

# Herramientas de productividad HDP115

UNIDAD II: HERRAMIENTAS PARA LA PROGRAMACIÓN DE  
PROYECTOS

BLADIMIR DIAZ CAMPOS

UNIVERSIDAD DE EL SALVADOR | Facultad de Ingeniería y Arquitectura, Escuela de Sistemas  
Informáticos



## Contenido

1. Programación de proyectos .....	2
1.1. Etapas de un proyecto .....	2
1.2 Conceptos generales.....	3
1.3 Etapas del análisis de la red de un proyecto .....	5
2. Modelos de redes para la programación de proyectos.....	5
2.1 Diagrama de redes de proyectos.....	6
2.2 Tipos de dependencias entre tareas.....	6
3. Proyectos de desarrollo colaborativos .....	11
3.1 Características .....	12
3.2 Organización de proyectos de desarrollo colaborativo .....	12
4. Herramientas para la gestión de proyectos.....	13
4.1 Sistemas para la gestión de proyectos.....	14
4.2 Sistemas de control de versiones .....	15
4.3 Sistemas de gestión de incidencias.....	17
4.4 Sistemas de gestión de requerimientos.....	18
4.5 La nube.....	18



## 1. Programación de proyectos

### 1.1. Etapas de un proyecto

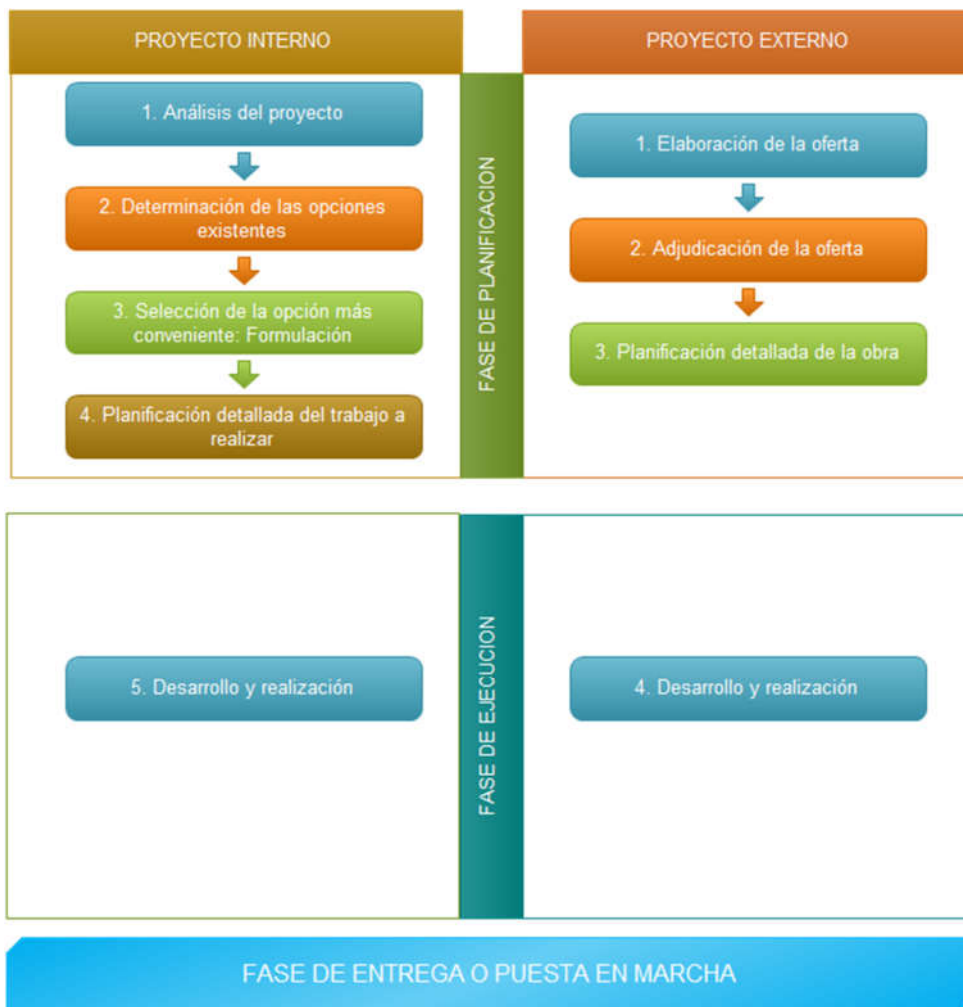
Desde un punto de vista general puede considerarse que todo proyecto tiene tres grandes etapas:

1. Fase de planificación: Se trata de establecer cómo el equipo de trabajo deberá satisfacer las restricciones de prestaciones, planificación temporal y costo. Una planificación detallada da consistencia al proyecto y evita sorpresas que nunca son bien recibidas.
2. Fase de ejecución: Representa el conjunto de tareas y actividades que suponen la realización del proyecto, es decir, la ejecución de la obra de que se trate. Responde ante todo a las características técnicas específicas de cada tipo de proyecto y supone poner en juego y gestionar los recursos en la forma adecuada para desarrollar la obra en cuestión. Cada tipo de proyecto responde en este punto a su tecnología propia, que es generalmente bien conocida por los técnicos en la materia.
3. Fase de entrega o puesta en marcha: Como ya se ha dicho, todo proyecto está destinado a finalizarse en un plazo predeterminado, culminando en la entrega de la obra al cliente o la puesta en marcha del sistema desarrollado, comprobando que funciona adecuadamente y responde a las especificaciones en su momento aprobadas. Esta fase es también muy importante no sólo por representar la culminación de la operación sino por las dificultades que suele presentar en la práctica, alargándose excesivamente y provocando retrasos y costos imprevistos.

