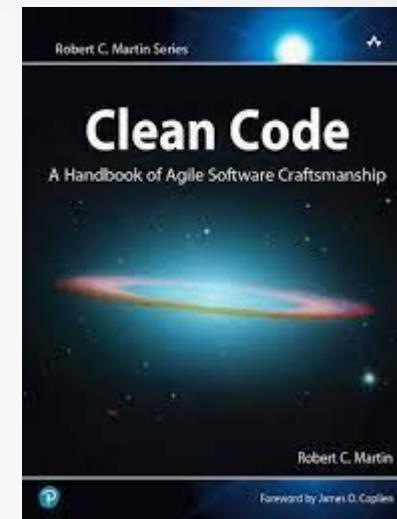


Kulturno programiranje ("Clean Code")

Clean Code knjiga



Robert C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*. Upper Saddle River, NJ: Prentice Hall, 2009.

Sadržaj

Šta je to kvalitetan kod?

Zašto je to uopšte važno?

Filozofija

Kako poboljšati kvalitet koda

- Imena
- Funkcije
- Komentari
- Formatiranje

Principi

Zabaviti se – ne postoji savršen kod!!!!

Hmm



“Objektivna” mera dobrog koda

ŠJBO/min

Izražena od strane nekoga ko gleda/ocenjuje/testira (dopuniti listu) dotični kod prvi put

U engleskoj literaturi se ta mera izražava kao:

WTFs/min

Objektivne mere

Kompleksnost koda

T. J. McCabe, "A Complexity Measure," in *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 308-320, Dec. 1976, doi: 10.1109/TSE.1976.233837.

Indeks održavanja (eng. *Maintainability index*)

$$MI = 171 - 5.2 \log_2 V - 0.23G - 16.2 \log_2 L + 50 \sin(\sqrt{2.4C})$$

Don M. Coleman et al. "Using Metrics to Evaluate Software System Maintainability"
IEEE Computer 27(8), 1994.

Ipak dosta toga ne može da se izmeri

Na primer:

Da li su upotrebljena imena odgovarajuća?

Da li je koncept realizacije dobar?

Da li su primenjeni algoritmi odgovarajući

...

Bjarne Stroustrup

Sviđa mi se da je moj kod elegantan i efikasan.

Jasnom logikom otežavam da se poneki bag sakrije.

Zavisnosti treba da su minimalne radi lakšeg održavanja.

Razrešenje grešaka u skladu sa jasnom strategijom.

Performanse bliske optimalnim da ne bi vabile ljude da menjaju kod sa neprincipijelnim optimizacijama.

Kvalitetan kod radi jednu stvar i radi je dobro.



Šta čini kvalitetan kod?

Samodokumentujući

Minimalno zavisan

Efikasan

Lepo (elegantno) pisan

Očekivan

Jednostavan

Ekspresivan

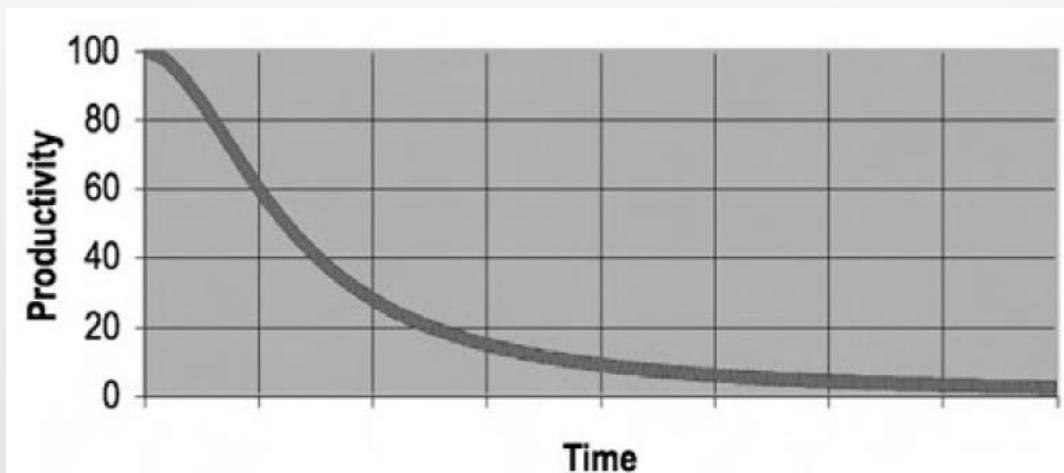
Testabilan i testiran

Lako čitljiv

Održavanje je lako

Zašto je ova tema važna?

