Ingeniería del Software

Unidad 1 – Introducción a la Ingeniería en Software v4.6

Cuerpo Docente, ¡Nos presentamos!

Ing. Oscar Schivo

Ing. Pablo Reitano

Ing. Diana Martinez

Introducción

La clase – reglas de convivencia

Responsabilidad de la clase

Reglas de convivencia

- La clases se harán en forma HIBRIDA
- Todos deberán estar en MUTE. Si hay dudas usar el chat y cualquier docente leerá la pregunta.
- Usar la cámara para generar más cercanía con los docentes y compañeros.
- Cada alumno es responsable de estar presente en la clase.
- Deberan todos estar logueados con el usuario de la facultad xxxxx@frba.utn.edu.ar
- La clase NO puede ser grabada.

La clase - formato

- Días y horarios de cursada
- Formato de cursada

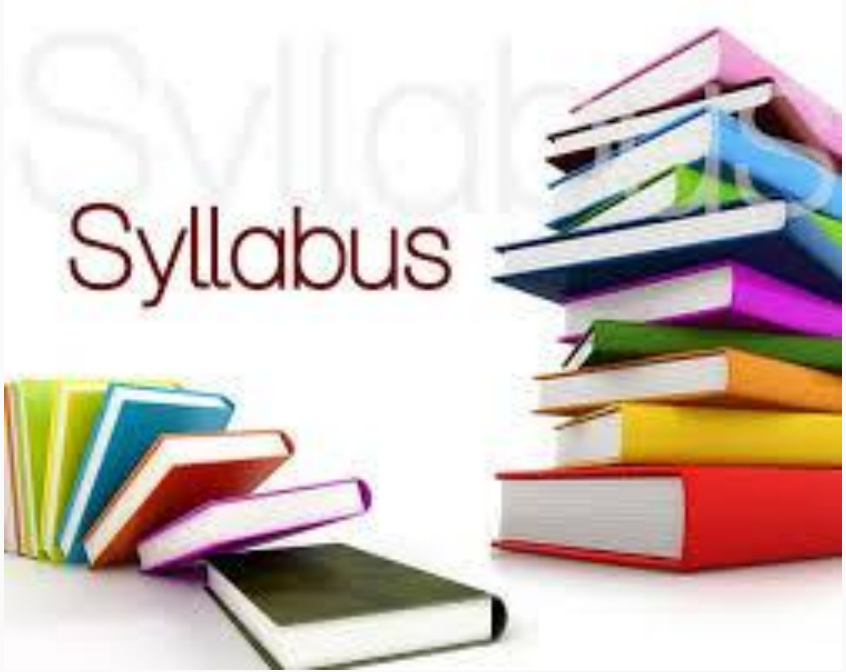
La Materia

Canales de comunicación

slack

- oschivo@frba.utn.edu.ar
- preitano@frba.utn.edu.ar
- dmartinezz@frba.utn.edu.ar
- https://join.slack.com/t/ingsw-2024-1c/shared_invite/zt-2f1ncdhua-8fTnc8j1xbgU21CloGUbWw

Contenidos



1. Ingeniería del Software
2. Calidad del Software
3. Software Engineering Approaches
4. Software Estimations
5. Software Quality & Testing
6. Software Configuration Management
7. Métricas (aplica a todos los ítems anteriores)

Bibliografía

Obligatoria: para cada uno de los temas en particular se da material específico (pueden ser capítulos de libros, papers, publicaciones, etc) en donde profundizar los conceptos vistos en clase. Son de carácter obligatorio.

Recomendada (Rec): si bien no es de lectura obligatoria, complementa los temas vistos en la materia y es recommendable leerla.

Básica (Ref): son libros que soportan el contenido de la materia en general. Sirven de guía o referencia continua.

Ingeniería De Software

Algunas definiciones …

IEEE

La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software.

SEI (Software Engineering Institute)

***Ingeniería** es la aplicación sistemática del conocimiento científico en la creación y construcción de soluciones (“**cost-effective**”) **para resolver** problemas prácticos al servicio del hombre.*

***Ingeniería del SW** es aquella parte de la ingeniería que aplica los principios de las ciencias de la computación y las matemática para alcanzar soluciones (“**cost-effective**”) **a** problemas de software.*

Body of Knowledge (BoK)

Un “cuerpo de conocimiento” es un requisito para calificar a cualquier área de la ingeniería como una profesión y describe el conocimiento relevante para una disciplina.

