**Introducción.**

En nuestra primera clase, veremos el concepto de software, desde el punto de vista de un proceso y desde la visión como producto. Su evolución, sus características y otros conceptos que complementan la idea de software.

El software si bien es un término y un concepto que podemos tener claro, vamos verlo en la perspectiva de la ingeniería de requerimientos.

**El software en el tiempo.**

La idea del software como programa de una computadora fue concebida por primera vez allá en el año 1935 por el matemático británico conocido por todos nosotros llamado alan turing hace casi 100 años.

Posteriormente recién en el año 1958 fue un científico estadounidense, john Tukey, comenta la siguiente frase:.

“It is better to solve the right problem approximately than solve the wrong problem exactly”.

Para que no queden dudas el software no son solo programas, el software es un conjunto de partes interrelacionadas que alcanzan algún objetivo.

Entender el concepto de software es un medio para poder construirlo y es nuestro objetivo real. Entender su naturaleza nos interesa en la medida que ese entendimiento nos ayude a construir instancias de software por eso nos interesa especialmente la imagen que nos podemos hacer, la imagen que nos hacemos y sobre todo cuál es el papel del software en un contexto mayor. Reducir al software a programas es una simplificación válida para una computadora pero no para quienes lo van a construir .



Por otra parte esta simplificación tiene una consecuencia negativa para el desarrollo de software en cuanto a la productividad de los desarrolladores y la calidad de los productos generados

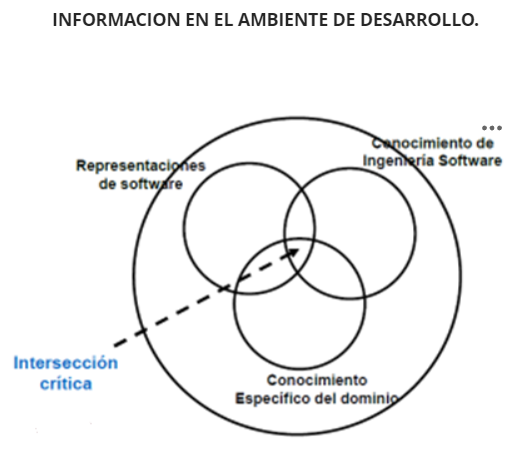
El software es:

* Alma y cerebro de una computadora
* Corporización de las funciones de un sistema
* El conocimiento capturado acerca de un área de aplicación
* Colección de los programas y datos necesarios para convertir a una computadora (de propósito general) en una máquina de propósito especial diseñada para una aplicación particular.
* Información (documentación) producida durante el desarrollo de un sistema software-intensivo.

El software es muchas cosas, pero todos son aspectos de la información

En definitiva es una cuestión de semántica:

* Si software = programas ejecutables, excluimos una cantidad de información que debemos llamar de alguna manera
* Si incluimos toda la información relevante a una pieza de software ejecutable, entonces nos debemos relacionar con esa información en la misma forma rigurosa y sistemática que lo hacemos con el software ejecutable.
* Esto es crucial para un desarrollo exitoso, pues si no se hace, la información se pierde o altera introduciendo errores.



**Representaciones del software**

Cualquier información que en forma directa representa un eventual conjunto de programas y los datos asociados

Incluye:

* programas
* diseños detallados
* diseños de arquitectura (representados como diagramas de estructura)
* especificaciones escritas en un lenguaje formal
* requerimientos del sistema expresados en una combinación de notaciones
* o centenares de posibilidades

**Conocimiento de ingeniería de software.**

Toda la información relativa al desarrollo en general (ej: cómo usar un método específico de diseño) o relativa a un desarrollo en particular (ej: programa de testeo en un proyecto)

•Incluye:

•información relativa al proyecto

•información sobre la tecnología de software (métodos, conceptos, técnicas)

•conocimiento acerca de sistemas similares

•información detallada relativa a la identificación y solución de problemas técnicos del sistema en desarrollo

