Ingeniería de Software

Conceptos Introductorios

1. Definición

- Existen diversas definiciones, podemos considerar la del glosario de IEEE denominado IEEE Standard Glossary of Software Engineering Terminology:
  La ingeniería de software es la aplicación de un enfoque sistemático, disciplinario y cuantificable para el desarrollo, operación y mantenimiento del software, es decir, es la aplicación de la ingeniería al software.
- En esencia tiene que ver con los siguientes temas:
  a) El desarrollo de grandes programas
  b) Manejo de la complejidad
  c) El software evoluciona tanto como la realidad donde se encuentra
  d) La eficiencia en el desarrollo del software; relativo a tiempos, costos y mantenimiento
  e) La cooperación regular entre las personas involucradas en un proyecto
  f) El suporte que el software brinda a sus usuarios
  g) Los integrantes de una rama del conocimiento crean artefactos para los integrantes de otra rama del conocimiento
  h) La ingeniería de software actúa como un estabilizador entre requerimientos (probablemente cambiantes) y restricciones que afectan un sistema.
- Existen también unas críticas a esta rama, por ejemplo leer el documento realizado por Dijkstra, pág.11, párrafo 2. (revisar adjunto EWD1036.pdf).

  Ejercicio 1: Elaborar en una media página sus opiniones acerca de lo expresado por Dijkstra, tratar que sean opiniones fundamentadas.

1.1 Crisis del Software

Por qué estudiar Ingeniería de Software? Algunos casos en los cuales la construcción de un software falló:

Therac-25

- Sistema médico con tres componentes:
  a) Field light mode: Ubicación correcta del paciente.
  b) Electron mode: El computador controlaba la cantidad de energía hasta un punto seguro.
  c) Photon (X-ray) mode: En este modo el rayo de energía se mantiene constante. Se tenía la presencia de un “beam flattener”, en algunos casos la cantidad de corriente en un lado se incrementaba en un 100% comparado con el modo electrón.

  Inconveniente: El beam flattener no se encontraba en posición ocasionando heridas a los pacientes.

London Ambulance

- Sistema contaba con un CAD (Computer Aided Dispatch)

- Cambiaba el sistema de despacho de ambulancias de un sistema manual a un sistema automatizado.
- Movilización del recurso más óptimo o el más cercano a una incidencia.

  Inconveniente: Si una ambulancia se encontraba cerca de una segunda incidencia y luego de una tercera, y así sucesivamente, podía alejarse completamente de su base. De este modo los demás recursos eran no utilizados.

Para información adicional puede leer el adjunto Chapter 1-Issues: Software Crisis (0818676094.excerpt.pdf)

Puede revisar el archivo anexo con mayores detalles de los casos dados anteriormente ()

Ejercicio 2: Buscar un par de casos más en los cuales un pobre diseño de software dio consecuencias graves. Referenciar adecuadamente.

2. Análisis de Métodos Tradicionales en Desarrollo de Software

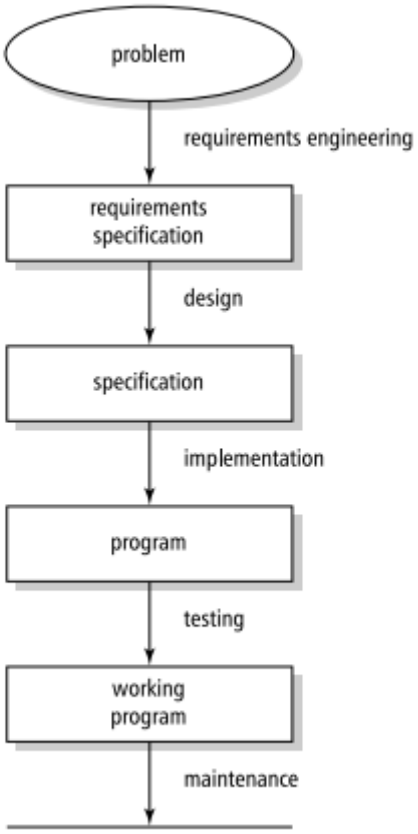


Fig. 1 Etapas en el desarrollo de un software

- Es un modelo genérico, podría guardar similitud con el modelo de la cascada.
- En cada etapa se puede hacer un retroceso a la etapa anterior, esto por errores o inconsistencias encontradas.

