Programa de la unidad didáctica:
**Desarrollo de software en equipo TSP**

6º Semestre

# Unidad 2. Implementación de TSP

**Clave:**
150930934

Ciudad de México, febrero del 2026

**Universidad Abierta y a Distancia de México**
**UnADM**

## Índice

Unidad 2. Implementación de TSP....2
Presentación de la unidad....2
Logros....3
Competencia específica....2
2.1. Formar equipos de trabajo....2
2.1.1. Documentar propósitos, objetivos y roles del equipo....6
2.1.2. Planear y ejecutar el lanzamiento del proyecto....8
2.2. Ejecutar el trabajo en equipo....14
2.2.1 Elaborar el plan del proyecto....14
2.2.2 Elaborar plan de calidad....18
2.2.3 Elaborar plan de riesgos....27
Cierre de la unidad....32
Para saber más....33
Fuentes de consulta....33

# Unidad 2. Implementación de TSP

## Presentación de la unidad

En el marco de la metodología TSP, la **implementación** se refiere a la ejecución de las actividades, en la cuales se debe tener en cuenta los equipos de trabajo, objetivos, roles, planeación de proyecto, calidad y riesgos.

En la Unidad 2. Implementación de TSP, aprenderás a implementar esta metodología para un proyecto de desarrollo de *software*, conocerás los componentes de la metodología del TSP y aprenderás la forma adecuada para realizar el propósito y los objetivos del proyecto, así como las estrategias de integración de los miembros del equipo y la asignación de los diferentes roles que TSP propone. También conocerás la forma de hacer los documentos para establecer el plan del proyecto, el plan de calidad y el plan de riesgos para llevar por buen camino el proyecto y lograr que los objetivos se cumplan.

## Logros

Al término de esta unidad lograrás:

- Identificar los componentes de la metodología *Team Software Process* (TSP) de acuerdo con la fase del proyecto correspondiente.
- Analizar los productos de trabajo de acuerdo con las funciones correspondientes.
- Elaborar los productos de trabajo de acuerdo con su objetivo.

## Competencia específica

- Analizar los componentes de la metodología *Team Software Process* (TSP) para implementar productos de trabajo en los equipos autodirigidos, mediante la elaboración de documentos.

## 2.1. Formar equipos de trabajo

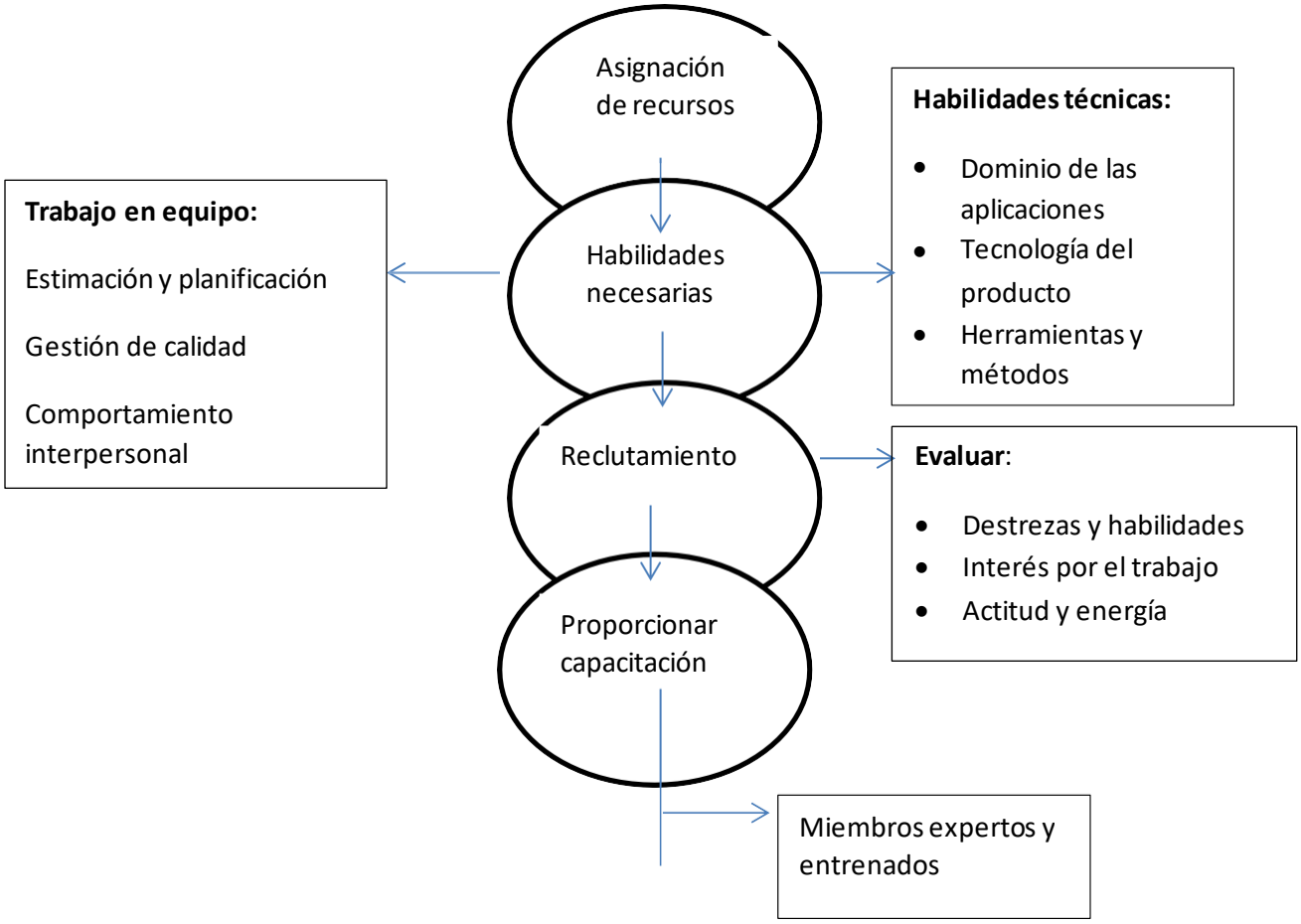
Durante la fase de lanzamiento que TSP propone, es muy importante saber cómo formar un buen equipo de trabajo, no sólo donde exista un ambiente cordial, sino también un equipo que sea capaz, con base en la actitud y aptitud, de llevar el desarrollo del proyecto por buen camino, siempre teniendo presente el objetivo principal del proyecto.

En este tema aprenderás cómo se forman equipos de trabajo adecuados para implementar la metodología TSP en un proyecto de desarrollo de *software*. El establecimiento de un equipo de trabajo se lleva a cabo en la fase de lanzamiento, y para que sea realmente efectivo todos sus miembros deben estar bien capacitados para las actividades que llevarán a cabo, además de trabajar de manera conjunta y así cumplir con los objetivos trazados al inicio.

TSP indica ciertas características que debe cumplir un equipo que trabaje bajo esta metodología (Humphrey, 1999):

- Contar con las habilidades requeridas. Si el proyecto se desarrollará en un lenguaje de programación específico, se buscarán expertos en él.
- Compromiso con el objetivo principal del proyecto a desarrollar.
- Ayuda mutua para solucionar problemas más rápido y, a la vez, fomentar un buen ambiente de trabajo.
- Disciplina en su trabajo.
- Innovación y eficacia. Para que esto se logre es necesario un ambiente de confianza y apoyarse los unos a otros en todo momento.
- El objetivo del proyecto debe ser claro, bien definido y realista.

Los principales elementos para la formación del equipo de trabajo se muestran en el siguiente esquema.




*Elementos necesarios para la formación del equipo.* Tomado de Humphrey, 1999.

### Asignación de recursos

El primer paso implica llegar a un acuerdo de administración de los recursos necesarios. La mayoría de los proyectos de desarrollo de *software* inicia siempre con una estimación preliminar de costos esperados, de acuerdo con el tamaño y complejidad del proyecto.

### Habilidades necesarias

Recaen en dos categorías generales, que son la técnica y el trabajo en equipo.

- **Dominio de las aplicaciones**: se necesitará, por lo menos, un miembro que tengan total conocimiento y dominio sobre el *software* que se va a desarrollar; por ejemplo, para el desarrollo de un *software* del área contable, se requiere a alguien que sepa perfectamente contabilidad para que oriente sobre ciertas características específicas

del área de contabilidad y que requieran considerarse para integrar en el *software* y así poder ayudar a los demás miembros del equipo a lograr los objetivos.

- **Tecnología del producto**: se refiere a lo complejo que puede ser el *software*, según lo grande o novedoso que se pretenda. Si el proyecto es complejo, se necesitará que el equipo cuente con la suficiente experiencia en el desarrollo de sistemas similares, además de constante ayuda mutua.
- **Herramientas y métodos**: las herramientas para el análisis y diseño del software son los elementos necesarios para el desarrollo del proyecto, tales como el lenguaje de programación y el motor de base de datos; por su parte los métodos indican las formas en que se utilizarán las herramientas. Se requiere que el equipo cuente con profesionales de *software* para saber qué herramientas y métodos se utilizarán.