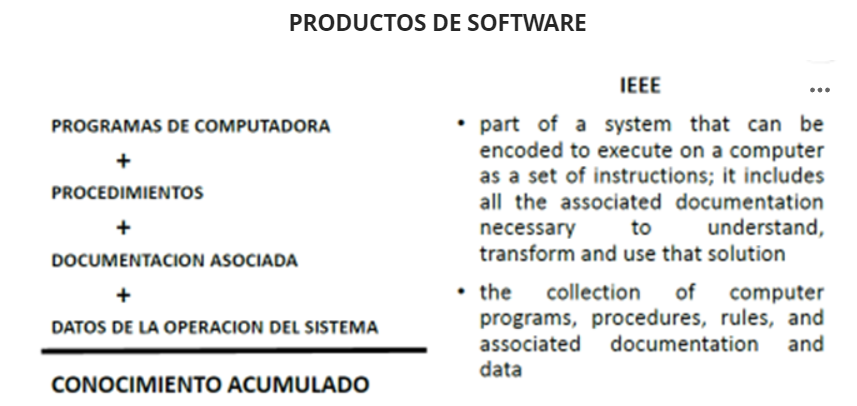
**Ejemplos de formas que toma el software.**

•Colecciones de programas: a testear, terminados, en lenguaje de máquina, en lenguajes de alto nivel

* Necesidades
* Requerimientos
* Especificaciones
* Diseños de arquitectura
* Diseños detallados
* Formatos de datos
* Sistemas en uso para producción
* Análisis de requerimientos
* Documentación del usuario
* Documentación de mantenimiento
* Pedidos de cambio
* Especificaciones de modificaciones
* Informes de errores
* Mediciones de performance
* Punto de vista: estudiar el proceso de desarrollo tratando de ver sus formas en el viaje desde la idea hasta el proceso

**¿EL SOFTWARE ES PRODUCTO O CONOCIMIENTO?**

|  |  |
| --- | --- |
| **Software como producto** | **Software como conocimiento** |
| • A medida que las computadoras extendieron su ámbito de aplicación se los vio como procesadores simbólicos, de allí que no se viera al software como producto.  • Desde los 60 (separación Sw y Hw) comenzó a constituirse como producto.  • El software es tanto un producto como un objeto técnico, esto es: conocimiento empaquetado. | • Razonamiento  1. Los programas contienen conocimiento,  2. El ejecutable es el final de una cadena de representaciones que llamamos software),  3. Entonces las versiones finales también contienen conocimiento  4. Perdemos ese conocimiento si nos limitamos a los conceptos del software ejecutable.  • No perder este conocimiento es una de las principales motivaciones de la reusabilidad del software.  • El software se puede comparar a un informe de investigación. |



**¿POR QUE EL SOFTWARE ES UNICO?**

* Es intangible: dificulta su control y medición
* Posee un alto contenido intelectual
* No se lo reconoce como un activo contable
* Su proceso de desarrollo es mano de obra intensivo, basado en equipos y por proyectos. Esto multiplica el problema de la comunicación, un aspecto clave de los proyectos
* No hay separación entre R&D y producción
* Potencialmente es modificable hasta el infinito y genera problemas a la hora de la evolución del software liberado
* Otras ? Degradación ? Costo de Reproducción ?

**CARACTERISTICAS CONCEPTUALES**

* Objetivo de la ingeniería es la construcción de producto.
* Objetivo de la Ingeniería Software es la: construcción de sistemas de software
* Principal diferencia el software es maleable.
* Existe la idea [errónea] que los cambios en el software son fáciles !!!.
* En este punto debe pensarse a los productos de software de otra forma:
* Un cambio debe ser visto como un cambio en el diseño más que en el código
* Su producción es humano-intensiva: requiere más ingeniería que manufactura. El proceso de producción de software se vincula más con el diseño e implementación que con la manufactura.
* **En la ingeniería clásica el ingeniero dispone de herramientas para describir el producto que son distintas del producto, no es así en la Ingeniería Software.**

***Para el desarrollo de un producto de software se utiliza un proceso.***

***Un proceso es un conjunto de actividades coordinadas y organizadas con ciertas restricciones, utilizando recursos de modo tal de cumplir con objetivos.***

