

Introducció a l'Enginyeria del Programari

Documentació i UML

Què hi ha en aquest material

2

- **Documentació**
 - Perquè necessitem documentar
 - Què hem de documentar en cada etapa del cicle de vida
- **UML**
 - L'UML és un llenguatge
 - Una sola notació per tot mètode i tota etapa
 - Com usar l'UML
 - ✦ Cal escollir
 - ✦ Documentar és donar informació
- **La trampa de l'UML**
 - Desenvolupar és fer enriquiments semàntics
 - En UML es promou l'enriquiment notacional
- **Diagrames UML emprats**
 - Breu descripció semàntica
 - Característiques específiques que usarem

Documentació en un desenvolupament

3

Dependència de les etapes del cicle de vida

4

- Les etapes d'un cicle de vida no són independents
 - Cal enllaçar una amb l'altra
 - Cal comunicació

Necessitat de la documentació

5

- Hem de tenir mecanismes per comunicar:
 - l'especificació al disseny
 - el disseny a la implementació
 - la implementació al programador

<= = Manteniment

Documentació de l'anàlisi

6

- De cara al desenvolupament pròpiament dit l'únic que ens interessa és l'especificació
 - Ens cal documentar l'especificació
- Doble vessant de l'especificació
 - Contracte amb l'usuari
 - ✦ Documentem què li oferirem
 - Contracte amb el desenvolupador
 - ✦ Documentem quin problema hem de solucionar
- El doble vessant pot significar “duplicar” la documentació
 - Si l'auditori és diferent cal canviar la manera de dir les coses

Documentació de disseny

7

- Cal documentar el procés
 - No n'hi ha prou en presentar el resultat final
 - Cal explicar quines decisions s'han pres per arribar-hi
 - ✦ Més important que quina decisió s'ha pres en cada moment és el motiu pel que s'ha pres

Documentació de la implementació

8

- És el propi codi
 - Només ens interessa el producte final
- Les decisions preses en la implementació que no s'extreuen fàcilment del codi cal documentar-les en forma de comentari
 - Hi ha eines de generació automàtica de documentació a partir dels comentaris

Mecanismes de comunicació

9

1. Cada etapa acaba amb una documentació interpretable pel responsable de la següent etapa
2. Notació comuna a totes les etapes
 - Artefactes (diagrames, codi, etc) comuns
 - ✦ Amb la mateixa “pinta”
 - ✦ Semàntica diferent
 - <== Propòsits diferents

Exemple: UML

UML

10

UML: Què és

11

- (UML: Unified Modelling Language)
- Notació unificada per a:
 - ✦ mètodes diferents
 - ✦ etapes diferents


UML: Aspectes a considerar


12

- Mètodes diferents
 - Molts artefactes
 - ✦ Per poder suportar “qualsevol” mètode” de desenvolupament
 - Ús discriminatori
 - ✦ Cal usar només els artefactes necessaris
- Etapes diferents
 - ✦ Artefactes (ex. Diagrames) similars diuen coses diferents
 - ✦ Desenvolupament com a enriquiment d'artefactes

Documentació amb UML

13

- Hi ha artefactes que no són els més apropiats per l'etapa del cicle de vida en la què estem 

Diversitat d'etapes
- Hi ha artefactes diferents que aporten exactament la mateixa informació 

Diversitat de mètodes

 - Caldrà escollir

Impacte de l'UML

14

- Pots fer el 80% del desenvolupament usant l'UML; i per fer-ho només usaràs un 20% de l'UML
 - Gran part de la potència i artefactes de l'UML no s'usen en un projecte
 - Una part del projecte no usa l'UML

Grady Booch, James Rumbaugh, Ivar Jacobson
The Unified Modelling Language User Guide

○ **Ho diuen més enllà de la pàgina 400 !!**

On és el bosc?