**Trabajo en equipo**

Según la figura anterior, hay tres puntos importantes sobre el trabajo en equipo:

- **Estimación y planificación:** cuando se habla de estimación, se hace referencia a las personas del equipo que son capaces de hacer una planificación con tiempos adecuados y reales. Ellas pueden lograr que el equipo sea autodirigido; es decir, si se cuenta con una persona con las cualidades de estimar y planear, ésta se encargará de dirigir al grupo sin necesidad de capacitaciones. Así se ahorrará tiempo en el desarrollo del proyecto y será más fácil lograr los objetivos trazados.
- **Gestión de la calidad**: es un aspecto esencial en todos los proyectos que utilicen la metodología TSP. Los miembros del equipo deben ser competentes y creer que es importante gestionar personalmente la calidad de su trabajo, incluso cuando están trabajando bajo presión.
- **Comportamiento interpersonal**: para ser eficaces en un equipo todos deben ser capaces de trabajar en conjunto, participar en la resolución de problemas y ayudar a los demás cuando sea necesario. Es importante enfatizar que, al momento de seleccionar al personal que formará parte del equipo, quizás se encuentre a personas muy aptas en cuanto a conocimientos y habilidades, pero si tiene una actitud negativa sobre los aspectos de colaboración y desempeño grupal, ello puede generar problemas al interior del grupo y obstaculizar el óptimo desarrollo de las actividades.

**Reclutamiento**

Para contratar a las personas se requiere identificar una mezcla particular de destrezas y habilidades. Por eso es muy importante que, antes de iniciar el proceso de reclutamiento, se definan las necesidades para formar un equipo adecuado.

**Proporcionar capacitación**

Es necesario que se capacite a los miembros del equipo respecto a la metodología TSP, las herramientas de comunicación que se utilizarán y los diversos procesos en los que intervendrán.

Una vez que se cuenta con un equipo es necesario documentar los propósitos, objetivos y roles que guiarán las diversas acciones, a continuación, se explicará la forma de documentación.

## 2.1.1. Documentar propósitos, objetivos y roles del equipo

Sin duda, la documentación es parte importante del proceso en la metodología de TSP, y se refiere a la integración en documentos de los propósitos, objetivos y roles del equipo. La creación de éstos se realiza al inicio de la fase de lanzamiento, para después redactarlos e integrarlos en una plantilla que se muestra, en una junta, a todos los miembros del equipo. Los objetivos, propósitos y roles del equipo se deben plasmar en un documento para que todos los miembros puedan verlos en cualquier instante; de esta manera, si se integra un miembro nuevo en una etapa ya avanzada, también tendrá acceso a la información. Esto es muy importante porque en la plantilla se verá reflejado lo que se deseaba alcanzar cuando el proyecto llegue a su fin. Cabe señalar que el documento se comparte por medio de una red Intranet.

Normalmente los objetivos, propósitos y roles se redactan en un formato o plantilla que debe contener los siguientes elementos:

**Logo de la empresa que desarrolla el *software***. Como encabezado que debe aparecer en todas las páginas del documento.

**Nombre del proyecto**. Se debe de poner debajo del logotipo de la empresa, también como encabezado del documento. Es importante saber a qué proyecto pertenece el documento. Recuerda que en una empresa de desarrollo de *software* normalmente se trabaja en varios proyectos a la vez, por eso TSP propone los equipos de trabajo para tener una estructura bien definida y mayor control de cada proyecto que se esté realizando.

**Control de cambios**. Es una pequeña tabla en la que se controlan los cambios que va sufriendo el documento original. En el encabezado debe llevar la fecha de modificación y el nombre de la persona que creó o modificó; por ejemplo:

| FECHA DE MODIFICACION | NOMBRE DEL RESPONSABLE |
|---|---|
| | |

**Tema del contenido**. Se debe poner como título lo que va a contener la plantilla (objetivos, propósitos, roles, etcétera).

Estos documentos se suben a **Intranet**, que es parecido a una página web, pero con la diferencia de que sólo permite ver el contenido a las personas que estén dentro de la red de la empresa. Esto con la finalidad de que todos los miembros del equipo, de acuerdo con el proyecto en el que estén trabajando, puedan acceder a los documentos en cualquier momento.

TSP como metodología proporciona algunas plantillas como la LAU1 (primeras letras de la palabra inglesa *launch*, "lanzamiento"), que se mostrará en el siguiente subtema.

**Propósitos**

Discurso claro que se debe redactar en máximo dos párrafos que reflejen las intenciones o lo que se pretende alcanzar ((RAE, 2013). Redactar los propósitos del proyecto es muy importante porque más tarde se reflejarán en los objetivos. Dan una visión de qué se quiere lograr al final del proyecto. El líder de proyecto, el auxiliar de planeación y el administrador de desarrollo son los encargados de elaborarlos, y la alta gerencia los aprueba. Más adelante se mencionará quiénes son estas personas.

**Objetivos**

Son las metas por alcanzar cuando el proyecto llegue a su fin. Siempre deben comenzar con un verbo en infinitivo, tales como *desarrollar, analizar, concluir, examinar, interpretar, describir.*

Para realizar un buen objetivo deben formularse preguntas tales como: **¿qué tipo de *software* se desarrollará**? Puede ser contable, para administración de alumnos en una escuela, para controlar el área de ventas, etcétera. **¿Para quién será el *software* a desarrollar?,** por último ¿**con qué se va realizar**?, es decir, las herramientas necesarias para llevarlo cabo.

El ejemplo de un objetivo de acuerdo con un proyecto a realizar que ha de ocupar la metodología TSP sería el siguiente:

Desarrollar un sistema de administración contable, completo y fácil de utilizar para la empresa Contab S. A de C. V, que permita a los usuarios del sistema realizar la contabilidad para sus clientes y obtener reportes accesibles. Con el uso de herramientas como .net, Hibernate, JQuery y SQL Server, crear un software de calidad que cumpla con todos los requerimientos, estándares y necesidades de la empresa.

**Roles**

En el subtema 1.1.3. Fase de lanzamiento, se describió y explicó los cinco roles básicos en TSP que conforman la parte medular de un equipo:

- Líder de proyecto
- Administrador de desarrollo
- Auxiliar de planeación
- Administrador de calidad
- Administrador de configuraciones

Estos roles son necesarios si se quiere implementar esta metodología TSP. Debe tomarse en cuenta que se requiere de un grupo de desarrolladores de *software* formados en PSP e ingenieros de calidad, el cual estará a cargo de los administradores de calidad y de desarrollo, quienes determinarán la cantidad con base en el tamaño del *software* a desarrollar. La experiencia y capacidad individual de cada desarrollador e ingeniero de calidad será un factor muy importante al momento de la selección.

Recuerda que lo visto hasta aquí se lleva a cabo en la fase de lanzamiento del proyecto, y todo se redacta en la plantilla que indica la metodología TSP llamada LAU1, la cual se detallará en el siguiente subtema. Es importante mencionar que ya finalizada la fase de lanzamiento, se puede utilizar la plantilla vista en el apartado 1.3.3. Fase de planeación.

## 2.1.2 Planear y ejecutar el lanzamiento del proyecto

En este subtema aprenderás cómo se planea y ejecuta un proyecto con base en la metodología TSP; también revisarás la forma en que se integran los datos en la plantilla LAU1 (*launch script* 1) para realizar el lanzamiento del *software* a desarrollarse.

Antes de empezar la fase de lanzamiento, TSP propone realizar un *checklist*, en el que se redactarán las actividades y el orden en que se llevarán a cabo todas las tareas de esta fase. Este *checklist* contiene lo siguiente:

Tabla. Ejemplo de *checklist*

| | Actividad | Estatus |
|---|---|---|
| **1** | Establecer productos y objetivos de la empresa. | |
| **2** | Establecer roles y objetivos del equipo. | |
| **3** | Definir estrategia de desarrollo. | |
| **4** | Hacer un plan general. | |
| **5** | Hacer un plan de calidad. | |
| **6** | Balancear el plan (carga de trabajo). | |
| **7** | Plan de riesgos. | |
| **8** | Diseñar reporte de administración. | |
| **9** | Revisar el plan con la administración. | |
| **10** | Análisis *post mórtem.* El equipo revisa el proceso. | |

La columna *Estatus* se pude palomear, pero lo mejor es poner todo como **pendiente**; y conforme se completen las actividades, se cambiará por **completado**.

Una vez que el líder del proyecto, el auxiliar de planeación y el administrador de desarrollo redacten los objetivos, propósito, roles y *checklist*, crearán la plantilla LAU1 tal como se muestra a continuación; posteriormente, se impartirá un curso y se mostrará esta plantilla a todos los miembros del equipo.