***¡La calidad del proceso afecta la calidad el del producto!***

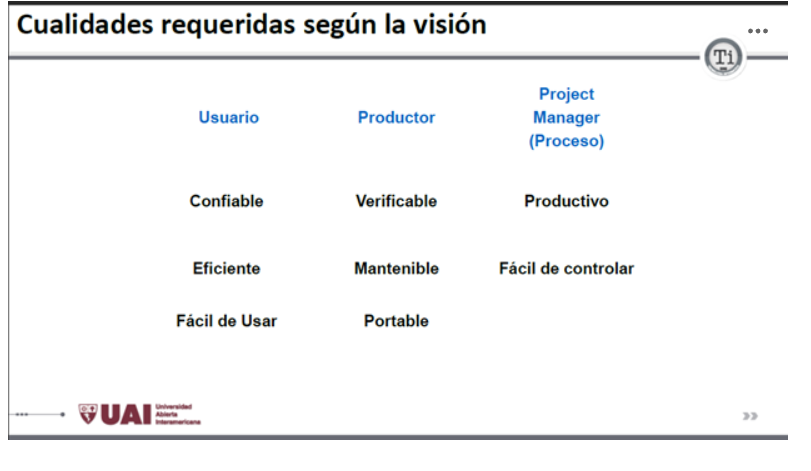
**CUALIDADES DEL SOFTWARE**

**A continuación verá cuales son las cualidades del software que debe cumplir.**

* **Corrección funcional:**se comporta de acuerdo a las especificación de requerimientos funcionales.
* **Confiabilidad:** el usuario puede depender del software
* **Robustez:** se comporta "razonablemente", incluso en circunstancias no previstas en la especificación de requerimientos
* **Performance:** uso económico de los recursos de computación (en Ingeniería Software se la identifica con eficiencia)
* **"Amistosidad":** fácil uso por los seres humanos
* **Verificabilidad:** sus propiedades pueden verificarse fácilmente
* **Mantenibilidad:** puede repararse y evolucionar
* **Reusabilidad:** utilizar componentes sin modificarlas en otros sistemas
* **Portabilidad:** puede correr en distintos ambientes
* **•Comprensibilidad:** facilidad de ser entendidos por los usuarios (desarrolladores)
* **Interoperatividad:** capacidad de coexistir y cooperar con otros sistemas
* **Productividad:** mide la eficiencia del proceso de producción de software
* **Oportunidad:** capacidad de liberar un producto en tiempo
* **Visibilidad:**sus pasos previos y estado actual están correctamente documentados

**Cualidades Internas y Externas.**

* Las cualidades internas sólo son visibles para los desarrolladores. En cambio las cualidades externas son visibles para los usuarios, Pero hay una fuerte relación entre unas y otras.
* Pero debemos tener claro que las cualidades externas no son consecuencia de la internas.
* Pero son las cualidades internas las que tienen que ver en gran medida con la estructura del software las que ayudan a los desarrolladores a que se cumplan las cualidades externas por ejemplo la cualidad interna de verificabilidad es necesaria para conseguir la cualidad externa de fiabilidad o de confiabilidad en muchos casos sin embargo las cualidades están estrechamente relacionadas y la distinción entre integra más externa no es tan tajante.



Es posible pensar en las cualidades requeridas en el software de acuerdo a diferentes visiones.

Es posible pensar en tres ediciones diferentes: el usuario final, el ingeniero de software o el desarrollador o productor y el project manager . Cada uno tiene intereses distintos :

El usuario final pretende que el sistema posea corrección funcional porque debe resolver su problema debe respetar la especificación de requerimientos, debe ser confiable, robusto, fácil de usar.

Al ingeniero del software o desarrollador le interesa que el software sea reusable, portable, comprensible internamente, que puedan interoperar con otros productos, que sea verificable, que se ha mantenido.

El project manager le interesan más aquellas características vinculadas con el proceso de producción de software entonces le interesa fundamentalmente la productividad, la visibilidad, la oportunidad.

**EL SOFTWARE Y LA DESECONOMÍA DE ESCALA**

Hasta ahora usted ha aprendido sobre el software como producto. Un producto que tiene características que lo hacen único. También vio las cualidades deseables del software tanto como producto o como proceso y ahora nos vamos a concentrar brevemente en una característica que es fundamental en el software: el costo que insume la producción del software viéndolo como un proceso económico.

