DĚLAT
DOBRÝ SOFTWARE
NÁS BAVÍ

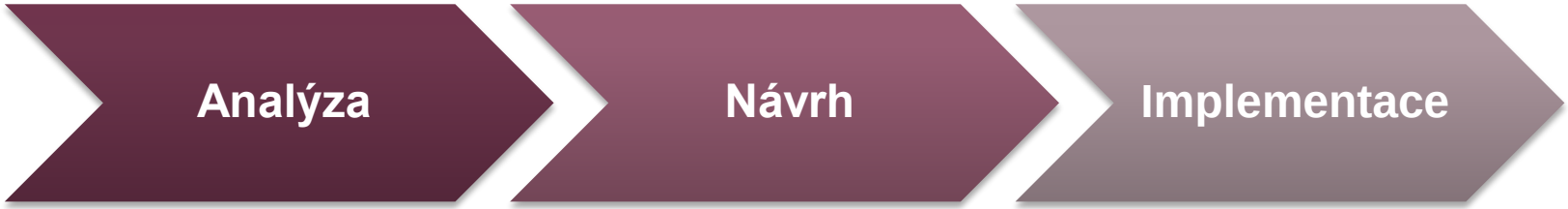
PROFINIT

Design and Construction

Tomáš Krátký, Bohumír Zoubek, Jiří Toušek
tomas.kratky, bohumir.zoubek, jiri.tousek@profinit.eu
http://www.profinit.eu/cz/podpora-univerzit/univerzitni-vyuka

30.3.2016

Připomenutí

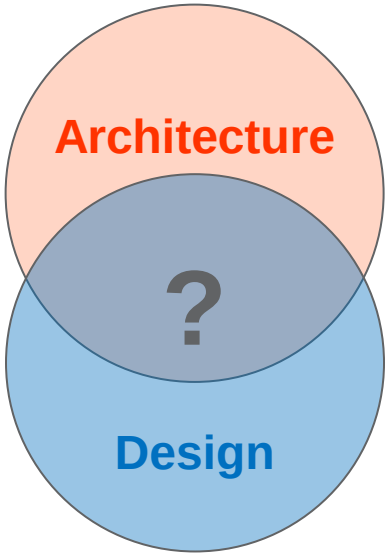


- „Co“
- „Jak“

PROFINIT

Návrh

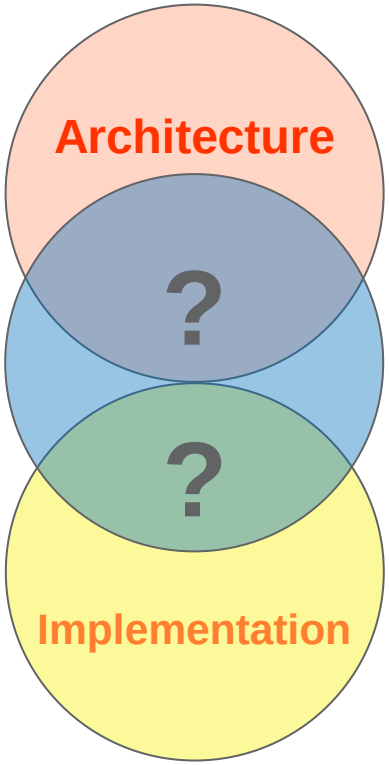
Architektura vs. návrh



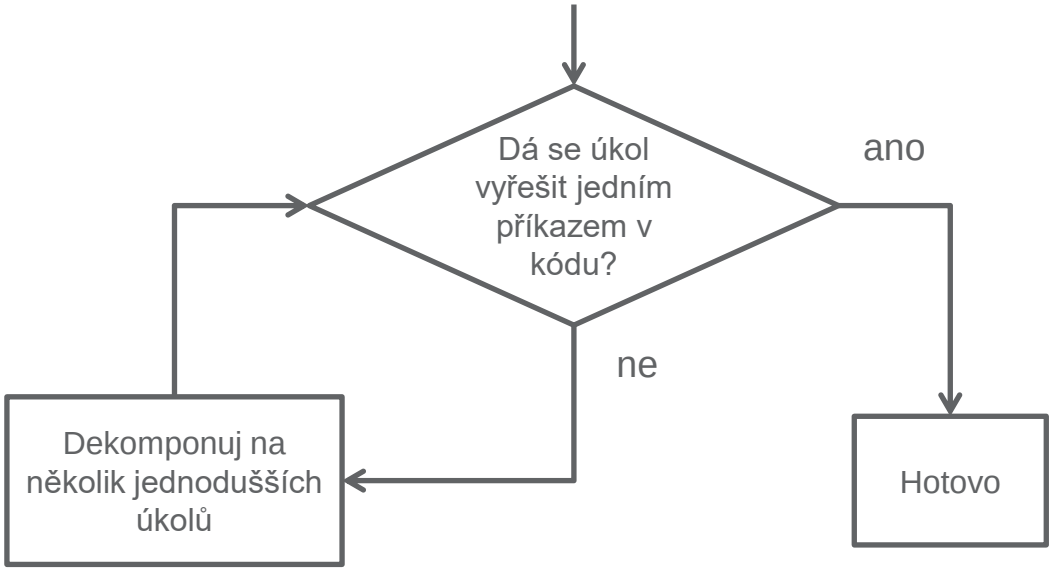
*„Architecture is about the **important** stuff. Whatever that is …“*
- Martin Fowler, Who needs an Architect ?

PROFINIT

Architektura vs. návrh vs. implementace



Návrh v kostce:

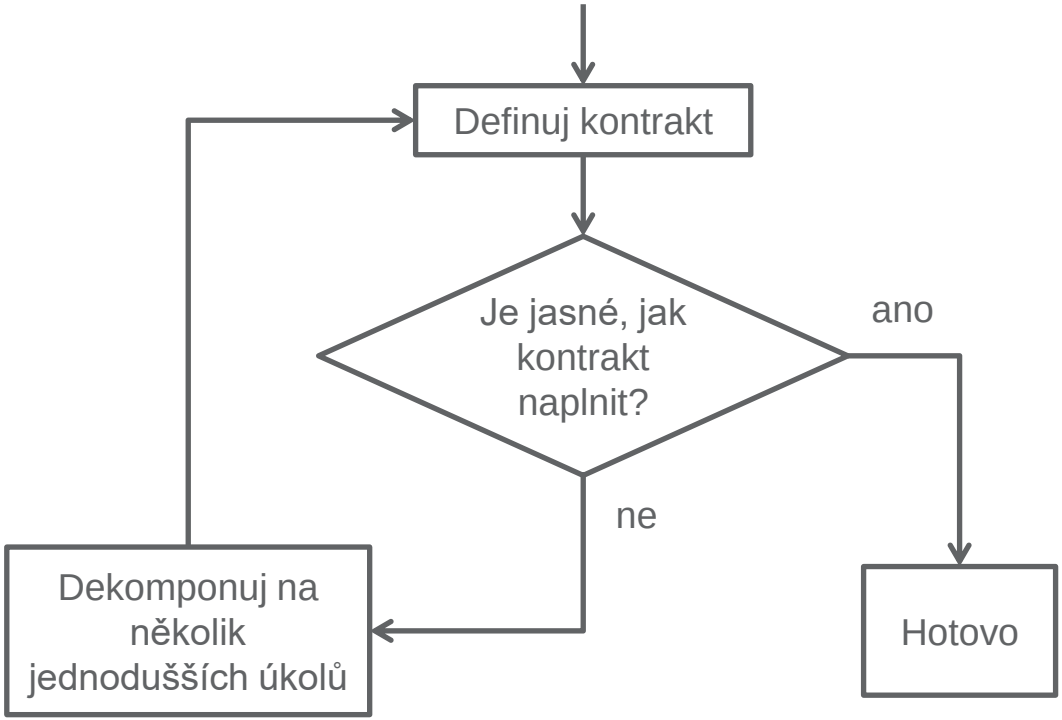


PROFINIT

Kontrakt

PROFINIT

Návrh v kostce – lépe



PROFINIT

Kontrakt

```
public class MyClass extends MyBaseClass implements MyInterface {

    public static final int THE_ANSWER = 42;

    protected MyClass() {
        System.out.printLn(“Hello World!");
    }

    public final ArrayList<String> handleStuff(ArrayList<Object> input)
                                                    throws IOException {
        return myHandleStuff(input);
    }

    protected abstract ArrayList<String> handleStuff(
            ArrayList<Object> input) throws IOException;

}
```

