Administración de Proyectos de Software

Estudiantes:

Camacho Baina Albert

Terrazas Sanabria Litzie

Docente:

Mgr. Indira Camacho

Universidad Mayor de San Simón

Cochabamba- Boliva

¿Qué es administrar?

“Es el proceso de lograr que las cosas se realicen por medio de la planeación, organización, delegación de funciones, integración de personal, dirección y control de otras personas, creando y manteniendo un ambiente en el cual la persona se pueda desempeñar entusiastamente en conjunto con otras, sacando a relucir su potencial, eficacia y eficiencia, logrando así fines determinados”.

Procesos y actividades
Actividades de la Administración

El trabajo del administrador varía de acuerdo a la organización y al producto de software a ser desarrollado, por lo que es imposible una descripción de trabajo estándar, sin embargo algunos de los aspectos a considerar se describen a continuación.

Actividades de la Administración

Actividades de responsabilidad de un administrador de software son:

1. **Redacción de propuestas de desarrollo**
   - Objetivos del proyecto y cómo se va a desarrollar
   - Incluye estimaciones de coste, tiempo, asignación a equipos,...
2. **Planificación y calendario del proyecto:** identificación de actividades, hitos y entregas del proyecto
3. **Estimación económica del proyecto**

Procesos y actividades

Actividades de la Administración

4. **Supervisión y revisión del proyecto**
   - Actividad continua
   - Conocimiento del progreso
   - Comparación de progreso y coste con lo planificado
   - Mecanismos formales e informales
5. **Selección y evaluación del personal**
6. **Redacción y presentación de informes**
   - Informes para el cliente, organizaciones contratantes e internos
   - Documentos concisos y coherentes
   - Presentaciones en las revisiones de progreso
   - Administrador: necesidad de comunicación efectiva oral y escrita

Procesos y actividades
Procesos de la Administración

Grupo de Procesos	Procesos
Modelado del ciclo de vida	• Selección de un modelo de ciclo de vida
Administración del proyecto	○ Inicio del proyecto ○ Supervisión y control del proyecto ○ Administración de la calidad del software
Pre-desarrollo	• Exploración de conceptos • Asignación del sistema
Desarrollo	○ Análisis: Se hacen modelos del sistema y ○ establecen requerimientos. ○ Diseño: Se separa el sistema en componentes. ○ Codificación: Codificación de cada componente.
Pos-desarrollo	• Instalación • Operación y soporte • Mantenimiento • Retiro
Procesos integrados	○ Verificación y validación ○ Administración de la configuración del software. ○ Desarrollo de la documentación ○ Entrenamiento

Funciones del Administrador

La administración puede verse como un proceso. Según Fayol, está compuesto por funciones básicas:

- **PLANIFICACION:** procedimiento para establecer objetivos y un curso de acción adecuado para lograrlos.
- **ORGANIZACION:** proceso para comprometer a dos o más personas que trabajan juntas de manera estructurada, con el propósito de alcanzar una meta o una serie de metas específicas.

Funciones del Administrador

- **DIRECCIÓN:** función que consiste en dirigir e influir en las actividades de los miembros de un grupo o una organización entera, con respecto a una tarea.
- **COORDINACIÓN:** integración de las actividades de partes independientes de una organización con el objetivo de alcanzar las metas seleccionadas.
- **CONTROL:** proceso para asegurar que las actividades reales se ajusten a las planificadas.

Elementos que debe Coordinar el Administrador de Proyecto de Software

Los administradores de software son responsables de la planificación y temporalización del desarrollo de los proyectos.

- Supervisan el trabajo asegurando que se lleve a cabo conforme a los estándares requeridos.
- Supervisan el progreso comprobando que el desarrollo se ajusta el tiempo previsto y al presupuesto.

La administración es necesaria debido a que la Ingeniería de Software siempre esta sujeta a restricciones organizacionales de tiempo y presupuesto.

Elementos que debe Coordinar el Administrador de Proyecto de Software

- **Elementos:**

1. Equipos = Conjuntos de participantes que trabajan en un problema común.
2. Papeles = Conjunto de responsabilidades. Los papeles se usan para distribuir el trabajo a participantes de un equipo.
3. Productos de trabajo = Productos finales e intermedios a entregar de un proyecto (resultados visibles).
4. Tareas = Son el resultado de separar el trabajo en función de pasos secuenciales para generar uno o más productos.
5. Calendarios = Correspondencia entre un modelo de tareas y una línea de tiempo.