Como se mencionó anteriormente, una empresa que se dedica al desarrollo de *software* puede trabajar en varios proyectos a la vez; para cada proyecto se conforman equipos llamados células de trabajo; dentro de cada una de estas células estarán cinco personas que ejercerán los roles propuestos por TSP, además de un grupo de desarrolladores e ingenieros de calidad, que se encargarán del desarrollo del proyecto al que hayan sido asignados dentro de su célula. Los reportes que cada equipo debe entregar son definidos en las reuniones que TSP señala para ejecutar el lanzamiento del proyecto. Más adelante se explicará a detalle qué temas se tocan en cada reunión.

Tabla. Plantilla LAU1

| | |
|---|---|
| **Propósito** | Crear los equipos en el primer ciclo de desarrollo. |
| **Criterios** | Todos los miembros del equipo deben completar el curso de PSP. |

| de entrada | | |
|---|---|---|
| **General** | Con esta plantilla se inicia el proyecto en equipo.<br><br>• Los objetivos se describen en esta junta.<br>• Se forman los equipos y se realiza la asignación de roles.<br>• Se explican los objetivos del proyecto a desarrollar.<br>• Se establecen las juntas de los equipos y los tiempos de entrega de los reportes. | |
| **Paso** | **Actividad** | **Descripción** |
| 1 | Resumen del curso | El instructor describe:<br>• Los objetivos del curso<br>• Qué se espera que los miembros del equipo logren.<br>• Cómo será evaluado y calificado su trabajo.<br>• Los principios básicos del trabajo en equipo.<br>• El proceso de TSP. |
| 2 | Información del integrante del equipo | El instructor explica:<br>• Los criterios para hacer la asignación de equipos.<br>• La información necesaria para la asignación adecuada.<br>• Los roles del equipo, responsabilidades y aptitudes. |
| 3 | Objetivos del producto | El instructor describe:<br>• Los objetivos del producto.<br>• Los objetivos críticos que deben ser satisfechos.<br>• Los objetivos deseables y opcionales.<br>• Los criterios para evaluar un producto terminado. |
| 4 | Asignación de equipos | El instructor asigna:<br>• Equipo y rol a los asesores. |
| 5 | Objetivos del equipo | El instructor describe:<br>• Los parámetros de los objetivos.<br>• Por qué se necesitan los objetivos y metas del equipo. |
| 6 | Reuniones del equipo | El instructor explica;<br>• El propósito de las juntas de equipo, fechas y reportes.<br>• Datos requeridos semanalmente. |
| 7 | La primera reunión del equipo | El líder y su equipo:<br>• Discuten los roles de los miembros del equipo.<br>• Discuten y se acuerdan los objetivos del ciclo 1.<br>• Establecen los tiempos de las reuniones semanales.<br>• Especifican y acuerdan los tiempos de todos los miembros del equipo para dar sus reportes semanales al administrador de planeación. |
| 8 | Datos requeridos | El administrador de planeación revisa:<br>• Los datos requeridos por cada miembro del equipo semanalmente.<br>• Los reportes que, sobre estos datos, serán generados y entregados al equipo |
| 9 | Inicia el proyecto | Los equipos inician su trabajo. |

A continuación, se explicarán los elementos que se deben integrar en cada columna de la plantilla LAU1; los cuales conforman el listado de acciones o *checklist* de la fase de lanzamiento de TSP.

**Propósito**

En el propósito general se describen los objetivos del curso:

- Asignar los roles (recuerda que cada rol que propone TSP debe ser adquirido por una persona cualificada).
- A cada uno de ellos se les asigna un equipo de trabajo.

**Criterios de entrada**

En esta fila (también llamada *entry criteria)* se integrará, como primer requisito, que los miembros del equipo hayan completado el curso de PSP (metodología de calidad a nivel personal). Si se quiere aplicar TSP, es de suma importancia que todos los integrantes del equipo conozcan y dominen PSP.

**General**

El instructor del curso se encarga de:

- Exponer los objetivos del software a desarrollar.
- Establecer los tiempos de las juntas.
- Establecer los reportes a entregar.

**Paso 1**

Se integra una introducción y un resumen del curso a los miembros el equipo. La persona encargada de la junta describe los siguientes puntos:

- Qué se espera que logren los miembros del equipo a lo largo del curso.
- Como será evaluado su desempeño dentro del curso.
- Explicar brevemente el tema del trabajo en equipo, pero la persona encargada de impartir el curso será quien lo desarrolle.
- Exponer ante el equipo qué es la metodología TSP y sus diferentes procesos.

**Paso 2**

Se expone la forma en que se realizará la asignación de los equipos y además:

- Se redacta a detalle la información sobre los diferentes roles del equipo, así como las responsabilidades y aptitudes que cada miembro debe cubrir.

**Paso 3**

Además de señalarse los objetivos del producto a desarrollar, deben exponerse los siguientes puntos:

- El objetivo principal. Generalmente se habla sobre la empresa que solicitó el recurso, el funcionamiento que tendrá el *software* y las herramientas para desarrollarlo.
- Los objetivos secundarios. Pueden ser más pequeños y a corto plazo, pero en conjunto permitirán alcanzar el objetivo principal.
- Los criterios para que se pueda evaluar el *software* como un producto terminado. En esta parte se describen los documentos que se entregarán al final del desarrollo del proyecto, tales como manual de usuario, manual técnico, ejecutable o instalador del *software* desarrollado, licencias requeridas, etcétera.

**Paso 4**

Se indica la forma en que se integrarán los nombres de cada uno de los miembros del equipo. El instructor del curso dará a conocer, a cada uno de los miembros, el equipo al que se integrarán y la asignación de roles para cada puesto.

**Paso 5**

**E**l instructor del curso describe los objetivos y roles para los equipos de trabajo.

**Paso 6**

El instructor da a conocer las fechas de las futuras juntas y el tiempo que durará el desarrollo del sistema.

**Paso 7**

Se indica que cada líder tendrá la primera junta con su equipo (aquí solo se mencionan las fechas; más adelante se explican las reuniones y qué se tratará en cada una), y algunos puntos a discutir, tales como:

- Los roles de los miembros del equipo.
- Objetivos del equipo.
- El estándar de las juntas semanales.

**Paso 8**

Se explica que el administrador de planeación junto con el equipo deberá revisar una vez por semana, el trabajo que se le haya solicitado a cada miembro del equipo, así como generar reportes (plantillas de PSP para el análisis individual de trabajo) donde se plasmen los avances.

**Paso 9**

Se inicia el proyecto con la fase de implementación.

El **lanzamiento** es una serie estructurada de las actividades del equipo guiado por un entrenador TSP, que también es el encargado de dar el curso donde se muestra la plantilla LAU1. El **lanzamiento** tiene una duración de **dos a cinco días**, incluye **nueve** reuniones y una ***post mórtem***.

Las reuniones de ejecución del lanzamiento son las siguientes:

**Reunión 1**. Los administradores presentan los objetivos del proyecto, fechas de entregas y requisitos de calidad; dan a conocer gastos críticos si no se entrega el proyecto. El equipo tiene la oportunidad de hacer preguntas acerca de los requisitos y limitaciones, con el fin de aclarar dudas sobre la presentación de los administradores.

**Reunión 2**. El equipo define sus objetivos.

**Reunión 3**. Se crea el diseño conceptual del *software* a desarrollar. Se definen el proceso y los planes de apoyo del equipo. Se crea la estrategia del proyecto y se reparte el trabajo.

**Reunión 4**. Se calcula el tamaño de las partes del diseño conceptual, y se crea el plan general donde vienen los recursos del proyecto, así como el calendario de desarrollo.

**Reunión 5**. Se crea el plan de calidad.

**Reunión 6**. Se divide el trabajo en proyectos individuales para cada fase del desarrollo y así balancear la carga de trabajo.

**Reunión 7**. Se identifican riesgos que puedan surgir durante el desarrollo del proyecto.

**Reunión 8**. Se prepara una estrategia para presentar el plan que se creó junto con la alta gerencia (comúnmente, estos puestos son ocupados por los dueños de la empresa que va a desarrollar el *software*).

**Reunión 9**. Se presentan los planes, y planes alternativos (si los hay), para la administración del proyecto y se recibe la aprobación de la alta gerencia. Una vez concluida la reunión se inicia el proyecto.

**Lanzamiento *post mórtem***. Se generan las mejoras a los procesos de la fase de desarrollo, y se hace una evaluación de toda la fase de lanzamiento.

## 2.2. Ejecutar el trabajo en equipo

En este tema aprenderás cómo se ejecuta el trabajo en equipo.

Una vez que todos conocen cuál es el rol que les toca desempeñar dentro del proyecto, los líderes se encargan de mantener a los miembros del equipo motivados para que realicen sus actividades de la mejor manera. Como se observó en la unidad 1, TSP brinda una metodología bien definida para trabajar con grupos en todos los ciclos de vida del desarrollo del proyecto, pero aquí surgen algunas preguntas para los administradores: **¿cómo controlar el proyecto de acuerdo con su tamaño?**, **¿cómo saber que el producto está cumpliendo con la calidad deseada?**, **¿cómo controlar los riesgos que puedan surgir durante el desarrollo?**

Afortunadamente, TSP brinda una serie de planes para resolver estas preguntas, conducir por buen rumbo el desarrollo del proyecto y llegar a los objetivos planteados en la fase de lanzamiento.