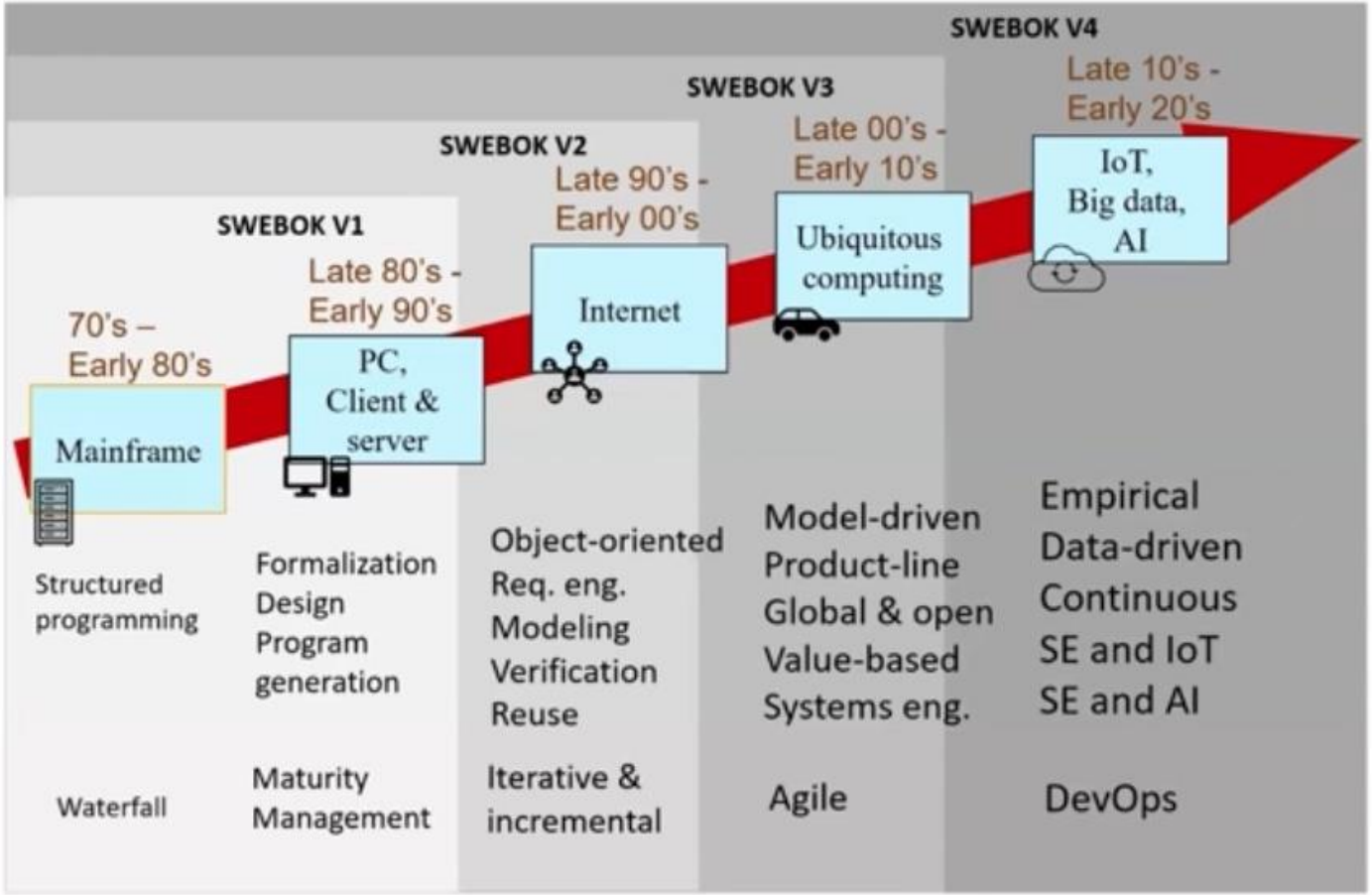
SWEBoK (Software Engineering Body of Knowledge)

