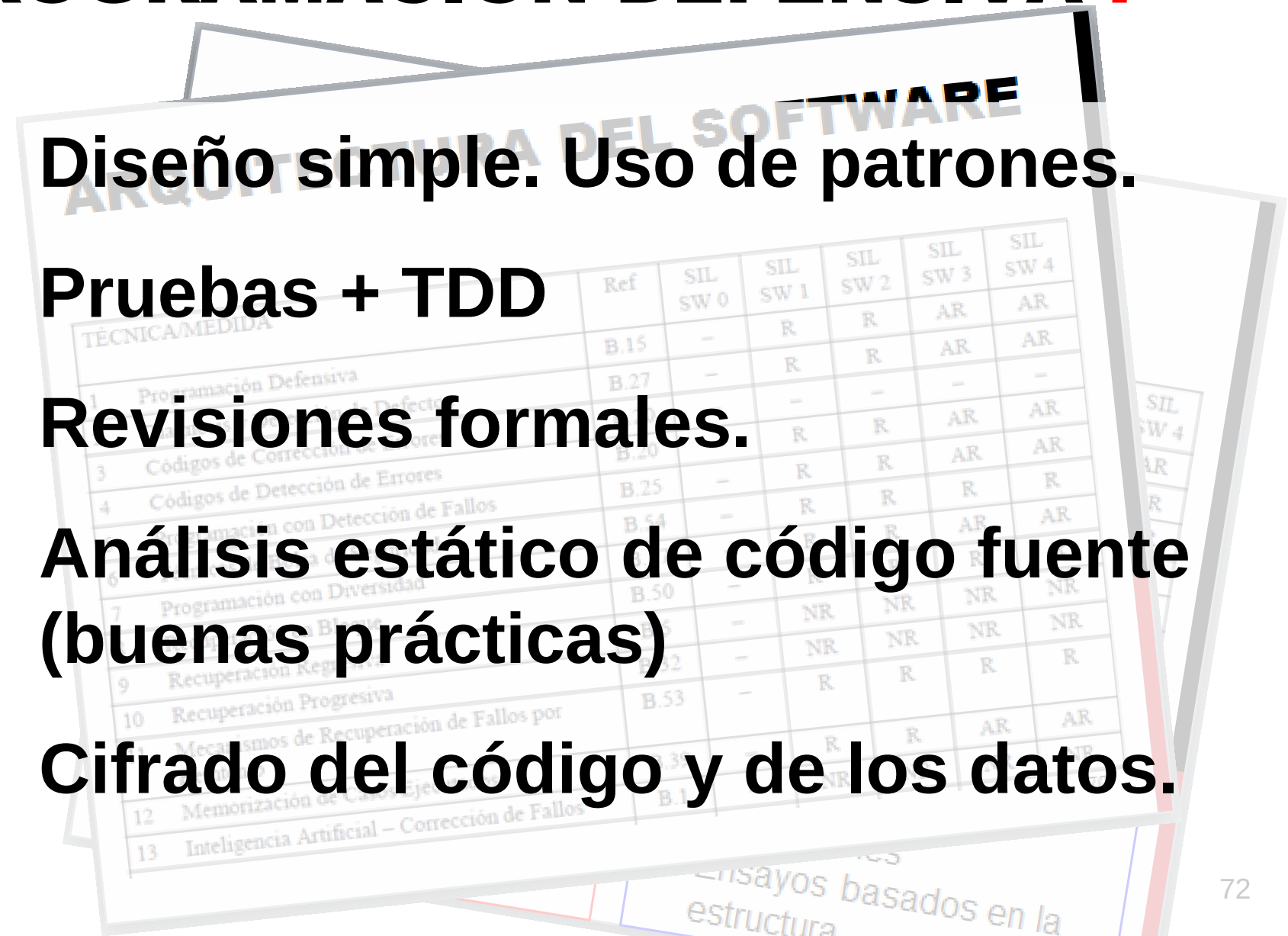
TÉCNICA/MEDIDA	Ref	SIL SW 0	SIL SW 1	SIL SW 2	SIL SW 3	SIL SW 4
1 Ensayo Formal	B.31	–	R	R	AR	AR
2 Ensayos Probabilisticos	B.47	–	R	R	AR	AR
3 Análisis Estático	D.8	–	AR	AR	AR	AR
4 Análisis y Ensayos Dinámicos	D.2	–	AR	AR	AR	AR
5 Evaluaciones	B.42		R	R	R	R
6 Matriz de Trazabilidad	B.69	–	R	R	AR	AR
7 Análisis de Efectos de Errores software	B.26	–	R	R	AR	AR

1. Análisis de flujo de control
2. Análisis de flujo de datos
3. Ensayos y revisiones de diseño

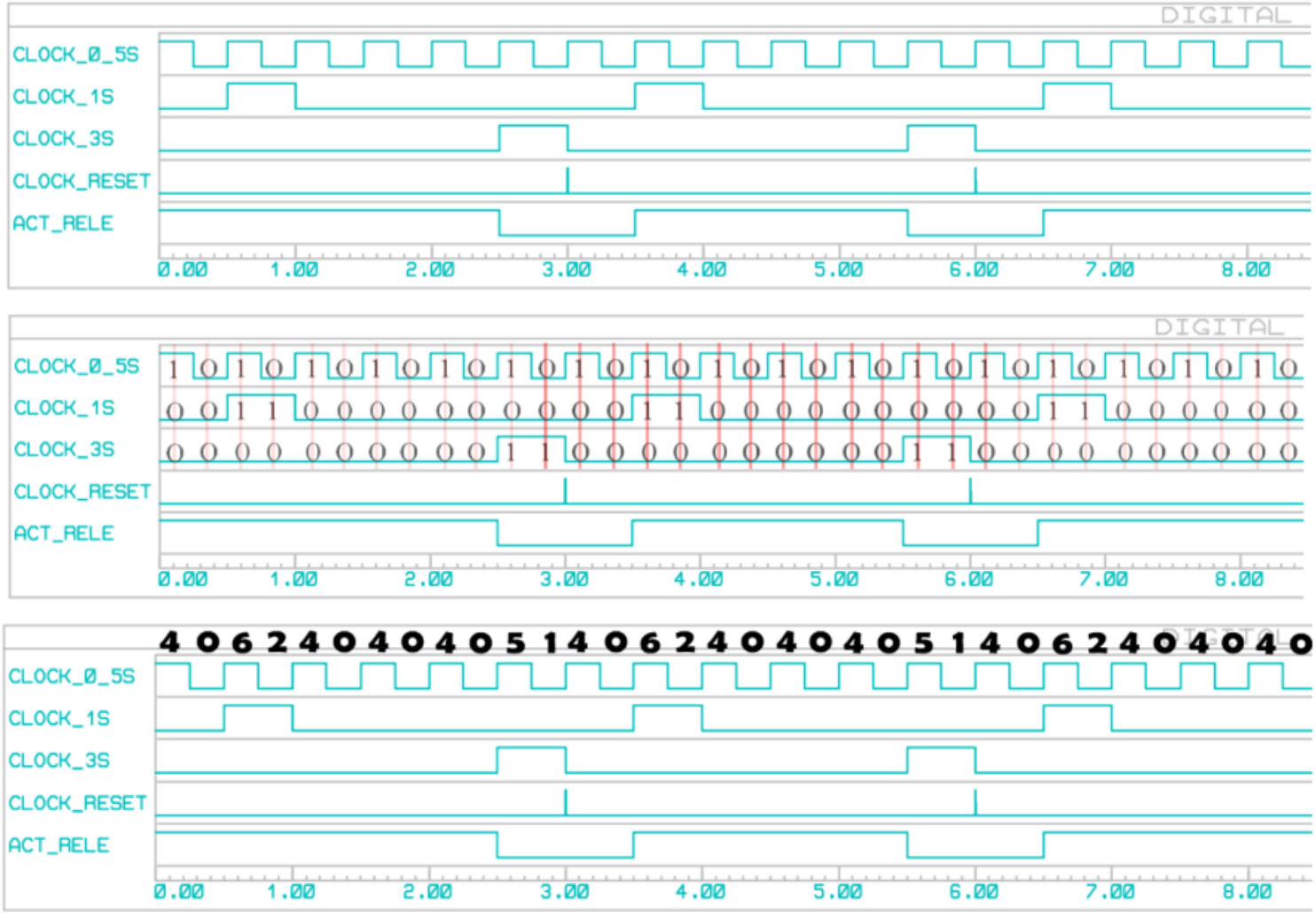
1. Casos de ensayo a partir de valores extremos
2. Modelado de prestaciones
3. Ensayos basados en la estructura

PROGRAMACIÓN DEFENSIVA !

