



Documentatia

CDL - Cursul 4

Adrian Scoică
(adriansc@rosedu.org)
5 noiembrie 2011
ROSEdu

- 1 Introducere
- 2 Cum se scrie documentația
- 3 Doxygen
- 4 Resurse

Documentația software

Ce este?

Documentația software

Ce este?

- Text

Documentația software

Ce este?

- Text
- Tutoriale

Documentația software

Ce este?

- Text
- Tutoriale
- Metainformații

Documentația software

Ce este?

- Text
- Tutoriale
- Metainformații

... care însoțesc un produs software.

Documentația software

Ce este?

- Text
- Tutoriale
- Metainformații

... care însoțesc un produs software.

Pe cine vizează?

Documentația software

Ce este?

- Text
- Tutoriale
- Metainformații

... care însoțesc un produs software.

Pe cine vizează?

- Proiectanți (Requirements, Standards, RFC, Design Docs)

Documentația software

Ce este?

- Text
- Tutoriale
- Metainformații

... care însoțesc un produs software.

Pe cine vizează?

- Proiectanți (Requirements, Standards, RFC, Design Docs)
- Dezvoltatori (API, SDK)

Documentația software

Ce este?

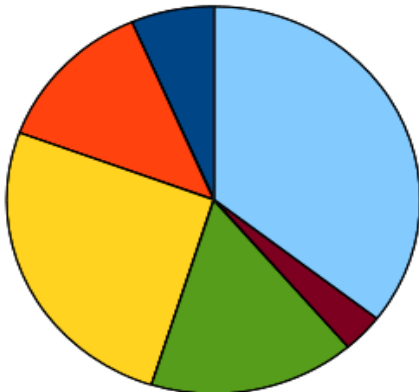
- Text
- Tutoriale
- Metainformații

... care însoțesc un produs software.

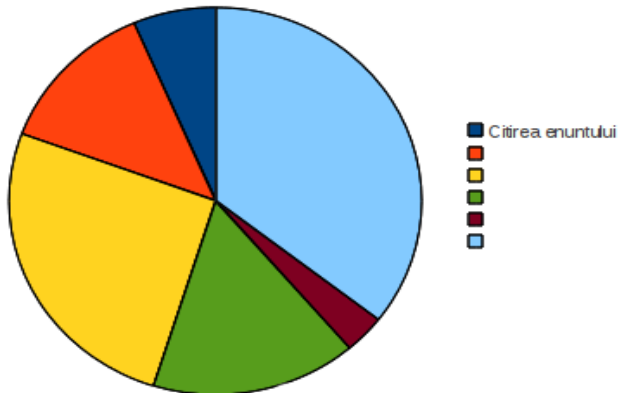
Pe cine vizează?

- Proiectanți (Requirements, Standards, RFC, Design Docs)
- Dezvoltatori (API, SDK)
- Utilizatori (Manuale, Tutoriale)

