UNIREMINGTON®
CORPORACIÓN UNIVERSITARIA REMINGTON
RES. 2661 MEN JUNIO 21 DE 1996

ARQUITECTURA DE SOFTWARE

INGENIERÍA DE SISTEMAS

FACULTAD DE CIENCIAS BÁSICAS E INGENIERÍA

Vicerrectoría de Educación a Distancia y virtual

2016

AUTOR

Cesar Augusto Jaramillo Henao
Ingeniero de sistemas
Cesar.jaramillo@remington.edu.co

Nota: el autor certificó (de manera verbal o escrita) No haber incurrido en fraude científico, plagio o vicios de autoría; en caso contrario eximió de toda responsabilidad a la Corporación Universitaria Remington, y se declaró como el único responsable.

RESPONSABLES

Jorge Mauricio Sepúlveda Castaño
Decano de la Facultad de Ciencias Básicas e Ingeniería
jsepulveda@uniremington.edu.co

Eduardo Alfredo Castillo Builes
Vicerrector modalidad distancia y virtual
ecastillo@uniremington.edu.co

Francisco Javier Álvarez Gómez
Coordinador CUR-Virtual
falvarez@uniremington.edu.co

GRUPO DE APOYO

Personal de la Unidad CUR-Virtual
EDICIÓN Y MONTAJE
Primera versión. Febrero de 2011.
Segunda versión. Marzo de 2012
Tercera versión. noviembre de 2015
Cuarta versión. 2016

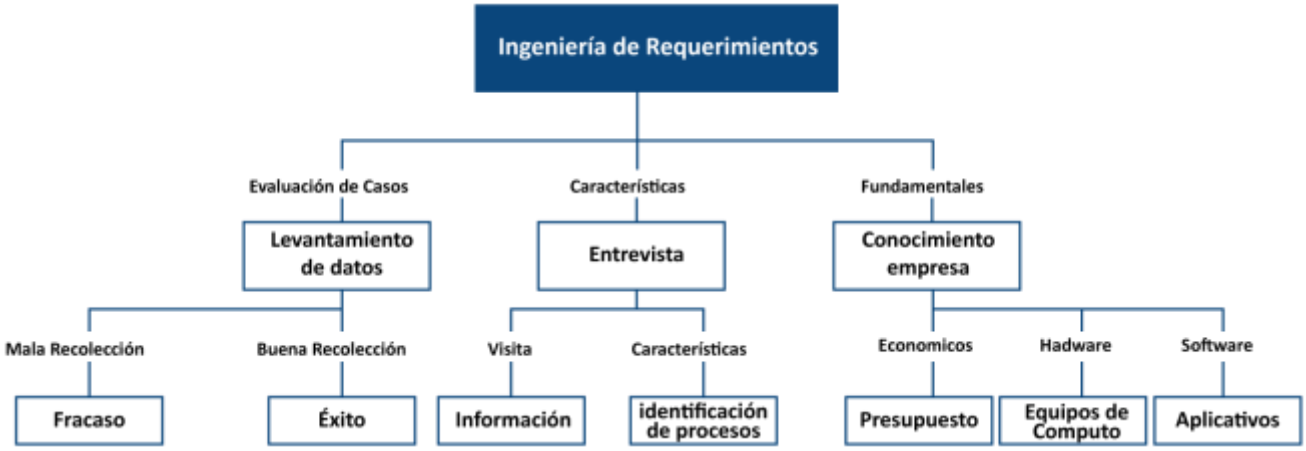
TABLA DE CONTENIDO

Pág.

1 MAPA DE LA ASIGNATURA

2 UNIDAD 1 INGENIERÍA DE REQUERIMIENTOS

RELACIÓN DE CONCEPTOS



Escriba la definición de todos los conceptos planteados en el mapa conceptual

Levantamiento de datos. Un buen levantamiento de datos comprende el conocimiento general de la empresa con todos sus factores, económicos, ambientales, políticos, esto dará una mejor experiencia de usuario y conocer el rumbo del aplicativo nuevo.

Entrevista. La entrevista con los distintos miembros de la empresa permite conocer en detalle los pros y contras del software

Conocimiento empresa: Evitar fracasos depende de la experiencia de la empresa y de la información suministrada por ella

OBJETIVO GENERAL

Profundizar los conceptos de la Ingeniería de requerimientos, los puntos a tener en cuenta, las condiciones de fracaso y de éxito en el desarrollo de software.

OBJETIVOS ESPECÍFICOS

- Reconocer las condiciones mínimas para la realización de un análisis e implementación exitosa en el desarrollo de un aplicativo.
- Tomar decisiones acertadas, evitando replanteamientos en la solución del aplicativo, esto tiene como consecuencia la pérdida de tiempo, esfuerzo y dinero.
- Brindar el mejor acompañamiento en las herramientas, factores económicos y arquitectura a desarrollar.

2.1 TEMA 1 DEFINICIÓN

> La Arquitectura de software es una de las tantas áreas que comprenden el desarrollo de software, no se puede pensar en que la única alternativa es un lenguaje de programación

existen las Bases de Datos, la Ingeniería de software, etc., dentro de las Arquitectura de toman todos los pormenores, **el análisis desde afuera**, la viabilidad, las condiciones físicas, lo actualizado o desactualizado que se tengan las herramientas dentro de la empresa, el Sistema Operativo, sus recursos operativos, entre muchas otras, **este análisis determinara como enfocar el aplicativo**.

Este aplicativo a desarrollar tendrá una proyección de largo alcance, no solo por **su construcción** sino por **su duración**, **no es rentable** para ninguna empresa el **cambiar de tecnología permanentemente**, debe de ser un **aplicativo escalable**, de **fácil manipulación** y de **gran ayuda** en lo solicitado por el propietario.

Es por esto que el arquitecto de software es **un gran conocedor** de **las herramientas**, pero también un **gran explorador** de **las necesidades** de la empresa y de cómo **enfocarlo** de la mejor manera, todo esto llevara a una sola respuesta, **éxito en el desarrollo**.

Componentes

COMPONENTE	CARACTERÍSTICAS
Elicitación	Es el proceso mediante el cual **el cliente** y **el analista** de un sistema de software **descubren**, **revisan**, **articulan** y **entienden las necesidades de clientes y usuario** y **las restricciones del software** y de **la actividad de desarrollo.**
Análisis	Es el proceso de **analizar las necesidades** de **los clientes** y de **los usuarios** para llegar a **la definición** de **los requerimientos** del software.

Especificación	Es **el desarrollo** de **un documento** que, de manera **clara** y **precisa**, **registre** cada uno de **los requerimientos** del software.
Verificación	Es el proceso de **asegurar** que la **especificación de requerimientos cumple** con **las necesidades** de **los clientes** y **usuarios**, **cumple** con los estándares definidos en la organización y es **una base adecuada** para establecer **la arquitectura preliminar** del proyecto.
Administración	Es la **planeación** y **control** de las actividades que ocurren durante el proceso de **desarrollo de los requerimientos.**

PISTAS DE APRENDIZAJE

Recuerde que:

Arquitecto: Es la persona coordinadora de un grupo de desarrollares que designa las condiciones del aplicativo a desarrollar o a mejorar.

Herramientas: Conjunto de opciones de hardware o software que se tienen en para la elaboración de un aplicativo

2.2 TEMA 2 CASOS DE ÉXITO Y FRACASO EN EL DESARROLLO

> Dentro del desarrollo de software como en muchas otras actividades existen buenas y malas prácticas de trabajo, dentro de las malas prácticas esta iniciar un proyecto nuevo sin bases de ninguna clase, sin conocimiento de la empresa ni de sus necesidades más básicas, este tipo de caso aunque se crea extraño es bastante común por la necesidad de mostrar algo o por el afán de un dinero "fácil", más del 85% de los aplicativos que inicia esta forma terminaran en fracaso, el 15% restante con demoras o funcionamientos parciales y un porcentaje muy pequeño tiene éxito.