En los siguientes subtemas, aprenderás la forma y estrategias de creación de estos planes para así contar con una métrica exacta de posibles errores durante el desarrollo del proyecto. Todo con el objetivo de controlar y dar solución de una manera más adecuada a los problemas, lo que permitirá entregar el producto final en el tiempo estimado y con la calidad requerida.

### 2.2.1. Elaborar el plan del proyecto

En este subtema conocerás qué es el plan de proyecto, la importancia de aplicarlo durante el desarrollo y las actividades necesarias para hacer un adecuado plan.

Un **plan de proyecto** es una guía de las actividades que se realizan en un proyecto, así como de la manera en que deben estar organizadas en cuanto a tiempo, secuencia y orden. Dentro del plan se separan tareas de actividades y se ordenan entre los hitos del proyecto. Un **hito** se refiere a la duración de una tarea o actividad que termina hasta la culminación de su objetivo; tiene descripción y fecha bien definidas (Rodríguez, 2007).

El plan de proyecto es un medio de comunicación por el se dará a conocer las decisiones que tomen los participantes. Debe hacerse una vez que el cliente y el equipo han convenido, de manera formal, el proyecto a desarrollar. El principal responsable de generarlo es el líder de proyecto; debe asegurar que se elabore correctamente, que contenga los elementos necesarios y se ejecute.

En un plan de proyecto se contemplan los siguientes cuestionamientos: ¿quién?, ¿qué?, ¿por qué?, ¿cuándo?, ¿dónde?, ¿cómo? y ¿cuándo? A continuación, se mencionan los elementos que lo componen.

- Definición del alcance
- Estructura de desglose de trabajo
- Cronograma de actividades
- Recursos requeridos
- Presupuesto definitivo del proyecto
- Asignación de roles y responsabilidades
- Riesgos

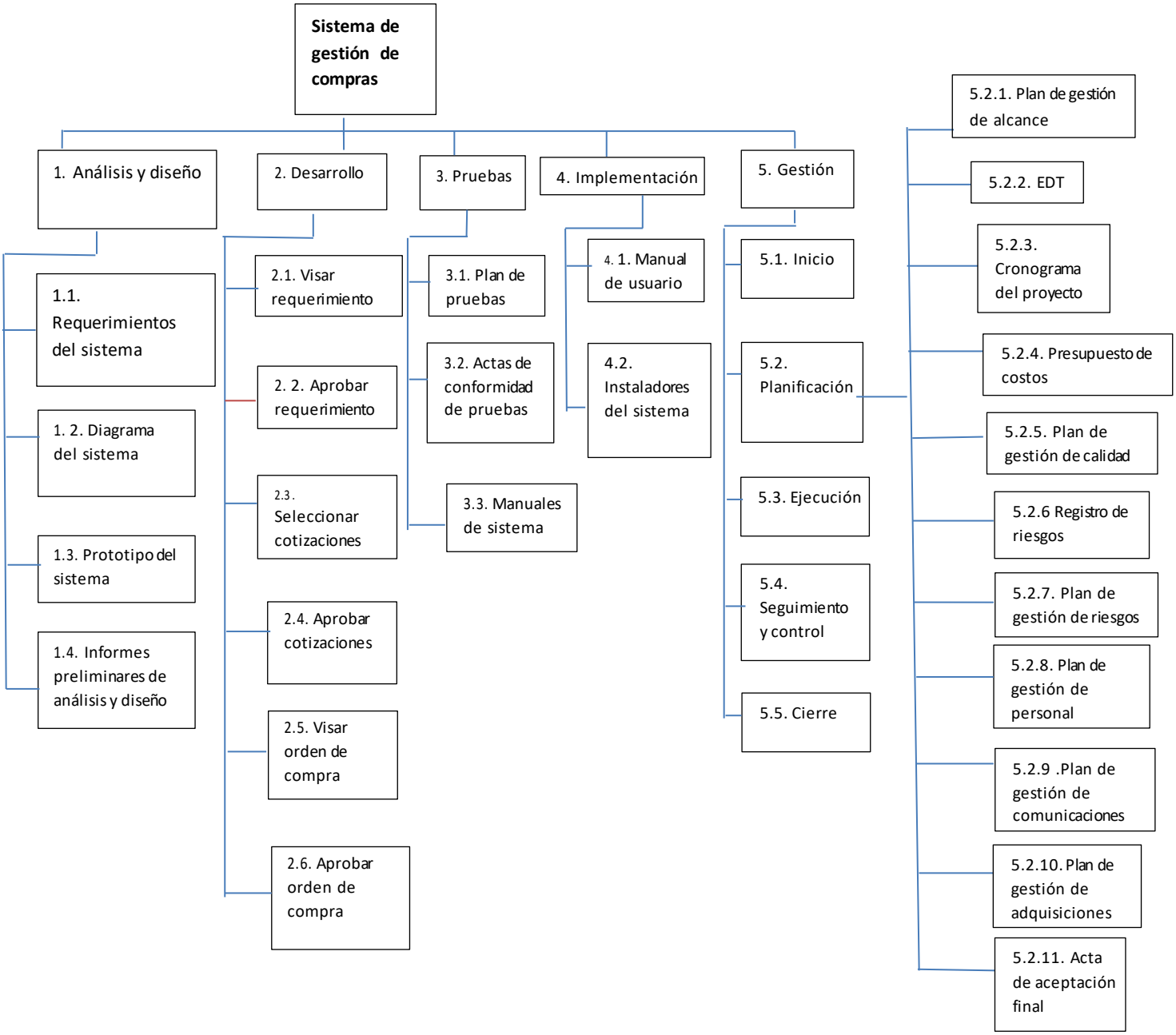
**Definición del alcance**

Para generar el enunciado del alcance se recomienda incluir los siguientes elementos:

- **Objetivos**: los planteados para el proyecto.
- **Descripción**: de los productos o servicios que se realizarán.
- **Límites del proyecto**: lo que se incluye y excluye.
- **Supuestos**: las situaciones que se dan por hecho; por ejemplo, se cuenta con el equipo de cómputo necesario para la programación del sistema.
- **Restricciones**: normas, reglas y políticas que se deben seguir.

**Estructura de desglose de trabajo (EDT)**

Es una herramienta que se utiliza para especificar el alcance de un proyecto. Es una representación gráfica de los elementos principales de la planeación. Es semejante al organigrama de una empresa, porque identifica los elementos de un proyecto. Dependerá de los requerimientos del proyecto que el equipo revisará. La siguiente ilustración ejemplifica la **EDT** para el desarrollo e implementación del sistema de gestión de compras de una empresa.




*Ejemplo Esquema de estructura de desglose de trabajo*. Tomada de http://cybertesis.upc.edu.pe/upc/2009/molina_np/xml/ressources/fig003a.jpg

**Cronograma de actividades**

Es una representación gráfica detallada de la secuencia en la que serán ejecutadas las actividades; se debe de establecer por medio de fechas. Existen diversas herramientas para generar este tipo de diagramas, tales como Microsoft Project.

**Recursos requeridos**

Los recursos humanos y materiales se establecen a partir de los roles y actividades de los miembros del equipo.

**Presupuesto definitivo del proyecto**

Las estimaciones se hacen con base en los costos y recursos asignados a las actividades que se hayan realizado en el proyecto.

**Asignación de roles y responsabilidades**

Es de suma importancia delegar a los miembros del equipo las tareas y actividades que desempeñarán según sus habilidades, actitudes y aptitudes.

**Riesgos**

La gestión de riesgos es una de las actividades más importantes dentro de la planeación del proyecto, y dependerá del cumplimiento de los objetivos. En el subtema 2.2.3. se abordará con mayor detalle la elaboración del plan de riesgos.

Dentro de un plan de proyecto, los objetivos y actividades a realizar deben ser claros y bien definidos. Cada una de las versiones del plan debe ser colocada en la administración de configuración y ha de contener un calendario para programar las actualizaciones futuras.

Es de gran importancia generar un plan de proyecto, ya que de éste depende que los objetivos propuestos en cada una de las actividades se cumplan en tiempo y forma. Así también, al generarlo se establece lo que se quiere y se hará.

## 2.2.2. Elaborar plan de calidad

En este subtema aprenderás a elaborar un plan de calidad de acuerdo con la metodología TSP. Es muy importante porque con él podrás tener una métrica exacta de qué se está haciendo bien, y los posibles fallos que surjan en el desarrollo del proyecto.

En la fase de lanzamiento se define el plan de calidad, que se basa en el tamaño del proyecto, y de acuerdo con esto se inyectarán defectos (Humphrey, 2006), que no es más que introducir defectos en cada una de las fases de desarrollo. El plan de calidad se basa totalmente en PSP, en el cual es necesario hacer plantillas en la que los desarrolladores plasmen los errores al codificar y la pérdida de tiempo que provocan; al final del día se hace un análisis de los fallos más recurrentes para tener oportunidad de mejora. Por ejemplo, para los desarrolladores de *software* los errores más comunes al codificar son los de punto y coma; el administrador de desarrollo retoma esta concurrencia de fallos y, al momento de aplicar el plan de calidad, se inyectan en el código para saber de manera exacta qué tan bien se está codificando.