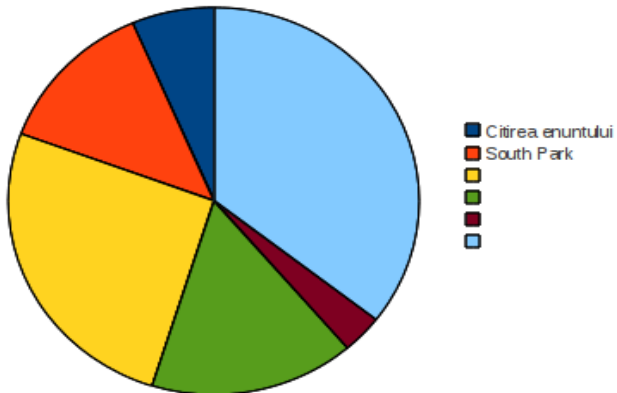
The ugly truth



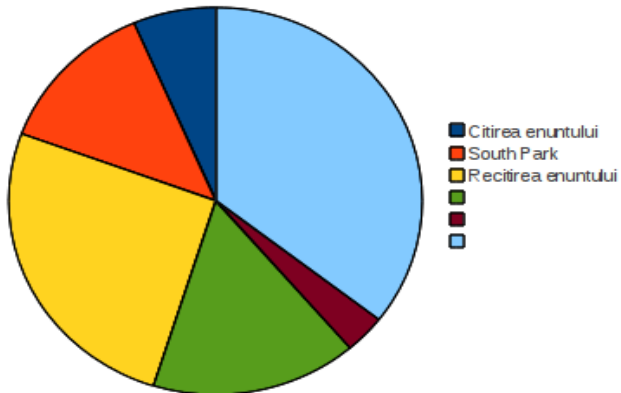
The ugly truth



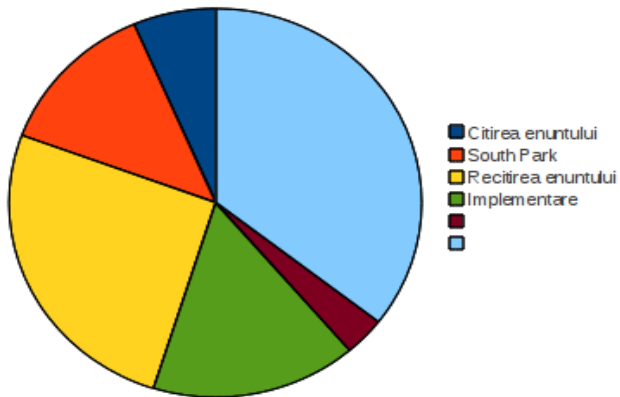
The ugly truth



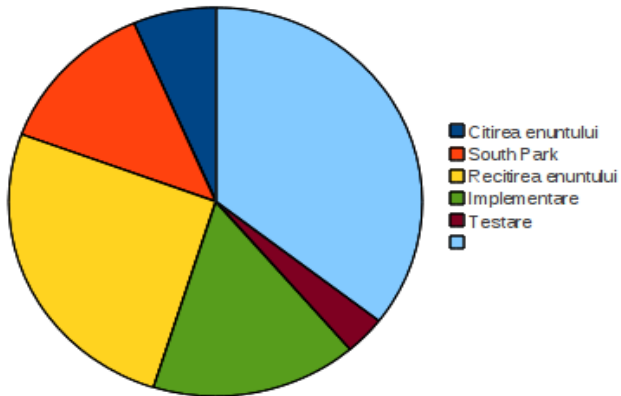
The ugly truth



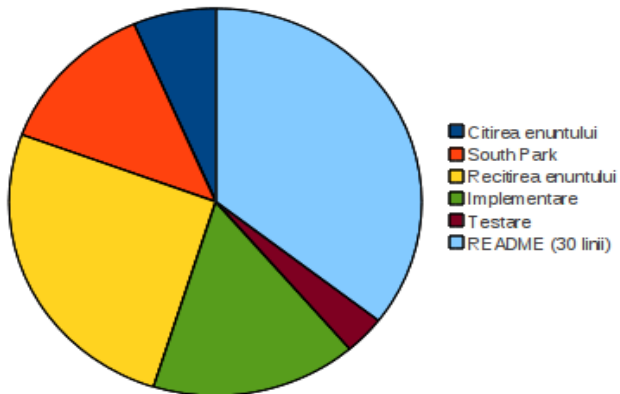
The ugly truth



The ugly truth



The ugly truth



The ugly truth

Programatorii (fără experiență) detestă să scrie documentație.

The ugly truth

Programatorii (fără experiență) detestă să scrie documentație.

- "Pentru că **pare** timp mort"

11 / 22

The ugly truth

Programatorii (fără experiență) detestă să scrie documentație.