Otro ejemplo clásico de esto es iniciar la construcción del aplicativo solo con la **información suministrada por el área administrativa** sin **conocer de fondo** donde se **gesta la información**, cuando se hace entrega de un aplicativo con estas condiciones siempre quedará faltando algo y se tendrá que **retomar** desde lo esencial **la construcción del aplicativo**.

Otro caso común es el inicio de un desarrollo sobre **plataformas nuevas**, en el momento de la entrega nos encontramos que **la infraestructura es antigua**, el **sistema operativo** o las terminales **no soportan** dicho aplicativo, **esto requiere una reingeniería de los procesos**, es por esto que se debe **conocer** en detalle **cada una** de las **condiciones** de la empresa.

Todo esto y muchas otras cosas llevaran a un **fracaso seguro** del desarrollo y se tendrá que **iniciar de nuevo** o **cambiar** de **desarrolladores**, esto trae como consecuencia **dinero** y **tiempo**, el hecho de estar **atrasado en tecnología** o **no poder** brindarle al **usuario final** unos resultados **más óptimos.**

En esto aspectos se mencionan algunas de las más comunes, pero tenga presente:

- Factor económico,
- Factor ambiental,
- Software experimental,
- Software construido sobre el que se van a realizar las actualizaciones,
- Sistemas Operativos,
- Restricciones entre departamentos.

Las anteriores y muchas otras son de **vital importancia**.

PISTAS DE APRENDIZAJE

Tenga siempre presente que:

Buenas prácticas: Una buena práctica es el proceso más óptimo y de mayor alcance a futuro.

Malas prácticas: Una mala práctica, aunque sea funcional no cumple estándares y en muy poco tiempo habrá que cambiar o reestructurar lo ya construido.

2.3 TEMA 3 CARACTERÍSTICAS

Antes de cualquier implementación se debe **convertir en un conocedor del tema a desarrollar** (actividad de la empresa), un entrevistador audaz para la adecuada recolección de información, además de contar con el apoyo de los miembros de la empresa para distintas solicitudes o inquietudes que se puedan presentar, todo esto lo dará el tiempo y la experiencia en nuevos proyectos.

Documentación

En esta etapa es fundamental realizar **documentación** de todos los aspectos:

Participantes,
Miembros del equipo,
Analista,
Diseñadores,
Desarrolladores, entre otros

Todas estas opiniones permiten **una visión** más **amplia**, más **general** y sobre todo **sin dejar pasar ningún detalle.**

> Esta documentación será la guía de trabajo y seguimiento de las actividades, esta etapa tendrá como consecuencia unas bases firmes para iniciar con las Bases de Datos y la codificación en el lenguaje de programación

Entrevista

Las entrevistas que se realizan para un desarrollo de software **debe de contar** con **todos los miembros de la empresa**, no solo con los administradores o mandos medios, todo parte del operador o el operario que inicia una actividad, ellos son **los verdaderos conocedores** del **problema** y de que pretenden con el desarrollo, donde están **las falencias**, **las debilidades, los problemas**, es por esto que esta recolección **es vital**, permitirá **alimentar** a los demás miembros del equipo.

Análisis

El análisis de los requerimientos consiste en estudiar las necesidades de los clientes para plantearlas en términos de un sistema de software. Para lograr esto, existen diferentes técnicas como:

- Diagramas de contexto,
- Diagramas de flujo, o
- Diagramas de estado.

UML utiliza **la técnica de Casos de Uso** para analizar **las necesidades** de los usuarios y **estructurarlas** a manera de **servicios** que el sistema debe proveer.

No debe confundirse el análisis en el **contexto de requerimientos** y el análisis en el **contexto de actividades de análisis y diseño del sistema**. En el paradigma de **objetos las actividades** de **análisis** y **diseño** no son siempre **claramente diferenciables** porque un **diagrama de clases** puede implicar:

- El qué del sistema, y
- El cómo.

Por lo tanto, en **el paradigma de objetos**, una clasificación para las actividades del **ciclo de vida** de **desarrollo** que suele usarse es:

- Requerimientos,

- Análisis, y
- Diseño y codificación.

Especificación

> La especificación consiste en la documentación de las actividades durante el proceso de Requerimientos. De acuerdo a la IEEE(1) existe una serie de atributos que debe tener una especificación de software:

ESPECIFICACIÓN	CARACTERÍSTICA
○ Claridad/Carencia de ambigüedad:	Cada uno de los requerimientos tiene **una sola interpretación** posible.
○ Completa	**Todo** lo que el software debe hacer **está incluido** en las especificaciones. Las respuestas a cualquier tipo de datos de entrada en **cualquier situación posible están incluidas** en las especificaciones.
○ Correcta	Cada uno de los requerimientos **representa** algo que **el sistema requiere**.
○ Entendible	**Cualquier tipo de lector** puede, fácilmente, **comprender el significado** de todos los requerimientos con **una explicación mínima.**
○ Verificable	Existe alguna técnica **finita** y **costeable** que puede ser usada para **verificar** que cada uno de los requerimientos especificados **está incluido** en **el sistema terminado**.
○ Consistente	**No existen conflictos** entre requerimientos.
○ Factible	Existe por lo menos **un diseño** y **una implementación** que **satisfacen todos los requerimientos especificados.**

○ Rastreable	Está escrita de manera que **facilita la referencia** de cada requerimiento individual. **El origen** de cada requerimiento **está claro**.
○ Concisa	Es lo **más corta posible**, sin **afectar** otras **cualidades** de la especificación.
○ Diseño Independiente	Existe más de un diseño e implementación que **satisface correctamente** todos los requerimientos del sistema.
○ Modificable	Tiene **una estructura** y **estilo** que permiten **hacer cambios** de una manera **fácil**, **completa** y **consistente**.
○ No redundante	Los requerimientos **no están repetidos.**
○ Precisa	Se usan **cantidades numéricas** cuando es posible con el apropiado **nivel de precisión**.

Verificación

Existen varias técnicas para validar una especificación, algunas de las más importantes son:

VERIFICACIÓN	CARACTERÍSTICA
○ Verificación a través de rastreo	Implica **la revisión** de la documentación por parte de **un especialista**. Algunos autores sugieren que el especialista debe invertir en promedio **30 minutos** en revisar una página de documentación.
○ Pruebas de Validación	Son **pruebas** que se hacen en el software para **comprobar** que este **cumpla** con algún **requerimiento especificado** en la documentación.
○ Pruebas de Aceptación	Son **pruebas** que realiza **el cliente antes de declarar** su **satisfacción** sobre el producto.

PISTAS DE APRENDIZAJE

Recuerde que:

Entrevista: Habilidad de escudriñar y de solicitar detalles de lo que se pretende realizar

Recolección: Encontrar múltiples opciones de información útil para un propósito.

2.4 TEMA 4 ASPECTOS FUNDAMENTALES

Después de pasar por una serie de etapas se tomarán **decisiones** para **la implementación**, **sistematización** o **desarrollo** de **una nueva plataforma**, es en este lugar donde **se determinará** si se **puede realizar** o **no** esta tarea.

¿Por qué no realizarla?, esta fase tiene consecuencias en muchos aspectos, como que:

- El equipo no cuenta con los recursos necesarios para realizar el desarrollo,
- No cuenta con los conocimientos específicos,
- La infraestructura de la empresa es muy vieja y no desea actualizarse,
- Los recursos económicos son muy altos,
- Tiene un sistema que posee problemas desde hace mucho y se desea corregir y agregar nuevos componentes, entre otras.