PROFINIT

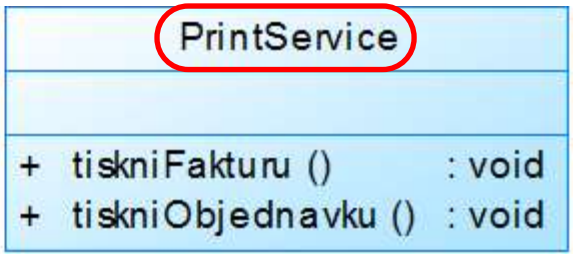
Základy dobrého (OOP) návrhu

- Decomposition
- Abstraction

- High cohesion
- Loose coupling
- Encapsulation

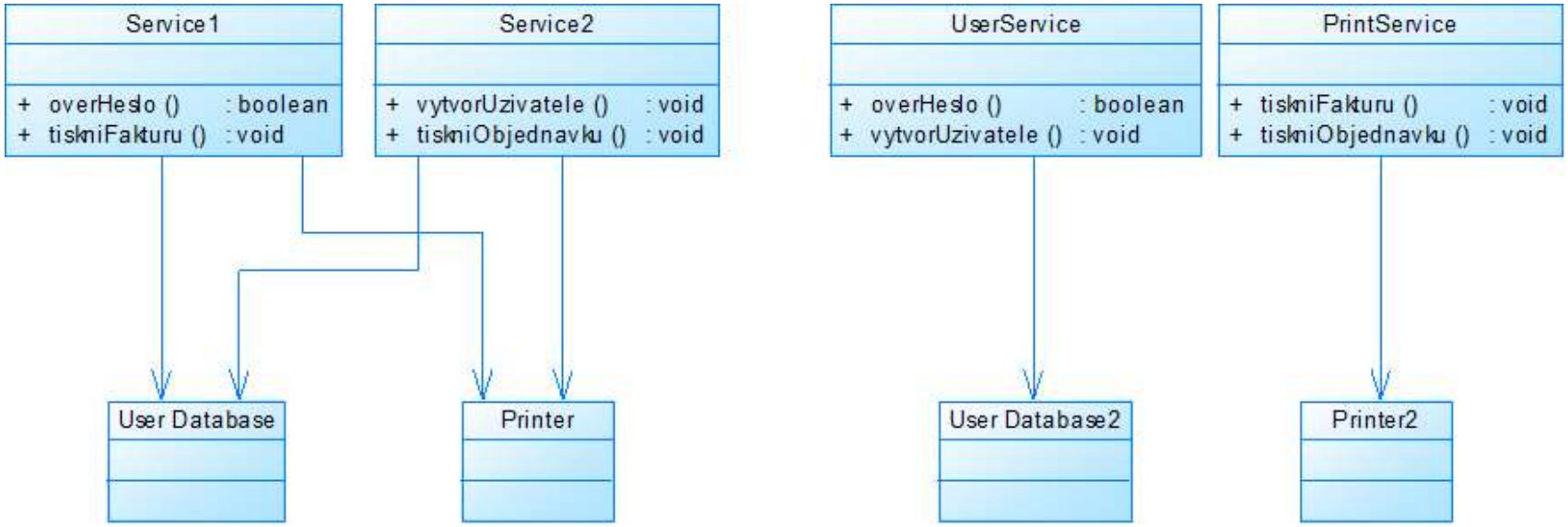
PROFINIT

Cohesion (soudržnost)



PROFINIT

Coupling (provázanost)



PROFINIT

Coupling

```
public class LimitedList<T> extends ArrayList<T> {
    …
}
```

```
ArrayList<Object> myList = new LimitedList<>();
```

PROFINIT

Coupling

```
public class LimitedList<T> implements List<T> {
    private final List<T> delegate = new ArrayList<T>();
    …
}
```

- Co když budu potřebovat dvě různé instance, jednu založenou na `ArrayList` a druhou na `LinkedList`?

PROFINIT

Coupling

```
public class LimitedList<T> implements List<T> {
    private final List<T> delegate;

    public LimitedList(List<T> backingList) {
        this.delegate = backingList;
    }
    …
}
```

```
List<Object> myList = new LimitedList<>(new ArrayList<>());
List<Object> myList2 = new LimitedList<>(new LinkedList<>());
```

PROFINIT

Loose coupling

- Cíl: Modul (třída, …) má co nejméně záviset na svém okolí
- Výhody:
  - Odolnost proti změnám okolí
  - Snazší pochopení
  - Znovupoužitelnost
  - Testovatelnost
- Android `Intents`
- Pozor na skryté vazby
  - Časové
  - Souslednost událostí

```
/**
 * ...
 * Pozor! Tato metoda musí být volána až po metodě …
 */
```

PROFINIT

Tight Coupling v praxi

Konfigurace.java:

```
public HashMap getTypZajisteni() {
    if (!typyZajisteniRead) {
        TypZajisteniFactory.readKonfiguraceAll(this);
    }
    return typyZajisteni;
}
```

TypZajisteniFactory.java:

```
protected static void readKonfiguraceAll(Konfigurace konfigurace) {
    HashMap list = …
    konfigurace.setTypZajisteni(list);
    konfigurace.typyZajisteniRead = true;
}
```

PROFINIT

Encapsulation

- Cíl:
  - Znemožnit okolí záviset na mých implementačních detailech
  - Ochránit vnitřní stav před vnějšími zásahy
- Výhody:
  - Omezení dopadu změn na okolí („Encapsulate what changes“)
  - Snazší testování a verifikace
  - Snazší debugging
- Interfaces
  - Ultimátní podoba zapouzdření – implementační detaily tu vůbec nejsou
  - Jasně definují kontrakt a oddělují ho od implementačních detailů
- Gettery a Settery

PROFINIT

Jak rozbít zapouzdření

```
public class GraphImpl implements Graph {
    public Node addNode(String name) {
        return new NodeImpl(name);
    }
}

/* package private */ class NodeImpl implements Node {
    NodeImpl(String name) {
        this.name = name;
    }
}
```

PROFINIT

Jak rozbít zapouzdření

```
public class GraphImpl implements Graph {
    public Node addNode(String name) {
        return new NodeImpl(name);
    }
}

/* package private */ class NodeImpl implements Node {
    NodeImpl(String name) {
        this.name = name;
    }
}

public class MyUnrelatedClass {
    …
    Node node = new NodeImpl(“blabla“);
}
```