A estas tres etapas se pueden añadir dos:

1. Fase de iniciación: Definición de los objetivos del proyecto y de los recursos necesarios para su ejecución. Las características del proyecto implican la necesidad de una fase o etapa previa destinada a la preparación del mismo, fase que tienen una gran trascendencia para la buena marcha del proyecto y que deberá ser especialmente cuidada. Una gran parte del éxito o el fracaso del mismo se fragua principalmente en estas fases preparatorias.
2. Fase de control: Monitorización del trabajo realizado analizando cómo el progreso difiere de lo planificado e iniciando las acciones correctivas que sean necesarias. Incluye también el liderazgo, proporcionando directrices a los recursos humanos, subordinados (incluso subcontratados) para que hagan su trabajo de forma efectiva y a tiempo.

Diferencias entre los tipos de proyecto.



## 1.2 Conceptos generales

### PROGRAMACIÓN DE PROYECTOS

Es el proceso de planear, organizar y administrar tareas y recursos para alcanzar un objetivo concreto, generalmente con limitaciones de tiempo, recursos o costo.

La mayoría de los proyectos comparten actividades comunes, como la división del proyecto en tareas de fácil manejo, la programación de las tareas, la comunicación entre los miembros del equipo y el seguimiento de las tareas a medida que progresa el trabajo.

Todos los proyectos constan de tres fases principales:



1. Crear el plan
2. Administrar y realizar un seguimiento del proyecto
3. Cerrar el proyecto

Programar un proyecto tiene como objetivo determinar de forma precisa los datos más importantes relacionados al proyecto:

1. El tiempo mínimo para la finalización del proyecto
2. Las actividades críticas.
3. El tiempo más temprano y más tardío para iniciar y terminar una actividad.
4. El tiempo de holgura de una actividad.
5. Las mejores alternativas.
6. Las alternativas en las cuales los recursos extras deben ser utilizados.
7. Si la marcha de un proyecto está acorde a la programación o al presupuesto.
8. Definir un nivel constante de utilización de recursos.
9. Completar el proyecto en un tiempo mínimo bajo recursos limitados.

## **TAREA**

La tarea es una actividad que tiene un comienzo y un fin. Todas las tareas deben tener asignado un tiempo (generalmente estimado) y muy frecuentemente costos, para su realización. Dado que un proyecto está compuesto de tareas, es necesario finalizar todas las tareas para dar por finalizado el proyecto.

## **ENFOQUES**

**Predictivo:** El enfoque predictivo es el que intenta anticipar las causas potenciales de problemas de programación, para que puedan ser corregidos por planes de contingencia

**Reactivo:** Este enfoque reacciona a problemas que se desarrollan sobre la programación en ejecución.

## **CARACTERÍSTICAS**

La programación de un proyecto debe considerar que los proyectos tienen una serie de características que determinan el éxito o el fracaso del mismo.

1. Complejidad
  - a. Interdependencia de actividades
  - b. Requerimientos de múltiples recursos
  - c. Múltiples eventos concurrentes
  - d. Objetivos en conflicto
  - e. Restricciones técnicas
  - f. Conflictos de secuenciación
2. Incertidumbre



- a. Inconsistencia de material insumo
  - b. Desglose de equipamiento
  - c. Rendimiento de operadores
  - d. Ausentismo de fuerza laboral
3. Dinámico
- a. Variabilidad de recursos
  - b. Cambios en ordenes de trabajo
  - c. Sustituciones de recursos

### 1.3 Etapas del análisis de la red de un proyecto

#### FASE DE PLANEAMIENTO DE LA RED

- Identificación de actividades y sus respectivas relaciones de precedencia
- Gráfica de la red
- Estimación de tiempo, costo y recursos

Las fuentes de las estimaciones pueden ser información histórica, valores estándares, pronósticos, funciones de regresión, u otros modelos cuantitativos

#### FASE DE PROGRAMACIÓN DE LA RED

Mediante el uso de algoritmos de avance y retroceso se determina las fechas más temprana y tardía de inicio y fin de las actividades (cronograma del proyecto).

- Las holguras de las actividades
- La duración del proyecto
- Rutas críticas del proyecto
- Adicionalmente se realiza la asignación de recursos y la evaluación costo – tiempo

#### FASE DE CONTROL DE LA RED

- Involucra el seguimiento del proyecto sobre la base de la programación.
- Se realizan las medidas correctivas.

