Facultat d'Informàtica de Barcelona

Proyecto Final de Carrera

Metodologías para el desarrollo de software seguro

Autor:
Ferran Lopez Provencio

Director:
Manuel García-Cervigón Gutiérrez

15 de enero de 2015

Índice

Índice de figuras

1. Introducción

1.1. Descripción del proyecto

Actualmente la UPC dispone de más de 200 redes de clase C, junto con alrededor de 50 unidades básicas (departamentos, facultades...), donde se llevan a cabo proyectos de diferentes envergaduras. Además sucede que la seguridad en muchos de los proyectos presenta debilidades en alguna de las diferentes partes de las que se compone el proyecto. Como puede ser en la infraestructura utilizada, el software desarrollado o la forma en que se realizan las comunicaciones entre diferentes grupos.

Esto provoca una necesidad de disponer de una metodología para gestionar la seguridad que se pueda aplicar en los diferentes proyectos que se llevan a cabo en la UPC. Además con el fin de facilitar el trabajo a los responsables de cada proyecto esta metodología ha de cubrir todas las fases del ciclo de vida que componen un proyecto.

Las fases que componen el ciclo de vida, tal y como se pueden identificar en CISA [4], de un proyecto son:

- **Análisis de requisitos y viabilidad**
  En esta fase del proyecto se establecen cuales son los objetivos del proyecto y se forma el equipo de trabajo que lo llevara a cabo.

- **Infraestructura y comunicaciones**
  En esta fase del proyecto se pone en marcha el conjunto de equipos informaticos que se utilizaran durante el desarrollo y puesta en marcha del proyecto. Esto incluye el equipo que utilizara cada miembro del equipo durante el desarrollo, los servidores y equipos de red que puedan ser necesarios para dar servicio a los miembros del equipo de trabajo y el equipamiento en que se ejecutaran las versiones en desarrollo del proyecto y las versiones a ser utilizadas por el cliente.

- **Metodologías de desarrollo**
  La metodología de desarrollo consiste en la forma en que el equipo de trabajo se organizará para repartir las diferentes tareas y establecer la prioridad con la que se irán completando de acuerdo a las necesidades del cliente.

  La tendencia actual en cuanto a metodologías de desarrollo es el empleo de metodologías ágiles como es el caso de scrum, kanban o extreme programming.

- **Implementación**
  En esta fase es en la que se lleva a cabo el trabajo necesario para cumplir los objetivos del proyecto, en el caso de proyectos de software suele consistir en el desarrollo de una o más aplicaciones.

- **Auditoria**
  En esta fase del proyecto se llega una vez el equipo de trabajo cree que ha cumplido con los objetivos establecidos. En este momento se comprueba que el trabajo realizado cumple de manera efectiva con todos los requisitos de comportamiento y calidad establecidos.

- **Despliegue**
  En esta fase del proyecto se pone a disposición del cliente el trabajo realizado, en algunos casos puede significar hacer entrega del trabajo llevado a cabo, mientras que en otros puede suponer desplegar el proyecto en un servidor donde el cliente se dispondrá a utilizar el proyecto para sus necesidades.

- **Monitorización**
  Esta fase del proyecto consiste en controlar que el funciona tal y como estaba previsto y detectar problemas que puedan surgir debido tanto a acciones externas como a problemas del proyecto que hayan pasado inadvertidos hasta el momento.

- **Mantenimiento**
  En esta fase del proyecto se llevan a cabo pequeñas modificaciones que puedan ser necesarias debido a fallos que se hallan detectado como a cambios en las condiciones en que el proyecto debe trabajar.

  En el caso de ser necesario realizar modificaciones de mayor envergadura seria necesario iniciar un nuevo proyecto con el objetivo de llevarlas a cabo.

El motivo por el cual se ha decido crear una nueva metodología en vez de aplicar alguna de las que existen hoy día se debe a que no se ha encontrado ninguna que cubra por completo todas las fases anteriormente descritas.

Los componentes tenidos en cuenta dentro de la metodología son:

- Plataforma
- Backend
- Aplicación en el servidor
- Aplicación del cliente

- Integridad de la comunicación cliente-servidor

Consideramos como plataforma el hardware en el que funciona el backend y la aplicación servidor, las conexiones de red necesarias para su funcionamiento, el sistema de ficheros y el sistema operativo utilizado por el equipo que esta ejecutando el producto.

El backend esta formado por los diferentes paquetes de software necesarios para ejecutar el producto. Dado el gran numero de paquetes de software existentes se limitara el análisis a los más utilizados dentro del InLab. Como ejemplos de paquetes de software tenemos: Apache, Tomcat, PHP, PostgreSQL, MySQL...

La aplicación del servidor es el software que se ejecuta en la plataforma, utiliza el backend y da servicio a la aplicación utilizada por el cliente.

La aplicación cliente es el software que proporciona funcionalidades de manera directa al usuario del producto, parte de las funcionalidades pueden requerir de servicios ofrecidos por una aplicación servidor.

Como integridad de la comunicación cliente-servidor entendemos que son los medios para detectar la alteración de los mensajes que se envían entre la aplicación cliente y servidor.

Finalmente se establecerán unas directrices para llevar a cabo una auditoria de seguridad de todos los elementos del producto.

1.2. Objetivo del proyecto

Como objetivo se desea obtener:

- La creación de un procedimiento fácil de usar con las indicaciones necesarias para prevenir los posibles ataques que puedan llevarse a cabo sobre los diferentes componentes de un producto y su posterior auditoria en conjunto.

- Crear una herramienta con el fin de facilitar la aplicación de la metodología creada a los responsables de proyecto.

- Probar el procedimiento creado en al menos uno de los proyectos que se estén llevando a cabo dentro de InLab. [1]

[1] Inlab anteriormente conocido como LCFIB da soporte a las aulas informáticas de la FIB y realiza desarrollo de software, tanto dentro de la UPC, como para empresas externas.
Para más informacion sobre InLab se puede visitar la pagina web http://inlab.fib.upc.edu/

2. Estado del arte

En primer lugar haremos un breve recorrido por diferentes metodologías que existen hoy en día en el campo de la seguridad informática. Las principales metodologías que se van a comentar son OSSTMM [1], OWASP [2], ISO 27001 [3] y CISA.

Se ha decido analizar estas metodologías debido a que son reconocidas internacionalmente y se aproximan en gran medida a los requisitos que hemos establecido para nuestro proyecto.

A continuación pasaremos a desglosar las diferentes fases en las que creemos que se debe dividir un proyecto a nivel de aplicación de controles de seguridad. Las fases en que dividimos un proyecto son:

- Análisis de requisitos y viabilidad
- Infraestructura y comunicaciones
- Metodología de desarrollo
- Implementación
- Auditoria
- Despliegue
- Monitorización
- Mantenimiento

2.1. Metodologías de seguridad existentes

Dado que ya existen metodologías de trabajo que tratan la seguridad del software, queremos analizarlas con el objetivo de establecer si alguna es capaz de cubrir nuestras necesidades.

2.1.1. OSSTMM

Open Source Security Testing Methodology Manual (OSSTMM) es un proyecto de software libre enfocado a la auditoria de seguridad de un sistema informático desarrollado por ISECOM [2]. La intención de OSSTMM es definir

[2]ISECOM (Institute for Security and Open Methodologies) es una organización sin animo de lucro con su sede principal en Barcelona.
Para más información sobre ISECOM, su pagina web es: http://www.isecom.org/home.html

el alcance de una auditoria de seguridad así como establecer una métrica que nos informe sobre los elementos que deben ser auditados y en que apartados de la auditoria.

- **Reglas de conducta**

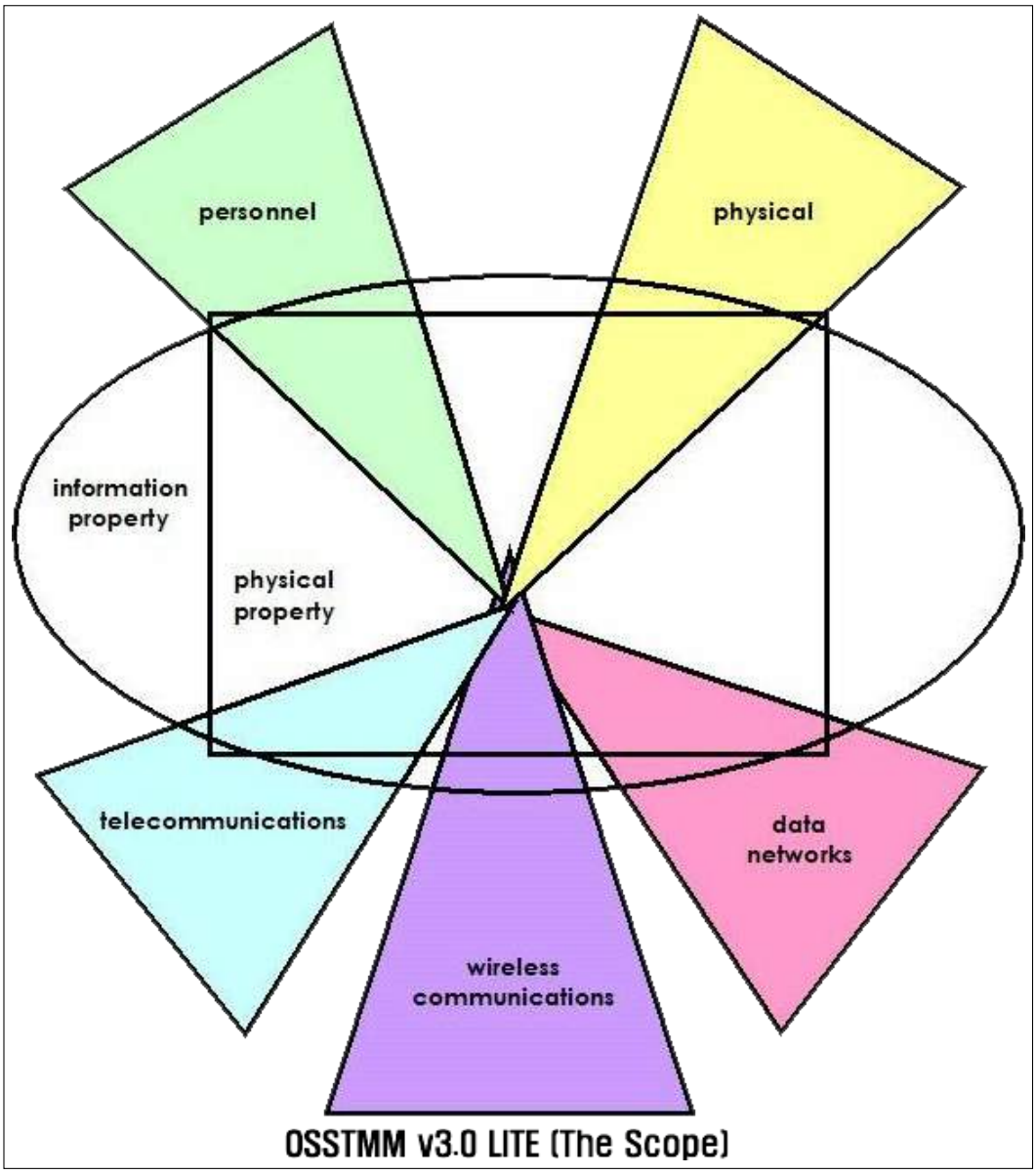

Figura 1: Dibujo en el que muestran los diferentes módulos de que se compone OSSTMM (OSSTMM 3 LITE)

El primer bloque de contenido de OSSTMM nos introduce en que es una auditoria de seguridad y sus diferentes tipos según el grado de conocimiento que tenga cada parte. Una parte a destacar de esta primera parte es un apartado dedicado a las reglas de conducta en que se esboza un código ético respecto a cuales son los limites respecto a lo que debemos hacer y hasta que punto, así como que y como debemos informar a nuestro contratante sobre los resultados de la auditoria.

- **Pensamiento critico**
  Otro punto en el que destaca OSSTMM es en alertarnos de los peligros que supone caer en la rutina al realizar tareas que van a ser muy parecidas una de otra y "acelerar" el proceso con saltos intuitivos que nos pueden llevar a cometer errores a la hora de obtener los datos a partir de los cuales vamos a obtener la información que utilizaremos para redactar el informe de la auditoria.

- **Métricas**
  La métrica utilizada en OSSTMM esta enfocada a calcular la superficie de ataque de un sistema, los puntos por los que puede ser atacado y los controles establecidos para impedir un ataque. Con estos datos nos ofrece una medida objetiva sobre que puntos del sistema analizado son vulnerables, están protegidos e incluso si están sobreprotegidos, es decir, se encuentra más de un elemento de control para ese punto que actúa sobre el aspecto que se controla. Esta redundancia provoca que sea necesario controlar a los elementos de control aumentando la complejidad del sistema sin aportar por ello una mayor seguridad, de hecho en muchos casos esta redundancia lleva a una menor seguridad.

- **Defensa en ancho**
  El ultimo concepto que se estudia en OSSTMM es la defensa en ancho. El motivo por el que plantean la defensa en ancho frente a la defensa en profundidad es que debido al elevado numero de interacciones y formas diferentes de interactuar que encontramos hoy en día en los sistemas nos encontramos con que el perímetro que hemos de defender en nuestro sistema acaba siendo todo el sistema en si.

Pros y contras encontrados en esta metodología:

- Pros:
  - Presenta diferentes módulos que se encargan de cubrir diferentes áreas de seguridad en una organización.

    - Ofrece una métrica sobre el nivel de seguridad de la organización, así como la forma de utilizarla.

- Contras:

    - Algunos de los controles de seguridad propuestos pueden resultar excesivos para la mayoría de organizaciones.
    - No se cubre la seguridad de un proyecto, solo a nivel de organización.

2.1.2. OWASP Testing Guide

OWASP es una organización sin animo de lucro que gestiona diferentes proyectos relacionados con la seguridad informática. Entre los proyectos que gestionan se encuentra la OWASP Testing Guide.

En la guia encontramos tres apartados diferenciados.

- **Que es el testing**
  En este apartado se hace una breve introducción de que es el testing de seguridad, cuales son las principales técnicas utilizadas y los requisitos que se deben cumplir para que el testing llevado a cabo sea significativo.

- **Framework OWASP**
  En esta sección se presenta la metodología de trabajo propuesta por OWASP al respecto de la seguridad de un software.

  En la metodología se hace una breve lista de diferentes consideraciones a tener en cuenta y tareas a añadir al propio desarrollo que facilitan la detección temprana de problemas de seguridad en el código.

  Como punto final de la metodología se recomienda llevar a cabo una auditoria del software para asegurar que no han quedado vulnerabilidades.

- **Métodos de ataque contra una aplicación web**
  Con diferencia esta es la parte más extensa de la guía. En esta sección se explican con cierto detalle diferentes técnicas utilizadas para llevar a cabo una auditoria de seguridad de una aplicación web.

Pros y contras encontrados en esta metodología:

- Pros:

    - Cubre la seguridad de un proyecto.

- Es la metodología que más se acerca a nuestras necesidades.

- Contras:

  - Los tipos de proyecto cubiertos por esta metodología solo son de aplicaciones web.
  - Algunas fases del ciclo de vida solo se comentan por encima.

2.1.3. ISO 27001

La norma ISO 27001 a diferencia de OSSTMM 2.1.1 y de OWSAP 2.1.2 no presenta casos concretos sobre los que trabajar en cambio proporciona un marco de trabajo con el que se facilita la gestión de la seguridad de la información de una organización.

- **Gestión de la seguridad**
  La norma en si se centra en proporcionar un modelo sobre el que basarse para establecer los procedimientos necesarios para gestionar la seguridad informática de una organización. Al establecer los objetivos que debe cumplir un SGSI sin marcar ningún procedimiento para conseguirlos nos permite escoger libremente el método con el cual la organización va a llevar a cabo los objetivos marcados. Este detalle hace que la ISO 27001 sea capaz de continuar siendo válida en el tiempo con pocas o ninguna modificación.

- **Puntos de control**
  Anexo al documento se encuentra una lista ordenada por categorías con los diferentes puntos que un sistema de seguridad debería cubrir. Algunos puntos interesantes que encontramos el la lista que ofrecen son:

  - Tratamiento de la seguridad en contratos con terceras personas.
    Tiene en cuenta que una organización puede necesitar que sus datos sean manejados por organizaciones o personas ajenas y que no por ello debemos despreocuparnos de la seguridad con que se lleva a cabo el procesado externo.
  - Compromiso de la gerencia con la seguridad de la información.
    Sin la colaboración de la gerencia no es viable lograr que se puedan aplicar las medidas necesarias para evitar que un ataque tenga éxito. Eso es debido a que es necesario comprar equipos dedicados, así como establecer protocolos a la hora de llevar a cabo ciertas tareas como puede ser la instalación o actualización de un software.

- Gestión de capacidad.
  Un sistema cuyo uso se acerca o esta por encima de su capacidad es especialmente vulnerable a cualquier fallo en la infraestructura facilitando la perdida de datos y una degradación en el servicio.
- Eliminación de medios.
  A la hora de desechar soportes como un disco duro hay que tener en cuenta que la información contenida en los mismos no desaparece al ser desconectados del equipo, y que por tanto la información contenida en ellos debe ser borrada antes de poder desecharlos.

Pros y contras encontrados en esta metodología:

- Pros:
  - Ofrece una buena guía para la gestión de toda la documentación referente a la seguridad.
- Contras:
  - La complejidad de su lectura se asemeja a la de textos legales.
  - No se cubre la seguridad de un proyecto, solo a nivel de organización.

2.1.4. CISA

CISA (Certified Information Systems Auditor) es una certificación para auditores de seguridad. Esta certificación presenta ciertas similitudes con OSSTMM en el alcance, cubriendo toda la seguridad de la información de una organización. Aun así logra diferenciarse gracias al hecho que es capaz de tener en cuenta la visión de negocio de una organización y los diferentes niveles de seguridad que pueden ser necesarios dependiendo de la información que debe ser salvaguardada.

A diferencia de la ISO 27001 que solo da una guía de actuación genérica en CISA encontramos que los controles ofrecidos son de mayor especificidad. Aun asi nos encontramos con la misma situación que hemos visto en OSSTMM y ISO 27001, solo se cubren aspectos referentes a la organización, como es la gobernanza o el trato con los empleados.

Pros y contras encontrados en esta metodología:

- Pros:
  - Cubre de forma clara y concisa la seguridad de la información dentro de la organización.

- Contras:
  - No se cubre la seguridad de un proyecto, solo a nivel de organización.

2.1.5. Otras guías

A continuación presentamos unas guías de carácter más especifico que han sido necesarias para cubrir los puntos débiles de las metodologías anteriormente descritas.

CIS Benchmarks

El CIS [8] nos ofrece una serie de guías [3] para la configuración de diferentes elementos que se encuentran en la infraestructura informática de una organización.

Cada una de las guías ofrece toda una serie de pasos para securizar el software o hardware al que se dedica ofreciendo incluso los comandos necesarios para llevar a cabo cada paso.

SAFECode

En SAFECode [5] [6] podemos encontrar guías dedicadas a diferentes temas enfocados al desarrollo de software, en nuestro caso han sido de particular ayuda las guías [4] dedicadas a las buenas practicas a la hora de desarrollar (Fundamental Practices for Secure Software Development) y la guía dedicada a las metodologías de desarrollo ágil (Guidance for Agile Practitioners).

MicrosoftSDL for Agile

Se trata de una guía [5] para incluir la seguridad en las metodologías ágiles ofrecida por Microsoft [7].

[3] Se pueden consultar todas las guías disponibles en la siguiente URL: https://benchmarks.cisecurity.org/downloads/browse/index.cfm?category=benchmarks

[4] Las guías ofrecidas por SAFECode pueden ser encontradas en la siguiente URL: http://www.safecode.org/publications/

[5] Se puede encontrar la guía en el siguiente enlace: http://www.microsoft.com/security/sdl/default.aspx

2.1.6. Comparativa

En la siguiente tabla 2.1.6 enumeramos los principales objetivos que queremos conseguir y en que grado los satisfacen las diferentes normas que hemos visto.