- "Pentru că **pare** timp mort"
- "Pentru că este clar ce parsează
`<?\w+((\s+\w+(\s*\s*(?:\s*\s*'\s*\s*—[\^>\s]+))?)+\s*—\s*)?>`
- "Pentru că oricum este evident ce face **int prior_load(const std::priority_queue<SCRTok::Lexem, std::vector<SCRTok::Lexem>, std::greater<SCRTok::Lexem> >& lex);**

Consecințele

Consecințele

- Codul este greu de refolosit, înțeles și modularizat

Consecințele

- Codul este greu de refolosit, înțeles și modularizat
- Devine un coșmar să încerci să adaugi funcționalități sau să reperi bug-uri

Consecințele

- Codul este greu de refolosit, înțeles și modularizat
- Devine un coșmar să încerci să adaugi funcționalități sau să reperi bug-uri
- Ceilalți programatori evită codul

Consecințele

- Codul este greu de refolosit, înțeles și modularizat
- Devine un coșmar să încerci să adaugi funcționalități sau să reperi bug-uri
- Ceilalți programatori evită codul

... și în cele din urmă, proiectul moare.

Step 1: naming conventions

- Primul pas către un cod lizibil.

Step 1: naming conventions

- Primul pas către un cod lizibil.
- Nume scurte și sugestive.

Step 1: naming conventions

- Primul pas către un cod lizibil.
- Nume scurte și sugestive.
 - `make_sanwdich_in_c();`

Step 1: naming conventions

- Primul pas către un cod lizibil.
- Nume scurte și sugestive.
 - `make_sanwdich_in_c();`
 - `makeSandwichInCamelCase();`

Step 1: naming conventions

- Primul pas către un cod lizibil.
- Nume scurte și sugestive.
 - `make_sanwdich_in_c();`
 - `makeSandwichInCamelCase();`
 - `SandwichClass;`

Step 1: naming conventions

- Primul pas către un cod lizibil.
- Nume scurte și sugestive.
 - `make_sanwdich_in_c();`
 - `makeSandwichInCamelCase();`
 - `SandwichClass;`
- **Uniformitatea** este cuvântul cheie.

Step 1: naming conventions

- Primul pas către un cod lizibil.
- Nume scurte și sugestive.
 - `make_sanwdich_in_c();`
 - `makeSandwichInCamelCase();`
 - `SandwichClass;`
- **Uniformitatea** este cuvântul cheie.

Alte convenții (doar câteva):

- `CONSTANTELE_IN_UPPER_CASE` // (SI MACRO-URILE)

Step 1: naming conventions

- Primul pas către un cod lizibil.
- Nume scurte și sugestive.
 - `make_sanwdich_in_c();`
 - `makeSandwichInCamelCase();`
 - `SandwichClass;`
- **Uniformitatea** este cuvântul cheie.

Alte convenții (doar câteva):

- `CONSTANTELE_IN_UPPER_CASE` // (SI MACRO-URILE)
- `_membrii _privati _cu _underscore` în față

Step 1: naming conventions

- Primul pas către un cod lizibil.
- Nume scurte și sugestive.
 - `make_sanwdich_in_c();`
 - `makeSandwichInCamelCase();`
 - `SandwichClass;`
- **Uniformitatea** este cuvântul cheie.

Alte convenții (doar câteva):

- `CONSTANTELE_IN_UPPER_CASE` // (SI MACRO-URILE)
- `_membrii` `_privati` `_cu` `_underscore` în față
- **pTable** este un **p**Pointer la un **Table**

Step 1: naming conventions

- Primul pas către un cod lizibil.
- Nume scurte și sugestive.
 - `make_sanwdich_in_c();`
 - `makeSandwichInCamelCase();`
 - `SandwichClass;`
- **Uniformitatea** este cuvântul cheie.

Alte convenții (doar câteva):

- `CONSTANTELE_IN_UPPER_CASE` // (SI MACRO-URILE)
- `_membrii` `_privati` `_cu` `_underscore` în față
- **pTable** este un **p**Pointer la un Table
- **hTable** este un **h**Handler pentru un Tabel

Step 1: naming conventions

- Primul pas către un cod lizibil.
- Nume scurte și sugestive.
 - `make_sanwdich_in_c();`
 - `makeSandwichInCamelCase();`
 - `SandwichClass;`
- **Uniformitatea** este cuvântul cheie.