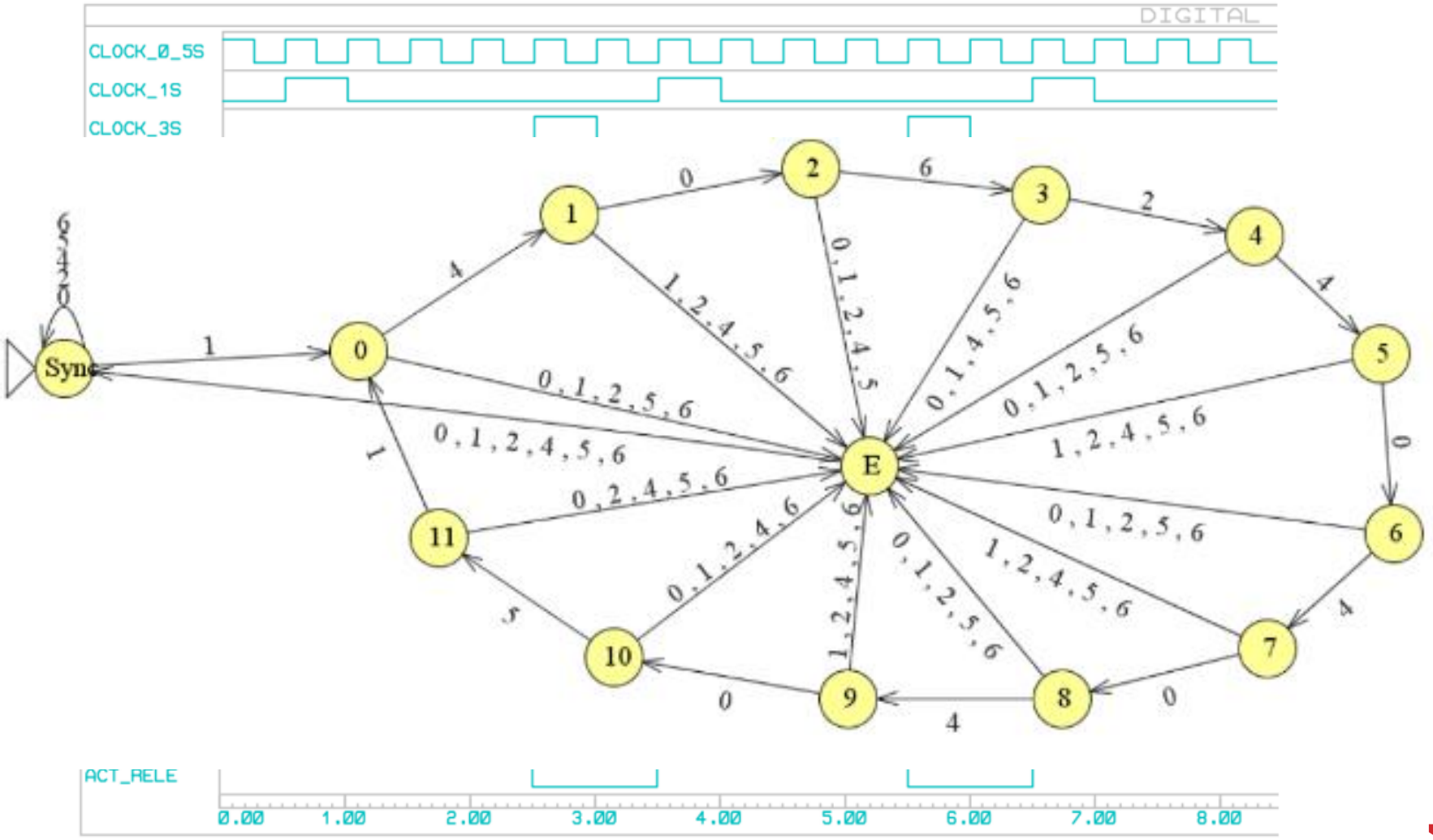
- Diseño simple. Uso de patrones.
- Pruebas + TDD
- Revisiones formales.
- Análisis estático de código fuente (buenas prácticas)
- Cifrado del código y de los datos.



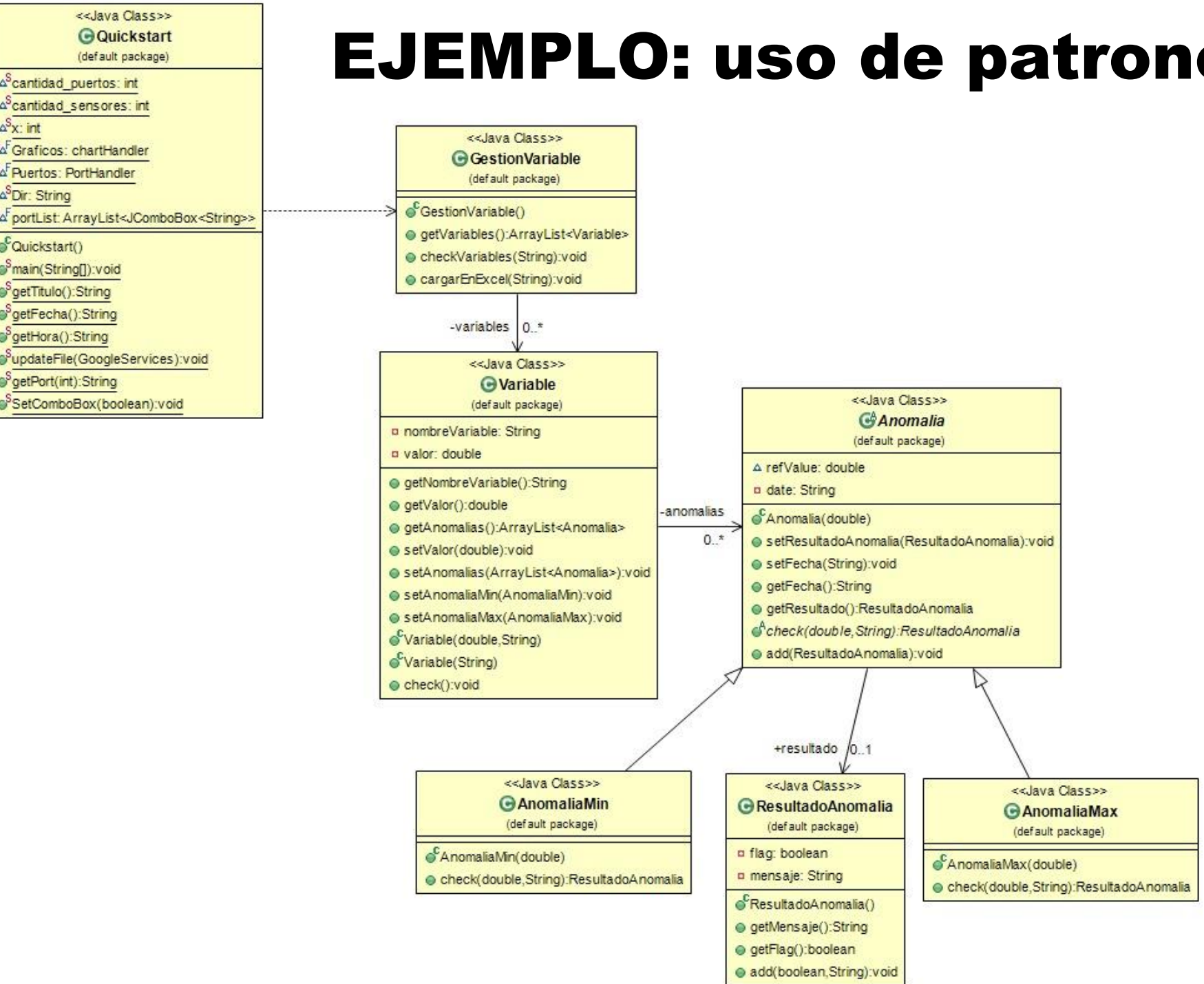
EJEMPLO: detección y corrección de errores



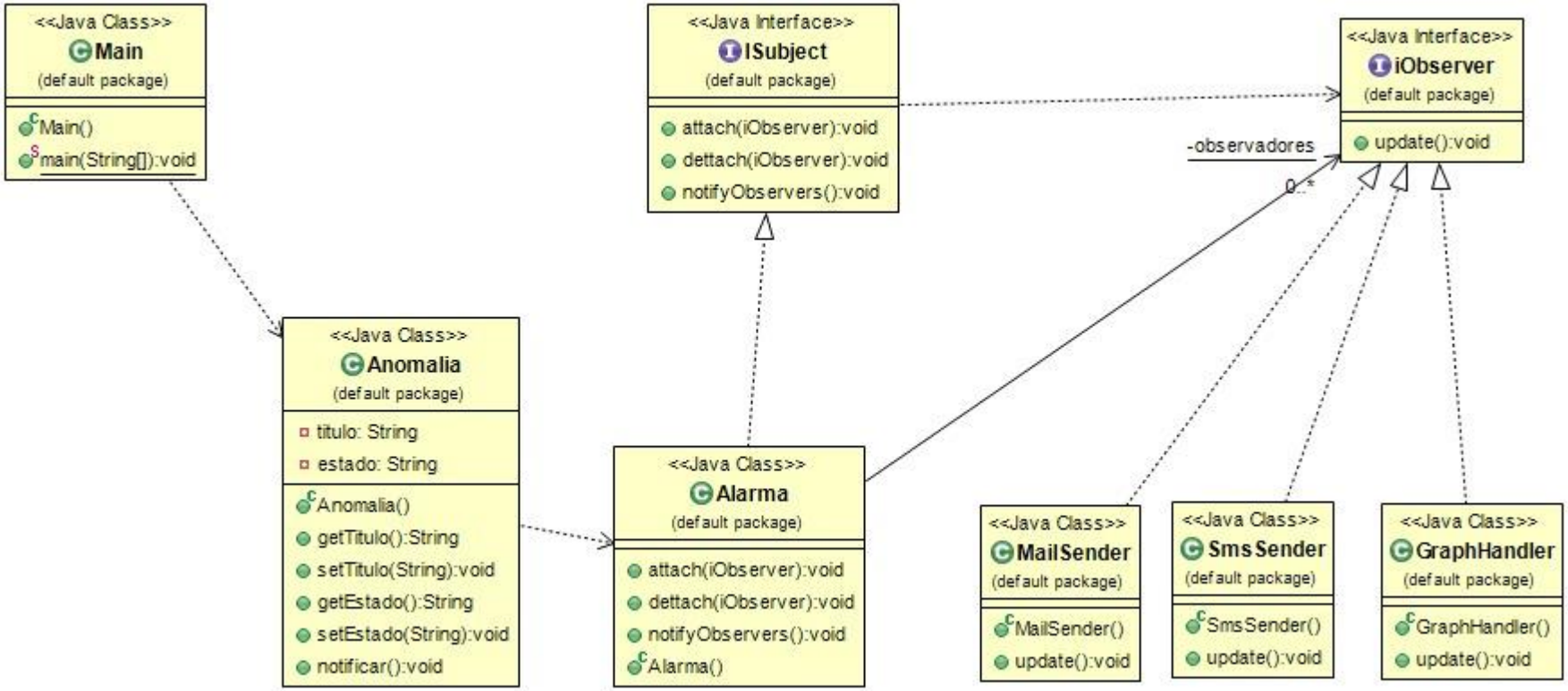
EJEMPLO: detección y corrección de errores



EJEMPLO: uso de patrones



EJEMPLO: uso de patrones



CUANDO ESCRIBÍ ESTE CÓDIGO,
SÓLO DIOS Y YO SABÍAMOS
CÓMO Y PARA QUÉ LO HICE

AHORA, SÓLO DIOS LO SABE


TE JURO QUE EN MI PC SI FUNCIONA
ENTIENDO, PERO EN EL AMBIENTE DEL CLIENTE FALLÓ

VIDA REAL
DE UN EQUIPO DE TRABAJO

Los cambios de cada desarrollador deben ser integrados (constantemente)