Dificultades en la Administración

Los administradores de software hacen el mismo tipo de trabajo que otros administradores, pero existen diferentes aspectos los que lo hace difícil.

El producto es intangible:

- No se puede ver ni tocar.
- Los administradores no pueden ver el progreso.
- Confían en otros para elaborar la documentación.

Dificultades en la Administración

No existen procesos del software estándar.

- Los procesos de software varían de una organización a otra.

Los proyectos grandes son únicos.

- Los proyectos grandes son diferentes a proyectos previos.
- Aunque se cuente con experiencia no es suficiente para anticipar los problemas.
- Los cambios tecnológicos y comunicaciones hacen parecer obsoleta la experiencia previa.

Una gestión eficaz se centra en 4P's

- **1. PERSONAL**

  El factor humano siempre será el más importante en el desarrollo de soluciones software, muchos empresarios famosos, líderes de empresas tecnológicas, coinciden que el éxito que han alcanzado sus empresas no se debe a las herramientas que utilizan, es la gente y el trabajo en equipo.

4P’s del Administrador de Proyectos de Software

- **2. PRODUCTO**

  Muchas veces cuando un cliente pide que le construyan una solución, siempre pregunta. ¿Cuánto me va a costar? Pues bien, todo producto requiere:

  - **Estimaciones cuantitativas y una adecuada planificación.**
  - **Una adecuada recolección de información y un análisis detallado de los requerimientos**.

  Con una buena planificación se puede estimar el tiempo que tomará desarrollar o construir el producto y redimensionar el valor cuantitativo del producto.

4P’s del Administrador de Proyectos de Software

- **3. PROCESO**

  Proporciona un marco de trabajo desde el cual se puede establecer un plan detallado para la construcción del software.

  El gestor del proyecto debe elegir el modelo de procesos adecuado para ser aplicado para:

  - **Los clientes que han solicitado el producto y el personal que hará el trabajo.**
  - **Las características del producto.**
  - **El ambiente del proyecto en el que trabaja el equipo de desarrollo del software.**

4P’s del Administrador de Proyectos de Software

- **4. PROYECTO**

  Cuando se gestiona un proyecto exitoso, es necesario entender que este puede llegar a fracasar. Según John Reel, existen 10 razones :

  1. El personal de software no entiende las necesidades del cliente.
  2. El ámbito del producto está mal definido.
  3. Los cambios se gestionan mal.
  4. La tecnología elegida cambia.
  5. Las necesidades comerciales cambian.

4P’s del Administrador de Proyectos de Software

- **4. PROYECTO**

  6. Los plazos de entrega no son realistas.

  7. Los usuarios se resisten a la utilización del software.

  8. Se pierde el patrocinio.

  9. El equipo del proyecto carece de personal con las habilidades apropiadas.

  10. Los gestores evitan las mejores prácticas y las lecciones aprendidas.

  Para tener éxito es necesario comenzar con pie derecho, esto se lo logra trabajando duro para entender el problema y dar una solución adecuada.

¿Calidad en el proceso?

Modelo de Madurez de la Capacidad del Desarrollo de Software.

Este modelo establece un conjunto de procesos clave agrupados en Áreas Clave de Proceso (KPA - Key Process Area). Para cada área de proceso define un conjunto de prácticas que habrán de ser:

- **Definidas en un procedimiento documentado**
- **Provistas (la organización) de los medios y formación necesarios**
- **Ejecutadas de un modo sistemático, universal y uniforme (institucionalizadas)**
- **Medidas**
- **Verificadas**

Modelo de Madurez de la Capacidad del Desarrollo de Software.

A su vez estas Áreas de Proceso se agrupan en cinco "niveles de madurez“.

1 - Inicial.

- Las organizaciones no disponen de un ambiente estable para el desarrollo y mantenimiento de software.
- Los esfuerzos se ven minados por falta de planificación.
- El éxito de los proyectos se basa en el esfuerzo personal, a menudo se producen fracasos, retrasos y sobre costes.
- El resultado de los proyectos es impredecible.

Modelo de Madurez de la Capacidad del Desarrollo de Software.

2 - Repetible.