Alte convenții (doar câteva):

- `CONSTANTELE_IN_UPPER_CASE` // (SI MACRO-URILE)
- `_membrii` `_privati` `_cu` `_underscore` în față
- **pTable** este un **p**Pointer la un Table
- **hTable** este un **h**Handler pentru un Tabel
- "**line**" este mai bine decât "**l**"

Step 1: naming conventions

- Primul pas către un cod lizibil.
- Nume scurte și sugestive.
 - `make_sanwdich_in_c();`
 - `makeSandwichInCamelCase();`
 - `SandwichClass;`
- **Uniformitatea** este cuvântul cheie.

Alte convenții (doar câteva):

- `CONSTANTELE_IN_UPPER_CASE` // (SI MACRO-URILE)
- `_membrii` `_privati` `_cu` `_underscore` în față
- **pTable** este un **p**Pointer la un Table
- **hTable** este un **h**Handler pentru un Tabel
- **"line"** este mai bine decât **"l"**
- **"index_to_line"** este mai rău decât **"l"**

Step 1: naming conventions

- Primul pas către un cod lizibil.
- Nume scurte și sugestive.
 - `make_sanwdich_in_c()`;
 - `makeSandwichInCamelCase()`;
 - `SandwichClass`;
- **Uniformitatea** este cuvântul cheie.

Alte convenții (doar câteva):

- `CONSTANTELE_IN_UPPER_CASE` // (SI MACRO-URILE)
- `_membrii` `_privati` `_cu` `_underscore` în față
- **pTable** este un **pPointer** la un `Table`
- **hTable** este un **hHandler** pentru un `Tabel`
- **"line"** este mai bine decât **"l"**
- **"index_to_line"** este mai rău decât **"l"**

Step 2: comentariile

- **Cel mai important** pas către un cod vizibil.

Step 2: comentariile

- **Cel mai important** pas către un cod vizibil.
- Urmăresc structura logică a codului (**paragrafe**)

Step 2: comentariile

- **Cel mai important** pas către un cod vizibil.
- Urmăresc structura logică a codului (**paragrafe**)
- Aduc completări necesare (nu redundante): complexitate, sfaturi, cazuri de eroare

Step 2: comentariile

- **Cel mai important** pas către un cod vizibil.
- Urmăresc structura logică a codului (**paragrafe**)
- Aduc completări necesare (nu redundante): complexitate, sfaturi, cazuri de eroare
- Nu trebuie să conțină povești, ASCII art, proverbe,...

Step 2: comentariile

- **Cel mai important** pas către un cod vizibil.
- Urmăresc structura logică a codului (**paragrafe**)
- Aduc completări necesare (nu redundante): complexitate, sfaturi, cazuri de eroare
- Nu trebuie să conțină povești, ASCII art, proverbe,...
- ... sau înjurături ...

Step 2: comentariile

- **Cel mai important** pas către un cod vizibil.
- Urmăresc structura logică a codului (**paragrafe**)
- Aduc completări necesare (nu redundante): complexitate, sfaturi, cazuri de eroare
- Nu trebuie să conțină povești, ASCII art, proverbe,...
- ... sau înjurături ...

În proiectele mari, este bine să includeți la începutul fișierului un antet cu:

- Versiunea de fișier (cu data curentă)

Step 2: comentariile

- **Cel mai important** pas către un cod vizibil.
- Urmăresc structura logică a codului (**paragrafe**)
- Aduc completări necesare (nu redundante): complexitate, sfaturi, cazuri de eroare
- Nu trebuie să conțină povești, ASCII art, proverbe,...
- ... sau înjurături ...

În proiectele mari, este bine să includeți la începutul fișierului un antet cu:

- Versiunea de fișier (cu data curentă)
- Autorul codului

Step 2: comentariile

- **Cel mai important** pas către un cod vizibil.
- Urmăresc structura logică a codului (**paragrafe**)
- Aduc completări necesare (nu redundante): complexitate, sfaturi, cazuri de eroare
- Nu trebuie să conțină povești, ASCII art, proverbe,...
- ... sau înjurături ...