¿Por qué realizarlo?, Porqué:

- Se cuenta con el conocimiento y las herramientas necesarias,
- La empresa brinda todo el apoyo económico y de infraestructura,
- Existe voluntad de todas las partes,
- Se puede innovar,
- Los recursos son amplios.

Aspectos a tener en cuenta

Económicas

¿Qué restricciones financieras o de presupuesto son aplicables?

Si dentro del desarrollo del proyecto existen restricción económica puede ocasionar una limitante de los procesos esperados, el manejo de licencias, software adicional, capacitaciones, casos de personal externo, todos estos casos y muchos otros podrían genera inconvenientes en el fin del trabajo.

¿Existe alguna restricción de licencias?

El manejo de licenciamiento es un factor de incremento de costos, aunque es conocido que hay software libre, no todos lo son, este manejo de licencias es fundamental porque se puede presentar que se requiera de una sola o de una licencia por máquina que lo opere, es ideal antes del desarrollo conocer de estos costos o informar de que se pueden presentar este tipo de novedad.

¿Una falla puede interrumpir o dañar las operaciones diarias críticas del negocio?

Si, si un proceso es fundamental con transacciones en línea o inclusive locales puede afectar todas las áreas de trabajo

¿Puede este proyecto incurrir o causar pérdidas financieras significantes?

Si, existen muchos riesgos en que se afecte este tipo de operación, imagínese un pago a proveedores, una compra de suministros y todos estos procesos dependen del accionar del sistema de la empresa, pararía todo el flujo operacional

¿Es este un esfuerzo grande en tiempo y dinero?

El desarrollo de software en cualquiera de sus etapas es costoso, mucho más cuando lo que se está implementando cubre todas las áreas o dependencias de la empresa, los esfuerzos crecen en tiempo y dinero a mayor volumen de trabajo en la empresa.

Políticas

¿Existen cuestiones políticas internas o externas que puedan afectar la solución?

Si, algunas decisiones de carácter administrativo o políticas de la empresa podrían afectar el buen funcionamiento del aplicativo o desarrollo del proyecto, esto se puede dar por infraestructura, factores económicos, de espacio, ambientales entre muchas otras.

¿Existen problemas o cuestiones interdepartamentales que puedan afectar la solución?

Cuando la empresa es demasiado grande este tipo de casos son comunes, inclusive las áreas de sistemas pueden estar conformado por departamentos como desarrollo, infraestructura, bases de datos, entre muchos otros, podemos encontrar que un área no cumple las condiciones mínimas para que otra puede operar adecuadamente.

¿Fallar en el proyecto puede dañar la reputación de la empresa?

Completamente, en muchos casos la operación con el nuevo aplicativo o proyecto está ligado a múltiples empresas o personas externas, si el aplicativo falla todos los usuarios tendrán el inconveniente, por ejemplo, un aplicativo bancario que falle afecta a personas naturales y personas jurídicas por igual, la gran afectada es la proveedora del servicio.

¿Este problema no ha podido ser resuelto en el pasado?

Cuando es un problema con largo historial es fundamental tomar medidas importantes, desde la reconstrucción de una operación o salir de este proceso, hace muy poco en Windows 8.1 encontraron un problema que se estaba presentado desde hace 19 años atrás, versión tras versión el problema se presentaba, esto afecta la reputación de la empresa y debió de resolverse mucho tiempo atrás.

¿Existe algún participante que se oponga o tenga muchas dudas del proyecto?

Cuando este tipo de casos se da, se debe de entrar a estudiar el porqué de esta inconformidad, es por esto que los equipos de trabajo deben ser multidisciplinarios con el fin de tener todas las visiones del proyecto y realizar un aporte exitoso, en todos los casos deben de existir argumentos de peso de avalen o que nieguen la construcción de un proyecto.

Técnicas

¿Existe alguna restricción en la elección de la tecnología?

La elección de la tecnología es fundamental, teniendo en cuenta la experiencia del equipo de trabajo y de la plataforma sobre la que se va a trabajar, lo nueva o antigua que esta sea o el alcance que se tiene de esta.

¿Existe alguna restricción para trabajar con las plataformas o técnicas existentes?

Las restricciones se pueden dar por la antigüedad de algunas tecnologías, el soporte de estas puede ser cada vez más complejo o el personal más escaso, lo mismo puede suceder con tecnologías demasiado nuevas o que están a prueba, no es una garantía de buen trabajo.

¿Está restringido el uso de alguna nueva tecnología?

Las nuevas tecnologías siempre buscaran una mejor experiencia de trabajo, pero al ser nuevas se podrán correr algunos riesgos por su poco recorrido en el mercado, además podría presentarse que alguna de estas tecnologías nuevas no se acople o no se adapte a la plataforma que se está construyendo, un ejemplo claro de esto es, en los primeros meses de Windows 10, no había compatibilidad con muchos antivirus, esto ocasionó que los pc estuvieran vulnerables durante algún tiempo por no estar adecuadamente compatible con otras herramientas.

¿Es necesario usar algún paquete de software adquirido por el cliente?

Es muy frecuente que empresas con una larga trayectoria tenga algún tipo de software existente de tiempo atrás, cuando se inicia la construcción de uno nuevo es muy seguro que deba de existir alguna compatibilidad o algún tipo de comunicación para evitar trabajo adicional en el accionar diario de la empresa

¿El producto depende de tecnología experimental?

No es el mejor panorama este tipo de situación, siempre que se trabajó con algo experimental se corre el riesgo de un mal funcionamiento en alguna etapa del proceso.

Si lo anterior ocurre, ¿estará involucrado más de un proveedor o componente crítico?

Es muy frecuente que esto suceda, una entidad pública puede tener 100, 200, 500 o más aplicativos trabajando en simultánea, en estos casos se tienen varios proveedores de aplicaciones o soluciones y de componentes, la tecnología experimental en estos casos es muy crítico, si algo falla parara toda la operación.

¿Existe un alto nivel de complejidad técnica involucrado?

No todos los aplicativos o todas las empresas requieren el mismo tipo de aplicativos o el mismo tipo de proyecto, la especialización de la empresa es el reflejo del software, una empresa comercial no se puede comparar con una empresa automotriz o una empresa de tecnología, la complejidad dependerá de esto y se tendrá que manera un personal desarrollador igualmente capacitado

Sistemas

¿La solución se construirá sobre un sistema existente?

Es más frecuente de lo que se pudiera pensar, la empresa desea su trabajo cotidiano, pero requiere de componentes nuevos sin cambiar tu operación rutinaria, es un proceso complejo y largo porque se debe de partir del conocimiento de la solución existente antes de iniciar las tareas nuevas.

¿Se debe mantener la compatibilidad con alguna solución existente?

Es el ideal, los cambios no se pueden realizar "porque *si*", todo parte de un estudio de riesgos, de lo existente, de lo nuevo, de las tecnologías, la compatibilidad es fundamental para evitar truncamientos en el manejo de la información.

¿Qué sistemas operativos y ambientes deben ser soportados?

Existen muchos aplicativos o herramientas de desarrollo que solo funcionan en un sistema operativo particular, esto es llevadero siempre y cuando para las condiciones se cumplan en el futuro, pero cuando existen múltiples sistemas operativos como Windows, Mac o Linux estos deben de estar soportados de la misma manera por el sistema actual.

Ambientales

¿Existen restricciones regulatorias?

Dependiendo del alcance del aplicativo, los recursos que requiere, sus infraestructuras pueden tener alguna restricción de uso, pero en esencia depende de la funcionalidad y de lo que pueda afectar operaciones externas

¿Existen requerimientos de seguridad?