15

- Massa arbres. Algú veu el bosc???
- Massa arbres
 - ✦ Tenim molts artefactes al nostre abast. Hem d'escollir!
 - Si no escollim correctament podem presentar artefactes que no ens diuen res de nou, perquè d'altres artefactes ja ens han dit el mateix
- Tots els arbres són iguals
 - ✦ Hem de saber què hem de dir en cada moment. I escollir l'artefacte adient
 - Presentar un artefacte simplement perquè el sabem dibuixar no té cap sentit. L'hem d'usar quan correspongui
 - Hi ha pins, àlbers, roures, alzines, pollancre, ... En descriure un paisatge hem d'usar els noms pertinents
 - Exemple. Un diagrama d'estats només el presentarem quan en el nostre problema hi hagi estats rellevants
- Detallisme
 - ✦ És molt el que estem fent. No cal explicar-ne tots els detalls. El que cal és explicar aquells trets o decisions més importants; ens cal donar una visió de conjunt que permeti reconstruir els detalls
 - Massa artefactes, massa informació, desinforma. Hem de donar la informació justa per entendre què estem fent

Documentar és informar

16

- Cal usar els artefactes
 - Necessaris
- S'han de documentar tots els aspectes rellevants del disseny
 - Suficients
- Cal evitar l'ús d'artefactes superflus (que no aporten informació, o que aporten informació ja coneguda)

Documentar és informar

Oportunitat i completenessa de la documentació

17

- Cal dir el que s'ha de dir en el moment en què s'ha de dir
 - Exemple de no oportunitat:
 - ✦ En l'anàlisi (domini del problema) parlar de char, string o varchar (domini d'una solució tecnològica específica)
 - Exemples de no completenessa
 - ✦ No expressar enlloc que en esborrar un albarà s'esborren totes les seves línies
 - ✦ No expressar enlloc que no permetem esborrar clients amb albarans pendants de cobrar

Imatges i textos

18

- El propòsit de l'UML és ajudar-nos en la documentació
 - És el paradigma de “Una imatge val més que mil paraules”
 - ✦ Ens n'ha de facilitar l'escriptura
 - ✦ Ens n'ha de facilitar la lectura
- Mai l'UML ens ha de ser una càrrega
- L'UML no és suficient
 - Recordem que un 20% del desenvolupament no és UML
 - Sempre ens caldrà combinar els artefactes UML amb text explicatiu

L'UML com a càrrega

19

- Típics casos en què l'UML esdevé una càrrega
 - No aporta res
 - L'expressió UML és massa complicada o artificial
 - No sabem com expressar determinada propietat usant l'UML
 - ✦ O bé sabem que no es pot expressar
- Tot seguit analitzem exemples d'aquests casos típics

L'UML esdevé innecessari

20

- L'artefacte no aporta res a l'alternativa textual
 - Ni claredat, ni concreció, ni facilitat d'escriptura o de lectura, etc.
- Exemple
 - Un Diagrama de CU amb un sol actor i sense interrelacions entre els diferents CU és molt més difícil de dibuixar que una llista amb els diferents CU; i a nivell de lectura, concreció i comprensió són equivalents
- En aquest cas, cal usar sempre l'alternativa textual

L'UML és feixuc: artificialitat

21

- El diagrama resultant és massa complex o artificial
 - En intentar expressar diagramàticament determinada propietat, hem de complicar excessivament el diagrama o introduir-hi **artificialitat**
- Exemple
 - El número de sala és únic dins de cada museu, però es pot repetir en museus diferents
 - Cal introduir un concepte Número
 - És un concepte exigible diagramàticament, però sense cap informació semàntica que ens permeti defensar-lo com a concepte
 - En Chen (Diagrama Entitat-Interrelació) sí que ho podem expressar naturalment

L'UML és feixuc: buscar tres peus al gat

22

- El diagrama resultant és massa complex o artificial
 - En intentar expressar diagramàticament determinada propietat, hem de **complicar excessivament el diagrama** o introduir-hi artificialitat
- Exemples
 - Els anirem veient al llarg del curs
- En general, es tracta de casos on l'alternativa de deixar el diagrama al més simple possible, i expressar la propietat “problemàtica” textualment, ha de ser considerada atentament
 - Recordem: modelitzar significa cercar el model més simple

L'article 29

23

- Sovint l'intent d'expressar amb UML determinades propietats ens permet descobrir elements que ens havien quedat ocults
 - Ex. Hi ha conceptes ocults que només els podem descobrir en intentar forçar el Model Conceptual per tal d'expressar determinada restricció
- Per tant, és interessant intentar sempre l'UML, encara que al final expressem la propietat sense UML