1. Produktivnost tima
2. Skupo je imati nekvalitetan kod
3. Produktivost opada asimptotski do 0
4. Stara narodna: “Piši kod poštjujući pretpostavku
kao da će ga održavati psihotični serijski ubica koji
zna gde stanuješ.“
5. Srećniji developeri



Filozofija

1. Principi i tehnike
2. Kod je “živa” stvar
3. Tretirati pisanje koda kao da je u pitanju fini zanat (eng. *craftsmanship*)

Imena

Smislena imena govore o nameri

```
int d; // elapsed time in days
```

```
int elapsedTimeInDays;
```

```
int daysSinceCreation;
```

```
int daysSinceModification;
```

```
int fileAgeInDays;
```

Imena

-Ne koristiti imena koja su dvosmislena ili koja mogu da se pogrešno protumače

-Čak i font ima uticaja

II1O0o != 1I100o?

```
int a = 1;  
if ( 0 == 1 )  
    a = 01;  
else  
    l = 01;
```

Imena

Uporediti:

```
for (int j=0; j<34; j++) {  
    s += (t[j]*4)/5;  
}
```

```
int realDaysPerIdealDay = 4;  
  
const int WORK_DAYS_PER_WEEK = 5;  
  
int sum = 0;  
  
for (int j=0; j < NUMBER_OF_TASKS; j++) {  
    int realTaskDays = taskEstimate[j] * realDaysPerIdealDay;  
    int realTaskWeeks = (realdays / WORK_DAYS_PER_WEEK);  
    sum += realTaskWeeks;  
}
```

Imena

-Izbegavati tzv. Mađarsku notaciju
(kodovani prefiksi koji bi služili da daju
više info)

dwTmp, iTmp, fTmp, dTmp

Funkcije

Prvo pravilo je da funkcije budu male.
Drugo pravilo je da budu još manje.
Koliko je to malo?

Funkcije

Funkcije treba da rade 1 (JEDNU) stvar,
da je rade DOBRO i samo to.

Funkcije - argumenti

Idealna funkcija ima NULA argumenata,
potom slede sa 1, 2 i 3 argumenta.

Više od 3 argumenta izbegavati.

Prosleđivanje boolean argumenta
implicira da funkcija radi više od 1
stvari!

Funkcije – struktuirano programiranje

Dijkstra:

Svaka funkcija i svaki blok unutar funkcije
mora imati jedan ulaz i jedan izlaz.

Prevod:

1 return, bez break/continue i NIKAD goto

Structured Programming, O.-J. Dahl, E. W. Dijkstra, C. A. R. Hoare, Academic Press, London, 1972.

“Mrtve” funkcije i deo koda

Ne treba se plašiti obrisati funkciju koja se ne koristi

```
int foo (int iX, int iY)
{
    int iZ = iX/iY;

    return iX*iY;
}
```

Iako se iZ ne koristi, pojaviće se izuzetak
u slučaju kad je iY=0 !

Duplikati, redundantan i neaktivan kod

```
int foo(int ix)
{
    int iy = ix*2;

    return ix*2;
}
```

```
int shorter_magnitude(int u1, int v1, int u2, int v2)
{
    int temp;
    if (u1*u1 + v1*v1 < u2*u2 + v2*v2)
        temp = u1*u1 + v1*v1; /* Redundant already calculated for comparison */
    else
        temp = u2*u2 + v2*v2; /* Redundant already calculated for comparison */
    return sqrt(temp);
}
```

Duplikati, redundantan i neaktivan kod

```
extern int array_a[];
extern int array_b[];

int sum_a = 0;

for (int i = 0; i < 4; i++)
    sum_a += array_a[i];

int average_a = sum_a / 4;

int sum_b = 0;

for (int i = 0; i < 4; i++)
    sum_b += array_b[i];

int average_b = sum_b / 4;
```

```
int calc_average_of_four(int* array) {
    int sum = 0;
    for (int i = 0; i < 4; i++)
        sum += array[i];

    return sum / 4;
}
```

```
extern int array1[];
extern int array2[];

int average1 = calc_average_of_four(array1);
int average2 = calc_average_of_four(array2);
```

Duplikati, redundantan i neaktivan kod

```
int foo(void)
{
    int a = 24;
    int b = 25; /* Assignment to dead variable */
    int c;
    c = a * 4;
    return c;
    b = 24; /* Unreachable code */
    return 0;
}
```



Neaktivan kod, osim nepotrebnog zauzimanja memorije, je opasan ukoliko program ipak dođe nekom greškom do njega.

Mrvaja (eng. Oxbow)

```
static void
as_global_escape(const fn_call& fn)
{
    // List of chars we must convert to escape sequences
    const string strHexDigits = "0123456789ABCDEF";
    string strInput = fn.arg(0).to_string();
    URL::encode(strInput);
    fn.result->set_string(strInput.c_str());
}
```



strHexDigits se više ne koristi, a nekad jeste.

Komentari

Komentari su često pokušaj da se “ispliva” iz lošeg koda koji se najčešće završava “davljenjem”.

Rešenje – srediti kod da bude kvalitetan.

Komentari

Korisni komentari:

1. Informišući
2. Objašnjenje namere koja nije očigledna
3. Upozorenje o posledicama
4. ToDo
5. Isticanje značajnog dela
6. Samodokumentovanje
7. Zaglavlja, legalne info itd.