- Las organizaciones disponen de unas prácticas institucionalizadas de gestión de proyectos.
- Existen unas métricas básicas y un razonable seguimiento de la calidad.
- La relación con subcontratistas y clientes está gestionada sistemáticamente.

Modelo de Madurez de la Capacidad del Desarrollo de Software.

3 - Definido.

Cuentan con una buena gestión de proyectos

Las organizaciones disponen de:

- Procedimientos de coordinación entre grupos. Formación del personal
- Técnicas de ingeniería más detalladas y un nivel más avanzado de métricas en los procesos.
- Se implementan técnicas de revisión por pares (peer reviews).

Modelo de Madurez de la Capacidad del Desarrollo de Software.

- **4 - Gestionado.**

  Se caracteriza porque las organizaciones disponen de un conjunto de métricas significativas de calidad y productividad, que se usan de modo sistemático para la toma de decisiones y la gestión de riesgos.

  El software resultante es de alta calidad.

Modelo de Madurez de la Capacidad del Desarrollo de Software.

- **5 - Optimizado.**

  La organización completa está volcada en la mejora continua de los procesos.

  Se hace uso intensivo de las métricas y se gestiona el proceso de innovación.

Modelo de Madurez de la Capacidad del Personal.

- **1.- Nivel inicial.**
  - Inconsistencia en la ejecución de prácticas.
  - Desplazamiento de la responsabilidad
  - Prácticas ritualistas y personal emocionalmente distante.
- **2.- Nivel Gerenciado.**
  - Sobrecarga de trabajo
  - Distracciones ambientales, objetivos o retroalimentación confusa del nivel de desempeño.
  - Carencia de conocimientos o habilidades relevantes, comunicación pobre, moral baja.

Modelo de Madurez de la Capacidad del Personal.

- **3.- Nivel Definido.**
  - Hay practicas básicas de desempeño del personal e inconsistencia en cómo estas prácticas y poca sinergia al interior de la organización.
  - La organización pierde oportunidades de estandardizar prácticas de personal.
- **4.- Nivel Fiable.**
  - La organización maneja y explota la capacidad creada como marco de capacidades de la mano de obra.
  - La organización puede ahora manejar su capacidad y desempeño cuantitativamente.
  - La organización es capaz de predecir su capacidad para realizar el trabajo porque puede cuantificar la capacidad de su mano de obra.

Modelo de Madurez de la Capacidad del Personal.

- **5.- Nivel de Optimización.**
  - La organización entera se centra en el mejoramiento continuo.
  - Estas mejoras se llevan gracias a la capacidad de individuos y de grupos de trabajo, al desempeño de procesos basados en competencias, y en las prácticas y actividades del personal.
  - La organización utiliza los resultados establecidos en las actividades cuantitativas de la gerencia en el nivel evolutivo 4, para las mejorías en el nivel 5.

¿Cómo medimos el proceso y el producto?
Métricas

La medición es fundamental para cualquier disciplina porque nos permite hacer una evaluación objetiva.

DEFINICIÓN:

El IEEE Standard Glossary of Software Enginnering Terms define:

“Métrica como una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado”

Volver a Estimación de Costos

¿ Cómo medimos?

- Las mediciones en el mundo físico pueden ser catalogadas en dos campos:
  - *medidas directas* (por ej. La longitud de un tornillo),
  - *medidas indirectas* (por ej. Calidad de tornillos producidos, medida por la cuenta de los tornillos rechazados).

¿ Qué medir y para qué medir?

- Proceso - duración, costo, efectividad o eficiencia
- Producto - tamaño, calidad
- Recursos - magnitud, costo, calidad

… Algunas métricas de Calidad

Aunque hay muchas medidas de la calidad de software, la *corrección, facilidad de mantenimiento, integridad,* y *facilidad de uso* proporcionan indicadores útiles para el equipo del proyecto.

- **Corrección:**
  - Un programa debe operar correctamente o proporcionará poco valor a sus usuarios.
  - La medida más común de corrección es *defectos por KLDC.*

… Algunas métricas de Calidad

- **Facilidad de mantenimiento:**
  - Facilidad con la que se puede corregir un programa si se encuentra un error, se puede adaptar si su entorno cambia, o mejorar si el cliente desea un cambio de requisitos.
  - Una simple métrica orientada al tiempo es el *tiempo medio de cambio (TMC),* es decir el tiempo que se tarda en analizar la petición de cambio, en diseñar una modificación adecuada, en implementar el cambio, en probarlo y en distribuir el cambio a todos los usuarios.

