Khier, Nadia

Díez Apolo, Èric

Gámez César, Ariadna

Corcuera Cucurull, Carlota

Patiño Ojeda, Adrián

**Sessió 9 SIO**

**Apunts sessió 9**

## Índex

1. Free Software
2. Open Source
3. Agile

## Introducció històrica del codi

Al principi, el codi dels programes era molt simple, amb unes ~1000 línies de codi escrites directament en llenguatge binari o assembler (ASM). Aquestes línies controlaven sistemes molt rudimentaris amb funcionalitats limitades.

Amb l'arribada de sistemes amb 64kb de RAM, el model de desenvolupament va canviar: el codi font va començar a ocultar-se i només es distribuïa el codi compilat, preparat per executar-se però inaccessible per a modificacions o estudi. Això va donar lloc a la necessitat de protegir legalment aquest codi amb acords com l'**EULA (End User License Agreement)**.

## Free Software

**Origen**: Richard Stallman funda la **Free Software Foundation (FSF)** el 1984, establint les bases d’un moviment que defensa la llibertat de l'usuari en relació amb el software. La FSF va crear la **GPL (General Public License)** per garantir aquestes llibertats.

**Les quatre llibertats fonamentals**:

* 1. **Ús**: Llibertat per utilitzar el programa amb qualsevol finalitat.
  2. **Accés i Estudi**: Accés al codi font per entendre com funciona.
  3. **Modificació**: Possibilitat d’adaptar el programa a les pròpies necessitats.
  4. **Redistribució**: Compartir les versions originals o modificades, mantenint la mateixa llicència.

## 

## 

## Open Source

**Diferències amb el Free Software**:

* Encara que també permet l’accés al codi font, l’**Open Source** posa més èmfasi en la col·laboració i l’eficiència en el desenvolupament de software.
* La filosofia del Free Software és més idealista, mentre que l'Open Source té un enfocament més pràctic i orientat al mercat.

## La crisi de l’enginyeria del software

Segons **Roger S. Pressman**, hi ha una crisi endèmica en l’enginyeria del software que impedeix desenvolupar programes sense errors, amb totes les funcionalitats requerides i dins dels terminis previstos.

Estadístiques desmoralitzadores:

* **50% dels projectes** només compleixen **50% de les funcionalitats**.
* Això es fa en **el doble de temps** i amb **quatre vegades més costos** dels previstos.
* El rendiment efectiu acaba sent només d’un **1/32** del que es planifica.

### 

### Reflexió de Marc Alier:

“El software és com la Ronda de Dalt. El dia que implantes software en una empresa, ja està obsolet.”

Això es deu a l’evolució constant dels requeriments i la tecnologia, especialment en sistemes d'informació (SI).

## 

## Catedral i Bazar

**Eric S. Raymond**, en el seu llibre *La Catedral i el Bazar*, utilitza aquesta metàfora per explicar dos models de desenvolupament de software:

* **Catedral**: Model tradicional, caracteritzat per ser lent i amb poca participació. Exemple: Projectes com els de la FSF.
* **Bazar**: Model més dinàmic i col·laboratiu. Exemple: Desenvolupament de Linux, on moltes persones treballen de manera descentralitzada.

També reflexiona sobre com la tecnologia "petita" acaba substituint la "gran". Exemple: Els **mainframes** van ser desplaçats pels **PC** gràcies a la seva accessibilitat i adaptabilitat.

## Agile Manifest

Publicat el 2001, el **Manifest Agile** planteja quatre valors fonamentals que han canviat la manera de desenvolupar software:

* **Individus i interaccions** per sobre de **processos i eines**.
* **Software funcionant** per sobre de la **documentació completa**.
* **Col·laboració amb el client** per sobre de la **negociació de contractes**.
* **Respondre al canvi** per sobre de la **planificació exhaustiva**.

### 

### 

### Agile avui en dia

Encara que va néixer com una alternativa flexible al model de cascada, la seva implementació actual sovint replica patrons rígids.

Exemple crític: Alguns equips fan **sprints de 15 dies**, tot i que molts cops el treball podria estar complet en 2-3 dies. Això genera ineficiències i pèrdues de temps.

Ironia: Agile ha esdevingut, en alguns casos, “la nova Cascada”.