
# Universidad Nacional Autónoma de México

# Programa de Posgrado en Ciencia e Ingeniería de la Computación

Desarrollo e implementación del Método SOFUNI “guía para desarrollo de software en el ámbito universitario”. apoyado por una herramienta digital

# T e s i s

Que para optar por el grado de:

## Maestro en Ciencia e Ingeniería de la Computación

Presenta:
**María Guadalupe Vázquez Salazar**

Tutor:
**Mtra. María Guadalupe Elena Ibargüengoitia González**
Programa de Posgrado en Ciencia e Ingeniería de la Computación

Ciudad Universitaria, CD. MX. noviembre 2019

# Tabla de contenido

# Introducción

Cada día la industria de desarrollo de software se vuelve más exigente, lo que conlleva a la necesidad de crear productos de software más complejos en menos tiempo con especificaciones de calidad más estrictas; como consecuencia la academia debe proporcionar a los futuros profesionistas los conocimientos y las herramientas de Ingeniería de Software, preparándolos para cumplir con las expectativas y necesidades de la industria de software.

## Antecedentes

Trabajando en la Facultad de Estudios Superiores Cuautitlán (FESC), como profesor de asignatura en las carreras de Informática e Ingeniería en Telecomunicaciones, Sistemas y Electrónica (ITSE). Se observa que, en varias materias, se desarrolla software para resolver diversos problemas. En esas aplicaciones se hace énfasis y se califican los aspectos relacionados con la asignatura correspondiente. En ocasiones, docentes de otras áreas, como por ejemplo las administrativas, les solicitan a sus estudiantes aplicaciones para automatizar procesos relacionados con temas propios de sus respectivas asignaturas; sin embargo, no existe un método para guiar a los alumnos en el desarrollo de esas aplicaciones, por lo que deben buscar sus propios recursos, cayendo muchas veces en errores comunes conocidos como "Antipatrones" (William J. Brown, 1998), que son vicios dentro del proceso de desarrollo, y como consecuencia, es común obtener aplicaciones que no resuelven el problema planteado, o no en su totalidad, o bien son terminados fuera de tiempo, además de no cumplir con las características mínimas de calidad que proponen los estándares internacionales.

Esta situación repercute en la vida laboral de los egresados, y les impide cubrir las expectativas que el campo laboral se exige.

Revisando el plan de estudios de las carreras de Licenciatura en Informática, y de Ingeniería en Telecomunicaciones Sistemas y Electrónica (ITSE), de la Facultad de Estudios Superiores Cuautitlán (FESC), se identificaron asignaturas relacionadas con algún aspecto del diseño, el desarrollo de software, o bien con la gestión de desarrollo de proyectos de

software. (Tabla I), En la carrera de Informática se imparten las asignaturas de Programación III, Programación IV, en las que se deben realizar productos de software, haciendo énfasis en su construcción mediante lenguajes de programación tales como C# y java respectivamente. En la asignatura de Informática III, se estudian conceptos de Ingeniería de Software y metodologías de desarrollo, mientras que en Informática IV se tratan temas relacionados con Gestión de Proyectos; por lo tanto; en ambas asignaturas se pide a los alumnos que desarrollen un producto de software usando una metodología específica. Una parte importante en el diseño de software son las Bases de Datos y en asignaturas tales como Introducción a Bases de Datos, Desarrollo de aplicaciones de Base Datos y Bases de Datos Avanzadas, se debe realizar el diseño y construcción Bases de Datos, así como consultas sobre los datos almacenados, para lo cual se solicita la elaboración de productos de software, que permitan verificar los conocimientos y habilidades adquiridas en Bases de Datos por los estudiantes. De forma similar en las asignaturas de Seminario de Aplicaciones I y II y Seminario de Desarrollo de Aplicaciones para Dispositivos Móviles los alumnos deben desarrollar productos de software para Web o para los dispositivos móviles.

En cuanto a la carrera de Ingeniería en Telecomunicaciones Sistemas y Electrónica (ITSE), las asignaturas también, exigen que los estudiantes desarrollen productos de software, ya sea que se trate de diseño (Base de Datos, Bases de Datos Avanzadas), Gestión de Proyectos (Desarrollo de Proyectos de Software), en la aplicación de metodologías para desarrollo de software y conceptos de Ingeniería de Software (Ingeniería de Software) o en el desarrollo de interface usando lenguajes de programación (Diseño de interfaces).

En todas las asignaturas antes mencionadas se requiere que los estudiantes desarrollen productos de software, dando especial atención a las áreas de conocimiento de Ingeniería de Software relacionadas con la asignatura correspondiente.

En este trabajo se propone el Método SOFUNI el cual servirá de guía para que los alumnos desarrollen software para sus proyectos universitarios, y un apoyo a los docentes, para que puedan enfocarse en el área de conocimiento referentes a sus asignaturas.

Tabla I. Asignaturas en la FES Cuautitlán. relacionadas con algún aspecto del diseño, el desarrollo de software, o bien relacionadas con la gestión de desarrollo de proyectos de software. (de creación propia)

| **Carrera: Informática** | |
|---|---|
| Programación III. Programación Visual<br>Informática III. Análisis y Diseño de Sistemas I | Tercer semestre |
| Informática IV. Análisis y Diseño de Sistemas II<br>Programación IV. Programación de Interfaces | Cuarto semestre |
| Informática V. Industria del Software<br>Introducción a las Bases de Datos | Quinto semestre |
| Desarrollo de Aplicaciones de Base de Datos | Sexto semestre |
| Seminario de Desarrollo de Aplicaciones Web I<br>Seminario de Bases de Datos Avanzadas<br>Seminario de Desarrollo de Aplicaciones Web II<br>Seminario de Desarrollo de Aplicaciones para Dispositivos Móviles | Optativas |
| **Carrera: Ingeniería en Telecomunicaciones, Sistemas y Electrónica (ITSE)** | |
| Ingeniería de Software | Tercer semestre |
| Base de datos | Quinto semestre |
| Base de datos avanzadas<br>Desarrollo de Proyectos de Software<br>Bases de Datos Espaciales<br>Diseño de Interfaces | Optativas |

## Objetivo General

Diseñar un método llamado SOFUNI (Software Universitario), para el desarrollo de software, basado en normas y estándares, que ayude a los estudiantes de licenciatura a crear productos de software de calidad, de manera rápida (máximo un semestre) y una herramienta digital, en la cual se apoyará el método; Implementar y evaluar el método SOFUNI; construir la herramienta digital.

**Objetivos particulares:**

- Analizar normas y estándares que se usarán como base en el diseño del método.
- Definir el método mediante el diseño de un conjunto de prácticas que lo integrarán, así como las actividades de cada práctica.
- Establecer una serie de plantillas para ser llenadas en las actividades de cada una de las prácticas del método, para la generación de la documentación necesaria basada en normas y estándares.
- Probar y evaluar el método.
- Construir la herramienta digital para apoyar la aplicación del método, que proporcione de manera gráfica la ayuda necesaria para el desarrollo de software.

## Contribución y relevancia

En este trabajo se propone crear **un método que guíe a los estudiantes de licenciatura a desarrollar software de calidad, usando procesos y buenas prácticas.** El método, guiará el diseño y desarrollo de un producto de software y será apoyado mediante la creación de una herramienta digital. El Método SOFUNI será consistente con las necesidades de gestión de proyectos y desarrollo de software del ámbito universitario; es decir, proyectos de desarrollo de software individuales o en equipos de trabajo pequeños, tomando en cuenta que los integrantes del equipo tienen otras actividades académicas, y cuyo tiempo total para su realización es menor a un semestre.

El método se compone de una serie de prácticas para administrar proyectos y desarrollar productos de software de calidad basándose en estándares internacionales como PMBOK® (PMI, 2017) SBOK (SCRUMstudy, 2016) , ISO/IEC 29110 (NTP-RT-ISO/IEC TR 29110-5-1-2, 2012).

La herramienta digital guiará a los estudiantes en la realización de cada una de las prácticas, proporcionando además una base de conocimientos con los fundamentos teóricos necesarios para su realización, como fuente de consulta, al mismo tiempo que se generará la documentación requerida y ayudará a los estudiantes a familiarizarse con la forma de trabajo mediante procesos.

# Capítulo 1. Marco teórico

## 1.1 La Esencia de la Ingeniería de Software

La Ingeniería de Software es una profesión que se enfoca en las innovaciones y mejores prácticas en el desarrollo de software. La Esencia es un marco de pensamiento en forma de un núcleo accionable, que puede ayudar a cualquier equipo de trabajo que desee mejorar su forma de trabajo y equilibrar sus riesgos. La Esencia surge como núcleo de la Ingeniería de Software en respuesta a la convocatoria lanzada por SEMAT (Software Engineering Method and Theory) (Jacobson, 2013) para definir una base teórica sólida y común para la Ingeniería de Software, promovida por Ivar Jacobson, Bertrand Meyer y Richard Soley, entre otros. El núcleo se refiere a los elementos esenciales que son universales a todos los esfuerzos de desarrollo de software y un lenguaje sencillo para describir métodos y prácticas. Cuando se desarrolla software hay "Cosas que siempre hacemos" y "cosas con las que siempre trabajamos". Tiene tres características únicas, las cuales son: accionable, extensible y práctico.

- **Accionable**. Las "cosas con qué trabajar" se capturan como alfas en lugar de productos de trabajo (tales como documentos).
- **Extendible**. Se puede extender para apoyar diferentes tipos de productos (por ejemplo, nuevos desarrollos, mejoras a sistemas legados, desarrollos internos, desarrollos externos, líneas de productos de software, etc.)
- **Práctico**. Es un marco de pensamiento tangible y práctico que apoya a los profesionales de software a medida que realizan su trabajo.

Tanto el núcleo y el lenguaje en la Esencia de la Ingeniería de Software están diseñados para apoyar a los profesionales, así como a los ingenieros de métodos a definir "métodos" a partir de una arquitectura como se muestra en la figura 1.1. Donde un método es una composición de prácticas que se describen usando los elementos del núcleo, y del lenguaje, lo que permite que una práctica se pueda integrar, de forma segura, con otras prácticas para formar el método. (Jacobson, 2013).


Figura 1.1. Arquitectura para la definición de métodos (de creación propia)

En el núcleo de la Esencia de la Ingeniería de Software se organiza en tres áreas específicas, cada una referente a un aspecto particular de Ingeniería de Software, las cuales se muestran en la figura 1.2

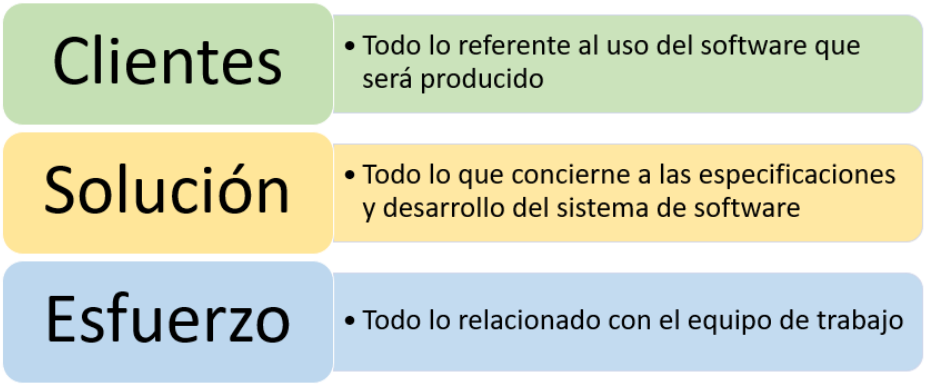



Figura 1.2. Áreas que integran la Esencia (de creación propia)

Cada área se integra mediante un conjunto de alfas, actividades y competencias.
Una alfa del inglés ALPHA (Abstract-Level Progress Health Attribute), es un elemento del esfuerzo de Ingeniería de Software que se usa para evaluar el progreso y la salud del esfuerzo.

La Esencia consta de 7 Alfas.  cada una tiene un conjunto de estados predefinidos, que expresan que una situación determinada se cumple. Figura  1.3

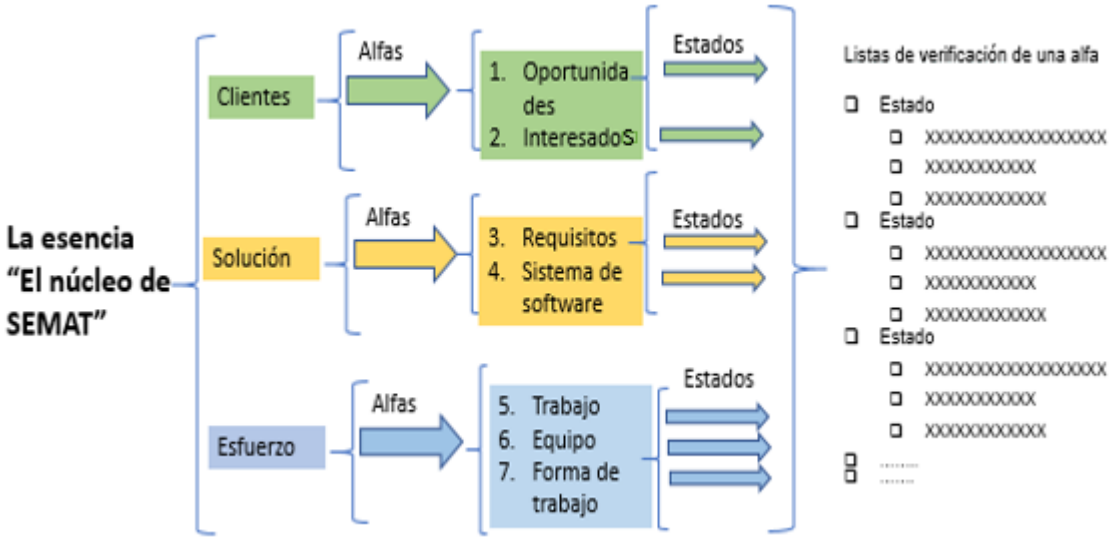



Figura 1.3. Áreas, Alfas y sus estados (de creación propia)

- Área del Cliente. Contiene las alfas de oportunidad (opportunity) e interesados (involucrados). la oportunidad representa todo lo que le da el valor al que va a invertir en el sistema. Los interesados son principalmente los que necesitan y pagan por el sistema de software, así como los que lo usarán y mantendrán

- Área de la Solución. Contiene las alfas de requerimientos (requirements) y sistema de software (software system). Como su nombre lo indica, todo lo que tiene que ver con la solución del software.

- Área del Esfuerzo/Proyecto. Esta área considera los elementos para realizar el proyecto tales como equipo de trabajo (team), forma de trabajar (way of working) y trabajo (work) para implementar el sistema. (Oktaba, 2013)

La forma en cómo se relacionan las alfas del núcleo que pertenecen a diferentes áreas entre sí se muestra en a Figura 1.4

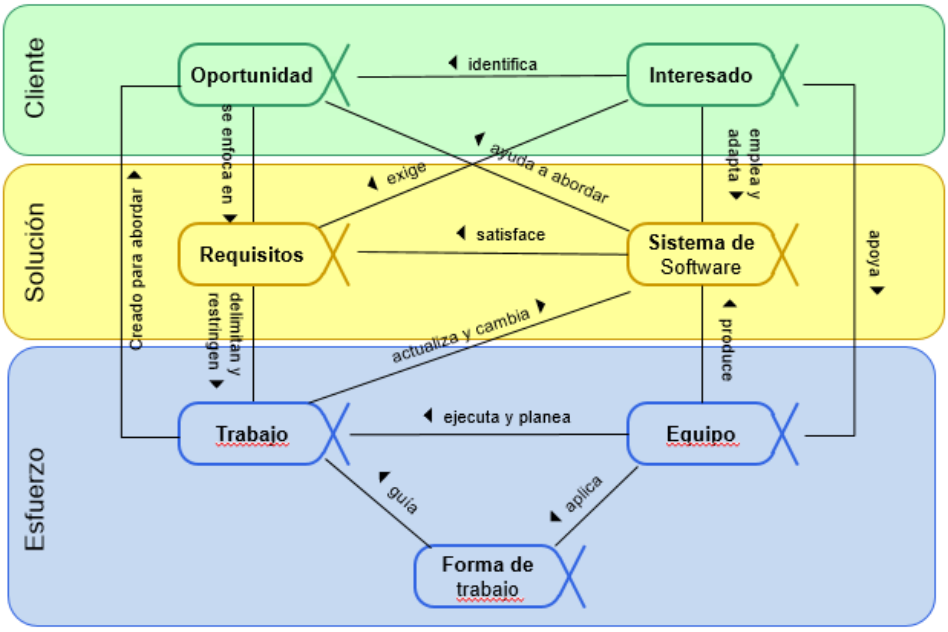



Figura: 1.4 Obtenida de La Esencia de la Ingeniería de Software: El Núcleo de Semat. Revista Latinoamericana de Ingeniería de Software, 1(3): 71-78, ISSN 2314-2642 Carlos Mario Zapata Jaramillo (Traductor), Ivar Jacobson, Pan-Wei Ng, Paul E. McMahon, Ian Spence, Svante Lidman. 2013

El núcleo, es un marco de pensamiento tangible y práctico que apoya a los profesionales de software a medida que realizan su trabajo. Empleando tarjetas (figura 1.5), Las tarjetas proporcionan recordatorios concisos y señales para los miembros del equipo, a medida que realizan sus tareas diarias. Proporcionando listas de verificación y sugerencias.


Figura: 2.5 obtenida de La Esencia de la Ingeniería de Software: El Núcleo de Semat. Revista Latinoamericana de Ingeniería de Software, 1(3): 71-78, ISSN 2314-2642 Carlos Mario Zapata Jaramillo (Traductor), Ivar Jacobson, Pan-Wei Ng, Paul E. McMahon, Ian Spence, Svante Lidman. 2013.

**Kuali Beh**

Kuali Beh Es un marco de trabajo para la expresión, comparación y mejora de las formas de trabajo de los practicantes de Ingeniería de Software involucrados en proyectos de software. Este marco de trabajo forma parte del apéndice B de la Esencia. Kuali Beh está integrado por un núcleo de conceptos comunes relacionados con Ingeniería de Software, dichos conceptos son aplicables a la definición de métodos y prácticas independientemente del tamaño y complejidad de los proyectos.

De acuerdo con las definiciones que proporciona Kuali Beh, un método persigue un propósito relacionado con desarrollar, mantener, operar o integrar un producto de software, considerando las necesidades de los involucrados. El conjunto de prácticas que lo compone debe contribuir al logro de este propósito (Management, Group Object, 2014) tal como se esquematiza en la figura 1.6

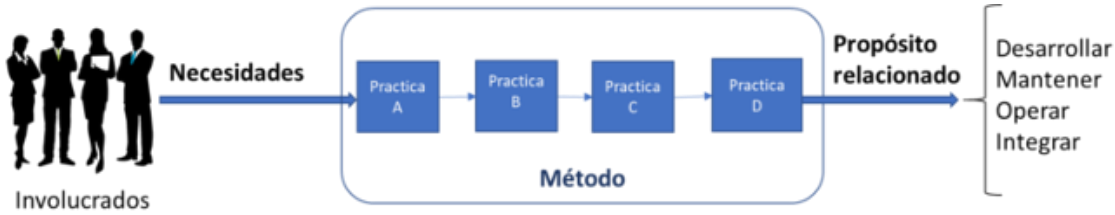



Figura 1.6. Definición de método según Kuali Beh (de creación propia)

Cada práctica debe generar un resultado a partir de una entrada. El resultado de una práctica siempre es la entrada de otra práctica, a menos que sea la última práctica y el resultado será el producto de software.

Cada práctica estará organizada en una serie de actividades y se puede definir como una unidad de trabajo.

## 1.2 Gestión de Proyectos

Un proyecto es un esfuerzo **temporal** que se lleva a cabo para producir un producto, servicio, o resultado, que es **único**. (Project Management Institute, 2017)

- **Temporal**, implica que un proyecto tiene un principio y un final definidos.
- **Único**, ya que es un conjunto específico de operaciones diseñadas para lograr una meta particular.

Ejemplos de proyectos pueden ser: La construcción de un edificio, el lanzamiento de un nuevo producto al mercado, el desarrollo de un software para mejorar un proceso de negocio, un festival de cine, etc.

La Gestión de Proyectos es la aplicación del conocimiento, habilidades, técnicas para ejecutar proyectos en forma eficiente y efectiva, se ha practicado desde épocas remotas; sin embargo, se empezó a buscar su reconocimiento como profesión a mediados del siglo XX.

Project Management Institute (PMI) es la organización mundial sin fines de lucro que aporta a la Gestión de Proyectos estándares y certificaciones reconocidas mundialmente, a través de comunidades de colaboración. ( (Project Management Institute (PMI), 2019).

La guía del PMBOK® ("Body Of Knowledge" (BOK) llamado "Project Management Body Of Knowledge") no es una metodología, ya que una metodología es un conjunto de prácticas, técnicas y reglas usadas por aquellos que trabajan en una disciplina específica, la guía del PMBOK® en cambio, proporciona los fundamentos sobre los cuales se pueden construir metodologías, políticas, procedimientos, reglas, herramientas y técnicas.

La guía del PMBOK®, es un estándar PMI, que evolucionó a partir de las buenas prácticas reconocidas de los profesionales dedicados a la Gestión de Proyectos que han contribuido a su desarrollo; proporciona, además detalles sobre conceptos clave, tendencias, consideraciones para adaptar los procesos a la Gestión de Proyectos, e información sobre

cómo aplicar las herramientas y técnicas a los proyectos siempre alineadas a los procesos del estándar. ( (Project Management Institute, 2017)

## Grupos de procesos

La Gestión de Proyectos según PMBOK® se lleva a cabo en los siguientes grupos de procesos.

1. **Inicio**: Aquellos procesos realizados para definir un nuevo proyecto o nueva fase de un proyecto existente y obtener la autorización para iniciar.
2. **Planeación**: Aquellos procesos requeridos para establecer el alcance del proyecto, refinar los objetivos y definir el curso de acción requerido para alcanzar los objetivos propuestos del proyecto.
3. **Ejecución**: Aquellos procesos realizados para completar el trabajo definido en el plan del proyecto.
4. **Monitoreo y control**: Aquellos procesos requeridos para rastrear, revisar y regular el progreso y el desempeño del proyecto, para identificar áreas en las que el plan requiera cambios y para iniciar los cambios correspondientes.
5. **Cierre:** Aquellos procesos realizados para finalizar todas las actividades a través de todos los Grupos de Procesos, a fin de cerrar formalmente el proyecto o una fase del mismo. (Ver figura 1.7)

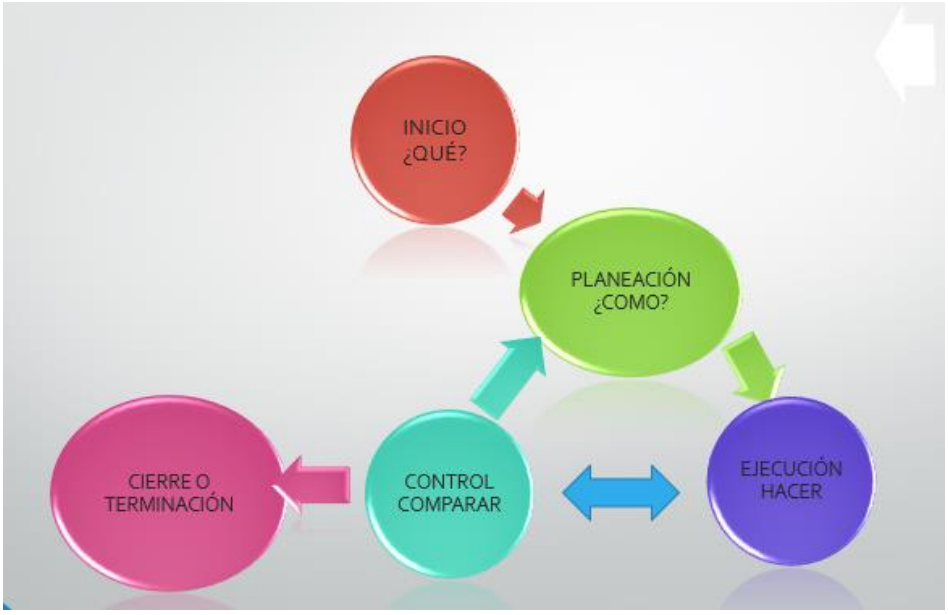


Figura 1.7 Los grupos de procesos y la forma de cómo se relacionan (de creación propia)

## Áreas de conocimiento

Además de grupos, los procesos también se organizan en áreas de conocimiento. Un Área de conocimiento es identificada y definida por sus requisitos y descrita en términos de sus procesos, prácticas, entradas, salidas, herramientas y técnicas. Las áreas de conocimiento se resumen en la tabla 1.2

Tabla 1.1. Las áreas de conocimiento definidas en el PMBOK®..(De creación propia)

| Área | Descripción |
|---|---|
| **Integración** | Manejar la relación entre todas las áreas de conocimiento |
| **Alcance** | Definir y controlar qué se incluye y qué no se incluye en el proyecto |
| **Tiempo** | Programar la duración y secuencia de las actividades, y desarrollar y controlar el cronograma |
| **Costos** | Estimados de costo, presupuesto, programa de erogaciones |
| **Calidad** | Actividades que determinan responsabilidades, objetivos y políticas de calidad para que el proyecto sea ejecutado satisfactoriamente |
| **Recursos humanos** | Equipo del proyecto, colaboradores internos y externos, roles y funciones de cada uno |
| **Comunicaciones** | Información requerida, quién la genera, y quién<br>La recibe, con qué frecuencia |
| **Riesgos** | Identificar amenazas a controlar, oportunidades y crear planes de contingencia |
| **Adquisiciones** | Compra o adquisición de los insumos, bienes y servicios que se requiere para realizar el proyecto |
| **Interesados** | Identificación de las personas, grupos u organizaciones que pueden afectar o ser afectados por el proyecto y conocer sus expectativas |

## Relación entre grupos de procesos y áreas de conocimiento

Un proyecto puede requerir de varias áreas de conocimiento; por tanto, la tabla 1.2 muestra la relación que hay entre cada una de las áreas de conocimiento y los diferentes grupos de procesos, así como los capítulos donde se puede encontrar la información correspondiente en la guía del PMBOK®.