Etapas:

a) Ingeniería de Requerimientos: Tener un visión completa del problema a resolver, puede incluir también el denominado *análisis de factibilidad*.

b) Diseño: Se realiza un modelo del sistema utilizando diversas técnicas. Se especifican los *componentes* que son las partes y las *interfases* que son los canales de comunicación entre estas partes.

c) Implementación: Construcción del software, se hace uso de algoritmos y pseudcódigos.

d) Testing: Es la parte de pruebas, está presente en todas las etapas. Se puede dar del tipo *verificación*, donde se ve que la transición entre etapas es la correcta y *validación* por medio de la cual se ve si se está cumpliendo con los requisitos del usuario.

e) Mantenimiento: Es la etapa que se encarga de corregir los errores o adaptar el sistema.

- La distribución porcentual de cada una de estas etapas sería la siguiente:

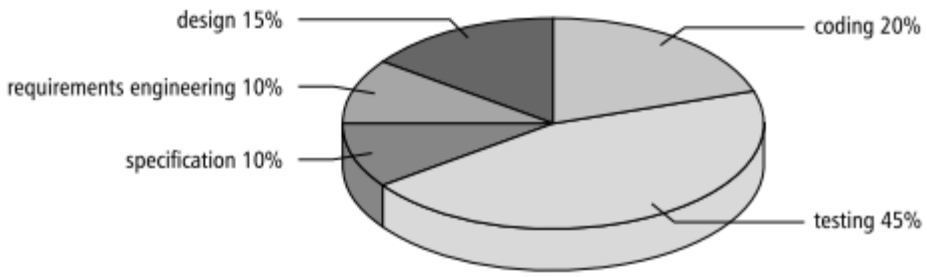


Fig. 2 Distribución porcentual de las distintas actividades en el desarrollo de un software

- Sigue la regla del 40-20-40, es decir solo el 20% se dedica a la codificación.
- Boehm expone que el esfuerzo debiera ser 60-15-25, es decir 60% en requerimientos y diseño, 15% en implementación y 25 % en testing.
- La *administración de un proyecto* se encarga de definir las tareas y controlar cada una de estas partes. Cabe recordar que es necesaria una adecuada documentación.

Tipos de mantenimiento:

a) Correctivo: Reparación de errores actuales

b) Adaptativo: Se adapta el software a situaciones cambiantes, por ejemplo: cambio de versiones.

c) Perfectivo: Se adapta el sistema a requerimientos cambiantes del usuario.

d) Preventivo: Trata de incrementar el mantenimiento futuro, haciendo uso de documentación adecuada, o de la adecuada estructuración de los módulos de un sistema.

El porcentaje de estas actividades se observa en la siguiente figura:

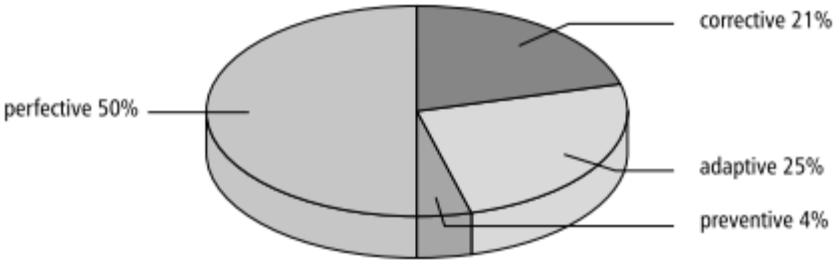


Fig. 3 Distribución actividades de mantenimiento

Ejercicio 3: Analice los casos adjuntos coloque en un informe breve si es que se encuentra adecuadamente documentado. Para el caso 4 trate de resolver preguntas como: Se pudo haber creado un manual de usuario? Qué partes hubiera incluido? Era necesario comentar el código? En base a la documentación del sistema hubiera sido suficiente para hacer cambios profundos en él?

Caso 1:
http://www.acqualia.com/soulver/
Caso 2:

Fixing Read-Only Projects

Scrivener cannot open files that have been set to be read-only, because it always needs write access to a file, even if it is only opening it. (It requires such access so that it can create a "lock file" inside the file, which will allow other copies of Scrivener to know that the project is open; it also requires it for auto-save to work correctly.)

If you try to open a project that has been set to read-only, you may encounter an error message such as, "Scrivener does not support read-only projects. Check the permissions for this project in the Finder, and ensure that it is not stored on a read-only volume such as a disk image."