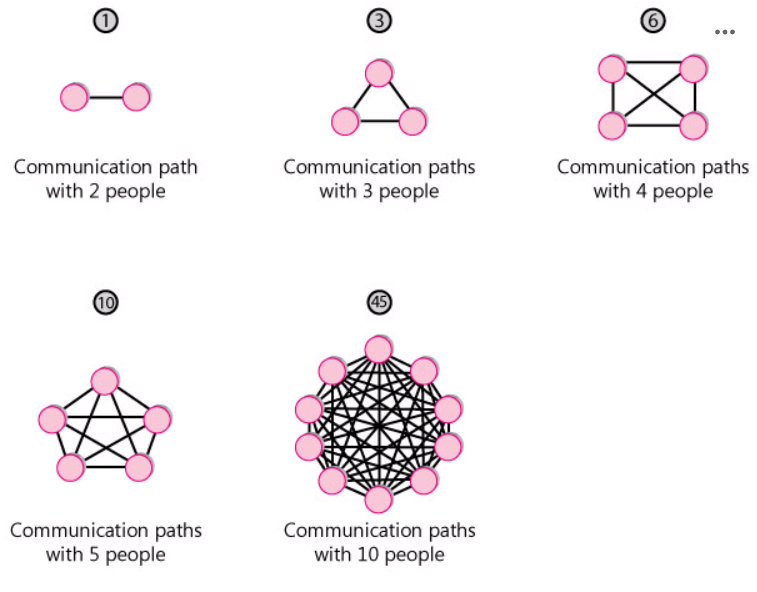
En este tema normalmente hablamos de economías de escala y qué tiene que ver el software con una economía de escala cómo se posiciona el software dentro de una economía de escala?

Claramente el desarrollo de software es una actividad económica viéndola como tal encaja en una economía de escala. En este punto, se han planteado ese interrogante y vamos a ver vamos a de alguna manera a desentrañar este tema de forma tal de clarificarlo.

Las personas asumen naturalmente que un sistema que es 10 veces más grande que otro sistema, este requerirá algo así como 10 veces más de esfuerzo para construirlo.

La cuestión básica es que, en el ámbito del software, los proyectos de mayor envergadura requieren la coordinación entre grupos mas grandes de personas, lo que exige una mayor comunicación. Y, a medida que aumenta el tamaño del proyecto, el número de vías de comunicación entre diferentes personas aumenta como una función cuadrática del número de personas en el proyecto.

**Breve esquema del problema de la comunicación entre equipos.**



Como puede ver en el gráfico, si tenemos un proyecto en el cual participan dos personas tenemos una sola vía de comunicación. Si agregamos una persona, la cantidad de vías o caminos de comunicación entre esas tres personas, como lo vemos en el gráfico, se aumenta linealmente a tres vías, pero al agregar una cuarta persona ya el crecimiento o la cantidad de caminos de comunicación es mayor. Ya tenemos cuatro personas y seis vías de comunicación.

Qué pasa si tenemos cinco personas, la cantidad de vías de comunicación aumenta a diez en el equipo o en el proyecto tenemos 10 personas sabemos que la cantidad de vías de comunicación con muestra el gráfico aumenta a 45 con esto vemos que el crecimiento de la cantidad de vías de comunicación entre personas que participan en un proyecto a medida que crece la cantidad de personas crece esa cantidad de vías de comunicación pero no en forma proporcional el crecimiento de las personas sino que más más aún crece en una manera diríamos exponencial para resumirlo de alguna manera.

La consecuencia de este aumento exponencial de las vías de comunicación (junto con otros factores) es que los proyectos también tienen un aumento exponencial del esfuerzo a medida que aumenta el tamaño del proyecto. Esto se conoce como deseconomía de escala.

Fuera del software, solemos hablar de economías de escala y no de deseconomías de escala. Una economía de escala es algo así como: "Si construimos una planta de fabricación más grande, podremos reducir el costo por unidad que producimos". Una economía de escala implica que cuanto más grande sea, menor será el costo por unidad.

La deseconomía de escala es lo contrario. En el software, cuanto más grande es el sistema, mayor es el costo de cada unidad. Si el software presentara economía de escala, un sistema de 100.000 LOC sería menos de 10 veces más costoso que uno de 10.000 LOC. Pero casi siempre ocurre lo contrario.

**CONCEPTO DE PROCESO**

•IEEE: “Una secuencia de pasos ejecutados para un propósito dado”

•se ajusta a las prácticas específicas de las distintas ingenierías;

•dos componentes: acciones y propósito;