Concepto	**OSSTMM**	**OWASP**	**ISO 27001**	**CISA**
Metodología de trabajo	Cubre la auditoria del producto final	Cubre ligeramente todo el desarrollo del producto	Da indicaciones para la gestión documental	Da indicaciones para la gestión documental
Métrica de la seguridad del sistema	Calculo detallado	No proporciona	No proporciona	No proporciona
Lista de vulnerabilidades en el código	No proporciona	Proporciona para aplicaciones web	No proporciona	No proporciona
Hardening [6]	No	No	No	Alguna indicación genérica

[6]El hardening consiste en asegurar un servidor mediante la correcta configuración del Sistema Operativo y el software instalado

2.2. Análisis de requisitos y viabilidad

Consideramos como pertenecientes a la fase de análisis de requisitos y viabilidad todos los controles de seguridad que deben ser aplicados antes del inicio de un proyecto, o como preparación para el inicio de un proyecto.

En consecuencia la mayoría de controles relativos a esta fase del proyecto hacen referencia a la selección y preparación del personal que se encargara del proyecto. Del mismo modo es en este punto en el que se ha de definir con claridad el alcance de las medidas de seguridad que van a ser aplicadas y quienes serán los encargados que se apliquen dentro del proyecto.

2.2.1. Política de seguridad de la organización

El primer paso necesario para poder llevar a cabo un proyecto aplicando metodologías que tengan en cuenta la seguridad es la voluntad por parte de la organización en la cual se esta desarrollando el proyecto de aplicar las medidas necesarias.

Esta voluntad normalmente se materializa adoptando estándares como son: COBIT 5, ISO 27001 o ITIL.

El objetivo de estos estándares no es la seguridad de la información como tal, sino dar un marco de referencia al equipo de dirección sobre la forma en que se debe gestionar la seguridad a un nivel burocrático que facilite la aplicación de medidas que se encarguen de la seguridad de la información.

Junto con la adopción del estándar escogido se suele formar un comité de dirección encargado de velar por el cumplimiento de las medidas establecidas y la elección de un CISO (Chief Information Security Officer), en un inicio el puesto puede ser ocupado por el CIO (Chief Information Officer) o por el CEO (Chief Executive Officer). En organizaciones de mayor tamaño y que debido a su funcionamiento previo ya llevaran a cabo alguna gestión de riesgos podrían delegar la función de CISO al CRO (Chief Risk Officer).

2.2.2. Personal

Una organización que ha decidido cumplir con unos estándares de seguridad debe tomar algunas medidas en la forma en que se relaciona con el personal desde el proceso de selección hasta el momento en que el empleado abandona la organización.

Contratación de personal

Tal y como se recomienda en el modulo de HUMSEC de OSSTMM y el apartado 2.9 de CISA conviene llevar a cabo una comprobación de la

información que nos hace llegar un candidato a un puesto de empleo en la organización.

El primer paso es comprobar que la información que nos ha proporcionado en su curriculum es correcta, esto se puede verificar, en el caso de la formación reglada, consultando con la escuela que ha proporcionado la formación. Otra forma de comprobar que posee la formación que se otorga es mediante pruebas de aptitud diseñadas a tal efecto.

Una vez que ya se ha seleccionado a los candidatos que cumplen con los requisitos para ocupar el puesto de trabajo ofrecido es recomendable hacer una comprobación de antecedentes, ademas de solicitar opinión de las empresas para las que ha trabajado es buena idea comprobar si tiene antecedentes criminales y su estado de crédito. El objetivo de estas comprobaciones es evitar el espionaje industrial y asegurar en la medida de lo posible que carece de malas intenciones para con nosotros y de motivos económicos más allá del salario ofrecido.

Como parte del contrato y dependiendo del puesto de trabajo es posible añadir medidas para mejorar nuestra protección.

- **Acuerdo de confidencialidad**

  Un acuerdo de confidencialidad nos protege frente a la divulgación de la propiedad industrial que un empleado haya tenido acceso durante su pertenencia a la organización.

- **Acuerdo de no competencia**

  Un acuerdo de no competencia impide que un empleado que abandona la organización pueda acudir, o ser reclutado por la competencia de nuestra organización para emplear en contra nuestro los secretos comerciales a los que haya tenido acceso.

  Para que un acuerdo de no competencia tenga validez ha de estar limitado en tiempo, el periodo de tiempo por el cual no puede acudir a la competencia ha de estar acotado. Y también en el ámbito, no se permite establecer un ámbito global para el acuerdo.

Formación de bienvenida

Como parte del proceso de bienvenida a la organización el empleado recibe una formación en la cual se le explica los procedimientos de seguridad que debe cumplir. Algunos de los procedimientos son:

- Seguridad de las contraseñas
- Uso de medios de almacenamiento extraíbles (pendrives, tarjetas SD...)

- Instalación de software en los equipos de la organización.

Ampliación de la formación

Ampliar la formación de un empleado mientras se encuentra trabajando para la organización cumple dos funciones. La primera función es la de mejorar la moral del empleado, esto reduce la posibilidad que el empleado traicione, o actúe en contra de la organización. En segundo lugar, en caso de una baja inesperada de otro empleado hace posible que temporalmente pueda hacerse cargo de las responsabilidades del empleado que se encuentra de baja.

Salida de la organización

Al abandonar la organización se debe hacer una ultima entrevista con el empleado. El objetivo de esta entrevista es, que ahora ya sin haber la posibilidad de represalias por parte de un superior, nos haga saber su opinión sobre los superiores bajo los cuales ha estado trabando, y también, de sus compañeros.

Es en esta entrevista cuando es mayor la posibilidad de llegar a ser informados de alguna mala practica llevada a cabo por otro de los trabajadores con los que ha estado trabajando.

2.3. Infraestructura y comunicaciones

Una red correctamente configurada y securizada permite ofrecer un servicio más estable a los usuarios, a la vez que reduce el coste derivado de la perdida de información almacenada. Esta perdida de información por lo general provoca desconfianza en el servicio proporcionado a los usuarios, que en el caso de una empresa supone desde una reducción en los beneficios hasta la quiebra de la misma.

Para lograr securizar de manera efectiva una red debemos tener en cuenta la disposición de los diferentes elementos que la forman así como su configuración. También se debe disponer métodos para impedir el acceder a los servicios cuyo acceso no este permitido a un usuario determinado.

2.3.1. Elementos de protección de una red

En una red podemos encontrar diferentes elementos de protección. A continuación procederemos a exponer cuales son estos elementos, que función llevan a cabo en la protección de una red y en que parte de la red es recomendable ubicar estos elementos.

- **Firewall**
  Posiblemente este sea el elemento de protección más conocido, se puede encontrar en dos variantes. Su principal función es filtrar las conexiones que pasan por el i decidir si debe permitir que continúen o bien bloquearlas a partir de un conjunto de reglas.

  A continuación se detallan los diferentes tipos básicos de firewall que se pueden encontrar.

  - **Software**
    Esta variante de firewall es común encontrarla en los sistemas operativos empleados en los terminales de los usuarios de la red y en equipos domésticos. El motivo por el cual es tan fácil de encontrar es su reducido coste a la hora de instalarlo, y que en muchos casos viene pre-configurado de serie o con asistentes gráficos para facilitar su configuración.

    El lugar donde se ubica este tipo de firewall es en los equipos de los usuarios de la red y en las maquinas que ofrecen servicio. Esto se hace ya que permite definir unas reglas de acceso a la red para cada equipo que son controladas por el administrador de la red.

    Si bien estos firewall nos permiten ejercer un control sobre las comunicaciones de red para cada cliente, no hay que olvidar que

no dejan de ser una aplicación más ejecutándose sobre un sistema operativo. Motivo por el cual son vulnerables a una manipulación externa por parte de un atacante.

- **Hardware**
  A diferencia de los firewall de software, los firewall de hardware se encuentran en equipos específicamente dedicados a la tarea de filtrar las conexiones de red que pasan por ellos. Al funcionar en un hardware dedicado de manera exclusiva para el firewall hace que estos tengan un mayor coste, y por tanto no sean tan comunes.

  Normalmente se ubican en un punto de la red desde el que puedan filtrar el trafico de múltiples equipos que compartan una misma subred.

  Actualmente este tipo de firewall ha ido añadiendo la capacidad de realizar funciones de network adress translation y proxy para aprovechar el incremento en la capacidad del hardware y reducir el numero de elementos por los que es necesario pasar antes de llegar al destino.

- **Sistema de detección de intrusiones**
  Un sistema de detección de intrusiones o IDS analiza el tráfico que pasa por el en busca de comportamientos que puedan asociarse con una intrusión. El método empleado para detectar un posible ataque es mediante la búsqueda de patrones de ataques conocidos y comportamientos sospechosos.

  El comportamiento del IDS al detectar un ataque puede variar, en unos casos puede limitarse a recoger registros de la actividad sospecha y dar aviso, mientras que en otros casos puede cortar directamente la conexión sospechosa impidiendo que esta pueda llevar a cabo cualquier acción.

  Un IDS se ubicará dentro de la red en una posición que le permita escuchar a todas las conexiones de la red para facilitar así la detección de comportamientos sospechosos. En el caso de querer que bloquee cualquier conexión sospechosa sera necesario que trabaje junto al firewall.

- **Inspección profunda de paquetes**
  La inspección profunda de paquetes o DPI consiste en analizar el contenido de los datos que circulan por la red con el fin de detectar comportamientos o contenidos sospechosos. A diferencia de un IDS un DPI toma e consideración el contenido de los paquetes de datos que circulan por la red.

Dependiendo de las capacidades del sistema DPI instalado puede variar el nivel de profundidad en la capa de protocolos de la red que puede analizar. En los modelos más básicos se limita al contenido de los paquetes de la capa de red (nivel 3 en el modelo OSI), mientras que en los más completos llega a analizar toda la capa de protocolos pudiendo ver directamente que datos esta enviando y recibiendo cada equipo de la red.

Debido a la capacidad de los sistemas IDS para analizar el contenido que circula por la red también es posible utilizarlos con otros fines como puede ser recoger estadísticas del uso de la red, monitorizar el comportamiento de los usuarios de la red, llevar a cabo escuchas de un usuario en concreto e incluso censurar contenidos.

2.3.2. Configuración de la red

A parte de los elementos de protección físicos una red debe ser configurada de una manera adecuada para poder ser considera segura.

La configuración que puede aplicarse a cualquier tipo de red incluye:

- La creación y gestión de registros de actividad en la red junto con la identificación de quien lleva a cabo dicha actividad.

  Tal y como expresa la norma ISO 17799 en el apartado 10.10

  > "Se debieran monitorear los sistemas y se debieran reportar los eventos de seguridad de la información. Se debieran utilizar bitácoras de operador y se debieran registrar las fallas para asegurar que se identifiquen los problemas en los sistemas de información.
  >
  > Una organización debiera cumplir con todos los requerimientos legales relevantes aplicables a sus actividades de monitoreo y registro.
  >
  > Se debiera utilizar el monitoreo del sistema para chequear la efectividad de los controles adoptados y para verificar la conformidad con un modelo de política de acceso."

  El motivo por el que se demanda una monitorizacion y registro de actividad de la red es que nos permite detectar intrusiones y establecer el modo en como estas se han llevado a cabo.

  Al poder seguir el proceso que se ha seguido para llevar a cabo la intrusión nos permite dar los pasos necesarios con el fin de evitar que

se produzca otra intrusión siguiendo el mismo procedimiento, haciendo de este modo que la red sea más segura.

Además de permitirnos bloquear futuras intrusiones del mismo tipo los registros nos dan una lista de los ficheros que han sido accedidos o modificados. Esto nos permite reparar los daños que hayan podido ser causados, o notificar a los usuarios pertinentes de la información que ha sido accedida sobre ellos.

- La asignación de permisos a cada usuario para poder utilizar los diferentes servicio que estén disponibles en la red.

  Tal y como expresa la norma ISO 17799 en el apartado 11.1

  > "Se debiera controlar el acceso a la información, medios de procesamiento de la información y procesos comerciales sobre la base de los requerimientos comerciales y de seguridad.
  >
  > Las reglas de control del acceso debieran tomar en cuenta las políticas para la divulgación y autorización de la información."

  A no ser que queramos que en nuestra red cualquier usuario tenga pleno acceso a todos los recursos que en ella podamos encontrar querremos establecer medidas para limitar el acceso a ciertos recursos dependiendo de los permisos que se hayan otorgado a cada usuario.

2.3.3. Arquitectura de la red

Dentro de una red podemos identificar tres tipos de servicios basándonos en su accesibilidad desde el exterior de la red.

- Servicios accesibles desde el exterior.

  Los servicios disponibles desde el exterior son aquellos que pueden ser utilizados por cualquier terminal conectado a internet. Este facilidad para poder acceder a ellos hace que sea necesaria una configuración muy cuidadosa para evitar accesos no deseados.

  Como ejemplo de servicios que habitualmente son visibles desde el exterior encontramos servidores web, servidores de correo electrónico y webservices.

- Servicios no accesibles desde el exterior, pero necesarios para los servicios disponibles para el exterior.

Los servicios incluidos en esta categoría no son accesibles desde el exterior, pero puede suceder que servicios accesibles desde el exterior requieran de ellos para su correcto funcionamiento. Esto significa que si bien no son directamente vulnerables aun son un objetivo viable para un ataque a través de los servicios externos.

Un ejemplo de este tipo de servicio es una base de datos o un repositorio de ficheros.

- Servicios no accesibles desde el exterior.

  Estos servicios en ningún caso deberían ser accesibles desde el exterior de la red, ni ser utilizados como apoyo para dar servicio a los servicios externos. Si bien esto los hace mucho menos vulnerables a un ataque externo tampoco elimina el riesgo de un ataque.

  Como ejemplo de este tipo de servicios tenemos el sistema de información empleado para la gestión de una empresa.

Basándonos en esta clasificación de los distintos servicios, que representa el diferente grado de exposición al exterior de los diferentes servicios habitualmente encontrados en una red, se recomienda la arquitectura de red en la figura 2.

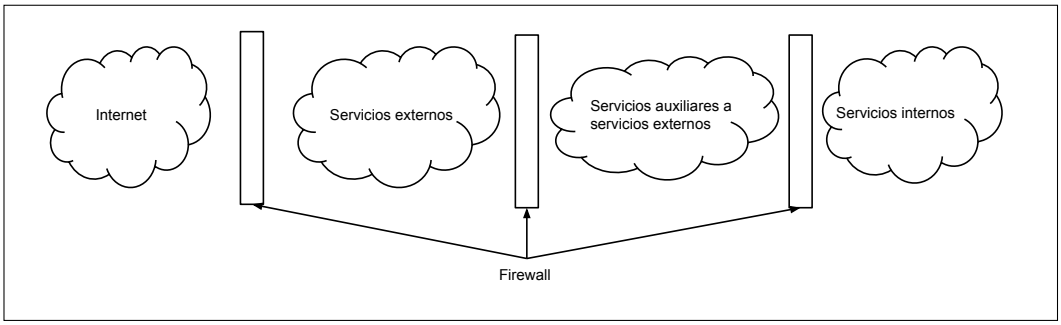

Figura 2: Arquitectura de la red

Esta arquitectura de red nos permite establecer una defensa en profundidad. El motivo para recomendar este tipo de defensa en la red es porque nos permite tener un mayor numero de elementos de protección a medida que deseamos reducir la posibilidad de un ataque hacia un servicio.

2.3.4. Superficie de ataque

En el contexto de seguridad informática la superficie de ataque de un sistema hace referencia a la cantidad de servicios que son visibles desde la red. Dado que cada servicio puede ser vulnerable a un ataque cuantos más sean accesibles más fácil puede ser para un atacante aprovechar una o varias vulnerabilidades para llevar a cabo con éxito un ataque sobre el equipo. Por ese motivo siempre es recomendable instalar el mínimo software necesario para lograr la funcionalidad que buscamos.

Por otra parte debemos asegurarnos que el software instalado recibe las ultimas actualizaciones de seguridad que eliminan las vulnerabilidades que se van descubriendo en el software. No hay que confundir estas actualizaciones con los cambios de versión que usualmente incorporan nuevas funcionalidades al software.

Existe una serie de servicios que por su misma naturaleza insegura se desaconseja su instalación en cualquier sistema. Los servicios a los que nos referimos son: telnet, ft, rlogin, rsh/rcp, vnc, tftp...

En su lugar podemos emplear otros servicios que proporcionan la misma funcionalidad, pero de manera más segura. Como es ssh en el caso de telnet, o de sftp para ftp.

Realizando esto logramos cumplir con una parte del punto A.10.6 de la ISO 27001. Para optimizarlo además hemos de configurar adecuadamente los servicios que determinemos que han de estar instalados en el sistema.

2.3.5. Identificación de los usuarios

La identificación de los usuarios que acceden a una maquina nos permite asociar los eventos que se produzcan en una maquina con su causante, este puede ser tanto una persona como un software.

Además del registro que generamos también permite limitar de diferente manera que acciones son permitidas a cada usuario. Una diferenciación bastante habitual la encontramos entre un usuario normal de un sistema y el administrados de sistemas del sistema. Mientras que el usuario por lo general no podrá instalar nuevo software en el sistema ni cambiar su configuración, el administrador de sistemas deberá poder realizar estas acciones pues forma parte de su responsabilidad.

Dependiendo del grado de seguridad que deseemos existen diferentes maneras de identificar a un usuario.

- Usuario y contraseña. Esta es la más común y salvo para sistemas que requieran de una seguridad especial con una política de contraseñas adecuada es suficiente para la mayoría de sistemas.

- Identificación mediante un token físico. En este caso una vez introducido el usuario y contraseña se solicita un código de autorización que es generado en ese momento por un elemento físico en posesión del usuario.

- Identificación con una smart card. En este caso la identificación del usuario la proporciona una smart card en posesión del usuario. Esta smart card contiene una certificado digital mediante el cual se identifica a un usuario.

- Identificación biométrica. Este tipo de identificación esta reservada por sus características para personas. Consiste en el empleo de patrones identificativos únicos del cuerpo, como es el caso de las huellas dactilares para confirmar la identidad del usuario.

  Este tipo de identificación debe utilizarse con precaución debido a que puede llegar a poner en peligro la integridad física de un usuario en caso que un asaltante desee obtener acceso a nuestro sistema.

	Precisión	**Coste**	**Equipo necesario**	**Aceptación**
Contraseña	Baja	Ninguno	Ninguno	Alta
Token	Media	Un token por usuario	Ninguno	Media
Smart card	Alta	Una smart card por usuario más el lector de smart cards	Lector de smart card	Baja
Biométrico	Alta	Lector biométrico	Lector biométrico	Alta

Cuadro 1: Comparativa entre los diferentes métodos de identificación presentados

2.3.6. Permisos de ejecución

Una vez que tenemos identificado al usuario dentro del sistema es posible delimitar su capacidad de acción en función del trabajo que deba llevar a cabo dentro del sistema.

A la hora de asignar los permisos debemos seguir la misma política que hemos seguido con los servicios visibles desde la red. Los mínimos necesarios para que el usuario pueda llevar a cabo su trabajo.

Dado que es habitual encontrar usuarios que requieren de permisos similares y para facilitar la tarea de monitorización a los administradores del sistema es habitual en la mayoría de sistemas crear grupos de permisos y asignar los usuarios a diferentes grupos dependiendo de las tareas que tenga cada uno.

Cumpliendo este apartado y el anterior satisfacemos el apartado A.10.1.3 de la ISO 27001 que hace referencia a la segregación de deberes.

2.3.7. Seguridad física de los equipos

Los servidores que dan servicio a una organización se suelen ubicar en salas destinadas específicamente a ello debido a la necesidad de controlar el acceso físico a los servidores y a la necesidad de mantenerlos en un ambiente optimo para ellos, pero que no resulta confortable para las personas.

- **Control de acceso**
  Dependiendo el nivel de seguridad que deseemos para nuestros servidores el control de acceso puede variar. En los casos que no se disponga de ninguna seguridad nos encontraremos con una puerta sin llave y sin cámaras de seguridad ni vigilantes que controlen el acceso. Mientras que en los casos de mayor seguridad sera necesario identificarse ante el personal de seguridad que registrara en busca de elementos no permitidos en el interior de la sala y habrá cámaras de seguridad registrando nuestros movimientos.

