

## **Tema 1 - Estructura de un equipo de desarrollo (DCA)**

HISTORIAL DE REVISIONES			
NÚMERO	FECHA	MODIFICACIONES	NOMBRE

## Índice

1. Conceptos clave	1
2. Estructuras típicas	1
3. Otras estructuras organizativas	2
4. A tener en cuenta	2
5. Algunos paradigmas de organización de equipos	2
6. Conceptos relacionados	3
7. Dictador benevolente de por vida - <i>BDFL</i> - (I)	3
8. Dictador benevolente de por vida - <i>BDFL</i> - (II)	3
9. ¿Dictador benevolente de por vida?	3
10. Aclaraciones	4

---

Logo DLSI

**Tema 1 - Estructura de un equipo de desarrollo (DCA)**  
**Curso 2018-2019**

## 1. Conceptos clave

- Eficiencia: En un equipo de desarrollo eficiente, cada uno sabe lo que tiene que hacer, no existe la sensación de que unos se entrometen en las tareas de los otros.
- Ortogonalidad
  - *Responsabilidades bien definidas*
  - *Mínimo solapamiento*
- La ortogonalidad depende del proyecto actual y del diseño del mismo que hagamos, también de las personas con las que contamos.
- Un buen consejo para conseguir un sistema ortogonal es separar lo que es la infraestructura <sup>1</sup> de lo que es la aplicación. Cada componente de la infraestructura es gestionado por un subequipo o subgrupo.
- A continuación asignamos las personas a los grupos. ¿Cómo medir la ortogonalidad del sistema?... viendo cuánta gente (*programadores*) hay implicada en cada cambio que se pida del proyecto:
  - cuanto mayor sea el número, menos ortogonal será el sistema.
- Procura que los integrantes de tu equipo lean y aprendan la [guía de referencia del programador pragmático](#)

## 2. Estructuras típicas

ESTRUCTURA JERÁRQUICA.

- Propuesta por primera vez por Harlan Mills y descrita en 1972.
- También se la conoce como la estructura del equipo quirúrgico
- El núcleo consta de:
  - Programador o ingeniero en jefe. Planifica, coordina y revisa todas las actividades técnicas del equipo.
  - Personal técnico. Llevan a cabo las actividades de análisis y desarrollo.
  - Ingeniero de apoyo. Ayuda al ingeniero en jefe y puede sustituirle sin pérdida de continuidad del proyecto.
- Especialistas que ayudan al ingeniero en jefe. Por ejemplo, expertos en bases de datos, comunicaciones, etc.
- Un bibliotecario de software (librarian) cataloga e indexa los módulos u objetos reutilizables.

<sup>1</sup> BB.DD., Interfaz de usuario, lógica de negocio, comunicaciones, etc...

### 3. Otras estructuras organizativas

#### ■ Descentralizado democrático (DD).

- No tiene un jefe permanente. Se nombran coordinadores de tareas a corto plazo y se sustituyen por otros para diferentes tareas.
- Las decisiones y los enfoques se hacen por consenso.
- La comunicación entre los miembros del equipo es horizontal.

#### ■ Descentralizado controlado (DC).

- Tiene un jefe definido que coordina tareas específicas y jefes secundarios que tienen responsabilidades sobre subtareas.
- La solución de problemas es una actividad del grupo, pero la implementación de soluciones se reparte entre los subgrupos por el jefe de equipo.
- La comunicación entre subgrupos e individuos es horizontal.
- También hay comunicación vertical a lo largo de la jerarquía de control.

#### ■ Centralizado controlado (CC).

- El jefe de equipo se encarga de la solución de problemas a alto nivel y la coordinación interna del equipo.
- La comunicación entre el jefe y los miembros del equipo es vertical.

### 4. A tener en cuenta

PODEMOS FIARNOS EN ESTOS FACTORES PARA ESTRUCTURAR UN EQUIPO DE DESARROLLO:

- Grado de dificultad del problema.
- Nivel de modularidad permitido por el problema.
- Ambito de vida del equipo que creamos.
- Cantidad de líneas de código a escribir.
- Grado de comunicación requerido para el proyecto.
- Calidad y fiabilidad que debe tener el proyecto.
- La fecha de entrega del producto.

### 5. Algunos paradigmas de organización de equipos

**Paradigma cerrado.** Jerarquía de autoridad similar al equipo Centralizado Controlado. Es útil en la producción de nuevo software similar a otro existente.

**Paradigma aleatorio.** El equipo se estructura de manera libre en función de la iniciativa individual de los miembros. Útil cuando se requiere innovación.

**Paradigma abierto.** Estructura el equipo según una mezcla de los dos anteriores. Debe haber muy buena comunicación. Son adecuados para resolver problemas complejos, pero pueden no ser tan eficientes como otros equipos.

**Paradigma sincronizado.** Las "*partes*" del problema nos sirven para organizar los miembros del equipo, los cuales suelen trabajar en estas *partes* del problema destacando la poca comunicación entre ellos.

---

## 6. Conceptos relacionados

- No son objeto de estudio en esta asignatura.
- Los estudiarás en otras asignaturas del grado.
- Son el resultado de aplicar una organización determinada a nuestro equipo de desarrollo.
  1. **Scrum**
  2. **Extreme programming**

## 7. Dictador benevolente de por vida -**BDFL**- (I)

- Es el nombre o título que se le otorga a determinadas personas en la comunidad de desarrolladores de software de código abierto.
- Esto se puede hacer por varios factores:
  - por su personalidad
  - por su experiencia
  - por su relación con los demás integrantes del proyecto
- Es *importante* que sea el originador del proyecto software.
- Son los encargados de velar por el proyecto software que tutelan, de rodearse de las personas que consideren para abarcar todos los aspectos de ese proyecto.
- El dictador benevolente puede delegar funciones y responsabilidades pero también puede tomar las decisiones finales en un momento determinado.

## 8. Dictador benevolente de por vida -**BDFL**- (II)

ALGUNOS EJEMPLOS CONCRETOS:

- **Linus Torvalds** creador del núcleo **Linux**
- **Guido van Rossum** creador de **Python**
- **Jimmy Donal "Jimbo" Wales** creador de la **Wikipedia**
- **Alexandre Julliard** mantenedor de **Wine**
- **Andrew "Tridge" Tridgell** por **Samba**
- **Patrick Volkerding** por **Slackware**
- **Larry Wall** por **Perl**
- **Mark Shuttleworth** se llama a sí mismo el "*auto-nombrado dictador benevolente de por vida*". Creador de **Canonical** y **Ubuntu**.

## 9. ¿Dictador benevolente de por vida?

- No tiene por qué.
  - Observa el caso de **Guido van Rossum**.
  - Hay más casos, ¿conoces Django, verdad? Qué te dice **éste commmmmit** en su proyecto alojado en github?
  - Puedes ver algo más de esa *historia* **aquí**.
-

## 10. Aclaraciones

EN NINGÚN CASO ESTAS TRANSPARENCIAS SON LA BIBLIOGRAFÍA DE LA ASIGNATURA.

- Debes estudiar, aclarar y ampliar los conceptos que en ellas encuentres empleando los enlaces web y bibliografía recomendada que puedes consultar en la página web de la [ficha de la asignatura](#) y en la [web propia de la asignatura](#).