Con mucha frecuencia estos casos se dan, sobre todo cuando es información o datos de vital importancia para le empresa requiere de un esfuerzo mayor en seguridad.

¿Existen restricciones legales o ambientales?

Las restricciones legales pueden ser más frecuentemente utilizadas por el uso de licencias, o autorización de algún proveedor, en lo ambiental depende del tipo de aplicativo que se esté usando y la finalidad de este.

¿Está involucrada más de una empresa?

En empresas medianas o grandes es común que un desarrollo de proyectos pueda tener varios proveedores que suministren los elementos de trabajo, pero también se da el caso de que el desarrollo creado involucre varias empresas operando dicha herramienta

¿Más de una empresa será impactada por el producto?

En el software actual es muy común de que varias empresas se vean impactadas con el producto, en las app o software para dispositivos móviles puede ser mucho más común.

Calendario y recursos

¿El calendario del proyecto está definido?

Es muy importante que lo esté definido en un calendario, con este se tendrán controles de los procesos y tareas a entregar

¿Es necesario establecer un plan o asignar responsabilidades?

Sí, es necesario, de esto dependerán los tiempos de trabajo, el éxito o fracaso de un proyecto, o la satisfacción o no del cliente.

¿El equipo de trabajo carece de alguna habilidad necesaria?

Es muy probable que esto suceda en tecnologías nuevas o en empresas con una alta complejidad, en muchas ocasiones requiere de esfuerzo extra para buscar o capacitar el personal para este tipo de trabajo

PISTAS DE APRENDIZAJE

Datos: **Información de utilidad para un desarrollo.**

Aplicativa: **Solución a un problema de la empresa mediante software.**

Recursos: **Elementos con los que se cuenta para realizar una actividad.**

2.4.1 EJERCICIO DE APRENDIZAJE

Nombre del taller de aprendizaje: entrevista	**Datos del autor del taller:** Cesar Augusto Jaramillo Henao
Escriba o plantee el caso, problema o pregunta: Es recomendable realizar la entrevista solo a los administradores	

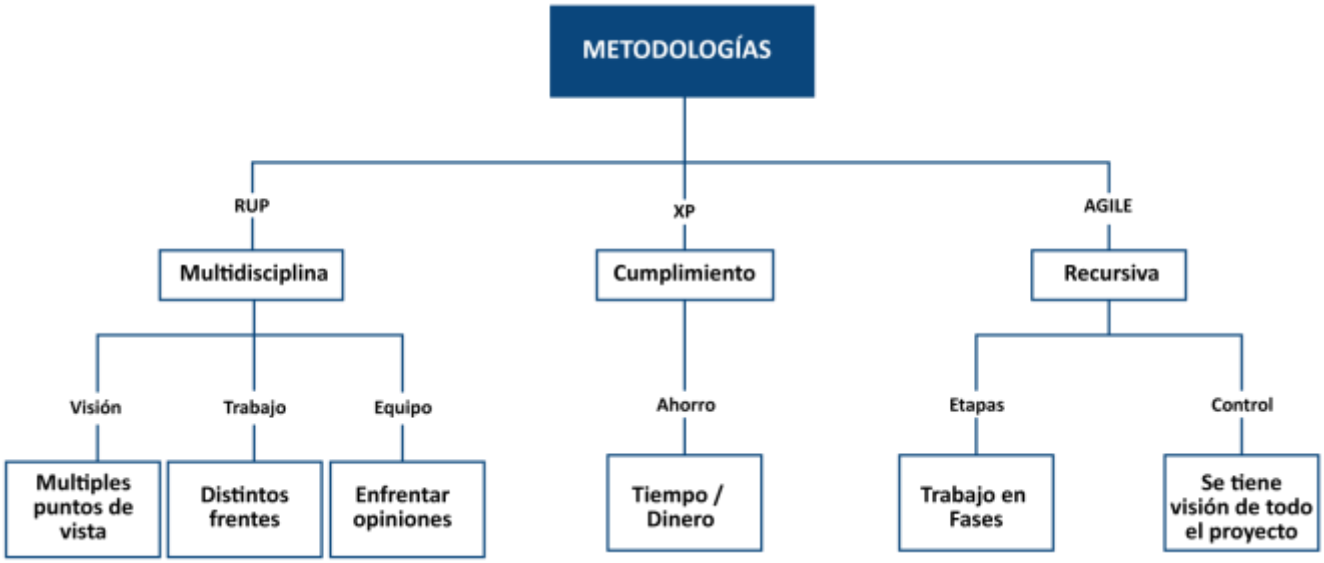
Solución del taller: Aunque sean un elemento fundamental dentro de la empresa, no siempre son los conocedores en detalle de la operación diaria, es recomendable entrevistar a personal que realmente opere con los procesos

2.4.2 TALLER DE ENTRENAMIENTO

Nombre del taller: recolección	**Modalidad de trabajo:** Individual
Actividad previa: Leer detenidamente los componentes de la arquitectura	
Describa la actividad: Entrevistar un familiar, amigo vinculado a una empresa que permita la interacción, la recolección, ver los suministros con los que se cuenta y realizar un documento que describa estos procesos.	

3 UNIDAD 2 METODOLOGIAS DE DESARROLLO

RELACIÓN DE CONCEPTOS



Escriba la definición de todos los conceptos plateados en el mapa conceptual

Multidisciplina: El ambiente RUP permiten involucrar a desarrolladores y no desarrolladores para una mejor visión del trabajo en equipo

Cumplimiento: La metodología XP busca el cumplimiento en tiempo y dinero

Recursivo: Trabajo en etapas, se termina una etapa e inicia la otra, así se tiene un producto funcionando desde muy temprano

OBJETIVO GENERAL

Identificar las diferentes metodologías de desarrollo, sus pros y sus contras, su implementación y sus características principales.

OBJETIVOS ESPECÍFICOS

- Reconocer las principales condiciones de las metodologías de programación
- Identificar los pros y contras de las metodologías y tener alternativas para mezclarlos según las necesidades

3.1 TEMA 1 INTRODUCCIÓN AL RUP

> RUP (RATIONAL UNIFIED PROCESS) proceso unificado relacional es una metodología creada por Rational (IBM), mismos creadores de UML, con la finalidad de especificar proyectos netamente orientados a la programación.

RUP es una forma disciplinada de asignar tareas y responsabilidades en una empresa de desarrollo (quién hace qué, cuándo y cómo).

Requiere un grupo grande de desarrolladores para trabajar con esta metodología.

RUP es un entorno del proyecto que describe una clase de los procesos que son iterativos e incrementales, define un sin número de las actividades y de los artefactos que se necesitan elegir para construir sus propios procesos

Los procesos de RUP determinan tareas y horario del plan midiendo la velocidad de tareas concerniente a sus estimaciones originales.

RUP proporciona muchas ventajas sobre otras metodologías, le da énfasis en los requisitos y el diseño, la ventaja principal es que se basa todo en las mejores prácticas que se han intentado y se han probado en el campo.

RUP se divide en cuatro fases:

- Inicio Define el alcance del proyecto
- Elaboración definición, análisis, diseño
- Construcción implementación
- Transición fin del proyecto y puesta en producción

Planear las 4 fases incluye:

- Asignación de tiempo
- Hitos Principales
- Iteraciones por Fases
- Plan de proyecto.

RUP define nueve disciplinas a realizar en cada fase del proyecto:

- Modelado del negocio
- Análisis de requisitos
- Análisis y diseño
- Implementación
- Test
- Distribución
- Gestión de configuración y cambios
- Gestión del proyecto
- Gestión del entorno

Cada fase en RUP puede descomponerse en otros subprocesos, estos a su vez son ciclos de desarrollo completos dando como resultado una entrega de producto ejecutable (interna o externa).