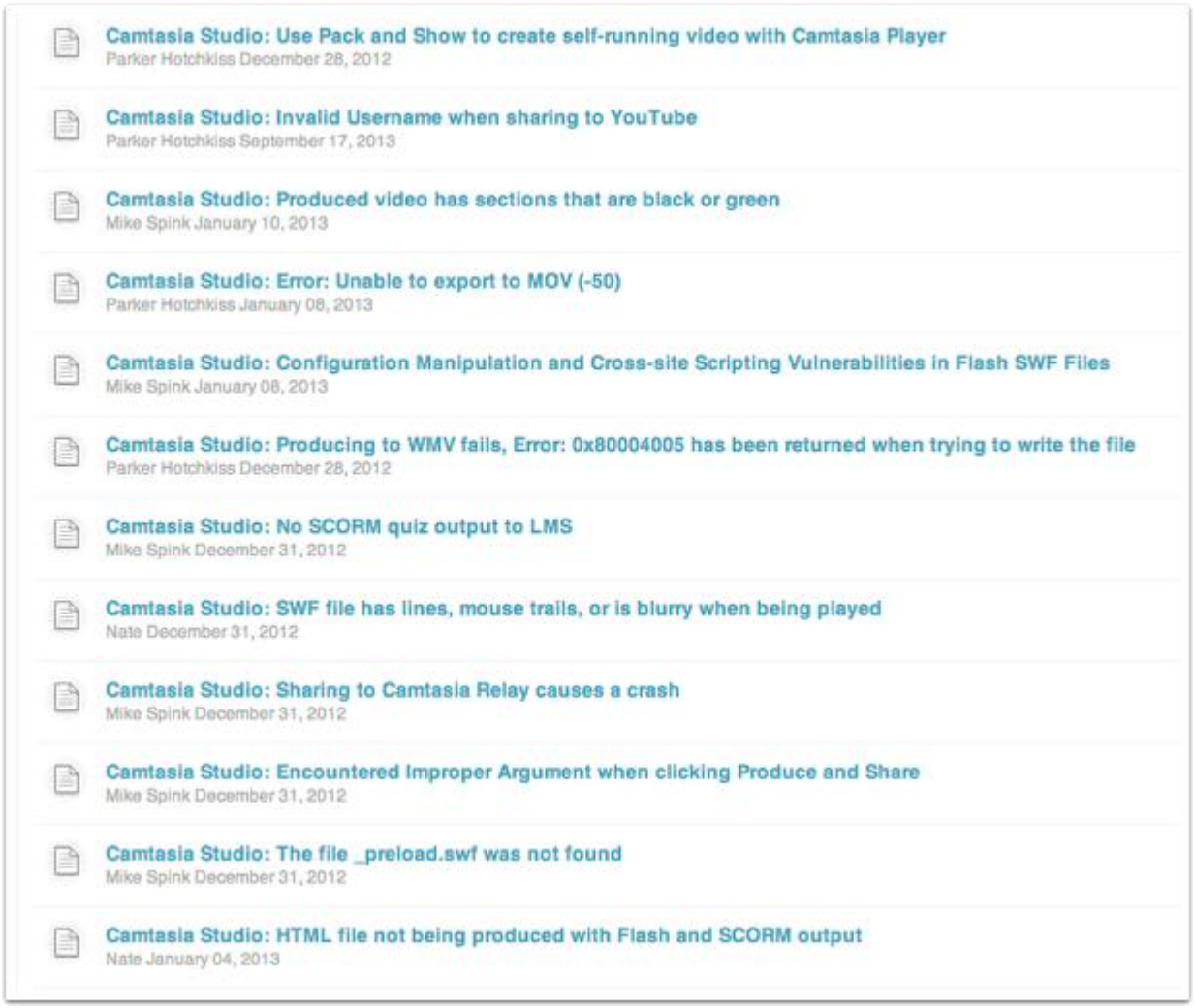
If you encounter this message, the first thing to check is that the project you are trying to open is not saved on a read-only device. For instance, if it has been burned onto a CD, you won't be able to open it because it isn't possible to save to a CD. In this case, you would need to copy the file and paste it somewhere on your hard drive.

Next, you may need to change the permissions on the file itself. A .scriv file is really a folder full of files that just looks like a regular file, so you need to make sure that all of the files inside the .scriv file have the correct permissions, too, which complicates matters. To ensure that the permissions are correct, follow these steps:

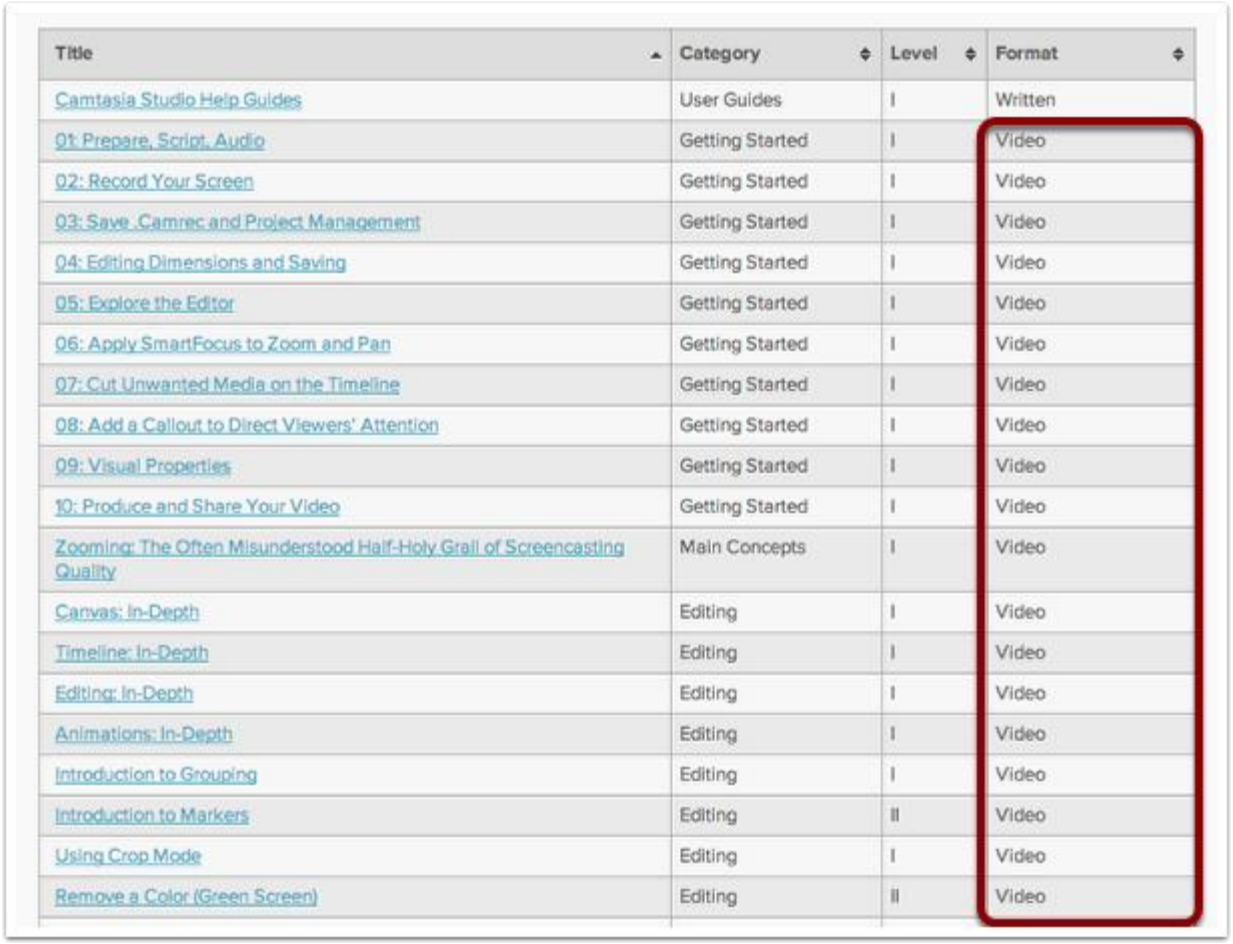
1. Ctrl-click on the .scriv file in the Finder and select "Get Info" from the contextual menu that appears.
2. In the Info panel that appears, delete the ".scriv" file extension from the file name under "Name & Extension" and hit return (e.g. if the file was entitled "Foo.scriv", change it so that it is just entitled "Foo"). A warning panel will ask you if you are sure that you want to remove the extension ".scriv" - confirm the action by clicking on "Remove". At this point, you will notice that the icon of the file changes so that it now looks like a regular folder rather than like a .scriv file.
3. Close the Info panel, and then ctrl-click on the file (which now looks like a folder) in the Finder and select "Get Info" to open the panel again. This step is necessary because if we don't close and reopen the panel, not all of the options we need will be available - the Finder needs to know that the file is a folder for the following steps to work.
4. At the bottom of the Info panel you will find a section entitled "Sharing & Permissions", with a table containing two columns beneath it - "Name" and "Privilege". You need to ensure that the name of your current user account is listed with "Read & Write" next to it. First, click on the icon of the lock at the bottom-right of the panel so that you can make changes - you will be prompted for your administrator password.
5. Once you have entered your administrator password, the buttons at the bottom-left of the Info panel should be available. If your user account name isn't listed in the "Name" column of Sharing & Permissions, click on the "+" button and choose it from the list that appears.
6. If necessary, change the "Privilege" next to your user account name to "Read & Write", by clicking on the row and picking this option from the list that appears.
7. Click on the button at the bottom with the gear icon and the downwards-pointing arrow next to it, and select "Apply to enclosed items...". This will ensure that the "Read & Write" permissions are applied to all files inside the project container.
8. Now add the ".scriv" extension back to the end of the file name under "Name & Extension" (so if you had renamed it to "Foo", now's the time to change it back to "Foo.scriv"), and hit return.
9. Close the Info panel.