Tabla 1.2. Project Management Institute, A Guide to the Project Management Body of Knowledge, (PMBOK® Guide)Sixth Edition, Project Management Institute, Inc., 2017, Table 1-4 Page 25.

| Áreas de Conocimiento | Grupos de Procesos de la Dirección de Proyectos | | | | |
|---|---|---|---|---|---|
| | Grupo de Procesos de Inicio | Grupo de Procesos de Planificación | Grupo de Procesos de Ejecución | Grupo de Procesos de Monitoreo y Control | Grupo de Procesos de Cierre |
| 4. Gestión de la Integración del Proyecto | 4.1 Desarrollar el Acta de Constitución del Proyecto | 4.2 Desarrollar el Plan para la Dirección del Proyecto | 4.3 Dirigir y Gestionar el Trabajo del Proyecto<br>4.4 Gestionar el Conocimiento del Proyecto | 4.5 Monitorear y controlar el Trabajo del Proyecto<br>4.6 Realizar el Control Integrado de Cambios | 4.7 Cerrar el Proyecto o Fase |
| 5. Gestión del Alcance del Proyecto | | 5.1 Planificar la Gestión del Alcance<br>5.2 Recopilar Requisitos<br>5.3 Definir el Alcance<br>5.4 Crear el ETD/WRS | | 5.5 Validar el Alcance<br>5.6 Controlar el Alcance | |
| 6. Gestión del Cronograma del Proyecto | | 6.1 Planificar la Gestión del Cronograma<br>6.2 Definir las Actividades<br>6.3 Secuenciar las Actividades<br>6.4 Estimar la Duración de las Actividades<br>6.5 Desarrollar el Cronograma | | 6.6 Controlar el Cronograma | |
| 7. Gestión de los Costos del Proyecto | | 7.1 Planificar la Gestión de los Costos<br>7.2 Estimar los Costos<br>7.3 Determinar el Presupuesto | | 7.4 Controlar los Costos | |
| 8. Gestión de la Calidad del Proyecto | | 8.1 Planificar la Gestión de la Calidad | 8.2 Gestionar la Calidad | 8.3 Controlar la Calidad | |
| 9. Gestión de los Recursos del Proyecto | | 9.1 Planificar la Gestión de los Recursos<br>9.2 Estimar los Recursos de las Actividades | 9.3 Adquirir Recursos<br>9.4 Desarrollar el Equipo<br>9.5 Dirigir al Equipo | 9.6 Controlar los Recursos | |
| 10. Gestión de las Comunicaciones del Proyecto | | 10.1 Planificar la Gestión de las Comunicaciones | 10.2 Gestionar las Comunicaciones | 10.3 Monitorear las Comunicaciones | |
| 11. Gestión de los Riesgos del Proyecto | | 11.1 Planificar la Gestión de los Riesgos<br>11.2 Planificar los Riesgos<br>11.3 Realizar el Análisis Cualitativo de Riesgos<br>11.4 Realizar el Análisis Cuantitativo de Riesgos<br>11.5 Planificar la Respuesta a los Riesgos | 11.6 Implementar la Respuesta a los Riesgos | 11.7 Monitorear los Riesgos | |
| 12. Gestión de las Adquisiciones del Proyecto | | 12.1 Planificar la Gestión de las Adquisiciones | 12.2 Efectuar las Adquisiciones | 12.3 Controlar la Adquisiciones | |
| 13. Gestión de los Interesados del Proyecto | 13.1 Identificar a los Interesados | 13.2 Planificar el Involucramiento de los Interesados | 13.3 Gestionar la Participación de los Interesados | 13.4 Monitorear el Involucramiento de los Interesados | |

## 1.3 Estándar ISO/IEC 29110 (perfil básico)

La familia ISO/IEC 29110 es una serie de normas internacionales que describen los perfiles del ciclo de vida del desarrollo de software orientado a "**Pequeñas Organizaciones"** (PO). Una PO es una empresa, organización, departamento o proyecto que tiene menos de 25 personas. La mayoría de las PO's de software pertenecen a esta categoría. De acuerdo con la Organización para Cooperación Económica y Desarrollo (OCED), las medianas y pequeñas empresas constituyen el tipo de organización de negocio dominante en el mundo. (NTP-RT-ISO/IEC TR 29110-5-1-2, 2012)

ISO/IEC 29110 se basa en la Norma Mexicana NMX-I-059-NYCE-2011 (MoProSoft), en la ISO/IEC 12207, la ISO/IEC 15289, la ISO/IEC 15504 entre otras. Sus objetivos son mejorar la calidad del producto y/o servicio de software y también el desempeño de la organización.

### Partes de ISO/IEC 29110 y audiencias

Se divide en 5 partes de acuerdo con el tipo de audiencia a la que está dirigida tal como lo muestra la tabla 1.3

Tabla 2.4 ISO/IEC 29110. Fuente: (NTP-RT-ISO/IEC TR 29110-5-1-2, 2012, pág. XI)

| ISO/IEC 29110 | Titulo | Audiencia Objetivo |
|---|---|---|
| **Parte1 IEC 29110-1** | Visión general | PO, Evaluadores, productores de estándares, vendedores de herramientas y metodologías |
| **Parte 2 ISO/IEC 29110-2** | Marco de trabajo y taxonomía | Productores de estándares, vendedores de herramientas y metodologías, No dirigido a PO |
| **Parte 3 ISO/IEC 29110-3** | Guía de evaluación | Evaluadores y PO |
| **Parte 4 ISO/IEC 29110-4-m** | Especificaciones de perfil | Productores de estándares, vendedores de herramientas y metodologías, no dirigido a las PO |
| **Parte 5 ISO/IEC 29110-5-m-n** | Guía de Gestión e Ingeniería | PO |

Cada parte tiene características específicas que se muestran en la tabla 1.3

Tabla 1.3 Características de cada parte del ISO/IEC 29110 . (De creación propia)

| ISO/IEC 29110 | Descripción |
|---|---|
| **Parte1**<br>**ISO/IEC 29110-1** | Términos que serán usados a lo largo de los documentos, que comprende el estándar ISO/IEC 29110.<br>Características de una PO<br>Beneficios potenciales que reciben gracias al uso del estándar<br>Conceptos de ciclo de vida<br>Mejora de procesos y estandarización |
| **Parte 2**<br>**ISO/IEC 29110-2** | Conceptos para los Perfiles de Ingeniería de Software estandarizados para PO's<br>Lógica de definición de los perfiles y su aplicación<br>Los elementos en común de los Perfiles<br>Taxonomía del estándar |
| **Parte 3**<br>**ISO/IEC 29110-3** | Guías de asesoramiento de procesos y cumplimiento de requisitos necesarios para satisfacer a un Perfil |
| **Parte 4**<br>**ISO/IEC 29110-4-m** | Proporciona la especificación para todos los perfiles en Grupos de Perfiles |
| **Parte 5**<br>**ISO/IEC 29110-5-m-n** | Proporciona una guía de implementación e ingeniería para el Perfil descrito en la Parte 4 |

### Perfiles del grupo genérico

En el perfil ISO/IEC 29110-4-1 provee la especificación para los perfiles del grupo genérico, el cual consta de 4 perfiles que son: entrada, básico, intermedio y avanzado. Cada perfil proporciona un conjunto de procesos, actividades, tareas y productos de trabajo. (Tabla 1.5)

De tal forma que la guía de implementación de ingeniería para el grupo genérico en el perfil de entrada queda como ISO/IEC 29110-5-1-1. Para el perfil básico queda como ISO/IEC 29110-5-1-2

Tabla 1.5 perfiles de ISO/IEC29110 (de creación propia)

| Perfil | Características |
|---|---|
| **Entrada** | Aplica a PO, trabando en pequeños proyectos, (Organizaciones que iniciaron sus operaciones hace menos de 3 años y trabajan en proyectos pequeños menores a 6 personas-mes). |
| **Básico** | Describe PO en el desarrollo de software de una sola aplicación por un solo equipo de proyecto, sin ningún riesgo especial. |
| **Intermedio** | Describe PO en el desarrollo de software en múltiples proyectos con más de un equipo de proyecto |
| **Avanzado** | Para PO que pretenden crecer y ser más competitivas en el negocio de desarrollo de software |

### Perfil básico del grupo genérico

Éste trabajo se enfoca en la implementación de ingeniería del perfil básico del grupo genérico; ISO/IEC 29110-5-1-2. Para el uso de la ISO/IEC 29110 la PO necesita cumplir con las siguientes condiciones de entrada. (NTP-RT-ISO/IEC TR 29110-5-1-2, 2012, pág. 6)

- El enunciado de Trabajo del Proyecto está documentado.
- La viabilidad del proyecto fue realizada antes de su inicio.
- El equipo del proyecto, incluyendo el Gestor del Proyecto, está asignado y entrenado; y los bienes, servicios e infraestructura para iniciar el proyecto están disponibles.

#### *Procesos del perfil básico del grupo genérico*

Este perfil consta de dos procesos: **Gestión de Proyectos** e **Implementación de Software** tal como se muestra el diagrama de la figura 1.8.

Los rectángulos grandes redondeados indican procesos, los rectángulos pequeños sin las esquinas redondeadas indican productos.

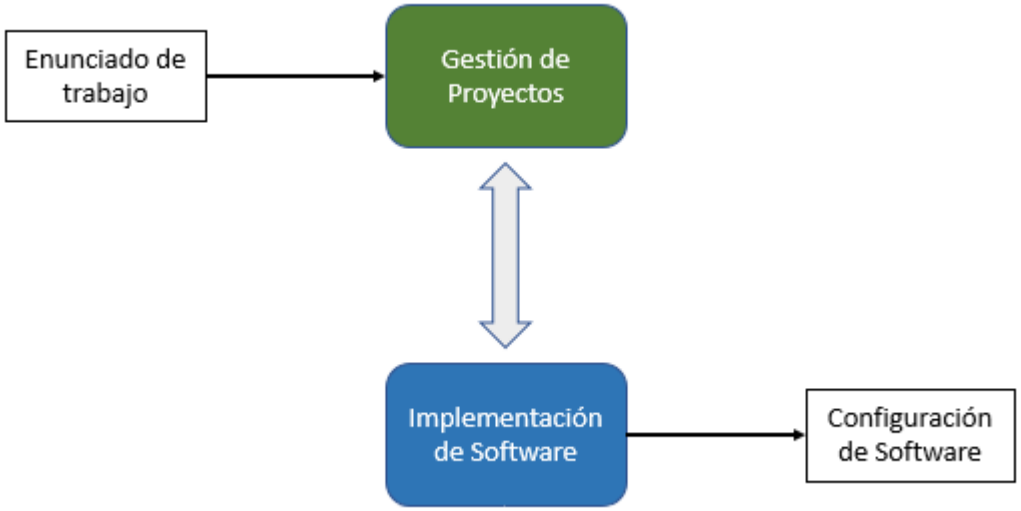



Figura 1.8 Procesos del Perfil Básico ISO/IEC 29110-5-2-1. Fuente: (NTP-RT-ISO/IEC TR 29110-5-1-2, 2012, pág. 7)

El propósito de cada proceso se muestra en la tabla 1.6.

Tabla 1.6 Procesos del Perfil Básico del ISO/IEC29110 (de creación propia)

| **Gestión de Proyectos** | **Implementación de Software** |
|---|---|
| El propósito del proceso Gestión del Proyecto es establecer y llevar a cabo de manera sistemática las *Tareas* de un proyecto de implementación de *Software*, que permitan cumplir con los Objetivos del proyecto en calidad, tiempo y costos esperados. (NTP-RT-ISO/IEC TR 29110-5-1-2, 2012, pág. 7). | "El propósito del proceso de Implementación de Software es la realización sistemática de las actividades de análisis, diseño, construcción, integración y pruebas para los productos Software, nuevos o modificados, de acuerdo a los requisitos especificados" (NTP-RT-ISO/IEC TR 29110-5-1-2, 2012, pág. 7). |

Esta guía fue creada para ser utilizada por las PO, independientemente del enfoque o metodología de desarrollo que se use: ágil. evolutivo, dirigido por pruebas.

**Objetivos de la Gestión de Proyectos**

- **GP. 01**. El Plan del Proyecto para la ejecución del proyecto es desarrollado de acuerdo al Enunciado de Trabajo y revisado y aceptado por el Cliente. Las Tareas y los Recursos necesarios para completar el trabajo son dimensionados y estimados.

- **GP. 02** El avance del proyecto es monitoreado contra el Plan del Proyecto y registrados en el Registro de Estado del Avance. Las correcciones para resolver los problemas y desviaciones respecto del plan son realizadas cuando los objetivos del proyecto no son logrados. El cierre del proyecto es ejecutado para conseguir la aceptación documentada del Cliente en el Documento de Aceptación.

- **GP. 03** Las Solicitudes de Cambio son atendidas mediante su recepción y análisis. Los cambios a los requisitos de Software son evaluados por su impacto técnico, en costo y en el cronograma.

- **GP. 04** Reuniones de revisión con el Equipo de Trabajo y el Cliente son realizadas. Los acuerdos que surgen de estas reuniones son documentados y se les hace seguimiento.

- **GP.05** Los riesgos son identificados en el desarrollo y durante la realización del proyecto.

- **GP. 06** Una Estrategia de Control de Versiones de Software es desarrollada. Los elementos de Configuración del Software son identificados, definidos e incorporados a la línea base. Las modificaciones y actualizaciones de los elementos son controladas y puestas a disposición del Cliente y del Equipo de Trabajo. El almacenamiento, la manipulación y la entrega de los elementos son controlados.

- **GP.07** El Aseguramiento de Calidad del Software es realizado para proporcionar garantía de que los productos y procesos de trabajo cumplen con el Plan del Proyecto y Especificación de Requisitos.

*Actividades de la gestión de proyectos*

1. Planificación del Proyecto
2. Ejecución del Plan del Proyecto
3. Evaluación y Control del Proyecto
4. Cierre del Proyecto

   (ver figura 1.9)

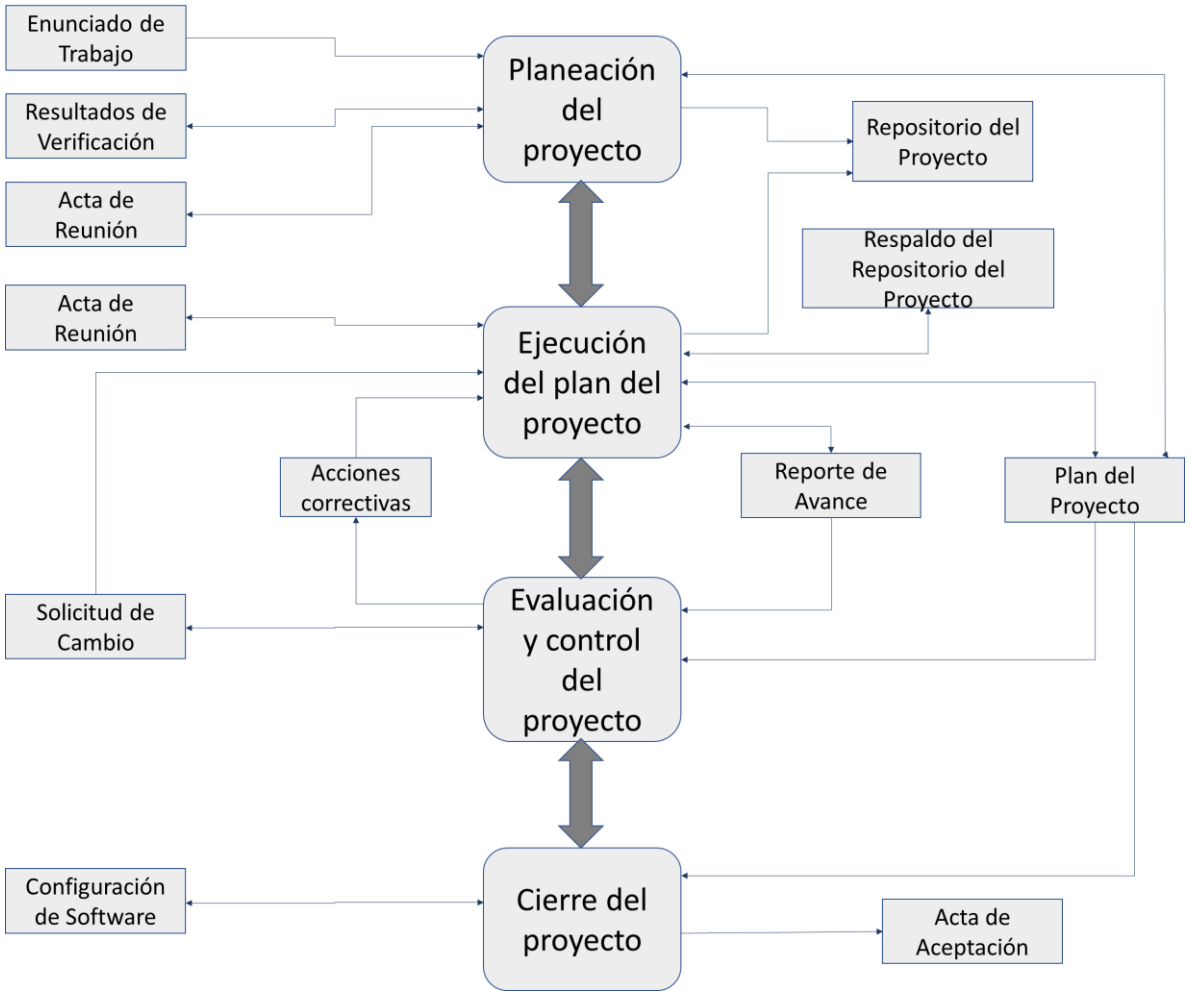



Figura 1.9 Diagrama del proceso de GP según la ISO/IEC 29110-5-1-2. Fuente**:** (NTP-RT-ISO/IEC TR 29110-5-1-2, 2012, pág. 15)**.**

**Objetivos del proceso de Implementación de Software**

- **IS.O1**. Las Tareas de las actividades son realizadas a través del cumplimiento del Plan del Proyecto actual.

- **IS.O2**. Los requisitos del Software son definidos, analizados para su correctitud y testabilidad, aprobados por el Cliente, incorporados a la línea base y comunicados.

- **IS.O3**. La arquitectura y diseño detallado del Software son desarrollados e incorporados a la línea base. Aquí se describen los Componentes de Software y sus interfaces internas y externas. La consistencia y trazabilidad de los requisitos de Software son establecidos.

- **IS.O4**. El Componente de Software definido por el diseño son producidos. Las pruebas unitarias son definidas y ejecutadas para verificar la consistencia de los requisitos y el diseño. La trazabilidad de los requisitos y el diseño son establecidas.

- **IS.O5**. El Software es producido ejecutando la integración de los Componente de Software y es verificado usando los Casos de Prueba y Procedimientos de Prueba. Los resultados son registrados en el Reporte de Pruebas. Los defectos son corregidos y la consistencia y trazabilidad hacia el Diseño de Software son establecidos.

- **IS.O6**. La Configuración de Software, que cumpla con la Especificación de Requisitos según lo acordado con el Cliente, que incluye la documentación de usuario, operación y mantenimiento es integrada, incorporada a la línea base y almacenada en el Repositorio del Proyecto. Las necesidades de cambios para la Configuración de Software son detectadas y las solicitudes de cambio relacionadas son iniciadas.

- **IS.O7**. Las Tareas de verificación y validación de todos los productos de trabajo requeridos son realizadas utilizando los criterios definidos para lograr la coherencia entre los productos de entrada y salida en cada actividad. Los defectos son identificados y corregidos; los registros son almacenados en los Resultados de Verificación / Validación

### *Actividades del proceso de implementación de software*

1. Inicio de la Implementación de Software
2. Análisis de Requisitos de Software
3. Arquitectura y Diseño Detallado del Software
4. Construcción de Software
5. Integración y Pruebas del Software
6. Entrega del producto.

(ver figura 1.10)

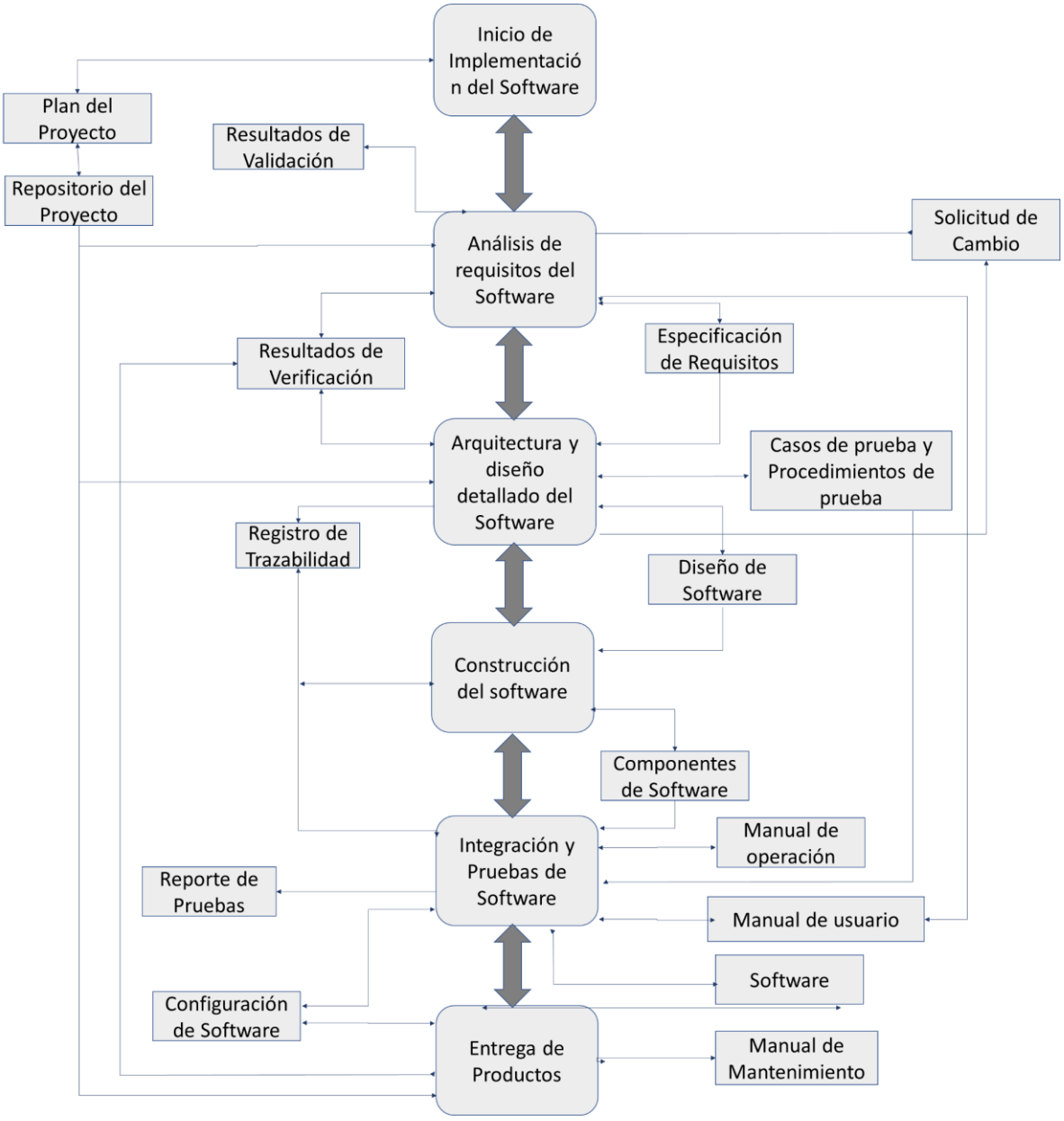



Figura 1.0 Diagrama del proceso de IS según la ISO/IEC 29110-5-1-2. Fuente**:** (NTP-RT-ISO/IEC TR 29110-5-1-2, 2012, pág. 35)**.**

## 1.4 Manifiesto ágil

De acuerdo al informe anual VersionOne 2018, la encuesta número 12 del estado de los Marcos de Desarrollo Ágil, encontró que las organizaciones se están dando cuenta de las razones y beneficios que pueden lograr al adoptar ágil (Collabnet VersionOne, 2018). Ver figura. 1.11, de acuerdo a ésta encuesta, se observa que para proyectos en los que se requiere entregas rápidas y mayor capacidad para adaptarse a los cambios, los marcos de desarrollo ágil son una buena opción.

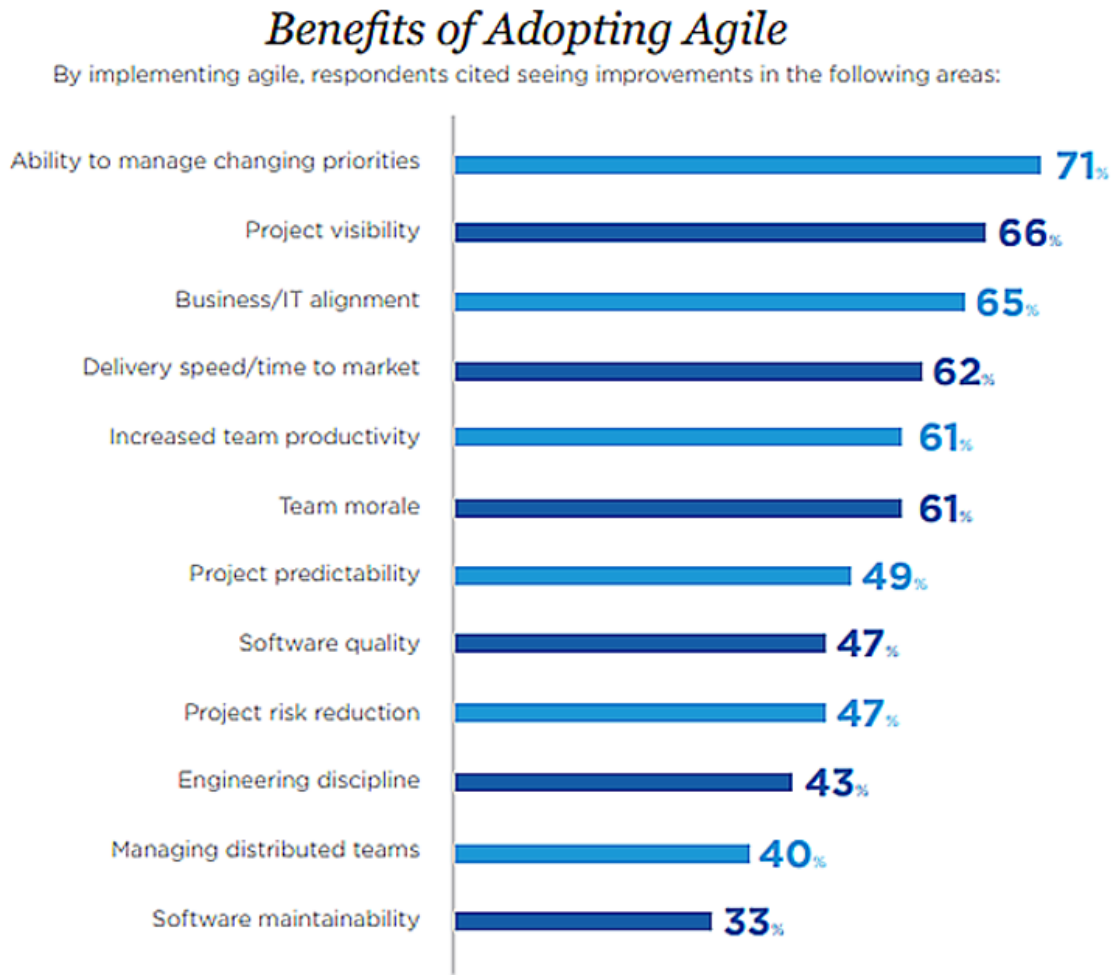



Figura 1.11. Beneficios de adoptar Ágil, fuente: Informe VersionOne 2018 VersionOne -12th annual state of agile report: https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report

**Surgimiento del manifiesto ágil**

En marzo de 2001, se reunieron 17 expertos en Salt Lake City, para tratar temas relacionados con técnicas y procesos para desarrollar software, alternativas a las tradicionales, las cuales se consideraban excesivamente "pesadas" y rígidas, dependientes en gran medida de la etapa de planeación; de esa forma surge el término de métodos ágiles, cuyo principal propósito es la obtención rápida de resultados y la satisfacción de cliente mediante retroalimentación, lo que además permite la detección y corrección temprana de errores. (Manifiesto por el Desarrollo Ágil de Software, 2019)

De esta reunión se resumieron 4 valores (figura. 1.12) y 12 principios (figura. 1.13) para los métodos alternativos a lo que se denominó como Manifiesto ágil.

**Valores definidos en el Manifiesto del Desarrollo Ágil**

Del lado izquierdo de la Figura 1.12, se muestran las características más representativas del paradigma ágil y de la derecha se muestras las características que definen las metodologías tradicionales, aun cuando todas las características forman parte de las del paradigma ágil, las de la izquierda tienen preferencia sobre las de la derecha.