- *Describe el conocimiento que existe de la Ingeniería de SW.*
- *Comenzó 1998, liderado por la IEEE Computer Society, para “convertir a la ingeniería del software en una disciplina legítima y una profesión reconocida”.*
- *Está confomardo por 18 Áreas de Conocimiento (Knowledge Areas – KA) que sintetizan conceptos básicos y referencian información más detallada de cada tema.*

- *La versión actual es SWEBoK 4.0 (publicada en el 2024).*
- *Tiene reconocimiento internacional como ISO/IEC Technical Report 19759:2015.*

SWEBOK - Propósito

- Define el contenido de la Ingeniería de Software como disciplina
- Promueve una comprensión a nivel global, consistente de la Ingenienría de Software
- Clarifica los límites entre de la Ingeniría de Software en relación con otras disciplinas
- Proporciona los fundamentos para material de entrenamiento y desarrollo de una currícula
- Puede soportar certificaciones y licencias de Ingeniería en Software
- Refleja el estado de desarrollo actual e integra prácticas emergente


SWEBOK V4
SWEBOK V3
SWEBOK V2
SWEBOK V1
70's – Early 80's
Late 80's - Early 90's
Late 90's - Early 00's
Late 00's - Early 10's
Late 10's - Early 20's
Mainframe
PC, Client & server
Internet
Ubiquitous computing
IoT, Big data, AI
Structured programming
Formalization
Design
Program generation
Object-oriented
Req. eng.
Modeling
Verification
Reuse
Model-driven
Product-line
Global & open
Value-based
Systems eng.
Empirical
Data-driven
Continuous
SE and IoT
SE and AI
Waterfall
Maturity Management
Iterative & incremental
Agile
DevOps

SWEBOK v4 – 18 KAs

SWEBOK V4
Introduction
1. Software Requirements
2. Software Architecture
3. Software Design
4. Software Construction
5. Software Testing
6. Software Engineering Operations
7. Software Maintenance
8. Software Configuration Management
9. Software Engineering Management
10. Software Engineering Process
11. Software Engineering Models and Methods
12. Software Quality
13. Software Security
14. Software Engineering Professional Practice
15. Software Engineering Economics
16. Computing Foundations
17. Mathematical Foundations
18. Engineering Foundations
Appendix A. Knowledge Area Specifications
Appendix B. Standards