- **Control ambiental**
  El control ambiental esta formado por todos los sensores que nos permiten monitorizar el estado de la sala donde se encuentran los servidores. Los parámetros a controlar son la temperatura y humedad de sala, también se incluyen detectores de incendio, inundación y terremoto en caso que proceda.

2.4. Metodologías para el desarrollo de software

A la hora de desarrollar un software existen diferentes metodologías que un equipo puede utilizar. A continuación exponemos las más conocidas o utilizadas.

2.4.1. Desarrollo en cascada

El desarrollo en cascada es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase.

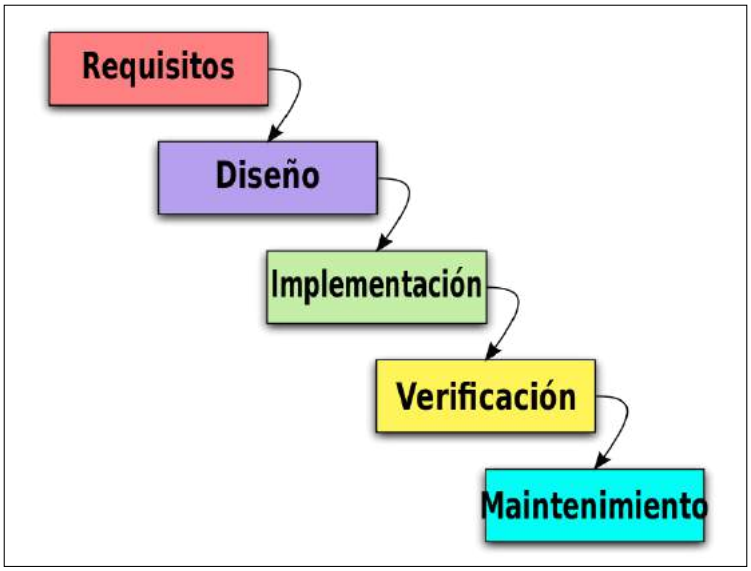

Figura 3: En el desarrollo en cascada se finaliza cada etapa del desarrollo antes de pasar a la siguiente etapa. (Wikipedia)

En el ámbito de la seguridad informática el desarrollo en cascada obliga a realizar un análisis de riesgos antes de iniciar la fase de diseño, donde podrían empezar a intuirse cuales pueden ser los puntos vulnerables del proyecto. Y no es hasta la fase de implementación que se establecen de manera clara, sin embargo siguiendo con las etapas establecidas, hasta la fase de verificación no se comprueba la seguridad. En este punto el software ya se ha dado por finalizado y los problemas de seguridad que se encuentren se solucionaran con parches sobre el software ya finalizado.

2.4.2. RUP

El Proceso Racional Unificado (Rational Unified Process en inglés) es un proceso de desarrollo de software desarrollado por la empresa Rational Software, actualmente propiedad de IBM.

A diferencia del desarrollo en cascada RUP es iterativo, es decir el proyecto se divide en pequeñas etapas que se deben completar cada pocas semanas. La ventaja que aporta un modelo de desarrollo iterativo frente al modelo clásico de cascada es la posibilidad de incorporar modificaciones al proyecto durante su desarrollo.

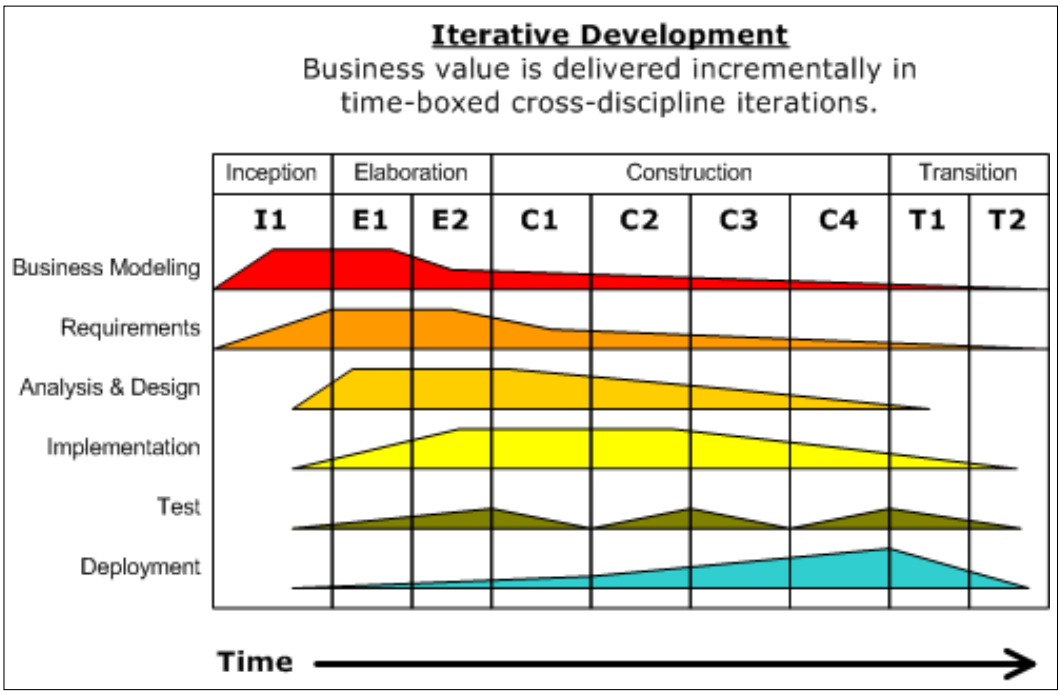

Figura 4: Progreso de las iteraciones y principal trabajo que se lleva a cabo en ellas (Wikipedia)

Una de las ventajas que se encuentra en RUP es la posibilidad de adaptar la metodología con alguno de los diferentes módulos de que dispone. Y si bien esto le confiere una gran versatilidad también supone un mayor tiempo de espera para adaptarse a los nuevos enfoques que puedan darse en el campo de las metodologías de desarrollo.

La principal desventaja que se encuentra en RUP es la dependencia de UML (Unified Modeling Language) y de herramientas propietarias de IBM. La fuerte dependencia de UML implica que todos los proyectos realizados con

RUP deberán utilizar lenguajes de programación orientados a objetos, con especial enfasis en el uso de Java. Añadido a esto se suma que las herramientas disponibles son en su mayoria de pago y propiedad de IBM.

2.4.3. Metodologías ágiles

El desarrollo ágil de software refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan mediante la colaboración de grupos auto organizados y multidisciplinarios. Existen muchos métodos de desarrollo ágil; la mayoría minimiza riesgos desarrollando software en lapsos cortos. El software desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar de una a cuatro semanas. Cada iteración del ciclo de vida incluye: planificación, análisis de requisitos, diseño, codificación, revisión y documentación. Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del producto al mercado, sino que la meta es tener una «demo» (sin errores) al final de cada iteración. Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto.

En las metodologías ágiles es muy fácil dejar de lado la seguridad debido a que no es una tarea propiamente, sino que es transversal a todas las tareas. Además debido a que la seguridad no se suele considerar importante por un cliente, hasta que falla, y que es el cliente quien asigna la prioridad de las tareas.

Por este motivo si bien las metodologías ágiles permiten integrar con facilidad la seguridad en el desarrollo, es necesario que el equipo de trabajo haga consciente al cliente de los problemas que acarrea dejar de lado la seguridad así como realizar el esfuerzo de mantener la seguridad en un entorno al que se le van incorporando elementos nuevos de manera continua.

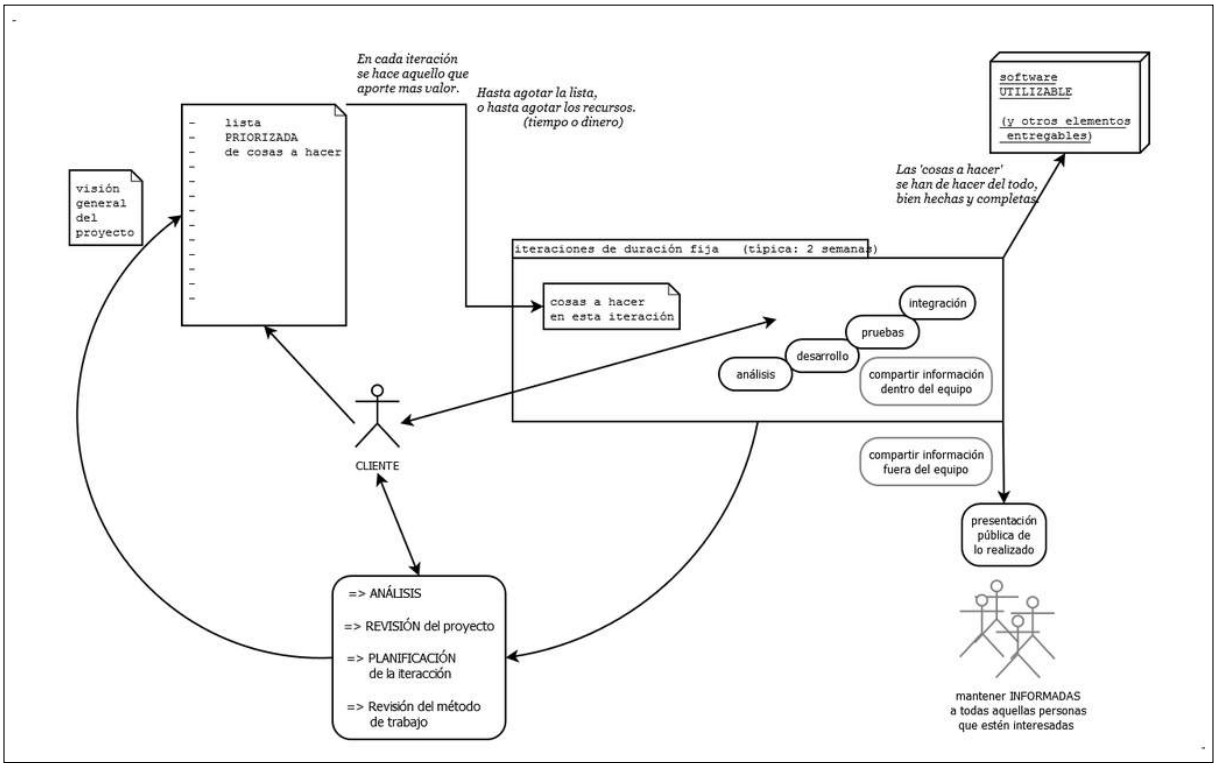

Figura 5: En las metodologías ágiles el cliente tiene la posibilidad de priorizar las diferentes tareas de las que se compone el proyecto en función de como de importantes las considera y del coste de recursos que asigna a cada tarea el equipo de desarrollo. (Wikipedia)

2.5. Implementacion

Consideramos como pertenecientes a la fase de implementacion aquellos controles que ayuden a mejorar el nivel de seguridad del software desarrollado en el proyecto. En consecuencia los controles relativos a esta fase del proyecto deben adaptarse a los diferentes lenguajes de programación utilizados en el proyecto.

A grandes rasgos los puntos que más se deben vigilar independientemente del lenguaje de programación utilizado son los que se describen a continuación.

2.5.1. Buffer Overflow

Un *buffer overflow* se produce cuando un programa escribe datos en memoria más allá de la memoria que tenia reservada, como consecuencia de esto

se puede sobrescribir el código que se esta ejecutando en la máquina.

Dada la criticidad de este tipo de fallo se ha de evitar en la medida que sea posible el uso de funciones o métodos que no controlen este tipo de fallos.

2.5.2. Validación de datos

Comprobar la validez de los datos que son entregados a una aplicación evita que esta pueda llegar a un estado inesperado y actuar de forma imprevista. Si bien se ha de llevar a cabo una validación de los datos que nos son entregados siempre, es en las aplicaciones que actúan como servidor donde se convierte en algo critico. Esto es debido a que sin una correcta validación nos podemos encontrar con que los ataques de denegación de servicio son mucho más fáciles de llevar a cabo contra la aplicación.

Ademas de comprobar los datos que entran en la aplicación, también es recomendable realizar una comprobación de los datos que vamos a enviar a otras aplicaciones, esto es recomendable con el fin de tener una mayor seguridad conforme nuestra aplicación esta actuando de la manera esperada.

2.5.3. Asignación y acceso a memoria

Una mala asignación de memoria puede llevar a que se accedan a datos que en otras condiciones no podrían ser accedidos, o incluso enviados al exterior. Un ejemplo de este tipo de fallo se ha podido ver recientemente con heartbleed, donde al llevar a cabo de manera errónea una asignación de memoria se podía llegar a consultar todo el contenido de la memoria RAM de un equipo.

2.5.4. Inyección de código

La inyección de código consiste en enviar como parámetro una cadena de texto que representa código con el fin que la aplicación que recibe dicho parámetro lo ejecute. Los casos más comunes de este tipo de fallo se dan en forma de Cross Site Scripting (también conocido como XSS), en donde se envía código javascript; y en forma de SQL injection, donde se envía parte de una query SQL.

Dado que este tipo de ataques se producen en servidores la repercusión de un solo ataque con éxito puede se devastadora, es por ello que se debe tener cuidado con la validación de datos que se ha comentado anteriormente.

2.5.5. Formato de los datos

Con el fin de evitar sucesos como el accidente que se produjo con la sonda Mars Climate, que se estrello debido a que diferentes módulos de software utilizaban unidades de medida diferentes, es conveniente utilizar el mismo formato de datos en todo el proyecto.

Hacer esto además de mejorar la seguridad del proyecto también nos permite ahorrar costes al no tener que programar, ni utilizar, rutinas que hagan una traducción entre los diferentes formatos.

2.5.6. Criptografía

Si bien cabe esperar que el uso de criptografía mejore la seguridad lo cierto es que el uso de algoritmos criptográficos débiles nos puede llevar a una falsa sensación de seguridad. Y debido a esta sensación de seguridad es posible que terceros sean capaces de obtener información que pensábamos se estaba transmitiendo de forma segura.

La mejor manera de evitar este tipo de fallos es utilizando algoritmos criptográficos que todavía se consideren seguros y que ademas hayan sido validados por la comunidad. Un ejemplo de algoritmo criptográfico débil es el cifrado DES, que debido a la capacidad de calculo de los equipos actuales puede ser roto en pocas horas.

2.5.7. Herramientas de análisis de código estático

El uso de herramientas de análisis de código estático ayuda a localizar puntos vulnerables del código durante la implementacion del proyecto.

2.5.8. Creación de logs

La creación y almacenamiento de logs no aporta seguridad adicional por el hecho de crearlos, la principal función que tienen es permitir que cuando se produzca un fallo o ataque sea posible rastrear el origen del problema para poder subsanarlo.

2.6. Auditoria

Consideramos como pertenecientes a la fase de auditoria aquellos controles que tengan como objetivo validar la seguridad de la implementación realizada del proyecto. En el proceso de auditoria se llevan a cabo dos tipos de análisis, el análisis de caja negra y el análisis de caja blanca.

Análisis de caja negra

En un análisis de caja negra el auditor no recibe información por parte de la organización del proyecto que va a auditar.

Análisis de caja blanca

En un análisis de caja blanca el auditor tiene acceso por parte de la organización a toda la información relacionada con el proyecto.

2.6.1. Técnicas de auditoria

Recolección de información

Es importante a la hora de llevar a cabo una auditoría obtener el máximo de información posible de los servidores o servicios auditados con el fin de encontrar información que no deberíamos poder encontrar.

Para ello disponemos de diferentes opciones que se complementan unas a otras:

- **DNS snooping**
  Consiste en enviar peticiones a un servidor DNS para obtener información sobre que páginas tiene en cache y deducir que accesos se suelen producir .

- **PTR (Reverse DNS Lookup)**
  Consiste en llevar a cabo peticiones PTR, que a partir de una IP nos dan el nombre de dominio.

- **Google hacking**
  Es una técnica que utiliza el buscador de google con parámetros específicos que permiten filtrar por url o ciertas strings que nos pueden permitir descubrir archivos internos del servidor que estén indexados.

- **Metadatos**
  Muchos documentos incluyen información extra, esta información adicional puede consistir en:

- Usuarios
- Ubicación (en el caso de las fotografías)
- Software utilizado y su versión

- **Deep Web y Pastebin**
  Siempre es buena idea buscar las apariciones de la entidad que estemos auditando en la Deep Web y Pastebin, dado que muchos grupos de "moralidad dudosa"suelen dejar ahí la información que descubren.

Escaneo de red y enumeración

Una de las primeras cosas que necesitamos saber es qué servicios están al descubierto, es decir, que son accesibles desde el exterior.

El objetivo del escaneo es para detectar qué IPs y puertos están abiertos y qué servicios hay tras ellos.

Vulnerabilidades comunes

Aunque en la fase de implementacion se deberían haber filtrado las vulnerabilidades más comunes no esta de más comprobar que no se ha escapado ninguna. Las vulnerabilidades más comunes son:

- XSS (Cross Site Scripting)
  Un Cross-Site Scripting ocurre por una falta de validación y tratamiento de los datos de entrada a la aplicación; lo que es aprovechado por los atacantes para realizar inyecciones de código script (Vbscript, Javascript,...)

- Inyección SQL
  Los ataques de inyección SQL son un tipo de ataques de inyección, en los que ordenes SQL son inyectadas en texto para afectar la correcta realización de una consulta SQL predefinida.

  El éxito en una inyección SQL puede conllevar:

  - Leer datos sensibles de la base de datos
  - Modificar los datos (insertar/actualizar/borrar)
  - Realizar operaciones de administración sobre la base de datos.
  - En algunos casos, ejecutar ordenes en el sistema operativo.

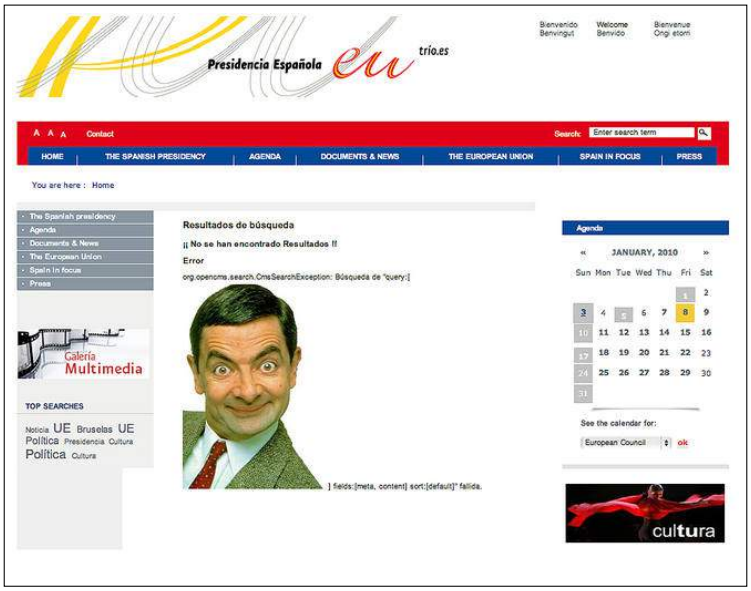

Figura 6: Ejemplo conocido de lo que se puede conseguir con un XSS. (El Mundo)

- Buffer Overflow
  Ocurre al no llevarse acabo una comprobación de una entrada la cual supera el tamaño reservado.

  A través del buffer overflow se puede escribir en partes de la memoria donde el programador en principio no tenía intención que escribiéramos.

  Suelen encontrarse o bien mediante un análisis del código fuente (si es código abierto), o mediante fuzzing de las entradas de una aplicación.

Fuzzing

Fuzzing es una técnica de detección de vulnerabilidades que consiste en introducir valores aleatorios, no válidos o con contenido "peligroso" con la intención de provocar un fallo o comportamiento indebido en un programa o aplicación.

Se utiliza sobretodo para encontrar nuevas vulnerabilidades desconocidas o errores de programación/configuración.

Fuerza bruta

Similar al fuzzing, pero se aplica con la intención de encontrar una entrada o combinación de entradas que provoque un resultado deseado (por ejemplo usuario y contraseña para accede a algún lugar).