## 2. Modelos de redes para la programación de proyectos

Los modelos de redes para la programación de proyectos consisten en describir en forma de red de actividades y sus relaciones, la forma en que se deberá ejecutar un proyecto. Estos modelos se dividen en:



No restringido por los recursos

- Análisis del camino crítico (CPM)
- Aceleración de actividades y reducción de programación (Evaluación de tiempo – costo)

Restringido por los recursos

- Técnicas heurísticas
- Técnicas de programación matemática

## 2.1 Diagrama de redes de proyectos

Es una representación esquemática de la lógica de las relaciones o secuencias entre las actividades del proyecto. Existen dos tipos:

- Activity On Arrow (AOA) o Arrow Diagramming Method (ADM)<sup>1</sup>
- Activity On Node (AON) o Precedence Diagramming Method (PDM)<sup>2</sup>

PRINCIPALES DIFERENCIAS ENTRE AOA Y AON




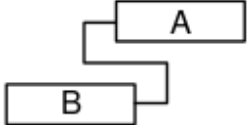
AOM (ADM)	AON (PDM)
Maneja solo relaciones de dependencia (Finish to Start)	Maneja cuatro tipos de relaciones de dependencia, aprovechando el paralelismo parcial de las actividades.
No permite manejar tiempo entre tareas	Permite manejar tiempo entre tareas
Requiere el uso de actividades ficticias entre tareas, lo que genera más cálculo.	No requiere el uso de actividades ficticias entre tareas. Opcionalmente, solo para los extremos del proyecto.
Es de mayor simplicidad e ideal para fines didácticos	Es implementado por la mayoría de software
	Presenta fechas de comienzo más realistas de las actividades iniciales del proyecto.

## 2.2 Tipos de dependencias entre tareas

<sup>1</sup> Actividad sobre flecha o Método de diagramación sobre flecha.

<sup>2</sup> Actividad sobre nodo o Método de diagramación por precedencias.



	<p>Finish to Start (FS) (Fin a comienzo)</p>	<p>La tarea (B) no puede comenzar hasta que finalice la tarea (A)</p>
	<p>Start to Start (SS) (Comienzo a comienzo)</p>	<p>La tarea (B) no puede comenzar hasta que comience la tarea (A)</p>
	<p>Finish to Finish (Fin a Fin)</p>	<p>La tarea (B) no puede finalizar hasta que finalice la tarea (A)</p>
	<p>Start to Finish (Comienzo a Fin)</p>	<p>La tarea (B) no puede finalizar hasta que comience la tarea (A)</p>

#### B. CASO DE APLICACIÓN

ACME es una fábrica de computadoras personales. Actualmente está en el proceso de diseño, manufactura y comercialización del modelo ACME 2002. Existen tres grandes tareas a realizar:

- Manufacturar la nueva computadora.
- Entrenar el nuevo staff y vendedores
- Publicitar la nueva computadora.



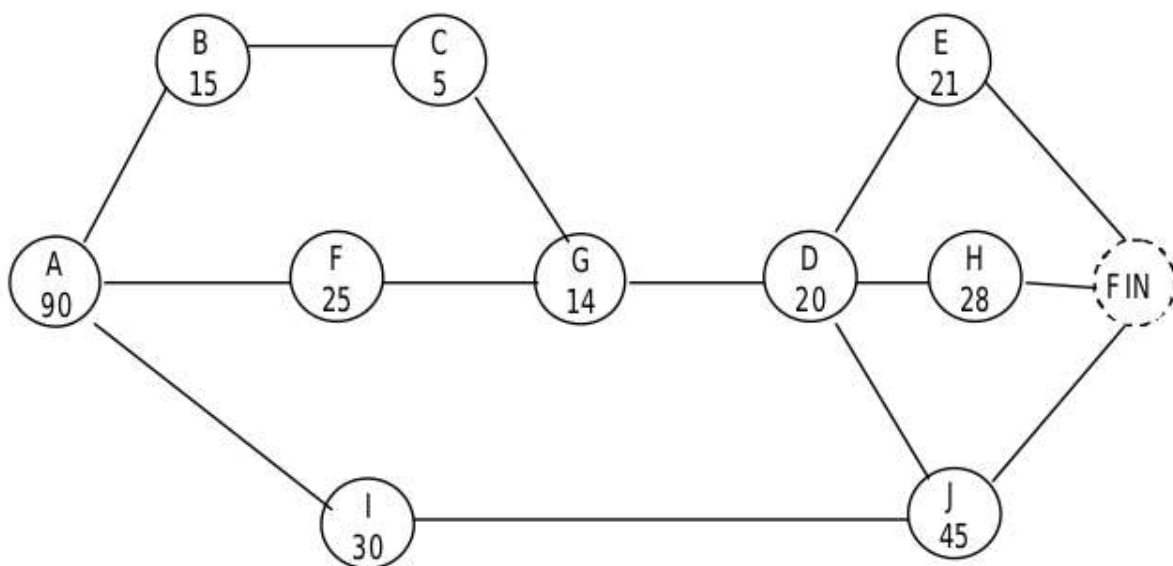
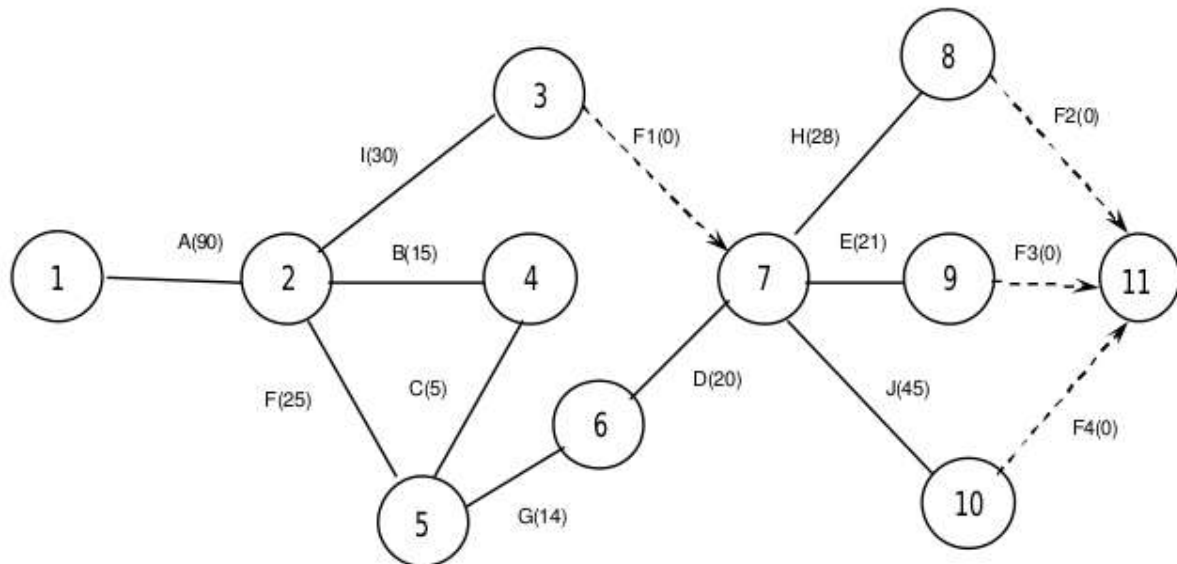
**UNIVERSIDAD DE EL SALVADOR EN LÍNEA**  
**FACULTAD DE INGENIERÍA Y ARQUITECTURA**  
**HERRAMIENTAS DE PRODUCTIVIDAD**