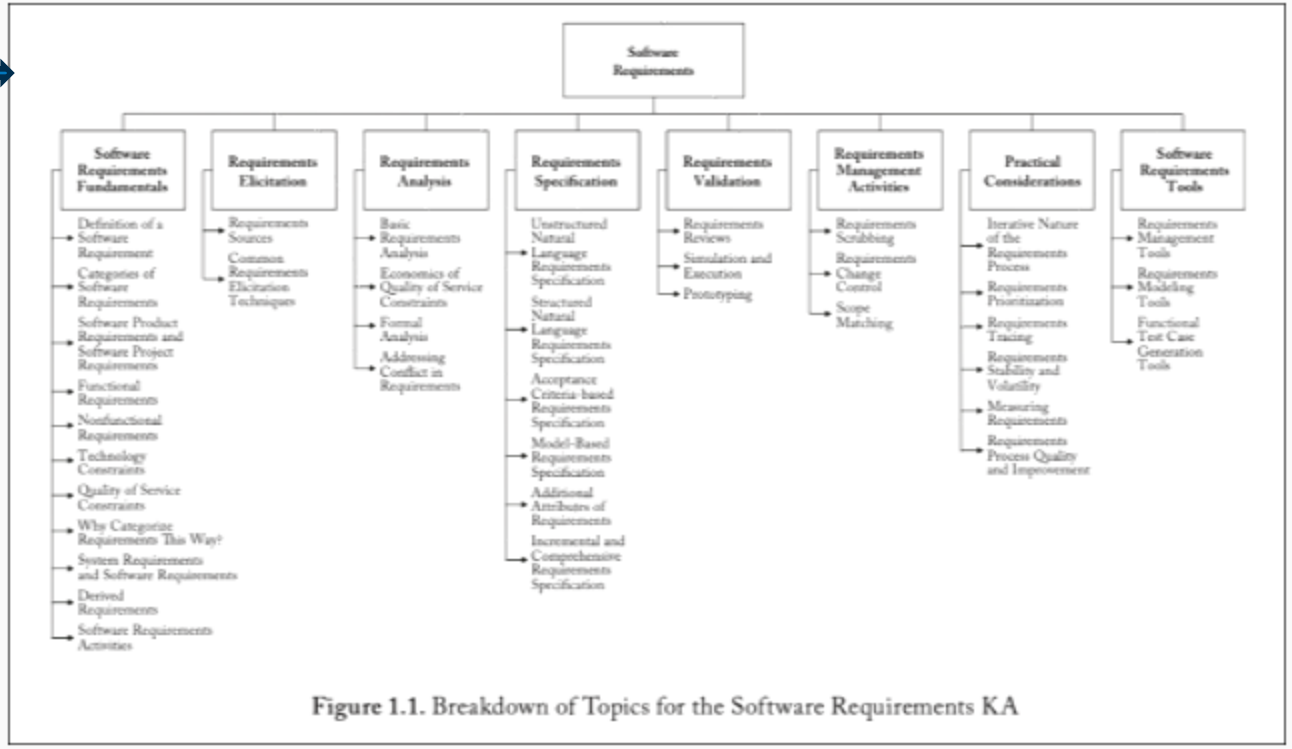


Figure 1.1. Breakdown of Topics for the Software Requirements KA

Problemas habituales en el D&M de Software

¿Cuáles son los problemas habituales en el D&M de SW?

Los proyectos se terminan con mucho atraso

A veces ni se terminan ...

- *Se congelan / suspenden / postergan*
- *Se postergan*
- *Quedan ahí ...mueren...*
- *Aparición del “Proyecto Fase II” (incompletitud) o “Reingeniería de ...” (deuda técnica)*

Los proyectos se exceden en el costo estimado

El producto final del proyecto no cumple con las expectativas del cliente

No hace lo que se supone debería hacer

El producto final no cumple con los reqs. de calidad mínimos esperados

Además de lo funcional, temas relacionados con performance, seguridad, escalabilidad, etc....

¿Cuáles son las causas comunes?

No hay administración y control de proyectos

- *No se puede controlar lo que no se mide*

El éxito depende de los gurúes

Se confunde la solución con los requerimientos

- *La tecnología no es la solución en si misma*

Las estimaciones son empíricas y no hay historia sobre proyectos realizados

La calidad es una noción netamente subjetiva que no se puede medir

No se consideran los riesgos

Se asumen compromisos imposibles de cumplir

Las actividades de mantenimiento van degradando el software

Se comienzan los proyectos con requerimientos no claros ni estables

- *Existe una tendencia a ser optimista (simplificando, subestimando, ...)*

Y la lista sigue ....

Y las reacciones más frecuentes …

Buscando al gurú que nos saque del problema

- *A veces funciona .....*

Agregando gente a un proyecto que está atrasado

Recortando / eliminando cualquier actividad que genere documentación

Recortando / eliminando la etapa de Testing

Recortando / eliminando cualquier actividad que no sea codificar

Asumiendo definiciones en lugar del usuario

Recurriendo a una "bala de plata"

- *Un nuevo paradigma / Una nueva herramienta / Un nuevo lenguaje*

Justificando con frases típicas como:

- *" ... Administrar SW es diferente ..."*
- *"... En definitiva, el usuario no sabe lo que quiere ..."*
- *"... Lo que ocurre que la tecnología tiene que madurar ..."*
- *"... En realidad ya casi está, está en un 90% ..."*

.....

Análisis de la Situación

¿ Por qué seguimos padeciendo los mismos problemas y cometiendo los mismos errores?
¿ Será porque ...