PROFINIT

Jak rozbít zapouzdření

```
public class GraphImpl implements Graph {
    public Node addNode(String name) {
        return new NodeImpl(name);
    }
}

public                  class NodeImpl implements Node {
    NodeImpl(String name) {
        this.name = name;
    }
}

public class MyUnrelatedClass {
    …
    Node node = new NodeImpl(“blabla“);
}
```

class NodeImpl is not visible

PROFINIT

Jak rozbít zapouzdření

```
public class GraphImpl implements Graph {
    public Node addNode(String name) {
        return new NodeImpl(name, this);
    }
}

public class NodeImpl implements Node {
    NodeImpl(String name, Graph parentGraph) {
        this.name = name; this.parentGraph = parentGraph;
    }
}

public class MyUnrelatedClass {
    …
    Node node = new NodeImpl(“blabla“);
}
```

constructor is undefined

PROFINIT

Další dobré rady

- Premature optimalization is root of all evil
- Jak implementovat vlastní cache – rada první: nedělejte to
- Jak implementovat vlastní transakce – viz předchozí bod
- Jak implementovat vlastní lazy fetching, zámky, … – však víte
- Bezpečnost nelze do kódu přidat dodatečně
- Thread-safety nelze do kódu přidat dodatečně

PROFINIT

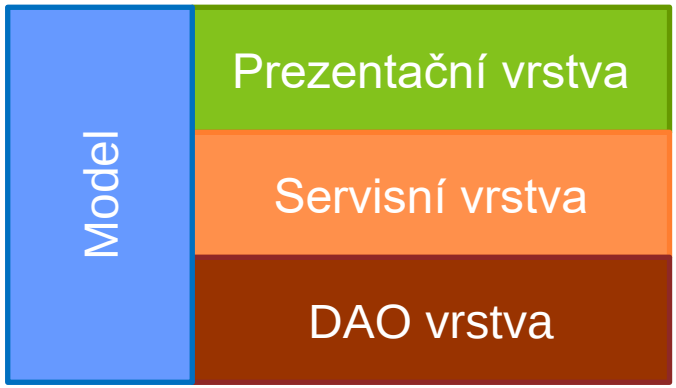
Thread-safety

- Immutable třídy jsou inherentně thread-safe
- Třídy bez vnitřního stavu jsou inherentně thread-safe
  - „Immutable once configured“
- Doporučuji všude, kde to jde:
  - Služby beze stavu (bez instančních proměnných kromě odkazů na jiné služby)
  - Stav předávat v parametrech metod a držet v lokálních proměnných
- Takto napsaný modul:
  - Nemá režii na synchronizaci (vhodný i pro aplikace nepotřebující thread-safety)
  - Není nezbytně thread-safe, ale půjde to snadno zařídit
  - Je čitelnější – je zřejmé, kudy tečou data

```
public Graph generateDataflow(Tree tree) {
    return handler.processNode(tree);
}
```

PROFINIT

Třívrstvá architektura



- Rozhraní vůči uživatelům
- Jádro – vlastní aplikační logika
- Komunikace s okolím

PROFINIT

Co zbylo z OOP?

- Encapsulation
  - Rozhodně
- Polymorphism
  - Ano, ale omezený často na interface inheritance
- Inheritance
  - Prefer composition over inheritance
  - „Implementation inheritance“ považována za nevhodnou

Z dokumentace java.sql.Timestamp (extends java.util.Date):

… As a result, the Timestamp.equals(Object) method is not symmetric with respect to the java.util.Date.equals(Object) method …

Due to the differences between the Timestamp class and the java.util.Date class mentioned above, it is recommended that code not view Timestamp values generically as an instance of java.util.Date. ***The inheritance relationship between Timestamp and java.util.Date really denotes implementation inheritance, and not type inheritance.***

PROFINIT

Design Patterns

- Model – View – Controller (MVC)

…

- Publish – Subscribe

…

- Iterator

PROFINIT

Zásady elementárního návrhu

DRY – Don’t Repeat Yourself

- **Controller:**

```
…
params.put("axis_length", axisLength);
…
```

- **Window:**

```
…
Integer axisLength = params.get("axis_length");
…
```

- **Dialog:**

```
…
if (params.get("axis_lenght") != null) {
    …
}
…
```

PROFINIT

SRP – Single Responsibility Principle

- Každý logický celek (modul, třída, metoda, …) má dělat vždy jen jednu věc
  - Co když budu potřebovat přepoužít jen jeden z několika kroků?
- Symptom: inline komentáře ve stylu: `// krok 2: teď uděláme …`

```
// Spocitej poradi, ve kterem je nutne parsovat DDL skripty
…

// Odstran existujici DDL skripty ve vystupnim adresari
…

// Pro kazdy databazovy objekt ve spocitanem poradi ziskej jeho
// DDL a zparsuj ho
for (DdlName ddl : extractionOrder) {
    …
    // Zparsuj DDL
    …

    // Uloz DDL
```

PROFINIT

Broken Windows

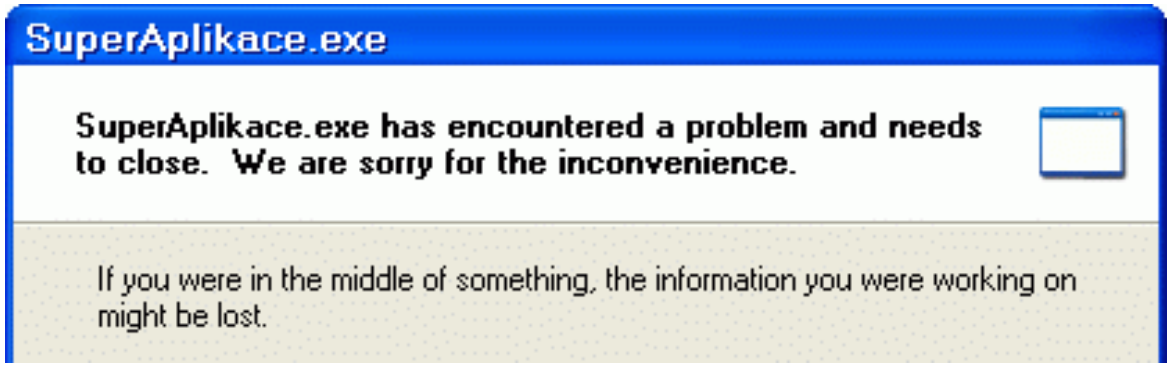
PROFINIT

Fail Fast

- Je-li v programu chyba, měl by selhat co nejdříve
- Aktivně kontrolujte konzistenci a ošetřujte možné chyby

- „Dead programs tell no lies“
- Nejhorší je na chybu přijít až po databázovém commitu

- Čím dřív chybu odchytíte, tím víc debugovacích informací můžete poskytnout
- Necháte-li chybu probublat až do obecného error handleru, můžete už říct jen:



PROFINIT

Konstrukce – zajímavá témata

Poznatky z praxe

- Programovat, když je jasné co (a jak)
  - viz vazba na LCA
  - některé části lze i předem (např. aspekty typu security, logování, výjimky, …)
- Ctít architekturu a návrh
  - jak zajistit dodržování návrhu?
  - jak zajistit dodržování základních pravidel?
    - automatizace kontrol, nástroje, …
- Používat vhodné nástroje a hotové věci
  - testování, IDE, CI, skripty, evidence, …
  - znalost frameworks a knihoven je zásadní → orientace, přehled