În proiectele mari, este bine să includeți la începutul fișierului un antet cu:

- Versiunea de fișier (cu data curentă)
- Autorul codului
- O metodă de contact

Step 2: comentariile

- **Cel mai important** pas către un cod vizibil.
- Urmăresc structura logică a codului (**paragrafe**)
- Aduc completări necesare (nu redundante): complexitate, sfaturi, cazuri de eroare
- Nu trebuie să conțină povești, ASCII art, proverbe,...
- ... sau înjurături ...

În proiectele mari, este bine să includeți la începutul fișierului un antet cu:

- Versiunea de fișier (cu data curentă)
- Autorul codului
- O metodă de contact

Step 2: comentariile - câte greșeli vedeți?

```
1 std::ifstream in("joc2.in"); // Deschide fis input
2 std::ofstream out("joc2.out"); /* Deschide fis output */
3 // ***** == Citirea datelor == *****
4 in >> N; // Better hope the s*** doesn't fail reading
5 for (int I = 0; I < N; I++)
6     in >> tab[I].first >> tab[I].second;
7 // *****
8 // This would've been much better in O(NlogN), but
9 // who cares? You don't always get what you want in life...
10 for (int w = 1; w <= 100 /* maxim 100 linii */; w++){
11     for (int h = 1; h <= 100; h++){
12         if (w <= h) // If w <= h
13             a[w][h] = count_all(w,h);
14         else
15             a[w][h] = a[h][w];
16         //std::cout << a[w][h] << " ";
17     }
18 }
```

Step 2: comentariile - un exemplu pozitiv

```
1  int fd_src, fd_dst, rc, bytesRead;
2
3  /* check if uage is correct */
4  if (argc < 2 || argc > 3){
5      printf("Wrong usage!\n");
6      return 0;
7  }
8
9  /* open source file for reading */
10
11 if (argc == 3) {
12     /* redirect stdout to destination file */
13 }
14
15 /* read from file and print to stdout */
16
17 /* close file */
```

Step 3: Doxygen

- Este dificil să scriem documentația de la 0, separat de cod.

Step 3: Doxygen

- Este dificil să scriem documentația de la 0, separat de cod.
- Am vrea ca documentația să fie generabilă din surse.

Step 3: Doxygen

- Este dificil să scriem documentația de la 0, separat de cod.
- Am vrea ca documentația să fie generabilă din surse.
- Am vrea un limbaj simplu prin care să legăm explicațiile de elemente din cod.

Step 3: Doxygen

- Este dificil să scriem documentația de la 0, separat de cod.
- Am vrea ca documentația să fie generabilă din surse.
- Am vrea un limbaj simplu prin care să legăm explicațiile de elemente din cod.
- Am vrea ca documentația să nu interacționeze cu codul compilabil ($\Rightarrow$ comentarii).

Step 3: Doxygen

- Este dificil să scriem documentația de la 0, separat de cod.
- Am vrea ca documentația să fie generabilă din surse.
- Am vrea un limbaj simplu prin care să legăm explicațiile de elemente din cod.
- Am vrea ca documentația să nu interacționeze cu codul compilabil (=> comentarii).
- Am vrea o soluție uniformă pentru mai multe limbaje: C, C++, Java, PHP, C#, Perl

Step 3: Doxygen

- Este dificil să scriem documentația de la 0, separat de cod.
- Am vrea ca documentația să fie generabilă din surse.
- Am vrea un limbaj simplu prin care să legăm explicațiile de elemente din cod.
- Am vrea ca documentația să nu interacționeze cu codul compilabil (=> comentarii).
- Am vrea o soluție uniformă pentru mai multe limbaje: C, C++, Java, PHP, C#, Perl
- Am vrea documentație generabilă în mai multe limbaje: HTML, LaTeX, RTF

Doxygen este un utilitar open source flexibil și foarte puternic, dar ușor de folosit.

Step 3: Doxygen

- Este dificil să scriem documentația de la 0, separat de cod.
- Am vrea ca documentația să fie generabilă din surse.
- Am vrea un limbaj simplu prin care să legăm explicațiile de elemente din cod.
- Am vrea ca documentația să nu interacționeze cu codul compilabil (=> comentarii).
- Am vrea o soluție uniformă pentru mai multe limbaje: C, C++, Java, PHP, C#, Perl
- Am vrea documentație generabilă în mai multe limbaje: HTML, LaTeX, RTF

Doxygen este un utilitar open source flexibil și foarte puternic, dar ușor de folosit.