- *... nos cuesta reconocer que tenemos un problema ?*
  Se piensa que después de haber tenido (relativo) éxito con algunos proyectos, se puede en consecuencia llevar adelante cualquier proyecto
- *... no compartimos conocimientos ?*
- *... rechazamos la complejidad ?*
- *... no admitimos que tenemos que aprender ?*
  No se puede conocer todo !!!!
- *... no nos gusta que nos “critiquen” (constructivamente) ?*
  Relación Proyecto SW <-> Persona
- *... menospreciamos a las tareas no técnicas ?*
  Solo se respeta a aquellos que conocen mucho de Tecnología ??
- *... pensamos que bajo presión existen soluciones mágicas que por el solo hecho de presionar hace que las cosas sucedan ?*
- *... asumimos que producir SW es simplemente así ... ?*

Diagrama Causa-Efecto

¿Qué es? ¿Para que se usa?

Es la representación de varios elementos (causas) de un sistema que pueden contribuir a un problema (efecto)

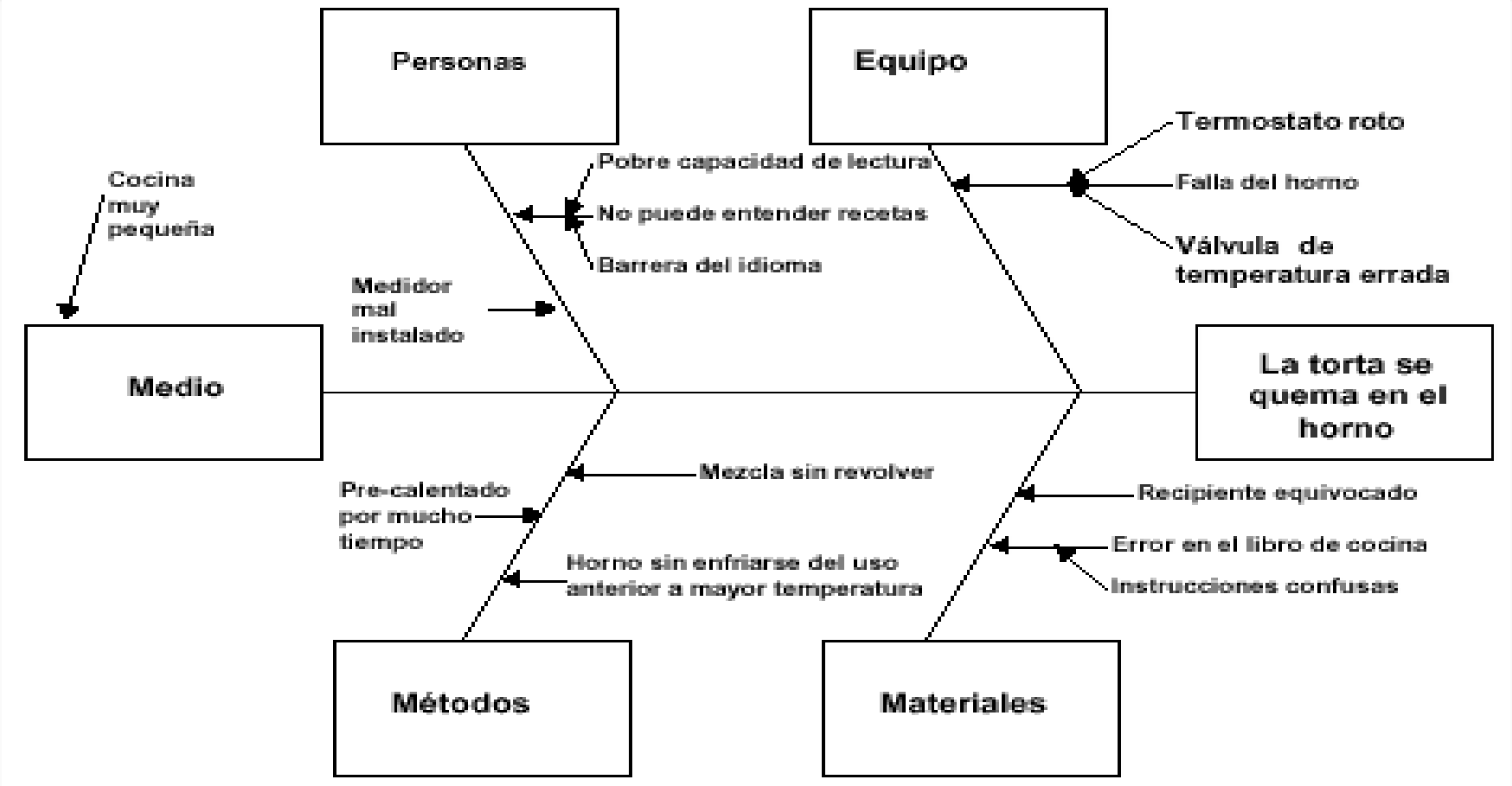
Es útil para la recolección de datos y es efectivo para estudiar los procesos y situaciones