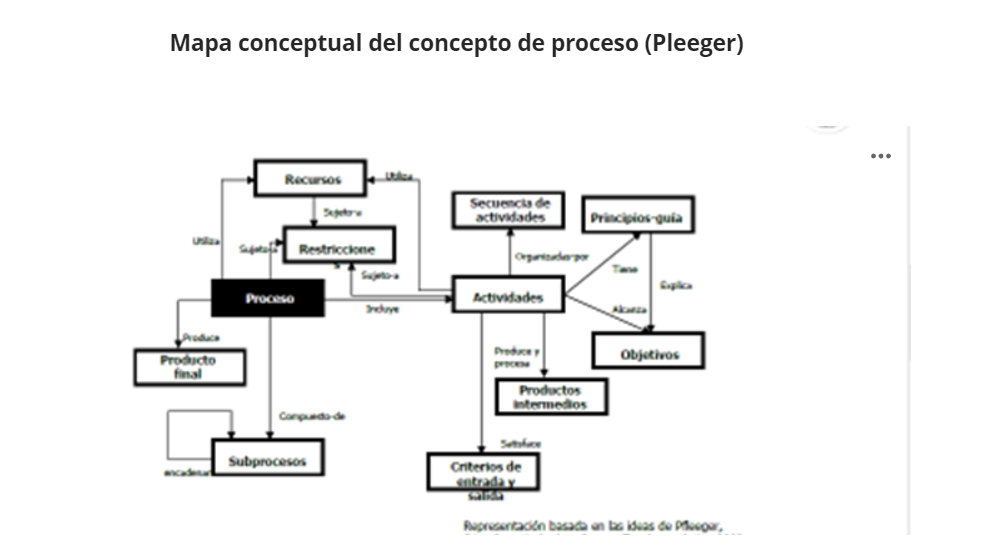
•la meta le da al proceso una serie de cualidades

•Pfleeger: “Podemos pensar al conjunto ordenado de tareas como un proceso: una serie de pasos que involucran actividades, restricciones y recursos que producen una determinada salida esperada”

•Basili: afirma de que los procesos de ingeniería del software son específicos y claramente como hemos visto que el software como producto es único no se aplican los mismos procesos para construir software que para construir cualquier producto.

**El proceso en detalle (Pfleeger)**

* Establece todas las principales actividades que involucra
* Utiliza recursos, está sujeto a una serie de restricciones económicas y técnicas (por ej. los RRHH pueden no estar capacitados para aplicar una cierta tecnología), y genera productos intermedios y finales
* Puede estar compuesto por subprocesos que se encadenan de alguna manera
* Cada actividad del proceso tiene criterios de entrada y salida. Se trata del concepto de poner condiciones a los inputs para ser tratados por una determinada actividad. Es el ejemplo típico que un programa para ser aceptado en testing debe satisfacer los juegos de prueba provistos y el diseño definido.
* Las actividades se organizan en una secuencia
* Tiene una serie de principios-guía que explican los objetivos de cada actividad
* Las restricciones o controles pueden ser de aplicación a una actividad, recurso o producto



Aquí se muestra el proceso que incluye actividades que están organizadas en una secuencia.

Tienen una serie de principios y guías. Esas actividades están diseñadas para alcanzar objetivos y esas actividades pueden generar productos intermedios siempre satisfaciendo una serie de criterios de entrada y salida.

Por otra parte los procesos utilizan recursos y están sujetos a restricciones tanto el proceso como los recursos obviamente que las actividades también utilizan recursos porque las actividades son parte del proceso o el proceso se descompone en actividades que utilizan recursos y esas actividades están sujetas a restricciones.

Por lo tanto si las actividades generan productos intermedios el proceso como un todo produce o genera un producto final y naturalmente como ya dijimos el proceso puede estar compuesto de sus procesos encadenados entre sí.

Esta visión de proceso nos ordena la generación de un producto. en el caso de el software tenemos un proceso que guía el desarrollo del software o que establece el ciclo de vida de desarrollo de software y para desarrollo desarrollar software hay distintos modelos de procesos

Para el desarrollo de la ingeniería de requerimientos va a tener un conjunto de actividades que van a estar organizadas.

Verá:

* la generación de productos intermedios
* criterios de entrada salida las actividades
* objetivos va se van a utilizar recursos
* que las actividades van a estar sujetas a restricciones
* que el proceso de ingeniería de requerimientos se compone de sus procesos
* Que así como hay productos intermedios hay un producto final por excelencia de la ingeniería de requerimientos .

En definitiva vamos a ver que así como utilizamos un proceso para el desarrollo de software vamos a utilizar un proceso un subproceso del desarrollo de software este es un proceso de requerimiento que la vamos a ver como un proceso en sí misma y a su vez va a estar compuesta por sus procesos y va a generar productos intermedios y un gran producto final o sea que esta definición esta conceptualización este mapa conceptual del concepto de proceso no sólo es aplicable en software sino que vamos a aplicarlo a un subproceso del desarrollo de software como lo es la ingeniería de requerimientos.