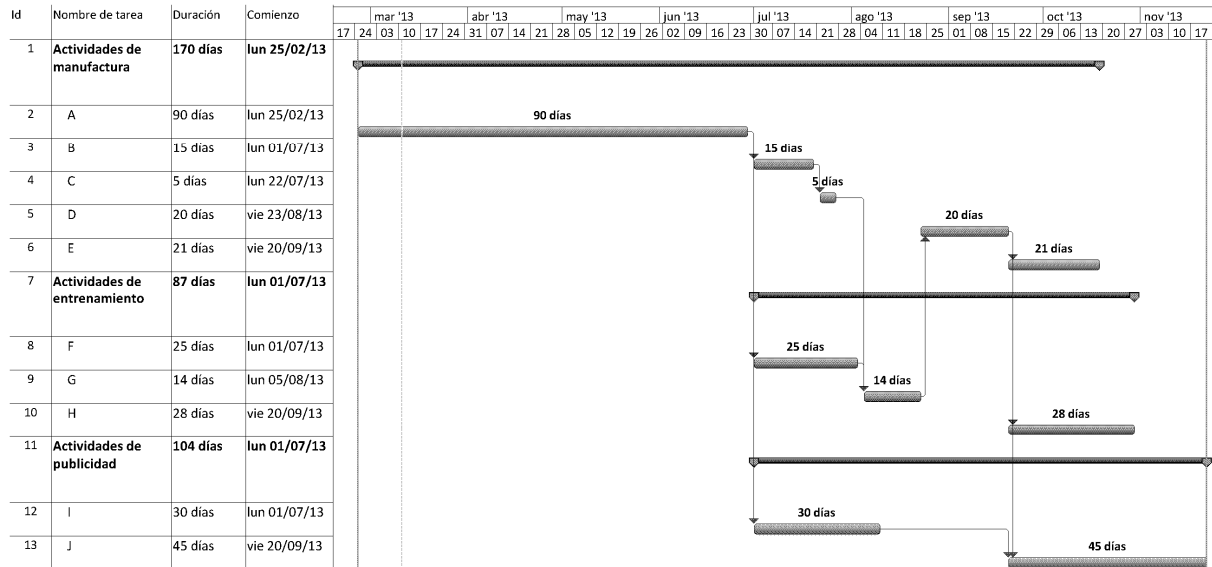
	ID	DESCRIPCION	TIEMPO ESTIMADO DE DURACION	PREDECESORAS INMEDIATAS
Actividades de manufactura	A	Diseño del prototipo del modelo	90	-
	B	Compra de materiales	15	A
	C	Manufactura del prototipo del modelo	5	B
	D	Revisión del diseño	20	G
	E	Producción de lote inicial	21	D
Actividades de entrenamiento	F	Entrenamiento general del staff	25	A
	G	Entrenamiento del staff en el prototipo del modelo	14	C, F
	H	Entrenamiento del personal de ventas	28	D
Actividades de publicidad	I	Pre-producción de la campana de publicidad	30	A
	J	producción de la campana de publicidad	45	D, I

N°	ACTIVIDAD	TIEMPO ESTIMADO DE DURACION	PREDECESORAS INMEDIATAS Y TIEMPOS DE FINALIZACION	PROGRAMACION DE ACTIVIDADES	
				DE	A
1	A	90		0	90
2	B	15	A(90)	90	105
3	C	5		90	115
4	D	20		90	120
5	E	21	B(105)	105	110
6	F	25	C(110), F(115)	115	129
7	G	14	G(129)	129	149
8	H	28	D(149)	149	170
9	I	30		149	177
10	J	45	D(149), I(120)	149	194

*Este material ha sido proporcionado al estudiante en el marco de su formación a través de una carrera en línea en la Universidad de El Salvador. Se han respetado los derechos de autor para su elaboración. El debido uso del mismo es responsabilidad del estudiante.*