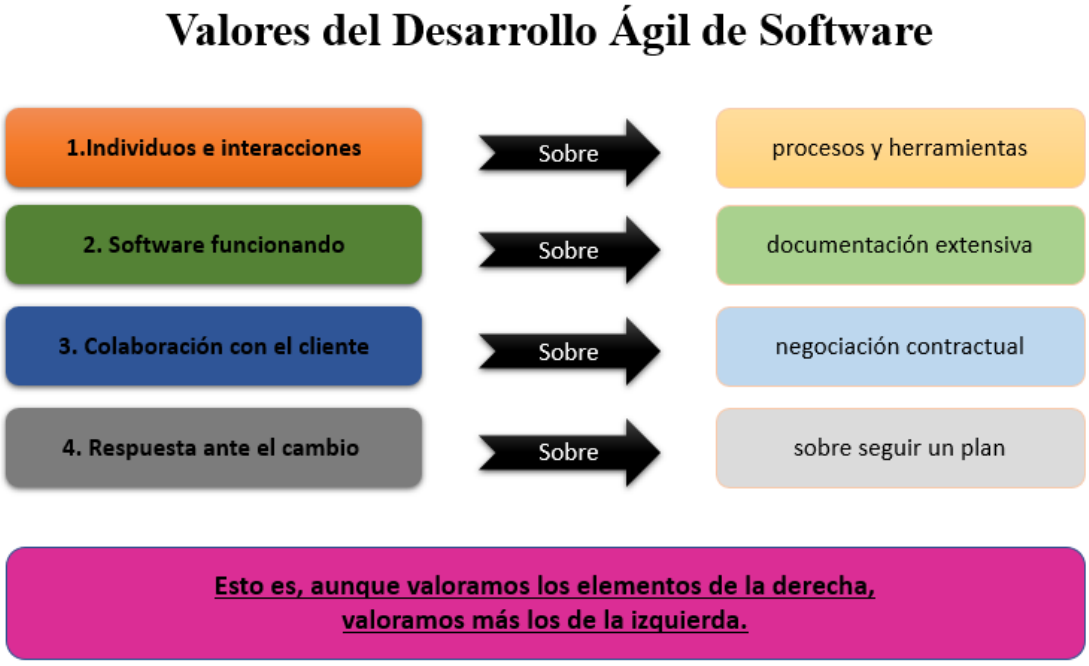



Figura 1.12 Valores del desarrollo Ágil. (de creación propia)

*Valores que define el Manifiesto del Desarrollo Ágil*

1. **Los individuos e interacciones por sobre los procesos y las herramientas**: las metodologías ágiles consideran que el valor principal para el éxito en el desarrollo de software, son las personas. Contar con un equipo de trabajo calificado con capacidades técnicas adecuadas, que se adapte al cambio y con la habilidad para trabajar en equipo es mejor que contar con herramientas y procesos rigurosos.

2. **Software funcionando por encima de la documentación**: las metodologías ágiles proponen generar solamente aquella documentación estrictamente necesaria, los documentos deben ser cortos, aunque completos y actualizados, dando prioridad al contenido sobre la presentación.

3. **La colaboración del cliente por encima de la negociación del contrato**: Existen muchos proyectos en los cuales no se pueden planear el proyecto completamente desde el principio, ya sea por su complejidad o porque aún no se puede determinar exactamente lo que el usuario necesita. En las metodologías agiles, el cliente o usuario se convierte en un miembro del equipo de trabajo, que colabora para ir definiendo los requerimientos en el transcurso del desarrollo; por lo tanto, no se considera necesario invertir tiempo en las negociaciones para establecer desde el principio cuáles serán las expectativas del proyecto que se deberán entregar en el producto terminado.

4. **La respuesta al cambio por encima del seguimiento de un plan**: El desarrollo de software frecuentemente se enfrenta a cambios durante su ejecución, por lo tanto, en las metodologías ágiles la planeación debe ser flexible. La estrategia es hacer iteraciones de pocas semanas, de tal forma que los cambios se puedan ir adaptando para las siguientes iteraciones.

**Principios que establece el Manifiesto del Desarrollo Ágil**

1. Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.

2. Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se favorecen al cambio como ventaja competitiva para el cliente.

3. Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en periodos breves.

4. Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.

5. Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen sus tareas.

6. La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.

7. El software que funciona es la principal medida del progreso.

8. Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.

9. La atención continua a la excelencia técnica enaltece la agilidad.

10. La simplicidad como arte de maximizar la cantidad de trabajo que se hace, es esencial.

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos que se autoorganizan

12. En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia. Ver figura 1.13


*Figura 1-13.* Los principios del paradigma de ágiles, que constituyen las características que fundamentales que lo diferencian de las metodologías tradicionales. (de creación propia)

Los marcos de desarrollo ágil no son anti procesos, sino que buscan adecuar los procesos a las necesidades actuales de equipos y proyectos. (Ramírez Liliana del Carmen, 2014)

### Marco de desarrollo Ágil Scrum

Scrum es el marco de desarrollo ágil más usado a nivel mundial, según la encuesta 2018 de VersiónOne (Collabnet VersionOne, 2018). Es iterativo, rápido, flexible y eficaz, diseñado para ofrecer un valor de forma rápida a lo largo del proyecto. Scrum garantiza transparencia en la comunicación y crea un ambiente de responsabilidad colectiva y de progreso continuo.

Una fortaleza clave de Scrum radica en el uso de equipos interfuncionales (cross-functional), autoorganizados y empoderados que dividen su trabajo en ciclos de trabajo cortos y concentrados llamadas Iteraciones. Observe la figura. 1.14

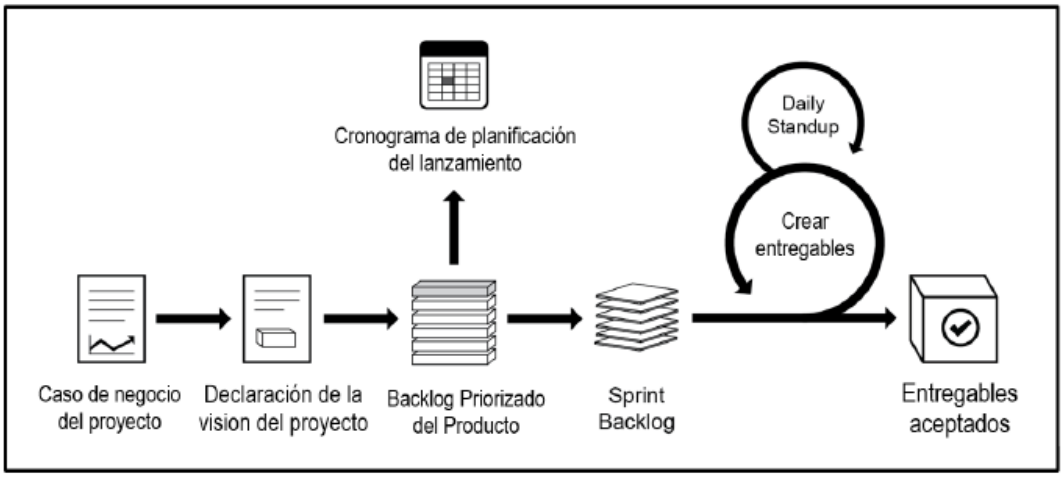



Fig. 2.14. Flujo de Scrum para una Iteracion (Iteración) Fuente:( SBOK :A Una guía para el Cuerpo de Conocimiento de Scrum (Guía SBOK™) – 3ra Edición (2017) pág. 2)

**Roles principales de Scrum**

**Dueño del producto (Product Owner)** es la persona responsable de lograr el máximo valor empresarial para el proyecto. Este rol también es responsable de la articulación de requisitos del cliente y de mantener la visión del negocio en el proyecto. El Dueño de Producto representa la voz del cliente.

**Scrum Master** es un facilitador que asegura que el Equipo Scrum cuente con un ambiente propicio para completar el proyecto con éxito. El Scrum Master guía, facilita y enseña las prácticas de Scrum a todos los involucrados en el proyecto; elimina los impedimentos que pueda tener el equipo y se asegura de que se estén siguiendo los procesos de Scrum.

**El Equipo Scrum** es el grupo o equipo de personas responsables de entender los requisitos especificados por el Dueño del producto y de crear los entregables del proyecto.

**Interesados** (Involucrados) es un término colectivo que incluye a clientes, usuarios y patrocinadores, que generalmente interactúan con el Dueño del producto, el Scrum Master y el Equipo Scrum para proporcionarles las entradas y facilitar la creación del producto del proyecto.

**El ciclo de Scrum**

Empieza con una reunión de Interesados, durante la cual se crea la visión del proyecto. Después, el Dueño del producto desarrolla un Backlog Priorizado del Producto (Prioritized Product Backlog), que contiene una lista de requerimientos del negocio y del proyecto por orden de importancia en forma de historias de usuario. Cada Iteración empieza con una reunión de Planificación de la Iteración (Iteración Planning Meeting) durante la cual se consideran las Historias de Usuario de alta prioridad para su inclusión en la Iteración.

Una Iteración generalmente tiene una duración de una a seis semanas durante las cuales el Equipo Scrum trabaja en la creación de entregables en incrementos del producto. Durante la Iteración, se llevan cabo una Reunión diaria (Daily Standups), donde los miembros del equipo discuten el progreso diario. Hacia el final de la Iteración, se lleva a cabo una Reunión de Revisión de la Iteración (Iteración Review Meeting) en la cual se proporciona una demostración de los entregables al Dueño del producto y a los Interesados relevantes. El Dueño del producto acepta los entregables sólo si cumplen con los criterios de aceptación predefinidos.

El ciclo de la iteración termina con una Reunión de Retrospectiva de la Iteración (Retrospect Iteración Meeting), donde el equipo analiza las formas de mejorar los procesos y el rendimiento a medida que avanzan a la siguiente iteración. (SCRUMstudy, 2016)

# Capítulo 2. Definición del método "SOFUNI"

### Motivación para el método

Cada día la industria de software se vuelve más exigente, lo que conlleva a la necesidad de crear productos de software más complejos en menos tiempo, con especificaciones de calidad más estrictas, como consecuencia la academia debe proporcionar a los futuros profesionistas los conocimientos y las herramientas de Ingeniería de Software, preparándolos para cumplir con las expectativas y necesidades de la industria de software. Cabe señalar que, aunque los alumnos no son profesionales de software suelen tener la necesidad de desarrollar productos de software acordes a su área.

### Objetivo del Método SOFUNI (Software Universitario)

Proporcionar una base conceptual referente a Ingeniería de Software mediante una serie de prácticas para administrar proyectos de software tomando como base el estándar ISO/IEC29110 en su Perfil Básico de tal forma que guie a los estudiantes de nivel universitario que trabajan de manera individual o en equipos pequeños a crear productos de software de calidad en un tiempo menor a un semestre, considerando una metodología iterativa.

El perfil básico del estándar ISO/IEC29110 proporciona dos procesos como se muestra en la figura 2.1, que se retoma del capítulo 1.

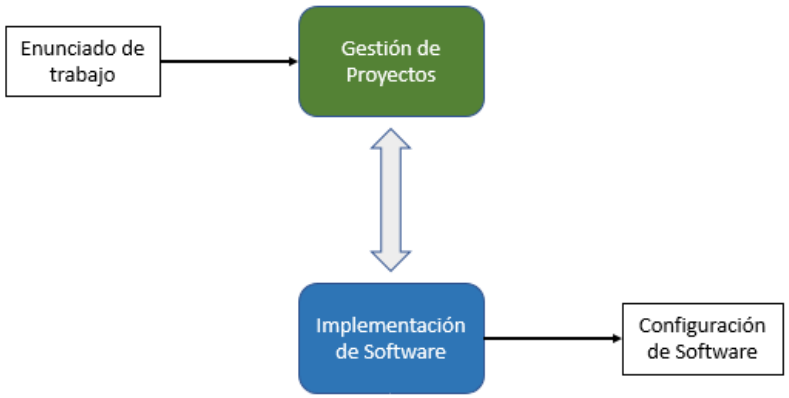



Figura 2.1 Procesos del Perfil Básico ISO/IEC 29110-5-2-1. Fuente: (NTP-RT-ISO/IEC TR 29110-5-1-2, 2012, pág. 7)

Cada proceso está formado por actividades como se muestra en la figura 2.2.


Figura 2.2 Procesos y actividades del Perfil Básico ISO/IEC 29110-5-2-1. (de creación propia)

Las actividades de ambos procesos se traslapan, como lo muestra la figura 2.3. Los rectángulos representan las actividades del proceso de gestión de proyectos y los óvalos representan las actividades del proceso de implementación de software.

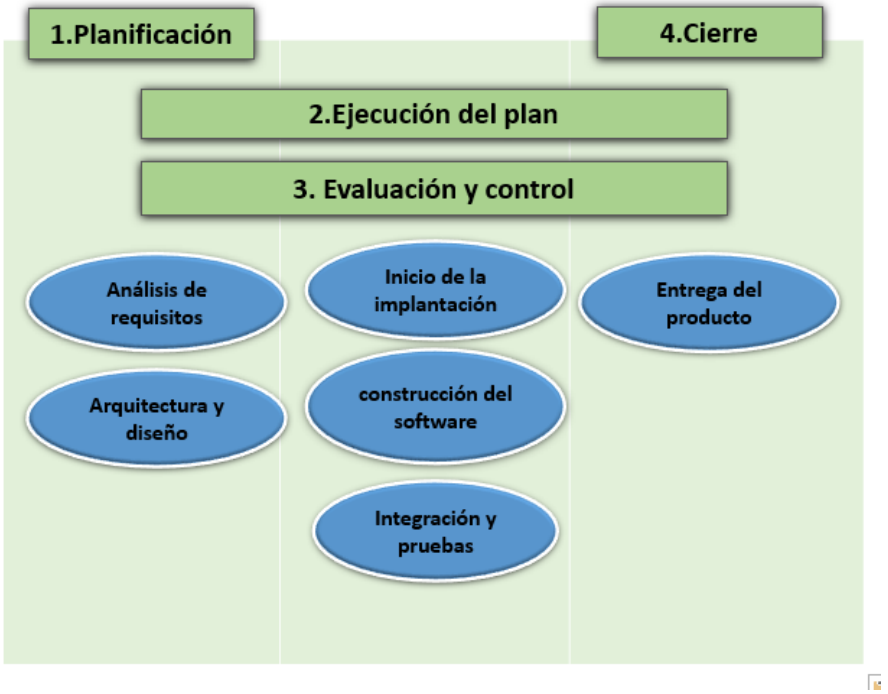



Figura 2.3 Las actividades de los procesos del Perfil Básico ISO/IEC 29110-5-2-1. (de creación propia)

Para el Método SOFUNI se proponen prácticas para llevar a cabo los dos procesos del Perfil Básico de ISO/IEC 29110 (figura 2.4)

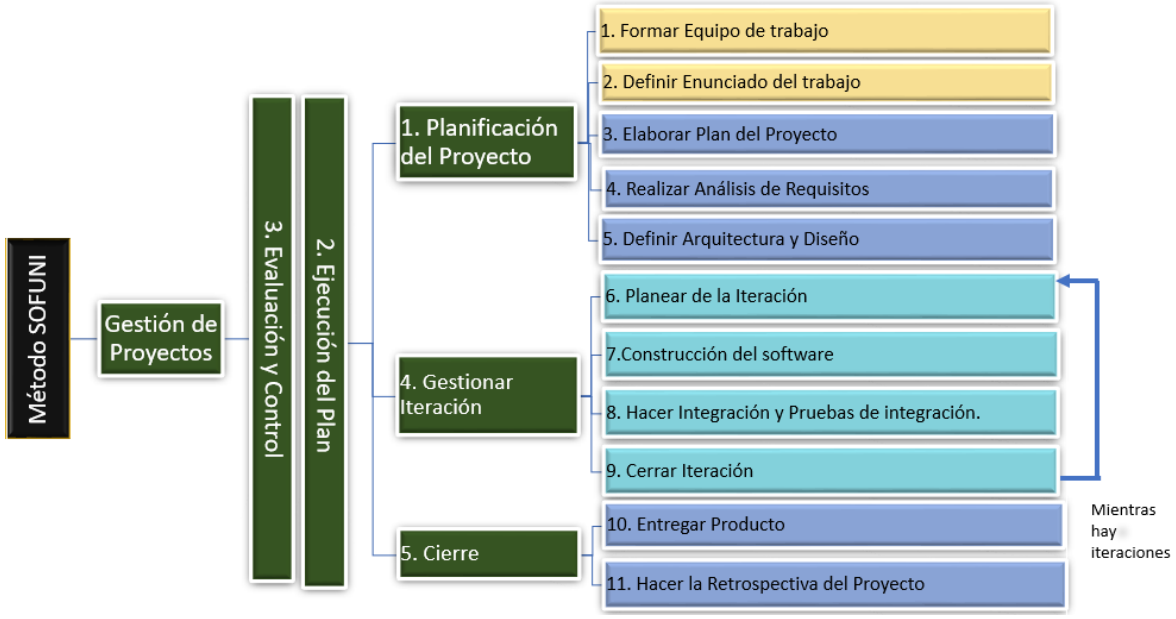



Figura 2. 4. Prácticas del Método SOFUNI (de creación propia).

Las prácticas del método se organizan de acuerdo a las actividades del proceso de Gestión de Proyectos del estándar ISO/IEC 29110 en su Perfil Básico. En el caso de las actividades correspondiente a Evaluación y Control y Ejecución del Plan, se integran en cada práctica mediante una serie de tareas.

La Práctica de Enunciado del trabajo, no tiene una correspondencia directa con las actividades del proceso de Gestión de Proyectos o de Implementación de Software, ya que como se puede observar en la figura 2.1, *El Enunciado del trabajo* es la entrada para el proceso de Gestión de Proyectos, sin embargo; se requiere que los estudiantes escriban el Enunciado del trabajo, de forma adecuada, a partir de una descripción básica del producto que deberán desarrollar.

A partir del planteamiento anterior se define el Método SOFUNI, tal como se muestra en la figura 2.5

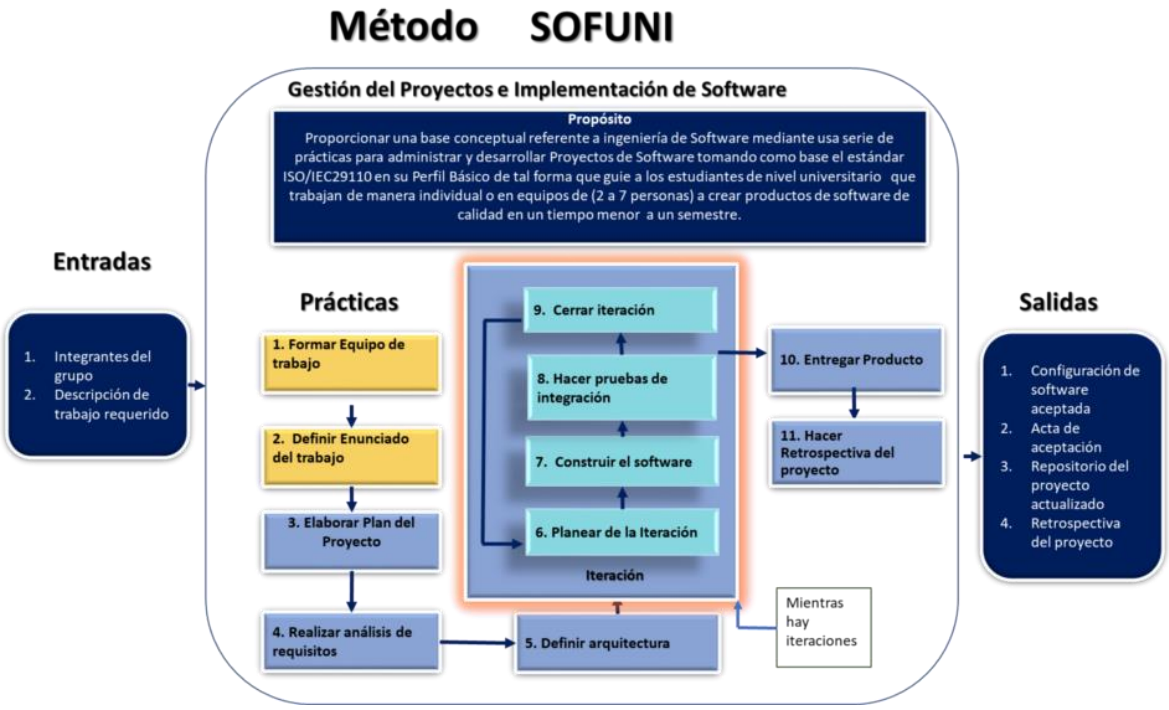



Figura 2.5 El Método SOFUNI (de creación propia).

El Método SOFUNI consta de 11 prácticas organizadas de forma que la salida de cada una es la entrada de la siguiente, proporcionando una guía para la Gestión de Proyectos en el Desarrollo de Software. El método se basa en un ciclo de vida iterativo; por lo tanto, la Iteración se repite mientras sea necesario hasta completar todas las tareas de desarrollo del producto de software.

**Elementos que integran cada Práctica del Método SOFUNI**

1. Diagrama
2. Plantilla genérica
3. Herramientas
4. Base de conocimientos

### *Diagrama*

Cada práctica del Método SUFUNI, se integra de un diagrama, en el que se visualiza una o varias Entradas, una o varias Salidas, un Objetivo y un conjunto de Actividades que, a su vez, se dividen en Tareas.

- Las **Entradas** se refieren a los insumos que se requieren para llevar a cabo cada práctica.
- Las **Salidas** son productos generados al término de la práctica.
- El **Objetivo** define la razón de ser cada práctica, lo que se debe lograr con su realización.
- Las **Actividades** se refieren a lo que los integrantes del equipo deben realizar para convertir las Entradas en Salidas, de acuerdo a un Objetivo.
- En ocasiones las Actividades requieren ser divididas en actividades simples llamadas Tareas

### *Plantilla genérica*

La plantilla genérica muestra las actividades y los elementos que en cada actividad deben ser documentados, no se propone un formato, ya que el equipo de trabajo deberá proponer su propio formato.

### *Herramientas recomendadas*

Las herramientas pueden ser documentos, aplicaciones, repositorios, técnicas, etc. sugeridos para llevar a cabo cada práctica, en la tabla correspondiente se describe el propósito de cada herramienta.

### *Base Conceptual*

Son los conceptos básicos que los integrantes del equipo deben poseer para llevar a cabo cada práctica.

# Capítulo 3. Definición de las prácticas

Cada una de las prácticas consta un objetivo, entrada(s), actividades y tareas cuya secuencia lleva a la generación de una o varias salidas. Para cada práctica se muestra un esquema, una plantilla genérica y los conocimientos y herramientas sugeridas para realizar las actividades.

## 3.1 Práctica 1. Formar el equipo de trabajo

Durante la práctica se lleva a cabo la integración del equipo de trabajo, para formar una identidad de manera, que cada miembro se sienta parte de él y responsable por el trabajo de todos. En esta práctica también se identifican las habilidades e interés de cada miembro, y se definen los roles que cada uno puede desempeñar, lo que facilita la autorregulación del equipo. Cabe señalar que una persona puede tener varios roles y estos pueden cambiar durante el proyecto, en algún momento todos los integrantes del equipo deben realizar el rol de desarrollador. La estructura de la práctica 1 se muestra en la Figura 3.1, la plantilla genérica se puede observar en la tabla 3.1 y la herramienta sugerida en la tabla 3.2

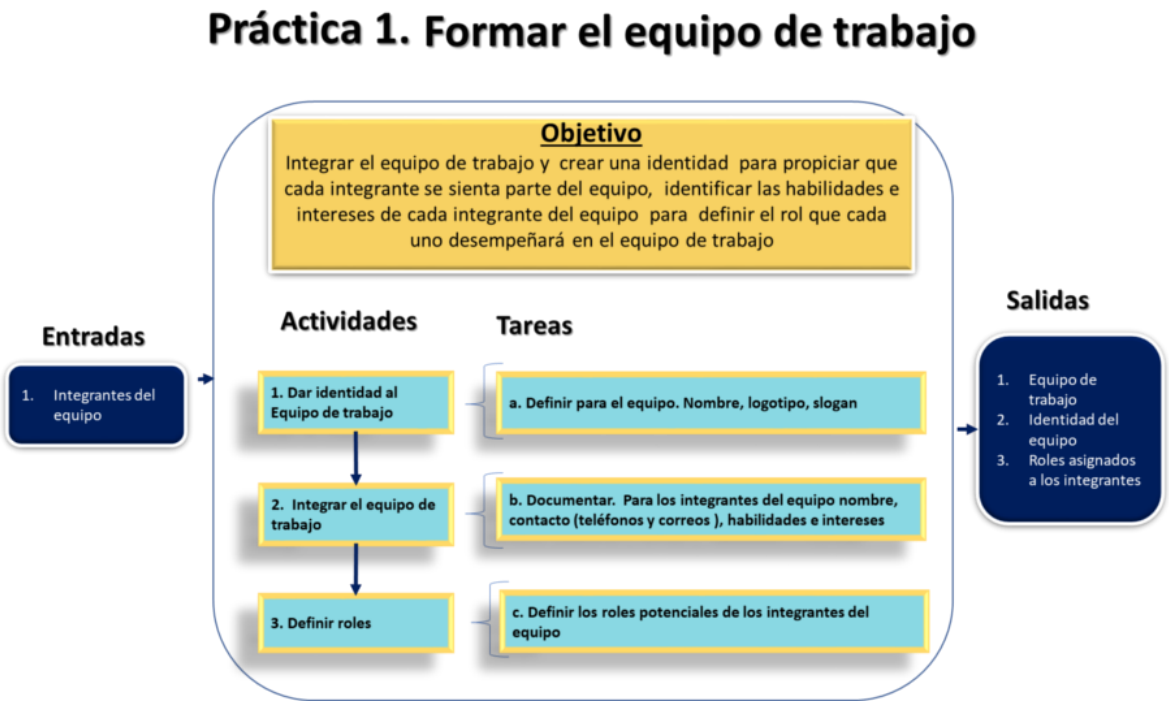



Figura 3.1 Práctica 1: Formar el equipo de trabajo (de creación propia).