The file should now look like a Scrivener project in the Finder again, and it should now have the correct permissions. Scrivener should now be able to open it without any problem.

Caso 3:


Camtasia Studio: Use Pack and Show to create self-running video with Camtasia Player
Parker Hotchkiss December 28, 2012
Camtasia Studio: Invalid Username when sharing to YouTube
Parker Hotchkiss September 17, 2013
Camtasia Studio: Produced video has sections that are black or green
Mike Spink January 10, 2013
Camtasia Studio: Error: Unable to export to MOV (-50)
Parker Hotchkiss January 08, 2013
Camtasia Studio: Configuration Manipulation and Cross-site Scripting Vulnerabilities in Flash SWF Files
Mike Spink January 08, 2013
Camtasia Studio: Producing to WMV fails, Error: 0x80004005 has been returned when trying to write the file
Parker Hotchkiss December 28, 2012
Camtasia Studio: No SCORM quiz output to LMS
Mike Spink December 31, 2012
Camtasia Studio: SWF file has lines, mouse trails, or is blurry when being played
Nate December 31, 2012
Camtasia Studio: Sharing to Camtasia Relay causes a crash
Mike Spink December 31, 2012
Camtasia Studio: Encountered Improper Argument when clicking Produce and Share
Mike Spink December 31, 2012
Camtasia Studio: The file _preload.swf was not found
Mike Spink December 31, 2012
Camtasia Studio: HTML file not being produced with Flash and SCORM output
Nate January 04, 2013

Caso 3:



Title	Category	Level	Format
Camtasia Studio Help Guides	User Guides	I	Written
01: Prepare, Script, Audio	Getting Started	I	Video
02: Record Your Screen	Getting Started	I	Video
03: Save .Camrec and Project Management	Getting Started	I	Video
04: Editing Dimensions and Saving	Getting Started	I	Video
05: Explore the Editor	Getting Started	I	Video
06: Apply SmartFocus to Zoom and Pan	Getting Started	I	Video
07: Cut Unwanted Media on the Timeline	Getting Started	I	Video
08: Add a Callout to Direct Viewers' Attention	Getting Started	I	Video
09: Visual Properties	Getting Started	I	Video
10: Produce and Share Your Video	Getting Started	I	Video
Zooming: The Often Misunderstood Half-Holy Grail of Screencasting Quality	Main Concepts	I	Video
Canvas: In-Depth	Editing	I	Video
Timeline: In-Depth	Editing	I	Video
Editing: In-Depth	Editing	I	Video
Animations: In-Depth	Editing	I	Video
Introduction to Grouping	Editing	I	Video
Introduction to Markers	Editing	II	Video
Using Crop Mode	Editing	I	Video
Remove a Color (Green Screen)	Editing	II	Video

Caso 4:
https://sourceforge.net/projects/java-galib/

3. Orientación a Objetos

Recordar conceptos previos de Programación Orientada a Objetos, estos lo utilizaremos para el modelado.

El adjunto nos puede servir de recordatorio (introduction.pdf obtenido de: http://people.cs.aau.dk/~torp/Teaching/E03/OOP/handouts/introduction.pdf)

Se puede utilizar el UML (Unified Modelling Language) para modelar un sistema basado en objetos. Revisar el adjunto de: http://www.digilife.be/quickreferences/qrc/uml%20quick%20reference%20card.pdf

Detalles serán analizados posteriormente.

Ejercicio 4:

Para cada uno de los siguientes casos realice un esquema de clases, puede utilizar Paint o cualquier editor:

Caso 1:

> Se desea implementar un simulador de un ecosistema de la vida de un bosque cercana a los ríos. Para esto se modelan dos tipos de criaturas: osos y pescados. El ecosistema consiste de un río el cual puede ser modelado como un arreglo unidimensional de tamaño relativamente grande. Cada elemento de

esta lista deberá ser un objeto del tipo Oso, un objeto del tipo Pescado o Ninguno. Al inicio de la simulación en el río se encuentran estos animales en posiciones al azar del mismo. Conforme pasa el tiempo los osos y los peces pueden desplazarse a otras ubicaciones al azar, en caso que la posición esté vacía solamente la ocupan, pero en caso que esté llena pueden pasar dos situaciones:

- Si dos animales de la misma especie están a punto de colisionar por querer ocupar una misma posición, estos permanecerán en sus posiciones originales, pero crearán una nueva especie (reproducción) y este nuevo animal ocupará una posición vacía en el río.
- En caso que un oso y un pescado colisionen en una misma posición del río, el pescado será devorado por el oso.