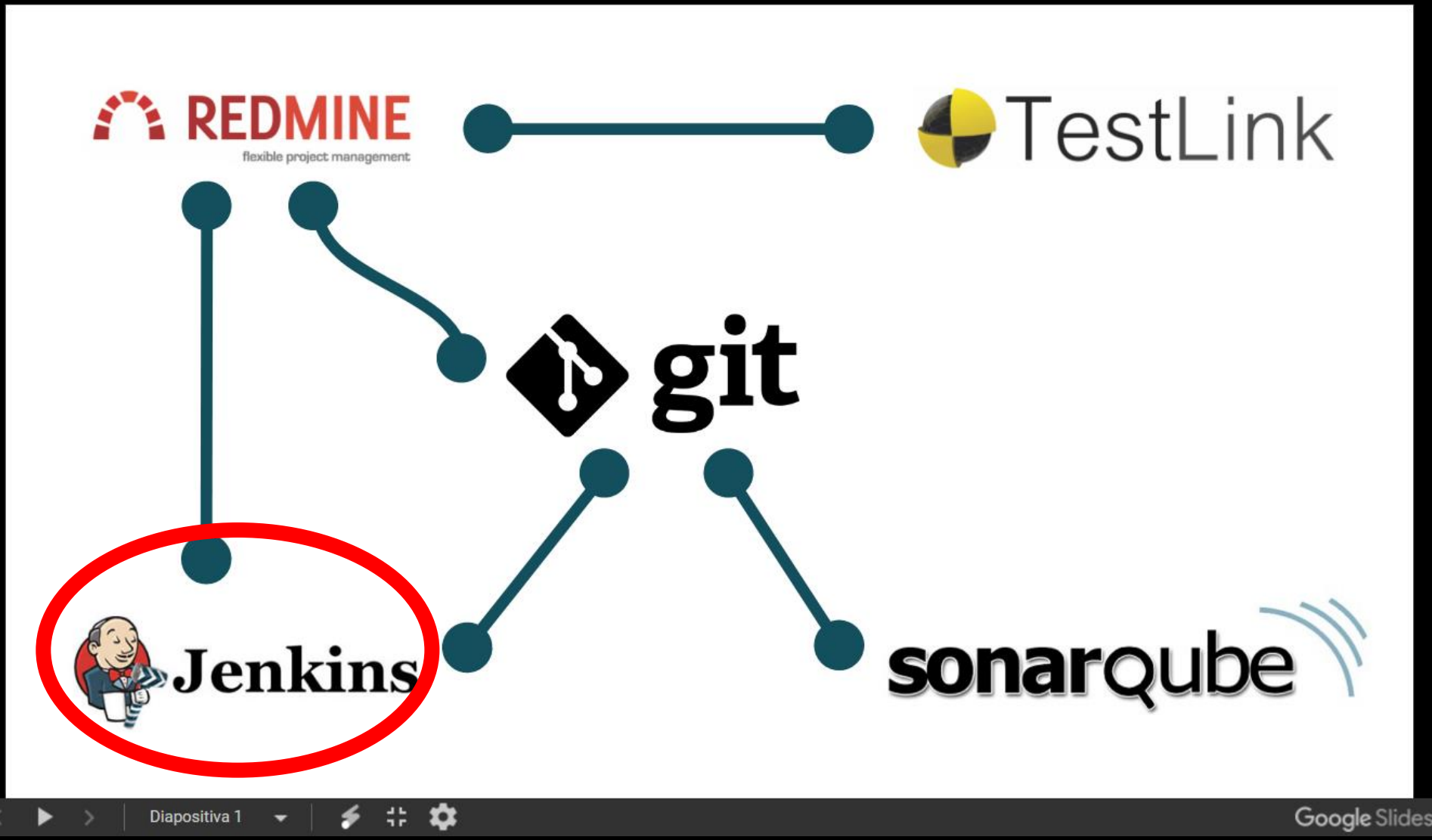
Y sus pruebas también

La forma de construir el software debe ser continua

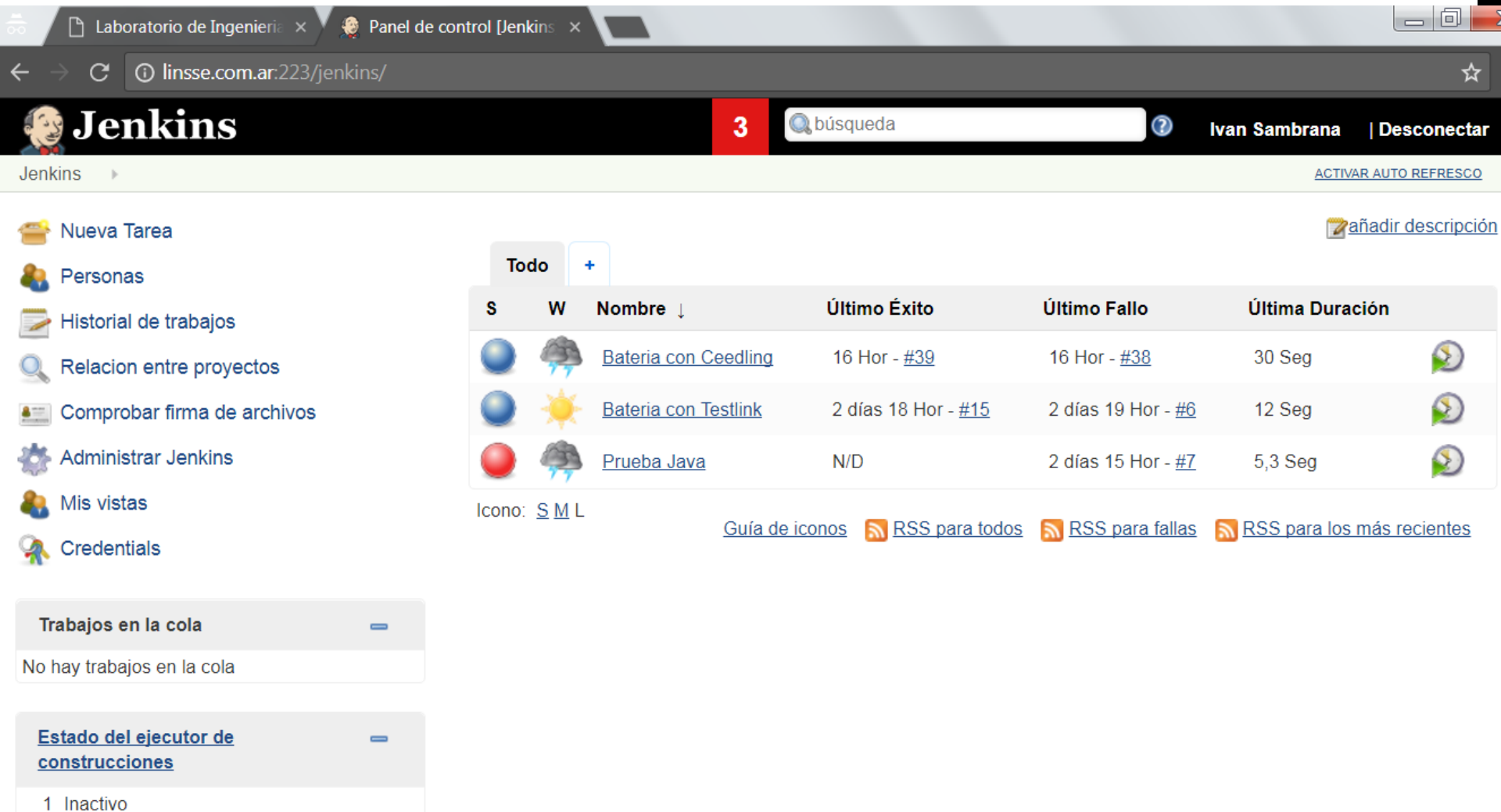
INTEGRACIÓN CONTINUA

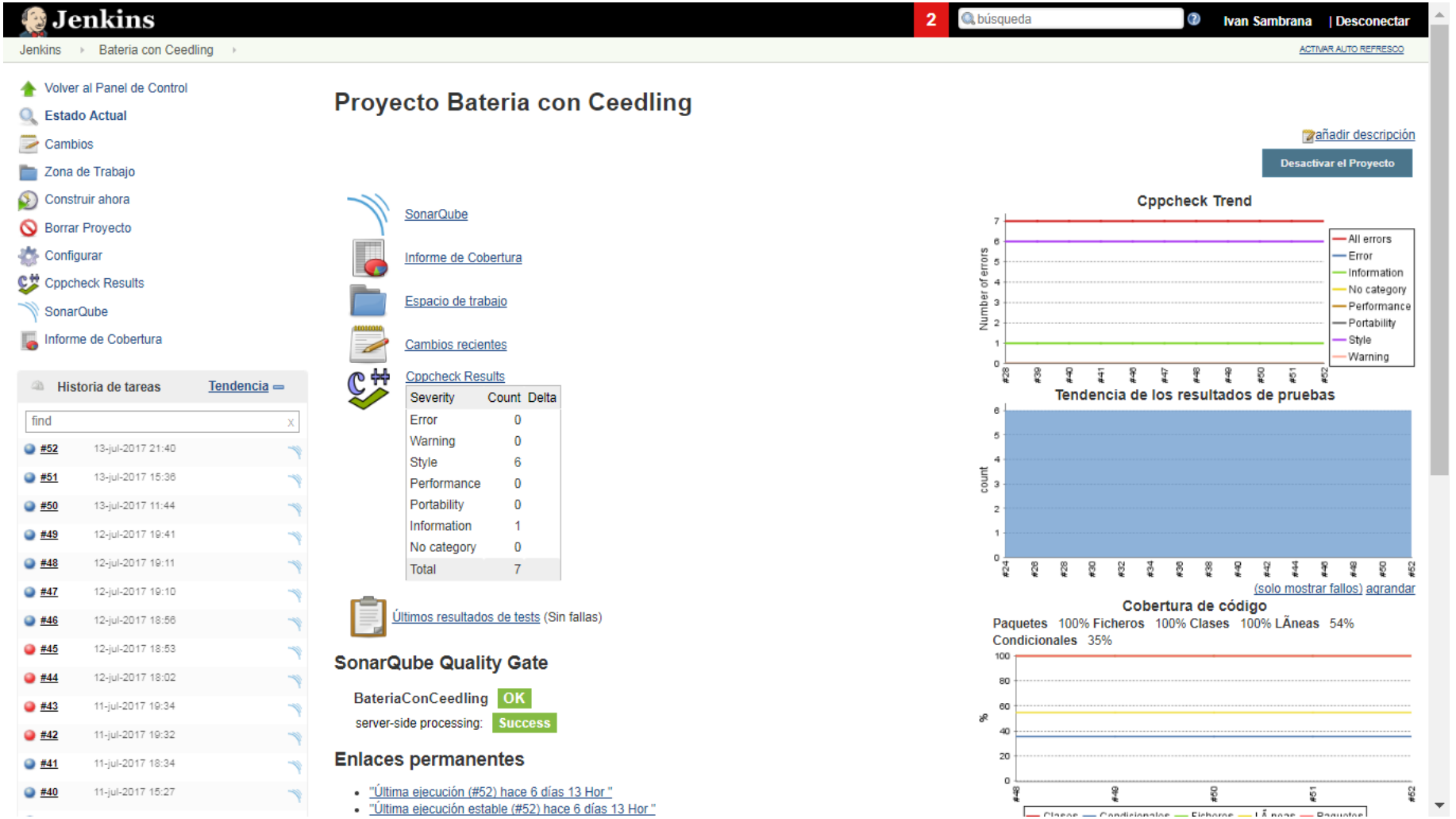
La integración continua consiste en construir el software en períodos extremos (cada día?).