Una vez que los ingenieros y administradores han identificado qué errores inyectar, TSP propone usar un histórico con estimaciones (*yield*); éstos darán el porcentaje de productividad de acuerdo con el trabajo realizado por cada miembro del equipo. Finalmente, el equipo examina este plan de calidad para ver si los parámetros son razonables, y si cumplen con los objetivos de calidad.

El contenido del plan de calidad debe integrarse por los siguientes elementos:

**Resumen de porcentaje**

Exposición breve y precisa del porcentaje de errores por línea de código. Tienen tres medidas en la calidad del proceso:

- **Porcentaje de reutilización:** mide el porcentaje del producto que fue reutilizado en algún otro proyecto.
- **Porcentaje de reutilización nuevo**: mide la contribución del ciclo en futuros proyectos.
- **Porcentaje libre de defectos (PDF)**: mide el porcentaje de componentes que no presentan defectos en una fase dada. Un ejemplo de PDF sería éste: en un componente de diez partes, nueve tienen errores. El PDF en fase de compilación es de 10% (no se toman en cuenta los errores, sólo la parte que está bien).

**LOC/Horas**

La línea de código (*line of code* por sus siglas en inglés) es la división de las líneas de código entre las horas por día o, según se establezca, con base en el registro de tiempos que se indica en PSP. El resumen del porcentaje indica la calidad global del equipo, un número grande significa costos bajos y productividad alta.

**Defectos por página**

Muestra los defectos removidos en cada página del documento de requerimientos.

**Defectos por KLOC**

KLOC significa "mil líneas de código". La experiencia muestra que cuando un producto tiene menos de 0.5 defectos/KLOC en construcción tanto como en pruebas de integración, y menos de 0.2 defectos/KLOC en pruebas de sistema, generalmente no habrá fallos que el usuario pueda encontrar (producto de alta calidad). Esta métrica se ejemplifica al final del capítulo.

**Proporción de defectos (RATIO)**

Proporciona mayor calidad a detalle en las revisiones de diseño y código. Se mide en número de defectos encontrados en revisión de diseño contra número de defectos encontrados en pruebas unitarias. La proporción para considerar un producto de calidad deberá ser mayor que 2.0.

**Proporción de tiempos de desarrollo**

Del tiempo de requerimientos 25% debe designarse a la inspección de requerimientos. Del tiempo de diseño de alto nivel 50% debe designarse a las inspecciones y revisiones. Del tiempo de diseño detallado 100% debe asignarse a realizar las inspecciones y revisiones.

**A/FR (*appraisal to failure ratio*)**

Se refiere a la revisión y pruebas del sistema. La medida para que se considere un producto de calidad sería de 2.0 para programas pequeños y 1.0 para grandes.

**Porcentaje de revisión e inspección**

En requerimientos debe ser <2.0 páginas de texto a espacio simple por hora. En diseño de alto nivel, <5.0 páginas de diseño por hora. En diseño detallado la revisión debe ser <100 líneas de seudocódigo por hora. En código debe ser <200 líneas de código por hora.

### Porcentaje de inyección de defectos (*yield*)

De acuerdo con experiencias basadas en PSP, la inyección de defectos debe ser de dos por hora en diseño detallado, y de seis por hora en codificación.

### Porcentaje de eliminación de defectos

En revisión de código de cero a 20 defectos por hora, serán seis defectos por hora para el 61.5% de los miembros del equipo de desarrollo. En revisión de diseño será de dos o más defectos por hora para el 57.9% de los miembros de equipo de diseño.

### Rendimiento *yield* de fase

Para entender este punto se expone un ejemplo: se tienen 19 defectos en un programa a la entrada de la revisión de código; se inyectó un defecto, pero se encontraron 15 defectos durante la revisión de código.

- Inyección del defecto.
- Se detectaron 15 defectos en la revisión.

Para obtener el rendimiento *yield* se aplica la siguiente fórmula:

Yield=100* (defectos encontrados) / (defectos en el producto)

Sustituyendo:

Yield=100*15/(19+1) = 75%

El *yield* sólo puede calcularse cuando se ha terminado el programa y se sabe la cantidad total de defectos.

### Rendimiento *yield* de proceso

Porcentaje de defectos removidos antes de cada fase. Puede decirse que antes de entrar a la fase de pruebas de sistema se debe de tener el 99% de defectos removidos.

### Ejemplo de plan de calidad

En el siguiente ejemplo se observará la forma en que se puede realizar un plan de calidad, a partir de 11 preguntas (Alvarado, 2009):

1. *¿Qué tipo de defectos se introdujeron durante el diseño y la codificación?*

En la fase de **diseño** los defectos más introducidos fueron los de tipo 70 y 80, pues ambos ocurrieron con una frecuencia de 4 y porcentaje de 33.3%, como se muestra en la tabla de abajo. Asimismo, los defectos de tipo 20 fueron los que tuvieron la mayor ocurrencia en la fase de codificación con una ocurrencia de 30 y un porcentaje de 52.6% como se puede observar en la siguiente tabla:

Tabla. Defectos introducidos en las fases de diseño y codificación

| | **Número inyectado** | | **Porcentaje inyectado** | |
|---|---|---|---|---|
| Tipo | Diseño | código | Diseño | Código |
| 10 | 0 | 1 | 0.0% | 1.8% |
| 20 | 2 | 30 | 16.7% | 52.6% |
| 30 | 0 | 3 | 0.0% | 5.3% |
| 40 | 1 | 2 | 8.3% | 3.5% |
| 50 | 1 | 2 | 8.3% | 3.5% |
| 60 | 0 | 0 | 0.0% | 0.0% |
| 70 | 4 | 9 | 33.3% | 15.8% |
| 80 | 4 | 6 | 33.3% | 10.5% |
| 90 | 0 | 4 | 0.0% | 7.0% |
| 100 | 0 | 0 | 0.0% | 0.0% |
| Total | 12 | 57 | | |
| | | | | |

Alvarado, 2009.

El número de defectos introducidos en la fase de diseño fueron doce, como se indica en la tabla anterior. En un número aceptable de errores de diseño pues son los más costosos en tiempo, tanto para encontrarlos como para corregirlos. A diferencia de la fase de **codificación**, los defectos de tipo 20 no figuraron entre los más comunes; tal como se muestra en la siguiente tabla.

Tabla. Defectos menos introducidos en la fase de diseño

| Tipo | Número inyectado | Porcentaje inyectado |
|---|---|---|
| 10 | 0 | 0,0% |
| 20 | 2 | 16,7% |
| 30 | 0 | 0,0% |
| 40 | 1 | 8,3% |
| 50 | 1 | 8,3% |
| 60 | 0 | 0,0% |
| 90 | 0 | 0,0% |
| 100 | 0 | 0,0% |

En la fase de codificación, el número de errores fue mucho mayor en comparación con la de diseño. Los errores más comunes fueron de tipo 20, con 30 ocurrencias y un porcentaje de 52.6%.

Es necesario hacer una clasificación de errores más comunes tanto en la fase de diseño como en la de codificación. Vale la pena tener una clasificación de éstos, pues ocurrieron por distintos motivos.

Tabla. Clasificación de los errores tipo 70 introducidos en la fase de diseño

| Tipo | Clasificación del defecto | Números de defectos introducidos |
|---|---|---|
| 71 | Mal cálculo de la función | 2 |
| 72 | Error de parámetros | 2 |
| Total | | 4 |

Tabla. Clasificación de los errores tipo 80 introducidos en la fase de diseño

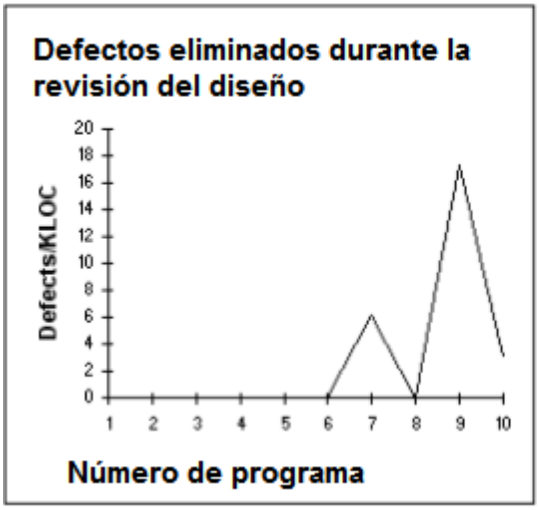
| Tipo | Clasificación del defecto | Número de defectos introducidos |
|---|---|---|
| 81 | Mal funcionamiento | 3 |
| 82 | Recodificación | 1 |
| Total | | 4 |