## Plantilla genérica. Práctica 1. "Formar el equipo de trabajo"

Tabla 3.1 Plantilla genérica de Práctica 1 (de creación propia)

| Plantilla de Práctica No. 1 "Formar el equipo de trabajo" | |
|---|---|
| Actividad | Elementos o subactividades |
| 1. Dar identidad al equipo de trabajo | Definir para el equipo<br>a) Nombre<br>a) Logotipo<br>b) Slogan |
| 2. Integrar el equipo de trabajo | Documentar para cada integrante del equipo<br>e) nombre<br>f) Contacto<br>i. Teléfono<br>ii. Correo electrónico<br>g) Habilidades<br>h) Intereses |
| 3. Definir roles | i) Definir el rol que cada integrante del equipo desempeñará en el marco Scrum<br>i) Equipo: Desarrollador, Responsable de documentos, Responsable técnico, y responsable de seguimiento.<br>* Los roles pueden modificarse posteriormente; cada integrante tendrá más de un rol a lo largo del proyecto, y todos los integrantes deben desarrollar en algún momento. |

## Herramientas recomendadas. Práctica 1. "Formar el equipo de trabajo"

Tabla 3.2 Herramientas sugeridas para la práctica 1 (de creación propia)

| **Herramientas Práctica No. 1 "Formar el equipo de trabajo"** | |
|---|---|
| Herramientas recomendadas | Propósito |
| 1. Management 3.0 Mapas personales. | Proporciona conocimientos y herramientas para formar equipos de alto rendimiento, busca involucrar a todos los trabajadores en la consecución de los objetivos de la empresa, de forma que la responsabilidad se comparte entre todos los integrantes de la empresa. (Jürgen A, 2010) |

## Base conceptual. Práctica 1. “Formar el equipo de trabajo”

*Management 3.0*

Es muy importante favorecer un ambiente de trabajo proactivo entre los equipos de trabajo, por lo que se recomienda tomar en cuenta Management 3.0 en la creación y autorregulación de los equipos los seis puntos que propone Jurgen Appelo, el pionero del Management 3.0 en su libro del mismo nombre, en que hace referencia a los siguientes principios:

1. **Energizar personas**: las personas son la parte más importante de una organización y los gestores deben hacer todo lo posible para mantener a las personas activas, creativas y motivadas.
2. **Empoderar a los equipos**: los equipos pueden autoorganizarse, y esto requiere empoderamiento, autorización y confianza desde la gestión.
3. **Alinear restricciones**: la autoorganización puede conducir a cualquier cosa, y por tanto, es necesario proteger a las personas y a los recursos compartidos, y brindar a las personas un propósito claro y unos objetivos definidos.
4. **Desarrollar competencias**: los equipos no pueden lograr sus objetivos si sus miembros no tienen las capacidades para ello, por tanto, los gestores deben contribuir al desarrollo de sus competencias.
5. **Aumentar la estructura**: muchos equipos operan dentro del contexto de una organización compleja. Por ello es importante considerar estructuras que mejoren la comunicación.
6. **Mejorar todo**: las personas, los equipos y las organizaciones necesitan mejorar continuamente para posponer el fallo tanto como sea posible. (Jurgen, 2010)

### *Roles de Scrum.*

El Método SOFUNI se basa en el marco de desarrollo ágil Scrum, por lo que se plantea, que los alumnos deben jugar en un determinado momento diferentes roles dentro de su equipo. Los roles que pueden tomar son:

**Responsable del equipo**: es quien modera y facilita las Interacciones del equipo y motivador del mismo. Este rol es responsable de asegurarse que el equipo tenga un ambiente de trabajo productivo protegiéndolo de influencias externas y eliminando todos obstáculos.

**Equipo de trabajo:** es responsable del desarrollo del producto, servicio o de cualquier otro resultado. Consiste en un grupo de personas que trabajan en los requerimientos de usuario para crear los entregables del proyecto

Dentro del equipo de trabajo se identifican los siguientes roles específicos:

- Desarrollador. Es responsable de generar productos de calidad, revisa y corrige los productos de que es responsable, se debe ajustar a los estándares del equipo.
- Responsable de documentos. Es el encargado del repositorio común de documentos, se encarga de autorizar o hacer la actualización de documentos cuando hay cambios autorizados, y no permite la modificación de cambios no autorizados.
- Responsable técnico. Dirigir al equipo en la toma de decisiones técnicas en las actividades de desarrollo, conseguir las herramientas necesarias para el trabajo, entrenar a los miembros del equipo en su uso.
- Responsable de seguimiento. Da seguimiento al plan, es el responsable del tablero de seguimiento, registra los riesgos y da seguimiento a los mismos. (Ibargüengoitia, 2018).

Es importante considerar que en el momento de la creación y conformación del equipo solamente se identifican los roles de los integrantes, de acuerdo a sus habilidades e intereses.

| |
|---|
| En el caso particular, en el que un producto de software sea desarrollado de manera individual, se iniciará en la Práctica 2 “Definir el Enunciado del trabajo” |

## Práctica 2. Definir el Enunciado del trabajo

La ISO/IEC 29110 parte del Enunciado del trabajo ya creado, sin embargo; es necesario que los estudiantes definan el Enunciado del trabajo a partir de una solicitud básica para la creación o modificación de un sistema o producto de software.

En esta práctica se debe realizar una descripción a detalle del producto que se realizará, identificando los contactos involucrados y el alcance de producto expresado mediante un mapa de impacto, también se debe establecer el protocolo de entrega del producto, dejando claro la fecha, lugar y forma, así como persona o personas que entregarán y recibirán el producto. La estructura de la práctica 2 se muestra en la Figura 3.2, la plantilla genérica se puede observar en la tabla 3.3 y la herramienta sugerida en la tabla 3.4


Figura 3.2 Práctica 2: Definir el Enunciado de trabajo (de creación propia).

## Plantilla genérica. Práctica 2. "Definir el Enunciado de trabajo"

Tabla 3.3 Plantilla genérica de Práctica 2 (de creación propia)

| Plantilla de Práctica No. 2 "Definir Enunciado del trabajo" | |
|---|---|
| Actividad | Elementos o subactividades |
| 1. Elaborar descripción del producto de software | a) Nombre de sistema o producto<br>b) Contexto en el que funcionará el software<br>c) Descripción<br>d) Diccionario de datos<br>e) Objetivos |
| 2. Identificar involucrados clave | e) Nombre<br>f) Contacto<br>i. Teléfono<br>ii. Correo electrónico |
| 3. Crear repositorio para documentación | g) Herramienta<br>h) Ubicación<br>i) Crear estrategia de versiones |
| 4. Hacer mapa de impacto | j) Mapa de impacto (Figura 3.3) |
| 5. Establecer protocolo de entrega | k) Fecha y hora de entrega<br>l) Lugar<br>m) Persona(s) que entregan<br>n) Personas(s) que reciben |

## Herramientas recomendadas. Práctica 2. "Definir el enunciado de trabajo"

Tabla 3.4 Herramientas sugeridas Práctica 2 (de creación propia).

| **Herramientas Práctica No. 2 "Definir el enunciado de trabajo"** | |
|---|---|
| Herramientas recomendadas | Propósito |
| 1. Almacenamiento en la nube<br>1. Drive<br>2. Dropbox<br>3. OneDrive<br>4. pCloud<br>5. Otros | Crear un repositorio de documentación para compartir información y trabajar de manera colaborativa entre los integrantes del equipo. |
| 2. Control de versiones<br>1. GitHub<br>2. Mercurial<br>3. GitLab<br>4. Apache Allura<br>5. Otros | Aunque se utiliza principalmente para administrar los cambios en el código fuente, también se puede utilizar para la administración de los cambios en la documentación. |

**Base conceptual. Práctica 2. "Definir el enunciado de trabajo"**

*Mapas de impacto*

Un mapa de impacto es una técnica de planificación y estrategia creada por Gojko Adzic. (Azdic, 2012) Es una herramienta fundamental que permite, además, generar la versión inicial del Backlog del producto (Lista de las tareas a realizar durante el proyecto). Se realiza durante las entrevistas con los involucrados y miembros del equipo de desarrollo, Con base en 4 preguntas básicas:

- **¿Por qué?** Es el centro del mapa. Es objetivo que se está tratando de lograr. Una buena meta debería ser SMART (Specific, Measurable, Action-oriented, Realistic and Timely).
- **¿Quién?** Se indica quiénes son los actores que pueden influenciar el resultado. Quién puede producir el efecto deseado, quién puede obstruirlo y quién será impactado. Se debe ser lo más específico posible.
- **¿Cómo?** Explicita el impacto que se trata de crear. Cómo debe cambiar la conducta de los actores y cómo pueden ayudar a lograrlo. Especificar estos impactos permite priorizar e identificar riesgos.
- **¿Qué?** Se refiere a los entregables. Qué puede hacer el equipo o la organización para lograr el impacto. Puede ser refinado de manera iterativa.

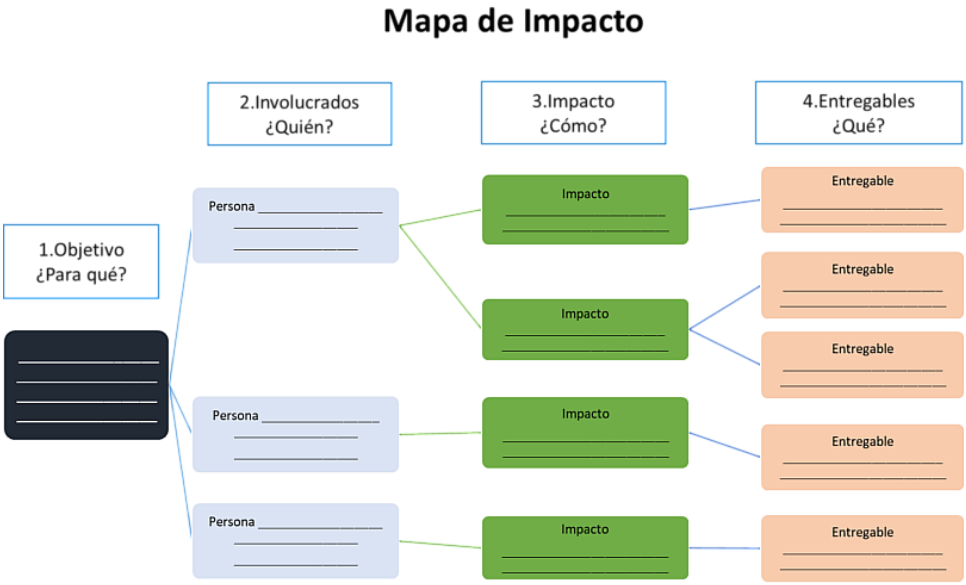



Figura 3.3 Práctica 2: Mapa de Impacto (de creación propia)

## Práctica 3. Elaborar el Plan del proyecto

En la planeación del proyecto se documentan los detalles necesarios para llevar a cabo una adecuada gestión del proyecto. El plan de Proyecto no solo proporciona una guía para el desarrollo de un proyecto, sino que también es un instrumento de control. En la práctica actual se identificarán las Historias de Usuario a partir del mapa de impacto, se organizan de manera prioritaria para establecer un plan de versiones, se establece el número de Iteraciones y se crea un cronograma para la gestión del proyecto, se determinan los recursos humanos y materiales necesarios, se establece una estrategia para el control de versiones y por último una estrategia para control del riesgo. La estructura de la práctica 3. "Elaborar Plan de Proyecto" se muestra en la figura 3.4. La plantilla genérica se muestra en la tabla 3.5 y las herramientas requeridas en la práctica en la tabla 3.6

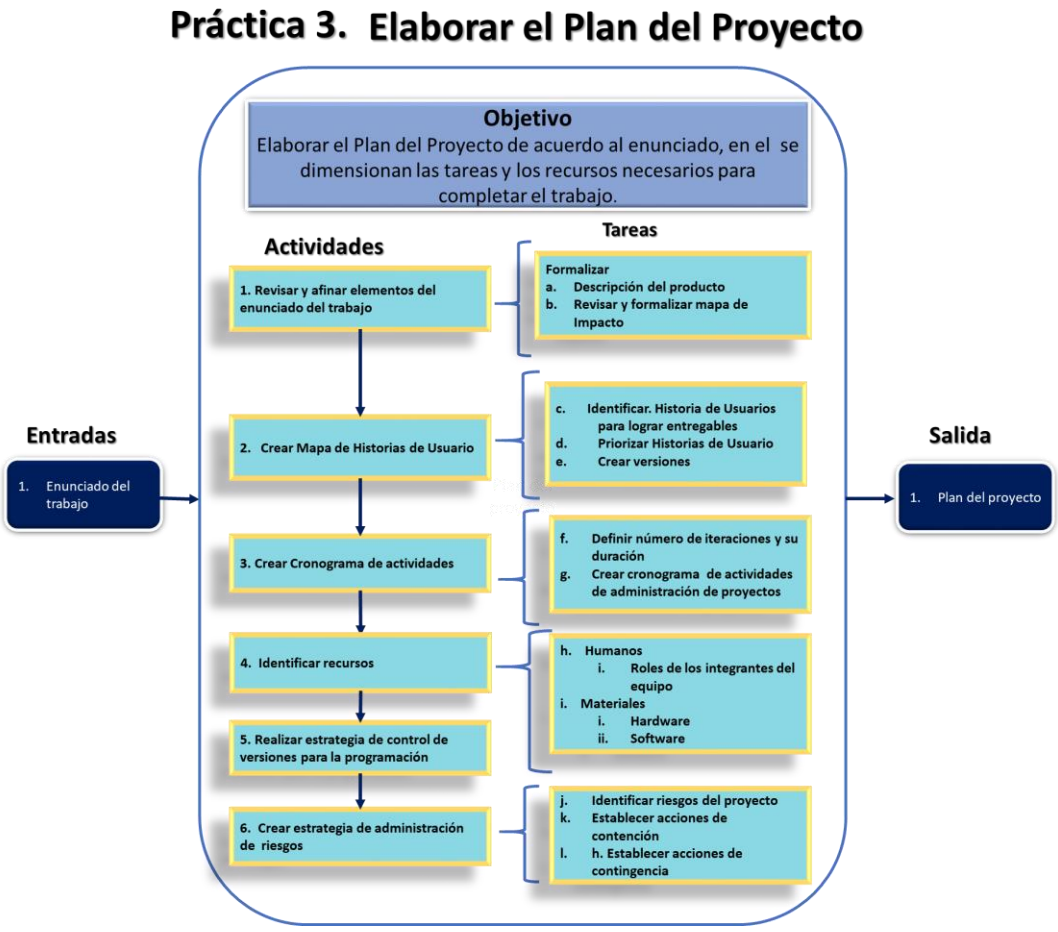



Figura 3.4 Elaborar el plan del proyecto (de creación propia).

## Plantilla genérica. Práctica 3. "Elaborar el Plan del Proyecto"

Tabla 3.5 Plantilla genérica de Práctica 3 (de creación propia)

| Plantilla de Práctica No. 3 "Elaborar el Plan del Proyecto" | |
|---|---|
| Actividad | Elementos o subactividades |
| 1. Nombre del sistema | a) Nombre del sistema |
| 2. Revisar y afinar elementos del Enunciado del trabajo | a) Descripción del problema<br>b) Revisar y formalizar el mapa de impacto |
| 3. Hacer cronograma para la Gestión del Proyecto | a) Definir número y duración de iteraciones<br>b) Actividades a programar<br>i. Plan del Proyecto<br>ii. Análisis de requisitos<br>iii. Definir Arquitectura y Diseño<br>iv. Desarrollo<br>i. (Número y duración de iteraciones)<br>v. Entrega del producto |
| 4. Crear mapa de Historias de usuario | a) Mapa de historias de usuario (figura 3.5) |
| 5. Priorizar Historias de Usuario Definir el MPV (Mínimo Producto Viable) | a) Priorizar de historias de usuario<br>b) Definir el MPV (figura 3.6) |
| 6. Identificar recursos de desarrollo | a) Hardware<br>b) Software |
| 7. Especificar estrategia de control de versiones (Documentación y código) | a) Herramienta<br>Ubicación<br>Estructura |
| 8. Estrategia de administración de riesgos | a) Nombre<br>b) Descripción<br>c) Probabilidad de ocurrencia<br>d) Plan de contingencia<br>e) Plan de contención |

## Herramientas recomendadas. Práctica 3. "Elaborar el Plan del proyecto"

Tabla 3.6 Herramientas Práctica No. 3 "Elaborar el Plan del Proyecto" (de creación propia)

| Herramientas Práctica No. 3 "Elaborar el Plan del Proyecto" | |
|---|---|
| Herramientas recomendadas | Propósito |
| 1. Mapa de Historias de Usuario<br>1. Definir el Mínimo Producto Viable | A partir de los entregables obtenidos del mapa de impacto, identificar tareas a realizar para lograr cada uno de ellos (fig. 3.8), y el Mínimo Producto Viable(MPV), que proporciona una versión inicial del producto y permite recabar información para el desarrollo del mismo.<br>(Patón Jeff, 2014) |
| 2. Técnica MoSCoW | Es una técnica para priorizar requisitos creada por Dai Clegg, es muy sencilla, consiste en asignar la prioridad a cada Historia de Usuario bajo los siguientes criterios<br>**M** Debe tener (*Must have*)<br>**S** Debería tener (*Should Have*)<br>**C** Podría tener (*Could Have*)<br>**W** No tendrá en esta ocasión(*Would not Have*)<br>(SBOK,2016) |

## Base conceptual. Práctica 3. "Elaborar el Plan del proyecto"

### *Mapa de Historias de Usuario*

A partir de los entregables del mapa de impacto, se crea el Mapa de Historias de Usuario

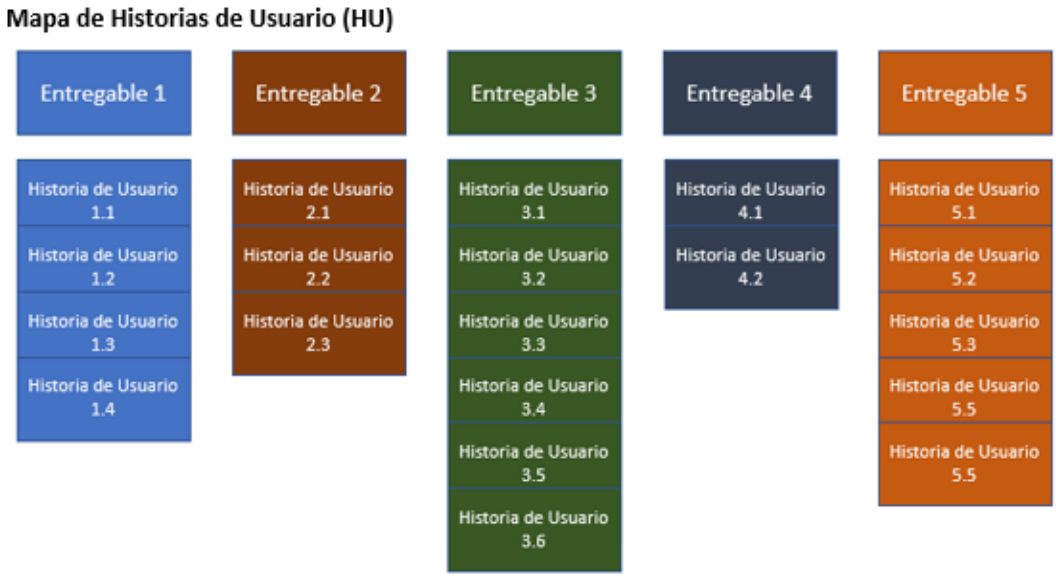



Figura 3.5 Mapa de Historias de Usuario (de creación propia)

Cada entregable tiene un conjunto de Historias de Usuario (H.U), a estas historias de usuario se les asigna una prioridad usando la técnica MoSCoW, tomando en cuenta las dependencias funcionales.

### *Mínimo Producto Viable (MPV)*

Una vez asignadas las prioridades se debe definir el Mínimo Producto Viable (MPV). Un MVP tiene sólo aquella funcionalidad requerida para mostrar el producto al cliente y su principal objetivo es evitar el desarrollar productos que los clientes no quieran y maximizar la información obtenida sobre los clientes con base en el costo y esfuerzo invertidos. (Stevens, 2011). Se obtiene tomando las Historias de usuario de cada entregable de más alta prioridad, según las necesidades del cliente y tomando en consideración las dependencias funcionales como se muestra en la figura 3.6

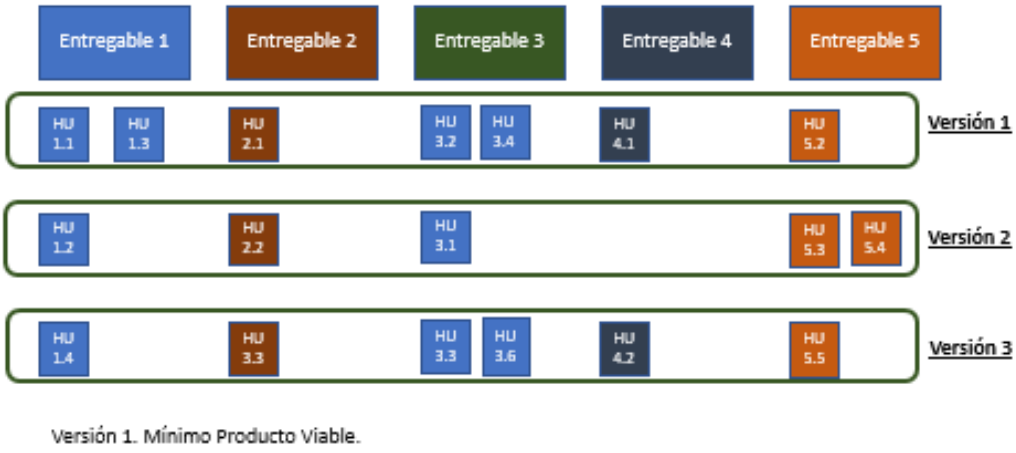



Figura 3.6 Identificación del Mínimo Producto Viable (MPV) (de creación propia).

## Práctica 4. Realizar análisis de requisitos

En esta práctica se afinan las características que el producto debe tener, tanto funcionales como no funcionales, se formalizan las Historias de Usuario, se especificarán sus criterios de aceptación y se lleva a cabo una estimación en puntos de historia, se escriben en un backlog priorizado, tomando como base el Mínimo Producto Viable (MPV). La estructura de la práctica 4 se muestra en la figura 3.8, La plantilla genérica se muestra en la tabla 3.7 y la tabla 3.8 muestra las herramientas requeridas.

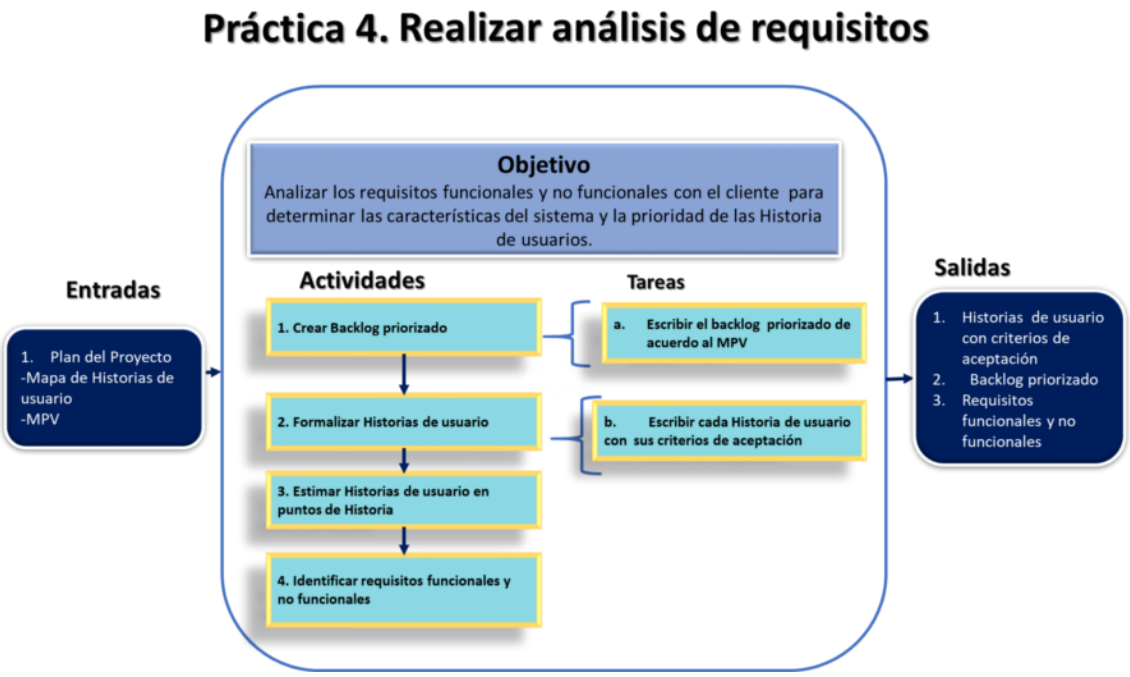



Figura 3.8 Práctica 4: Realizar análisis de requisitos (de creación propia)

### Plantilla genérica. Práctica 4. "Realizar análisis de requisitos"

Tabla 3.7 Plantilla genérica de Práctica 4. (de creación propia)

| Plantilla de Práctica No. 4 "Realizar análisis de requisitos" | |
|---|---|
| Actividad | Elementos o subactividades |
| 1. Crear Backlog priorizado | a) Escribir el Backlog priorizado de acuerdo al MPV. Ver tabla 3.9 |
| 2. Formalizar Historias de usuario | b) Escribir cada Historia de Usuario con sus criterios de aceptación |
| 3. Estimar Historias de usuario en puntos de Historia | c) Definir los Puntos de Historia para cada Historia de usuario. Ver tabla 3.10 de acuerdo a la técnica MoSCoW |
| 4. Identificar requisitos funcionales y no funcionales | d) Para cada entregable identificar sus Historias de usuario relacionadas y sus requisitos funcionales y no no funcionales relacionados. Ver tabla 3.11 |

## Herramientas recomendadas. Práctica 4. "Realizar análisis de requisitos"

Tabla 3.8. Herramientas de práctica 4 "Realizar análisis de requisitos" (de creación propia)

| Herramientas Práctica No. 4 "Realizar análisis de requisitos" | |
|---|---|
| Herramientas recomendadas | Propósito |
| 1. Estimación por Puntos de Historia | La estimación se realiza para determinar con cuántas Historias de usuario se puede comprometer el equipo en una iteración. Cuando se hace una estimación por Puntos de Historia, lo que se compara es el grado de dificultad de cada Historia de Usuario con respecto a una Historia de Usuario llamada *pivote*. |
| 2. Planning Poker | Realizar la estimación de esfuerzo o complejidad de Historias de usuario por parte del equipo de trabajo. Se usan tarjetas con los números de la serie de Fibonacci. Cada integrante del equipo tiene sus tarjetas, y una vez, entendida una Historia de usuario, cada integrante, de forma individual debe proponer la tarjeta, cuya puntuación es acorde a el esfuerzo o complejidad, se debe lograr un consenso, si hay diferencias, cada integrante explica sus razones y se repite el procedimiento anterior. (Grenning, 2011) |

## Base conceptual. Práctica 4. "Realizar análisis de requisitos"

### *Backlog Priorizado*

El Backlog Priorizado del producto es un documento de requisitos que define el alcance del proyecto, proporcionando una lista de prioridades de las características del producto o servicio a ser entregado por el proyecto. Las características necesarias se describen en forma de Historias de usuario. Dichas Historias son requisitos específicos señalados por varios involucrados que se relacionan con el producto o servicio propuesto. (SCRUMstudy, 2016). Ver tabla 3.9

Tabla 3.9 Backlog priorizado (de creación propia)

| Id | Historia de usuario | Prioridad |
|---|---|---|
| HU 1.1 | Nombre de Historia de Usuario | 1 |
| HU 1.2 | | 2 |
| HU 1.3 | | 1 |
| HU 1.4 | | 3 |
| HU 2.1 | | 1 |
| HU 2.2 | | 2 |
| ... | | |

Una vez que se conocen las prioridades, se propone inicialmente en que Iteración se realizará y, ésta tabla se estará actualizando después de cada Iteración. También ayuda a visualizar, con cuántos Puntos de historia, se está comprometiendo el equipo en cada Iteración. (Tabla 3.10)

Tabla 3.10 Historias de usuario con prioridad y Puntos de usuario (de creación propia)

| Id | Historia de usuario | Prioridad | Puntos de historia | Iteración |
|---|---|---|---|---|
| HU 1.1 | | | | |
| HU 1.2 | | | | |
| HU 1.3 | | | | |
| ... | | | | |
| | | | | |

Los entregables se integran de una o más Historias de usuario y cada entregable, tiene implícito uno o más requisitos funcionales y puede tener uno o más requisitos no funcionales. (Tabla 3.11)

Tabla 3.11 Requisitos funcionales y no funcionales (Ejemplo) (de creación propia).

| Requisitos funcionales | Historia de usuario para lograr requisitos funcional | Requisito no funcional relacionado |
|---|---|---|
| De entregable 1 | HU 1.1, HU 1.2 | Usabilidad, seguridad |
| De entregable 2 | HU 1.3 | No aplica |
| De entregable 3 | HU 1.4 | Eficiencia |
| ... | ... | ... |

## Práctica 5. Especificar arquitectura y diseño

La arquitectura de software es de especial importancia, ya que es la manera en que se estructura un sistema, tiene impacto directo sobre la capacidad de éste, para satisfacer lo que se conoce como los atributos de calidad del sistema. (Cervantes, 2010).