Extremo: cada check-in inicia el proceso de construcción del código fuente


REDMINE
flexible project management
TestLink
git
Jenkins
sonarqube

JENKINS


Laboratorio de Ingenieria
Panel de control [Jenkins
linsse.com.ar:223/jenkins/
Jenkins
3
búsqueda
Ivan Sambrana
| Desconectar
Jenkins
ACTIVAR AUTO REFRESCO
Nueva Tarea
Personas
Historial de trabajos
Relacion entre proyectos
Comprobar firma de archivos
Administrar Jenkins
Mis vistas
Credentials
añadir descripción
Todo
+
S
W
Nombre ↓
Último Éxito
Último Fallo
Última Duración
Bateria con Ceedling
16 Hor - #39
16 Hor - #38
30 Seg
Bateria con Testlink
2 días 18 Hor - #15
2 días 19 Hor - #6
12 Seg
Prueba Java
N/D
2 días 15 Hor - #7
5,3 Seg
Icono: S M L
Guía de iconos
RSS para todos
RSS para fallas
RSS para los más recientes
Trabajos en la cola
No hay trabajos en la cola
Estado del ejecutor de construcciones
1 Inactivo


Jenkins
2
búsqueda
Ivan Sambrana | Desconectar
Jenkins › Bateria con Ceedling ›
ACTIVAR AUTO REFRESCO
Volver al Panel de Control
Estado Actual
Cambios
Zona de Trabajo
Construir ahora
Borrar Proyecto
Configurar
Cppcheck Results
SonarQube
Informe de Cobertura
Historia de tareas
Tendencia
find
#52 13-jul-2017 21:40
#51 13-jul-2017 15:36
#50 13-jul-2017 11:44
#49 12-jul-2017 19:41
#48 12-jul-2017 19:11
#47 12-jul-2017 19:10
#46 12-jul-2017 18:56
#45 12-jul-2017 18:53
#44 12-jul-2017 18:02
#43 11-jul-2017 19:34
#42 11-jul-2017 19:32
#41 11-jul-2017 18:34
#40 11-jul-2017 15:27
Proyecto Bateria con Ceedling
añadir descripción
Desactivar el Proyecto
SonarQube
Informe de Cobertura
Espacio de trabajo
Cambios recientes
Cppcheck Results
Severity Count Delta
Error 0
Warning 0
Style 6
Performance 0
Portability 0
Information 1
No category 0
Total 7
Últimos resultados de tests (Sin fallas)
SonarQube Quality Gate
BateriaConCeedling OK
server-side processing: Success
Enlaces permanentes
"Última ejecución (#52) hace 6 días 13 Hor "
"Última ejecución estable (#52) hace 6 días 13 Hor "
Cppcheck Trend
Number of errors
All errors
Error
Information
No category
Performance
Portability
Style
Warning
Tendencia de los resultados de pruebas
count
(solo mostrar fallos) agrandar
Cobertura de código
Paquetes 100% Ficheros 100% Clases 100% LÃneas 54%
Condicionales 35%
%

Jenkins 2 búsqueda Ivan Sambrana | Desconectar

Jenkins › Bateria con Ceedling › #52

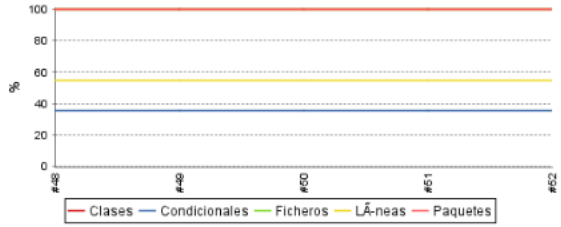
- Volver al proyecto
- Estatus
- Cambios
- Console Output
- Editar información de la ejecución
- Borrar Proyecto
- Git Build Data
- No Tags
- Informe de Cobertura
- Cppcheck Results
- Resultado de los tests
- Ejecución previa

Pulsa F11 para salir del modo de pantalla completa

Cobertura de código

Cobertura

Tendencia



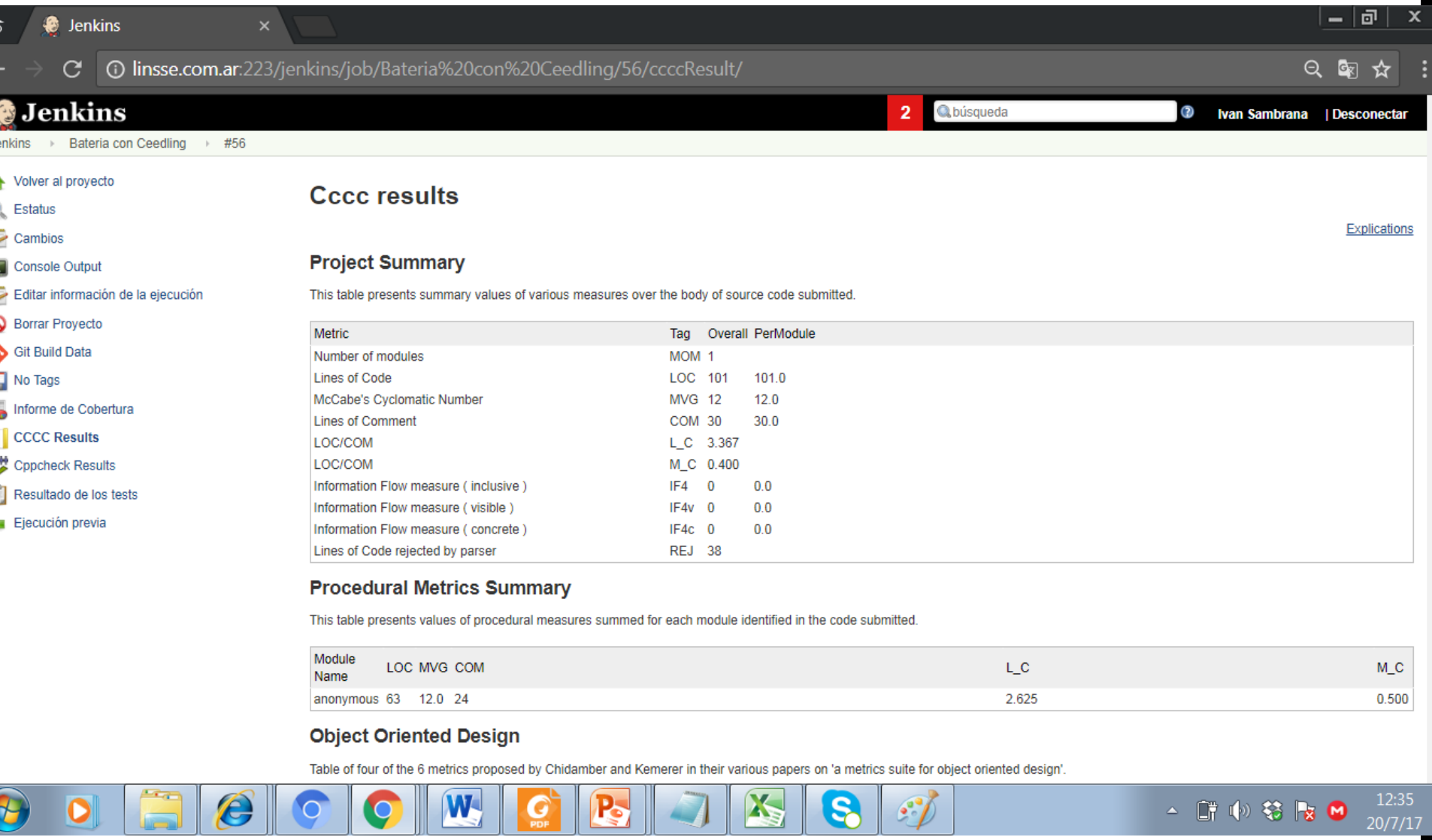
Proyecto (resumen)