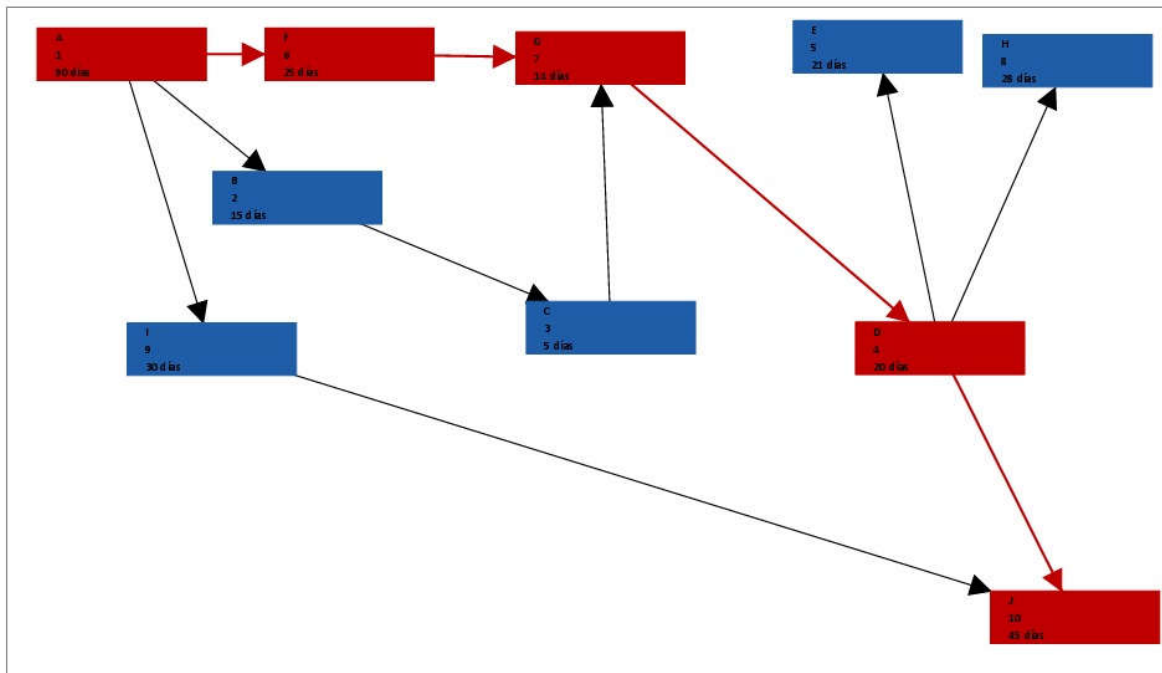


**UNIVERSIDAD DE EL SALVADOR EN LÍNEA**  
**FACULTAD DE INGENIERÍA Y ARQUITECTURA**  
**HERRAMIENTAS DE PRODUCTIVIDAD**



Estadísticas del proyecto 'EjemploGuia2'			
	Comienzo		Fin
Actual	lun 25/02/13		jue 21/11/13
Previsto	NOD		NOD
Real	NOD		NOD
Variación	0d		0d
	Duración	Trabajo	Costo
Actual	194d	0h	\$0.00
Previsto	0d	0h	\$0.00
Real	0d	0h	\$0.00
Restante	194d	0h	\$0.00
Porcentaje completado:			
Duración: 0%      Trabajo: 0%			
			Cerrar

Este material ha sido proporcionado al estudiante en el marco de su formación a través de una carrera en línea en la Universidad de El Salvador. Se han respetado los derechos de autor para su elaboración. El debido uso del mismo es responsabilidad del estudiante.



### 3. Proyectos de desarrollo colaborativos

El concepto de Desarrollo colaborativo es relativamente nuevo, situado principalmente en el contexto de los procesos de enseñanza-aprendizaje, al referirse al Aprendizaje colaborativo.

En el contexto de la informática, podemos definir el concepto de Desarrollo colaborativo de software como un modelo de desarrollo de software distribuido, donde el flujo de trabajo descansa principalmente en tecnologías de informática.

El concepto de modelo distribuido se refiere al uso de tecnologías de informática, especialmente de comunicación, que permitan establecer flujos de trabajo controlados, sin que esto signifique necesariamente que todos los miembros del equipo se encuentren físicamente en el mismo lugar o zona geográfica.

Es muy frecuente que el concepto de desarrollo colaborativo se asocie principalmente a proyectos de software libre. Dado que son proyectos abiertos, cualquier persona interesada puede formar parte del equipo de desarrollo, aportando sus propios conocimientos y experiencias. Sin embargo, este modelo puede reproducirse en proyectos que no son públicos o abiertos, en aquellos escenarios en los que se requiere la descentralización y distribución de las actividades del proyecto.



### 3.1 Características

1. Un modelo de desarrollo descentralizado y distribuido
2. Basado en un conjunto de tecnologías que permiten al equipo una comunicación fluida
3. Flujos de trabajo (Workflows) claramente definidos soportados por herramientas de informática.
4. Mecanismos de seguimiento implementados a través de herramientas de control.