Los patrones arquitectónicos proporcionan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados.

En ésta práctica se seleccionará el estilo arquitectónico de acuerdo a los requisitos no funcionales del producto, se debe especificar el ambiente de desarrollo requerido, se realizarán los diagramas de distribución, de paquetes, de navegación y se diseñará y creará la base de datos. La figura 3.9 muestra la estructura de la práctica 5. La plantilla genérica se muestra en la tabla 3.12 y las herramientas se muestran en la tabla 3.13.

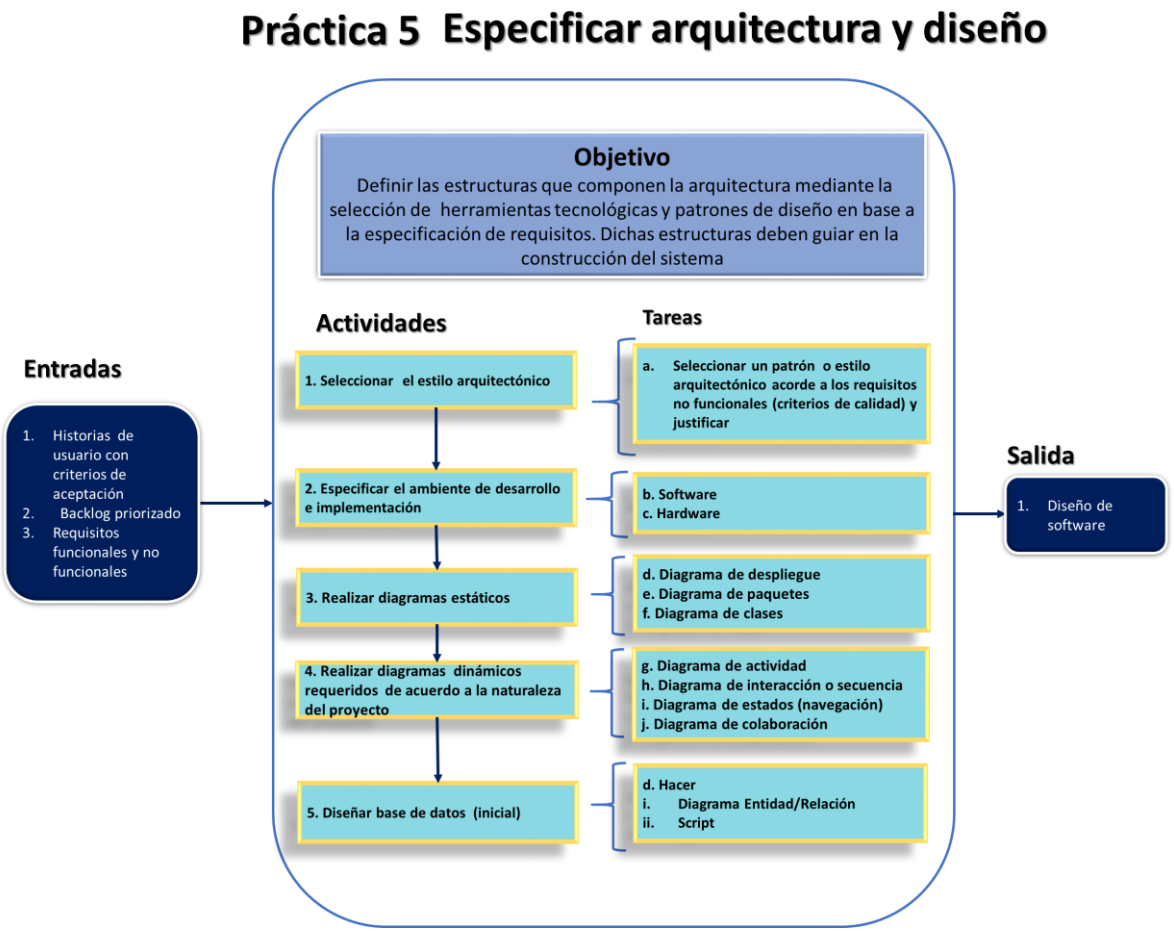



Figura 3.9 Práctica 5: Especificar arquitectura y diseño (de creación propia).

## Plantilla genérica. Práctica 5. "Especificar arquitectura y diseño"

Tabla 3.12 Plantilla genérica Práctica 5 (de creación propia)

| Plantilla de Práctica No. 5 "Especificar arquitectura y diseño" | |
|---|---|
| Actividad | Elementos o subactividades |
| 1. Seleccionar estilo arquitectónico | a) Seleccionar un estilo arquitectónico acorde a los requisitos no funcionales (criterios de calidad) y justificar .<br>b) Seleccionar ´patrón de diseño a utilizar y justificar |
| 2. Implementar el ambiente de desarrollo e implementación | c) Software<br>d) Hardware |
| 3. Realizar diagramas estáticos | e) Diagrama de despliegue<br>f) Diagrama de paquetes<br>g) Diagrama de clases |
| 4. Realizar diagramas dinámicos requeridos de acuerdo a la naturaleza del proyecto | h) Diagrama de actividad<br>i) Diagrama de interacción o secuencia<br>j) Diagrama de estados (navegación)<br>k) Diagrama de colaboración |
| 5. Diseñar base de datos | l) Diagrama Entidad/Relación<br>m) Script |

## Herramientas recomendadas. Práctica 5. "Especificar arquitectura y diseño"

Tabla 3.13 Herramientas de práctica 5 "Especificar arquitectura y diseño" (de creación propia)

| **Herramientas Práctica No. 5 "Especificar arquitectura y diseño"** | |
|---|---|
| Herramientas recomendadas | Propósito |
| 1. Diagramas UML<br>1. Diagramas de despliegue<br>2. Diagrama de paquetes<br>3. Diagrama de clases<br>4. Diagrama de actividad<br>5. Diagrama de secuencia<br>6. Diagrama de estados<br>7. Diagrama de colaboración | **El diagrama de despliegue.** Muestra una vista estática de la configuración en tiempo de ejecución de los nodos de procesamiento y los componentes que se ejecutan en esos nodos. Así como el protocolo utilizado para conectar los nodos entre sí.<br>**Diagrama de paquetes.** Muestra cómo se organizan los elementos del modelo en paquetes, así como las dependencias entre ellos.<br>**Diagrama de clases.** Muestra las clases del sistema, sus interrelaciones, operaciones y atributos de las clases.<br>**Diagrama de actividad.** Se utilizan para explorar la lógica de: Una operación compleja, un proceso de negocio, un proceso de software, etc.<br>**Diagrama de secuencia.** Proporciona una descripción de una manera potencial de utilizar su sistema. Ayuda a validar y desarrollar la lógica de una Historia de Usuario o un Caso de Uso.<br>**Diagrama de estados.** Se usa para mostrar la transición entre estados, por lo tanto es adecuado para mostrar la navegación entre ventanas en una aplicación o páginas en un sitio Web.<br>**Diagrama de colaboración.** muestran el flujo de mensajes entre objetos en una aplicación OO así como las relaciones entre clases. (Ambler, 2003-2018) |
| 2. Diagrama de Entidad/Relación | Para modelar los objetos de datos y sus relaciones con respecto a las necesidades de un producto de software específico. (Silberschatz, Korth, & Sudarshan, 2006) |

## Base conceptual. Práctica 5. "Especificar arquitectura y diseño"

### *Arquitectura de software*

La Arquitectura de Software es, una vista del sistema que incluye los componentes principales del mismo, el comportamiento de esos componentes y las formas en que los componentes interactúan y se coordinan para alcanzar los requisitos del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones. (Clements, 1996)

El Método SOFUNI recomienda el estilo de **Arquitectura Cliente Servidor de dos o tres capas** o bien **Arquitecturas Web** (implícitamente una arquitectura cliente-servidor de dos o tres capas), de acuerdo a la naturaleza del producto de software a desarrollar. Se recomienda también el uso del **patrón** de diseño **Modelo Vista Controlador (MVC)**.

### *Arquitectura de capas*

Organiza el sistema en capas con funcionalidad relacionada con cada capa. Una capa da servicios a la capa de encima, de modo que las capas de nivel inferior representan servicios núcleo que es probable se utilicen a lo largo de todo el sistema. (Sommerville, 2004)

### *Patrón: Modelo - Vista - Controlador (MVC)*

Se estructura en tres componentes lógicos que interactúan entre sí.

- **Vista.** define y gestiona cómo se presentan los datos al usuario**.**
- **Modelo.** contiene la funcionalidad básica de la aplicación, encapsula los datos y las operaciones sobre estos, crear, editar, borrar datos etc.
- **Controlador**. Recibe las órdenes del usuario y se encarga de solicitar los datos al modelo y de comunicárselos a la vista.

Este patrón se recomienda para aplicaciones interactivas con una interfaz flexible. (Buschman, Regine, Rohnert, Sommerland, & Stal, 1996)

### *Diagramas UML para representar la arquitectura de un sistema*

Para representar adecuadamente la arquitectura de un sistema, es necesario contar con varios diagramas o vistas (Bass, Clements, & Kazman, 1998). Cada una de estas vistas es una estructura de la arquitectura del sistema en la que se muestra una parte del sistema como un conjunto de componentes, conectores y restricciones sobre sus tipos y relaciones.

La arquitectura, conformada por diferentes visiones del sistema, constituye un modelo de cómo está estructurado dicho sistema. (Hurtado, 2006). UML permite representar de manera general la estructura de un sistema, en sus diferentes etapas.

### *Diagramas de despliegue*

La vista de despliegue muestra la disposición física de los nodos. Un nodo es un recurso computacional de ejecución, como computadoras u otros dispositivos. Durante la ejecución, los nodos pueden contener artefactos, entidades físicas como los archivos. Un nodo modela un recurso computacional de tiempo de ejecución, Un artefacto modela una entidad física como un archivo. Un artefacto se muestra mediante un rectángulo con la palabra clave «artifact»

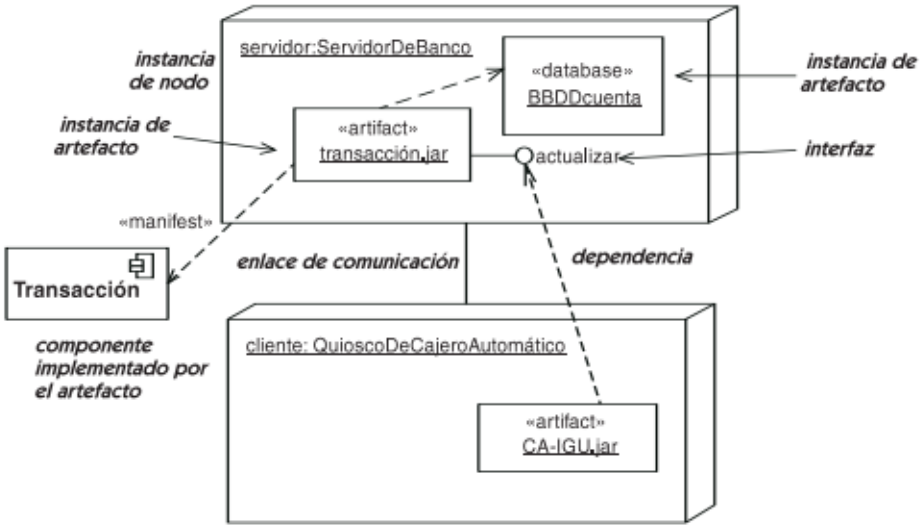



Figura 3.10 Diagrama de despliegue. El Lenguaje Unificado de Modelado, manual de referencia UML 2.0 (Segunda ed.). Madrid: Addison Wesley Pearson. Rumbaugh, J., Jacobson, I., & Booch, G. (2007). Figura 10.1 pág. 98

*Diagramas de paquetes*

Los paquetes permiten dividir sistemas grandes en unidades más pequeñas, de forma que las personas puedan trabajar con una cantidad limitada de información en cada momento, y los equipos que trabajan en cada paquete no interfieran entre sí. Los paquetes contienen elementos de alto nivel del modelo, como las clases y sus relaciones, máquinas de estados, diagramas de casos de uso. Ver figura. 3.11

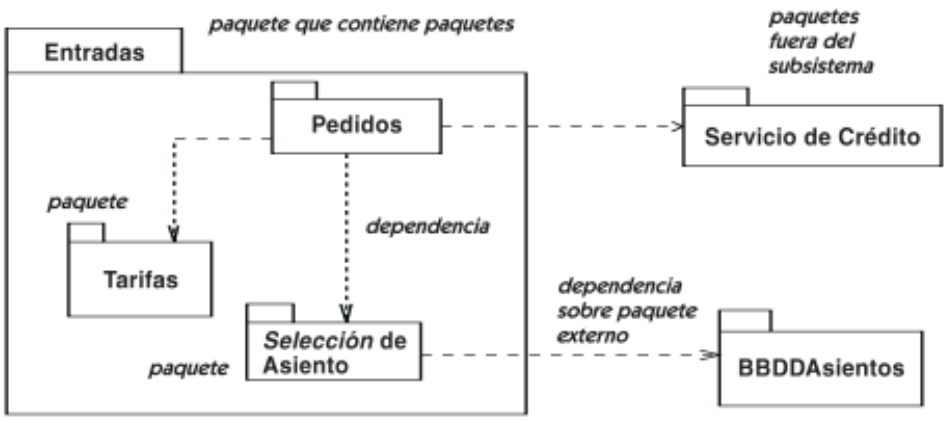



Figura 3.11 Paquetes y sus relaciones. El Lenguaje Unificado de Modelado, manual de referencia UML 2.0 (Segunda ed.). Madrid: Addison Wesley Pearson. Rumbaugh, J., Jacobson, I., & Booch, G. (2007). Figura 11.1 pág. 100

*Diagramas de estado*

Un diagrama de estados o máquina de estados es un grafo de estados y transiciones. Es un modelo de todas las posibles historias de vida de un objeto. Cuando el objeto detecta un evento, responde de una manera que depende de su estado actual. Como las interfaces de usuario y los controladores de dispositivos. (Rumbaugh, Jacobson, & Booch, 2007)

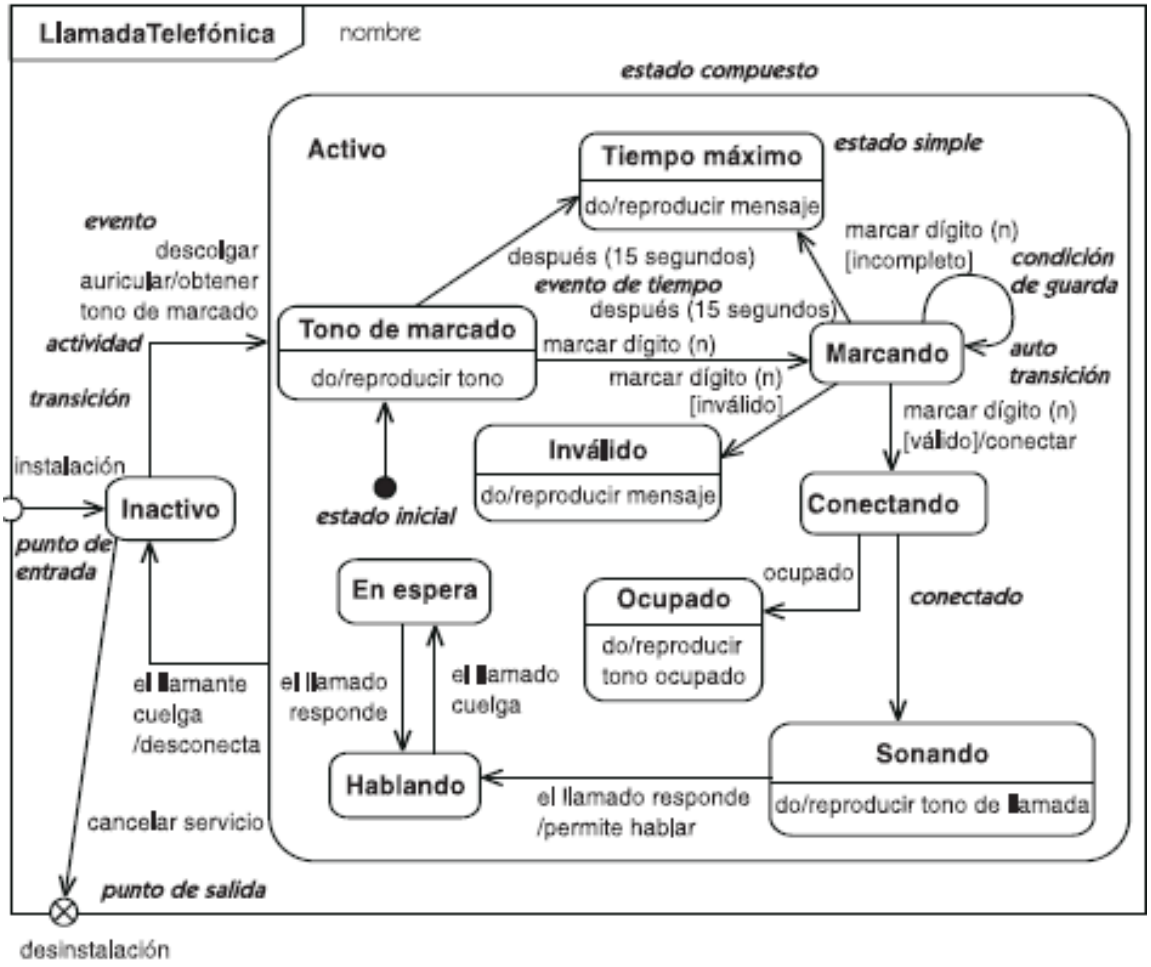



Figura 3.12 Diagrama de estado. El Lenguaje Unificado de Modelado, manual de referencia UML 2.0 (Segunda ed.). Madrid: Addison Wesley Pearson. Rumbaugh, J., Jacobson, I., & Booch, G. (2007). Figura 14.175 pág. 428

## Práctica 6. Planear Iteración

Las prácticas 6, 7 8 y 9 corresponden a la Iteración, y se llevan a cabo tantas veces como Iteraciones haya. Ver figura 3.4 correspondiente al Método SOFUNI. Cada Iteración empieza con una Reunión de Planificación de la Iteración (Sprint Planning Meeting), durante la cual se consideran las Historias de usuario de alta prioridad para su inclusión en la Iteración.

Una Iteración generalmente tiene una duración de una a seis semanas durante las cuales el equipo trabaja en la creación de entregables de forma incremental. (SCRUMstudy, 2016). En esta práctica se realizará la Reunión de Planificación de la Iteración, en la que se seleccionarán las Historias de Usuario (HU) a realizar en el Iteración a partir del Backlog priorizado, se definirán las tareas de cada HU, cada integrante del equipo seleccionará las tareas a realizar durante la Iteración y debe hacer una estimación del tiempo requerido en horas. Durante la reunión se creará la gráfica del Burndown inicial, se analizarán los riesgos del Iteración, y se establecerá una estrategia de control de riesgos. En esta reunión participan, de manera colaborativa el equipo de desarrollo y el Dueño del producto. El esquema de la práctica 6 se muestra en la figura 3.14, la plantilla genérica en la tabla 3.14 y las herramientas en la tabla 3.15

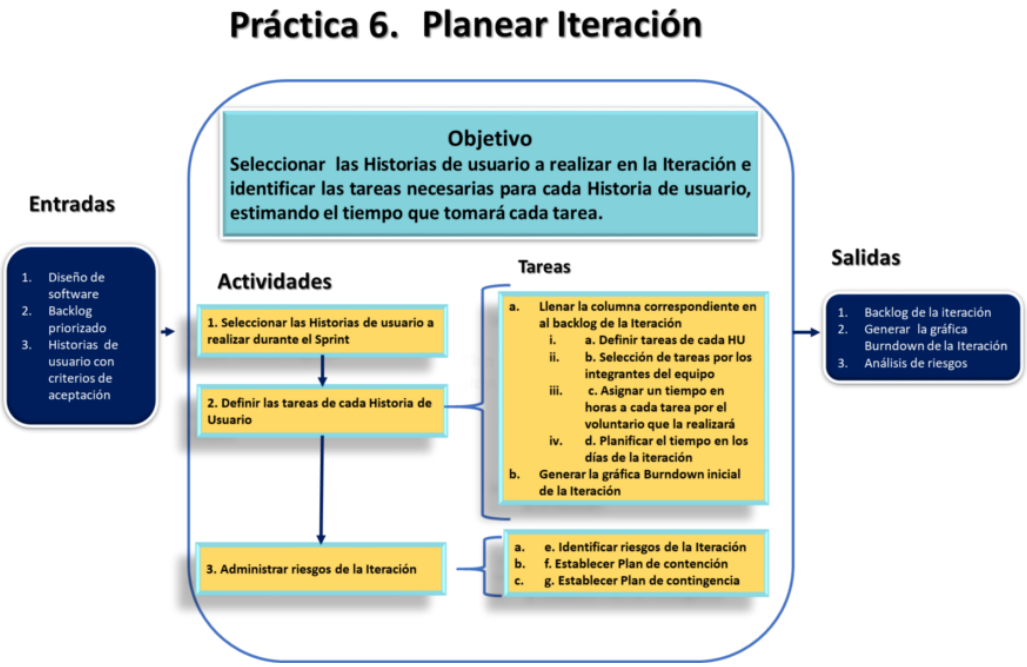



Figura 3.14 Práctica 6: Planear iteración (de creación propia).

## Plantilla genérica. Práctica 6. "Planear la Iteración"

Tabla 3.14 Práctica 6: Plantilla genérica (de creación propia).

| Plantilla de Práctica No. 6 "Planear la Iteración" | |
|---|---|
| Actividad | Elementos o subactividades |
| 1. Seleccionar las Historias de Usuario (HU) a realizar durante la Iteración | La selección de (HU) se hace a partir del Backlog priorizado durante la Reunión de Planificación de la Iteración, participan, el Scrum Master, el Dueño del producto y el Equipo de desarrollo (Llenar columnas Id e Historia de Usuario en el Backlog de la Iteración, ver tabla 3.16) |
| 2. Definir la tareas de cada (HU) | a) Llenar la columna correspondiente al Backlog de la Iteración. Ver Tabla 3.16)<br>i. Definir y/o afinar tareas de cada HU (columna tareas)<br>ii. Selección de tareas por los integrantes del equipo (columna ¿Quién realiza?)<br>iii. Asignar un tiempo en horas a cada tarea por el voluntario que la realizará (columna Horas totales)<br>iv. Planificar el tiempo en los días de la iteración(columna Día)<br>b) Generar la gráfica del Burndown inicial de la Iteración |
| 3. Administrar riesgos de la iteración | e) Identificar riesgos de la Iteración<br>a) Establecer Plan de contención<br>b) Establecer Plan de contingencia<br>(ver tabla 3.17) |

## Herramientas recomendadas. Práctica 6. "Planear la iteración"

Tabla 3.15 Herramientas Práctica 6 (de creación propia).

| **Herramientas Práctica No. 6 "Planear la Iteración"** | |
|---|---|
| Herramientas recomendadas | Propósito |
| 1. Backlog de la Iteración | El equipo se compromete con las Historias de usuario, que realizará durante la Iteración, cada miembro del equipo, se propone para realizar tareas específicas y hace una estimación para esas tareas, de acuerdo con su experiencia y habilidad. Ver tabla 3.16 |
| 2. Gráfica Burndown de la Iteración | La gráfica Burndown de la Iteración muestra la cantidad de trabajo pendiente en la Iteración actual. Inicialmente, muestra el trabajo que deberá realizarse durante la toda la Iteración evaluado en unidades de Puntos de historia. Ver figura. 3.15 |

## Base conceptual. Práctica 6. "Planear la Iteración"

Durante la Reunión de Planificación de la Iteración, el equipo se compromete con las Historias de usuario para la Iteración, e identifica y estima las tareas de cada HU comprometida.

### *Backlog de la Iteración*

La lista de tareas que llevará a cabo el equipo en la siguiente Iteración se denomina, Backlog de la Iteración. Una vez que el equipo finaliza y se compromete con el Backlog de la Iteración, no se deben agregar nuevas Historias de usuario; sin embargo, las tareas que pudieron haberse pasado por alto de las historias de usuario comprometidas si pueden ser agregadas, ver tabla 3.15.

Tabla 3.16 Tabla Backlog de la iteración (de creación propia)

| Id | Historia de usuario | Tarea | Horas Totales | ¿Quién Realiza? | Día 1 | | Día 2 | | Día 3 | | Día 4 | | Día 5 | | ... | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | PLAN | REAL | PLAN | REAL | PLAN | REAL | PLAN | REAL | PLAN | REAL | PLAN | REAL |
| HU 1.1 | Nombre de H.U 1.1 | Tarea 1 | | | | | | | | | | | | | | |
| | | Tarea 2 | | | | | | | | | | | | | | |
| | | ... | | | | | | | | | | | | | | |
| HU 1.n | Nombre de H.U n.1 | Tarea n | | | | | | | | | | | | | | |
| | | Tarea n+1 | | | | | | | | | | | | | | |
| | | ... | | | | | | | | | | | | | | |
| HU 1.m | Nombre de H.U 1.m+1 | Tarea m | | | | | | | | | | | | | | |
| | | Tarea m+1 | | | | | | | | | | | | | | |
| | | ... | | | | | | | | | | | | | | |

*Gráfica Burndown de la Iteración*

La gráfica Burndown de la Iteración, muestra la cantidad de trabajo pendiente en la a Iteración actual. En el eje horizontal de la gráfica se colocan los días del Iteración y en el eje vertical los Puntos de Historia, y se traza una línea que muestra cómo deben ir disminuyendo los Puntos de Historia conforme avanza el Iteración, ver figura 3.15.