La regla d'or

24

- Intent i prospecció
 - Hem d'intentar forçar els diagrames per tal d'expressar al màxim de propietats possibles
- Avaluació
 - Cal avaluar el resultat d'aquest intent
 - Si hem descobert nous elements caldrà avaluar fins a quin punt ens són rellevants
 - En tot cas caldrà que ens plantegem l'alternativa: UML simple + text
- Decisió
 - El resultat ha de ser el:
 - Més simple
 - Més clar
 - Semànticament més ric pel que fa al nostre problema

La trampa de l'UML

25

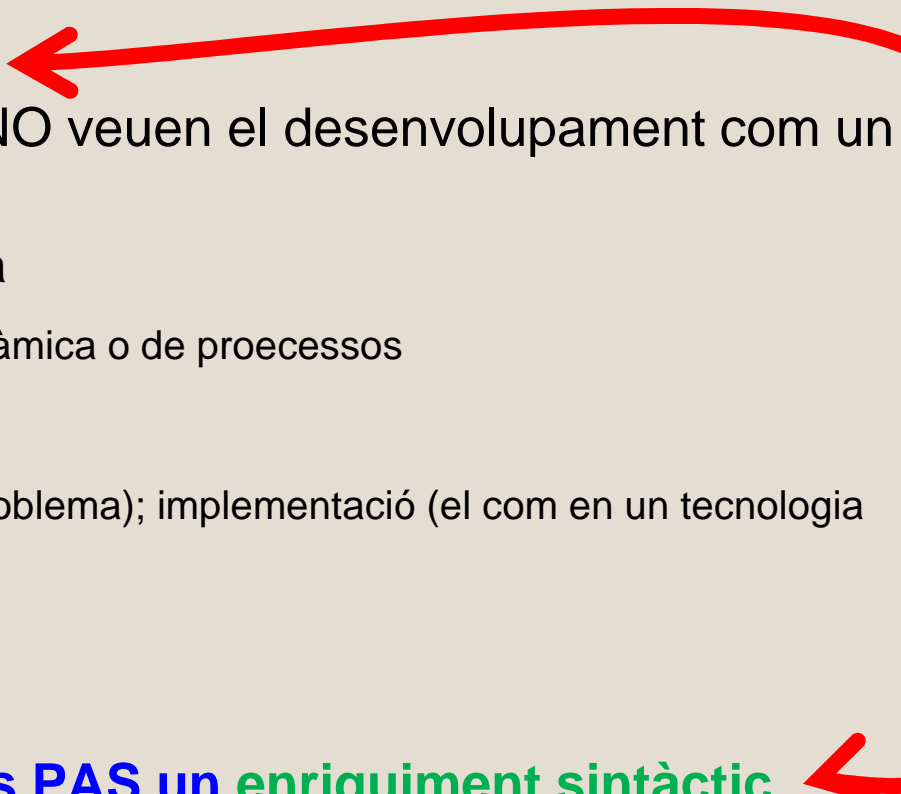
Desenvolupar amb l'UML

26

- L'UML s'ha dissenyat com un conjunt d'artefactes que es van refinant i enriquint
 - Bé si el que volem és documentar un producte final (anàlisi, implementació)
 - Perillós si el que ens interessa és documentar un procés (Disseny)
 - Per això les eines CASE (Computer Aid Software Engineering) ofereixen mecanismes per recuperar la història

La gran trampa de l'UML

27

- Per l'UML el desenvolupament és un enriquiment dels diagrames
 - **Enriquiment notacional**
 - Els diferents cicles de vida NO veuen el desenvolupament com un continu enriquiment
 - Hi ha canvis de perspectiva
 - Estàtica o d'informació; dinàmica o de processos
 - Hi ha canvis d'objectius
 - Especificació (el què del problema); implementació (el com en un tecnologia determinada)
 - Hi ha canvis de domini
 - Problema; solució
 - **El desenvolupament NO és PAS un enriquiment sintàctic**
- 