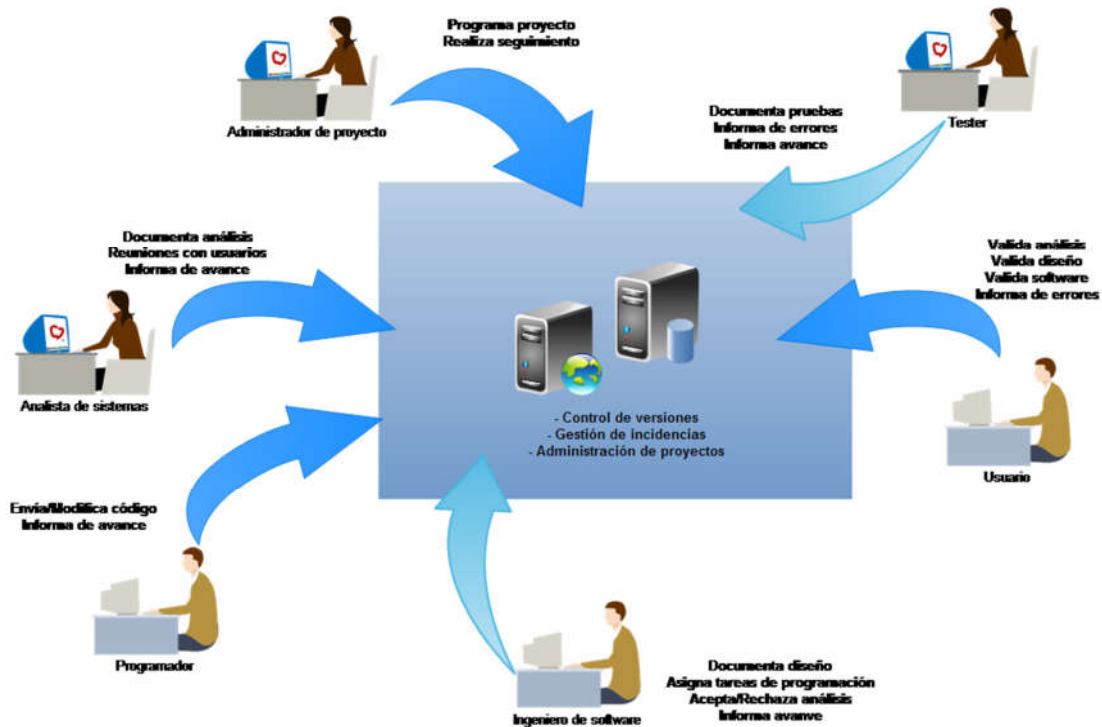
### 3.2 Organización de proyectos de desarrollo colaborativo

La organización de un proyecto de desarrollo colaborativo depende en primer lugar de la metodología de desarrollo que se adopte para cada proyecto. Sin embargo, la característica de distribuido permite incorporar al proyecto a miembros del equipo con distintos niveles de formación, competencias y experiencias.

El modelo también plantea la necesidad de definir claramente roles de los equipos que en algunas metodologías son pasados por alto por cuestiones de tiempo o infraestructura. Algunos de los roles pueden ser:

- Administrador del proyecto.
- Ingeniero de software
- Analista
- Programador
- Usuarios líderes
- Usuarios finales
- Testers (probadores)
- Documentadores
- Analistas de procesos
- Técnicos de control de calidad

Una vez definida la estructura organizacional del proyecto, es necesario contar con una plataforma que garantice la funcionalidad, para ello existen herramientas para la gestión y ejecución de proyectos de desarrollo colaborativo.



#### 4. Herramientas para la gestión de proyectos

En la actualidad, en un modelo de desarrollo distribuido, casi todas las herramientas que permitan mantener a los miembros del equipo comunicados y con capacidad de enviar resultados e informar el avance pueden considerarse herramientas para el soporte de proyectos colaborativos, esto incluye herramientas de comunicación como chats, video conferencias, etc.

Sin embargo, nos centraremos en tres de estas herramientas, para lo que definiremos los conceptos de Cliente y Servidor.



Los conceptos de Cliente y Servidor son utilizados muy frecuentemente en la ingeniería de procesos. El cliente es identificado como cualquier entidad externa que requiera un servicio de algún elemento del sistema que se está diseñando, sea esto un software, un proceso, un equipo, etc. Y el servidor es el sistema del que se trate o uno de sus elementos.

Un servidor puede ser un cajero, un técnico de soporte, una persona de atención al cliente, un técnico electricista, el software, un impresor, una máquina en un proceso de producción, etc. Un sistema puede contar con varios servidores en paralelo o en serie.

#### 4.1 Sistemas para la gestión de proyectos

El sistema de administración de proyectos con el que usualmente tenemos contacto es MS Project, que nos permite programar un proyecto y realizar un seguimiento del mismo.

Hasta hace algunos años, la mayoría de las herramientas para programación y seguimiento de proyectos operaban en estaciones de trabajo aisladas (stand alone), permitiendo el registro del avance de cada actividad para determinar el avance total del proyecto, la ejecución del presupuesto de acuerdo a la programación y otras variables que un administrador de proyectos debe conocer.

Sin embargo, estas versiones permitían que una sola persona a la vez, fuera capaz de actualizar los avances del equipo. Con el avance en las capacidades de comunicación y procesamiento, las versiones tipo Servidor comienzan a tomar auge. Estas versiones permiten a los miembros del equipo, actualizar en tiempo real los avances directamente sobre el proyecto, lo que permite a los administradores tener información fluida sobre los avances del proyecto, para poder tomar decisiones.