La gráfica Burndown de la Iteración debe actualizarse al final de cada día conforme se concluye el trabajo. Dicha gráfica muestra el progreso que ha realizado el equipo y permite la detección de estimaciones que pudieron haberse hecho incorrectamente. (SCRUMstudy, 2016)

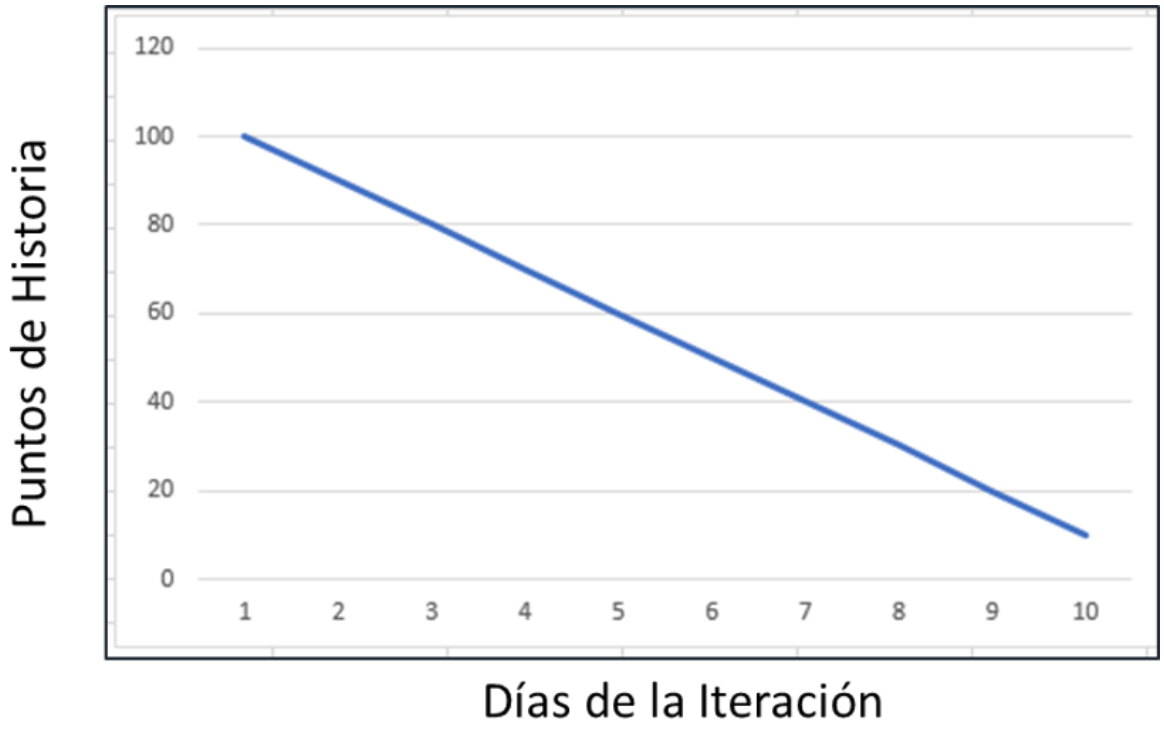



Figura 3.15 Gráfica del Burndown inicial de la Iteración (de creación propia)

*Gestión de Riesgos*

Los objetivos de la gestión de los riesgos del proyecto son aumentar la probabilidad y/o el impacto de los riesgos positivos y disminuir la probabilidad y/o el impacto de los riesgos negativos, a fin de optimizar las posibilidades de éxito del proyecto. (Project Management Institute, 2017)

### Esquema general de la Gestión de Riesgos

- Identificar los riesgos
- Hacer un análisis cuantitativo (de 1 a 5 ¿qué tan probable es que ocurra?)
- Hacer análisis cualitativo (de 1 a 5 ¿Cuál es el impacto si ocurriera?)
- Priorizar los riesgos (Teniendo mayor prioridad, aquellos que al multiplicar el valor cuantitativo y el valor cualitativo el resultado sea mayor
- Establecer un Plan de contingencia. (para evitar la ocurrencia o disminuir la probabilidad de ocurrencia)
- Establecer un Plan de contención. (acciones para disminuir o evitar el impacto y ocurre el riesgo)
- Monitorear (Pressman, 2010)

Tabla 3.17 Gestión de Riesgos de la Iteración (de creación propia).

| Riesgos de la Iteración | |
|---|---|
| Nombre | |
| Descripción | |
| Probabilidad de ocurrencia | |
| Impacto del riesgo | |
| Plan de contingencia | |
| Plan de contención | |

| | |
|---|---|
| Nombre | |
| Descripción | |
| Probabilidad de ocurrencia | |
| Impacto del riesgo | |
| Plan de contingencia | |
| Plan de contención | |

## Práctica 7. Construir el software

La construcción de los componentes de software definidos en el Backlog de la Iteración actual, así como su correspondiente seguimiento, control y verificación se llevan a cabo en esta práctica, teniendo como tareas la Junta Diaria (Daily Meeting), el seguimiento del tablero Kanban, el seguimiento de la gráfica Burndown, la prueba y depuración de cada componente individual de software. La estructura de la práctica se muestra en la figura 3.16, Su plantilla genérica se muestra en la tabla 3.18 y las herramientas a utilizar en la tabla 3.19


Figura 3.16 Práctica 7. Construir el software (de creación propia)

## Plantilla genérica. Práctica 7. "Construir el software"

Tabla 3.18 Práctica 7: Plantilla genérica (de creación propia).

| Plantilla de Práctica No. 7 "Construir el Software" | |
|---|---|
| Actividad | Elementos o subactividades |
| 1. Administrar el flujo del trabajo | a) Establecer las tareas del Backlog de la Iteración, en un tablero Kanban<br>b) Dar seguimiento al tablero Kanban<br>c) Llevar a cabo la junta<br>d) Actualizar la gráfica Burndown |
| 2. Construir los componentes de software | e) Construir componentes |
| 3. Crear casos de prueba unitarios | g) Crear pruebas unitarias para cada componente de software.<br>h) Probar componente<br>i) Corregir |
| 4. Actualizar documentación | j) Actualizar documentación |

## Herramientas recomendadas. Práctica 7. "Construir el software"

Tabla 3.19 Práctica 7: Herramientas de la práctica 7 (de creación propia)

| **Herramientas Práctica No. 7 "Construir el software"** | |
|---|---|
| Herramientas recomendadas | Propósito |
| 1. Tablero Kanban | Se utiliza para hacer un seguimiento visual a un flujo o producción, evitando retrasos e identificando cuellos de botella, proporciona a los miembros del equipo visibilidad sobre el avance en el proceso. Consiste en dividir el trabajo en fases bien definidas(pueden ser diferente número de fases). Ver imagen 3.17 |
| 2. Junta diaria(Daily Standup Meeting) | Breve reunión diaria con duración de 15 minutos. Los miembros del equipo se reúnen para informar sobre cómo avanza el proyecto, respondiendo a las siguientes tres preguntas: 1. ¿Qué he hecho desde la última reunión? 2. ¿Qué tengo planeado hacer antes de la siguiente reunión? 3. ¿Qué impedimentos u obstáculos (si los hubiera) estoy enfrentando en la actualidad?. Se recomienda tener presente el tablero de **Kanban** y la **gráfica burndown** |
| 3. Casos de prueba | Los casos de prueba se usan para verificar que los componentes cumplan con los requisitos. se deben considerar los criterios de aceptación para cada Historia de Usuario. Ejemplo ver figura 3.18 |

## Base conceptual. Práctica 7. "Construir el Software

### *Tablero Kanban*

En el tablero Kanban, la organización de las filas y los colores es flexible, se puede organizar con respecto a la prioridad, de tal manera que las primeras filas sean las de prioridad más alta, en cuanto a los colores se puede asignar un color a cada integrante del equipo para distinguir las tareas de la que es responsable, también se puede organizar por área o categoría, etc. (Kniberg & Skarin, 2010) Ver figura 3.16

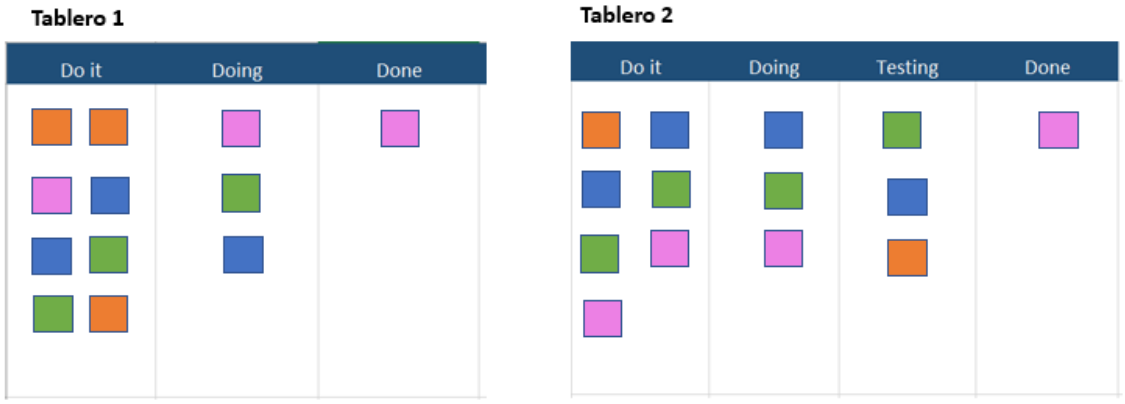



Figura 3.16 Práctica 7. Tableros Kanban (de creación propia)

Kanban también permite identificar cuellos de botella en algún estado y tomar las acciones correspondientes. Se puede asignar un valor máximo a los estados para indicar una alerta.

### *Reunión diaria*

Suelen recibir muchos nombres, reunión diaria o daily meeting, stand-up meeting (reuniones de pie), etc. Popularizadas por Scrum. La razón de que se realicen de pie es para que sea corta de máximo 15 min. En dicha reunión cada integrante del equipo debe responder a las siguientes preguntas (como en la tabla 3.18).

- ¿Qué hiciste ayer?
- ¿Qué harás hoy?
- ¿Hay algún impedimento?

Una reunión diaria debe cubrir los siguientes puntos:

1. Motivar al equipo para empezar el día con fuerza y energía
2. Compartir y resolver problemas
3. Reforzar los objetivos del proyecto
4. Reforzar el sentido de equipo

Se recomienda que la junta de ser posible se lleve a cabo en el mismo lugar, a la misma hora, teniendo presente el tablero de Kanban y la gráfica Burndown que debe actualizarse cada día. En caso de que la reunión no se pueda llevar a cabo de forma presencial, se recomienda que se lleve a cabo usando alguna tecnología como WhatsApp, Skype, etc.

Se debe revisar la gráfica Burndown actualizada para visualizar el avance del equipo (Fowler, 2016). La línea azul muestra el Burndown inicial, cada día se gráfica el resultado de restar los puntos de historia realizadas desde el día anterior (línea roja) y comparando con la gráfica inicial, de tal forma que si la línea roja está por encima de la azul significa que los puntos de historia realizados son menos que los que se planearon, si está por debajo, significa que se tiene holgura con respecto a la planeación.

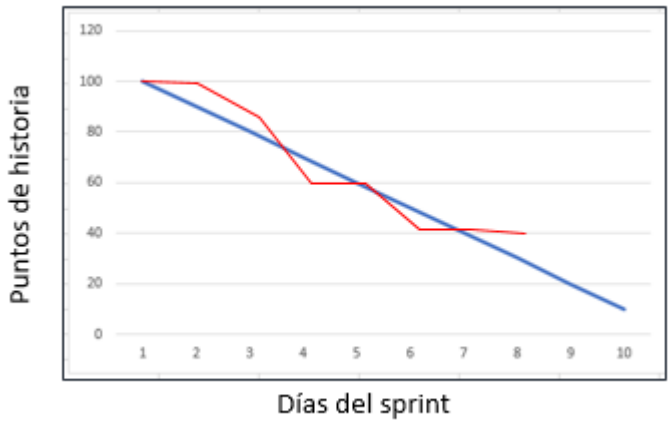



Figura 3.17 Práctica 7. Gráfica Bundown actualizada (de creación propia)

### *Casos de prueba*

Los componentes se realizan de acuerdo con las tareas derivadas de cada Historia de usuario, por lo tanto, se deben considerar los criterios de aceptación en la creación de casos de prueba; Bajo este enfoque las pruebas de software han tomado un papel crucial, dada la necesidad de realizar pequeñas entregas de forma periódica "funcionalmente" estables, surgiendo así el "testing ágil". Existen varias técnicas. Entre ellas una de las más conocidas es TDD (Desarrollo Dirigido por Pruebas)

### *TDD (Desarrollo Dirigido por Pruebas)*

TDD fue creado por Kent Beck (quien también inventó Extreme Programming y JUnit), Consiste en escribir primero las pruebas (pruebas unitarias) para posteriormente escribir el código que pase las pruebas planteadas inicialmente. (Ruiz, 2012)

Pasos a seguir.

1. El dueño del producto escribe su Historia de usuario.
2. El Dueño del producto y el equipo escriben los criterios de aceptación de esta Historia, desglosándolos para simplificarlos todo lo posible.
3. Se escoge el criterio de aceptación más simple y se traduce en una prueba unitaria.
4. Se comprueba que esta prueba falla.
5. Se escribe el código que hace pasar la prueba.
6. Volvemos al punto 3 con los criterios de aceptación que falten y repetimos el ciclo una y otra vez hasta completar nuestra aplicación.

La figura 3.18 muestra un ejemplo de casos de prueba

| Componente | Vista | Casos de prueba |
|---|---|---|
| Autenticación |  | a) El usuario está registrado en la base de datos.<br>Usuario: José<br>Contraseña:1234<br>Resultado: Bienvenida<br><br>b) El usuario y/o contraseña no están en la base de datos.<br>Usuario: Juan<br>Contraseña: Juanito<br>Resultado: Mensaje " verifique su usuario y contraseña"<br><br>c) No se llenaron uno o ambos campos.<br>Usuario:_____<br>Contraseña:_____<br>Resultado: Mensaje " Debe llenar ambos campos" |

Figura 3.18 Práctica 7. Casos de prueba (de creación propia)

## Práctica 8. Hacer Pruebas de integración

En la práctica anterior se prueba cada componente de forma unitaria. Ahora cada componente debe ser integrado al producto de software y una vez acoplado se prueba su correcta interacción a través de sus interfaces, de tal forma que los fallos o problemas de integración detectados puedan ser corregidos lo antes posible, para lo que se realizarán un conjunto de casos de prueba. La estructura de la práctica se muestra en la figura 3.19, la plantilla genérica se muestra en la tabla 3.20 y las herramientas a utilizar en la tabla 3.21

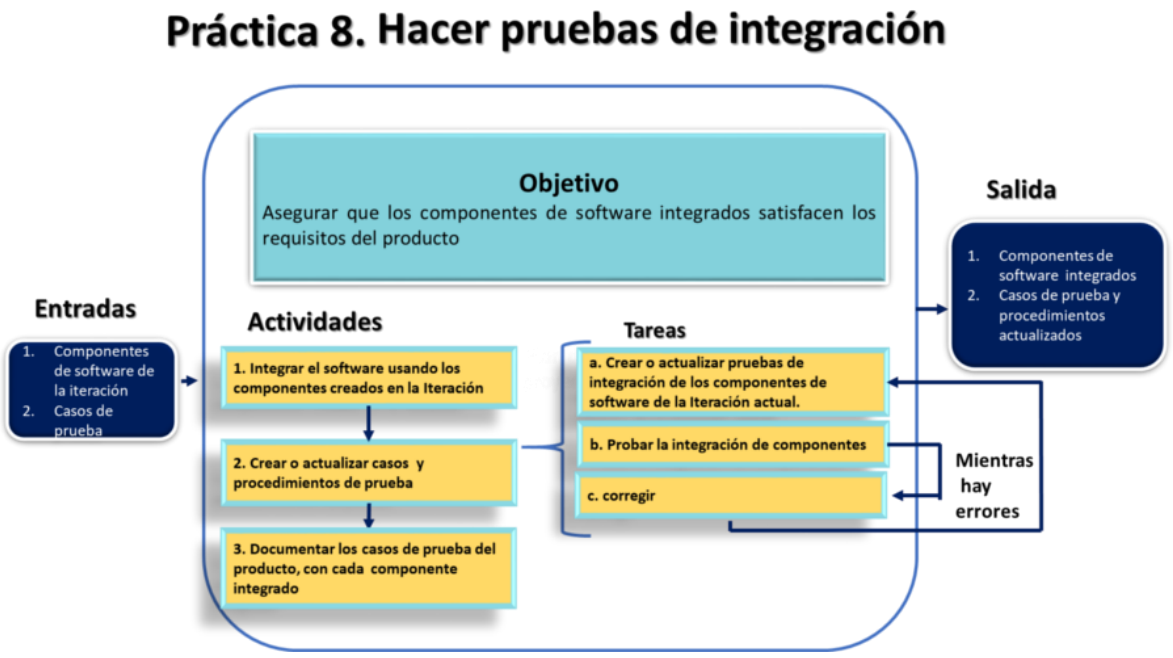



Figura 3.19 Práctica 8: Hacer pruebas de integración (de creación propia)

## Plantilla genérica. Práctica 7. "Hacer Pruebas de integración"

Tabla 3.20 Práctica 7: Plantilla genérica (de creación propia)

| Plantilla de Práctica No. 8 "Hacer pruebas de integración" | |
|---|---|
| Actividad | Elementos o subactividades |
| 1. Integrar el software usando los componentes creados en la iteración | a) Integrar cada componente de software creado en la Iteración actual a la versión del producto generado hasta la Iteración anterior |
| 2. Crear o actualizar casos y procedimientos de prueba | b) Crear casos de prueba que permitan verificar el funcionamiento de los componentes ya integrados; si ya se cuenta con casos de prueba que muestren el correcto funcionamiento hasta la Iteración anterior, se pueden modificar para verificar el correcto funcionamiento del producto una vez integrado el nuevo componente.<br>c) Probar la integración de componente<br>d) Corregir |
| 3. Actualizar la documentación de casos de prueba para cada componente integrado. | e) Cada caso de prueba del producto con cada componente integrado debe ser documentado |

## Herramientas recomendadas. Práctica 8. "Hacer pruebas de integración"

Tabla 3.21 Práctica 7: Herramientas de la práctica 8 (de creación propia)

| **Herramientas Práctica No. 8 "Hacer pruebas de integración"** | |
|---|---|
| Herramientas recomendadas | Propósito |
| 1. Casos de prueba | Los casos de prueba de integración se usan para verificar que los componentes nuevos son compatibles con los ya integrados hasta la versión del producto generado hasta la iteración anterior, y que en su conjunto cumplan con los requisitos de las correspondientes Historias de usuario. Los casos de prueba se basan en los criterios de aceptación de las historias de usuario. Ver ejemplo (figura 3.23) |

**Base conceptual. Práctica 8. "Hacer pruebas de integración"**

Una vez que se verifica que el producto con el nuevo componente funciona de acuerdo con lo previsto, se van acoplando nuevos componentes. Algunas estrategias de integración (Ministerio de Administraciones Públicas, 2001) se muestran en la figura 3.20. El equipo debe seleccionar la estrategia más adecuada de acuerdo a la naturaleza del producto de software que se está desarrollando, ver figura 3.22.

| Estrategias de integración | | |
|---|---|---|
| | Nombre | Descripción |
| Incrementales | De arriba-abajo (top-down ) | El primer componente en la jerarquía se que se desarrolla y se prueba. Los componentes de nivel más bajo se sustituyen por componentes auxiliares para simular a los componentes invocados. Una de las ventajas de aplicar esta estrategia es que las interfaces entre los distintos componentes se prueban en una fase temprana. |
| | De abajo-arriba (bottom-up) | En este caso se crean primero los componentes de más bajo nivel y se crean componentes conductores para simular a los componentes que los llaman. A continuación se desarrollan los componentes de más alto nivel y se prueban. Por último dichos componentes se combinan con el que los llama. Este tipo de enfoque permite un desarrollo más en paralelo que el enfoque de arriba abajo, pero presenta mayores dificultades a la hora de planificar y de gestionar. |
| | Estrategias combinadas | Es útil aplicar las estrategias anteriores conjuntamente. De este modo se desarrollan partes del sistema con un enfoque "top-down", mientras que los componentes más críticos en el nivel más bajo se desarrollan siguiendo un enfoque "bottom-up". En este caso es necesaria una planificación cuidadosa y coordinada |
| No Incrementales | Big-Bang | Se prueba cada componente por separado y posteriormente se integran todos de una vez realizando las pruebas pertinentes |

Figura 3.22 Práctica 8: Estrategias de integración comunes (de creación propia)

Casos de prueba para la integración de nuevos componentes.

Para probar nuevos componentes los casos de prueba para componentes unitarios, se modifican para verificar el correcto acoplamiento del nuevo componente, con respecto a los componentes ya integrados y probados en el producto.

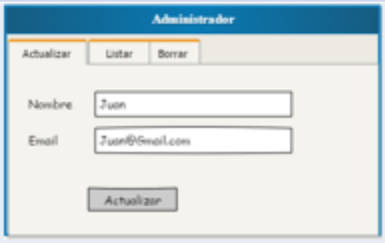
| Componentes | Vista | Casos de prueba |
|---|---|---|
| Autenticación (Sprint previo) |  | a) El usuario está registrado en la base de datos.<br>Usuario: Admin<br>Contraseña:123<br>Resultado: Mostrar pantalla de administrador con lista de clientes. |
| Administrador Listar clientes |  | b) El administrador selecciona la pestaña Actualizar, sin seleccionar un registro previamente<br>Resultado: Mensaje " Debe seleccionar el registro que desea actualizar" |
| | b)<br>  | c) El administrador selecciona un registro y selecciona la pestaña Actualizar<br>Resultado: Se muestra una ventana con los datos del registro seleccionado para editar |
| | c)<br>  | d) El administrador selecciona un registro y la pestaña Actualizar<br>Resultado: Se muestra una ventana con los datos del registro seleccionado para editar |

Figura 3.23 Práctica 8: Caso de prueba para integración (de creación propia).

## Práctica 9 Cerrar la Iteración

Durante el cierre de la iteración, se hace la entrega al cliente, de los componentes creados e integrados durante la Iteración actual, se documenta la aceptación o los cambios requeridos para agregarlos en el Backlog del producto y programar su realización en Iteraciones posteriores. Se lleva a cabo la retrospectiva del Iteración, que consiste en una reunión en la que se asisten el dueño del producto, el Scrum Master y el equipo de trabajo, se identifican tareas que quedaron pendientes en el Iteración, aciertos del equipo Scrum, áreas de oportunidad y se elabora un plan de mejoras para un área de oportunidad que el equipo considere prioritaria, y se implementará en el siguiente Iteración. La estructura de la práctica se muestra en la Figura 3.24, la tabla 3.21 muestra la plantilla genérica y la tabla 3.22 las herramientas recomendadas.

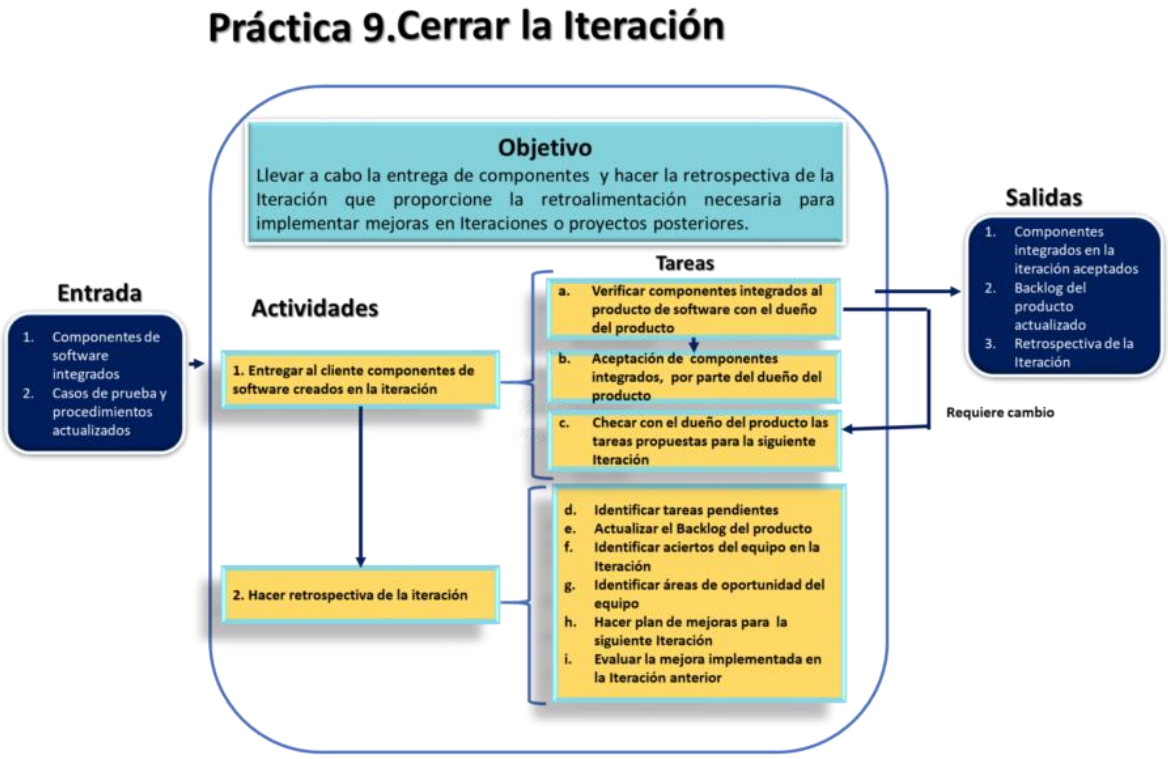



Figura 3.24 Práctica 9: Cerrar la Iteración (de creación propia)

## Plantilla genérica. Práctica 9. "Cerrar la Iteración"

Tabla 3.21 Práctica 9: Plantilla genérica (de creación propia).

| Plantilla de Práctica No. 9 "Cerrar la iteración " | |
|---|---|
| Actividad | Elementos o subactividades |
| 1. Entregar al dueño del producto, los componentes de software creados en la iteración | a) Verificar componentes integrados al producto de software con el dueño del producto<br>b) Solicitar aceptación de componentes integrados<br>c) Checar con el dueño del producto las tareas propuestas para la siguiente Iteración |
| 2. Hacer retrospectiva de la iteración | a) Identificar tareas pendientes<br>b) Actualizar el backlog del producto<br>c) Identificar aciertos del equipo en la Iteración<br>d) Identificar áreas de oportunidad del equipo<br>e) Hacer plan de mejoras parala siguiente Iteración<br>f) Evaluar la mejora implementada durante la Iteración |

## Herramientas recomendadas. Práctica 9. "Cerrar la iteración"

Tabla 3.22 Herramientas de la práctica 9 (de creación propia)

| **Herramientas Práctica No. 9 "Cerra la Iteración"** | |
|---|---|
| Herramientas recomendadas | Propósito |
| 1. Reunión de retrospectiva de la Iteración | El equipo se reúne al final del sprint, el Scrum Master dirige la reunión, el objetivo es identificar.<br>a) Lo que el equipo está haciendo bien (Buenas prácticas)<br>b) Las áreas de oportunidad (lo que el equipo debe mejorar)<br>c) Lo que el equipo debe dejar de hacer (problemas de proceso y embotellamiento) |
| 2. Técnica para llevar a cabo una retrospectiva | Existen muchas técnicas para llevar a cabo la retrospectiva, entre las más conocidas está "La estrella de mar" y "El barco de vapor"; pero existe una lista bastante grande de técnicas que pueden ser usadas, dependiendo de las características de cada equipo de trabajo. |

## Base conceptual. Práctica 9. "Cerrar la iteración"

### *Retrospectiva del Iteración*

Una retrospectiva es una oportunidad para aprender y mejorar, en un ambiente divertido. Para descomponerlo de la manera más simple debe cubrir:

¿Qué funcionó bien?

¿Qué no funcionó bien?

¿Qué elementos de acción podemos llevar a cabo para mejorar? (Caroli & Caetano, 2015). El equipo debe elegir una técnica o estrategia de acuerdo a sus características.

## Práctica 10. Entrega del producto

Una vez que ya no hay más Iteraciones, y se tiene una versión final del producto, se procede a la entrega del producto de acuerdo con el protocolo de entrega que se formalizó en el Plan del proyecto. Se entregará la configuración del software, según sea el caso, en el equipo del Dueño del producto o en algún servidor acordado junto con un guión general de pruebas, el cual demostrará de manera general el cumplimiento de los criterios de aceptación definidos para cada una de las Historias de usuario (HU). Además, se debe entregar un manual de usuario breve y conciso (puede ser mediante un tríptico o infografía) que muestre cómo acceder a cada una de las funciones del producto o bien integrar en el software la ayuda pertinente para acceder a cada una de sus funciones. También se proporcionará un manual de instalación, el cual debe indicar el ambiente y herramientas requeridas tanto de hardware como de software para la instalación y configuración del producto, la descripción de los procedimientos y los diagramas y documentos requeridos, ya sea para llevar a cabo alguna modificación en el producto o expandir sus funciones.