Modele el sistema anteriormente mencionado y visualice en cada unidad de tiempo cómo va quedando el río. Coloque mensajes que sirvan para poder visualizar cuando es que ocurre una colisión, cuando los animales se han reproducido o cuando un pescado ha desaparecido consumido por un oso. Considere los siguientes datos:

- Al inicio el río deberá estar lleno en un 50% de su capacidad
- La cantidad de osos y peces que ocupen el río serán colocados al azar
- Un oso o pescado que desee moverse a una posición será escogido al azar

- La simulación termina cuando la población de peces es menor al 10% que la población de osos del río.

Caso 2:

Se nos solicita hacer un sistema que permita encriptar y desencriptar mensajes los cuales serán transportados por Internet. El proceso de encriptación consiste en transformar un texto plano en un texto cifrado mediante distintas técnicas. Por ejemplo:
Texto Plano: TREATY IMPOSSIBLE
Texto Cifrado: WUHDWB LPSRVVLEOH
El proceso de desencriptación sería transformar el texto cifrado a texto plano nuevamente.
Usted puede considerar que existe una clase denominada MENSAJE que almacena un texto plano definido por el usuario y que cuenta con dos métodos, uno para encriptar y otro para desencriptar el mensaje.
Los tipos de encriptación, cada uno a estar desarrollado en una clase distinta serán los siguientes:
- Cifrado César:
En este cifrado las letras tienen un equivalente numérico cómo se puede ver en la Fig. 1:

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Fig. 1. Cada letra del alfabeto tiene su correspondiente valor numérico. A la letra A le corresponde el 0, la letra B el 1 y así de forma sucesiva. Se consideran las 26 letras del alfabeto inglés.

El cifrado César consiste en desplazar cada letra a la derecha del alfabeto dado en la Fig. 1 por un valor de n posiciones definidas por el usuario. Por ejemplo:
Texto plano=hola
n=3
Texto cifrado:
h: 7+3=10→k
o: 14+3=17→r
l:11+3=14→o

a:0+3=3→d

El texto cifrado será: krod

Un caso especial se da cuando por el valor del desplazamiento nos encontramos en una posición fuera del alfabeto, por ejemplo si quiero encriptar lo siguiente:
texto plano: yz
n=5
y: 24+5=29-26=3→d
z: 25+5=30-26=4→e

El texto cifrado será: de
Para el proceso de desencriptación en vez de sumar se restará el valor del dfesplazamiento.

- Rail fence:
Este tipo de cifrado lo que hace es colocar un texto en diagonales dependiendo del número de filas que no desee ingresar, por ejemplo:
Texto Plano: this system is secure
número de filas: 3

El resultado será:

```
t       s       s       m       s       u
  h       s       t       i       e       r
    i       y       e       s       c       e
```

y al leerlo fila por fila tendríamos:
Texto cifrado: tssmsuhstieriyesce
Para el descifrado sería el proceso inverso, en vez de leer por filas se leería los datos columna por columna.

- Caesar on rails
Este cifrado es una mezcla del cifrado César y del Rail fence, en este tipo de encriptación se pedirán dos datos; un dato será el desplazamiento de las letras en el alfabeto y el otro el número de filas donde se colocará el mensaje. El proceso de desencriptamiento es la inversa de los dos procesos anteriormente descritos.

- Cifrado Vernam:
En este cifrado se escoge un alfabeto como el de la fig. 1, pero al mismo tiempo se escoge una palabra que no contenga caracteres repetidos. Se suma ambos caracteres y se ve a qué nueva letra corresponden según el alfabeto dado. Por ejemplo:

Texto plano:helloworld
clave: wasp

```
H  E  L  L  O  W  O  R  L  D
7  4  11 11 14 22 14 17 11 3        (posición en el alfabeto de la Fig. 1)
```

W	A	S	P	W	A	S	P	W	A
22	0	18	15	22	0	18	15	22	0
29	4	29	26	36	22	32	32	33	3
3	4	3	0	10	22	6	6	7	3
D	E	D	A	K	W	G	G	H	D

(se suman ambos valores y se reduden los que pasan el valor de 26)

Para los textos ingresados el programa deberá eliminar primeramente signos de puntuación y espacios en blanco.