Parsează codul sursă și interpretează comentariile din el.

Step 3: Sintaxa Doxygen

Doxygen analizează acele comentarii care preced declarații sau definiții și au următoarea sintaxă:

```
1  /*!  
2  ... Acesta este un comentariu specific Qt (C++, dar si C)  
3  */  
4  
5  /*! Acesta este un comentariu specific Qt pe o singura linie  
6  
7  /**  
8   * Acesta este un comentariu specific Java  
9   */  
10  
11  /// Acesta este un comentariu specific Java pe o singura linie  
12  
13
```

Step 3: Sintaxa Doxygen - specificații

```
1  /**
2   * Descriere detaliata a functiei
3   * @brief Descriere pe scurt a functiei
4   * @param x Descrierea parametrului x
5   * @param y Descrierea parametrului y
6   * @return Descrierea valorii returnate
7
8   * Un <b>exemplu</b> folosire form. matematice in text:
9   * Distanța e  $\sqrt{(x_2-x_1)^2 + (y_2 - y_1)^2}$ 
10  * Daca dorim ca formula sa fie centrata pe o singura linie:
11  *  $\sqrt{(x_2-x_1)^2 + (y_2 - y_1)^2}$ 
12  * Observatie: toate formulele trebuie sa fie comenzi LaTeX.
13  * Trebuie ca LaTeX sa fie instalat pe calculator.
14  */
15
16
17 double distantaInPlan(Point2D& x, Point2D& y);
```

Step 4: Generarea documentatiei cu Doxygen

Avem mai intai nevoie de un fisier de configurare pentru doxygen.

Step 4: Generarea documentatiei cu Doxygen

Avem mai intai nevoie de un fisier de configurare pentru doxygen.

- Un fisier de configurare se poate obtine cu comanda **doxygen -g fisier_cofig**

Step 4: Generarea documentatiei cu Doxygen

Avem mai intai nevoie de un fisier de configurare pentru doxygen.

- Un fisier de configurare se poate obtine cu comanda **doxygen -g fisier_cofig**

Al doilea pas este editarea fisierului conform cu cerintele noastre.

Step 4: Generarea documentatiei cu Doxygen

Avem mai intai nevoie de un fisier de configurare pentru doxygen.

- Un fisier de configurare se poate obtine cu comanda **doxygen -g fisier_cofig**

Al doilea pas este editarea fisierului conform cu cerintele noastre.

- In final, vom folosi fisierul pentru a genera documentația **doxygen fisier_config**

Step 4: Generarea documentatiei cu Doxygen

Avem mai intai nevoie de un fisier de configurare pentru doxygen.

- Un fisier de configurare se poate obtine cu comanda **doxygen -g fisier_cofig**

Al doilea pas este editarea fisierului conform cu cerintele noastre.

- In final, vom folosi fisierul pentru a genera documentația **doxygen fisier_config**
- **TASK** Descărcați de la adresa http://swarm.cs.pub.ro/~adrian.sc/doxygen_task fisierele sursă de C++ și generați documentația aferentă lor.

Resurse

- http://www-glast.slac.stanford.edu/software/CodeHowTo_20021202/doxygen/using_c

Resurse

- http://www-glast.slac.stanford.edu/software/CodeHowTo_20021202/doxygen/using_c
- <http://www.devtopics.com/13-tips-to-comment-your-code/>

Resurse

- http://www-glast.slac.stanford.edu/software/CodeHowTo_20021202/doxygen/using_c
- <http://www.devtopics.com/13-tips-to-comment-your-code/>
- <http://www.codinghorror.com/blog/2004/11/when-good-comments-go-bad.html>

Resurse

- http://www-glast.slac.stanford.edu/software/CodeHowTo_20021202/doxygen/using_c
- <http://www.devtopics.com/13-tips-to-comment-your-code/>
- <http://www.codinghorror.com/blog/2004/11/when-good-comments-go-bad.html>

- Întrebări?