La actualización de la documentación en el repositorio es necesaria para generar un registro, que posteriormente pueda ser útil a cada uno de los integrantes del equipo como base de conocimiento para futuros proyectos de desarrollo de software. También es necesaria para que los docentes puedan llevar a cabo una retroalimentación objetiva. La práctica se muestra en la Figura 3.25, la tabla 3.23 muestra la plantilla genérica y la tabla de herramientas recomendadas se muestra en la figura 3.24

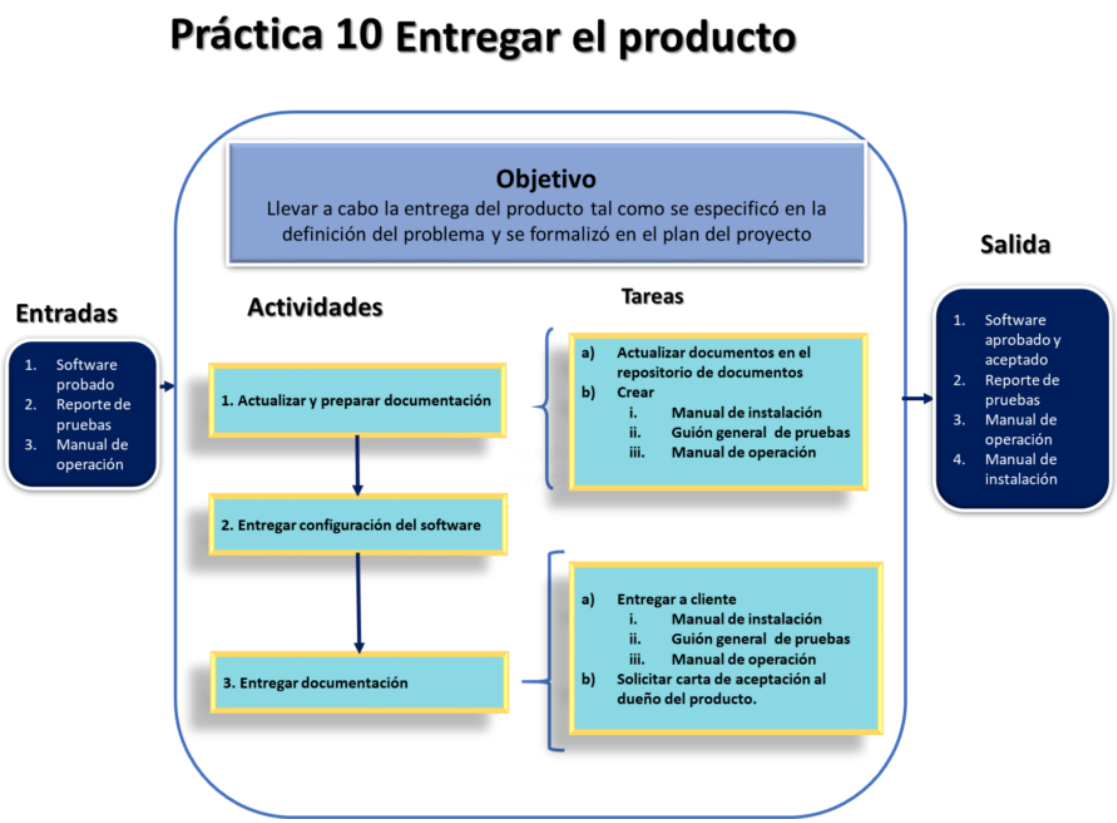



Figura 3.25 Práctica 10: Entregar el producto (de creación propia).

## Plantilla genérica. Práctica 10. "Entregar el producto"

Tabla 3.23 Práctica 10: Plantilla genérica (de creación propia)

| Plantilla de Práctica No. 10 "Entregar el producto" | |
|---|---|
| Actividad | Elementos o subactividades |
| 1. Documentar el manual de instalación | a) Actualizar documentos en el repositorio de documentos<br>b) Preparar<br>i. Manual de instalación<br>ii. Guión general de pruebas<br>iii. Manual de operación |
| 2. Entregar configuración del software | De acuerdo con el protocolo de entrega se debe instalar, configurar, instalar las herramientas necesarias y desplegar el producto de software en el equipo, previamente acordado con el dueño del producto en el Plan del proyecto |
| 3. Entregar documentación | c) Entregar al dueño del producto<br>i. Manual de instalación<br>ii. Guión general de pruebas<br>iii. Manual de operación<br>d) Solicitar carta de aceptación. |

## Herramientas recomendadas. Práctica 10. "Entregar el producto"

Tabla 3.24 Herramientas de la práctica 10 (de creación propia).

| Herramientas Práctica No. 10 "Entregar el producto" | |
|---|---|
| Herramientas recomendadas | Propósito |
| 1. Guión general de pruebas | En las prácticas 7 y 8 se llevaron a cabo las pruebas de componentes unitarios y de integración respectivamente, para esta práctica se preparará un conjunto de casos de prueba "Guión general de pruebas", para verificar los criterios de aceptación de cada Historia de usuario, ver tabla 3.25 |

## Base Conceptual. Práctica 10. "Entregar el producto"

### *Manual de instalación*

El manual de instalación tiene el propósito de describir los componentes de hardware, de software y la forma de preparar el ambiente de trabajo, así como el despliegue del producto en el equipo acordado con el dueño del producto. El manual de instalación se prepara a partir de la documentación de las diferentes prácticas, definición del problema, y arquitectura y diseño.

### *Guión de pruebas*

El guión de pruebas es para el producto de forma global, se crea a partir a las Historias de usuario y sus criterios de aceptación, en la figura 3.26 se muestra un formulario para la Historia de usuario "Autenticar" y en la tabla 3.25 se muestra el guión de pruebas para ésta Historia de usuario.

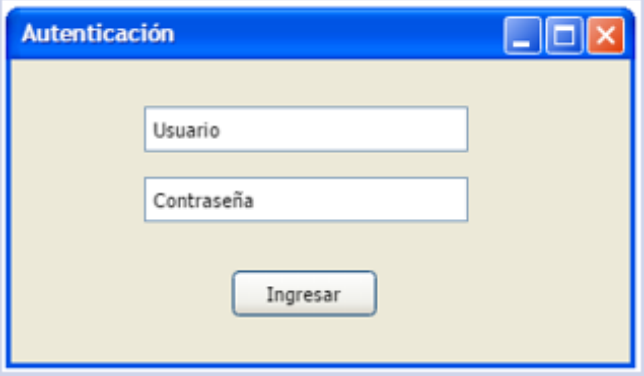



Figura 3.26 Práctica 10: Formulario para la Historia de Usuario "Autenticar" (de creación propia)

H.U: Autenticar (Usuario: admin y Contraseña: admin123 existe en la base de datos)

Tabla 3.25 Guión de pruebas la H.U “Autenticar” (de creación propia)

| Usuario | Contraseña | Resultado esperado | Resultado obtenido |
|---|---|---|---|
| Admin | admin123 | Pantalla de Bienvenida de administrador | |
| Vacío | vacío | Mensaje: Campos vacíos, llenar campos | |
| 12324234 | admin123 | Mensaje: Campo de usuario, no válido | |
| Admin | ()”#$&/ | Mensaje: Campo de contraseña , no válido | |
| Admin | admin789 (no existe en la BD) | Mensaje: Usuario o contraseña no existente | |
| smin (no existe en la BD) | admin123 | Mensaje: Usuario o contraseña no existente | |

El guión de prueba se usa para que el dueño del producto pruebe cada una de las Historias de usuario. El reporte de pruebas del producto, es el resultado del guión de pruebas revisado por el dueño del producto. Si hubiera algún criterio que no se cumple, se corrige y el dueño del producto lo verifica. La configuración del guión de pruebas se muestra en la figura 3.27

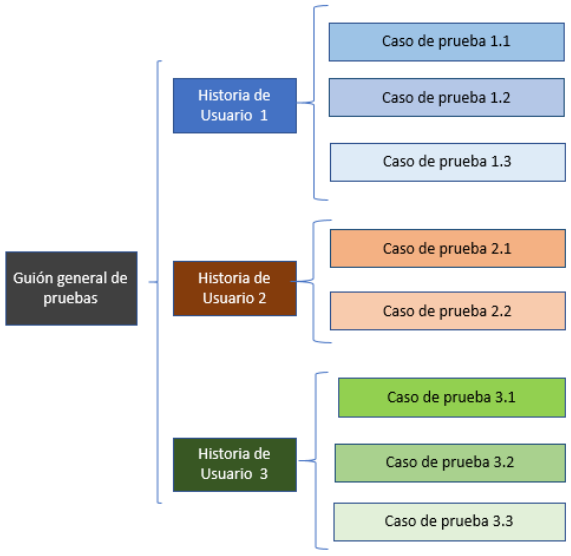



Figura 3.27 Práctica 10: Configuración del guion de pruebas (de creación propia)

## Práctica 11. Hacer retrospectiva del proyecto

La retrospectiva del proyecto es una reunión con el Scrum Master y el equipo de trabajo que se realiza para reflexionar sobre el trabajo realizado, y de esta forma obtener las lecciones aprendidas que pueden ayudar a los miembros del equipo en proyectos posteriores. El diagrama de la práctica se muestra en la figura 3.28. Su plantilla genérica se muestra en la tabla 3.26 y las herramientas a utilizar en la tabla 3.27

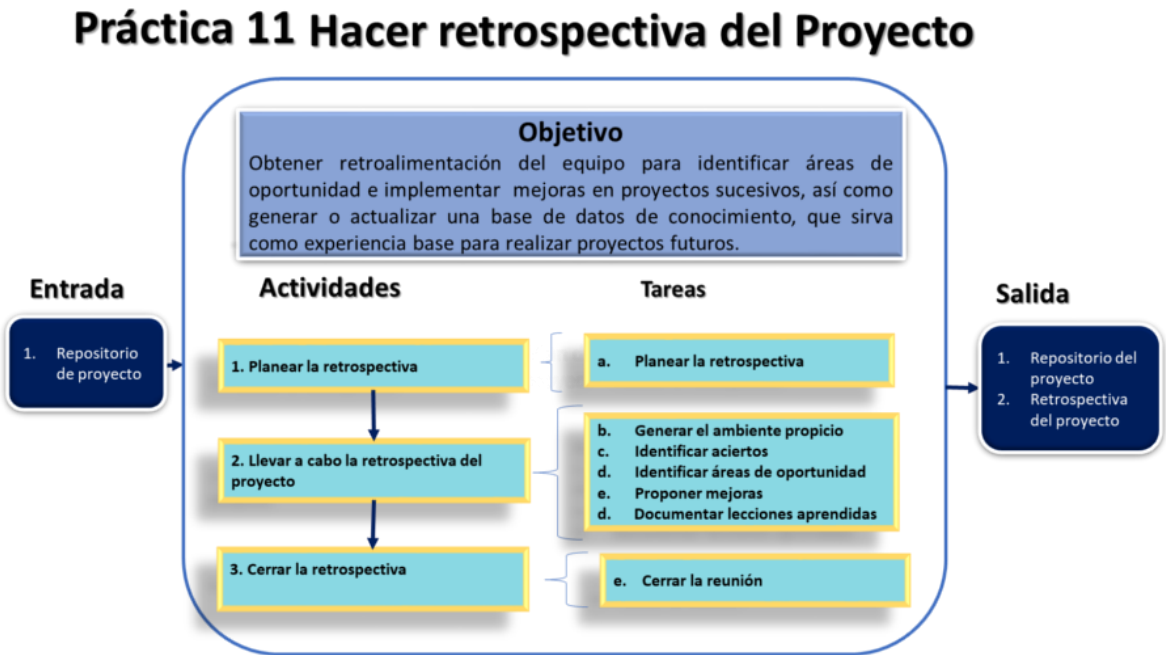



Figura 3.25 Práctica 11: Hacer retrospectiva del proyecto (de creación propia)

## Plantilla genérica. Práctica 11. "Hacer retrospectiva del Proyecto"

Tabla 3.26 Práctica 11: Plantilla genérica (de creación propia)

| Plantilla de Práctica No. 11 "Hacer retrospectiva del Proyecto" | |
|---|---|
| Actividad | Elementos o subactividades |
| 1. Planear la retrospectiva | a) Planear la retrospectiva |
| 2. Llevar a cabo la retrospectiva del proyecto | b) Generar el ambiente propicio<br>c) Identificar aciertos<br>d) Identificar áreas de oportunidad<br>e) Proponer mejoras<br>f) Documentar lecciones aprendidas |
| 3. Cerrar la retrospectiva | g) Cerrar la reunión |

## Herramientas recomendadas. Práctica 11. "Hacer retrospectiva del Proyecto"

Tabla 3.27 Herramientas de la práctica 11 (de creación propia)

| Herramientas Práctica No. 11 "Hacer Retrospectiva del proyecto" | |
|---|---|
| Herramientas recomendadas | Propósito |
| 1. Reunión de retrospectiva del proyecto | Es una reunión con el Scrum Master, y el equipo de trabajo. En esta reunión se analizan los aciertos, las áreas de oportunidad y las mejoras potenciales, para futuros proyectos. Durante la reunión se documentan las lecciones aprendidas. |
| 2. Técnica para llevar a cabo una | Lo mismo que para llevar a cabo una retrospectiva de Sprint, existen muchas técnicas para llevar a cabo la retrospectiva entre las más conocidas está "Explorador-Comprador-Vacacionista-Prisionero" y "El Barco de vapor". Pero existe una lista bastante grande de técnicas que pueden ser usadas, dependiendo de las características de cada equipo de trabajo. |

## Base Conceptual. Práctica 11. "Hacer retrospectiva del proyecto"

La retrospectiva del proyecto al igual que en las retrospectivas de Iteración. Se deben contestar las siguientes preguntas.

- ¿Qué se hizo bien?
- ¿Cuáles son las áreas de oportunidad?
- ¿Qué se puede hacer para mejorar?

Según Diana Larsen y Ester Derby, una buena retrospectiva suele seguir las siguientes 5 fases: (Derby & Diana, 2006).

1. **Preparar el escenario**: establecer el objetivo, dar tiempo a las personas para "llegar" y generar un ambiente propicio.
2. **Recolectar datos**: ayudar a todos a "acordarse", obtener información(tomando en cuenta distintos puntos de vista).
3. **Obtener la causa**: entender porqué las cosas ocurrieron cómo ocurrieron, identificar tendencia, mirar el bosque y no el árbol.
4. **Proponer mejoras**: proponer acciones concretas para mejorar en futuros proyectos.
5. **Cerrar la retrospectiva**: agradecer la participación de todos, cerrar la reunión.

Cabe señalar que también se pueden utilizar las mismas técnicas que se utilizan en una retrospectiva de Iteración. (Caroli & Caetano, 2015)

# Capítulo 4. Prueba y evaluación del método “SOFUNI”

Para probar el Método SOFUNI, éste se implementó con el grupo 2452 de la licenciatura en Informática de la FES Cuautitlán en la materia de Informática IV, análisis y diseño de sistemas II, clave 0402, durante el semestre 2019-II, para lo cual se formaron 4 equipos; de los cuales: 2eran de 4 integrantes; uno de 5 integrantes, y otro, de 3 integrantes. A cada equipo se le asignó un enunciado genérico, en el cual un cliente real solicita un producto de software.

La implementación de las prácticas, inicialmente la planeación quedó como se muestra en la figura 4.1

Tabla 4.1 Calendario de prácticas (de creación propia)

| | Calendario de prácticas | |
|---|---|---|
| Semana | Fecha (Período 2019-II) | Práctica o actividades a realizar |
| 1 | 28 de Enero a 1 de Febrero | Encuadre, introducción a la asignatura |
| 2 | 4 a 8 de Febrero | Práctica 1: Formar el equipo de trabajo |
| 3 | 11 a 15 de Febrero | Práctica 2: Definir el Enunciado del trabajo |
| 4 | 18 a 22 de Febrero | Práctica 3: Elaborar el Plan del Proyecto |
| 5 | 25 de Febrero a 1 de Marzo | |
| 6 | 4 a 8 de Marzo | Práctica 4: Realizar análisis de requisitos |
| 7 | 11 a 15 de Marzo | Práctica 5: Definir arquitectura y diseño |
| 8 | 18 a 22 de Marzo | |
| 9 | 25 a 29 de Marzo | Práctica 6: Planear iteración<br>Práctica 7: Construir software<br>Práctica 8: Integración y pruebas de componentes<br>Práctica 9: Cerrar la iteración |
| 10 | 1 a 5 de Abril | |
| 11 | 8 a 12 de Abril | |
| 12 | 22 a 26 de Abril | |
| 13 | 29 de Abril a 3 de Mayo | |
| 14 | 6 a 10 de Mayo | |
| 15 | 13 a 17 de Mayo | Práctica 10: Entregar el producto |
| 16 | 20 a 24 de Mayo | Práctica 11: Hacer retrospectiva de proyecto |

Sin embargo, la planeación inicial se retrasó una semana debido a situaciones más allá de nuestro alcance, tales como amenazas de bomba y paro laboral, por lo que la retrospectiva del proyecto se llevó a cabo en la semana del 27 al 31 de mayo de 2019.

Durante el proceso de prueba del Método SOFUNI, se hicieron varios ajustes a las prácticas, como aumentar o disminuir actividades o cambiar el orden de las mismas, también se realizaron algunos cambios en la redacción y se propusieron nuevas herramientas.

La evaluación del método se hizo mediante un comparativo de los resultados obtenidos con un grupo de la misma asignatura en el semestre 2018-II a quienes se les solicitó la elaboración de un producto de software a desarrollar durante el semestre, para lo cual se formaron tres equipos; uno de 4 personas; uno de 5 personas, y por último uno de 3 personas, se les dio un enunciado básico sobre el producto y un cliente real con quien debían trabajar durante el semestre y se les proporcionó la base teórica sobre Gestión de Proyectos basándonos en el PMBOK®, las herramientas correspondientes a UML, las arquitecturas MVC y tres capas, así como la teoría necesaria sobre diseño de bases de datos relacionales; pero no se puso restricción alguna a cerca de la metodología a utilizar, al inicio del semestre se establece una revisión del avance del proyecto, en la semana 8 y la entrega del producto de software al cliente y la retroalimentación y documentación se llevó a cabo en la semana 16 del semestre. Primero se exponen los resultados de grupo 2018-II, sin aplicar el Método SOFUNI mediante la tabla 4.2

Tabla 4.2 Resultados obtenidos en el desarrollo de software sin la aplicación del Método SOFUNI con el grupo de prueba 2018-II (de creación propia)

| Grupo "2018-II" (sin Método SOFUNI) | | | | | |
|---|---|---|---|---|---|
| **Equipo** | **No. de integrantes** | **Producto solicitado** | **Requisitos cumplidos** | **Entregado en tiempo** | **Satisfacción del cliente** |
| 1. ASAEQ | 4 | Página Web para la profesora María Guadalupe Hernández Santiago, ella requería que sus alumnos de 2 dos asignaturas diferentes, "ecuaciones diferenciales y Variable compleja" pudieran entrar y descargar el material de sus respectivos temarios y las tareas. | Requisitos funcionales al 100% ; sin embargo no cumplió con los requisitos no funcionales de usabilidad y seguridad | Entrega en la semana 16 | La profesora expresó que a pesar de cumplir con los requisitos funcionales la interfaz, resultó ser poco intuitiva, y le resultaba complicado utilizarla, además que presentaba el problema de seguridad, ya que los alumnos de ambas asignaturas podían a ingresar al material de las dos asignaturas. |
| 2. ONE | 5 | Sistema de inventario para la zapatería Charly. El cliente requiere un sistema para llevar el control de ventas y existencias de sus productos en el almacén. | Los requisitos funcionales se terminaron al 70% aproximadamente | Entrega semana 16 | El cliente expresó que el módulo de consultas y reportes, que no fue realizado era muy importante y aunque el sistema le ayudaría en la organización de sus productos, todavía tendría que hacer varias operaciones en forma manual, en cuanto a la interfaz no fue de su completo agrado. Tenía demasiados colores |
| 3. APIEF | 3 | Una aplicación de escritorio para ayudar al Profesor Julio Moisés Sánchez, a comprobar algunas funciones que enseña a sus alumnos en la asignatura de matemáticas financieras Interés simple, interés compuesto, descuento y depreciación | El equipo cumplió completamente con los requisitos funcionales y no funcionales del cliente | La aplicación se completó dos semanas después del compromiso (semana 18) | Los requisitos funcionales se cumplieron en su totalidad, sin embargo; el profesor pidió cambios en la interfaz al final; por lo que el proyecto se retrasó 2 semanas. Una vez terminado el proyecto, el cliente estuvo complacido y expresó su disposición para empezar a utilizarlo |

Algunas observaciones generales sobre los equipos de trabajo en el desarrollo de su producto son las siguientes:

- Solo tuvieron una o dos entrevistas con sus clientes, previas a la entrega de su producto, lo que propició que en el caso de los equipos ONE y ASAEQ, la interfaz no fuera del agrado del cliente; en el caso del equipo ONE, ni siquiera hubo un completo entendimiento sobre la prioridad de las funciones requeridas. En cuanto al equipo APIEF, tuvieron que modificar la interfaz, lo que ocasiono un retraso en la entrega del producto.
- Al llevar a cabo la revisión del producto, en la semana 8, los tres equipos apenas estaban terminando el análisis de requisitos; por lo que de la semana 13 en adelante, los equipos trabajaron a marchas forzadas.

- En los tres equipos hubo inconsistencias en la documentación, todos realizaron cambios; ya sea en los requerimientos, en el diseño de interfaces, en la estructura de la base de datos, etc. Sin embargo; al comparar el producto de software con la documentación entregada, no coincidía. Otra constante, es que ninguno de los casos la documentación entregada estaba completa. Los resultados del grupo 2019-II, se muestran en la siguiente tabla 4.3

Tabla 4.3 Resultados obtenidos en el desarrollo de software aplicando el Método SOFUNI con el grupo de prueba 2019-II (de creación propia)

| Grupo "2019-II" (con Método SOFUNI) | | | | | |
|---|---|---|---|---|---|
| **Equipo** | **No. de integrantes** | **Producto solicitado** | **Requisitos cumplidos** | **Entregado en tiempo** | **Satisfacción del cliente** |
| 1. UNITY TEAM | 5 | Aplicación para registro y control de usuarios y pagos del gimnasio Bushido Academy (Cliente: Emanuel Osorno "Encargado") | Requisitos al 90%, quedó pendiente la consulta de asistencia de socios (estadística) | Entrega semana 17 | El cliente quedó satisfecho con el trabajo realizado por e equipo, en la interfaz, se utilizaron los colores y el logotipo del gimnasio |
| 2. NITTIOTREMO | 5 | Aplicación para la gestión de artículos en la papelería "El sol" Incluye la ubicación de artículos, consulta de precios e inventario (cliente: Juan Torres Valencia "Dueño") | Se cumplió con todos los requisitos acordados con el cliente | Entrega semana 17 | Al inicio del desarrollo el equipo modificó varias veces la interfaz, ya que no lograba la aceptación del cliente, sin embargo; al realizar la entrega del producto, el cliente se mostró muy contento con el resultado, el producto se instalo en el equipo del cliente, quedando listo para su uso. |
| 3. SOLUTIONS INTELLIGENTS | 4 | Una aplicación de escritorio para la administración de empleados y medicamentos en una farmacia, se requiere un punto de venta ,con actualización de inventarios y consultas (cliente: Guadalupe Monroy Rodríguez "Dueña") | El equipo cumplió completamente con los requisitos funcionales y no funcionales del cliente | Entrega semana 17 | Al igual que el equipo Nittiotremo, el equipo tuvo dificultades para llegar a un acuerdo con el cliente en cuanto a la interfaz, sin embargo; a la entrega del producto el cliente externó que le gustó el resultado final, haciendo énfasis en la sencillez dela interfaz |
| 4. DEVELOOPERS | 4 | Reingeniería a página Web para preinscripciones a talleres culturales de la Facultad de Estudios Superiores Cuautitlán . (cliente: Alejandro Emmanuel Suberza Luque " Director de actividades culturales FESC") | El equipo cumplió completamente con los requisitos funcionales y no funcionales, superando las expectativas del cliente | Entrega semana 17 | El cliente estuvo muy complacido con el trabajo realizado por el equipo, la interfaz realizada por los alumnos cumplió con las expectativas, así como la usabilidad, con respecto a la página original, también se agregaron funcionalidades para mejorar la experiencia de los usuarios, el equipo se comprometió a realizar otros proyectos en la modalidad de Servicio Social. |

Observaciones

- El Método SOFUNI requiere de estar en contacto constante con el cliente, lo que propició que el producto generado por los equipos realmente cumpliera con los requerimientos de sus respectivos clientes, los equipos Nittiotremo y Solutions Intelligens tuvieron que hacer modificaciones al inicio del proyecto, a diferencia del equipo APIEF del grupo 2018-II, que realizó la modificaciones al final, por lo que el costo en tiempo y trabajo para los equipos Nittiotremo y Solutions Intelligens fue mucho menor. Cabe señalar que la función del cliente (Dueño del producto) fue muy importante, ya que mantuvo al equipo enfocado en los requerimientos.
- El trabajo que los equipos realizaban cada Iteración es muy fácil de monitorear para el profesor ya que la documentación se va actualizando en el repositorio, y el tablero Kanban permite ir siguiendo la actividad de los equipos y sus integrantes, lo que permite que la retroalimentación sea constante.
- La documentación se realiza durante todo el semestre, por lo que al final los equipos solo se preocupan de preparar una carpeta con las últimas versiones, y las inconsistencias que se presentan con respecto al producto son menores. Los manuales de instalación, de operación y el guion general de las pruebas, se obtienen de la documentación de las prácticas.

**Conclusiones de la implementación del método SOFUNI.**

Al implementar el Método SOFUNI, la comunicación con el cliente se mantuvo constantemente durante el proyecto, lo propició que el equipo tuviera en cuenta las requerimientos del cliente en las reuniones diarias, los cambios se solicitados se realizaron durante todo el proyecto, y no al final, lo cual reduce considerablemente el costo para el equipo tanto en tiempo como en esfuerzo, por lo que el grupo que siguió el Método SOFUNI obtuvo mejores resultados en cuanto a satisfacción del cliente como una mejora en los requisitos funcionales y no funcionales entregados, en cuanto a la fecha de entrega no hubo retrasos adicionales al retraso por aplicación de medidas de inherentes a las contingencias presentadas. Con el Método SOFUNI, la documentación se genera o actualiza durante todo el semestre, por lo que las inconsistencias son menores que cuando no se hace uso del método.

En cuanto a la aceptación del método, por parte de los estudiantes, se obtuvieron buenos resultados; en la encuesta realizada, algunos de los argumentos expresados fueron:

- Que el Método SOFUNI les proporciona una guía para llevar a cabo los procesos de Gestión de Proyectos y desarrollo de software.
- La experiencia de estar en contacto con el cliente les mostró que la mejor manera de crear un producto, que el cliente usará, es que realmente cumpla con sus expectativas.
- La creación de la documentación se lleva a cabo durante el semestre, lo que les proporciona una ayuda y no una carga.
- El control de versiones les permitió llevar un mejor control de los avances del producto
- Seguimiento, es mucho más sencillo usando las herramientas como el tablero de Kanban y las juntas diarias.

El método SOFUNI También se está probando de forma individual con un alumno cuya tesis consiste en la realización de un producto de software. Hasta ahora realizó la práctica 2 “Enunciado del trabajo”, la práctica 3 “Elaboración del plan del proyecto”, la práctica 4 “Realizar Análisis de requisitos” y 5. “Definir Arquitectura y diseño”. Cabe señalar que la práctica 1 “Formar el equipo de trabajo” se omite cuando el desarrollo del producto de software es individual.