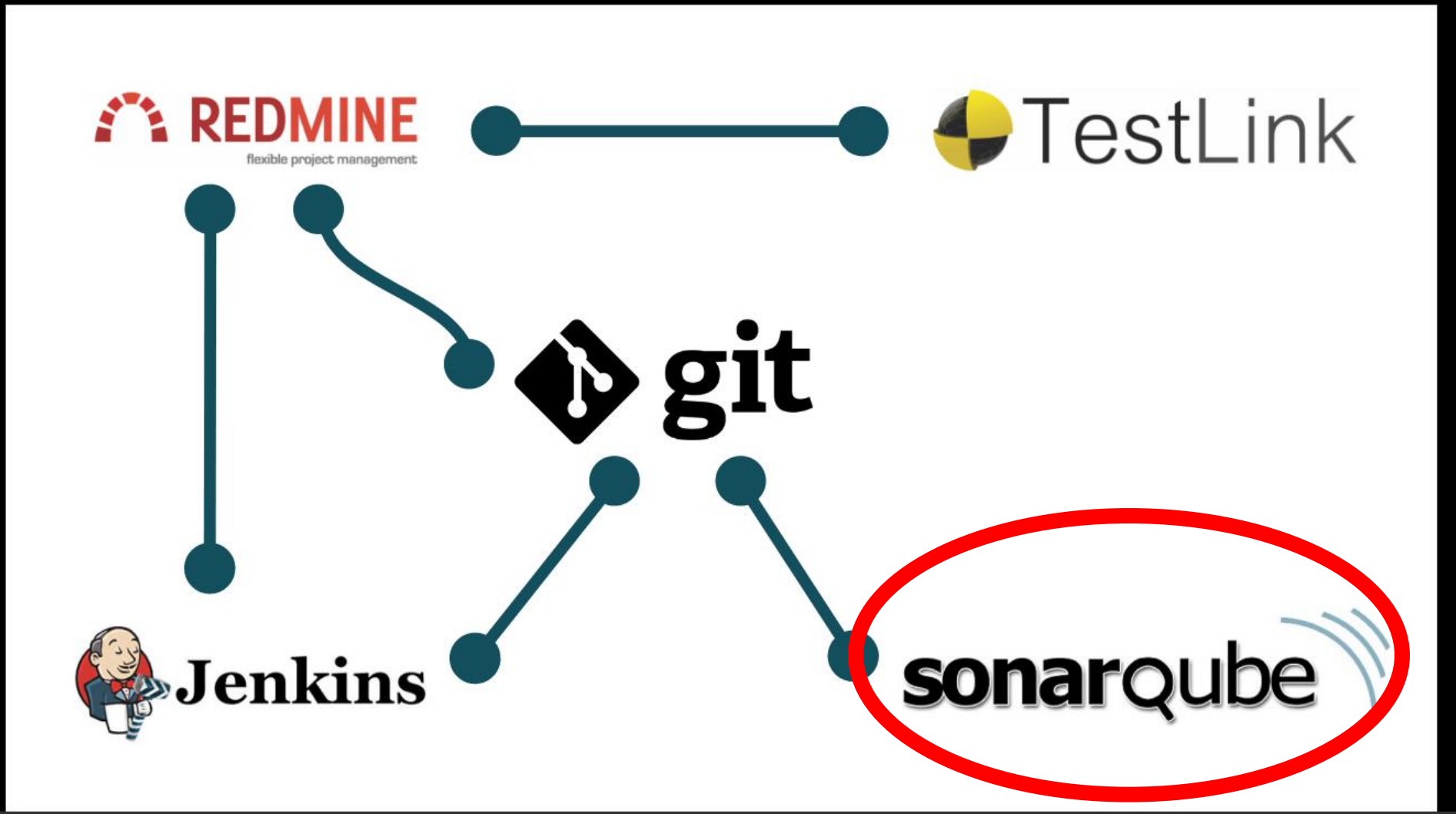
Nombre	Paquetes	Ficheros	Clases	LÃneas	Condicionales
Cobertura	100% 4/4	100% 5/5	100% 5/5	54% 141/260	35% 29/82

Paquete (detalle)

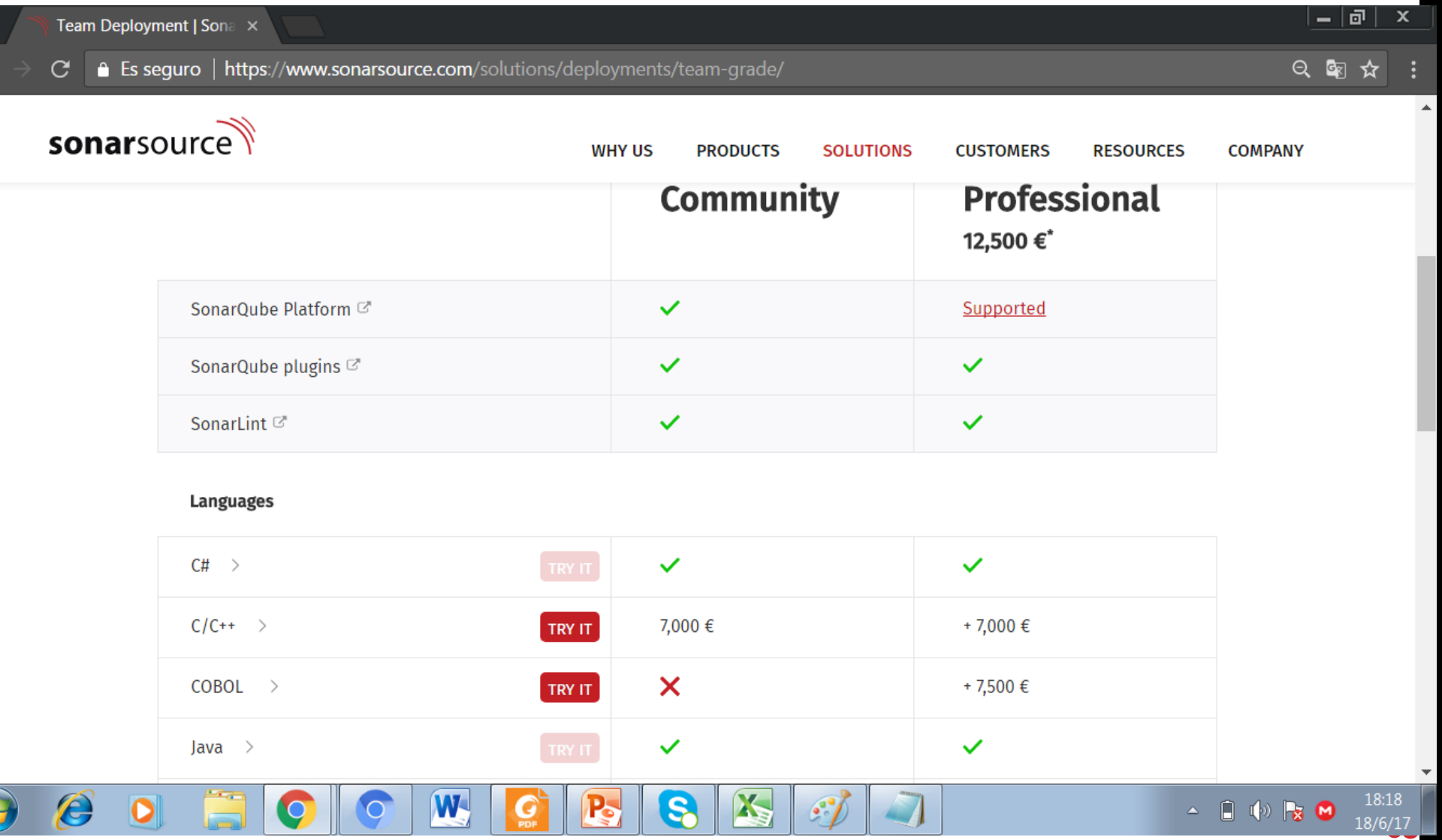
Nombre	Ficheros	Clases	LÃneas	Condicionales
build/test/mocks	100% 1/1	100% 1/1	44% 49/112	30% 12/40
build/test/runners	100% 1/1	100% 1/1	79% 22/28	50% 12/24
src	100% 2/2	100% 2/2	66% 21/32	50% 5/10
test	100% 1/1	100% 1/1	56% 49/88	0% 0/8

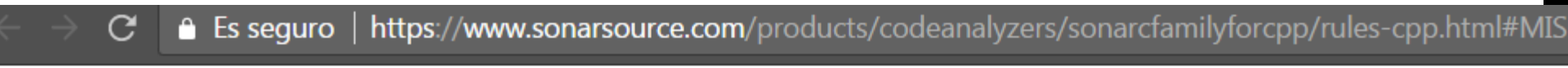
Página generada: 20-jul-2017 11:30:09 ART REST API Jenkins ver. 2.46.3


Jenkins
linsse.com.ar:223/jenkins/job/Bateria%20con%20Ceedling/56/ccccResult/
Jenkins
2
búsqueda
Ivan Sambrana | Desconectar
enkins › Bateria con Ceedling › #56
Volver al proyecto
Estatus
Cambios
Console Output
Editar información de la ejecución
Borrar Proyecto
Git Build Data
No Tags
Informe de Cobertura
CCCC Results
Cppcheck Results
Resultado de los tests
Ejecución previa
Cccc results
Explications
Project Summary
This table presents summary values of various measures over the body of source code submitted.
Metric Tag Overall PerModule
Number of modules MOM 1
Lines of Code LOC 101 101.0
McCabe's Cyclomatic Number MVG 12 12.0
Lines of Comment COM 30 30.0
LOC/COM L_C 3.367
LOC/COM M_C 0.400
Information Flow measure (inclusive) IF4 0 0.0
Information Flow measure (visible) IF4v 0 0.0
Information Flow measure (concrete) IF4c 0 0.0
Lines of Code rejected by parser REJ 38
Procedural Metrics Summary
This table presents values of procedural measures summed for each module identified in the code submitted.
Module Name LOC MVG COM L_C M_C
anonymous 63 12.0 24 2.625 0.500
Object Oriented Design
Table of four of the 6 metrics proposed by Chidamber and Kemerer in their various papers on 'a metrics suite for object oriented design'.
12:35
20/7/17