Komentari

Nekorisni komentari:

1. Nejasni
2. Nepotrebni `i++; // increment i`
3. Duplikati
4. Pogrešni/neispravljeni/zastareli
5. } // kraj funkcije Umesto ovoga skratiti funkciju
6. Izbacivanje koda komentarisanjem
7. Prekomentarisan kod

Formatiranje

- Formatiranje koda je bitno. Ne radi funkcionalnosti nego radi bolje komunikacije
- Vertikalno: Kolika da bude veličina fajla?
(nekada pravilo jednog ekrana)
- Analogija sa dobriim novinskim člankom
- Vertikalne praznine između koncepata

Vertikalne praznine

```
package fitnesse.wikitext.widgets;

import java.util.regex.*;

public class BoldWidget extends ParentWidget {
    public static final String REGEXP = "'''.+?''''";
    private static final Pattern pattern = Pattern.compile("'''(.+?)'''",
        Pattern.MULTILINE + Pattern.DOTALL
    );

    public BoldWidget(ParentWidget parent, String text) throws Exception {
        super(parent);
        Matcher match = pattern.matcher(text);
        match.find();
        addChildWidgets(match.group(1));
    }

    public String render() throws Exception {
        StringBuffer html = new StringBuffer("<b>");
        html.append(childHtml()).append("</b>");
        return html.toString();
    }
}
```

```
package fitnesse.wikitext.widgets;
import java.util.regex.*;
public class BoldWidget extends ParentWidget {
    public static final String REGEXP = "'''.+?''''";
    private static final Pattern pattern = Pattern.compile("'''(.+?)'''",
        Pattern.MULTILINE + Pattern.DOTALL);
    public BoldWidget(ParentWidget parent, String text) throws Exception {
        super(parent);
        Matcher match = pattern.matcher(text);
        match.find();
        addChildWidgets(match.group(1));
    }
    public String render() throws Exception {
        StringBuffer html = new StringBuffer("<b>");
        html.append(childHtml()).append("</b>");
        return html.toString();
    }
}
```

Vertikalna gustina

```
public class ReporterConfig {  
  
    /**  
     * The class name of the reporter listener  
     */  
    private String m_className;  
  
    /**  
     * The properties of the reporter listener  
     */  
    private List<Property> m_properties = new ArrayList<Property>();  
  
    public void addProperty(Property property) {  
        m_properties.add(property);  
    }
```

Ističe pripadnost

```
public class ReporterConfig {  
    private String m_className;  
    private List<Property> m_properties = new ArrayList<Property>();  
  
    public void addProperty(Property property) {  
        m_properties.add(property);  
    }
```

Formatiranje

- Horizontalno: koliko dugačak red da bude?
- pravilo 2 ili 4 space-a

```
public class FitNesseServer implements SocketServer { private FitNesseContext  
context; public FitNesseServer(FitNesseContext context) { this.context =  
context; } public void serve(Socket s) { serve(s, 10000); } public void  
serve(Socket s, long requestTimeout) { try { FitNesseExpediter sender = new  
FitNesseExpediter(s, context);  
sender.setRequestParsingTimeLimit(requestTimeout); sender.start(); }  
catch(Exception e) { e.printStackTrace(); } } }
```

Formatiranje

```
public class FitNesseServer implements SocketServer {  
    private FitNesseContext context;  
  
    public FitNesseServer(FitNesseContext context) {  
        this.context = context;  
    }  
  
    public void serve(Socket s) {  
        serve(s, 10000);  
    }  
  
    public void serve(Socket s, long requestTimeout) {  
        try {  
            FitNesseExpediter sender = new FitNesseExpediter(s, context);  
            sender.setRequestParsingTimeLimit(requestTimeout);  
            sender.start();  
        }  
        catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Formatiranje

There are two types of people.

```
if (Condition)
{
    Statements
    /*
     ...
     */
}
```

```
if (Condition) {
    Statements
    /*
     ...
     */
}
```

Dodatno

- Build zahteva više od jedne radnje.
- Upotreba različitih programskih jezika u istom izvornom fajlu
- Ne podrazumevati da radi – testirati
- Zameniti hard kodovane brojeve sa imenovanim konstantama

Za kraj (nepotpuna) lista principa

- DRY
- KISS
- Pravilo izviđača (*boy scout rule*)
- Uvek potraži osnovni uzrok problema
- YAGNI (You Aren't Gonna Need It)
- Princip najmanjeg iznenadenja
- Doslednost

HVALA NA PAŽNJI !



KARATE



SUMO



WUSHU



KICKBOXING



TAEKWONDO



KENJUTSU



MUAY THAI



NINJUTSU



KENDO



SAVATE



AIKIDO



BOXING



CAPOEIRA



JUDO



KUNG FU



NUNTYAKU JUTSU