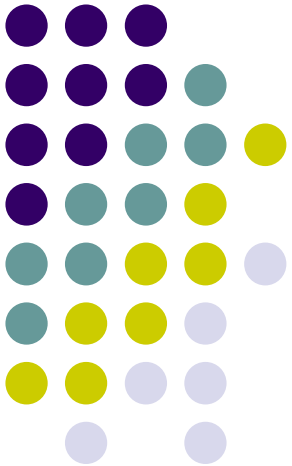
Reglas de Disciplina de Ingeniería de Software

Edgar Sarmiento Calisaya

Departamento Académico de Ingeniería de Sistemas e Informática

UNSA

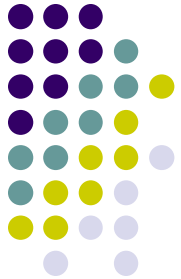


Basado en

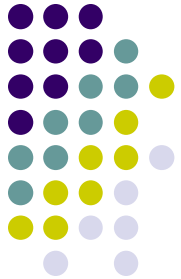
- Leite, J.C.S.P. Notas de Aula. Principios de Engenharia de Software. Puc-Rio. Brasil.

Objetivo

- Entender los principios básicos de la ingeniería de software.

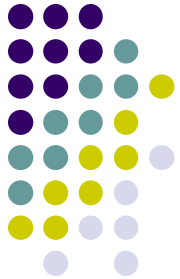


Introducción



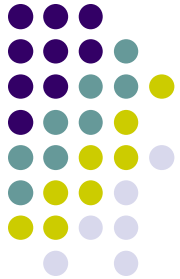
- **Disciplina** es fundamental para que un profesional tenga un *buen desempeño* en la ejecución de sus tareas.
- El ingeniero de software tiene la posibilidad y la libertad de ser creativo, sin embargo:
  - Hasta que punto va la creatividad del ingeniero de software?
  - Hasta que punto su trabajo es seguir **reglas**/normas?
    - ***Disciplina***.
- La ingeniería de software es una disciplina que proporciona los medios (métodos, técnicas, herramientas) para que el profesional sea disciplinado.

Introducción



- La ingeniería de software ayuda a los profesionales en la **definición**, **diseño** y **implantación** de artefactos de software.
- Artefactos de software son complejos:
  - Sistemas de Bibliotecas x Sistemas Operativos.
- Uno de los medios fundamentales para construir ***Artefactos de Software de Calidad*** es la disciplina.
  - **Reglas** de Disciplina (**6**);
  - Cada **regla** transmite una **idea simple**;
  - **Reglas** requieren **conocimientos** asociados.

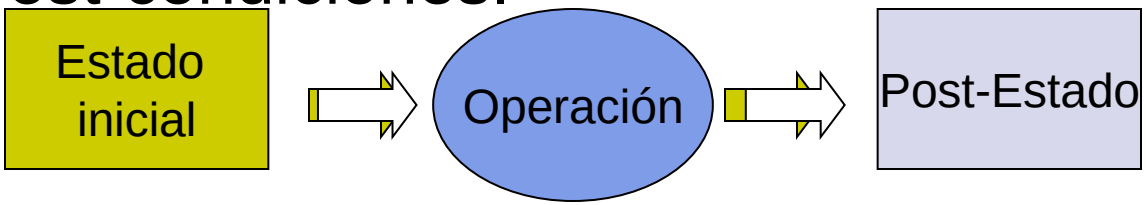
Regla 1



- Todo documento (artefacto de software) debe ser descrito con las siguientes informaciones:
  - a) título,
  - b) autoria,
  - c) fecha,
  - d) **versión**, y
  - e) Indicador de contenido (tamaño).
- Sistema de control de versiones.

Regla 2

- Todo documento debe ser descrito:
  - Pre-condiciones.
  - Post-condiciones.