Los ataques de fuerza bruta pueden funcionar de dos maneras:

- **Diccionarios**
  Los diccionarios son listados de combinaciones o entradas habituales.
- **Recorriendo todas las posibles entradas**

Por lo general se suele emplear primero la variante de diccionario, y solo en caso que este falle se intenta recorriendo todos los valores. La razón por la que se sigue esta secuencia se debe al elevado coste en tiempo asociado a recorrer todos los posibles valores.

Análisis Manual

Debido a que no las herramientas automáticas presentan limitaciones

- Generan falsos positivos
- No encuentran todas las vulnerabilidades posibles
- A veces no distinguen bien cosas que a un experto le resultan "obvias"

Tenemos la necesidad de recurrir finalmente a un análisis manual. Este tipo de análisis requiere que el auditor posea:

- Conocimientos específicos de lo que se desea analizar
- Intuición y creatividad

2.7. Despliegue

Consideramos como pertenecientes a la fase de despliegue aquellos controles cuyo objetivo sea preparar el proyecto para su puesta en producción. En las guías ofrecidas por SAFECode se han encontrado diferentes controles que se han de tener en cuenta.

2.7.1. Tratamiento de excepciones

Debemos asegurar que el código que sea puesto en producción no muestra al usuario ningún mensaje de error que no haya sido capturado previamente por el propio código. Esto se debe a que una excepción sin capturar proporciona información sobre la estructura del software que puede ayudar a realizar un ataque.

Además de capturar las excepciones también debemos asegurarnos que los mensajes de error mostrados no aporten información adicional sobre el software, en el caso de las excepciones que se produzcan sin haber sido previstas se ha de mostrar un mensaje de error genérico.

2.7.2. Pruebas de carga

Llevar acabo pruebas de carga en el sistema nos permite evaluar cual es el numero máximo de peticiones por unidad de tiempo que nos es posible atender, y por lo tanto configurar la infraestructura de acuerdo con la cantidad de usuarios que esperamos tener.

Además de calcular la cantidad de peticiones que podemos atender también podremos ajustar el sistema para que en caso de un ataque de denegación de servicio (DOS) el sistema no caiga por el exceso de peticiones hasta que logremos filtrar los atacantes.

2.7.3. Comprobación de los binarios

Tras compilar los binarios que serán destinados a producción se recomienda pasar un antivirus en los binarios. Esto se debe a que el objetivo de un ataque puede ser convertir el trabajo llevado a cabo en una fuente de propagación de virus informáticos.

Otra tarea a llevar a cabo sobre los binarios obtenidos es el computo de su firma digital. Con ella podremos detectar cambios que se puedan producir, pues la firma dejaría de coincidir que la almacenada.

2.8. Monitorización

Consideramos como pertenecientes a la fase de monitorizacion aquellos controles cuyo objetivo sea la supervisión del funcionamiento del proyecto una vez se encuentra en producción.

2.8.1. Recolección de logs

A la hora de recolectar los logs generados por los distintos servicios que se encuentran en la infraestructura de la organización disponemos de dos métodos a la hora de enviarlos.

- El primer método consiste en establecer una conexión de red directa con el servidor de logs a través de la cual se envían los logs generados.
- El segundo método consiste en realizar un broadcast de los logs generados en la LAN esperando que el servidor de logs los recoja.

Como se puede ver en el primer caso vamos a tener la seguridad que los logs que se envíen van a ser recibidos, aunque un atacante podrá encontrar sin muchas dificultades donde esta ubicado el servidor de logs. En el segundo caso no sabremos si los logs están siendo recogidos, sin embargo un atacante no sabrá cual es la ubicación del servidor de logs dentro de la infraestructura.

2.8.2. Centralización de logs

Recoger todos los logs generados por el sistema en un solo nodo permite realizar un análisis conjunto. Esto facilita la tarea de encontrar irregularidades producidas por un ataque que se reparten entre múltiples programas.

Si además logramos que el formato de los logs generados sea compatible con herramientas de análisis de logs disponemos de la posibilidad de que el análisis de los mismos sea realizado en parte de manera automatizada, por lo que los ataques más comunes serán detectados con mayor facilidad.

2.9. Mantenimiento

Consideramos como pertenecientes a la fase de mantenimiento a aquellos controles dedicados a asegurar el correcto funcionamiento del proyecto una vez puesto en producción.

2.9.1. Monitorización de vulnerabilidades

Es necesario conocer, almacenar y gestionar las vulnerabilidades que se descubren para comprobar rápidamente si se conoce alguna vulnerabilidad para alguna plataforma si conocemos su versión.

Con el fin de facilitar la identificación de una vulnerabilidad y a que plataforma afecta existe un estándar para la identificación de vulnerabilidades, este consiste en:

- **CVE – Common Vulnerabilities and Exposures**
  Identificador estándar de vulnerabilidades, se trata de un código único para cada vulnerabilidad conocida.
- **CPE – Common Platform Enumeration**
  Identificador estándar de plataforma, se trata de un codigo unico para cada plataforma conocida.
- **Fuente**
  Quien haya emitido el aviso de vulnerabilidad.
- **Descripción**
  Efectos que provoca la vulnerabilidad.
- **Solución**
  Medidas a tomar para protegerse de la vulnerabilidad.

2.9.2. Corrección de vulnerabilidades

Con el fin de corregir una vulnerabilidad el proyecto vuelve a la fase de implementacion, para corregir la vulnerabilidad, donde debemos volver a seguir los controles dispuestos hasta volver a la fase de mantenimiento.

3. Metodología

A continuación se una breve descripción de las metodologías utilizadas como fuente para obtener los diferentes controles de seguridad que se deben cumplir, así como de a que ámbito afectan principalmente.

3.1. Análisis de requisitos y viabilidad

En este apartado destaca la aportación de CISA y de la ISO 27001 a los controles correspondientes a esta fase.

Esto se debe a que en esta fase se debe realizar un trabajo que enfatiza en la preparación de la organización en gestión de la seguridad, apartados en los que destacan CISA y la ISO 27001.

3.2. Infraestructura y comunicaciones

Esta fase concentra un importante volumen de controles y tareas. Este volumen es debido a que suele ser a través de debilidades en la infraestructura y comunicaciones por donde se suelen infiltrar la mayoría de atacantes a la hora de robar información. Otro motivo para este volumen de controles es debido a la cantidad de subsistemas que se han de ser protegidos de ataques.

En este caso la repartición de los controles y tareas es más equitativa entre aquellos controles que han de depender de la organización y aquellos que son propios de cada proyecto.

Aquí destaca la aportación de los benchmarks de seguridad que ofrece CIS, así como la de OSSTMM la única fuente consultada en la que se trata en profundidad la seguridad de las comunicaciones inalámbricas en el modulo de SPECSEC.

3.3. Metodologías de desarrollo

En este apartado encontramos documentación especifica para el apartado de seguridad en metodologías de desarrollo, como es el caso de Microsoft SDL for Agile o la Guidance for for Agile Practioners ofrecida por SAFECODE. OSSTMM también ofrece algo al dar importancia a la forma de trabajar de los empleados dentro de su modulo de HUMSEC.

En este caso y debido a las características de esta fase apenas existen controles que afecten a la organización en su conjunto, pues cada equipo de trabajo debe poder tener su propia metodología de trabajo que se adapte al proyecto en el cual se esta trabajando en ese momento.

3.4. Implementación

Esta fase esta fuertemente dominada por la aportación de las guías ofrecidas por SAFECODE, principalmente porque el objetivo de la organización es la seguridad del código implementado.

En este caso los controles y tareas se encuentran totalmente asignados al proyecto, pues en esta fase el trabajo a realizar recae sobre el equipo de trabajo del proyecto y la organización ya debería de haber aportado las herramientas necesarias en fases anteriores.

3.5. Auditoria

En esta fase la principal aportación es llevada a cabo por OWASP, ya que la documentación que ofrece trata precisamente sobre el modo de llevar a cabo una auditoria de una aplicación. CISA y OSSTMM también ofrecen algún control al ser guías para llevar a cabo una auditoria, sin embargo su objetivo es la seguridad de la información perteneciente a una organización en vez de la seguridad del software desarrollado.

Igual que en el caso de la implementación todos los controles pertenecientes a esta fase están enfocado al proyecto y no abarcan a la organización.

3.6. Despliegue

En esta fase la aportación de controles se reparte entre OSSTMM Y SAFECODE Guidance for Agile Practitioners.

Una vez se ha llegado a esta fase de un proyecto la mayor parte del trabajo ya se ha llevado a cabo y los controles restantes suele ser posible automatizarlos con alguna de las herramientas ya existentes para ello.

3.7. Monitorización

En esta fase la aportación de controles es llevada a cabo principalmente por CISA, con algún aporte de OSSTMM.

3.8. Mantenimiento

En esta fase la aportación de controles es llevada a cabo principalmente por CISA, con algún aporte de OSSTMM y de la ISO 27001.

Aquí vuelve a aparecer algún control del que se debe hacer cargo la organización como es el caso de la eliminación de soportes de almacenamiento y equipos.

4. S2D2

S2D2 es una aplicación creada con la finalidad de facilitar la aplicación de la metodología desarrollada.

4.1. Requisitos funcionales

S2D2 es un sistema que permite gestionar la aplicación y adaptación de la metodología desarrollada en un proyecto concreto.

La adaptación de la metodología se llevara a cabo a través de la información proporcionada a un asistente. Además se dará la opción de modificar manualmente la metodología generada por el asistente en caso que se quiera ajustar con mayor precisión al proyecto a supervisar.

En la vista en la que se permita la edición manual de la metodología deberemos poder

- Añadir nuevos elementos a la metodología.
- Eliminar elementos de la metodología.
- Editar los elementos existentes en la metodología.
- Guardar la metodología que estamos editando.
- Recuperar una metodología guardada.

Una vez generada la metodología que se va a utilizar en el proyecto se procede a iniciar la supervision de su aplicación.

En la vista de supervision se deberá poder

- Consultar la información relativa a los diferentes controles y sus tareas.
- Indicar que tareas ya han sido realizadas.
- Indicar en que fecha se prevé finalizar un control.
- Marcar el momento en que empezamos a trabajar en un control.
- Incrementar el tiempo que se ha estado trabajando en un control.
- Consultar el progreso de un control a lo largo del tiempo
- Consultar el estado de una sección de controles.
- Guardar el progreso de aplicación de la metodología.
- Recuperar un progreso guardado anteriormente.

4.2. Requisitos no funcionales

4.2.1. Utilización

El sistema deberá ofrecer una interfaz de uso sencillo e intuitivo, en particular en la interfaz de supervisión. Pero al ser destinado a personal con conocimientos en informática se espera que sean capaces de entender los términos más técnicos.

4.2.2. Confiabilidad

El sistema deberá ser estable y ser capaz de responder a los posibles fallos que se produzcan sin dañar los datos que se estén utilizando.

4.2.3. Funcionamiento

La interfaz de usuario debe tener una fluidez y tiempo de respuesta que permita una cómoda utilización de la aplicación.

Por ese motivo el número de ventanas y elementos mostrados deberá ser el menor posible sin afectar a la comodidad de uso de la aplicación.

4.2.4. Restricciones de diseño

Debido a que la aplicación esta destinada a personas que llevan a cabo el desarrollo de software, no es posible determinar sobre que sistema operativo deberá ejecutarse la aplicación.

Por este motivo se deberá hacer que la aplicación sea multiplataforma.

4.2.5. Documentación

Además de la presente memoria, el proyecto deberá ir acompañado de una documentación detallada del proceso de instalación del sistema así como de un manual de uso.

4.2.6. Requisitos de licencia

Las licencias de los componentes externos al proyecto deberán autorizar su inclusión en éste así como ser compatibles entre ellas.

Licencias como la GNU General Public Licence (GPL) v2 o v3 cumplen con estos requisitos.

4.3. Arquitectura

4.3.1. Ciclo de vida de una metodología

En S2D2 se diferencian tres estados diferentes en los que se puede encontrar una metodología:

- **Nueva metodología**
  En este estado la metodología todavía no ha sido creada, nos encontramos en este estado en la pantalla de bienvenida del programa y mientras estamos en el asistente de creación de una nueva metodología.

- **Editar metodología**
  En este estado todavía no hemos finalizado de preparar la metodología para su uso en un proyecto concreto. Estamos en este estado tras finalizar el asistente de nueva metodología o escoger editar una metodología existente.

- **Supervisar metodología**
  En este estado estamos aplicando los controles de la metodología en un proyecto. Solo estamos en este estado tras iniciar la supervisión de la metodología que hemos estado editando o escoger la opción de supervisar metodología en la pantalla de bienvenida.

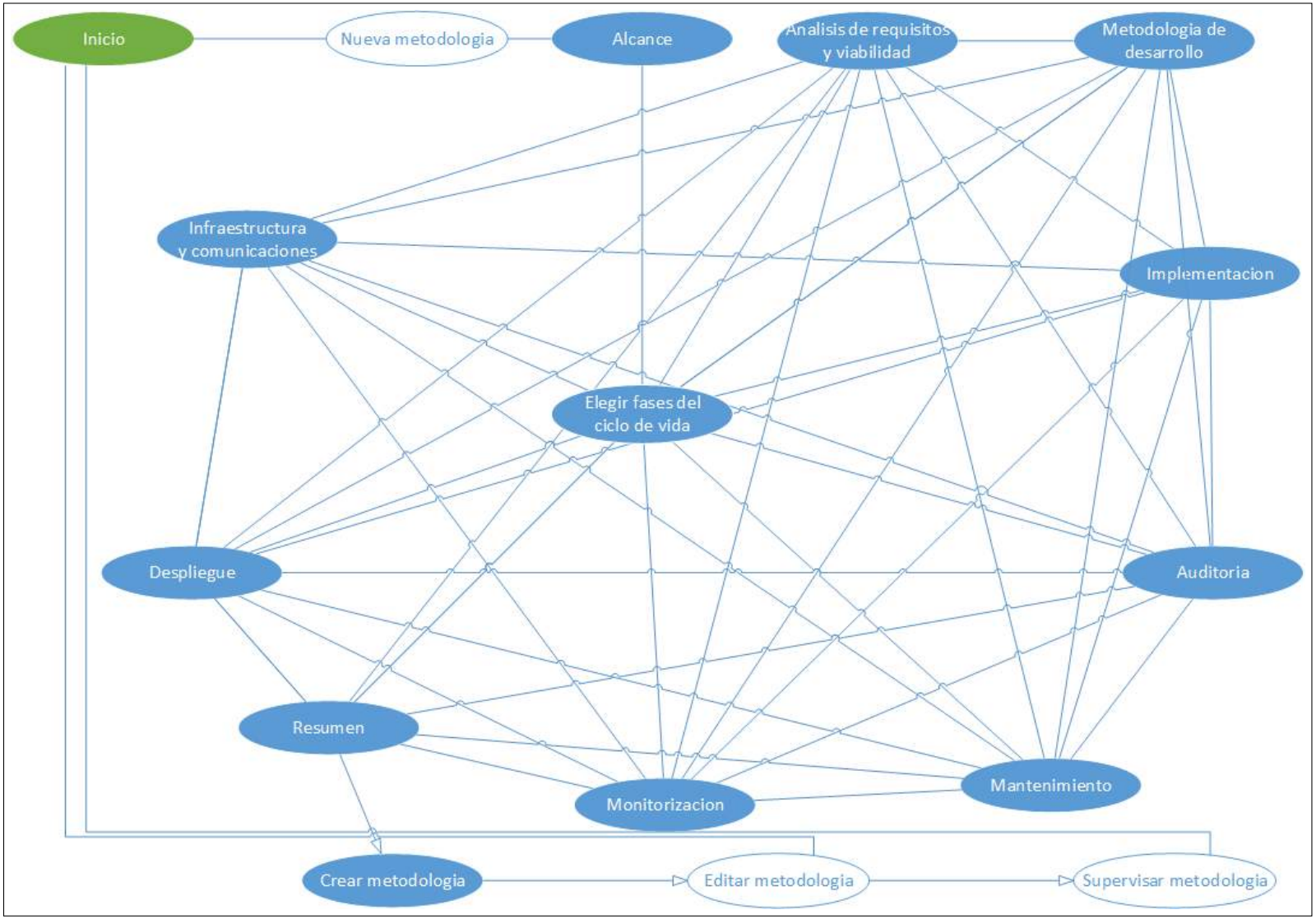

Figura 7: Ciclo de vida de una metodología en S2D2

En la versión actual del programa solo permitimos avanzar en el progreso de aplicación de una metodología, es decir, una vez iniciada la supervisión no es posible añadir, eliminar o modificar los controles que se han de aplicar.

4.3.2. Arquitectura del software

En el diseño de la arquitectura del sistema se ha tenido en cuenta la posibilidad de ampliar o modificar funcionalidades, por lo que se ha optado por aplicar un patrón de MVC (Model View Controller) con la intención de reducir el acoplamiento entre clases del programa.

Además del propio programa se utiliza una base de datos sqlite donde se almacena la información de los controles de seguridad y las tareas con las que el asistente genera una metodología para un proyecto.

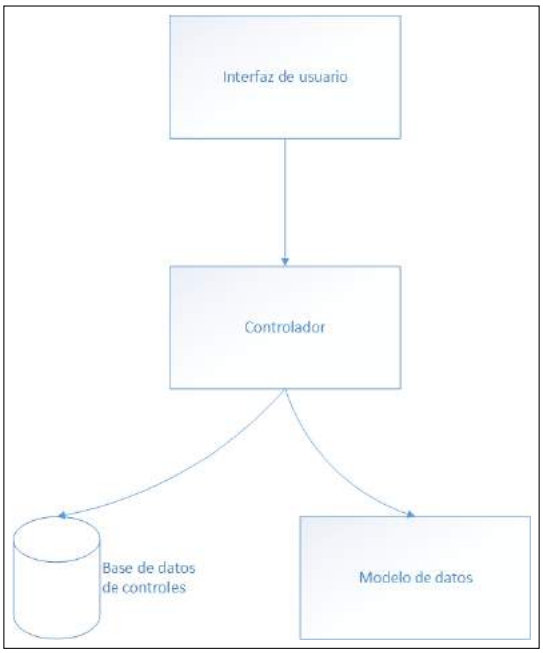

Figura 8: Arquitectura de S2D2

4.4. Decisiones de diseño

En este apartado, se presentan las principales decisiones de diseño que afectan a todo el sistema y a la propia arquitectura del sistema.

4.4.1. Lenguaje de programación

La necesidad de hacer una aplicación que sea multiplataforma y con entorno gráfico limita las opciones disponibles, además como se trata de una herramienta a la metodología desarrollada interesa disponer de una gran variedad de librerías que se encarguen de tareas ajenas a la metodología.

Con el fin de evitar el tiempo de aprendizaje de un nuevo lenguaje se ha decidido emplear Java como lenguaje de programación.

4.4.2. Formato del archivo guardado

Debido a que la aplicación debe poder gestionar más de un proyecto y que la metodología ha de poder adaptarse a las necesidades de cada proyecto el programa ha de poder manejar diferentes ficheros en los que se encuentra guardada la metodología que se esta aplicando, o editando, para un proyecto en concreto.

Para generar los ficheros se dispone de diferentes opciones:

- Utilizar la interfaz Serialize de Java
  Esta interfaz nos permite almacenar el estado de un objeto en formato binario, siendo por tanto muy fácil guardar y recuperar un objeto.

- Utilizar XML
  XML es un estándar para el almacenado de información de forma que sea legible para humanos, en consecuencia permitiría editar el fichero sin necesidad del programa original. Además al ser un estándar abierto asegura independencia del lenguaje de programación utilizado.

- Utilizar JSON
  Al igual que XML con JSON disponemos de un medio de almacenar la información que es independiente de un lenguaje de programación concreto y que puede ser editado manualmente de ser necesario.

Finalmente se ha decidido utilizar XML por ser un estándar con mayor recorrido y encontrarse más fácil de editar manualmente para las pruebas iniciales.

4.4.3. Asistente de nueva metodología

Para la implementación de un asistente se ha optado por emplear una librería que nos facilite la tarea de implementación.

La librería utilizada para esta tarea ha sido la única encontrada para tal efecto, el nombre de la librería utilizada es qdwizard [7].