Durante el proceso de la implementación del método surgieron situaciones que propiciaron la modificación de la estructura original de las prácticas propuestas para el Método SOFUNI. A continuación, se listan algunos de los cambios realizados a las prácticas originales:

- Inicialmente en la práctica 2 “Enunciado del trabajo”, no se pedía a los alumnos la elaboración del Mapa de impacto, sin embargo; tenían dificultades para definir el alcance del proyecto, por lo tanto; se propuso como recurso visual que facilitará la definición del alcance.

- En la práctica 3 “Elaborar Plan del Proyecto”, una de las actividades es hacer el cronograma para la Gestión del Proyecto, originalmente no se especificaba que actividades debían programarse; situación que causó confusión; por lo que se agregaron las actividades a programar en la plantilla genérica.
- También en la práctica 3, otra de las actividades es priorizar las Historias de usuario, para determinar que Historias de usuario, deben realizarse en cada Iteración, en este punto se presentó la dificultad de priorizar tomando en cuenta las dependencias y las prioridades que manifestó el dueño del producto, por esta razón la plantilla genérica se modificó para solicitar el Mínimo Producto Viable (MPV) y la creación de versiones incrementales para cada Iteración, con esa información se facilitó la creación del Backlog inicial del producto en la práctica 4.
- En la práctica 6, se modificó el formato del Backlog de la Iteración, se agregó la columna de tiempo real, junto a la columna de tiempo estimado, para hacer más sencilla la evaluación del avance en las Juntas Diarias.
- En varias prácticas, se modificó el orden de las actividades y/o las tareas.
- Se agregó a la Base de conocimiento algunos conceptos, otros se quitaron o se modificaron.

El Método SOFUNI, presentado en este documento, ya incluye las mejoras generadas a partir de los cambios realizados durante la etapa de prueba.

# Capítulo 5. Desarrollo de la Herramienta digital para apoyar la implementación del Método SOFUNI

"*Los tiempos están cambiando de forma acelerada y tanto los docentes como los estudiantes somos conscientes de que la escuela no puede seguir dando la espalda a las nuevas formas culturales, de comunicación, de difusión y acceso a la información que generan las tecnologías digitales*" (Area, 2009)

### Objetivo de la Herramienta digital

Proporcionar a los docentes y a los estudiantes de licenciaturas relacionadas con el desarrollo de productos de software, una herramienta de fácil acceso.

### Descripción

Para los docentes la herramienta le permitirá:

- Monitorear el avance de los equipos de trabajo bajo su dirección, teniendo acceso a la base de datos, donde se almacenan las prácticas, para una posterior retroalimentación.
- Tener en una base de datos toda la documentación generada por los equipos a través de las plantillas en la herramienta, incluyendo las versiones que se van generando.
- Gestionar los equipos e integrantes de los equipos.

En el caso de los estudiantes la herramienta les proporciona:

- Una guía digital, que muestra una introducción al Método SOFUNI y el conjunto de prácticas que lo integran, para cada práctica, se muestra su esquema, previamente explicado en Capítulo 2.

- Una base conceptual digital, para obtener los conceptos necesarios y, una serie de enlaces a páginas relacionadas para profundizar en el conocimiento, de la Ingeniería de Software, en temas relacionados con la práctica correspondiente.
- Un conjunto de herramientas recomendadas para realizar la práctica correspondiente.
- Un formulario (plantilla) equivalente a cada plantilla genérica, para ir generando la documentación a medida que se va avanzando en el proyecto.
- La posibilidad de consultar su documentación guardada a través de sus formularios.

El diagrama de casos de uso muestra las funciones del Método SOFUNI, para sus roles de “Docente” e “Integrante de equipo” (ver figura 5.1).

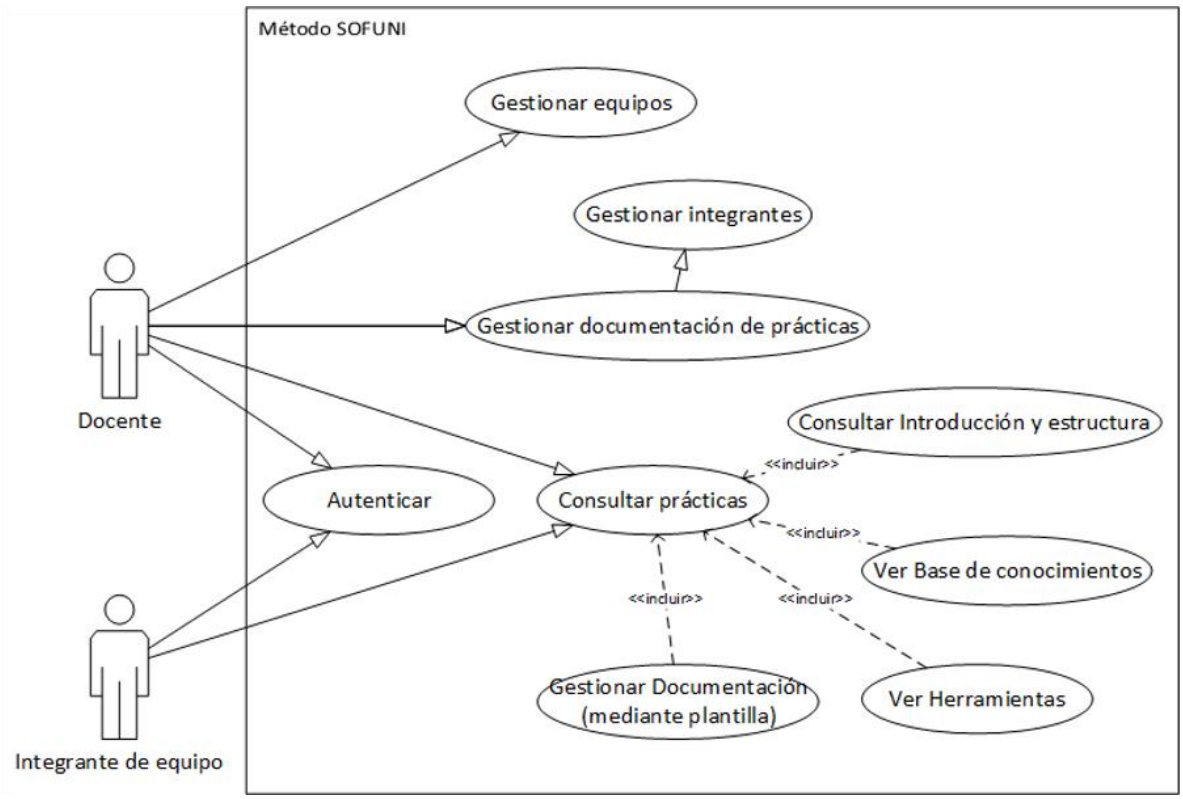



Figura 5.1 Diagrama de Casos de uso (de creación propia)

## Presentación de la Herramienta digital del Método SOFUNI

La página de inicio del Método SOFUNI se muestra en la figura 5.2, en la cual se solicita la autenticación del usuario. Una vez que se ingresa correctamente el “usuario” y “contraseña”, se muestra la página de Introducción al Método (figura 5.3).


Figura 5.2 Página de inicio, solicita la autenticación del usuario (de creación propia)


Figura 5.3 Página de introducción al Método SOFUNI(de creación propia)

En la página de introducción se hace una breve descripción del Método SOFUNI.

El menú de prácticas muestra de la 1 a la 11 (figura 5. 4)


Figura 5.4 Opciones del menú prácticas (de creación propia)

Para cada práctica se tiene cuatro opciones: la primera, muestra la estructura y descripción general de la práctica; la segunda muestra la plantilla en forma de formulario que almacena la información en una base de datos; la tercera, muestra las herramientas recomendadas para realizar las diferentes actividades de la práctica, y la cuarta muestra los conceptos básicos requeridos para llevar a cabo las actividades. Ver figura 5.5

Se debe seguir el orden de las prácticas 1, 2, … 11. Ya que la información generada en una práctica, es requerida para la siguiente, sin embargo; siempre es posible regresar a una práctica anterior para hacer modificaciones y generar una nueva versión.


Figura 5.5 Opciones de práctica 1(de creación propia)

La primera opción muestra la descripción de la práctica y el diagrama que indica las entradas, el objetivo, las actividades con sus respectivas tareas y la salida generada por la práctica. (figura 5.6)


Figura 5.6 Descripción y esquema de la práctica 1(de creación propia)

La opción “Base de conocimientos” muestra los conceptos y teoría para la realización de la práctica (figura 5.7)


Figura 5.7 Base de conocimientos práctica 1 (de creación propia)

Las herramientas que se proponen para cada práctica tienen la finalidad de ayudar a los integrantes del equipo a realizar las actividades correspondientes, de manera sencilla, pueden ser aplicaciones, técnicas, sugerencias, también se proporcionan enlaces a páginas relacionadas para perfeccionar los conocimientos de Ingeniería de Software, correspondientes a la práctica en cuestión. Ver figura 5.8


Figura 5. 8 Herramientas. Práctica 1 (de creación propia)

La plantilla se refiere al formulario que permite ingresar la información de la práctica en una base de datos. Para la práctica 1. Por ejemplo, se muestran los equipos registrados, se puede, registrar un nuevo equipo, se puede modificar el registro de uno existente, borrar el registro o ingresar integrantes al equipo. Ver las figuras 5.9 y 5.10

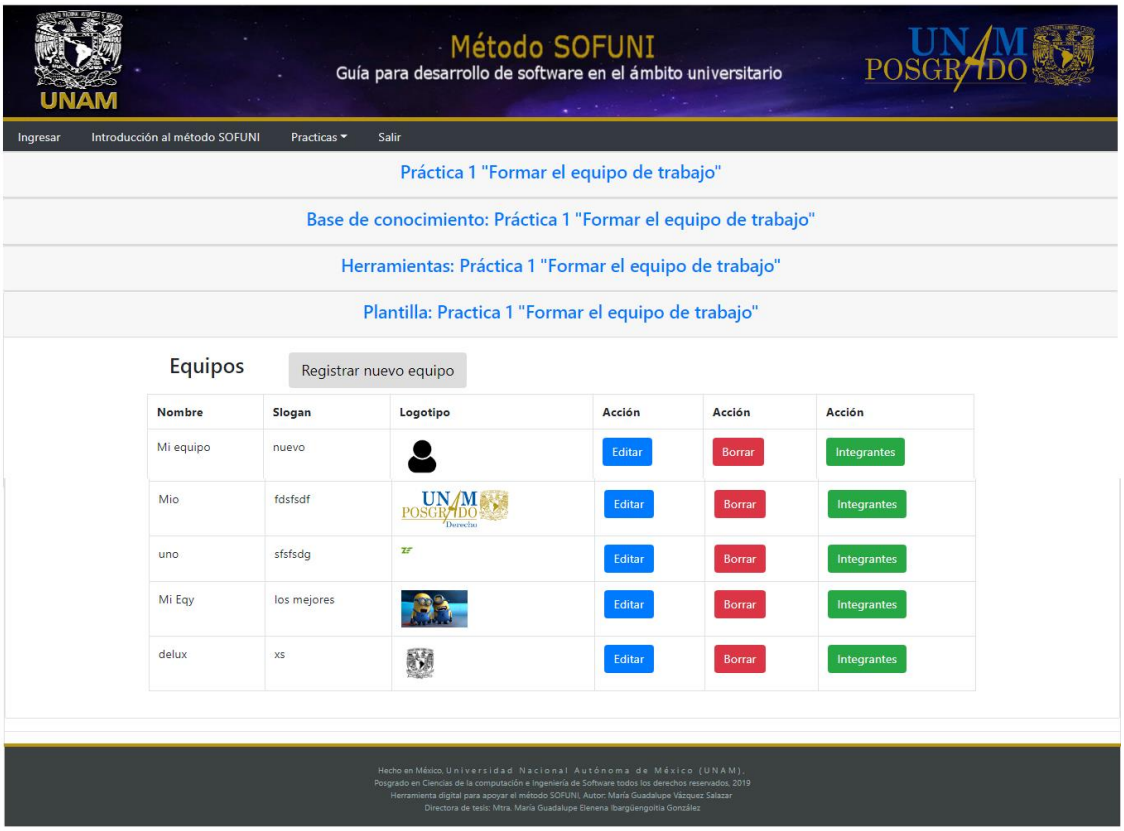



Figura 5.9 Equipos registrados


Figura 5.10 Registro de nuevo equipo (de creación propia)

La figura 5.11 muestra el formato para editar el registro de un equipo


Figura 5.11 Formulario para editar un registro de equipo ya existente. (de creación propia)

La figura 5.9 muestra el botón de integrantes; cuando se presiona, se muestra la lista de integrantes del equipo seleccionado con las opciones de ingresar un nuevo integrante, modificar uno ya existente, o bien borrarlo, como se muestra en la figura 5.12. El formulario para un nuevo registro se muestra en la figura 5.13.


Figura 5.12 Formulario que muestra los integrantes registrados en un equipo. (de creación propia)

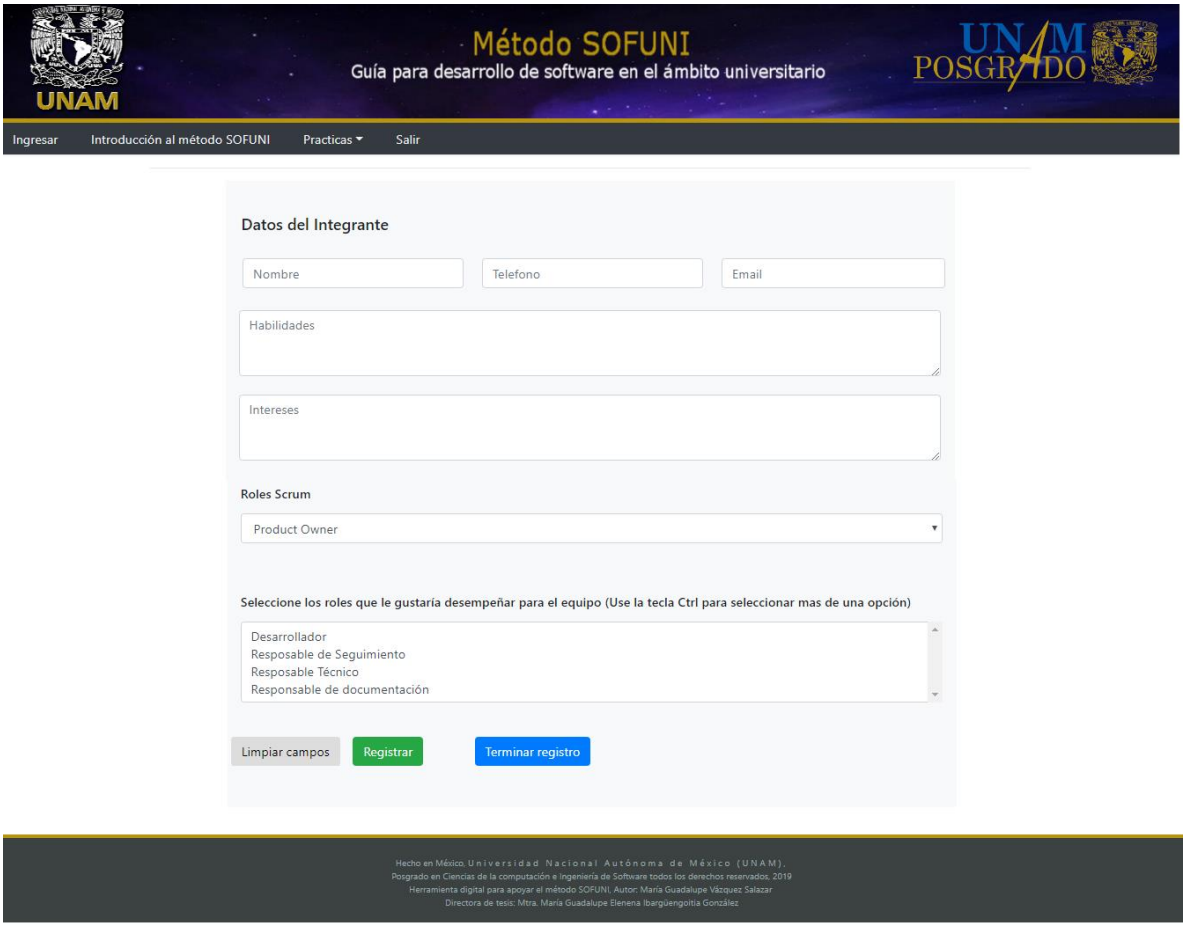



Figura 5.13 Formulario para el registro de un nuevo integrante del equipo (de creación propia)

Todas las prácticas del método tienen la misma estructura que se mostró para la práctica 1: La descripción de la práctica, la base de conocimientos en la que se define la base teórica para realizar la práctica, las herramientas sugeridas que pueden ser aplicaciones, técnicas, ejemplos y enlaces a páginas relacionadas; y por último, plantillas, que capturan la información necesaria en una base de datos, para que el docente pueda acceder a ésta y dar seguimiento al avance de los equipos en sus respectivos proyectos; también para que los integrantes del equipo puedan consultar en cualquier momento su documentación generada, o generar una nueva versión, de tal forma que la documentación se va generando y actualizando durante la realización del proyecto.

## Trabajo a futuro

La herramienta digital tiene áreas de oportunidad, para ser implementadas a futuro, una de ellas es, agregar la funcionalidad para mostrar en formato PDF las prácticas y poder imprimirlas. En el periodo 2020-II será probada con un grupo de la carrera de Informática, con la asignatura de Informática IV, se espera que, durante la prueba, se detecten nuevas áreas de oportunidad para continuar mejorando la herramienta, para su implementación, en asignaturas relacionadas con el desarrollo de productos de software.

# Conclusiones

Para construir productos de software de calidad se plantea la necesidad de lograr una metodología disciplinada mediante métodos, herramientas y procedimientos que proporciona la Ingeniería de Software (Cataldi, Pessacc, García Martínez, & Lage, 2007)

El Método SOFUNI está integrado por 11 prácticas basadas en el estándar ISO/IEC 29110 en su perfil básico, que es un estándar para pequeñas organizaciones(1 a 25 personas), por lo tanto, debió ser adaptado para implementarse en un ambiente estudiantil universitario, atendiendo a sus dos procesos: gestión de proyectos y desarrollo de software, el método fue pensado para aplicarse bajo un marco de .desarrollo ágil, para lo cual se seleccionó Scrum, ya que actualmente es el marco de desarrollo ágil, .más aceptado. (CollabNet, 2018).

Para expresar al Método SOFUNI se utilizó como herramienta Kuali Beh, ya que es necesario que sea coherente y suficiente; es decir, que la salida de cada práctica proporcione la entrada de la siguiente, y que el número de prácticas sea suficiente para lograr que los estudiantes desarrollen software de calidad en menos de un semestre. Cada práctica consta de una serie de actividades relacionadas entre sí, además de la definición de las prácticas, se proporciona a los estudiantes una base conceptual, que consiste en teoría necesaria para poder llevar a cabo cada una de las actividades definidas en cada práctica, también se proporciona una serie de herramientas consistentes en aplicaciones, técnicas, ejemplos, sugerencias, etc., que le ayuden a realizar las actividades, por último, una plantilla que sirve como guía para que cada alumno lleve a cabo su práctica correspondiente.

Para la implementación y evaluación del método, se tomó como grupo de prueba el grupo 2452 de la licenciatura en Informática de la FES Cuautitlán del cual sus integrantes cursaron la asignatura de Informática IV, durante el semestre 2019-II, se pidió a los estudiantes formar equipos de trabajo, y se le asignó a cada equipo un cliente que tenía una solicitud de desarrollo de software, cada equipo realizó cada una de las prácticas del Método SOFUNI, los resultados obtenidos se compararon con el mismo grupo del semestre 2018-II, este grupo también se organizó en equipos, se le asignó un cliente con una solicitud de

software, pero sin indicarles algún método en particular. A continuación, veremos las diferencias de ambos grupos, el que utilizó un método libre (2018-II) y el que uso el Método SOFUNI (2019-II)

a) Los equipos del grupo 2019-II mantuvieron una relación constante con sus respectivos clientes, por lo que las peticiones de cambios se atendían casi de inmediato, y los clientes al final del proyecto estuvieron satisfechos, según lo expresaron en la entrega del producto. En cuanto a los equipos del grupo 2018-II, solo tuvieron contacto con el cliente al inicio del proyecto, lo que ocasionó que al final del proyecto, los clientes solicitaran cambios y se generará un retraso en la entrega del producto.
b) Los equipos del grupo 2019-II terminaron los requerimientos establecidos en el plan de proyecto, casi en su totalidad, Los equipos del grupo 2018-II en general lograron satisfacer menos requerimientos, y además tuvieron retrasos en el tiempo de entrega.
c) La documentación de los equipos del grupo 2019-11 resultó más completa y ordenada que la de su contraparte del 2018-II.

Cabe señalar que durante la experiencia con el Método SOFUNI, se llevaron a cabo varios ajustes al método, como: cambiar de orden algunas actividades, alguna herramienta o proponer una adicional.

En cuanto a la herramienta digital, se probará con un grupo en el período 2020-II, en la carrera de Informática con la asignatura de Informática IV, para identificar y aprovechar las áreas de oportunidad que permitan realizar las mejoras pertinentes;

# Referencias


Ambler, S. W. (2013-2018). *Agile Modeling*. (A. Inc., Productor) Recuperado el 5 de 06 de 2019, de Introduction to the Diagrams of UML 2.X: http://www.agilemodeling.com/essays/umlDiagrams.htm

Area, M. (2009). La competencia digital e informacional en la escuela. (U. d. Laguna, Ed.) Santander. Obtenido de http://files.competenciasbasicas.webnode.es/200000167-814ad8244d/CompetenciaDigital-MArea.pdf

Azdic, G. (2012). *Impact Mapping.* Obtenido de https://www.impactmapping.org/book.html

Bass, L., Clements, P., & Kazman, R. (1998). *Software Architecture in Practice Sei Series In Software Architectures.* Addison Websley.

Buschman, F., Regine, M., Rohnert, H., Sommerland, P., & Stal, M. (1996). *Patters-Oriented Software Architecture:A System o Patterns.* John Wiley & Sons.

Caroli, P., & Caetano, T. (2015). *Ebook Fun Retrospectives: Activities and ideas for making agile retrospectives more engaging.* LeanPub.

Cataldi, Z., Pessacc, R., García Martínez, R., & Lage, F. (2007). Ingeniería de software educativo. *Revista Latinoaméricana de Tecnología Educativa*.

Cervantes, H. (2010). Arquitectura de Software. *Software Gurú*, 32-33. Obtenido de https://sg.com.mx/revista/27/arquitectura-software

Clements, P. (1996). *A Survey of Architecture Description Languages.* Alemania: Proceedings of the International Workshop on.

Cohn, M. (2005). *Agile Estimating and Planning.* Prentice Hall.

Collabnet VersionOne. (2018). *12th annual State of Agile report.* Obtenido de https://explore.versionone.com/state-of-agile/12th-annual-state-of-agile-report-overview

CollabNet, I. (2018). *VersionOne (2018) . 12th annual state of agile development survey.* Atlanta, GA. Recuperado el 06 de 2019, de https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report

Derby, E., & Diana, L. (2006). *Agile Retrospectives: Making good teams great.* Estados Unidos: The Programatic Bookshelf.

Fowler, M. (21 de 02 de 2016). *martin.Fowler.com.* Obtenido de martin.Fowler.com: https://www.martinfowler.com/articles/itsNotJustStandingUp.html

Grenning, J. W. (2011). Recuperado el 5 de 06 de 2019, de https://wingman-sw.com/slides/Beyond-Planning-Poker-v1r1.key.pdf

Hurtado, S. (2006). Representación de la arquitectura de software usando UML. Universidad Icesi-I2T.

Ibargüengoitia, G. M. (2018). *Apuntes Ingeniería de Software Orientada a Objetos.* Posgrado en Ciencia e Ingeniería en Computación, UNAM.

Jacobson, P. N. (2013). La Esencia de la Ingeniería de Software:. *Revista Latinoamericana de Ingeniería de Software*, 71-78.

Jurgen, A. (2010). *Management 3.0.* Addison-Wesley Professional.

Kniberg, H., & Skarin, M. (2010). *Kanban y Scrum –obteniendo lo mejor de ambos.* (D. Plesa, Ed.) Estados Unidos de América: C4Media, editores de InfoQ.com.

Management, Group Object. (2014). *KUALI-BEH: Software Project Common Concepts.Normative Annex n ESSENCE–Kernel and Language for Software Engineering Methods.* Obtenido de http://kualikaans.fciencias.unam.mx/images/kuali/documentos/KUALI-BEH%20v1.1.pdf

*Manifiesto por el Desarrollo Ágil de Software*. (Febrero de 2019). Obtenido de http://agilemanifesto.org/iso/es/manifesto.html

Ministerio de Administraciones Públicas. (2001). *Metodología MÉTRICA Versión 3.* Recuperado el 2 de julio de 2019, de http://administracionelectronica.gob.es

NTP-RT-ISO/IEC TR 29110-5-1-2. (2012). *ISO/IEC 29110.*

Oktaba, H. (2013). ESENCIA y su uso para reforzar la gestión de los proyectos de software. *Software Guru*(46). Obtenido de https://sg.com.mx/revista/46/esencia-y-su-uso-para-reforzar-la-gestion-los-proyectos-software

Patton Jeff, E. P. (2014). *Usser Story Mapping: Discover de Whole Story, Build the Right Product.* Peter Economy.
Paul, C. (1996). *A Survey of Architecture Description Languages.* Alemania: Proceedings of the International Workshop on Software Specification and Design.
Pérez del Castillo, R., García, R. d., Ruíz, G. F., Polo, U. M., & Piattini, V. M. (2018). *Mantenimiento y Evolución de Sisiyemas de información.* España: Ra-Ma.
PMI. (2017). *Guía de los Fundamentos Para la Dirección de Proyectos (Guía del PMBOK®).*
Pressman, R. S. (2010). *Ingeniería del Software. Un enfoque práctico.* Mc-Graw-Hill.
Project Management Institute (PMI). (20 de Febrero de 2019). *PMI Project Management Institute*. Obtenido de https://americalatina.pmi.org/latam/aboutus/whatispmi.aspx
Project Management Institute. (2017). *A guide to the project management body of knowledge.* PMI.
Ramírez Liliana del Carmen, F. F. (2014). Buenas prácticas, Una solución para un mejor desarrollo de software. *Mundo FESC*, *2*(8), 37-47.
Ruiz, E. S. (2012). Marcando la Pauta para las pruebas ágiles. *Software Gurú*. Recuperado el junio de 2019, de https://sg.com.mx/revista/46/marcando-la-pauta-para-las-pruebas-agiles
Rumbaugh, J., Jacobson, I., & Booch, G. (2007). *El Lenguaje Unificado de Modelado, manual de referencia UML 2.0* (Segunda ed.). Madrid: Addison Wesley Pearson.
SCRUMstudy. (2016). *Una guía para el cuerpo de conocimientos de Scrum (Guía SBOK™).*
Silberschatz, A., Korth, H. F., & Sudarshan, S. (2006). *Fundamentos de Bases de datos* (4 ed.). España: McGRAW-HILL.
Sommerville, I. (2004). *Ingeniería de Software* (Séptima ed.). Addison Wesley.
Stevens, H. (2011). Mínimo Producto Viable: ¿Qué es y Para qué? *Software Gurú*, 16-18. Recuperado el 4 de 06 de 2019, de https://sg.com.mx/revista/31
William J. Brown, R. C. (1998). *Antipatterns: Refactoring Software, Architectures, and Projects in Crisis.* John Wiley and Sons.