REDMINE
flexible project management
TestLink
git
Jenkins
sonarqube

SONAR: CUADRO DE MANDO PARA LA CALIDAD DEL CF





sonarsource

WHY US PRODUCTS SOLUTIONS CUSTOME

SonarCFamily C++ Rules

Products > Code Analyzers > Sonar CFamily for C/C++

Type

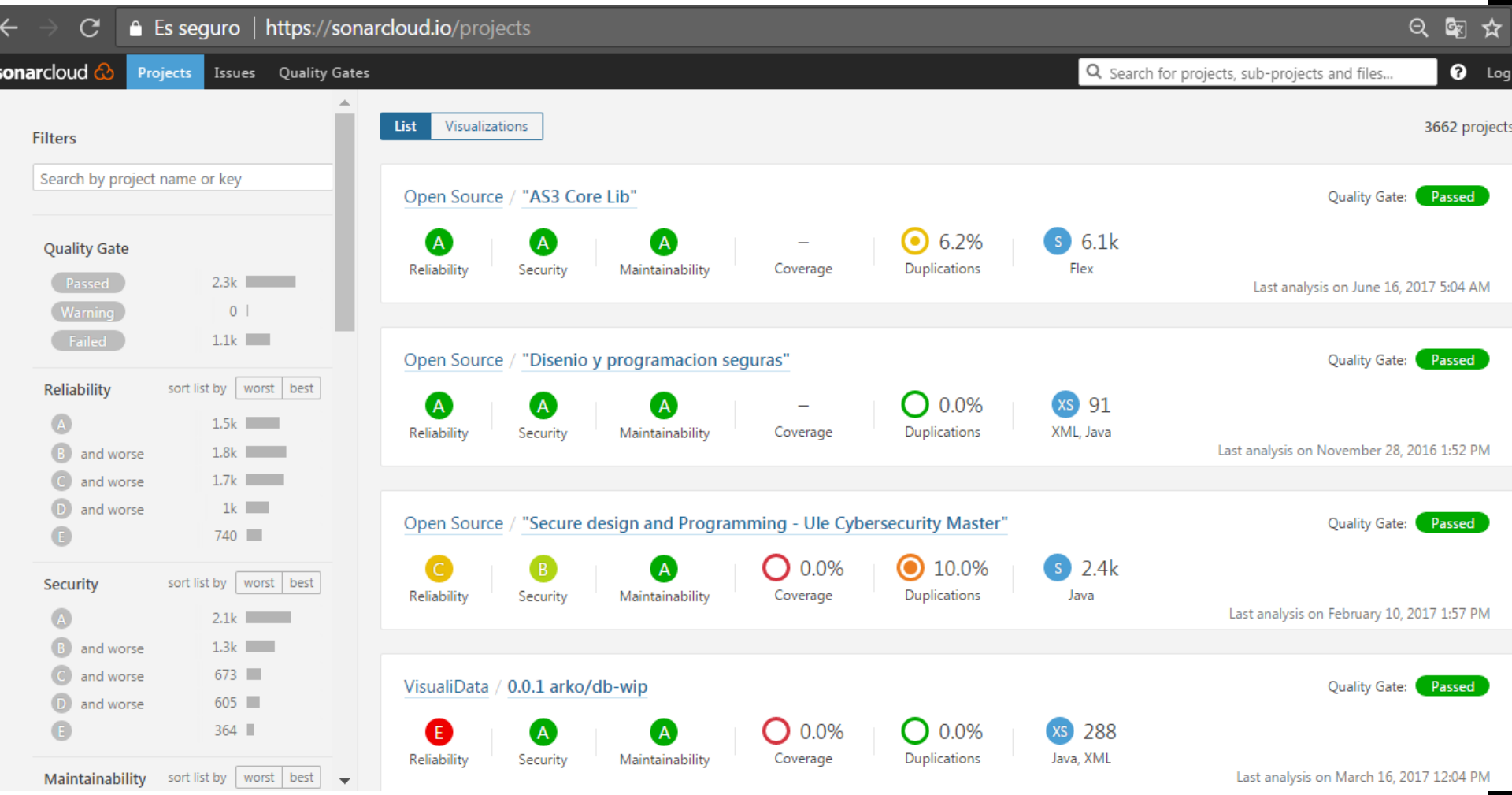
Bugs | Code Smells | Vulnerabilities

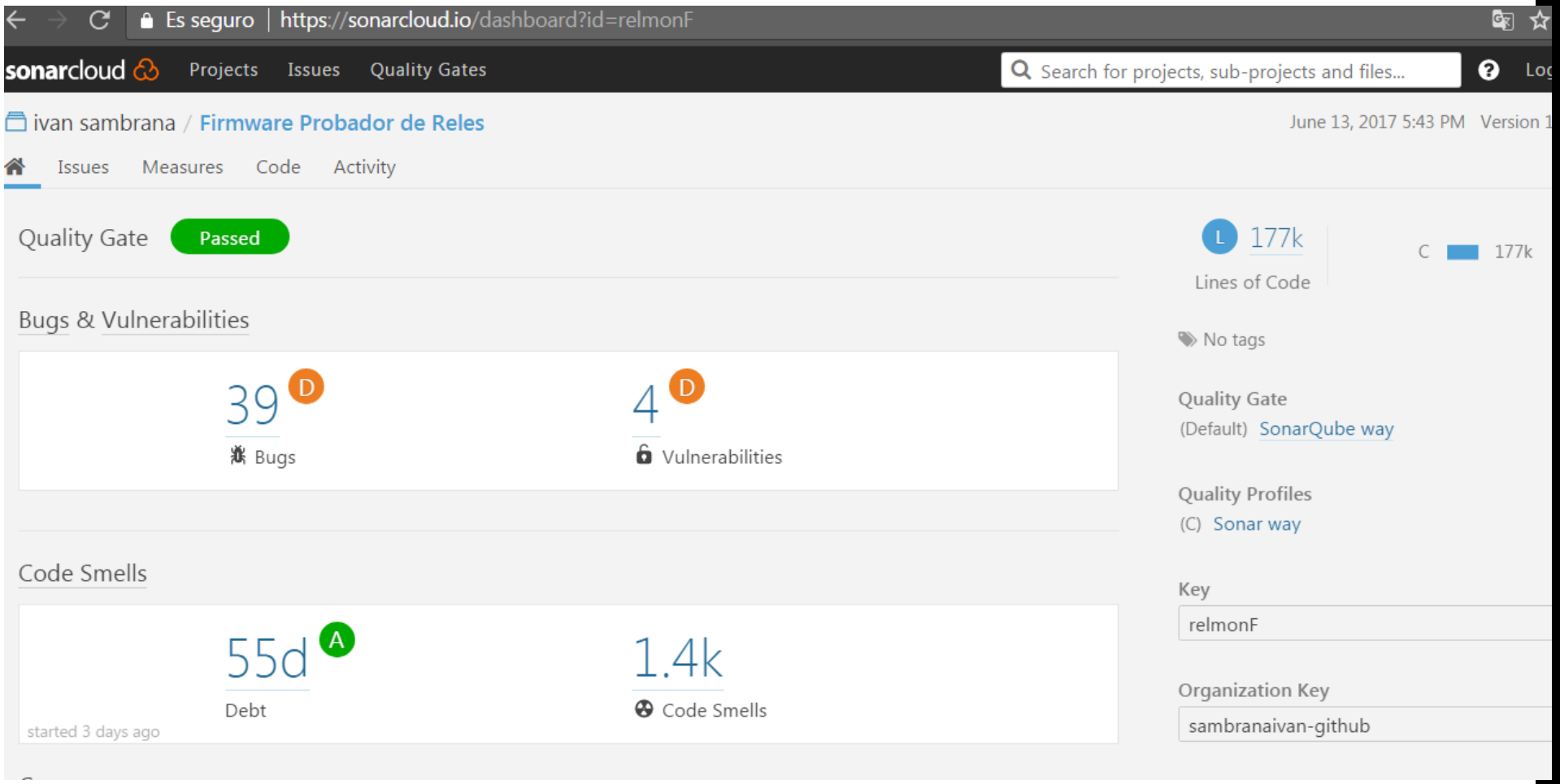
Standard

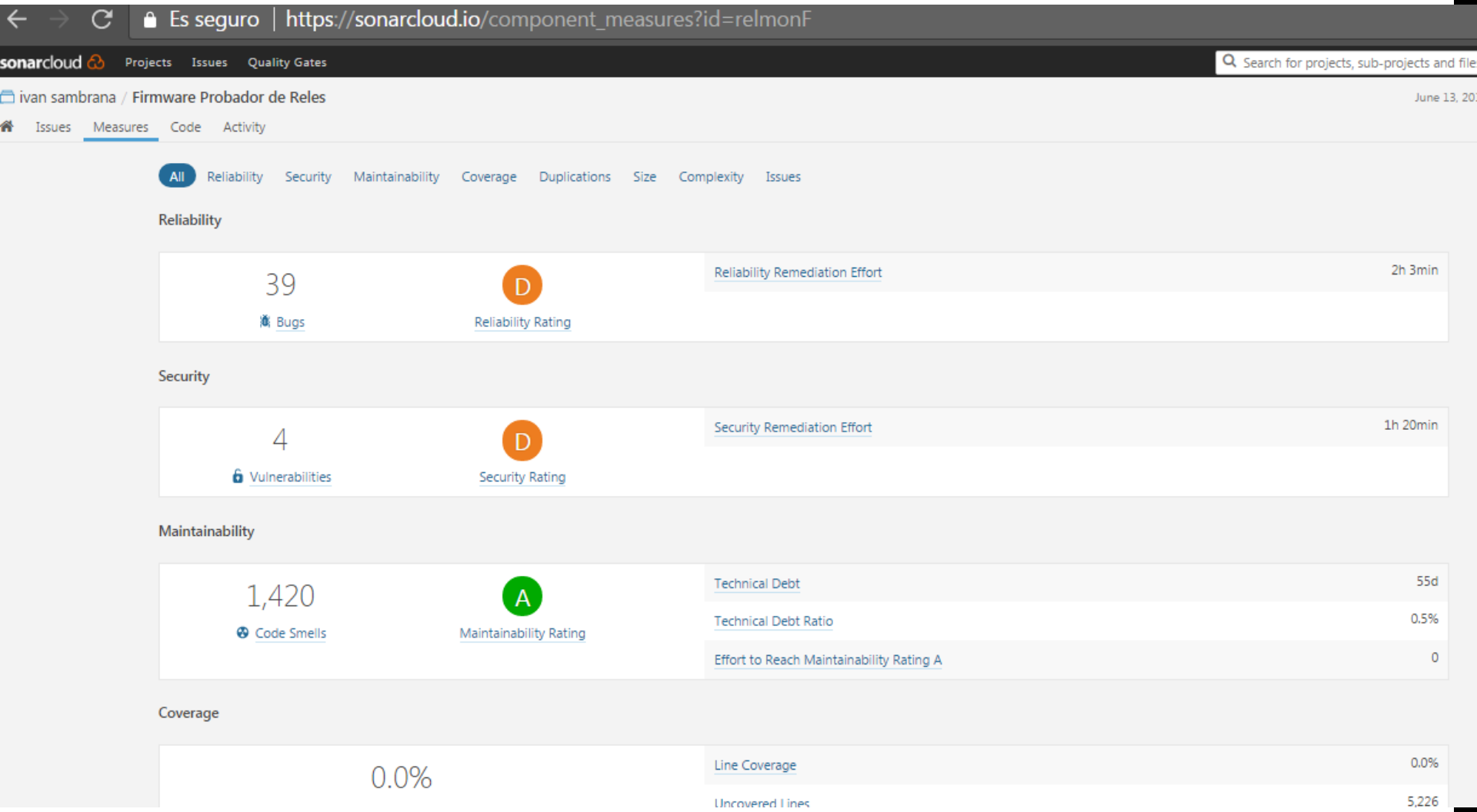
CWE | SANS_TOP_25 | **MISRA** | CERT

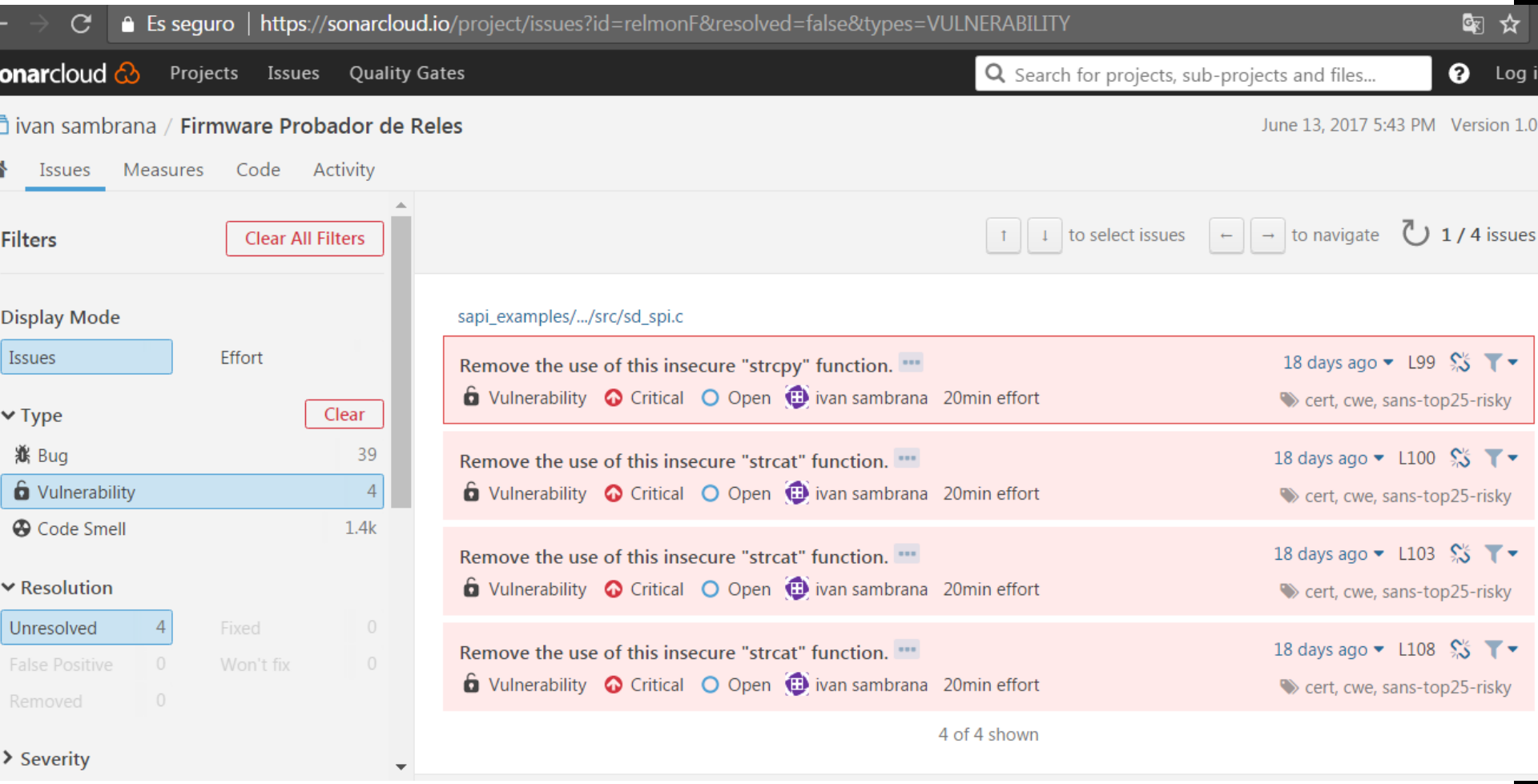
> "goto" should jump to labels declared later in the same function

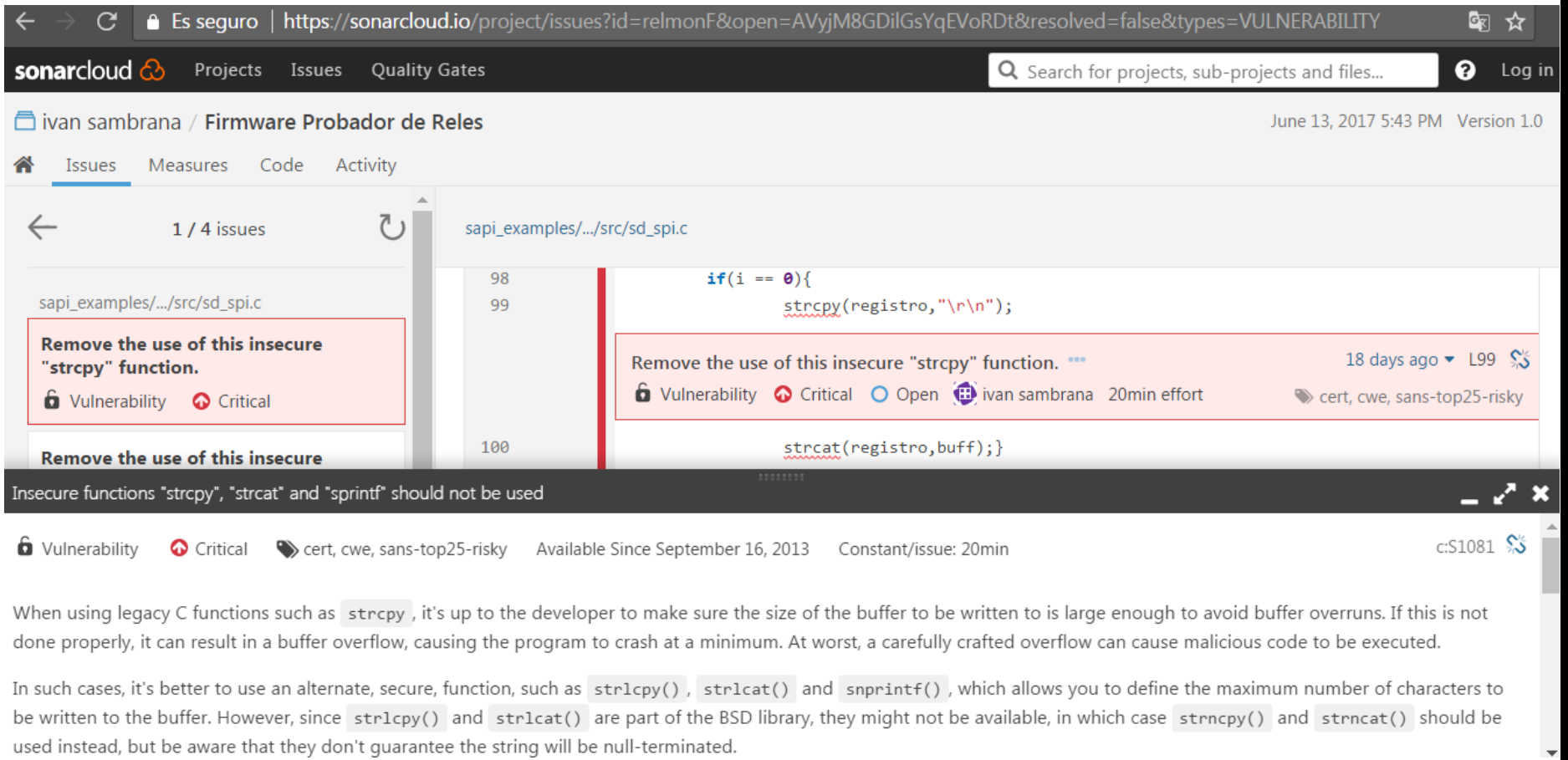
> "goto" statements should not be used to jump into blocks


Es seguro | https://sonarcloud.io/projects
sonarcloud
Projects
Issues
Quality Gates
Search for projects, sub-projects and files...
Log
Filters
Search by project name or key
Quality Gate
Passed 2.3k
Warning 0
Failed 1.1k
Reliability sort list by worst best
A 1.5k
B and worse 1.8k
C and worse 1.7k
D and worse 1k
E 740
Security sort list by worst best
A 2.1k
B and worse 1.3k
C and worse 673
D and worse 605
E 364
Maintainability sort list by worst best
List
Visualizations
3662 projects
Open Source / "AS3 Core Lib"
Quality Gate: Passed
A Reliability
A Security
A Maintainability
– Coverage
6.2% Duplications
S 6.1k Flex
Last analysis on June 16, 2017 5:04 AM
Open Source / "Disenio y programacion seguras"
Quality Gate: Passed
A Reliability
A Security
A Maintainability
– Coverage
0.0% Duplications
XS 91 XML, Java
Last analysis on November 28, 2016 1:52 PM
Open Source / "Secure design and Programming - Ule Cybersecurity Master"