- *Es utilizado para identificar las posibles causas de un problema específico. La naturaleza gráfica del diagrama permite que los grupos organicen grandes cantidades de información sobre el problema y determinar exactamente las posibles causas. Aumenta la probabilidad de identificar las causas principales*

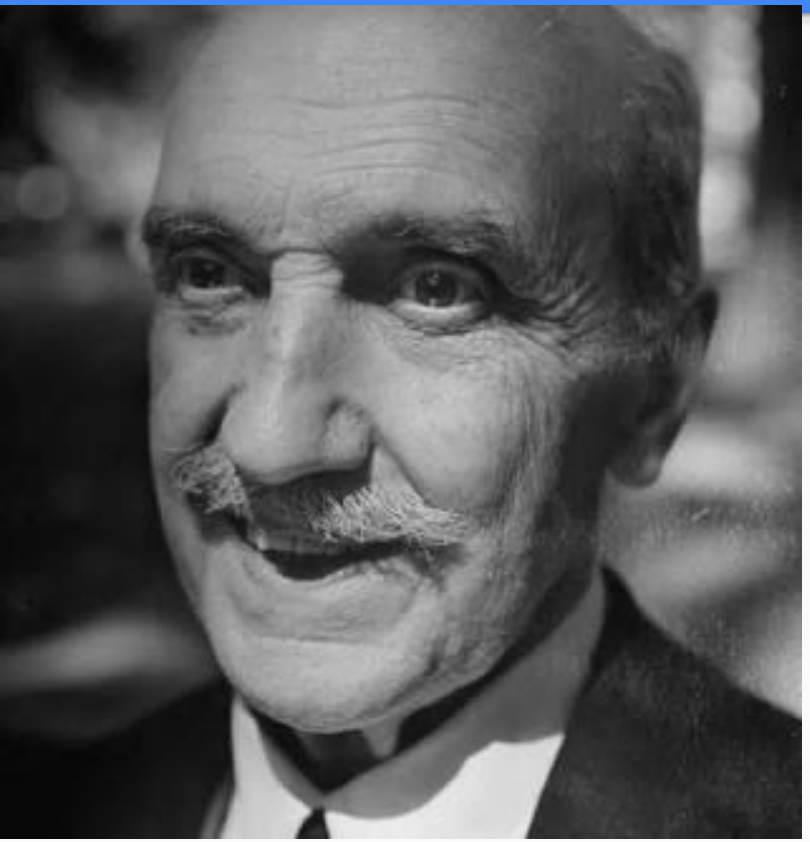
Un diagrama causa y efecto bien preparado es un vehículo para ayudar a los equipos de mejora contínua a tener una concepción común de un problema complejo

También conocido por Diag. De Ishikawa o Espina de Pescado

¿Qué es? ¿Para qué se usa?



Para ir cerrando….



“Quien no conoce su historia está condenado a repetirla”

Pero…a veces es bueno repetirla:

* Si no recuerdo los errores….probablemente los repita
* Si no recuerdo los aciertos…probablemente no los repita

Concepto en IS-> lecciones aprendidas, retro

Tool /ayuda IS-> Classic Mistakes

Jorge Santayana: Filósofo, ensayista, novelista español-americano

¿Preguntas?
¿Dudas?¿Com
entarios?

¡GRACIAS!

Bibliografía

Obligatoria:

- Software Development Classic Mistakes 2008, Steve Mc Connell (Construx)

Recomendada:

- Efectividad de las comunicaciones: http://www.agilemodeling.com/essays/communication.htm

- The Mythical Man-Month Capítulo 1: The Tar Pit, Frederick P. Brooks, Jr

Referencia:

- SWEBOK v4 (link en Campus)