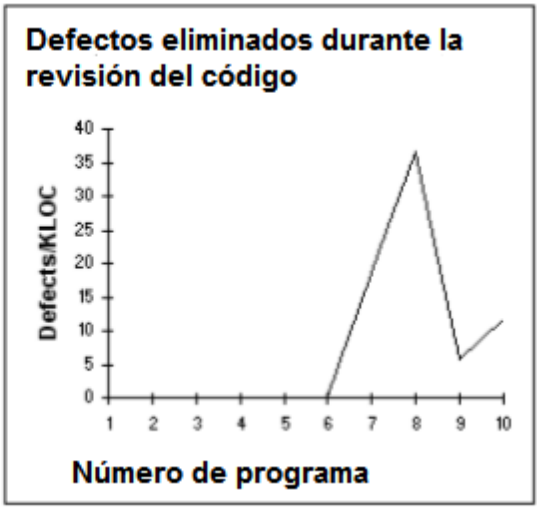
Tabla. Clasificación de los errores tipo 20 introducidos en la fase de codificación

| Tipo | Clasificación del defecto | Número de defectos introducidos |
|---|---|---|
| 21 | Error de dedo | 18 |
| 22 | Inicialización de variables | 5 |
| 23 | Tipo de dato incompatible | 6 |
| 24 | Error de parámetros | 1 |
| Total | | 30 |

2. *¿Qué tendencias son evidentes en defectos/KLOC encontradas en las revisiones, en la compilación y en las pruebas?*

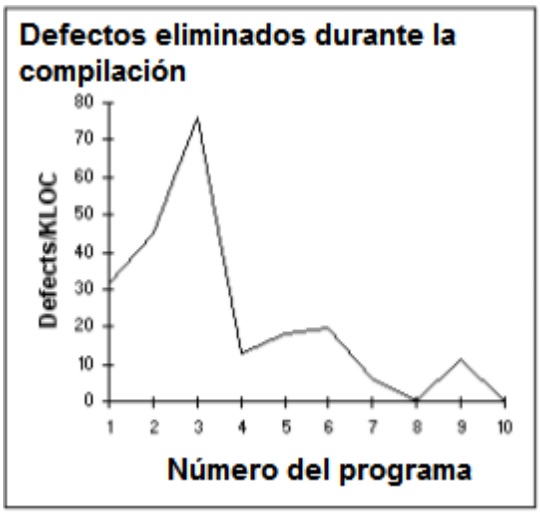
Las revisiones permitieron mitigar un gran número de defectos; así los programas llegaron a las fases posteriores con menos errores.

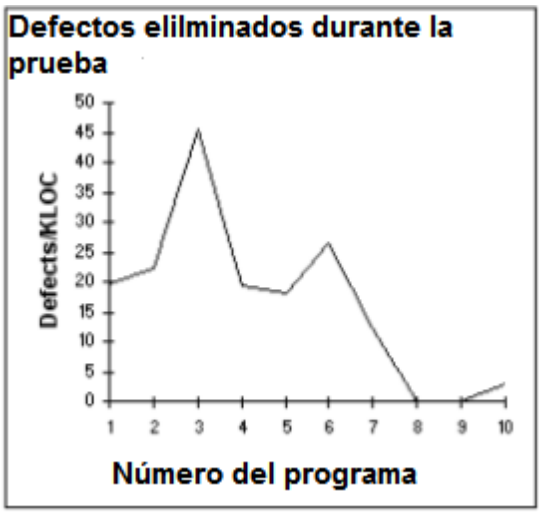



Gráficas de defectos eliminados durante las revisiones de diseño y codificación

En la gráfica anterior se muestra la tendencia a dejar pasar cada vez menos defectos a las fases posteriores; esto ayudó a que los programas se hicieran con mayor calidad. En las fases de compilación y pruebas se observaron cada vez menos errores, situación favorable originada por la intervención de las revisiones de diseño y codificación, donde más errores se encontraron.

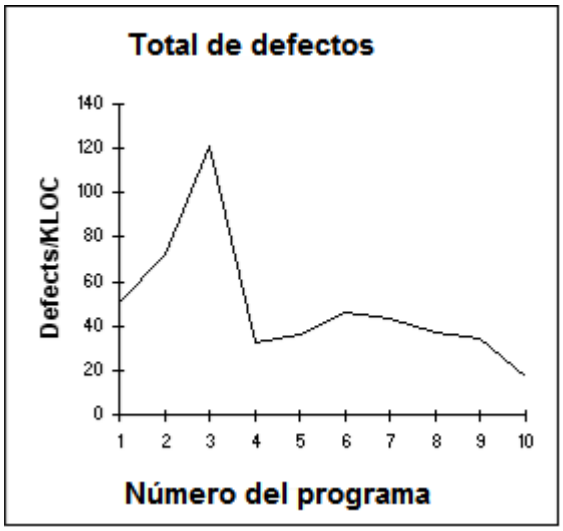



Gráficos de defectos eliminados en las fases de compilación y pruebas

3. *¿Qué tendencias son evidentes en el total de defectos/KLOC?*

El número de errores en la primera mitad del curso fue muy elevado, como se observa en la gráfica anterior; pero cuando se introdujeron las revisiones de diseño y código, el número de errores bajó considerablemente. La tendencia que se advierte es que, si se integraran revisiones personales adicionales (*checklist*) que permitan tener un menor número de defectos, se obtendría una mayor calidad en los programas.




Gráfica del total de defectos por programa y por KLOC

4. *¿Cómo se comparan las tasas de defectos removidos (en defectos eliminados/KLOC) con la revisión de diseño, la revisión de código, la compilación y las pruebas?*

En la primera mitad del curso todos los errores eran captados por las fases de compilación y pruebas; pero en la segunda mitad, ya introducidas las revisiones de diseño y código, se redujo el número de errores que llegaba a compilación y pruebas. Esto se nota claramente en la siguiente tabla. Si se hubieran realizado estas revisiones desde el inicio, las revisiones de diseño y código tendrían la mayor tasa de defectos eliminados.

Tabla. Tasa de defectos eliminados en las fases de revisión de diseño, revisión de código, codificación y pruebas

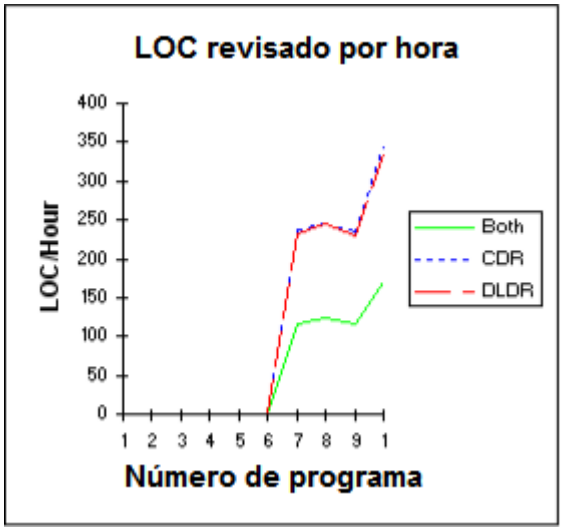
| Fase | Número de defectos eliminados | Tasa de defectos eliminados (%) |
|---|---|---|
| DLDR | 5 | 7.0% |
| CR | 11 | 15.5% |
| COMPILE | 31 | 43.7% |
| TEST | 24 | 33.8% |
| TOTAL | 71 | 100.0 |

5. *¿Cuáles son las influencias de defectos removidos por revisión de diseño, revisión de código y compilación contra pruebas unitarias?*

Las pruebas unitarias se realizan después de las revisiones de diseño, código, y de la fase de compilación; por lo tanto, el número de defectos encontrados y eliminados en esta fase es mucho menor que en las anteriores.

6. *¿Cuáles son las tasas de revisión (en LOC revisadas/hora) para revisiones de diseño y de código?*

En la siguiente figura se muestra la tendencia a aumentar las LOC revisadas por hora, tanto en la revisión de diseño como en la de código.




LOC revisadas por hora en revisiones de diseño de código

En la siguiente tabla se muestran las tasas de revisión para diseño y código. En las revisiones por separado, la tasa está en un promedio de 261 LOC revisadas por hora; para diseño, código 267 LOC por hora; pero tomándolas juntas se cuenta con una revisión promedio de 130 LOC revisadas por hora.