Quality Gate: Passed
C Reliability
B Security
A Maintainability
0.0% Coverage
10.0% Duplications
S 2.4k Java
Last analysis on February 10, 2017 1:57 PM
VisualiData / 0.0.1 arko/db-wip
Quality Gate: Passed
E Reliability
A Security
A Maintainability
0.0% Coverage
0.0% Duplications
XS 288 Java, XML
Last analysis on March 16, 2017 12:04 PM


Es seguro | https://sonarcloud.io/dashboard?id=relmonF
sonarcloud
Projects
Issues
Quality Gates
Search for projects, sub-projects and files...
ivan sambrana / Firmware Probador de Reles
June 13, 2017 5:43 PM Version 1
Issues
Measures
Code
Activity
Quality Gate
Passed
Bugs & Vulnerabilities
39 D
Bugs
4 D
Vulnerabilities
Code Smells
55d A
Debt
1.4k
Code Smells
started 3 days ago
L 177k
Lines of Code
C 177k
No tags
Quality Gate
(Default) SonarQube way
Quality Profiles
(C) Sonar way
Key
relmonF
Organization Key
sambranaivan-github


Es seguro | https://sonarcloud.io/component_measures?id=relmonF
sonarcloud
Projects
Issues
Quality Gates
Search for projects, sub-projects and file
ivan sambrana / Firmware Probador de Reles
June 13, 20
Issues
Measures
Code
Activity
All
Reliability
Security
Maintainability
Coverage
Duplications
Size
Complexity
Issues
Reliability
39
Bugs
D
Reliability Rating
Reliability Remediation Effort
2h 3min
Security
4
Vulnerabilities
D
Security Rating
Security Remediation Effort
1h 20min
Maintainability
1,420
Code Smells
A
Maintainability Rating
Technical Debt
55d
Technical Debt Ratio
0.5%
Effort to Reach Maintainability Rating A
0
Coverage
0.0%
Line Coverage
0.0%
Uncovered Lines
5,226