4.4.4. Gráficas de supervisión

Para la elaboración de las gráficas de progreso se ha utilizado la librería JFreeChart, esta nos permite elaborar una gráfica a partir de los datos recogidos durante la aplicación de cada control.

4.5. Casos de uso

A continuación se presentan los casos de usos del sistema.
Los casos de uso del sistema se dividen en dos grupos.

- **Edición de metodología**
  En estos casos de uso el objetivo es modificar la estructura de la metodología, ya sea añadiendo nuevos elementos, eliminando algún elemento existente o editando algún elemento.

- **Supervisión de metodología**
  En estos casos de uso el objetivo es actualizar el progreso de aplicación de la metodología y obtener gráficas que nos muestren el progreso que se ha ido realizando.

4.5.1. Edición de metodología

[7] El proyecto de desarrollo de la libreria se puede encontrar en: https://github.com/bflorat/qdwizard

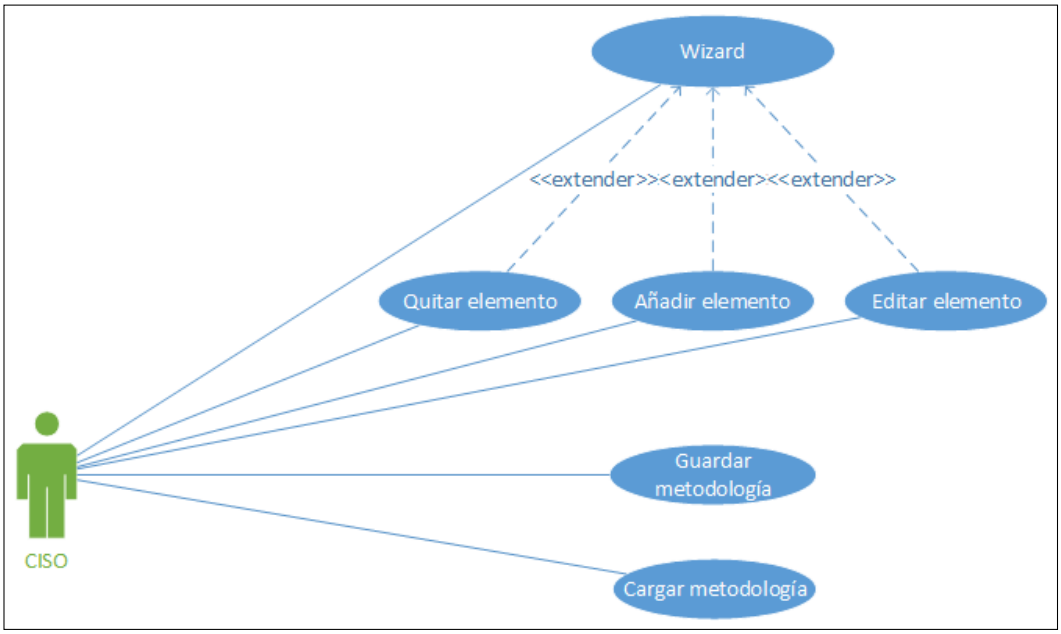

Figura 9: Caso de uso de edición de metodología

4.5.2. Supervisión de metodología

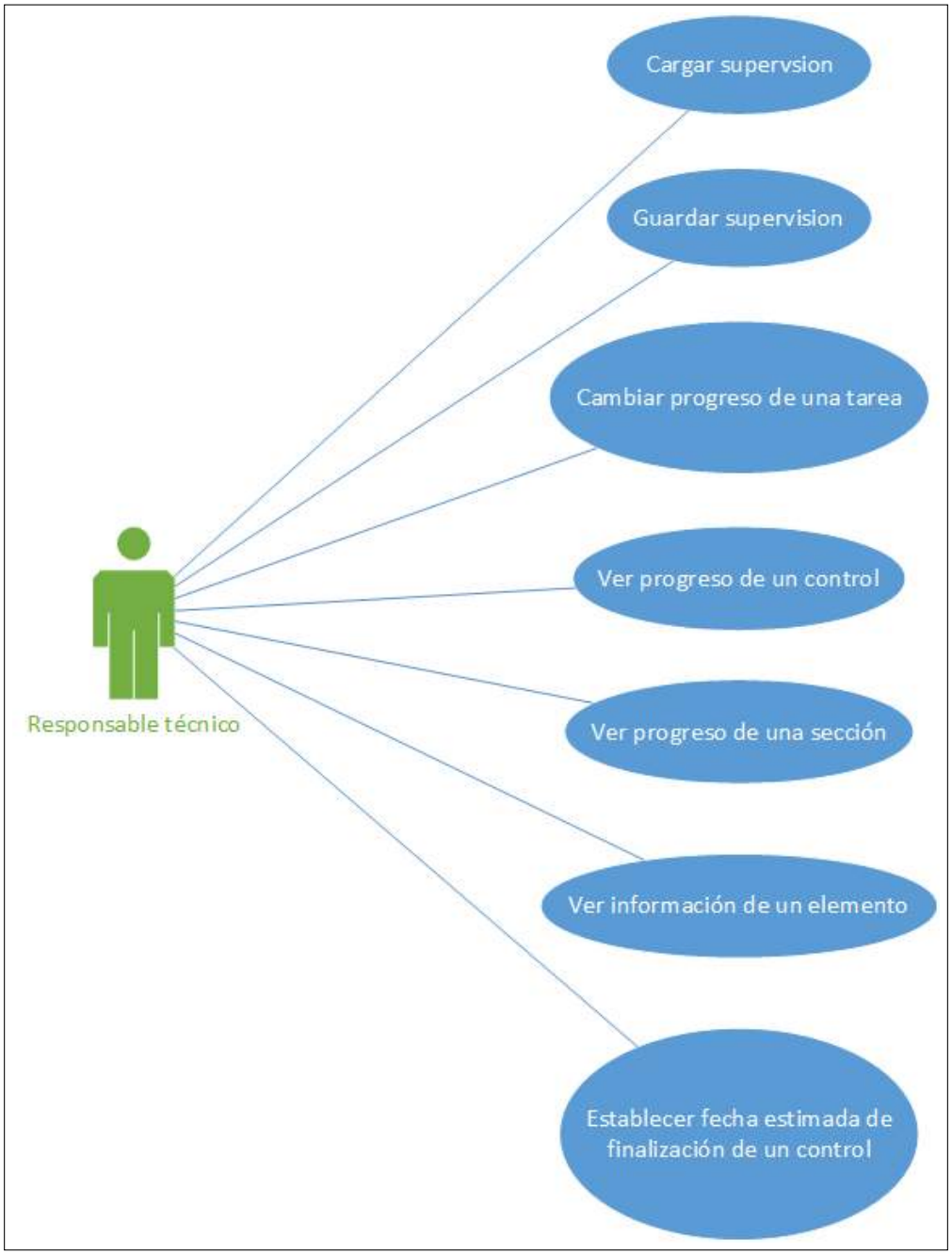

Figura 10: Caso de uso de supervisión de metodología

4.6. Vista lógica

4.6.1. Asistente de nueva metodología

La vista lógica del asistente de creación de una nueva metodología solo tiene como requisito una clase (MethodologyWizard), aunque esta clase es dependiente del resto de clases mostradas en la figura para poder mostrar las diferentes pantallas que pueden ser necesarias en la evolución del asistente.

La clase MethodologyWizard es la responsable de mantener el estado y la información del asistente. Adicionalmente también se responsabiliza de elegir cual sera la pantalla mostrada en caso de querer volver atrás en el asistente, o de querer avanzar. Al finalizar el asistente esta clase procede a consultar la base de datos de controles para generar la metodología solicitada a través de las diferentes respuestas recibidas.

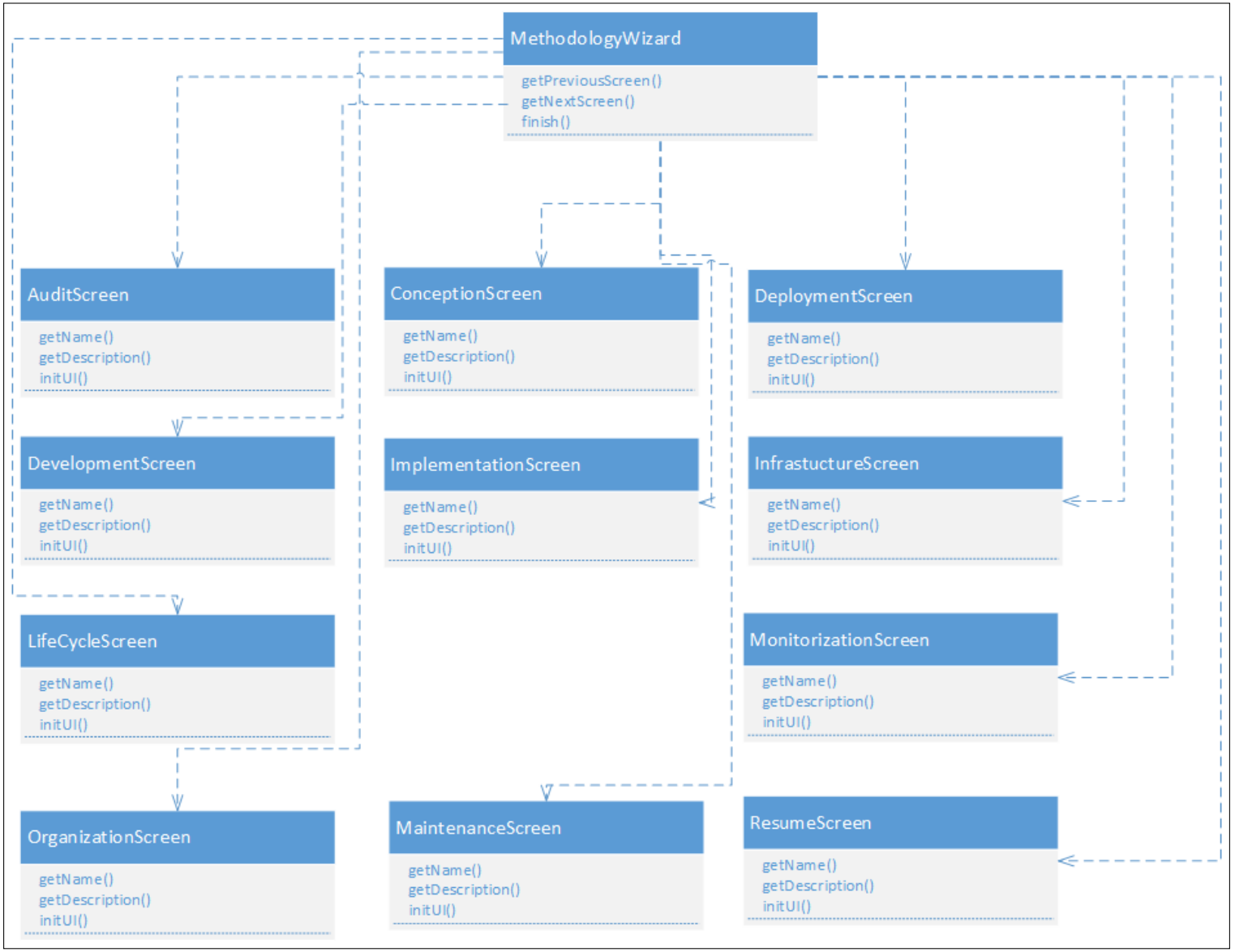

Figura 11: Diagrama de clases del asistente de creación de una nueva metodología

4.6.2. Modelo de datos de la vista de edición de metodología

El modelo de datos utilizado para la visualización de la metodología en edición es un objeto de tipo DefaultTreeModel que requiere de un objeto de tipo DefaultMutableTreeNode que actúe como raíz del árbol, y que a su vez ha de poder tener hijos. Estos nodos a su vez contienen alguna de las implementaciones de EditMethodologyNode.

Esta estructura nos permite poder manipular el contenido y la estructura del árbol sin tener que crear una variante para cada posible combinación de tipos. Además facilita la adición de nuevas funcionalidades al árbol mientras no se modifique la estructura básica del mismo.

La estructura de una metodología tiene la siguiente forma

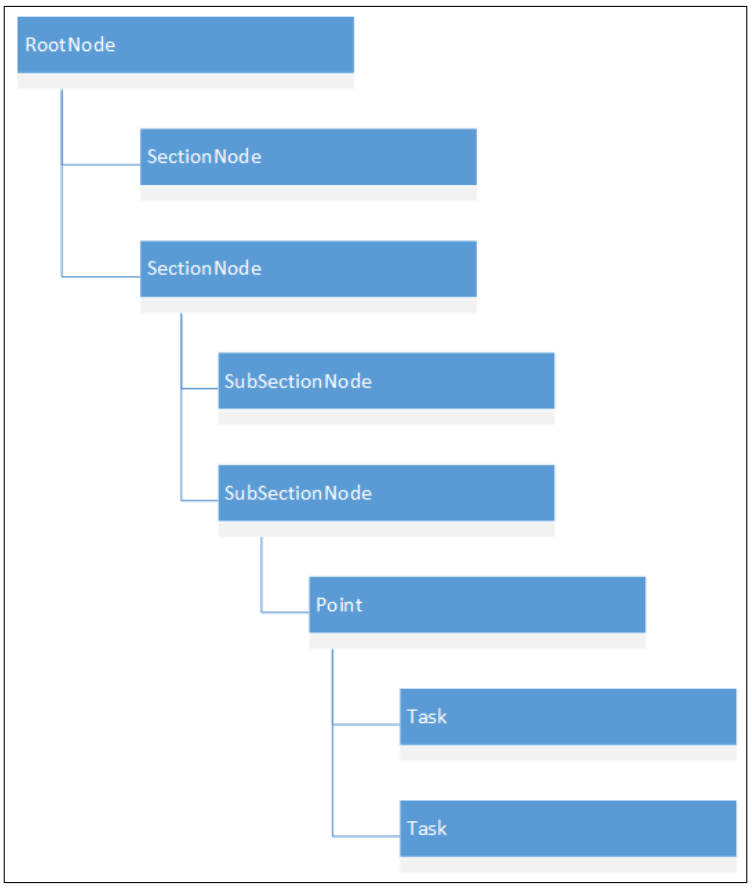

Figura 12: Estructura de una metodología

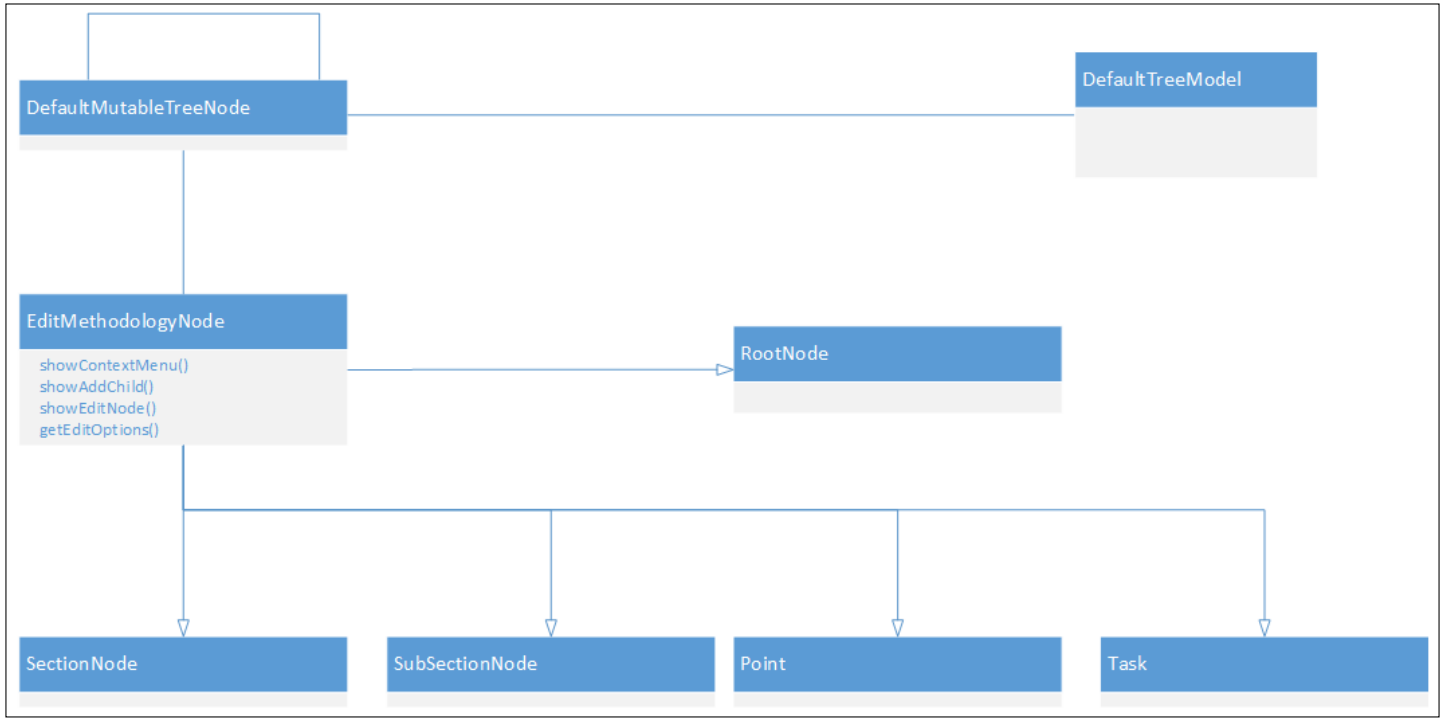

Figura 13: Diagrama de clases del modelo asociado a editar una metodología

4.7. Diagramas de secuencia

A continuación se presentan los diagramas de secuencia correspondientes a los casos de uso no triviales.

4.7.1. Asistente

El asistente de creación de una metodología debe recoger la información necesaria para la selección de los controles y tareas que deberán ser incluidos en la metodología que se genere.

La información que debe recoger incluye:

- **Ámbito de la metodologia**
  Dependiendo de la selección que se lleve a cabo se mostraran controles que afecten a la organización, al proyecto o a ambos.

- **Requisito de confidencialidad**
  En caso que el proyecto no sea de dominio publico permite añadir controles destinados a tal efecto.

- **Etapas del ciclo de vida del proyecto a cubrir**
  Dependiendo del proyecto, o de los servicios requeridos por el cliente puede haber etapas del ciclo de vida que no sea necesario incluir en la metodologia generada.

- **Nombre de los servidores utilizados en el proyecto**
  La lista de nombres facilitada permite crear un grupo de controles identificados por el nombre facilitado para cada servidor empleado en el proyecto.

- **Lenguajes de programación utilizados en el proyecto**
  Dado que cada lenguaje de programación tiene sus peculiaridades, esto permite seleccionar que controles serán necesarios mostrar para la fase de Implementación.

Figura 14: Diagrama de secuencia del asistente para adaptar la metodología a un proyecto concreto

4.8. Implementación

4.8.1. Pantalla de bienvenida

Al iniciar S2D2 se muestra la pantalla que aparece a continuación, en esta pantalla se permite elegir con que tarea se desea empezar a trabajar.

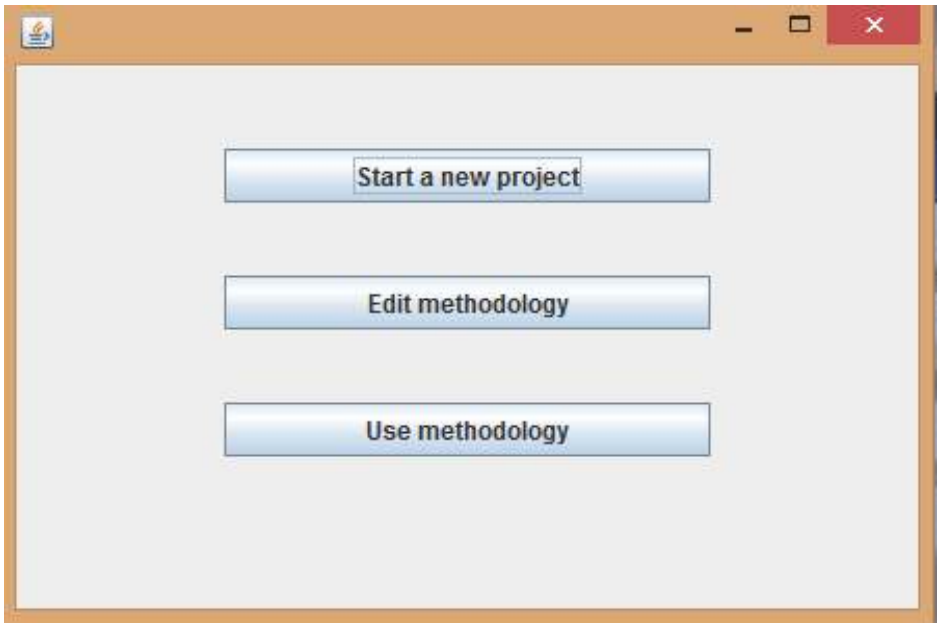

Figura 15: Pantalla de inicio de S2D2

4.8.2. Asistente de nueva metodología

Al seleccionar "Start new project.[en] la pantalla de bienvenida iniciamos el asistente con el que creamos una nueva metodología adaptada a las características del proyecto que facilitamos a S2D2.