El proceso define una serie de roles: Los roles se distribuyen entre los miembros del proyecto y que definen las tareas de cada uno y el resultado que se espera de ellos.

Todos los miembros del equipo comparten:

- 1 Base de conocimiento
- 1 Proceso
- 1 Vista de cómo desarrollar software
- 1 Lenguaje de modelamiento (UML)

RUP realiza un levantamiento exhaustivo de requerimientos, **buscando detectar** defectos en las fases iniciales además intenta **reducir** al número de cambios tanto como sea posible y realiza el Análisis y diseño tan completo como sea posible, diseño genérico, intenta **anticiparse** a futuras necesidades.

Las necesidades de clientes no son fáciles de discernir para esto existe un contrato prefijado con los clientes y este **interactúa** con el equipo de desarrollo mediante reuniones a diferencia de otras metodologías en las que **el cliente es parte del equipo**.

PISTAS DE APRENDIZAJE

RUP: Es una metodología creada por IBM para mejorar le experiencia en el desarrollo de aplicativo

Metodologías: Es una buena práctica de desarrollo para contar con la mayor cantidad de alternativas en e desarrollo

3.2 TEMA 2 INTRODUCCIÓN A LA METODOLOGÍA AGILE

A principios de la década del 90, **surgió un enfoque que fue bastante innovador** ya que iba en contra de toda creencia de que mediante procesos altamente definidos **se iba a lograr obtener software en tiempo y costo**. Se dio a conocer en la comunidad de Ingeniería de Software con el nombre de **RAD** o **Rapid Application Development** y consistía en un entorno de desarrollo altamente productivo, en el que participaban grupos pequeños de programadores utilizando herramientas que generaban código en forma automática tomando como entradas sintaxis de alto nivel.

> la historia de las Metodologías Ágiles y su apreciación como tal en la comunidad de la Ingeniería de Software tiene sus inicios en la creación de una de las metodologías utilizada como arquetipo: XP - eXtreme Programming, dada la poca calidad del sistema que se estaba desarrollando.

Es así como que este tipo de metodologías fueron inicialmente llamadas "**metodologías livianas**", aunque no contaban con una aprobación pues se le consideraba por muchos desarrolladores como meramente intuitiva. Luego, con el pasar de los años, en febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace formalmente el término "ágil" aplicado al desarrollo de software.

En esta misma reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software con el objetivo de esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto.

Principales características

- Basadas en heurísticas provenientes de prácticas de producción de código.
- Especialmente preparadas para cambios durante el proyecto.
- Impuestas internamente (por el equipo).

- Proceso menos controlado, con pocos principios.
  No existe contrato tradicional o al menos es bastante flexible.
  El cliente es parte del equipo de desarrollo.
- Grupos pequeños trabajando en el mismo sitio.
- Pocos artefactos.
- Pocos roles
- Menos énfasis en la arquitectura del software.

Ventajas de esta metodología

- Capacidad de respuesta a cambios de requisitos a lo largo del desarrollo
- Entrega continua y en plazos breves de software funcional
- Trabajo conjunto entre el cliente y el equipo de desarrollo
- Importancia de la simplicidad, eliminado el trabajo innecesario

Atención continua a la excelencia técnica y al buen diseño

Mejora continua de los procesos y el equipo de desarrollo

PISTAS DE APRENDIZAJE

XP: Programación extrema, es un tipo de metodología de desarrollo.

Agile: Metodología de trabajo liviana.

3.3 TEMA 3 INTRODUCCIÓN A LA METODOLOGÍA XP

XP (eXtreme programming) programación extrema, XP Nace en busca de simplificar el desarrollo del software y que se lograra reducir el costo del proyecto, no produce demasiado overhead sobre las actividades de desarrollo, una de las principales características es reduce el costo del cambio en las etapas de vida del sistema.

> Se requiere un grupo pequeño de programadores para trabajar con esta metodología entre 2 – 15 personas y estas irán aumentando conforme sea necesario, combina las que han demostrado ser las mejores prácticas de desarrollo de software, y las lleva al extremo, de ahí su nombre.

Se harán pruebas todo el tiempo, no sólo de cada nueva clase conocidas como pruebas unitarias, sino que también los clientes comprobarán que el proyecto va satisfaciendo los requisitos.

Las pruebas de integración se efectuarán siempre, antes de añadir cualquier nueva clase al proyecto, o después de modificar cualquiera existente, las iteraciones serán radicalmente más cortas de lo que es usual en otros métodos, esto permite beneficiarse de la retroalimentación tan a menudo como sea posible.

XP define 4 variables para el proyecto de software:

- Costo
- Tiempo
- Calidad
- Alcance.

XP tiene como valores lo siguiente:

- Comunicación
- Simplicidad
- Realimentación
- Coraje

Este es un conjunto mínimo y consistente de valores que permitirán hacer la vida más fácil del grupo, los administradores y los clientes. Sirve tanto a los fines personales como a los comerciales,

XP deriva una docena de Principios Básicos tales como:

- Realimentación rápida,

- Asumir la Simplicidad,
- El Cambio Incremental,
- Adherirse al Cambio,
- Trabajo de Alta Calidad.

XP desarrolla además 4 actividades que guiarán el desarrollo:

- Codificar
- Testear
- Atender
- Diseñar

Existen 12 prácticas de XP muy comunes:

- Jugar el juego de planificación.
- Hacer pequeños Releases.
- Hacer historias y usar metáforas.
- Diseñar simple.
- Probar –Testear.
- Rearmar – Refactorizar.
- Programar por pares.
- Propiedad Colectiva.
- Integrar Continuamente.
- Semanas de 40 horas.
- Cliente On-Site.
- Usar Standares de Codificación

XP intenta reducir la complejidad del software por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción, tiene una debilidad cuando se utiliza en dominios de aplicaciones complejas o situaciones difíciles en la organización, el rol del cliente no refleja los diferentes intereses, habilidades y fuerzas a las que enfrentan los programadores durante el desarrollo de proyectos.

También se puede decir que es un sistema de prácticas mínimas, le suponen utilizarlas todas en el principio de un proyecto y adaptarlas y agregar los adicionales como cuando usted experimenta la necesidad.

3.3.1 EJERICICIO DE APRENDIZAJE

Nombre del taller de aprendizaje: evidencia	**Datos del autor del taller: Cesar** Augusto Jaramillo Henao
Escriba o plantee el caso, problema o pregunta: Las evidencias o recolección de datos que tan importante puede ser	
Solución del taller: Demasiado, a mejor recolección de información mayor cantidad de alternativas a tener presente desde el inicio del proyecto.	