Es seguro | https://sonarcloud.io/project/issues?id=relmonF&resolved=false&types=VULNERABILITY
onarcloud
Projects
Issues
Quality Gates
Search for projects, sub-projects and files...
Log i
ivan sambrana / Firmware Probador de Reles
June 13, 2017 5:43 PM Version 1.0
Issues
Measures
Code
Activity
Filters
Clear All Filters
Display Mode
Issues
Effort
Type
Clear
Bug 39
Vulnerability 4
Code Smell 1.4k
Resolution
Unresolved 4
Fixed 0
False Positive 0
Won't fix 0
Removed 0
Severity
to select issues
to navigate
1 / 4 issues
sapi_examples/.../src/sd_spi.c
Remove the use of this insecure "strcpy" function.
18 days ago L99
Vulnerability Critical Open ivan sambrana 20min effort
cert, cwe, sans-top25-risky
Remove the use of this insecure "strcat" function.
18 days ago L100
Vulnerability Critical Open ivan sambrana 20min effort
cert, cwe, sans-top25-risky
Remove the use of this insecure "strcat" function.
18 days ago L103
Vulnerability Critical Open ivan sambrana 20min effort
cert, cwe, sans-top25-risky
Remove the use of this insecure "strcat" function.
18 days ago L108
Vulnerability Critical Open ivan sambrana 20min effort
cert, cwe, sans-top25-risky
4 of 4 shown


Es seguro | https://sonarcloud.io/project/issues?id=relmonF&open=AVyjM8GDilGsYqEVoRDt&resolved=false&types=VULNERABILITY
sonarcloud
Projects
Issues
Quality Gates
Search for projects, sub-projects and files...
Log in
ivan sambrana / Firmware Probador de Reles
June 13, 2017 5:43 PM Version 1.0
Issues
Measures
Code
Activity
1 / 4 issues
sapi_examples/.../src/sd_spi.c
Remove the use of this insecure "strcpy" function.
Vulnerability
Critical
Remove the use of this insecure
sapi_examples/.../src/sd_spi.c
98 if(i == 0){
99 strcpy(registro,"\r\n");
Remove the use of this insecure "strcpy" function.
18 days ago
L99
Vulnerability
Critical
Open
ivan sambrana
20min effort
cert, cwe, sans-top25-risky
100 strcat(registro,buff);}
Insecure functions "strcpy", "strcat" and "sprintf" should not be used
Vulnerability
Critical
cert, cwe, sans-top25-risky
Available Since September 16, 2013
Constant/issue: 20min
c:S1081
When using legacy C functions such as strcpy, it's up to the developer to make sure the size of the buffer to be written to is large enough to avoid buffer overruns. If this is not done properly, it can result in a buffer overflow, causing the program to crash at a minimum. At worst, a carefully crafted overflow can cause malicious code to be executed.
In such cases, it's better to use an alternate, secure, function, such as strlcpy(), strlcat() and snprintf(), which allows you to define the maximum number of characters to be written to the buffer. However, since strlcpy() and strlcat() are part of the BSD library, they might not be available, in which case strncpy() and strncat() should be used instead, but be aware that they don't guarantee the string will be null-terminated.

MISRA C : 2012

MISRA is very pleased to announce today that the next edition of MISRA C Guidelines for the use of th language in critical systems, to be known as MISRA C:2012, is now available from the MISRA webstor PDF copies are available for immediate to purchase, with print copies available to pre-order with dispat from 8 April 2013.

MISRA C:2012 extends support to the C99 version of the C language (while maintaining guidelines for in addition to including a number of improvements that can reduce the cost and complexity of complia whilst aiding consistent, safe use of C in critical systems. Improvements, many of which have been m as a result of user feedback, include: better rationales for every guideline, identified decidability so use better interpret the output of checking tools, greater granularity of rules to allow more precise control, number of expanded examples and integration of MISRA AC AGC. A cross reference for ISO 26262 ha been produced.

The MISRA Bulletin Board includes Forums for discussing and asking questions about all the MISRA publications. We provide official answers from time to time on questions concerning interpretation of t documents. Now that MISRA C:2012 has been published we have added new discussion topics on the directives and rules in the new version, as well as an area for discussing migration issues.

Es seguro | https://www.misra.org.uk/Publications/tabid/57/Default.aspx

MISRA

MISRA Home | Activities | Publications | Buy online | News | Events | Discussions | Resources | Links | FAQ | Search

Quick links

Click below for more details on ...

MISRA C:2012 Addendum 2
MISRA C:2012 Amendment 1
MISRA Compliance
MISRA C:2004 Permits
MISRA AC INT
MISRA AC GMG
MISRA AC SLSF
MISRA AC TL
MISRA AC AGC
MISRA C++:2008
MISRA C ADC
MISRA C:2012 (MISRA C3)
MISRA C:2004 (MISRA C2)
MISRA C:1998 (MISRA C1)
MISRA SA
MISRA SRfP
MISRA Guidelines

MISRA publications

***MISRA C:2012 - Addendum 2: Coverage of MISRA C:2012 against ISO/IEC TS 17961:2013 "C Secure"*, ISBN 978-906400-15-6 (PDF), April 2016.**

While it is widely considered that MISRA C provides best practice guidelines for the development of safety-related systems, the publication of "C Secure" has generated discussion on the applicability of MISRA C for secure applications.

This document contains a mapping of MISRA C coverage of the "C Secure" requirements and shows that for freestanding applications, MISRA C already has excellent coverage of the "C Secure" requirements. Additional guidelines are provided in *MISRA C:2012 Amendment 1* to address gaps in coverage.

Please note, this document is a free download (click the name above or visit the MISRA Bulletin Board) and is not available from the web store.

***MISRA C:2012 - Amendment 1: Additional security guidelines for MISRA C:2012*, ISBN 978-906400-16-3 (PDF), April 2016.**

While it is widely considered that MISRA C provides best practice guidelines for the development of safety-related systems, the publication of "C Secure" has generated discussion on the applicability of MISRA C for

MISRA-C

1.4 Code design

Dir 4.14 The validity of values received from external sources shall be checked

C90 [Undefined 15, 19, 26, 30, 31, 32, 94]
C99 [Undefined 15, 16, 33, 40, 43-45, 48, 49, 113]

Category Required

Applies to C90, C99

Amplification

"External sources" include data:

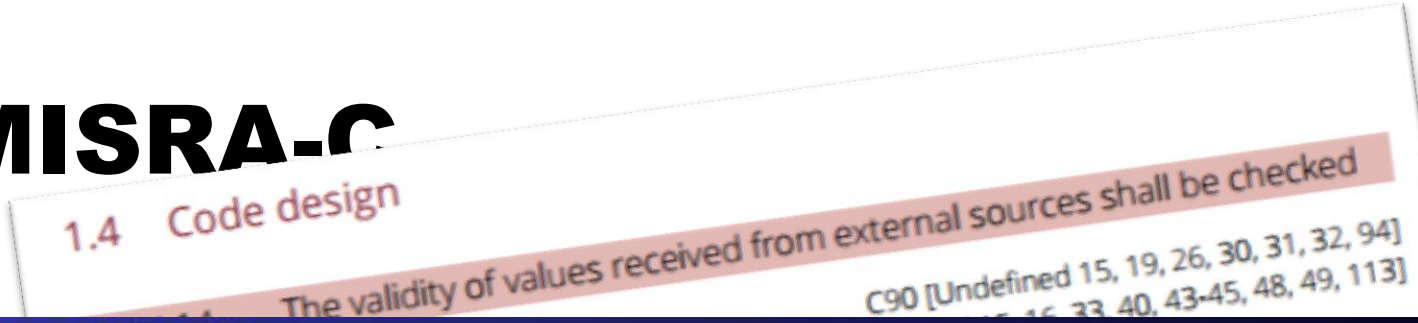
- Read from a file;
- Read from an environment variable;
- Resulting from user input;
- Received over a communications channel.

Rationale

A program has no control over the values given to data originating from external sources. The values may therefore be invalid, either as the result of errors or due to malicious modification by an external agent. Data from external sources shall therefore be validated before it is used.

In the security domain, external sources of data are usually regarded as untrusted as they may have modified by someone trying to harm or gain control of the program and/or system it is running validated before it can be used safely.

MISRA-C


1.4 Code design

The validity of values received from external sources shall be checked

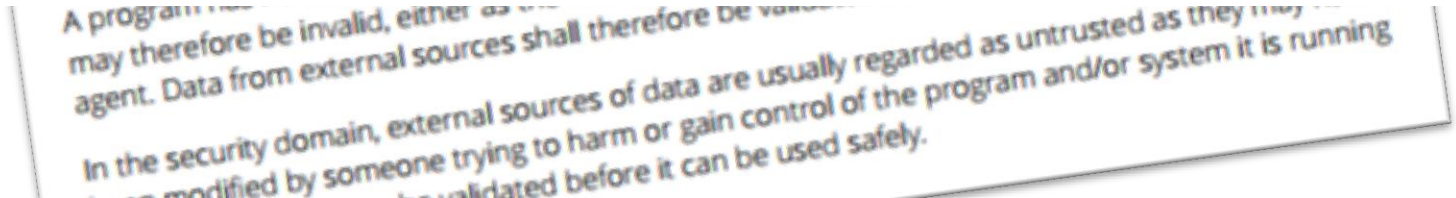
C90 [Undefined 15, 19, 26, 30, 31, 32, 94]

Loop blocks

Identifiers modified within the increment expression of a loop header shall not be modified inside the block controlled by that loop header.

```
int flag, si,array[10];
char *pc;

flag=1;
for (si=0; (si<5) && (flag==1); si++){
   flag=0; /* OK, even if it is a loop control variable */
   si=si+3; /* NOT OK, it is involved in the loop variables */
   }
```


may therefore be invalid, either

agent. Data from external sources shall therefore be

In the security domain, external sources of data are usually regarded as untrusted as they

modified by someone trying to harm or gain control of the program and/or system it is running

validated before it can be used safely.

REGLAS Y DIRECTIVAS

Rules

Rule 8.5	An external object or function shall be declared once in one and only one file

Rule 11.3	A cast shall not be performed between a pointer to object type and a different pointer to object type

Directives

Dir 3.1	All code shall be traceable to documented requirements

Dir 4.3	Assembly language shall be encapsulated and isolated

Parte 2

Dr. Ing. Emanuel Irrazabal (UNNE)

Cómo llevar a la realidad estas buenas prácticas?


DESPACITO

¿Preguntas?

Muchas gracias!