Entre las preguntas que se realizan tienen especial importancia el ámbito en que se quiere aplicar la metodología desarrollada; solo al proyecto, solo a la organización o al proyecto y la organización.

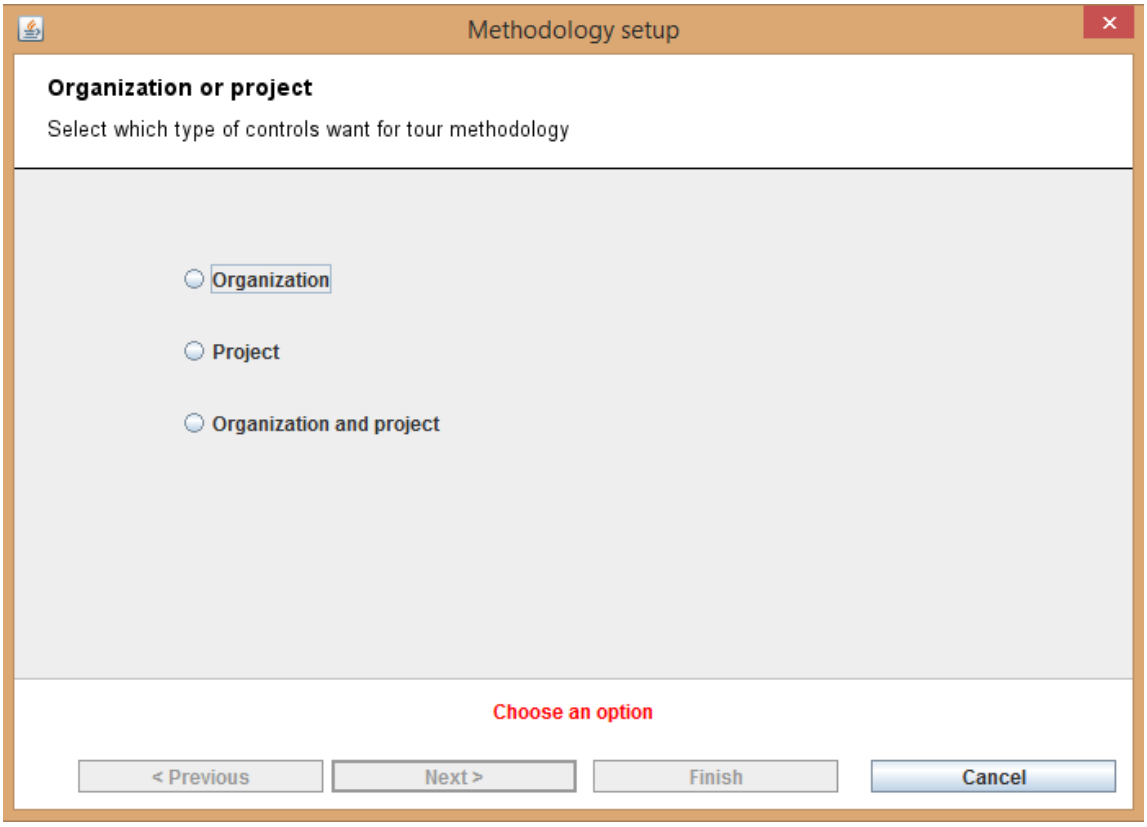

Figura 16: Selección del ámbito de la nueva metodología

También destaca la selección de que fases del ciclo de vida de un proyecto se quieren cubrir.

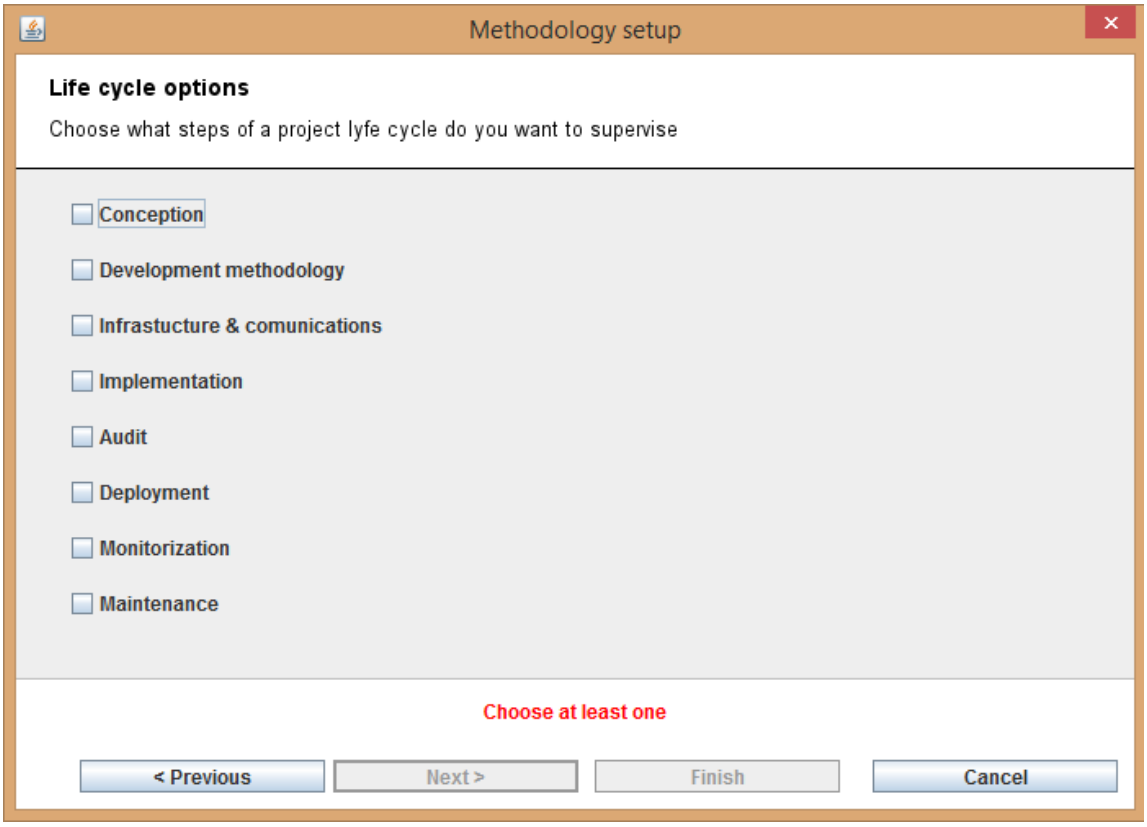

Figura 17: Selección de las fases del ciclo de vida

A la hora de generar la metodología se han creado unos métodos que nos permiten obtener e insertar en el modelo los controles con sus tareas en el modelo de datos que se emplea para la vista de editar metodología.

Este método nos permite obtener los controles pertenecientes a una fase del ciclo de vida del proyecto de la base de datos de controles.

```
private ResultSet retrievePhaseControls(Connection
    connection, int phase, boolean organization, boolean
     project){
  PreparedStatement retrievePhase;
  try {
     retrievePhase= connection.prepareStatement("SELECT␣
         *␣FROM␣control␣c␣WHERE␣c.phase=?");
  } catch (SQLException e) {
     System.err.println( e.getClass().getName() + ":␣" +
```

```
            e.getMessage() );
        JOptionPane.showMessageDialog(null, e.getMessage(),
            "Error", JOptionPane.ERROR_MESSAGE);
        return null;
    }

    try {
        retrievePhase.setInt(1, phase);
        ResultSet queryResult = retrievePhase.executeQuery
            ();
        return queryResult;

    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}
```

A partir de los controles obtenidos con el método anterior podemos proceder a obtener las tareas asociadas al control y a su inserción en el modelo de visualización.

```
private void insertQueryToSection(Connection connection
    , ResultSet query, DefaultMutableTreeNode
    organizationNode, DefaultMutableTreeNode projectNode
    ) throws InvalidOrganizatioalReach{
    try {
        PreparedStatement taskQuery = connection.
            prepareStatement("SELECT * FROM tasks t WHERE t.
            control=?");
        while (query.next()){
            Point control= new Point(query.getString("name"),
                query.getString("description"), query.
                getString("timeCost"), query.getInt("weight"))
                ;
            DefaultMutableTreeNode controlNode= new
                DefaultMutableTreeNode(control);

            //Obtener las tareas para el control
            taskQuery.setInt(1, query.getInt("id"));
            ResultSet tasksQueryResult = taskQuery.
```

```
            executeQuery();
        while (tasksQueryResult.next()){
          Task task= new Task(tasksQueryResult.getString(
              "name"), tasksQueryResult.getInt("weight"));
          DefaultMutableTreeNode taskNode= new
              DefaultMutableTreeNode(task);
          controlNode.add(taskNode);
        }

        //Va en organizacion
        if (query.getBoolean("organization") && !query.
            getBoolean("project")){
          organizationNode.add(controlNode);
        }
        //Va en proyecto
        else if (!query.getBoolean("organization") &&
            query.getBoolean("project")){
          projectNode.add(controlNode);
        }
        //Error sobre donde va
        else{
          System.out.println("Organizational reach: "+
              query.getBoolean("organization")+" and "+
              query.getBoolean("project"));
          throw new InvalidOrganizatioalReach("The
              control with name "+control.getName()+" has
              an invalid organizational reach");
        }
      }
    } catch (SQLException e) {
      e.printStackTrace();
    } finally{
      try {
      query.close();
      } catch (SQLException e) {
      System.err.println("Error al cerrar el resultado de
          la query. "+e.getMessage());
      }
    }
}
```

4.8.3. Editar metodología

En la vista de editar metodología disponemos de una barra de menú desde la cual podemos editar la metodología en desarrollo, también se dispone de las mismas opciones desde un menú contextual.

A la hora de crear o editar un control se ha de establecer los siguientes parametros:

- **Nombre**
  Es el nombre que se da al control
- **Descripción**
  Breve descripción de la finalidad que se quiere lograr con el control
- **Peso**
  Indica la importancia del control respecto al resto de controles de la misma fase. El calculo de la importancia se calcula según la diferencia de peso entre los diferentes controles.
- **Coste temporal**
  Intenta dar una estimación del tiempo que se deberá dedicar para lograr cumplir el control.

Adicionalmente cada control puede tener asociado un fichero externo al programa en caso que se desee añadir documentación adicional.

Para lograr mostrar un menú contextual diferente para cada elemento del árbol aprovechamos la abstracción creada en el modelo de datos de la vista de editar metodología. Con ello podemos llevar a cabo una selección de las acciones relativas al árbol que permite cada elemento e incluso añadir acciones adicionales si así se desea.

```
@Override
public void mouseClicked(MouseEvent e) {
  //Mostrar menu contextual
  if (SwingUtilities.isRightMouseButton(e)) {

    int row = methodologyTree.getClosestRowForLocation(
        e.getX(), e.getY());
    methodologyTree.setSelectionRow(row);
    System.out.println("Right click");
    List<AbstractAction> myActions= new ArrayList<
        AbstractAction>();
    myActions.add(removeNode);
```

```
        myActions.add(addNode);
        myActions.add(editNode);
        ((EditMethodologyNode)((DefaultMutableTreeNode)
            methodologyTree.getPathForRow(row).
            getLastPathComponent()).getUserObject()).
            showContextMenu(methodologyTree, e.getX(), e.
            getY(), myActions);
    }
}
```

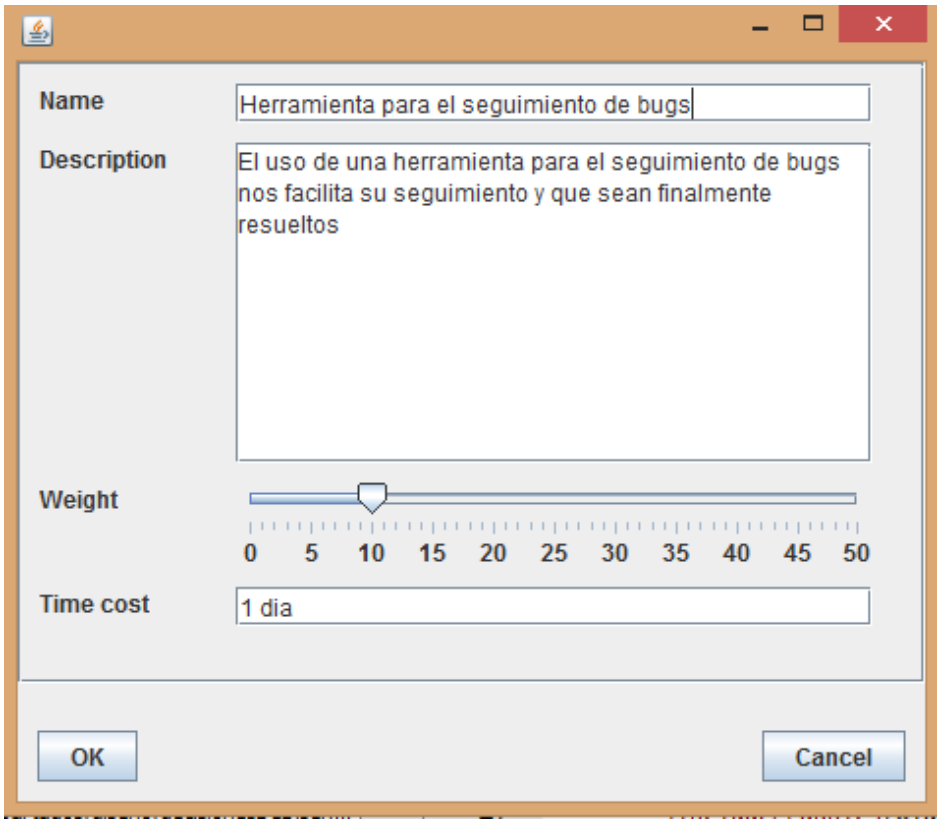

Figura 18: Editar control

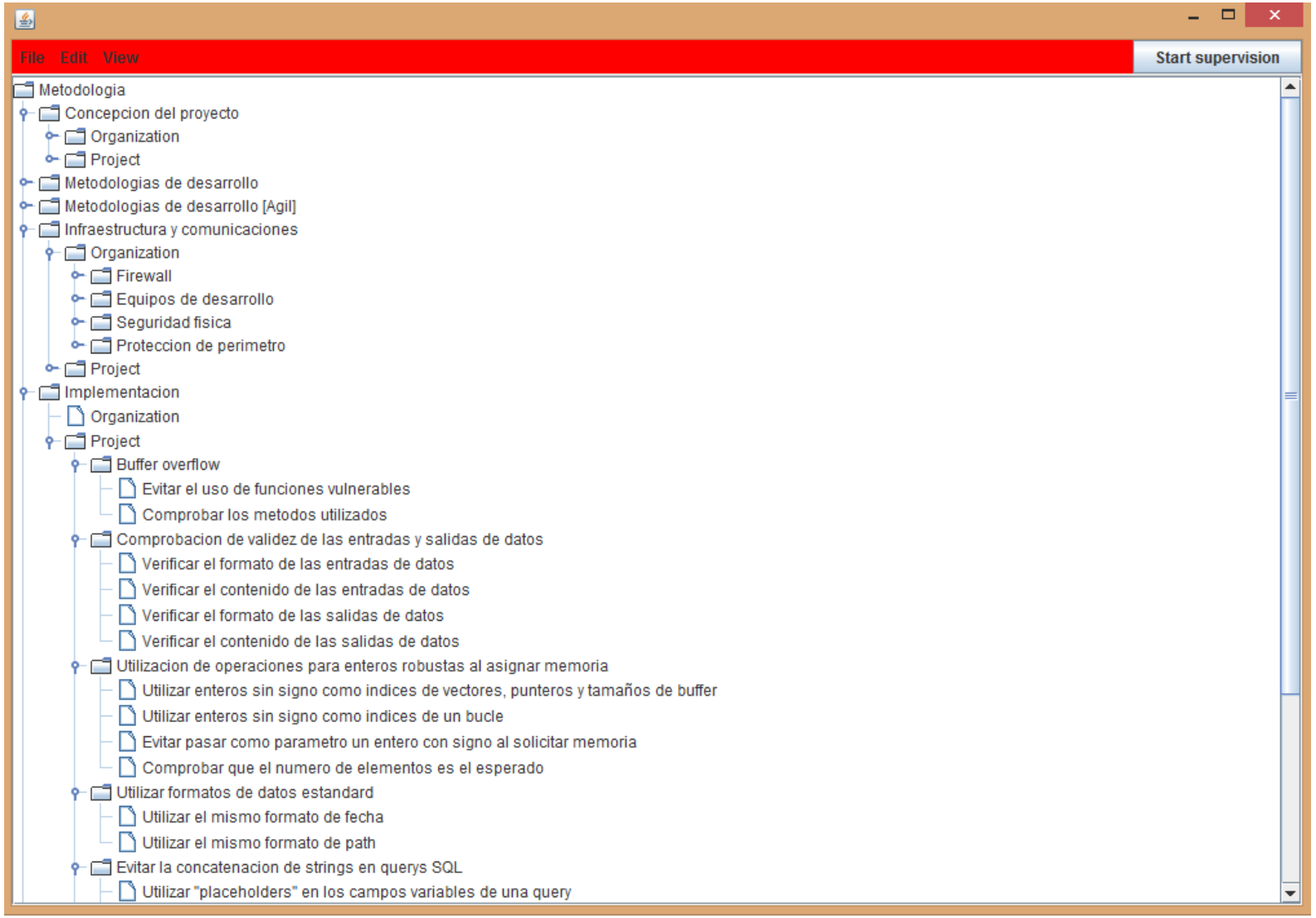

Figura 19: Editar metodología

4.8.4. Supervisar metodología

Una vez se esta supervisando la metodología se puede:

- Marcar una tarea como realizada
- Consultar la información de un control
- Incrementar el numero de horas dedicadas a un control
- Indicar la fecha en que un control debe finalizar
- Marcar el inicio de la aplicación de un control
- Visualizar la gráfica de progreso a lo largo del tiempo de un control
- Visualizar la gráfica con el progreso actual de cada control de una fase

El cálculo de progreso para un control se realiza mediante el siguiente método. La forma en que se realiza el cálculo es similar para el resto de elementos de la metodología.

```
public void updateProgress(UseMethodologyNode node,
    String name) {
  double totalWeight= 0;
  double weightDone= 0;
  for (int childCount=0; childCount<node.getChildCount
      (); childCount++){
    UseMethodologyNodeData child= (
        UseMethodologyNodeData) node.getChildAt(
        childCount).getUserObject();
    if (child instanceof UseMethodologyTaskData){
      UseMethodologyTaskData task= (
          UseMethodologyTaskData) child;
      totalWeight+= task.getWeight();
      if (task.isDone()){
        weightDone+=task.getWeight();
      }
    }
    double newProgress= (weightDone/totalWeight);
    //Anotar modificacion en el progreso del control
    addSnapshot(newProgress, name);
  }
}
```