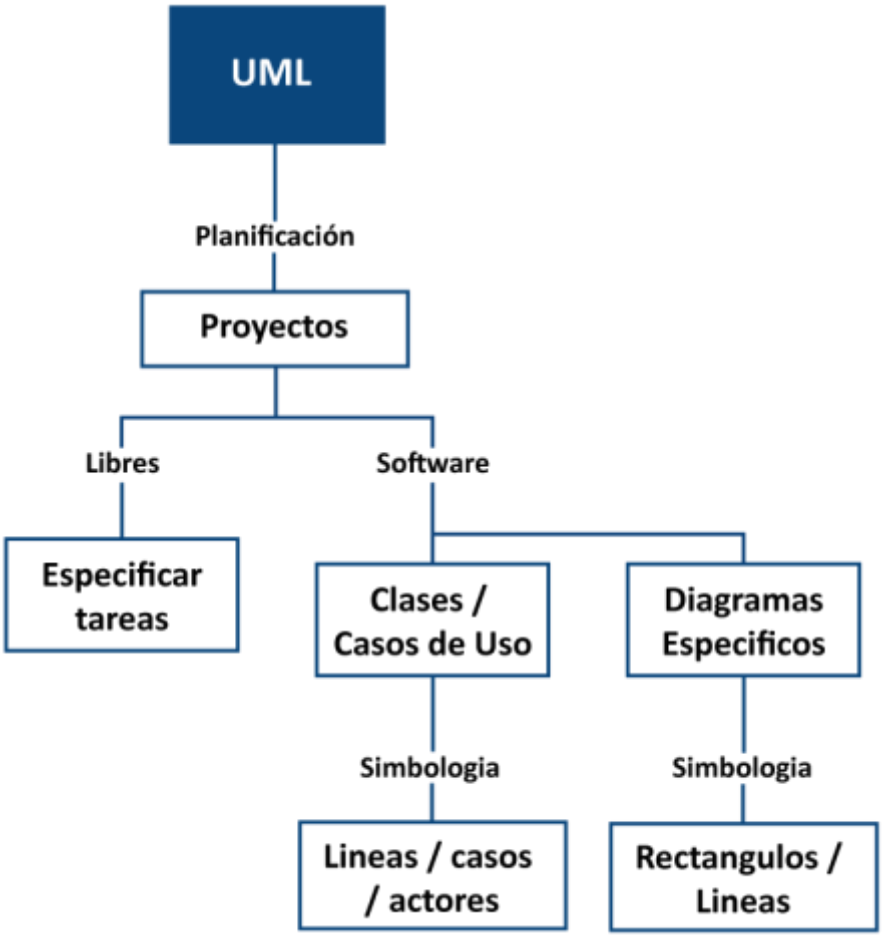
3.3.2 TALLER DE ENTRENAMIENTO

Nombre del taller: innovación	**Modalidad de trabajo:** individual
Actividad previa:	

Leer detenidamente cada metodología de trabajo
Describa la actividad: Diseñe su propia metodología con un derrotero basado en un proyecto de la empresa en la que labora.

4 UNIDAD 3 UML

RELACIÓN DE CONCEPTOS



Escriba la definición de todos los conceptos planteados en el mapa conceptual

Libre: Se pueden realizar proyectos libres sin estar amarrados a el desarrollo de software.

Clases: Diagrama específico para desarrollo que muestra los atributos y los métodos de trabajo.

Simbología: Forma de representar una tarea o un problema legible para cualquier público.

OBJETIVO GENERAL

Aplicar conceptos de UML como herramienta de planificación de proyectos.

OBJETIVOS ESPECÍFICOS

- Reconocer los principios del UML dentro de la programación.
- Identificar dos de los diagramas más comunes y útiles a la hora de programar.

4.1 TEMA 1 HISTORIA

En 1994 luego de la unión de **Grady Booch**, que era experto en el diseño orientado a objetos, **Ivar Jacobson**, el creador del método de ingeniería de software orientado a objetos y **James Rumbaugh** experto en análisis orientado a objetos todos dentro de la compañía Rational y conocidos como los tres amigos por sus constantes discusiones metodológicas, empieza una tarea sobre la mejor forma de realizar los análisis de aplicativos orientados a objetos, en 1996 aparece UML como Lenguaje de Modelos Unificados y 1997 fue mostrado al público.

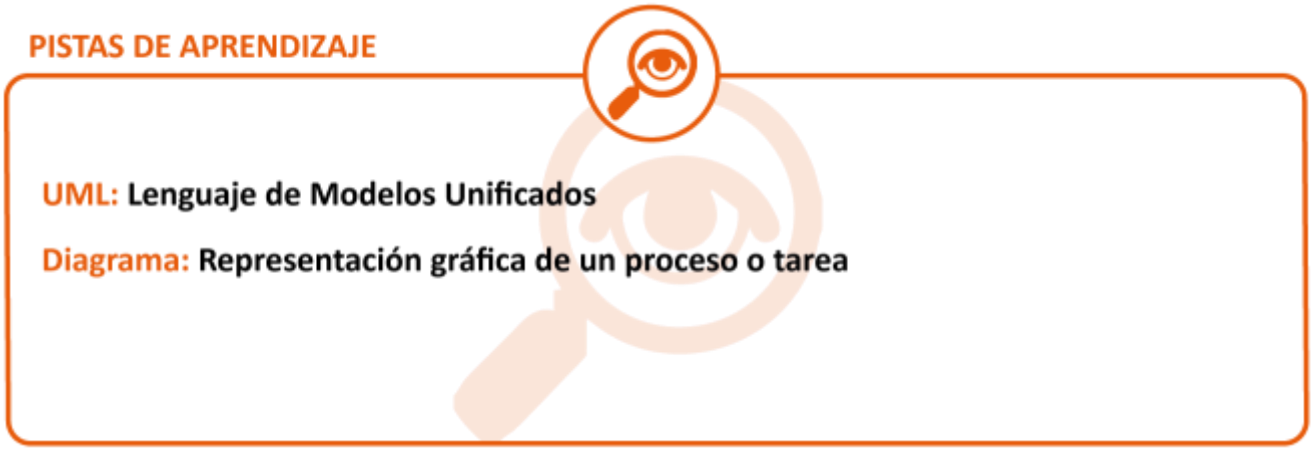
A partir de este momento y hasta la actualizada el ambiente sigue evolucionando en diagramas, métodos, formatos estándar que permiten mediante una herramienta el Esquema de trabajo mediante gráficos que hacen que la interpretación sea **más amigable** para los usuarios en general.

Pista de aprendizaje:

UML: Lenguaje de Modelos Unificados

Diagrama: Representación gráfica de un proceso o tarea

4.2 TEMA 2 CARACTERÍSTICAS FUNDAMENTALES



Con esto se llega a la conclusión que es más corto y económico y **fácil resolver problemas** con imágenes que con código.

La principal característica es conocer su simbología y su gramática para interpretar cualquier tipo de gráfico y el propósito de estos.

El trabajar sin modelos y solamente el BD y un lenguaje de programación generan que más del 80% de los aplicativos terminen en fracaso, los modelos como el UML permiten que exista una etapa previa que llevará de la mano al programador y evitará en el futuro que este porcentaje siga creciendo y que por el contario el software sea cada vez más exitoso.

Tipo de diagramas en UML

Aunque en esta unidad solo se tocaran 2 tipos de diagramas (casos de uso y de clase) por la cercanía con el desarrollo, existen muchos otros que permiten llegar un poco más lejos según el tipo de software que se pretende trabajar.

- Diagramas de cajas de uso
- Diagrama de actividades
- Diagrama de clases
- Diagrama de interacción
- Diagramas de estado
- Diagrama de componentes

Cuanto texto debe complementar los modelos

En el caso particular de UML la representación se aplica mediante gráficos, el texto complementario o adicional debería de ser mínimo, el grafico en sí debería de ser lo suficientemente claro y comprensible mediante su simbología como para tener un texto que lo explique.

PISTAS DE APRENDIZAJE

Metodología: Formas de trabajo ideales para realizar una tarea guiada por pasos y algunos estándares de la industria

Gráficos: Forma práctica de leer e interpretar un proceso con un mimo de texto

4.3 TEMA 3 CASOS DE USO / CLASES

El UML soporta el análisis y el diseño orientado a objetos proporcionándole una manera de captar los resultados del análisis y diseño, en general, se inicia con la comprensión de los problemas, y un tipo excelente de modelo para captar esto es el diagrama de casos de uso.

> La finalidad de un caso de uso es describir la forma de usar un sistema, describiendo sus finalidades esenciales, pero por medio de lo visual.