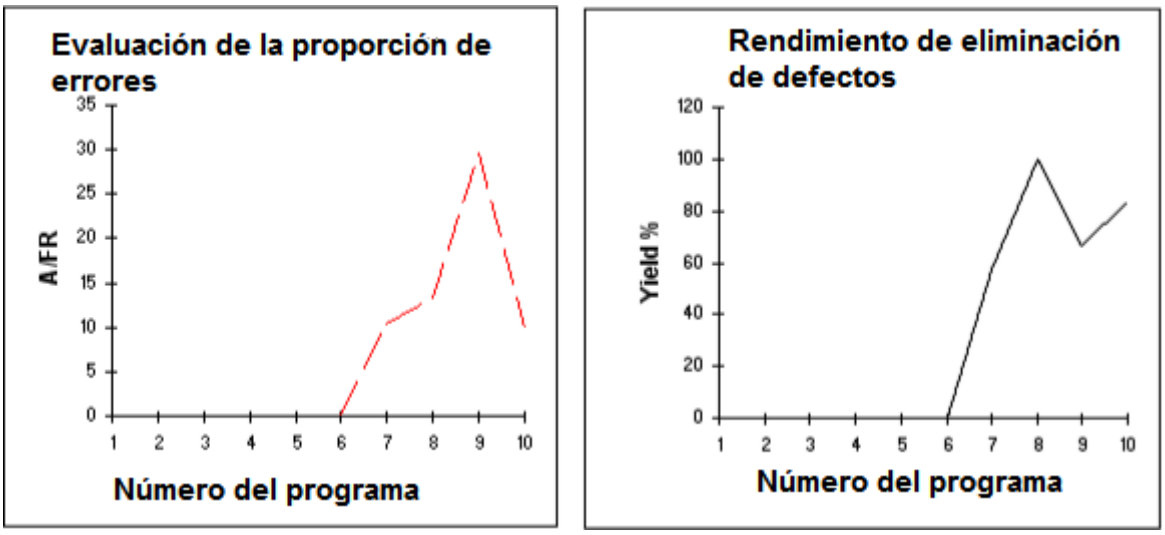
Tabla. Tasa de revisión (en LOC revisadas por/hora para revisiones de diseño y código)

| Programa | Tasa de revisión de diseño (LOC revisadas/hora) | Tasa de revisión de código (LOC revisadas/hora) | Tasa para ambas revisiones (LOC revisadas/hora) |
|---|---|---|---|
| 7 | 231.4 | 237.1 | 115.7 |

| | | | |
|---|---|---|---|
| 8 | 246 | 246 | 117.1 |
| 9 | 230 | 235.9 | 115.3 |
| 10 | 335.4 | 346.8 | 170.5 |

7. *¿Hay una relación entre el* yield *y el A/FR para los programas del siete al diez?*

La relación que hay entre el *yield* y el A/FR de las tareas en la segunda parte del curso estuvieron muy relacionadas. En la gráfica siguiente se puede observar que cuando el AF/R aumentaba, también lo hacia el y*ield,* y viceversa.




Gráficas de *yield* y AF/R de las tareas 7A a la 10A

En la gráfica anterior se observa que el *yield* siempre crece, a excepción de la tarea nueve. El promedio del *yield* para la segunda mitad del curso fue de 76.785%, esto indica que una gran parte de los defectos se eliminaron antes de llegar a la fase de desarrollo. El AF/R de la gráfica representa una tendencia creciente, eso significa que como los valores de A/FR siempre fueron mayores a uno, el tiempo dedicado a las revisiones tanto de diseño como de código fue mayor que los tiempos de compilación y pruebas.

8. *¿Cómo se puede hacer el proceso más efectivo y eficiente?*

Es necesario realizar pruebas de escritorio después de la fase de revisión de diseño para eliminar los errores de pruebas que se pudieran introducir. El tiempo en el diseño aumentaría, pero eso beneficiaría al tiempo de pruebas y calidad del programa.

9. *Basados en los datos históricos ¿Cuáles son los objetivos de calidad?*

- Obtener una tasa menor que 20 defectos/KLOC.
- Reducir el número de los errores más comunes.
- Alcanzar un *yield* promedio del 85%.

10. *¿Cómo se puede cambiar el proceso para llegar a esos objetivos?*

Para mejorar la tasa de defectos, que en estos momentos es de 29 defectos/KLOC, es necesario dedicar más tiempo a entender los requerimientos del programa, pues éstos han presentado los más difíciles de resolver. Para reducir el número de defectos comunes es necesario agregar validaciones al *checklist* y dedicar más tiempo a su aplicación, pues por realizarlo rápido no se eliminaron defectos detectables. El *yield* actual es de 76.785%, para alcanzar 85% es necesario usar con más eficiencia el estándar de diseño; parte de esto conlleva el incluir documentación entre el código con el fin de que pueda interpretarse fácilmente. Para lograr los objetivos es necesario integrar nuevos *checklist* que permitan reducir errores y tiempos.

### 2.2.3. Elaborar plan de riesgos

En este subtema se explicarán los riesgos que se presentan en un proyecto de *software* y de qué manera se deben tratar los problemas que pueden ocasionar; para lo cual se debe generar y examinar un plan de riesgos que ayude al equipo a definir y evaluar los riesgos del proyecto.

Un **riesgo** "es una medida de la probabilidad y gravedad de los efectos adversos" (Lawrence, 1976). Es algo que puede suceder. Cuando se está seguro de que un evento se producirá no es un riesgo, porque se sabe que se presentará en algún momento; por ello se dice que es un evento de certidumbre (Humphrey, 2006).

El surgimiento de los riesgos tiene relación con la incertidumbre que rodea a las decisiones y los resultados del proyecto. Cuando se presenta la incertidumbre en la toma de decisiones, se está ante el riesgo de no cumplir los requerimientos del cliente. La incertidumbre acarrea complicaciones a la planeación y ejecución del proyecto. Lamentablemente esto es inevitable; por eso una adecuada toma de decisiones requiere que siempre se considere la incertidumbre, y que se evalúen los posibles impactos que llegue a tener.

Un **plan de riesgos** es una serie de métodos para complementar la toma de decisiones, bajo un ambiente de incertidumbre; se da a través de la identificación, análisis, respuesta y seguimiento de las situaciones de incertidumbre, las cuales puede llegar a afectar o cambiar lo planeado. Este plan es de gran importancia ya que, si se prevén los riesgos y se toman las decisiones apropiadas con anticipación, existe una alta posibilidad de evitar los riesgos y problemas en el proyecto.

Otro aspecto importante es el costo de la recuperación que, en la mayoría de los casos en que presenta un riesgo, es más alto de lo que hubiera costado aplicar las medidas para evitarlo. Por esta razón, la prevención de riesgos y la toma de decisiones acertadas para evitarlos brindarán el ahorro de tiempo y esfuerzo.

Finalmente, otro aspecto importante que se toma en cuenta para realizar la planeación de los riesgos es que, en la mayoría de los casos, los problemas que llegaran a presentarse se pueden conocer con suficiente anticipación.

El plan de riesgo TSP contempla cinco directrices básicas (Humphrey, 2006):

1. Aprender de los errores del pasado.
2. Todos los equipos deben gestionar los riesgos.
3. Empoderar a los miembros del equipo para gestionar los riesgos.
4. A cada riesgo significativo se debe asignar un propietario.
5. Revisar periódicamente los riesgos.

**Aprender de los errores del pasado**

La gran mayoría de los riesgos son conocidos; son pocos los errores nuevos que se llegan a presentar. Al conocer los problemas más comunes de los proyectos pasados, los equipos pueden determinar los problemas más comunes que se pueden presentar en los proyectos futuros.

**Todos los equipos deben gestionar los riesgos**

Los participantes del proyecto conocen mejor los riesgos, por esta razón los equipos de trabajo pueden anticipar de una manera más fácil los problemas y solucionarlos cuando aún sea posible.

**Empoderar a los miembros del equipo para gestionar los riesgos**

Los participantes que están directamente relacionados con el plan de riesgos deben de ser tratados como si fueran parte de la gestión del sistema; esto ayudará a tener una mejor cooperación y participación. Empoderar a los miembros del equipo es darles cierta

responsabilidad para que tomen las decisiones adecuadas, y así dar seguimiento y solución a un riesgo. Pero esto sólo se hará con personas que estén involucradas en el desarrollo del proyecto.

**A cada riesgo significativo se debe asignar un propietario**

Cuando se identifica un riesgo significativo, tiene que asignarse a un responsable, con esto se logrará que la cobertura sea la más adecuada y que, al darle seguimiento, se tomen las medidas necesarias para mitigar, corregir el riesgo y evitar futuros problemas. A manera de ejemplo, si se encontró un riesgo relacionado con el entorno de desarrollo y programación, el responsable de darle seguimiento será el programador.

**Revisar periódicamente los riesgos**

Debe llevarse a cabo después de que el equipo identificó los riesgos de más importancia, para así darles el seguimiento adecuado y asignar al responsable que se encargara de hacer una revisión periódica junto con el equipo; esto dependerá de la importancia del riesgo y sus consecuencias. Para la administración de los riesgos se deben realizar las siguientes acciones:

1. Identificar los riesgos.
2. Analizar los riesgos.
3. Elaborar planes de contingencia.
4. Controlar el estado de los riesgos.
5. Analizar los resultados y aprender de ellos.

*Identificar los riesgos*

En la identificación de los riesgos se recuperan todas las inquietudes y preocupaciones que estén ligadas con la elaboración del proyecto. Para ello, los miembros del equipo trabajan en conjunto mediante reuniones en las que se genera una lluvia de ideas; cada participante expone los posibles riesgos que considere puedan presentarse durante el desarrollo del proyecto. A continuación, se muestra una tabla de identificación de riesgos de un sistema en web en la que se evalúa la seguridad.