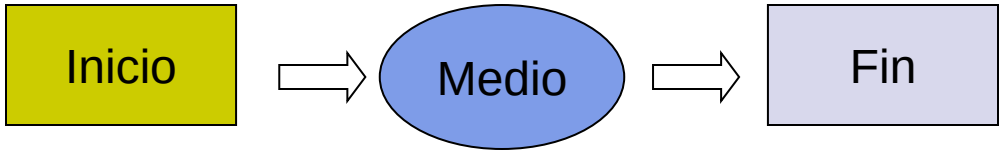
lenguajes de Especificación de Requisitos:

- Escenarios o Casos de Uso:
  - Pre-condiciones.
  - Post-condiciones.

```
void write_sqrt(double x) {
  //   Precondition:  x  >=  0.
  //   Postcondition:  The square
root of x has
  //   been written to the
standard output.
 ...
}
```

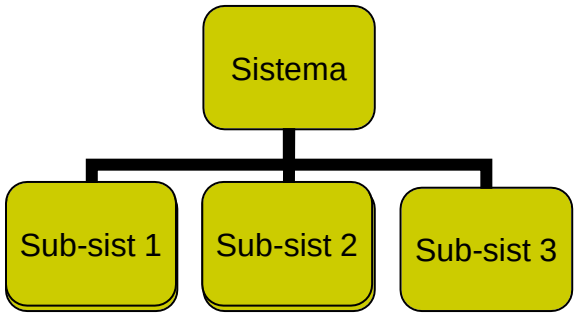
Regla 3

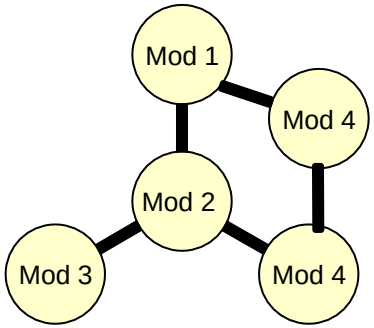
- Cuando **descomponemos/dividimos** algo debemos hacerlo de tal modo que la división resulte en **mínimo 3 partes y máximo 6 partes;**
- Limite mínimo: **[3,6]**



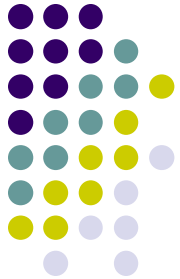
- Limite máximo: Número mágico **7+-2**
  - número máximo de posibles interconexiones entre partes;

- Arquitectura de Software.





Regla 4



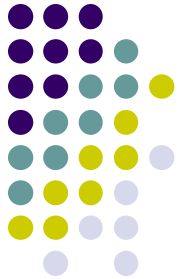
- **Evite inventar nombres**.
  - Contexto.
- Debemos utilizar nombres que tengan su semántica compartida e que sean lo **más** simples posibles;
  - Utilizar el lenguaje del dominio de aplicación.
    - Ontologías;
    - Léxico.

- Internet namespace

```
String login;
String password;
Boolean registrarUsuario(String login, String password) {
 .... }
```

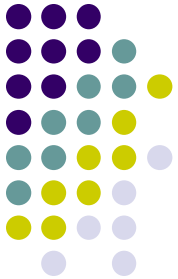
```
String A;
String B;
Boolean C(String A, String B) {
 .... }
```

Regla 5



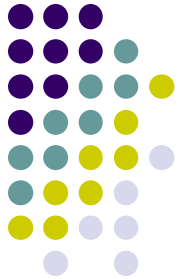
- Una solución simple es mejor que una compleja, mejor aun si la solución parece estúpida.
  - Trate de trabajar con contenidos de tamaño pequeño.
    - "**Small is beautiful**"
  - Trate de crear diseños que sean lo mas limpios posibles.
    - "**Clean Design**"
  - ***Busque soluciones simples.***
    - "**Simple Solutions**"
- Es la regla **más vaga**, deja a la **creatividad**.
- El **principio KISS** (*Keep It Simple, Stupid!*
  - Ex. Use simple brute-force solutions instead of complicated algorithms. Slower algorithms will work in the first place.
- DRY (Don't Repeat Yourself), YAGNI (You Aren't Gonna Need It),

Regla 6



- Mantenga un libro diario.
  - Anotaciones sobre su trabajo.
  - Diversos formatos.
  - Ayudan a tomar decisiones.
- Actas de reunión.
- Libro diario (Donald **knuth**)
  - Entradas que describen eventuales errores y como ellos fueron tratados en **TeX**.
  - **The errors of TEX**

Prácticas de Ingeniería de Software



Desarrollo iterativo

Gestión de Requisitos

Arquitectura de Software

Integración Continua

Verificación de calidad

Control de cambios