PROFINIT

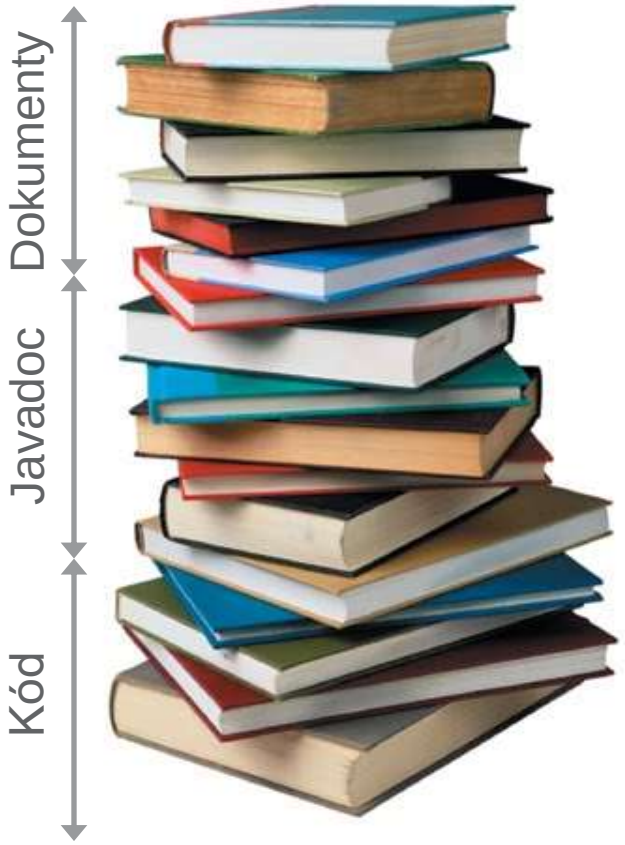
Test driven development

- Co to znamená?
  - Testy
  - Implementace
  - Testy
  - ...
- Jaká je podstata?
  - Testovatelný SW
  - Existence testů
- Pozor
  - Na zahlcení testy
  - Na údržbu testů

PROFINIT

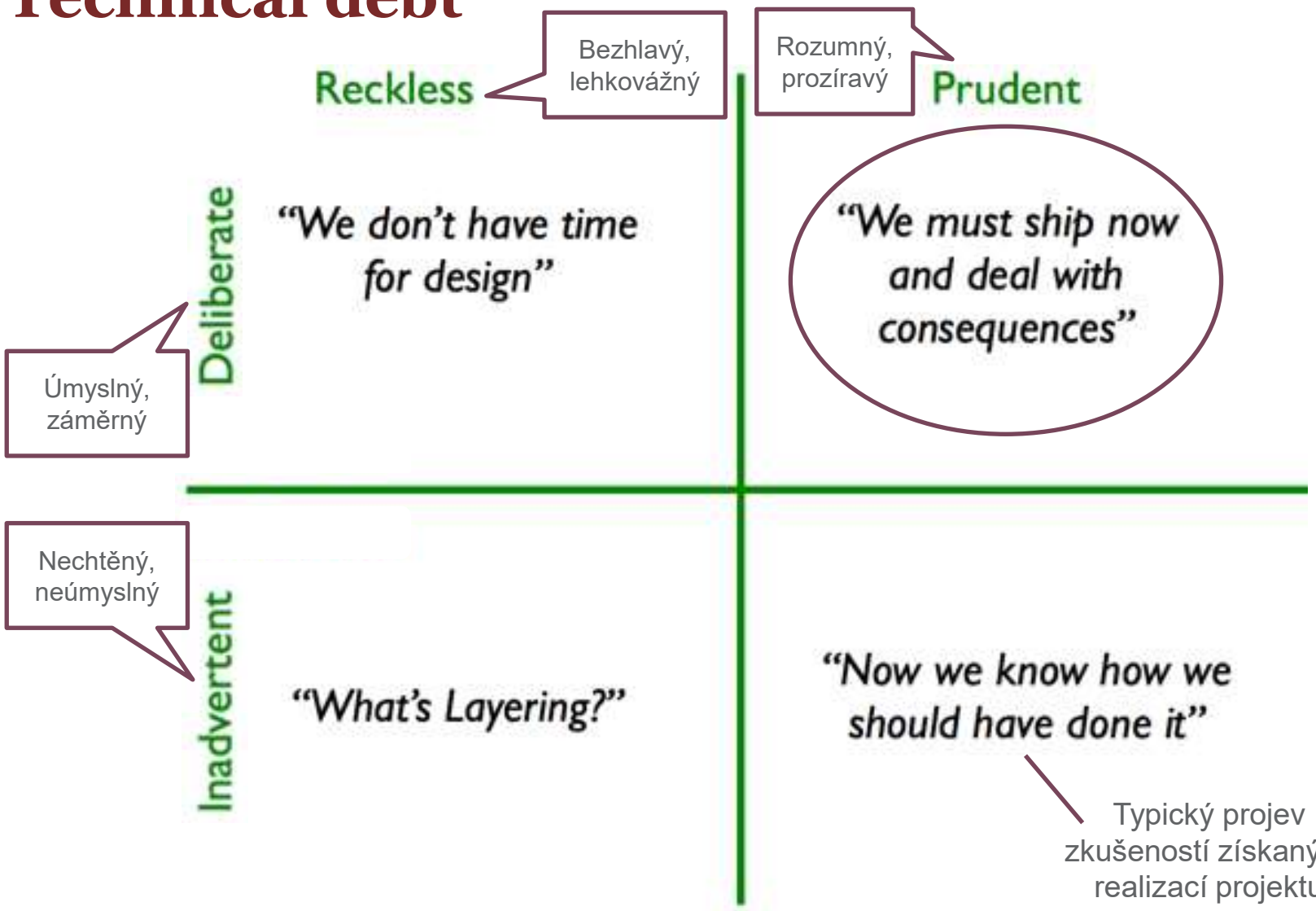
Self-documenting code

- Kód, který se svou formou (strukturou, jmennými konvencemi apod.) snaží omezit nutnost číst dokumentaci
- Neznamená úplnou absenci dokumentace
  - čitelný kód nenahradí koncepční dokumentaci (architektura, design moments, …)
- Problémy při chybějící dokumentaci
  - Význam větších funkčních celků
  - Pre/post conditions, invariants
  - Inheritance – kontrakt vůči potomkům