Existen muchas alternativas de software comercial para este tipo de herramientas, una de ellas y quizá la más conocida es Project Server. Otras alternativas de tipo libre, se listan a continuación:

##### a. Achievo



- b. ClockingIT
- c. Todoyou
- d. WebCollab
- e. Redmine
- f. EGroupware Community Edition
- g. dotProject
- h. Collabtive
- i. NetOffice
- j. Open Atrium

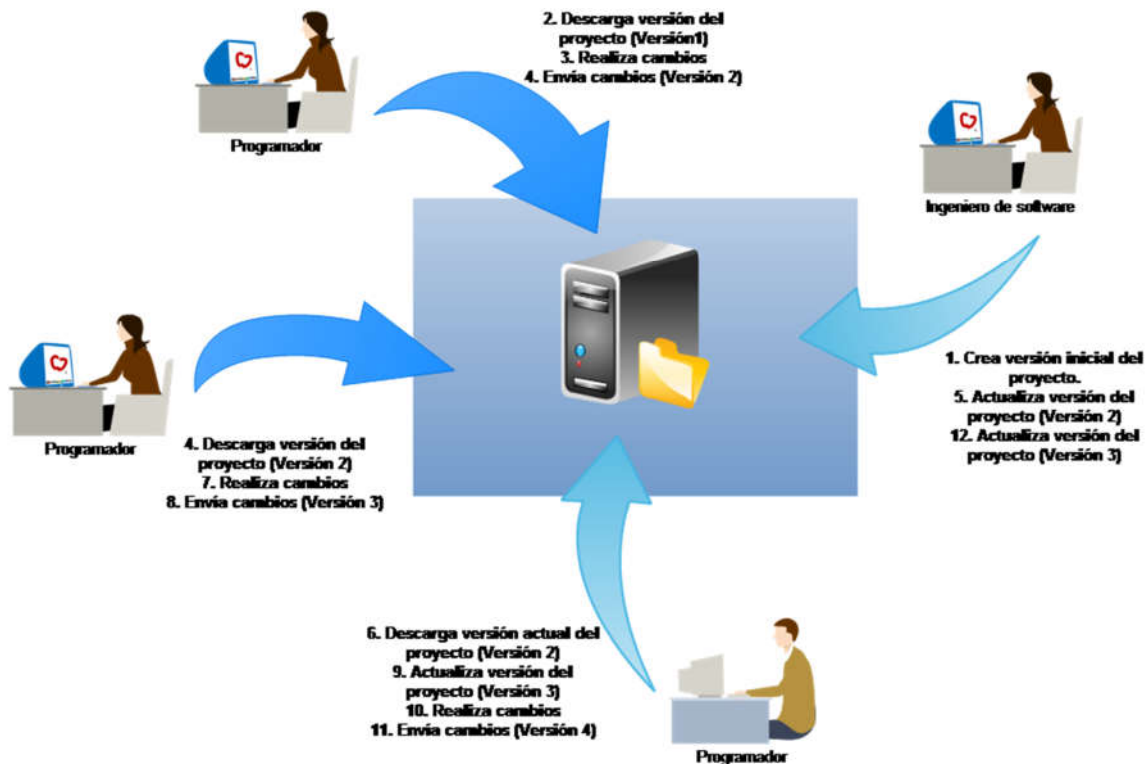
La mayoría de estas aplicaciones permite la administración de múltiples proyectos, programación de las actividades, asignación de recursos, controles de costos, registro de avances por parte de los miembros del equipo, etc.

#### 4.2 Sistemas de control de versiones

Los Sistemas de control de versiones (SCV) fueron creados originalmente para permitir el control de las versiones del código de un software en el proceso de desarrollo. Actualmente existen herramientas para el versionado de documentos, archivos binarios en formatos específicos y por supuesto código fuente.

El flujo de trabajo de un SCV supone que los programadores del equipo comparten el mismo proyecto en forma completa o parcial. Cada uno realizará sus cambios en la forma en que lo necesite y luego este código será enviado al SCV donde quedará registrado como una nueva versión.





Los SCV se basan en un repositorio o servidor en el que se almacenarán las versiones que se vayan recibiendo desde las estaciones de trabajo.

Los SCV basan su funcionamiento en una revisión (o versión) local (estación de trabajo de un miembro del equipo) de los archivos. Estos archivos pueden ser modificados y posteriormente enviados y sustituidos a la copia que se encuentre guardada en el repositorio, lo que genera una nueva revisión del proyecto.

La mayoría de los SCV permiten realizar las siguientes operaciones:

- **Import.** Permite enviar al repositorio, la primera versión de un proyecto.
- **Checkout.** Permite descargar la versión del proyecto que se indique. Por defecto es la versión actual que usualmente es llamada HEAD.
- **Update.** Permite actualizar la versión local de los archivos a la versión indicada. Por defecto la versión actual.
- **Commit.** Permite enviar los cambios realizados localmente a la copia del servidor, lo que generalmente produce una nueva versión del proyecto.
- **Merge.** Permite realizar una mezcla de dos versiones de código. Usualmente es usada cuando existe algún conflicto entre ambas versiones.
- **Branches.** Son ramas de versiones de un proyecto. Cada una de las ramas tendrá una línea de evolución independiente, aunque es posible que en algún momento puedan ser mezcladas.