Tabla. Identificación de riesgos

| Seguridad | | | | |
|---|---|---|---|---|
| **ID** | **Tipo** | | **Descripción del riesgo** | **Posibles consecuencias** |
| | **Interno** | **Externo** | | |
| R18MS | | X | Ataques contra contraseñas de usuarios | Pérdida de acceso a información |
| R19MS | | X | Desencriptación de información | Copia de datos |

| | | | | |
|---|---|---|---|---|
| R20MS | X | | Vulnerabilidad del antivirus | Inestabilidad en el servicio |
| R21MS | | X | Vulnerabilidad en los puertos | Pérdida de servicios |
| R22MS | | X | Ataques de SQL a la base de datos. | Pérdida de consistencia |
| R23MS | | X | Vulnerabilidad en el *firewalls* | Contaminación por virus informáticos |
| R24MS | | X | Vulnerabilidad en el servidor | Robo de información |
| R25MS | | X | Robo de conexión WI-FI | Pérdida de calidad en el servidor |
| R26MS | | X | Vulnerabilidad de certificados digitales | Pérdida de datos |
| R27MS | | X | Vulnerabilidad de caché ARP (Address Resolution Protocol) | Retraso de servicios |

(Romero, Barragán, Romero, 2013, pág. 22)

La tabla anterior se construye con los elementos que a continuación se describen:

- **ID**: clave con la que se identificará el riesgo en las siguientes tablas para su control y seguimiento.
- **Tipo**: debe ser evaluado y clasificado por los involucrados en el desarrollo del proyecto. Puede ser interno o externo, dependiendo el área de afectación que pueda provocar el riesgo. Ejemplo: si se presenta en los estándares, financiamiento, gestión del alcance, integración del equipo etcétera, se dice que es de tipo interno; pero si se presenta en los clientes, proveedores, mercado etcétera, es de tipo externo.
- **Descripción del riesgo**: se explican a detalle sus características.
- **Posibles consecuencias**: se describe la forma en que pude afectar al proyecto.

*Analizar los riesgos*

Cuando se tienen identificados los riesgos, se debe realizar un análisis cualitativo en el que se prioricen los más significativos con base en lo siguiente:

- **Probabilidad:** puede ser 1 a 100 porcentual, que es el grado de probabilidad de que ocurra.
- **Impacto**: efecto que tendrá el riesgo si se llegara a presentar.

Para clasificar los eventos y la magnitud se utiliza la escala del uno a cien porcentual; pero para realizar el cálculo se utiliza 0.10 cuando el riesgo es de bajo impacto, y 1.0 cuando es de mayor impacto. Se debe tomar en cuenta que si el valor es de 90%, será 0.90 para realizar la operación. La magnitud se obtiene al multiplicar la probabilidad por el impacto del riesgo en el proyecto. Ejemplo:

$$0.50\, x\, 0.90 \;=\; 0.45$$

Finalmente, para representar el resultado obtenido se utiliza una matriz cualitativa del riesgo que contiene la probabilidad y la magnitud del impacto, la cual observarás a continuación.

Tabla. Matriz cualitativa de riesgo

| | Impacto | | | | | Escala de prioridad de eventos de riesgo |
|---|---|---|---|---|---|---|
| Probabilidad | 0.10 | 0.35 | 0.50 | 0.80 | 1.00 | |
| 0.90 | 0.09 | 0.32 | 0.45 | 0.72 | 0.90 | Riesgo bajo |
| 0.70 | 0.07 | 0.25 | 0.35 | 0.56 | 0.70 | Riesgo medio |
| 0.50 | 0.05 | 0.18 | 0.25 | 0.40 | 0.50 | Riesgo alto |
| 0.30 | 0.03 | 0.11 | 0.15 | 0.24 | 0.30 | |
| 0.10 | 0.0.1 | 0.04 | 0.05 | 0.08 | 0.10 | |

En la siguiente tabla se muestra un ejemplo de análisis de riesgo.

Tabla. Análisis del riesgo y estrategias

| **ID** | **Análisis del riesgo** |
|---|---|
| 18MS | **Magnitud:**<br><br>Alta<br><br>**Descripción:**<br><br>Las cuentas de usuarios pertenecientes a un sistema Microsoft Windows están guardadas en pares de datos; es decir, hacen referencia a sus respectivas contraseñas.<br><br>**Impacto:**<br><br>En la seguridad. Otras personas podrían adquirir los privilegios de administrador y cambiar ciertas funciones.<br><br>**Estrategia para tratar el riesgo:**<br><br>Establecer métodos de autentificación como: NTLM y SSO Kerberos. |

*Elaborar planes de contingencia*

Conjunto de procedimientos alternativos a la operativa normal de cada empresa, cuya finalidad es permitir su funcionamiento, aun cuando alguna de sus funciones deje de operar por culpa de algún incidente tanto interno como ajeno a la organización (Castillo, 2013). Se elaboran con la finalidad de disminuir riesgos; contienen recomendaciones para reducir las amenazas e incrementar las oportunidades dentro de los objetivos del proyecto.

*Controlar el estado de los riesgos*

Dentro del control de los riesgos se implementan planes de respuesta contra los que se identificaron.

*Analizar los resultados y aprender de ellos*

Los resultados de las observaciones en proyecto pasados sirven para aprender de éstos y así estimar y prever la presencia de los riesgos. Para llevar un análisis y gestión se debe generar una matriz de riesgos, la cual se aplicará en las reuniones con los participantes en el desarrollo del proyecto, y así dar el seguimiento necesario.

La utilización de un plan de riesgos es muy importante porque ayuda al equipo a identificar los riesgos potenciales dentro del proyecto. En la mayoría de los casos pueden ser evitados, lo que reduce problemas en el proyecto.

## Cierre de la unidad

Dentro del TSP, para formar los equipos de trabajo es primordial tener cuenta que los integrantes deben tener habilidades, actitudes, aptitudes y la capacidad para realizar las actividades que les tocará desempeñar dentro del proyecto, además de compromiso con los objetivos del proyecto bajo una estricta disciplina de trabajo.

Para realizar el trabajo los miembros del equipo deben de tener claro el rol que desempeñarán en el proyecto, así como las tareas que les corresponden. El líder de proyecto tiene la responsabilidad de motivar a los miembros del equipo para que desempeñen sus tareas de una manera adecuada.

La importancia del plan de proyecto es fundamental para determinar lo que se va a hacer y hasta dónde se quiere llegar; por su parte, llevar un control de las actividades que se realizarán asegura su cumplimiento y la culminación el proyecto. De igual manera, la planeación de calidad evitará retrasos debidos a errores de codificación, y se garantiza la calidad del proyecto. Otra actividad de suma importancia es la generación del plan de

riesgos, que evita y mitiga los posibles contratiempos que se puedan presentar a lo largo del proyecto, y que puedan llevar al no cumplimiento de los objetivos planteados.

## Para saber más

En esta página encontrarás información diversa sobre las herramientas de medición de la calidad del desarrollo de *software*, entre ellas TSP, así como diversos artículos y documentos acerca de dicha metodología.

*Software* Engineering Institute Carnegie Mellon (2013). Pittsburgh: https://www.sei.cmu.edu/ Fecha de consulta: 22 de enero del 2020.

## Fuentes de consulta

Alvarado, A. (2009). *Desarrollo de sistemas con PSP, TSP y UML*. https://docplayer.es/25202328-Desarrollo-de-sistemas-con-psp-y-tsp.html

Castillo, C. (2013). *Proyecto plan de contingencia*. Scribd. http://es.scribd.com/doc/33729961/Proyecto-4-Plan-de-Contingencia

Colomo, R. et al. (2011). *Team software process*. Universidad Carlos III de Madrid. http://www.rcolomo.com/papers/74.pdf

Esparza Maldonado, A. L. (2019, 1 de agosto). *Adaptación del modelo Team Software Process (TSP) para equipos transdisciplinarios en la producción de aplicaciones de calidad para personas con discapacidad* [Tesis de maestría, Universidad Veracruzana]. Repositorio Institucional de la UV. https://cdigital.uv.mx/handle/1944/49537

Gómez de Silva Garza, A. (2008). *Introducción a la computación*. Cengage Learning.

Humphrey, W. S. (1999). *Introduction to the team software process*. Addison-Wesley Professional.

Humphrey, W. S. (2006). *TSP(SM) coaching development teams*. Addison-Wesley Professional.

Lawrence, W. (1976). *Of acceptable risk: Science and the determination of safety*. William Kaufmann.

Piattini, M. (2011). *Calidad de sistemas de información*. Alfaomega/Ra-Ma.

Queensland, T. U. (2010). *Lecture 14: The team software process*. CSSE3002 The Software Process. https://my.uq.edu.au/programs-courses/course.html?course_code=csse3002

Real Academia Española. (s. f.). *Diccionario de la lengua española*. Recuperado el 3 de febrero de 2026, de https://dle.rae.es/

Rodríguez, J. (2007). *Gestión de proyectos informáticos: Métodos, herramientas y casos*. Editorial UOC.

Romero Gabriel, C., Barragán Méndez, H. I. y Romero Rivera, O. (2013). *Desarrollo de aplicación para el SGB*. Universidad Tecnológica de Nezahualcóyotl.