PROFINIT

Technical debt



Více informací na http://martinfowler.com/bliki/TechnicalDebtQuadrant.html

PROFINIT

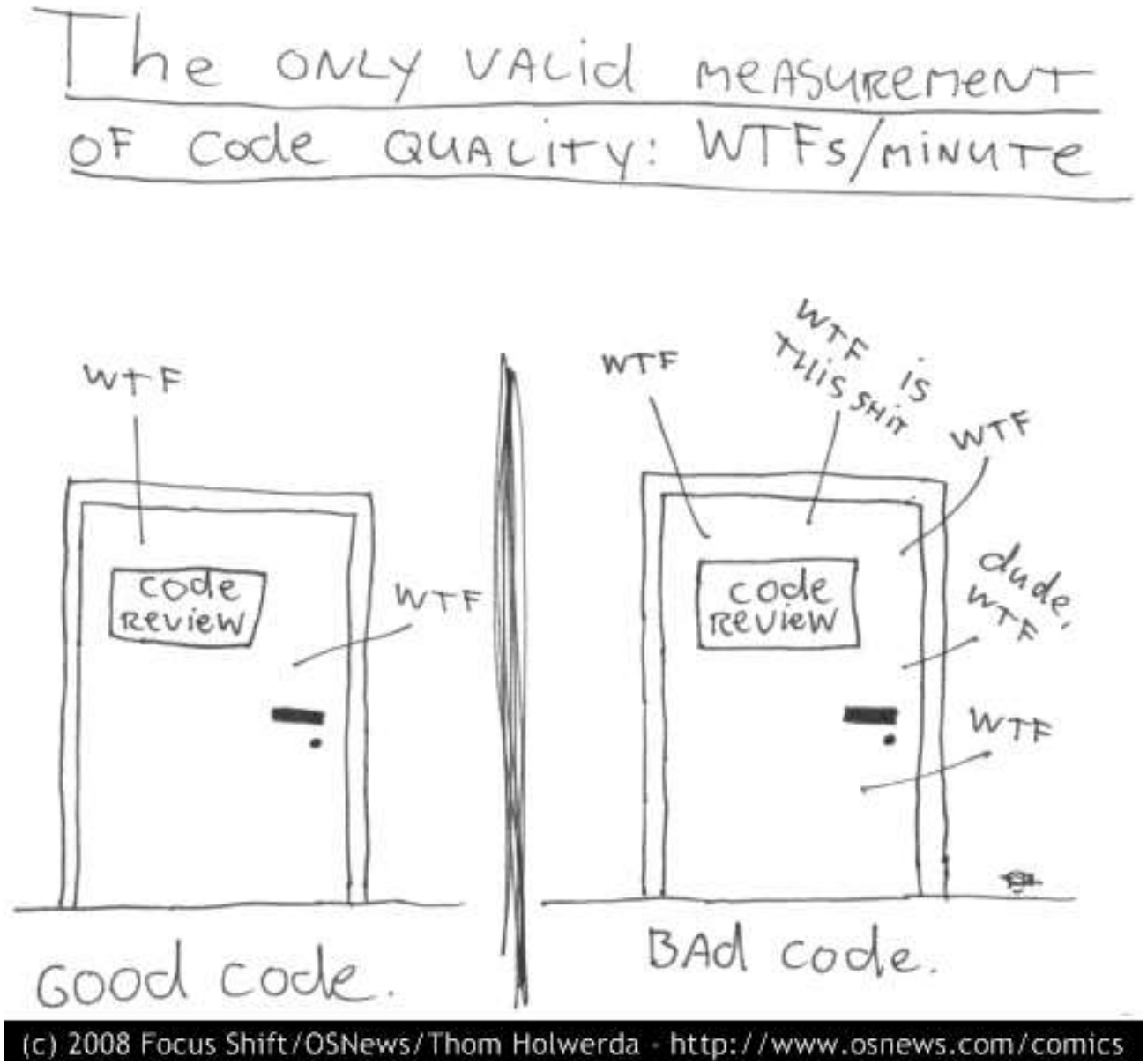
Co je to kvalitní kód?

Kvalitní kód je…

- Bezchybný
- Rychlý / Efektivní
- Krátký?
- Rafinovaný?
- **Srozumitelný**
- **Udržovatelný**

PROFINIT

PROFINIT

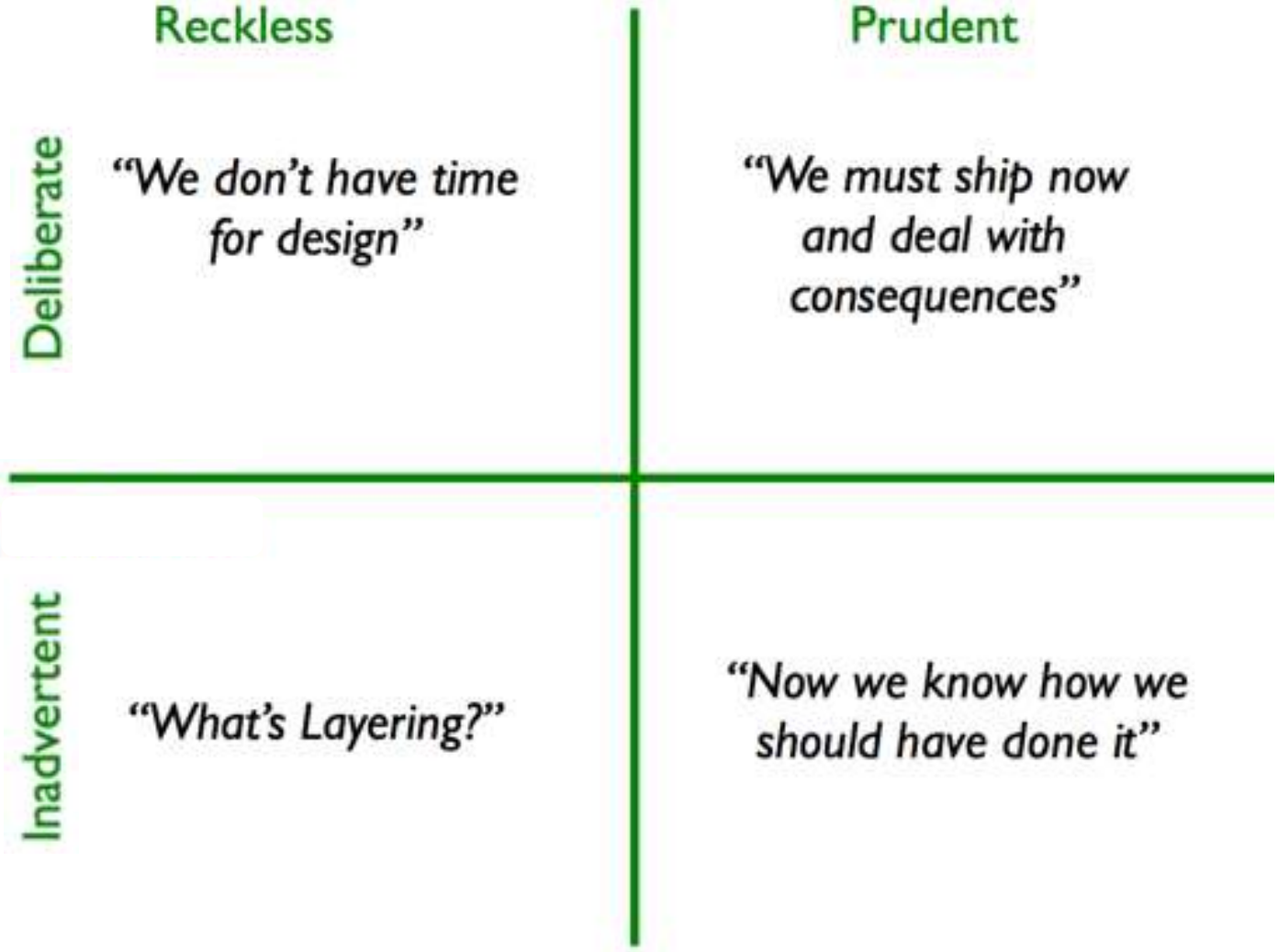

The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE
WTF
code review
WTF
Good code.
WTF
WTF is THIS SHIT
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics

Proč psát srozumitelný kód

- Kód píšete jednou, číst ho (vy nebo někdo jiný) budete pravděpodobně vícekrát
- Pokud váš kód někdo nepochopí, snadno v jeho použití nebo úpravě udělá chybu
- Nesrozumitelný kód nikdo nebude chtít používat

PROFINIT

Kdy nepsat kvalitní kód?