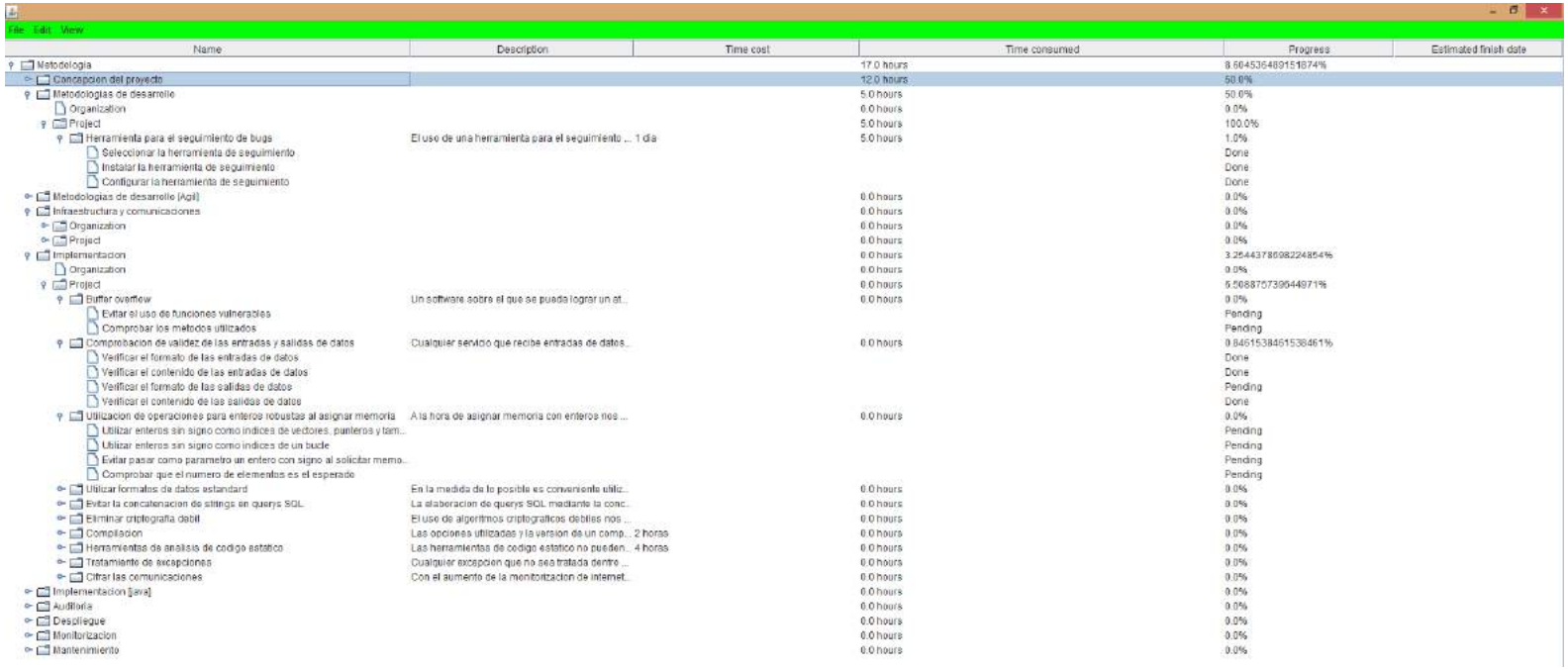

Figura 20: Supervisar metodología

4.8.5. Gráficas de progreso

En la gráfica de progreso de un control se muestra el porcentaje del control que se ha llevado a cabo hasta el momento. La fecha de inicio que se muestra es la fecha en que se indica que se inicia el trabajo en el control, o en caso que no se haya indicado, la fecha en que se indico como completada la primera tarea.

Adicionalmente se indica a que se debe cada variación en el progreso en la propia gráfica.

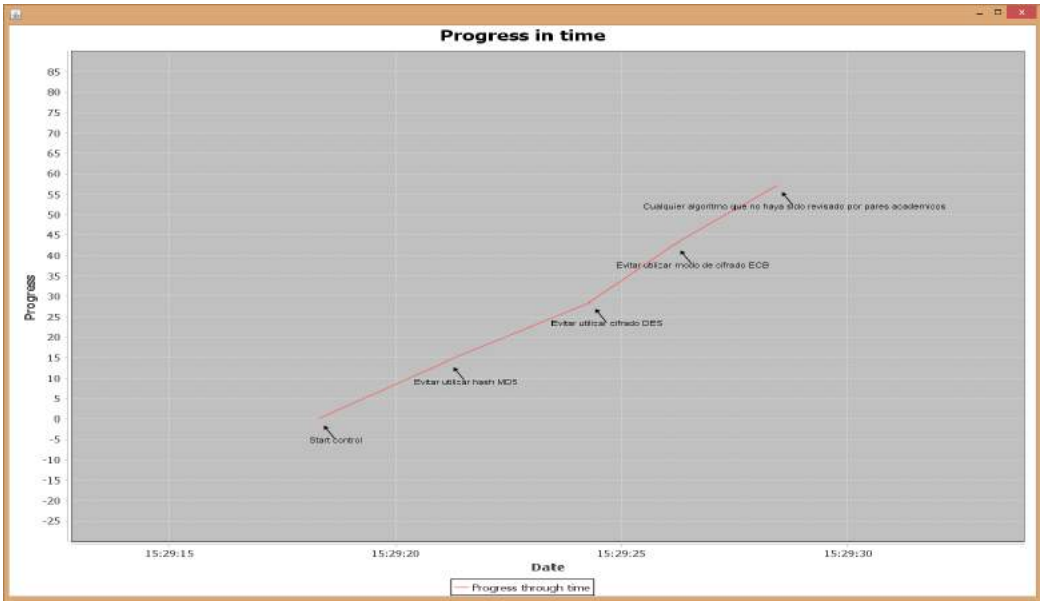

Figura 21: Gráfica de progreso en el tiempo de un control

En esta gráfica se muestra un histograma donde se puede visualizar el progreso actual de cada control asociado a una sección de la metodología. Esto permite al responsable del proyecto obtener una vista rápida de que controles están pendientes por completar.

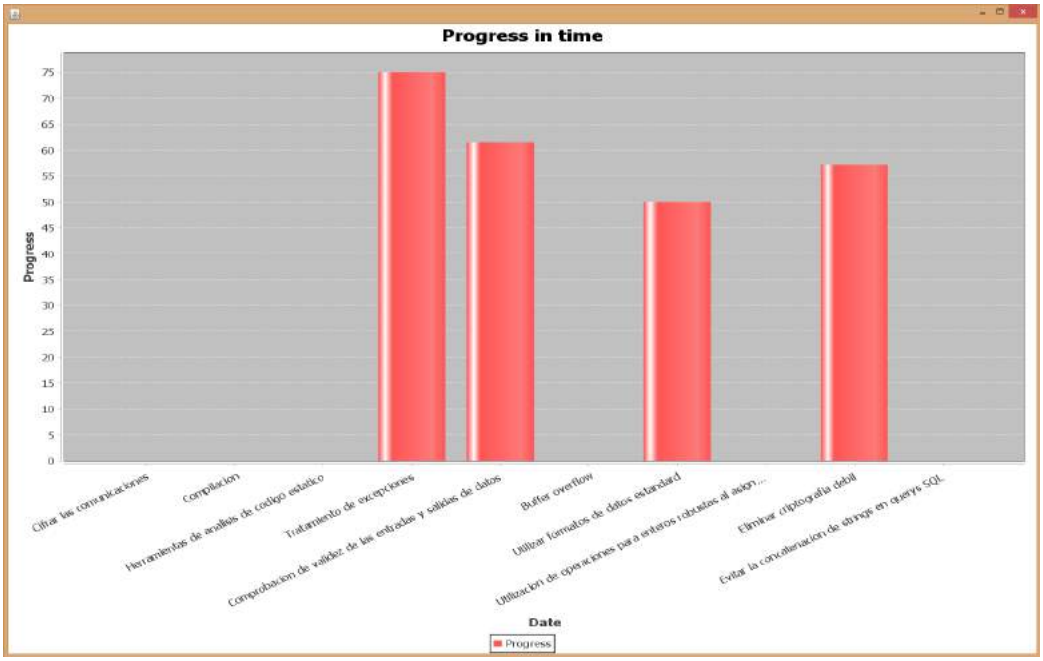

Figura 22: Progreso de los diferentes controles de una fase

5. Casos de uso

Con el fin de poner a prueba la metodología creada y recabar la opinión sobre la herramienta creada (S2D2) y la metodología, se ha ofrecido a diferentes proyectos dentro de InLab que la pusiesen a prueba.

La prueba de la metodología se ha llevado a cabo durante el mes de diciembre donde se les ha solicitado lo siguiente:

- Adaptar la metodología a su proyecto el primer día.
- Comprobar el trabajo realizado sobre la metodología al menos una vez por semana
- Al finalizar el mes de diciembre rellenar un pequeño cuestionario con la información que pudiesen facilitar sobre el proyecto y su opinión sobre la metodología y S2D2.

5.1. TalentSI

Datos del proyecto

Responsable: Albert Obiols
Estimación de personas en personas/mes: 21
Estimación de tiempo en horas o días laborables:

Descripción del proyecto

Es un sistema de formación desde el cual se pueden resolver todas las necesidades derivadas del Programa Talent en una misma aplicación. Cumple determinados procesos administrativos que se llevan a cabo para la realización de la gestión de los estudiantes y permitiendo realizar de manera más sencilla el seguimiento del trabajo por parte de los estudiantes.

Datos técnicos

Lenguajes de programación:

- Java
- Javascript
- HTML / CSS
- Oracle

Descripción Servidores

- Servidor de desarrollo
  - Tomcat
  - Oracle
- Servidor de preproducción
  - Tomcat
  - Oracle
- Servidor de producción
  - Tomcat
  - Oracle

Conclusiones

En este caso de uso la metodología ha aportado una orientación sobre diferentes vulnerabilidades que actualmente se pueden encontrar dentro del proyecto, así como vías para subsanarlas.

5.2. coCar Simulative Evaluation

Responsable: MªPaz Linares
Estimación de personas en personas/mes: 2personas/mes
Estimación de tiempo en horas o días laborables: 8 meses

Descripción del proyecto

Debido a acuerdos de confidencialidad con el cliente no es posible proporcionar una descripción del proyecto.

Datos técnicos

Lenguajes de programación:

- C++
- Python
- R

Descripción Servidores

Ubuntu Server 12.04 16Gb RAM, PostgreSQL, Redis

- Ubuntu Server 12.04 16Gb RAM
  - PostgreSQL
  - Redis

Conclusiones

La responsable de este proyecto ha aportado diferentes ideas sobre como mejorar la usabilidad de S2D2, la aportación más destacada ha sido la sugerencia de crear un asistente para adaptar la metodología al proyecto en curso.

Otra aportación que ha ofrecido este proyecto es la necesidad de establecer controles para diferentes lenguajes de programación, esto se ha debido a la diferencia que hay entre los diferentes lenguajes de programación utilizados en el proyecto.

5.3. Tercer proyecto

Responsable: Carlos Carmona

Nombre del proyecto: CONFIDENCIAL

Estimación de personas en personas/mes: 3 personas/mes (persona = 8h/dia)

Estimación de tiempo en horas o días laborables: 9 meses – 27 meses (depende de ampliaciones)

Descripción del proyecto

Debido a acuerdos de confidencialidad con el cliente no es posible proporcionar una descripción del proyecto.

Datos técnicos

Lenguajes de programación:

- Python
- R

Descripción Servidores

Ubuntu Server 12.04 16Gb RAM, PostgreSQL, Redis

- Ubuntu Server 12.04
  - PostgreSQL 9.3
  - PostGIS 2.1.4

Conclusiones

Debido a los acuerdos de confidencialidad establecidos con el cliente la aportación que ha llevado a cabo el responsable de este proyecto se ha centrado en temas de usabilidad de S2D2. Algunas de las sugerencias que ha ofrecido son: evitar que el árbol que conforma la metodología durante la edición de metodología se colapse cada vez que se añade un nuevo elemento a la metodología y mejorar la información que muestran las gráficas disponibles durante la supervisión de la metodología.

6. Planificación y costes

6.1. Planificación

El proyecto se divide en las siguientes fases:

- Estado del arte, de duración estimada 24 semanas
- Elaboración de la metodología, de duración estimada 4 semanas
- Creación de S2D2, de duración estimada 12 semanas
- Memoria, de duración estimada 2 semanas
- Presentación, de duración estimada 1 semanas

A continuación se presenta el diagrama de Gantt que refleja la planificación inicial.

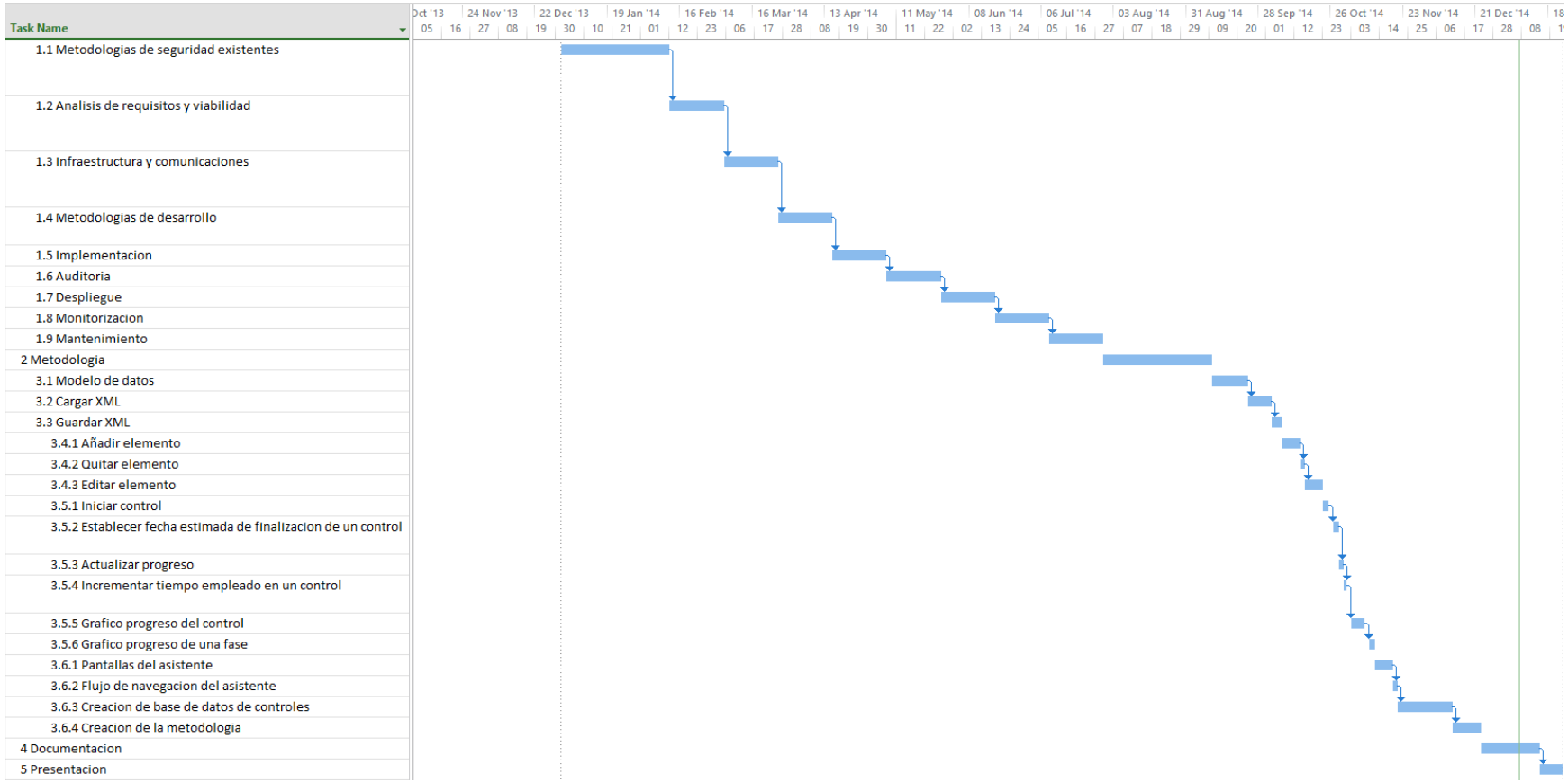

Figura 23: Diagrama de gantt con la planificación del proyecto

6.2. Costes

En este apartado se presenta un análisis económico del coste del proyecto. En cuanto a recursos humanos, deben tenerse en cuenta los costes según el rol responsable de la realización de cada etapa del proyecto. De 760 horas, un 15 % corresponden al jefe de proyecto, un 60 % al consultor y el 25 % restante al programador.

En la tabla siguiente se presentan los costes en recursos humanos por rol.

Rol	Carga de trabajo	Coste hora	Coste total
Consultor	450 horas	60 €/h	27000 €
Programador	190 horas	40 €/h	7600 €
Jefe de proyecto	120 horas	80 €/h	9600 €
			44200 €

Cuadro 2: Tabla de costes en recursos humanos

En cuanto a material, el proyecto ha necesitado de un ordenador de desarrollo, del manual de CISA y de la ISO 27001.

Concepto	**Coste inicial**	**Amortizacion**	**Valor actual**
Ordenador de desarrollo	800 €	200 €	600 €
Manual CISA	100 €	0 €	100 €
ISO 27001	47,61 €	0 €	47,61 €

Cuadro 3: Tabla de costes en material

En cuanto a programas y sistemas operativos, el coste es nulo. Tanto los programas incluidos en el proyecto como los que se han utilizado para desarrollarse son de licencias libres y gratuitas.

A continuación se presenta el resumen y cálculo total del coste del proyecto.

Concepto	**Coste**
Recursos humanos	44200 €
Material	747,61 €
Programas y sistemas operativos	0 €
Total	**44947,61 €**

Cuadro 4: Coste total del proyecto

7. Conclusiones

En este apartado se presentan las conclusiones del desarrollo del proyecto así como algunas funcionalidades que no se han incluidas en el proyecto pero que podrían ser el objeto de un trabajo futuro.

7.1. Desarrollo del proyecto

En este proyecto se ha generado una nueva metodología a partir de diferentes metodologías de auditoria, manuales de buenas practicas, metodologías de desarrollo y aportaciones de responsables de proyecto. Esta metodología logra cubrir todo el ciclo de vida de un proyecto sin obviar ninguna de sus etapas.

Lograr crear esta metodología ha presentado un desafío por diferentes razones.

La primera dificultad que se encontró fue crear controles de la metodología, cuyo objetivo es proteger de un ataque, a partir de los controles ofrecidos por metodologías cuyo objetivo es el opuesto, es decir, lograr vulnerar la seguridad de un proyecto.

La segunda dificultad fue llevar a cabo la fusión de los controles y sus tareas asociadas obtenidos de las diferentes fuentes de información. Esto se debe a que no toda la nomenclatura utilizada es la misma en las fuentes consultadas y obliga a tener cuidado a la hora de unir múltiples controles.

La tercera dificultad ha sido el volumen de controles y tareas obtenido. Con el fin de facilitar la aplicación de los controles y su adaptación a diferentes proyectos ha sido necesario desarrollar un software especifico. Este software permite la edición de la metodología para su posterior aplicación en un proyecto.

El desarrollo de la herramienta S2D2 no ha estado exento de dificultades al ser necesario crear una interfaz de usuario capaz de adaptarse a la estructura cambiante del modelo de datos. Esto ha obligado a emplear capas de abstracción, en forma de Interface en Java, que en caso de no existir una interfaz gráfica habrían sido innecesarias.

Otra de las dificultades que ha presentado el desarrollo de S2D2 ha sido la creación del asistente, que ha obligado a modificar el origen de los controles de XML a una base de datos SQLite para poder generar dinámicamente una nueva versión de la metodología a emplear en cada proyecto.

Durante el desarrollo del proyecto y a medida que se ha progresado en el mismo se ha hecho patente que, si bien es posible y necesario aplicar medidas de seguridad en un proyecto de desarrollo de software, la implicación de la

organización en que se lleva a cabo el proyecto es necesaria si se espera obtener un resultado aceptable en el apartado de seguridad.

7.2. Trabajo futuro

Debido a limitaciones de tiempo no ha sido posible desarrollar en profundidad algunos apartados de la metodología.