Un buen caso de uso, bien descrito es uno de los procesos más importantes que se puedan presentar y permite conocer con claridad y organizar los objetivos del desarrollo que se tiene planificado.

Características de los casos de uso

Los casos de uso habitualmente constan de líneas, óvalos, figura de palillos también conocido como actor y que representan a alguien o algo que actúa sobre un sistema, sean estos personas o software, las líneas pueden ser punteadas o pueden ser continuas, con o sin fechas y que nos indican el tipo de relación con el actor y / o con los óvalos, estos últimos son los casos de uso y habitualmente tienen un texto que describe de forma muy corta un proceso.

para iniciar con los casos de uso pensemos en una lista de necesidades, un derrotero, en algunos casos y dependiendo de nuestra experticia en un tema tendremos más argumentos para sacar este derrotero, por ejemplo, pensemos en una serie de pasos para ir a una sala de cine a ver nuestra película favorita de cartelera, y encontramos que dependiendo de nuestro conocimiento y de las ayudas tecnológicas se podrán encontrar múltiples situaciones simpáticas que no se habían tenido en cuenta por ejemplo:

- elegir película
- elegir sitio
- si este sitio es cercano o esta considerablemente distante
- cuál es mi capacidad económica
- deseo reservar o comprar
- el pago lo hago en efectivo, tarjeta crédito, tarjeta debito
- que deseo consumir durante la película
- ingreso a la sala de cine

estas son algunas situaciones típicas, pero con toda seguridad ustedes encontraran otras variantes que posiblemente olvide o que no tuve presente, esto en síntesis en **un derrotero**, son procesos generales, no estoy entrando en detalles como si el pago en efectivo lo hago con monedas o billetes de alguna denominación o si mi desplazamiento lo hago en bicicleta, bus, taxi, metro, etc., simplemente indico que hay un desplazamiento, cuando tenemos esto claro podemos pensar en la creación de **un caso de uso,** pero antes de esto también debemos de pensar en **las prioridades de estas tareas**, por ejemplo no es lo mismo colocar:

- Compra de las boletas
- Elegir película

Este **orden de procesos** no es coherente, el **segundo paso** después de tener mi lista de tareas es **un ordenamiento lógico de las tareas**, posterior a esto tendremos que determinar cuáles son **las dependencias**, por ejemplo, mi capacidad económica con la forma de pago, o el lugar de la película con la cinta que deseo ver, no sería muy lógico desplazarme primero al lugar sin tener certeza de que la película si se esté proyectando en ese lugar.

Tenga muy presente la experiencia, de este dependerá **la cantidad** de alternativas y pasos que se pueden dar.

Elementos:

Actor	
Casos de uso	
Conector línea simple (asociación)	

Autoría Propia

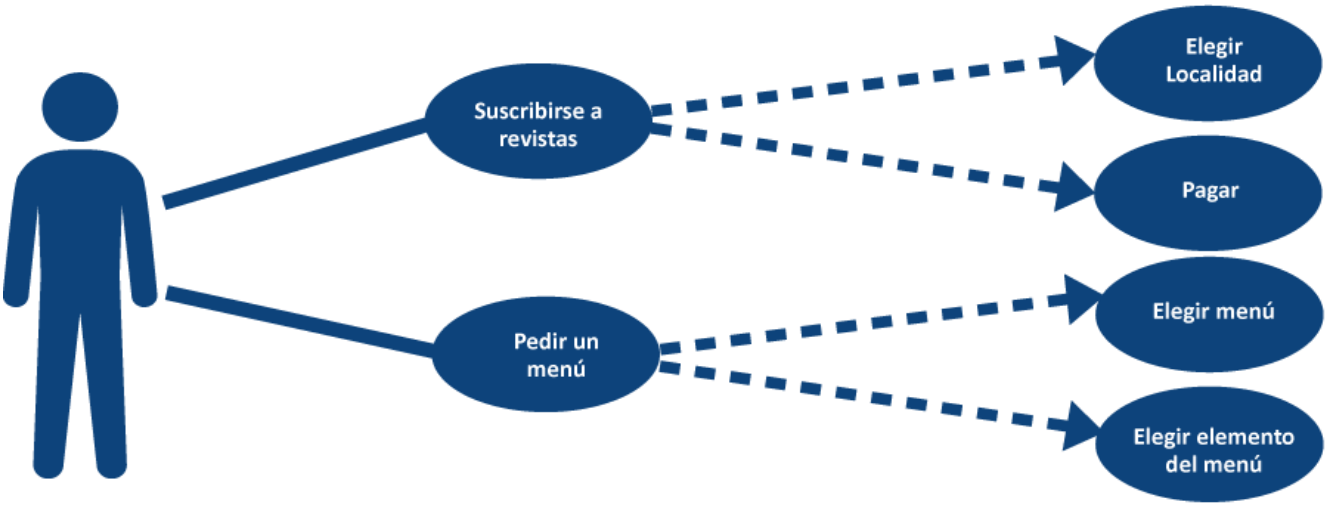
Ejemplo

Autoría Propia

Estamos indicando que la creación de una lista de trabajo para la empresa depende del patrón

Observemos que este tipo de grafico no requiere de mucha explicación, lo podemos intuir sin haber colocado el texto descriptivo

Observemos otro ejemplo de casos de uso



Autoría Propia

Tenga presente, pueden existir varios actores y múltiples casos de uso, así como múltiple tipo de conector, determine cuál es el más indicado según su significado.

Pero ante todo parta de un derrotero, cada punto del derrotero comprenderá un caso de uso, el texto debe de ser resumido y concreto y los conectores darán la lectura que determina esta tarea.

Después de tener estos procesos claros se puede empezar con tareas más complejas, estos diagramas en muchos casos no tienen una sintaxis, es usted el que determina la coherencia de los procesos, usted es el dueño del proyecto, el análisis que se le de mostrar que tareas hay que organizar o complementar.

Existen otros elementos de los casos de uso, pero con estas 5 iniciales, el actor, el caso de uso y los 3 conectores se puede trabajar gran parte de un proyecto, a medida que su experiencia avance podrá utilizar otros que le permitirán tener una visión más amplia de los casos.