… Algunas métricas de Calidad

- **Integridad.**

  Para medir la integridad, se tienen que definir dos atributos adicionales: amenaza y seguridad.

  - ***Amenaza*** **es la probabilidad de que un ataque de un tipo determinado ocurra en un tiempo determinado.**
  - ***Seguridad*** **es la probabilidad de que se pueda repeler el ataque de un tipo determinado.**

  La integridad del sistema se puede definir como:

$$\text{integridad} = \sum [(1 - \text{amenaza}) \times (1 - \text{seguridad})]$$

… Algunas métricas de Calidad

- **Facilidad de uso:**

  Si un programa no es «amigable con el usuario», frecuentemente está abocado al fracaso,

  Se puede medir en función de cuatro características:

  1. Habilidad intelectual y/o física requerida para aprender el sistema;
  2. Tiempo requerido para llegar a ser moderadamente eficiente en el uso del sistema;
  3. Aumento neto en productividad (sobre el enfoque que el sistema reemplaza) medida cuando alguien utiliza el sistema moderadamente y eficientemente;
  4. Valoración subjetiva (a veces obtenida mediante un cuestionario) de la disposición de los usuarios hacia el sistema.

Herramientas CASE

Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como:

- Proceso de realizar un diseño del proyecto
- Calculo de costes
- Implementación de parte del código,etc

Clasificación de Herramientas CASE

- Se pueden clasificar teniendo en cuenta los siguientes parámetros:
  - Las plataformas que soportan.
  - Las fases del ciclo de vida del desarrollo de sistemas que cubren.
  - La arquitectura de las aplicaciones que les presta apoyo
  - Su funcionalidad.

(Pressman 2002)

Clasificación de Herramientas CASE

- **Upper CASE (U-CASE),** herramientas que ayudan en las fases de planificación, análisis de requisitos y estrategia del desarrollo, usando, entre otros diagramas UML.
- **Middle CASE (M-CASE),** herramientas para automatizar tareas en el análisis y diseño de la aplicación.
- **Lower CASE (L-CASE),** herramientas que semiautomatizan la generación de código, crean programas de detección de errores, soportan la depuración de programas y pruebas. Además automatizan la documentación completa de la aplicación.

Clasificación de Herramientas CASE

- **Integrated CASE (I-CASE), herramientas que engloban todo el proceso de desarrollo software, desde análisis hasta implementación.**
- **MetaCASE, herramientas que permiten la definición de nuestra propia técnica de modelado, se guardan en un repositorio y pueden ser usados por otros analistas, es como si definiéramos nuestro propio UML.**
- **CAST (Computer-Aided Software Testing), herramientas de soporte a la prueba de software.**
- **IPSE (Integrated Programming Support Environment), herramientas que soportan todo el ciclo de vida, incluyen componentes para la gestión de proyectos y gestión de la configuración.**

Clasificación de Herramientas CASE

Por funcionalidad podríamos diferenciar algunas como:

- Herramientas de generación semiautomática de código.
- Editores UML.
- Herramientas de Refactorización de código.
- Herramientas de mantenimiento como los sistemas de control de versiones.

Taxonomía q´utiliza como criterio principal la función:

- *Herramientas del ingeniería de procesos de negocio*: Al modelar los requisitos de información estratégica de una organización, las herramientas CASE proporcionan un metamodelo, del cual se derivan sistemas de información específicos.
- *Modelado de procesos y herramientas de gestión*: Estas herramientas ayudan a entender el proceso de una organización (o software). Puesto que para mejorarlo lo primero que hay que hacer es comprenderlo.
- *Herramientas de planificación de proyectos*: Estas herramientas por centran en dos áreas: estimación de costos y de esfuerzos y la planificación temporal del mismo.
- *Herramientas de análisis de riesgos*: El análisis de los riesgos, plan para mitigar, monitorizar y gestionar estos riesgos es fundamental en proyecto grandes. Estas herramientas hacen posible que el gestor del proyecto construya una tabla de riesgos proporcionando una guía detallada en la identificación y análisis de riesgos.