Uno de los apartados que ofrece margen para la mejora es el de añadir controles específicos para diferentes lenguajes de programación en la fase de implementación. En la fase de infraestructura y comunicaciones, al igual que en la fase de implementación, cabe la posibilidad de añadir controles específicos para los diferentes servicios que pueda ejecutar un servidor. Un ejemplo serian los servidores web y de bases de datos, en vez de utilizar controles específicos para los diferentes servicios existentes.

Otro apartado que ofrece un margen de mejora interesante se encuentra en S2D2, la herramienta creada para la gestión de la metodologia, donde cabe una mejora en la su usabilidad y las funcionalidades ofrecidas.

Alguna de las funcionalidades que seria interesante añadir al software serian:

- Control de acceso a diferentes usuarios
- Integrar el apartado de infraestructura y comunicaciones con herramientas como son OpenVas
- Permitir la edición de la metodología durante la supervisión de la misma
- Poder ordenar los controles de la metodología según diferentes criterios

Apéndices

A. Tablas metodología

A.1. Análisis de requisitos y viabilidad

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Control *Tarea* **Contratación**								
Comprobación de antecedentes		✓	✓	✓				
Acuerdos de confidencialidad		✓						
Seguro de fidelidad		✓						
Acuerdos de dedicación exclusiva		✓						
Códigos de conducta		✓		✓				
Acuerdos de no competencia		✓						
Manual del empleado								
Políticas y procedimientos de seguridad		✓		✓				
Conductas aceptables y no aceptables		✓		✓				
Legislación			✓	✓				
Satisfacción del empleado								
Ascensos justos		✓						
Complementar la formación de los empleados		✓						
Preparación para usar equipo nuevo		✓						
Ampliar áreas de conocimiento		✓						
Evaluación de rendimiento		✓						
Vacaciones pagadas		✓						

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Despido								
Recoger todas las llaves e identificaciones		✓	✓	✓				
Revocar los permisos en el sistema		✓	✓	✓				
Notificar al personal de seguridad		✓	✓	✓				
Entrevista		✓						
Subcontratación								
Comprobar la legislación vigente		✓		✓				
Establecer un acuerdo de nivel de servicio o SLA		✓						
Definir claramente las responsabilidades		✓		✓				
Presupuesto								
Existe un presupuesto para la seguridad de la información		✓						
El presupuesto es consistente con las necesidades de TI		✓						
Se toman en cuenta las necesidades a largo plazo		✓						
Intercambio de información								
Requisitos			✓					
Detectar falsificaciones			✓					
Medidas contra ingeniería social			✓					

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Se conocen los requisitos para lograr el caos entre el personal			✓					
Segregación de la información personal								
Definir que es información personal			✓					
Protección de la información personal			✓					
Limitar el acceso a la información personal			✓					
Activos materiales								
Controlar con qué frecuencia se intercambian activos materiales asignados a una persona de manera exclusiva			✓					
Controlar con qué frecuencia se venden falsificaciones o licencias de ámbito personal			✓					
Verificar el origen de los activos materiales adquiridos			✓					
Intercambio de información								
Documentar y probar los requisitos necesarios para acceder a información de otros proyectos			✓					
Establecer medios para detectar la validez de las peticiones			✓					

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Existen medios para impedir el envío de información restringida a personas que carezcan del acceso necesario			✓					
Perfil público del personal								
Identificar quien puede llevar a cabo comunicaciones públicas y mediante que formas			✓					
Evaluación de los diferentes métodos de comunicación publica y su eficacia			✓					

A.2. Infraestructura y comunicaciones

Control *Tarea*	**Microsoft SDL for Agile**	**CISA**	**OSSTMM 3**	**ISO 27001**	**CIS Security Benchmark**	**OWASP Testing guide 3**	**SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION**	**SAFECODE Guidance for Agile Practitioners**
Acceso físico								
Definir como se identifica al personal y su nivel de permisos		✓	✓	✓				
Existe un proceso para otorgar permisos y el personal dedicado al cumplimiento del proceso			✓	✓				
Tener en cuenta los horarios y días festivos			✓					
Comunicaciones inalámbricas								
Detectar intrusiones o manipulaciones de la señal inalámbrica			✓					
Se dispone de un mapa de cobertura de la señal			✓					
La señal no es accesible desde fuera de las instalaciones			✓					
Los puntos de acceso no utilizan la configuración por defecto			✓					
Los puntos de acceso solo están conectados el periodo de tiempo que sean necesarios			✓					
La contraseña para acceder al punto de acceso no puede ser adivinada			✓					

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Los equipos están ubicados en armarios o salas con protección electromagnética (jaulas de Faraday, paneles metálicos...)			✓					
La comunicación del punto de acceso inalámbrico con el resto de la infraestructura se realiza mediante fibra óptica siempre que sea posible			✓					
Es necesario autenticarse para acceder al WiFi		✓						
Evitar el uso de WEP		✓						
Comunicaciones								
Enumerar todos los sistemas de comunicación utilizados para poder ser supervisados			✓					
Se monitoriza todo el tráfico de las redes de comunicación en busca de comportamientos anómalos			✓					
Se impide la realización de actividades no autorizadas			✓					
Se realiza un informe sobre cada ataque detectado			✓	✓				
Se identifica la actividad producida por sondas en busca de información del sistema			✓					
Existen sistemas para impedir ataques de fuerza bruta			✓					

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
No es posible conseguir acceso al panel de configuración de la centralita telefónica			✓					
No es posible acceder a los buzones de voz desde el exterior			✓					
Es necesario utilizar algún protocolo de autenticación al conectar con los servicios remotos			✓		✓			
No es posible acceder a paneles de configuración desde terminales o servicios que puedan ser accedidos desde el exterior			✓					
Se dispone de más de un canal para acceder a internet			✓					
Verificar que no existen conexiones carentes de autorización o supervisión			✓					
Todos los servicios existentes son necesarios			✓		✓			
Autenticación (Organización)								
Periódicamente se comprueba la seguridad de las contraseñas mediante ataques de fuerza bruta y diccionario			✓					
Las contraseñas deben ser cambiadas periódicamente		✓	✓	✓				
No es posible para un usuario obtener más privilegios de los que tiene asignados explícitamente			✓					

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Autenticación (proyecto)								
No existe ninguna contraseña por defecto		✓	✓					
Las contraseñas no son fáciles de adivinar			✓					
Las credenciales no se almacenan en claro			✓	✓				
Los algoritmos criptográficos utilizados para codificar las credenciales son considerados como seguros			✓					
Las credenciales se transmiten utilizando criptografía segura			✓					
Todos los intentos de acceso quedan registrados, tengan éxito o no			✓					
Redundancia								
Todos los servicios disponen de una o más alternativas en caso de fallo								
Protección frente a desastres		✓						
Clasificación de criticidad de los diferentes sistemas		✓						
Inundaciones		✓						
Incendios		✓						
Terremotos		✓						
Productos tóxicos		✓						
Pandemias		✓						
Ataques DOS		✓						

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Virus informáticos		✓						
Correo electrónico								
Control del spam		✓						
Antivirus		✓						
Firma digital del mensaje		✓						
Cifrado del mensaje		✓						
Soportes de información extraíbles								
Cifrado de la información		✓						
Limitar los equipos en los que es posible conectar los soportes		✓						
Limitar el contenido que se puede volcar en los soportes		✓						
Informar de la perdida de soportes y su contenido		✓						
Sistema operativo								
Detectar modificaciones en el sistema operativo			✓		✓			
Aislar los procesos unos de otros			✓		✓			
Los procesos se ejecutan con los mínimos privilegios necesarios			✓		✓			
Bases de datos								

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Se llevan a cabo copias de seguridad periódicas		✓						
Hay diferentes grupos de privilegios		✓						
Controlar los privilegios de los diferentes usuarios					✓			
Solo el personal autorizado puede acceder a la base de datos		✓						
Limitar desde donde se puede acceder al servicio					✓			
Eliminar usuarios anónimos					✓			
Renombrar usuario root					✓			
Inventario								
Realizar inventario de equipos y soportes		✓	✓					
Mantener relación de quien tiene cada equipo		✓	✓					
Mantener relación de la ubicación de cada equipo		✓	✓					
Permisos de usuario								
Definir roles de usuario		✓						
Asignar roles a cada usuario		✓						
Dispositivos móviles								

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Bloqueo remoto		✓						
Borrado remoto		✓						
El dispositivo nos puede mandar su posición cuando se lo pidamos		✓						
El contenido del dispositivo tiene replica en los servidores de la organización		✓						
El contenido del dispositivo está cifrado		✓						
Firewall								
Eliminar reglar redundantes					✓			
Comprobar accesos permitidos					✓			
Denegar acceso por defecto					✓			
Equipos de desarrollo								
El equipo dispone de antivirus			✓					
Limitación de permisos a los usuarios					✓			
Seguridad física								
Protección contra electricidad estática				✓	✓			
Protección contra temperaturas elevadas				✓	✓			
Protección de sobretensiones				✓	✓			

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Sistemas antiincendios				✓	✓			
Protección contra inundaciones				✓	✓			
La BIOS o UEFI está protegida por contraseña					✓			
La BIOS o UEFI está actualizada					✓			
Se requiere de una llave para acceder a los servidores				✓				
Protección contra la manipulación del cableado			✓	✓				
Protección de perímetro								
Se rechazan conexiones a servicios comprometidos					✓			
No se ofrece información de los sistemas a través de los documentos públicos					✓			
No se ofrece información de los sistemas internos a través de DNS					✓			
Existe un control de acceso de red centralizado					✓			
Configuración del servidor web								
Creación de certificados					✓			
Configuración de SSL					✓			

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Hacer servir .htacces para evitar el acceso a directorios sensibles					✓			
Configuración del servidor								
Instalar el mínimo conjunto de software que nos permita llevar a cabo la función asignada					✓			
El sistema de arranque está protegido por contraseña					✓			
Se ha instalado un sistema HIDS					✓			
El sistema esta actualizado					✓			
Los permisos para cada aplicación están limitados					✓			
Se ha hecho "chroot" para los diferentes servicios					✓			
Administración del servidor								
El acceso es a través de SSH					✓			
El servidor se identifica con un certificado					✓			
Cada administrador se identifica con su certificado					✓			
Los archivos se transmiten por SFTP o SCP					✓			

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
El acceso desde el exterior de la LAN se hace a través de una VPN					✓			

A.3. Metodologías de desarrollo

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Control *Tarea*								
Formación en seguridad								
Identificar la frecuencia y el tipo de formación de seguridad que ha recibido el personal			✓					
Comprobar el nivel de cumplimiento de las medidas de seguridad por parte del personal			✓					
Comprobar la facilidad con la que un agente externo puede afectar en el cumplimiento de las medidas de seguridad			✓					
Dar formación en seguridad	✓							
Testing								
Automatizar tests	✓	✓						✓
Tests unitarios	✓	✓						✓
Tests de integración	✓	✓						✓
Añadir tests específicos de seguridad	✓							
Lista de herramientas aprobadas								
Software de ejecución	✓							
Entorno de desarrollo	✓							

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Librerías	✓							
Asignar especialista de seguridad al proyecto								
Establecer requisitos mínimos de seguridad	✓							
Llevar a cabo un análisis de riesgo	✓							
Establecer requisitos de diseño	✓							
Crear un plan de respuesta a incidentes	✓							
Reducir la superficie de ataque	✓							
Tareas periódicas								
Revisar la superficie de ataque	✓							
Hacer revisiones de código	✓							✓
Tareas durante el desarrollo								
Eliminar funciones inseguras	✓							

A.4. Implementación

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Control *Tarea*								
Buffer overflow								
Comprobar los métodos utilizados						✓	✓	
Evitar el uso de métodos vulnerables						✓	✓	
Comprobación de validez de las entradas y salidas de datos								
Verificar el formato de las entradas de datos						✓	✓	
Verificar el contenido de las entradas de datos							✓	
Verificar el formato de las salidas de datos						✓	✓	
Verificar el contenido de las salidas de datos							✓	
Utilización de operaciones para enteros robustas al asignar memoria								
Utilizar enteros sin signo como índices de vectores, punteros y tamaños de buffer							✓	

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Utilizar enteros sin signo como índices de un bucle							✓	
Evitar pasar como parámetro un entero con signo al solicitar memoria							✓	
Comprobar que el número de elementos es el esperado							✓	
Utilizar formatos de datos estándar								
Utilizar el mismo formato de fecha						✓	✓	
Utilizar el mismo formato de path						✓	✓	
Utilizar la misma unidad de medida						✓	✓	
Inyección de código								
Utilizar "placeholders" en los campos variables de una query						✓	✓	
Filtrar los datos a ubicar en los placeholders						✓	✓	
Comprobar el tipo de dato						✓		
Comprobar la longitud del string						✓		
Comprobar el rango del entero						✓		
Eliminar criptografía débil								
Evitar utilizar hash MD4							✓	

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Evitar utilizar hash MD5							✓	
Evitar utilizar hash SHA1							✓	
Evitar utilizar cifrado DES							✓	
Evitar utilizar cifrado RC4							✓	
Evitar utilizar modo de cifrado ECB							✓	
Cualquier algoritmo que no haya sido revisado por pares académicos							✓	
Compilación								
Tratar los warnings como errores						✓	✓	
Utilizar la versión más actualizada del compilador						✓	✓	
Herramientas de análisis de código estático								
Escoger la herramienta de análisis estático de código							✓	
Integración de la herramienta con el entorno de desarrollo							✓	
Tratamiento de excepciones								
El sistema debe capturar todas las excepciones								✓

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Los códigos de error deben estar claramente definidos								✓
Las excepciones inesperadas deben tener un código de error genérico								✓
Ninguna excepción debe ofrecer su traza al usuario final								✓
Creación de logs								
Se generan entradas de registro con los eventos producidos			✓				✓	
Las entradas de registro utilizan un formato compatible con la herramienta de recolección de logs utilizada							✓	

A.5. Auditoria

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Control *Tarea*								
Information gathering								
Google hacking						✓		
Metadatos de los documentos accesibles						✓		
DNS Snooping						✓		
Reverse DNS lookup						✓		
System fingerprint						✓		
Descubrimiento de servicios públicos						✓		
Vulnerabilidades comunes								
SQL injection						✓		
Cross Site Scripting						✓		
Buffer overflow						✓		
Denegación de servicio		✓	✓			✓		
Comprobar autenticación								
Transmisión segura de credenciales								
Obtener lista de nombres de usuario								

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Comprobar si es posible esquivar el sistema de autenticación						✓		
Verificar la robustez del sistema de recuperación de contraseña						✓		
Verificar la desconexión de una sesión						✓		

A.6. Despliegue

Control *Tarea*	**Microsoft SDL for Agile**	**CISA**	**OSSTMM 3**	**ISO 27001**	**CIS Security Benchmark**	**OWASP Testing guide 3**	**SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION**	**SAFECODE Guidance for Agile Practitioners**
Comprobar existencia de virus en el binario								
Pasar el antivirus a los binarios que se despliegan								✓
Pruebas de capacidad								
Buscar el máximo de carga que puede soportar el sistema			✓					✓
Establecer límites en la configuración para evitar alcanzar el máximo de carga			✓					✓
Confirmar el funcionamiento de los limites			✓					✓
Los límites de carga forman parte de la configuración del sistema			✓					✓
Integridad de los binarios								
Obtener firma digital de los binarios desplegados			✓					
Comprobar que la firma de los binarios desplegados no ha cambiado periódicamente			✓					

A.7. Monitorización

Control *Tarea*	**Microsoft SDL for Agile**	**CISA**	**OSSTMM 3**	**ISO 27001**	**CIS Security Benchmark**	**OWASP Testing guide 3**	**SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION**	**SAFECODE Guidance for Agile Practitioners**
Recolección de logs								
Centralización de logs		✓	✓					
Rotación de logs		✓						
Clasificación de los logs		✓						
No es posible introducir desinformación o manipular el contenido de los logs			✓					
Monitorización de red								
Controlar los fallos de transmisión		✓						
Monitorizar protocolos de comunicación		✓						
Monitorización de máquinas y servicios								
Instalación de la herramienta de monitorización		✓	✓					
Añadir máquinas y servicios al sistema de monitorización		✓	✓					

A.8. Mantenimiento

Control *Tarea*	**Microsoft SDL for Agile**	**CISA**	**OSSTMM 3**	**ISO 27001**	**CIS Security Benchmark**	**OWASP Testing guide 3**	**SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION**	**SAFECODE Guidance for Agile Practitioners**
Software equipos								
Todo el software esta actualizado con los últimos parches de seguridad			✓					
Migración de datos								
Preparar los datos para la migración		✓						
Preparar el proceso de migración		✓						
Es posible hacer rollback de la migración de datos		✓						
Realizar la migración de datos								
Soportes de almacenamiento								
Eliminar información de soportes				✓				
Actualizar el inventario con el nuevo status del medio		✓						
Limitar los soportes en que se puede almacenar información		✓	✓					
Backups								
Establecer plan de backups		✓		✓				

	Microsoft SDL for Agile	CISA	OSSTMM 3	ISO 27001	CIS Security Benchmark	OWASP Testing guide 3	SAFECODE Fundamental Practices for Secure Software Development 2ND EDITION	SAFECODE Guidance for Agile Practitioners
Inventariar los soportes utilizados		✓						
Guardar las copias de seguridad en una ubicación física diferente		✓	✓					
Eliminación de equipos								
Borrado seguro de los discos duros				✓				
Borrado y eliminación del etiquetado				✓				
Actualizar inventario		✓						

Glosario

certificado digital Un certificado digital consiste en una clave publica para todos y una clave privada. Mediante la clave privada se pueden cifrar mensajes que solo sera posible descifrar mediante la clave publica.

Deep Web La deep web consiste en aquellos servicios que no son catalogados por los buscadores de internet, como es Google. En la deep web resulta fácil encontrarse con ofertas de servicios que son ilegales.

firma digital La firma digital de un documento se obtiene mediante algoritmos hash, estos generan una cadena de bits de tamaño fijo a partir de la entrada que se les suministra. Un buen algoritmo hash ofrece una salida cuyo tamaño supera los 256 bits y además garantiza que un pequeño cambio en la entrada produce un gran cambio en la salida.

gobernanza En una organización consiste en la forma de trabajar que establece la dirección.

Interface En Java, consiste en un tipo de objeto diferente de las class. El objetivo es proporcionar acceso a un mismo grupo de métodos, cuyo objetivo es el mismo, a diferentes class.

LAN Siglas de Local Area Network. Una LAN es una red de comunicación digital el alcance de la cual se limita a un grupo reducido de equipos; que ademas se encuentran geográficamente cerca, habitualmente dentro del mismo edificio.

network adress translation Traducción de una dirección de red en otra.

noSQL Se dice de una base de datos que no sigue el modelo relacional.

Pastebin Pastebin es un servicio online donde sus usuarios pueden colgar de manera anónima cualquier tipo de información que queda disponible para el público.

PostGis Extensión para la base de datos PostgreSQL. Esta extensión añade soporte para bases de datos con información de geoposicionamiento.

Redis Redis es una base de datos noSQL que almacena el contenido de la base de datos en la memoria RAM, permitiendo así un rápido acceso a la información guardada.

S2D2 Nombre de la herramienta software desarrollada con el fin de administrar la metodologia creada en este PFC. El nombre S2D2 deriva de Software Security Driven Development.

sistema de ficheros Método de almacenamiento de datos utilizado por el sistema operativo.

Talent Programa de formación para estudiantes dentro de InLab con una beca de colaboración, esta patrocinado por: Everis, Incubio y UpcNet.

Referencias

[1] ISECOM, *OSSTMM 3 – The Open Source Security Testing Methodology Manual.*

[2] OWASP, *OWASP TESTING GUIDE 2008 V3.0*

[3] ISO/IEC, *ISO 27001 Primera Edición 2005-10-15*

[4] ISACA, *CISA Review Manual 2014*

[5] SAFECODE, *Fundamental Practices for Secure Software Development 2ND EDITION*

[6] SAFECODE, *Guidance for Agile Practitioners*

[7] Microsoft, *Simplified Implementation of the Microsoft SDL*

[8] CIS, *Center for Internet Security Benchmark for Debian Linux v1.0*