Diagrama de clase

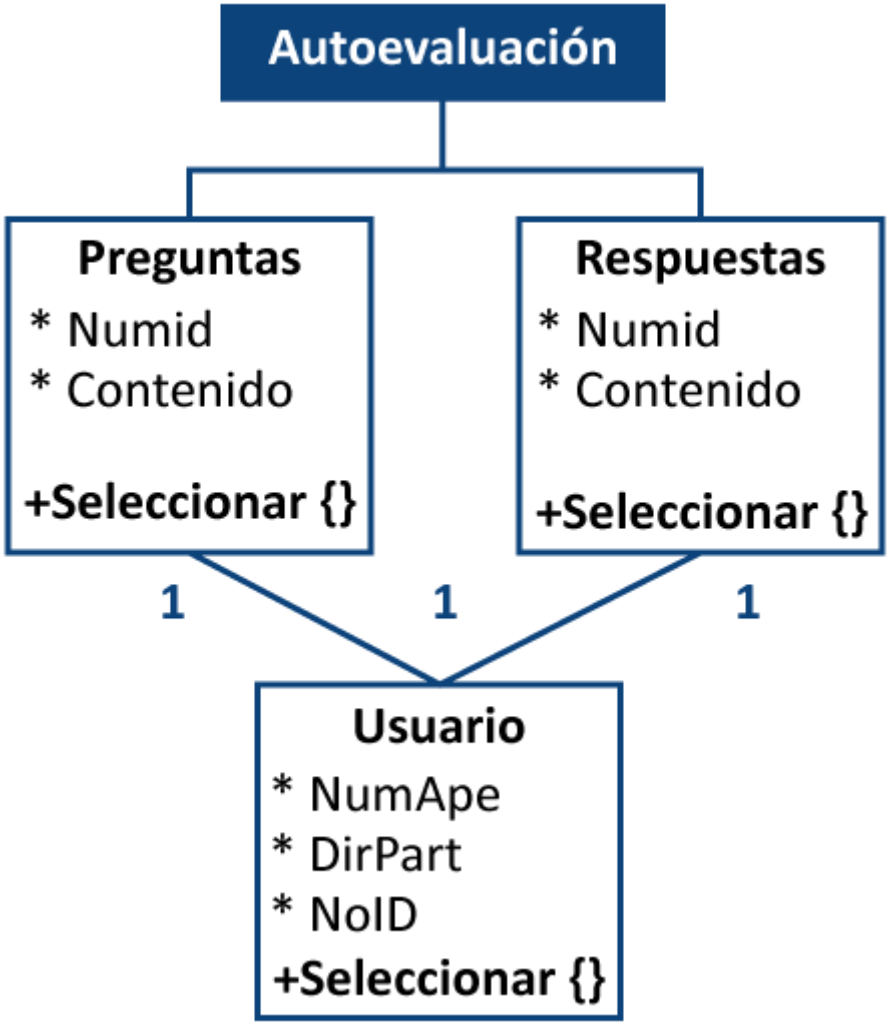
Los diagramas de clases hacen parte de uno de los dos diagramas tratados en este capítulo, tenga presente que esto es un recuento de los procesos, usted ya ha trabajado con anterioridad estos diagramas en otras áreas de la carrera, dentro del diagrama de clases encontramos que los elementos más importantes son rectángulos y líneas:

- Los rectángulos son clases, y
- Las líneas son conectores que muestran la relación entre esas clases.

Nota: Como regla general de este proceso enfoque las clases y sus relaciones.

El rectángulo en un diagrama de clase y se llama clasificador, este puede decir el nombre de la clase, las clases contendrán comportamientos y atributos, llamado también características, los atributos pueden ser campos, propiedades o ambos, los comportamientos se consideran métodos.

Ejemplo:



Autoría Propia

En este diagrama se encuentran 3 clases, la primera de ella es preguntas, que tiene dos atributos NumId y contenido, pero después de la línea divisoria se encuentran los métodos que en este caso se llama seleccionar, en algunos casos las clases van muy de la mano de las tablas, aunque no es una regla general, las líneas que se encuentran en las demás clases indica el tipo de relación entre ellas, similar a un modelo entidad relación.

Otro ejemplo:

EMPLEADO (from mundo)
String nombre String apellido Int sexo String image Int salario
Void inicializar(String p Nombre, String pApellido, intpSexo, Fecha pFechaN, Fecha pFechal, int pSalario) Void cambiarFechaIngreso(Fecha pFechaIngreso) Void cambiarImagen (String pImagen) Void cambiarSalario (int pImagen) int darApellido () String darApellido () int darEDAD () String darFechaIngreso () String darFechaNacimiento () String darImage () String darNombre () double darPrestaciones () int darSalario () int darSexo ()

FECHA (from mundo)
Int dia Int mes Int anio
Void inicializar(int d, int m, int a) void inicializarHoy () int darAnio () int darDia () int darMes () int darDiferenciaEnMeses (Fecha f) String to String()

Este nuevo ejemplo lo vemos un poco más complejo, pero no lo es, en **la primera clase** encontramos empleado, que se compone de **nombre**, **apellido**, **sexo**, **imagen** y **salario**, en la parte inferior se encuentran los métodos que lo componen, esta es **la diferencia fundamental** con los modelos relacionales, es de **vital importancia** respetar los métodos para una guía clara a la hora de iniciar el desarrollo.

PISTAS DE APRENDIZAJE

Diagrama: **Forma gráfica de representar una tarea**

Modelo Relacional: **Conjunto de tablas que se relacionan entre si**

4.3.1 EJERCICIO DE APRENDIZAJE

Nombre del taller de aprendizaje: clases	**Datos del autor del taller:** Cesar Augusto Jaramillo Henao
Escriba o plantee el caso, problema o pregunta: Es indispensable crear un diagrama de clases para un proyecto pequeño.	
Solución del taller: Podríamos decir que no es fundamental, pero por más pequeño que sea un aplicativo debemos de tener la buena práctica de documentar y de crear este tipo de diagrama, nos dará una visión general del problema, de la soluciones y de un control de los elementos que lo componen.	

4.3.2 TALLER DE ENTRENAMIENTO

Nombre del taller: Nomina	**Modalidad de trabajo:** Individual
Actividad previa: Leer e investigar procesos que tienen relación con las clases y los casos de uso	
Describa la actividad: Crear un diagrama de casos de uso y diagrama de clases para una nomina	

5 PISTAS DE APRENDIZAJE

Recuerde que:

Arquitecto: Es la persona coordinadora de un grupo de desarrollares que designa las condiciones del aplicativo a desarrollar o a mejorar.

Herramientas: Conjunto de opciones de hardware o software que se tienen en para la elaboración de un aplicativo

Tenga siempre presente que:

Buenas prácticas: Una buena práctica es el proceso más óptimo y de mayor alcance a futuro.

Malas prácticas: Una mala práctica, aunque sea funcional no cumple estándares y en muy poco tiempo habrá que cambiar o reestructurar lo ya construido.

Recuerde que:

Entrevista: Habilidad de escudriñar y de solicitar detalles de lo que se pretende realizar

Recolección: Encontrar múltiples opciones de información útil para un propósito.

Datos: Información de utilidad para un desarrollo.

Aplicativa: Solución a un problema de la empresa mediante software.

Recursos: Elementos con los que se cuenta para realizar una actividad.

RUP: Es una metodología creada por IBM para mejorar le experiencia en el desarrollo de aplicativo

Metodologías: Es una buena práctica de desarrollo para contar con la mayor cantidad de alternativas en e desarrollo

XP: Programación extrema, es un tipo de metodología de desarrollo.

Agile: Metodología de trabajo liviana.

Codificar: Expresión de los procesos lógicos mediante un lenguaje de programación

Testear: Pruebas unitarias o globales de un proyecto

UML: Lenguaje de Modelos Unificados

Diagrama: Representación gráfica de un proceso o tarea

Metodología: Formas de trabajo ideales para realizar una tarea guiada por pasos y algunos estándares de la industria

Gráficos: Forma práctica de leer e interpretar un proceso con un mimo de texto

Diagrama: Forma gráfica de representar una tarea

Modelo Relacional: Conjunto de tablas que se relacionan entre si

6 GLOSARIO

- **Análisis** : Es la parte fundamental de lo que se pretende realizar
- **Arquitectura**: Es un paso previo a la codificación, comprende el contorno de lo que se pretende desarrollar
- **Casos de uso**: Es un tipo de diagrama básico, apto para cualquier publico este o no familiarizado con el desarrollo de software
- **Diagramas de clase**: Es la forma de representar los campos o atributos y los métodos que lo conformaran.
- **Entrevista:** Recolección de información para la sistematización
- **Gráficos**: La forma en la que se representan los procesos de un aplicativo
- **Ingeniería:** La ingeniería de requerimientos es la ciencia que estudia toda la recolección de información previo a una sistematización.
- **Métodos**: Bloques de código que permiten realizar o complementar una tarea

7 BIBLIOGRAFÍA

- Perdita, STEVENS. Pearson,
- Kimmel, Paul, McGraw Hill,
- Villalobos, Jorge, Pearson