PROFINIT

YAGNI a KISS

- YAGNI – You Ain’t Gonna Need It
- KISS – Keep It Simple Stupid

PROFINIT

Jak psát srozumitelný kód

Co bude uživatel vaší metody číst

1. Jméno metody
2. Deklarované typy parametrů a návratové hodnoty
3. Jména deklarovaných parametrů
4. Javadoc
5. Samotný kód metody

› Čím dříve se dozví vše, co potřeboval, tím dříve může přestat číst (a ušetřit čas)

PROFINIT

Příklad: StringUtils

`join()`

PROFINIT

Příklad: StringUtils

```
public static String join(Object[], String, int, int)
```

PROFINIT

Příklad: StringUtils

```
public static String join(Object[] array, String separator, int startIndex, int endIndex)
```

PROFINIT

Příklad: StringUtils

```
/**
Joins the elements of the provided array into a single String containing the provided list of elements.

No delimiter is added before or after the list. A null separator is the same as an empty String (""). Null
objects or empty strings within the array are represented by empty strings.

 StringUtils.join(null, *)               = null
 StringUtils.join([], *)                 = ""
 StringUtils.join([null], *)             = ""
 StringUtils.join(["a", "b", "c"], "--")  = "a--b--c"
 StringUtils.join(["a", "b", "c"], null)  = "abc"
 StringUtils.join(["a", "b", "c"], "")    = "abc"
 StringUtils.join([null, "", "a"], ',')   = ",,a"

Parameters:
    array the array of values to join together, may be null
    separator the separator character to use, null treated as ""
    startIndex the first index to start joining from. It is an error to pass in an end index past the end of the
        array
    endIndex the index to stop joining from (exclusive). It is an error to pass in an end index past the end of
        the array
Returns:
    the joined String, null if null array input
*/
public static String join(Object[] array, String separator, int startIndex, int endIndex)
```

PROFINIT

Příklad: StringUtils

```
/**
Joins the elements of the provided array into a single String containing the provided list of elements.

No delimiter is added before or after the list. A null separator is the same as an empty String (""). Null
objects or empty strings within the array are represented by empty strings.

 StringUtils.join(null, *)               = null
 StringUtils.join([], *)                 = ""
 StringUtils.join([null], *)             = ""
 StringUtils.join(["a", "b", "c"], "--")  = "a--b--c"
 StringUtils.join(["a", "b", "c"], null)  = "abc"
 StringUtils.join(["a", "b", "c"], "")    = "abc"
 StringUtils.join([null, "", "a"], ',')   = ",,a"

Parameters:
    array the array of values to join together, may be null
    separator the separator character to use, null treated as ""
    startIndex the first index to start joining from. It is an error to pass in an end index past the end of the
        array
    endIndex the index to stop joining from (exclusive). It is an error to pass in an end index past the end of
        the array
Returns:
    the joined String, null if null array input
*/
public static String join(Object[] array, String separator, int startIndex, int endIndex)
```

PROFINIT

Příklad: StringUtils

```
public static String join(Object[] array, char separator, int startIndex, int endIndex) {
    if (array == null) {
        return null;
    }
    int bufSize = (endIndex - startIndex);
    if (bufSize <= 0) {
        return EMPTY;
    }

    bufSize *= ((array[startIndex] == null ? 16 : array[startIndex].toString().length()) + 1);
    StrBuilder buf = new StrBuilder(bufSize);

    for (int i = startIndex; i < endIndex; i++) {
        if (i > startIndex) {
            buf.append(separator);
        }
        if (array[i] != null) {
            buf.append(array[i]);
        }
    }
    return buf.toString();
}
```

PROFINIT

Název metody

- To první a často jediné, co z vaší metody bude většina lidí číst – věnujte mu pozornost
- Měl by být stručný a přitom dostatečně popisovat, co metoda dělá
- Pokud se vám nedaří vhodný název vymyslet, může to být špatným návrhem metody
  - Metoda dělá více různých věcí (měla by dělat vždy jen jeden svůj úkol)
  - Nemáte vlastně jasno, co má metoda dělat
  - …
- Pokud jste při vymýšlení jména přišli na špatný návrh metody, ušetřili jste si dost času, který byste jinak strávili později přepisováním kódu

- Kvíz: co dělá tato metoda?
  `dontStoreDataToCache()`

PROFINIT

Název metody

```
isSaveBasicDataAndTemplateAttributeValuesDeactivateDocumentReturnPossible()
```

PROFINIT

Jména a typy parametrů

- Jméno by mělo popisovat, co daný objekt reprezentuje – nejen sám o sobě, ale i v kontextu použití parametru:

```
void merge(Graph graph1, Graph graph2)
```
vs.
```
void merge(Graph inputGraph, Graph outputGraph)
```

- U typovaných jazyků využívejte datové typy
  - Toto by měla být naprostá samozřejmost
  - Speciálně v Javě se ale často setkávám zejména s ignorováním enums (místo enumu se často používají booleany nebo soustavy konstant)
  - Zdá se, že velmi populární je místo „strongly typed“ používat „stringly typed“ přístup:
    - Text: `"abc"`
    - Číslo: `"42"`
    - Objekt obsahující jednu textovou a dvě číselné položky: `"abc_42_666"`
    - Přístup k první číselné položce objektu: `myObjectString.split("_")[1]`

PROFINIT

Co nepovažuji za užití datových typů

- Map<Vertex, Set<Boolean>>
- List<List<List<String>>>
- Map<List<Object>, HashSet<Pair<Integer, Integer>>>

PROFINIT

Parametry

- Metoda s příliš velkým počtem parametrů je nepřehledná

```
protected PdfPTable getPrijemcePlneniTable(float documentWidth,
   String nazevLeasingovky, String cisloLeasingoveSmlouvy, String
   smluvniServis, String prijemceCisloUctu, String prijemceJmeno,
   String prijemceUliceCP, String prijemceObecPSC, boolean
   pojisteniJindeExistuje, String pojisteniJindeNazev, Boolean
   dphAnoNeNull, String zpusobUhradyKey, String datum, boolean
   isDPH, String sTextPodPodpisem)
```

- Parametry typu boolean nejsou příliš vhodné
  - Nutno v názvu parametru jasně deklarovat, co znamená true a co false
    - Jméno parametru ale nemusí být vždy dostupné (např. v Javě, mám-li knihovnu bez javadoc a zkompilovanou bez jmen parametrů, vidím jen typ bez jména)
  - Při volání metody není na první pohled význam boolean parametru obvykle zřejmý
    - `newTable(tableName, parentDatabase, true)`
    - `newTable(tableName, parentDatabase, TableType.GLOBAL_TEMPORARY)`

PROFINIT

Javadoc

- Dohodněte se na jazyku a dodržujte ho
  - V angličtině může být problém se vyjádřit (zejména odborné pojmy)
  - Komentáře v češtině vyžadují neustále přepínat klávesnici a jazyk „v hlavě“
  - Komentáře v „cestine“ nevypadají dobře
- Dokumentujte chování vůči null (všude tam, kde to není očividné)
  - U atributů objektu / třídy – zda může někdy být null a co to znamená
  - U parametrů metod – zda může být null a co se v tom případě stane
  - U návratových hodnot – zda a v jaké situaci může metoda vrátit null
- U kolekcí a polí vnímejte rozdíl mezi prázdnou kolekcí a null
  - Většinou dává lepší smysl prázdná kolekce („seznam nalezených prvků je prázdný“) než null („seznam nalezených prvků neexistuje“)
  - Nenechte se vést leností nebo dojmem větší efektivity / úspory paměti při použití null (Java: `Collections.emptyList()`)