En la terminología de SCV se distingue la forma de colaborar con un proyecto:

- **Exclusiva.** El repositorio impedirá que se realicen modificaciones si otro usuario ha solicitado realizar modificaciones a los mismos elementos.
- **Colaborativa.** El repositorio permitirá que se modifiquen elementos del proyecto, y posteriormente intentará mezclar los cambios.

Las arquitecturas de almacenamiento de los SCV pueden ser:

- Centralizados. Existe un repositorio centralizado al que todos los miembros del equipo enviarán sus modificaciones.
- Distribuido. Además de un repositorio centralizado, existen repositorios locales que son capaces de llevar su propio control de versiones de los elementos, y es hasta que el miembro del equipo decide enviar su versión local más reciente (HEAD) al repositorio central que podrán ser vistos por el resto del equipo. No todos los SCV permiten contar con repositorios distribuidos.

Los SCV permiten establecer cuál será el flujo de trabajo para el control de las versiones. Los flujos de trabajo dependerán de la arquitectura que se adopte para el versionado, las políticas de colaboración y centralización.

Los sistemas de control de versiones más populares son:

- a. CSV
- b. Subversion (SVN)
- c. Bazaar
- d. Mercurial
- e. Git

#### 4.3 Sistemas de gestión de incidencias

Los sistemas de gestión de incidencias permiten llevar un registro de los incidentes que ocurren en un proceso específico. Son utilizados en una gran gama de procesos como: Soporte técnico, Seguimiento de errores, Validación de software, Mesas de trabajo (Help desk), entre otros.

En el proceso de desarrollo de software son generalmente utilizados para pruebas y gestión de errores, aunque también permiten dar un seguimiento al proceso de desarrollo asignando actividades en forma de incidencias.

El término más importante en este tipo de sistemas es el de Incidencia (Ticket). Los flujos de trabajo se centran en las incidencias de manera que el cliente crea una incidencia que deberá ser resuelta por un servidor. Usualmente el primer servidor que esté disponible o el que más tiempo lleve en ocio.



Estos sistemas también se basan en un repositorio de incidencias, y un registro del flujo de trabajo por el que ha pasado. Esto es especialmente útil en los casos en los que una incidencia deba pasar por varias estaciones de servicio (arreglo de servidores en serie) antes de darse por cerrada (resuelta o abortada).

Algunos de los Sistemas de gestión de incidencias más conocidos son:

- Trac
- Bugzilla
- NetOffice
- osTicket

#### 4.4 Sistemas de gestión de requerimientos

La gestión de requerimientos es un proceso transversal al proceso de desarrollo. Es el proceso de documentar, analizar, seguimiento, priorización y aceptación de los requerimientos. Lo que implica además el control de los cambios por parte de los usuarios.

Los requerimientos están presentes en todas las fases del ciclo de vida de desarrollo, independientemente de la metodología adoptada para el desarrollo:

- a. **Enunciación.** El requerimiento es identificado y enunciado.
- b. **Análisis.** Son evaluadas y propuestas las alternativas de solución.
- c. **Diseño.** El diseño del software deberá permitir satisfacer todos los requerimientos.
- d. **Construcción y pruebas.** Las pruebas del sistema persiguen comprobar si los requerimientos han sido satisfechos.

Los sistemas de gestión de requerimientos persiguen dar soporte a todo el proceso de desarrollo desde el punto de vista de los requerimientos. Así, los requerimientos pueden ser asignados a los miembros del equipo de desarrollo de forma individual o en grupos de requerimientos, usualmente componentes o módulos.

Posteriormente, estos requerimientos pueden ser sometidos a pruebas en cada fase: pruebas unitarias, pruebas de rendimiento, pruebas de estrés, pruebas de integración y pruebas de aceptación.

Los sistemas de gestión de requerimientos están enfocados a estos dos aspectos: la fase de programación y la fase de pruebas.

Este enfoque permite que la los Sistemas de gestión de incidencias

#### 4.5 La nube

Cada vez son más sistemas y servicios que se ofrecen en lo que se llama “La nube”. La computación en la nube o Cloud computing es un conjunto de tecnologías que permiten ofrecer sistemas y servicios de alta disponibilidad para resolver los problemas más críticos del software:

**UNIVERSIDAD DE EL SALVADOR EN LÍNEA**  
**FACULTAD DE INGENIERÍA Y ARQUITECTURA**  
**HERRAMIENTAS DE PRODUCTIVIDAD**



- **Disponibilidad.** Uno de los factores más críticos para cualquier persona o institución es tener disponible siempre que lo necesite, los sistemas y servicios que requiere para realizar sus funciones. La computación en la nube es una forma de trasladar este problema al proveedor del servicio.
- **Accesibilidad.** Los servicios de computación en la nube suponen una alta accesibilidad desde cualquier estación de trabajo que pueda conectarse a la nube.
- **Rendimiento.** Es un factor de calidad que hoy en día ya no es opcional. Las empresas que optan por contratar servicios en la nube, delegan a la empresa proveedora de dichos servicios las responsabilidades de las actualizaciones del hardware y software que se requiera.

Hoy en día existen plataformas de gestión de proyectos en la nube, que prestan sus servicios de forma gratuita o pagada tales como Zoho Project, Assembla y Launchpad.

En la mayoría de los casos, estos servicios cuentan con varios o todos los servicios de gestión y control.