No és or tot el que lluu

28

- Alguns mètodes prioritzen l'enriquiment notacional
 - *En el fons, però, realitzen enriquiments semàntics*
 - En no explicitar, obviar o negar els enriquiments semàntics creen una corba d'aprenentatge molt més elevada
 - ✦ Potser el mètode s'aprèn ràpid
 - ✦ Però una altra cosa és obtenir resultats decents
- L'exigència de treballar explícitament amb enriquiments semàntics i canvis de domini:
 - ✦ Pot dificultar la creació del primer desenvolupament
 - ✦ S'arriba molt abans a fer desenvolupaments decents

Enriquiments i aprenentatge

29

- Desenvolupar amb un aparent refinament notacional només té sentit quan un és molt conscient de les subtiletes semàntiques que hi ha al darrera
 - Per pintar un quadre abstracte abans s'han de conèixer les lleis de la perspectiva, la teoria del color, la teoria de les textures, etc.
 - Una cosa és pintar; una altra aprendre a pintar
 - ✦ Tothom pot pintar com vulgui, però abans n'ha d'aprendre les tècniques
 - Aquí estem aprenent a pintar
 - ✦ Els mètodes basats en l'aparent refinament notacional són mètodes per a professionals, no són pas mètodes per aprendre a desenvolupar

Diagrames UML emprats

30

Diagrames de casos d'ús

31

- No els farem servir
 - Són útils per veure les interaccions entre els diferents CU
 - ✦ Quan els CU són pocs o hi ha poques interaccions, no tenen cap sentit
 - La semàntica de les interaccions entre els CU és complexa
 - ✦ Els `includes` i `extends` sovint semblen més una qüestió d'oportunitat que una distinció semàntica
 - ✦ ICONIX les substitueix per `invokes` i `preceeds`

Farem servir els casos d'ús, però no els
diagrames de Casos d'ús

Diagrames d'interacció

32

- Expliquen com els diferents elements interactuen, col·laboren, es comuniquen, ...
 - Són diagrames de **comportament**
- Considerarem dos tipus
 - Diagrames de comunicació
 - Diagrames de seqüència
- Els diagrames de comunicació i els de seqüència diuen exactament el mateix, però ho diuen de manera lleugerament diferent
 - Caldrà escollir

Diagrames de comunicació

33

- Sobre la sintaxi base hem de tenir present que:
 - Usarem numeració jeràrquica dels missatges
 - Els missatges s'escriuen en notació de pseudocodi
 - Podem incloure pseudocodi en forma de nota
 - ✦ Com a pseudocodi podem usar un llenguatge de programació

Diagrames de seqüència

34

- Sobre la sintaxi base hem de tenir present que:
 - No numerarem els missatges

Diagrames de “classes”: Un o molts?

35

- Una mateixa aparença amb múltiples semàntiques
 - Diagrama de classes de l'especificació
 - ✦ L'anomenarem **Diagrama Conceptual**
 - Diagrama de classes del disseny
 - ✦ L'anomenarem **Diagrama de Components**
 - Diagrama de classes de la implementació
 - ✦ L'anomenarem **Diagrama de Classes**
 - Diagrama de classes de la base de dades
 - ✦ L'anomenarem **Diagrama de la Base de Dades**

Diagrames de “classes”: La semàntica compartida

36

- Tots els diferents tipus de diagrames de “classes” comparteixen un mateix tret semàntic:
 - Descriuen propietats estructurals
 - ✦ Les propietats estructurals descrites dependran de l'etapa del cicle de vida
 - Anàlisi:
 - Informació rellevant del problema
 - Disseny:
 - Elements software que necessitem per a la nostra solució del problema
 - Implementació:
 - Elements del llenguatge de programació que usem per a construir els elements software de la nostra solució del problema

Diagrames de “classes”: Sintaxi

37

- Sobre la sintaxi base cal tenir present que:
 - Els elements diagramàtics usats dependran de l'etapa en la que estem construint el diagrama
 - ✦ Per exemple, en un diagrama conceptual no usarem visibilitats; en un diagrama de components usarem les visibilitats
 - El nom dels elements diagramàtics dependran de l'etapa en la que estem construint el diagrama
 - ✦ Per exemple, en un diagrama conceptual les “caixes” són conceptes; en un diagrama de components són components
- La nostra nomenclatura no és estàndard UML
 - La nomenclatura UML és confusional:
 - ✦ Tot té el mateix nom, independentment de la semàntica