PROFINIT

Javadoc

```
/**
 * Metoda cislo zaznamu milestone v cache nebo -1
 * @param firstRow
 * @param lastRow
 * @param extraRecords
 * @return
 * @throws ExpiredDataException
 */
public CachedData getTimeMilestoneCachedData(..)
```

PROFINIT

Kód

- Dodržujte konvence daného jazyka (jména, formátování, …)
- Orientace na čtenáře – kód bude čten víckrát, píšete jej jednou
  - Lenost zde není dostatečným argumentem
- Jména proměnných (a všeho ostatního) volte dostatečně srozumitelná
  - Příklad z jedné revize:
    v cele třídě mnoho nevhodných názvů proměnných – „bi“, „is“, „bos“, „asr“, „s“, „m“, „baos“, „dos“ a můj favorit, pole s názvem „array“
  - Kvíz: co ošetřuje následující podmínka?

```
if(lv >= 0 && df >= 0 && this.hasRows() &&
    df < this.getRows().length) { … }
```

  - Všeho s mírou – obecně čím větší rozsah platnosti, tím důležitější je jasné jméno
  - Srozumitelnost != délka – např. význam proměnné „i“ je obecně známý
- Nepoužívejte jednu proměnnou ke dvěma účelům
- Ternární operátor je efektní, ale leckdy poněkud nečitelný
  - Kvíz: co vrátí následující výraz?

```
(0 < 1) ? 2 : 3 + 4
```

PROFINIT

A Roguelike in less than 512 Bytes

```
#include<stdlib.h>
#define F(n)for(j=0;j<n;j++)
#define r rand()
int main(){int x,s=46,n,i,j,z=77,l[z];char m[z*s],h[z];initscr();raw();F(z*s)j[
m]=35;F(s)for(j[l]=i=(r%4+3)*z+(n=r%17*z+r%s+z);n<=i;n+=z)for(x=n;x<=n+j/2;m[++
x]=s);F(9)l[j][m]=z,j[h]=2;m[*l]=64;*h=5;l[j][m]=62;F(z){x=n=l[i++,i%=9];if(i)!
i[h]||*l^(n+=r%3+r%3*z+~z)||--*h?0:abort();else{F(25)mvaddnstr(j,i,m+j*z,z);j=s
-getch();m[n+=j/3*z-j%3+153]^62||main();F(9)l[j+1]^n||--h[j+1]||n[m]--;}n[m]^s
||(m[l[i]=n]=x[m],x[m]=s);}}
```

This weighs in at a hefty 493 bytes of C source code. I call it the "Monster Caves".

Note that the game doesn't have enough bytes to remember (and print out) your kill total, so you'll have to do that yourself.

If you get stuck in a disconnected part of a level, try pressing some other keys... you may just be able to teleport out.

http://locklessinc.com/articles/512byte_roguelike/

PROFINIT

Programming by Coincidence

Programming by Coincidence

```
if (typZadosti == ZALOZENI_SMLOUVY) {
    zalozSmlouvu();
} else {
    zrusSmlouvu();
}
```

PROFINIT

Programming by Coincidence

```
if (typZadosti == ZALOZENI_SMLOUVY) {
    zalozSmlouvu();
} else if (typZadosti == ZRUSENI_SMLOUVY) {
    zrusSmlouvu();
}
```

PROFINIT

Programming by Coincidence

```
if (typZadosti == ZALOZENI_SMLOUVY) {
    zalozSmlouvu();
} else if (typZadosti == ZRUSENI_SMLOUVY) {
    zrusSmlouvu();
} else {
    throw new IllegalArgumentException(
                "Nepodporovany typ zadosti: " + typZadosti);
}
```

PROFINIT

Programming by Coincidence

```
if (typZadosti == ZALOZENI_SMLOUVY) {
    zalozSmlouvu();
} else if (typZadosti == ZRUSENI_SMLOUVY) {
    zrusSmlouvu();
}
// else do nothing
```

PROFINIT

Programming by Coincidence 2

```
int pocetCelychMesicuDoKonceKvartalu = 0;
switch(mesicVRoce%3) {
    case 0: pocetCelychMesicuDoKonceKvartalu = 0;break;
    case 1: pocetCelychMesicuDoKonceKvartalu = 2;break;
    case 2: pocetCelychMesicuDoKonceKvartalu = 1;break;
}
```

PROFINIT

Programming by Coincidence 2

```
int pocetCelychMesicuDoKonceKvartalu = 0;
switch(mesicVRoce%3) {
    case 0: pocetCelychMesicuDoKonceKvartalu = 0;break;
    case 1: pocetCelychMesicuDoKonceKvartalu = 2;break;
    case 2: pocetCelychMesicuDoKonceKvartalu = 1;break;
    default: throw new IllegalStateException(
                    "Nastala situace, ke ktere nemuze dojit");
}
```

PROFINIT

Je tento konstruktor prázdný záměrně?

```
protected MyObject() {

};
```

PROFINIT

Je tento konstruktor prázdný záměrně?

```
protected MyObject() {
    /* empty */
};
```

PROFINIT

Dřív to fungovalo...

```
entity.id = new EntityId(this.allEntities.size());
```

PROFINIT

Perličky

Kdy se zobrazí nabídka úvěrů?

```
/**
 * Method returns true if credit lines should be shown.
 *
 * @return true if credit lines should be shown otherwise false
 */
public boolean isShowCreditLines() {
    return
        // Showing depends of dynamic property and if user is in pilot mode (show only in pilot to
        // users in pilot mode if constant is true, if false, show to all)
        ((ebankingUser.getPilot() == null) || (!EbankingConstants.CREDIT_LINES_FUNCTIONS_PILOT) ||
(Boolean.TRUE.equals(ebankingUser.getPilot()) && EbankingConstants.CREDIT_LINES_FUNCTIONS_PILOT))
                // and (home tab is shown and there are no campaigns)
                && ((menuBean.isHomeTab() && !bcs.isShowCampaigns())
                // or ((we are on loans tab (TAB_LOANS) on specific menu item (MNU_LST_CRE_MORA_GET)
                // or on eshop (TAB_ESHOP)) and client has no approved application)
                || (((menuBean.isWantedTab(MenuNameS24.TAB_LOANS.name())
                // THU: tento kod je zakomentovan, protoze nefuguje nastavovani actualMenuItem spravne
                                /*&& (menuBean.getActualMenuItem() != null) &&
                                      menuBean.getActualMenuItem().name().toUpperCase().equals(
                                          MenuNameS24.MNU_LST_CRE_MORA_GET.name())*/
        )
                || menuBean.isWantedTab(MenuNameS24.TAB_ESHOP.name())) &&
                                                        bcs.getApprovedApplicationID() == null))
                // and there exist some credit lines
                && !isItemListEmpty();
}
```

PROFINIT

Map není nikdy dost

```
public interface IValidator {

   public void validate(
        Object[] objects,
        Map<Object, Object> globalParams,
        Map<Object, Object> localParams,
        Map<Object, Object> rowParams,
        Locale locale);

}
```