Taxonomía q´utiliza como criterio principal la función:

- *Herramientas de gestión de proyectos*: El plan de proyecto y su planificación deberán ser rastreados y monitorizados de forma continua. Así mismo se hará necesario que se recojan métricas que proporcionen una indicación de la calidad del producto del software (estas herramientas son una extensión de la planificación de proyecto)
- *Herramientas de seguimiento de requisitos*: Cuando se desarrollan grandes sistemas, el sistema suele no satisfacer los requisitos especificados por el cliente. El objetivo de las herramientas es proporcionar un enfoque sistemático para el aislamiento de los requisitos.
- *Herramientas de métricas y de gestión*: Las métricas del software mejoran la capacidad del gestor para controlar y coordina el proceso de ingeniería del software y la capacidad del ingeniero para mejorar la calidad del software que se produce. Las métricas actuales se centran en las características de procesos y productos.
- *Herramientas de documentación*: Estas herramientas apoyan casi todos los aspectos de la ingeniería del software

Taxonomía q´utiliza como criterio principal la función:

- *Herramientas de software de sistema*: CASE es para estaciones de trabajo por lo que el entorno CASE se debe adaptar a las redes y todo el software de comunicaciones disponibles.
- *Herramientas de control de calidad*: La mayor parte de estas herramientas lo que hacen es proporcionar métricas por ejemplo de código fuente: (y hacen auditoría para ver si se ajusta o no a estándares), o extraen métricas técnicas
- *Herramientas de gestión de Bases de Datos*: Este software permite establecer una base de datos CASE (repositorio) o base de datos del proyecto.
- Herramientas de gestión de configuración de software: Estas herramientas ofrecen en las cinco tareas de GCS: identificación, control de versiones, control de cambios, auditoria y contabilidad de estados. Para el proceso de control de cambios se utiliza la BD CASE y otras herramientas especializadas.

Taxonomía q´utiliza como criterio principal la función:

- *Herramientas de análisis y diseño*: Estas proveen el apoyo para que el ingeniero cree modelos del sistema a construir. Por ejemplo para el análisis: representaciones de datos, función y comportamiento, para el diseño: caracterizaciones del diseño de datos, arquitectura, a nivel de componentes y de interfaz, comparación de consistencia y validez de los modelos.
- *Herramientas PRO/SIM*: de construcción de prototipos y simulación, permiten predecir el comportamiento de un sistema en tiempo real antes de llegar a construirlo y efectuar simulaciones.
- *Herramientas de desarrollo y diseño de la interfaz*: Son un conjunto de componentes como botones, menús, estructuras de ventanas, íconos etc.
- *Herramientas de construcción de prototipos*: generadores de pantallas, diseño de datos informes de pantalla, y herramientas PRO/SIM generan un diseño esquemático de código en ADA y C (par software de tiempo real).

Algunas Herramientas CASE

Actualmente la mayoría de los estudiantes de la Universidad utilizamos las siguientes herramientas

- Power Designer – Modelos, diagramas UML
- MySQLFront – Administrador de base de datos
- Eclipse – Desarrollo de aplicaciones JAVA
- NeatBeans – Desarrollo de aplicaciones JAVA
- DreamWeaver – Desarrollo paginas Web.
- JCreator - Desarrollo de aplicaciones JAVA
- ArgoUML - Modelos, diagramas UML

Entorno de desarrollo

Integrated Development Environment ('IDE')

- A diferencia de una herramienta CASE un entorno de desarrollo integrado o en inglés Integrated Development Environment ('IDE') es un programa compuesto por un conjunto de herramientas para un programador.
- Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.
- Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI

Herramientas

- En la sección 6.3.2 Infraestructura (pag.151), del libro calidad en el desarrollo de software nos muestra el conjunto de herramientas como una ejemplificación de : un posible ambiente de integración continua con herramientas integradas y los nodos sobre los cuales fueron instaladas. Ver fig. 6.3, 6.7 y 6.8

Infraestructura

- Leer libro Calidad en el Desarrollo de software, sección 6.3 Integración continua, pag. 149. (toda la sección hasta pag. 6.10).
- Leer https://speakerdeck.com/adrianmoya/infraestructura-de-desarrollo-agil, del blog. Del Ing. Adrian Moya

- **Tarea:** escoger en grupo 3 de las herramientas mencionadas en las lecturas y ¿ explicar por qué las eligieron y con cuál empezarían primero, y por qué?

Bibliografía

- Ingeniería de Software, Somerville
- Ingeniería de Software, Pressman