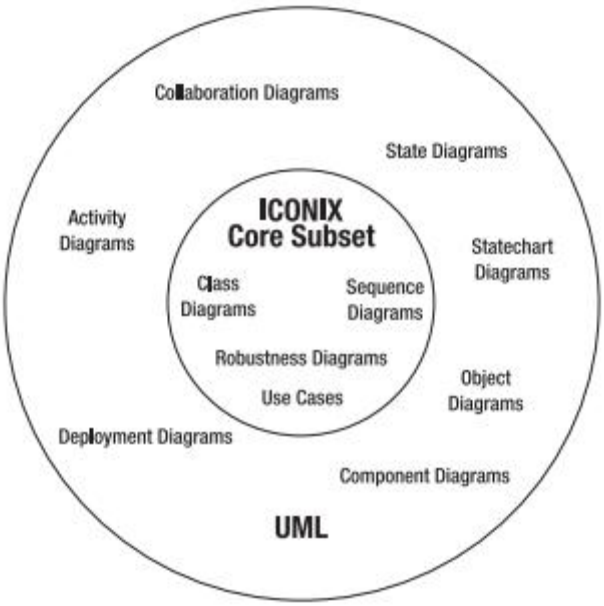
4. Métodos de Análisis Orientado a Objetos

Existen diversos métodos de Análisis Orientados a Objetos entre estos tenemos:

a) Object Oriented Analysis and Design (OOAD)

b) Rational Unified Process (RUP)

c) Iconix



(otros más que se puedan averiguar)

5. Fuentes en temas de Ingeniería de Software

Existen diversas fuentes de donde conseguir información en temas relacionados de Ingeniería de Software, por ejemplo:

http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=32

https://www.sigsoft.org/SEN/

http://tosem.acm.org/

https://www.journals.elsevier.com/journal-of-systems-and-software/

Ejercicio 5:

En grupos de dos descargarse un par de artículos de las anteriores referencias, darle una leída breve a los artículos y responda las siguientes preguntas (se recomienda distribuirse el trabajo y discutir entre ustedes sobre el artículo):

a) De qué trata el artículo de forma general? (Se recomienda leer el abstract)

b) Qué conclusiones se pueden desprender del artículo?

c) Qué relación tiene el artículo escogido con la rama de Ingeniería de Software?

d) Describa qué metodologías son familiares para usted según el texto leído.

e) Describa qué metodologías no son conocidas por ustedes.

Presentar la respuestas a estas preguntas en un breve informe, incluya los artículos escogidos.

6. Introducción a la Administración de Proyectos de Software

- Todo proyecto tiene que ser cuidadosamente planeado.
- El *plan de proyecto* es el documento que nos brinda un escenario entre los clientes y el equipo encargado del proyecto.
- Se deben de considerar los siguientes componentes para tener un control del proyecto: Tiempo, como evaluar el progreso del proyecto? Información, el cual se refiere al manejo de la documentación. Organización, como se distribuye las labores entre el equipo del proyecto. Calidad, cómo evaluar los requerimientos de calidad del proyecto? Costo del proyecto.

7. Introducción a los Requerimientos

- Un requerimiento es una condición o capacidad que necesita un usuario a fin de resolver un problema o de alcanzar un objetivo (definición dada por el IEEE 610)
- Se puede dar un trade-off entre los requisitos solicitados en caso que se encuentren conflictos entre los stakeholders.
- La documentación que se obtiene al terminar la fase de la ingeniería de requerimientos se denomina *especificación de requerimientos (requirement specification)*.
- Este proceso cuenta con las siguientes etapas:
  a) Elicitación de requerimientos, se trata del entendimiento del problema; esto puede ser hecho al hablar con especialistas en el tema a resolver.
  b) Especificación de requerimientos, después de entender el problema se pasa a describirlo. Se puede hacer uso de lenguaje natural o lenguaje formal; del mismo modo se pueden utilizar técnicas de modelado orientado a objetos.
  c) Validación y verificación de requerimientos: Validación se refiere a que los requerimientos correctos son establecidos, mientras que la verificación se encarga de determinar que estos requerimientos son definidos correctamente.

Verificación: Se ha construido el software que el usuario quería?
Validación: Lo que se ha construido da el resultado esperado?
Ejemplo: Editor de texto solicitado por un usuario.
Se supone que el menú de usuario tenga las opciones de edición y formateo (verificación)
Si es que el usuario selecciona la opción de edición, le permite modificar el texto (validación)
d) Negociación de requerimientos, esto puede darse en caso de conflictos o restricciones entre los distintos stakeholders.

7.1 Requerimientos Funcionales y no funcionales

- Los requerimientos funcionales son aquellos que describen la interacción entre un sistema, el cual puede ser un usuario u otro sistema, y el ambiente. Estos son independientes de su implementación. Por ejemplo:

`SatWatch` is a wrist watch that displays the time based on its current location. `SatWatch` uses GPS satellites (Global Positioning System) to determine its location and internal data structures to convert this location into a time zone.

The information stored in `SatWatch` and its accuracy measuring time is such that the watch owner never needs to reset the time. `SatWatch` adjusts the time and date displayed as the watch owner crosses time zones and political boundaries. For this reason, `SatWatch` has no buttons or controls available to the user.