PROFINIT

DB procedura

```
...
**  Nazev ulozene procedury:  esipo_s_prehl_rotv
**
**  Funcionalita:  Procedura slouzi
**
**  Vstupni parametry:
...
```

PROFINIT

Trocha logiky

```
public void setPlatnostDoOdvolani(boolean platnostDoOdvolani) {
    if (platnostKontaktniAdresy == null) {
        platnostKontaktniAdresy = new IntervalOdDo();
    }
    else {
        platnostKontaktniAdresy = new IntervalOdDo(
            platnostKontaktniAdresy.getZacatek(), null);
    }
}
```

PROFINIT

Velmi defenzivní

```
String tmpLang = null;
if ((tmpLang == null) && ...) { ... }
```

PROFINIT

Čím se od sebe metody liší?

```
/**
 * Directory interface poskytovany modulum skrze WebService webovou
 * aplikaci datovych schranek.
 */
public interface IDSDirectoryService {

    public ListResponse<Issuer> getIssuer(
        IssuerSearchCriteria searchCriteria, String personId);

    public ListResponse<Issuer> findIssuer(
        IssuerSearchCriteria searchCriteria, String personId);
}
```

PROFINIT

Handle stuff

```
dataProcessor.processData(elementsOnPages);
```

PROFINIT

Goodies

Domain Specific Languages (DSL)

- Specifické jazykové konstrukce relevantní pro určitou doménu
- Limitovaný slovník a gramatika
- Čitelné a pochopitelné pro doménové experty
  - Většinou utopie
- **Internal DSL**
  - Tvoříme je v obecném programovacím jazyce
  - Lze dobře v určitých jazycích (Ruby, Groovy, …)
  - Řada patterns (expression builder, method chaining, …)
  - NotORM framework (PHP):

```
$employees = $db->employee()->select("name")
->where("dept_no = ?", "123")->order("salary", "desc")->limit(10)
```

- **External DSL**
  - Vytváříme je ve speciálním jazyce
  - Potřebujeme interpret, parser nebo překladač
  - Vhodné vytvořit vlastní gramatiku
    - Mnoho nástrojů – ANTLr, SableCC, …

PROFINIT

Templates, checklists, literatura

Materiály SWENG - Construction

ČLÁNKY

- "Best Practices" Columns by Steve McConnell

KNIHY

- Steve McConnell, Code Complete, 2nd Edition. Redmond, Wa.: Microsoft Press, 2004

CHECKLISTS

- CxCheck_Conditionals.txt
- CxCheck_ConstructingRoutine.txt
- CxCheck_ControlStructureIssues.txt
- CxCheck_DataCreation.txt
- CxCheck_DocumentingCode.txt
- CxCheck_HighQualityModules.txt
- CxCheck_HighQualityRoutines.txt
- CxCheck_Layout.txt
- CxCheck_Loops.txt
- CxCheck_NamingData.txt
- CxCheck_OrganizingCode.txt
- CxCheck_UnusualControlStructures.txt
- CxCheck_UsingData.txt

Všechny odkazované materiály jsou poskytnuty výhradně za účelem výuky softwarového inženýrství.

© Of Respective Parties 2007-2009

SWEBOK

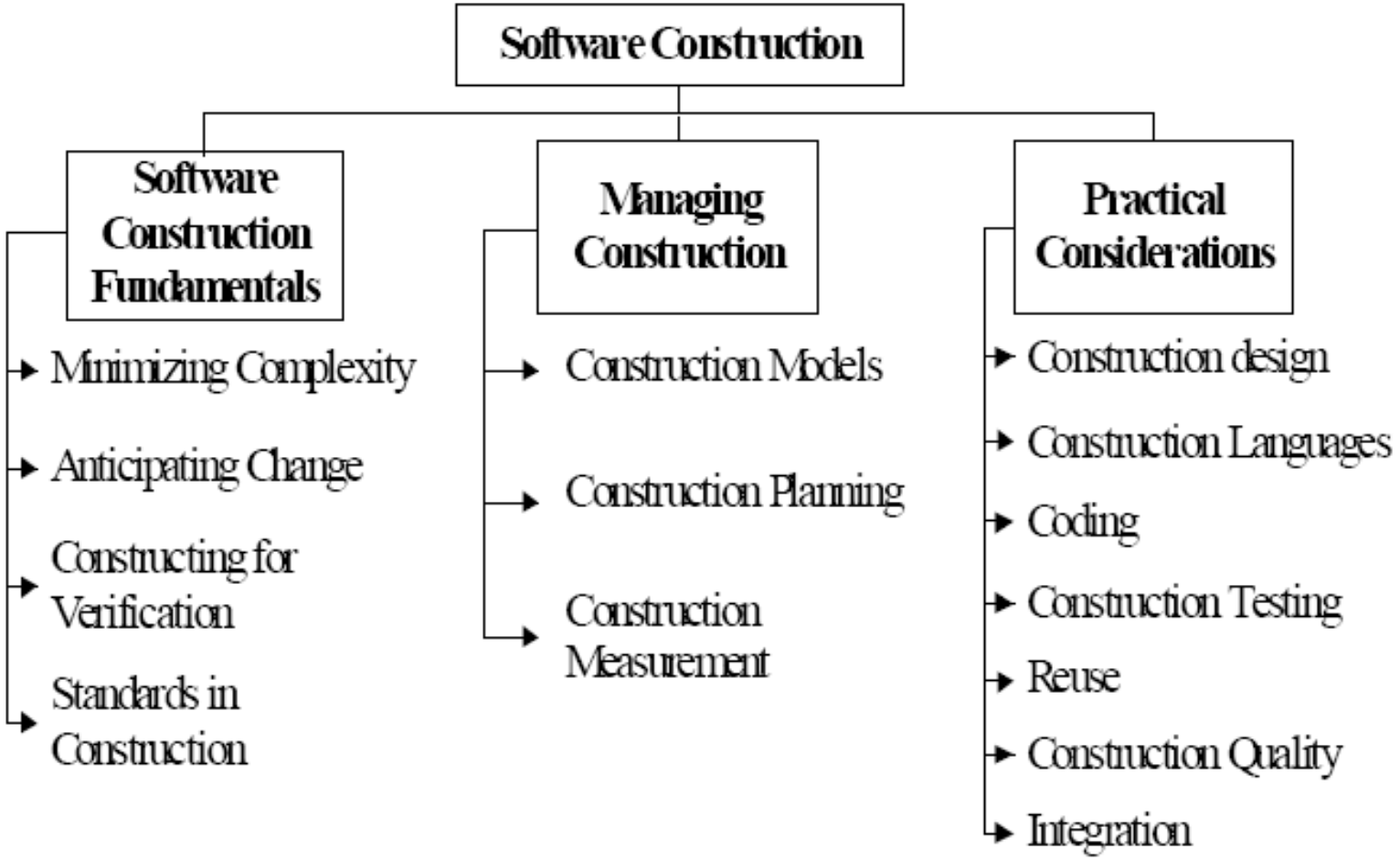


Figure 1. Breakdown of topics for the Software Construction KA.

PROFINIT

Zájem, vzdělávání, komunita

- http://www.javacodegeeks.com/
- http://www.infoq.com/
- http://www.theserverside.com/
- http://martinfowler.com/
- http://www.joelonsoftware.com/

… a mnoho dalších …

… o knihách ani nemluvě …

PROFINIT