`SatWatch` determines its location using GPS satellites and, as such, suffers from the same limitations as all other GPS devices (e.g., inability to determine location at certain times of the day in mountainous regions). During blackout periods, `SatWatch` assumes that it does not cross a time zone or a political boundary. `SatWatch` corrects its time zone as soon as a blackout period ends.

`SatWatch` has a two-line display showing, on the top line, the time (hour, minute, second, time zone) and on the bottom line, the date (day, date, month, year). The display technology used is such that the watch owner can see the time and date even under poor light conditions.

When political boundaries change, the watch owner may upgrade the software of the watch using the WebifyWatch device (provided with the watch) and a personal computer connected to the Internet.

- Los requerimientos no-funcionales son aquellos no relacionados con el comportamiento del sistema. Por ejemplo: Rendimiento, Usabilidad, Disponibilidad, Soporte, etc.
- A veces los no-funcionales son llamados *requisitos de calidad.* Ejemplo:

Quality requirements for SatWatch

- Any user who knows how to read a digital watch and understands international time zone abbreviations should be able to use SatWatch without the user manual. [Usability requirement]
- As the SatWatch has no buttons, no software faults requiring the resetting of the watch should occur. [Reliability requirement]
- SatWatch should display the correct time zone within 5 minutes of the end of a GPS blackout period. [Performance requirement]
- SatWatch should measure time within 1/100th second over 5 years. [Performance requirement]
- SatWatch should display time correctly in all 24 time zones. [Performance requirement]
- SatWatch should accept upgrades to its onboard via the Webify Watch serial interface. [Supportability requirement]

Constraints for SatWatch

- All related software associated with SatWatch, including the onboard software, will be written using Java, to comply with current company policy. [Implementation requirement]
- SatWatch complies with the physical, electrical, and software interfaces defined by WebifyWatch API 2.0. [Interface requirement]

- Forman la base de las tácticas de arquitectura de software que definen los patrones de arquitectura.

Ejercicio 6:

Identifique 5 requerimientos funcionales y 5 no-funcionales del sistema de ventas de Amazon.

8. Recomendaciones IEEE 830

Ejercicio 7:

Buscar el estándar IEEE 830, averiguar en qué consiste y ver si esta versión es actualizada. Describa 5 diferencias entre esta versión y una actualizada (si es que existe). Igual que en los casos anteriores trabaje en parejas.

9. Lenguaje de Modelado UML

- Es una forma de modelar el diseño de un sistema.
- Está basado en un conjunto de figuras estándares a fin de representar un modelado orientado a objetos.
- En este punto veremos algunas gráficas referentes al diagramado de clases, posteriormente se harán otros tipos de grafos conforme se vaya avanzando con el curso.
- Puede revisar el adjunto uml_intro.pdf donde se hace una breve descripción del uso de este lenguaje de modelado para la gráfica de clases.
  Ejemplo diferencia agregación y composición:
  Agregación:

```
class B(object): pass

class A(object):
   def __init__(self, b):
      self.b = b

b = B()
```

```
a = A(b)
```

Composición:

```
class A(object):
   def __init__(self):
      self.b = B()
```

Ejercicio 8:

En grupos de dos personas y haciendo uso de la nomenclatura en UML modelen el siguiente caso:
Una biblioteca cuenta con libros, revistas y medios digitales (CDs, DVDs, Blu-Ray) para su préstamo a los distintos usuarios. Cada uno de estos medios puede tener uno o más autores y cada uno cuenta con diversas características; por ejemplo un libro posee un ISBN, mientras que un DVD sólo contiene un código de producto. Una persona si desea prestarse o reservar un material de la biblioteca debe de tener una cuenta; en esta cuenta se registran sus datos principales así como su estado el cual puede ser activo o bloqueado. La biblioteca también cuenta con un catálogo el cual ayuda a los usuarios a fin de que pueda buscar sus artículos con mayor facilidad.

10. Referencias (sin un formato determinado)

[1] Hans van Vliet. Software Engineering: Principles and Practice, 3rd Edition, Wiley, 2007.

[2] Erick J. Braude, Michael E. Bernstein. Software Engineering: Modern Approaches, 2nd Edition, Wiley 2010.

[3] Bernd Bruegge and Allen H. Dutoit. 2009. Object-Oriented Software Engineering Using Uml, Patterns, and Java (3rd ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.

[4] Mark Collins-Cope, Doug Rosenberg, and Matt Stephens. 2005. Agile Development with ICONIX Process: People, Process, and Pragmatism. Apress, Berkely, CA, USA.

[5] Boehm, B. (1987b). Industrial Software Metrics Top 10 List. IEEE Software, 4(5):84--85.