Analýza požadavků na SW

Tvorba programových systémů Y36SI3

Ondřej Macek
macekond@fel.cvut.cz

Dnešní přednáška

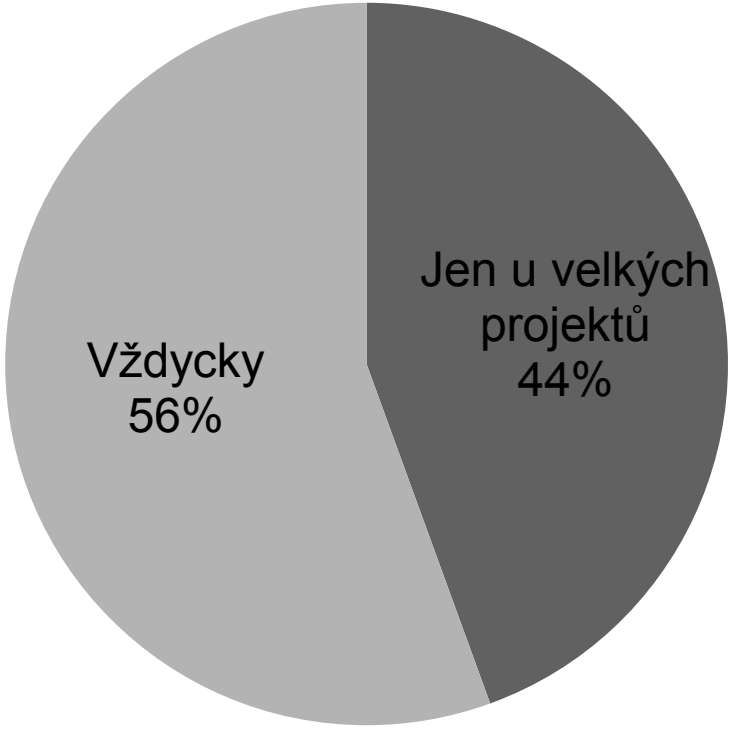
- Požadavky na SW
- Komunikace se zákazníkem
- DSL
- Prototypování

Co zákazník chce? To nikdo neví

SBĚR POŽADAVKŮ

Kdy analyzovat požadavky?

Podle ankety



Motivace pro sběr - Statistika

- Projekty NASA, které investovaly alespoň 10% prostředků do sběru požadavků měly ve výsledku nižší náklady a lépe plnili termíny

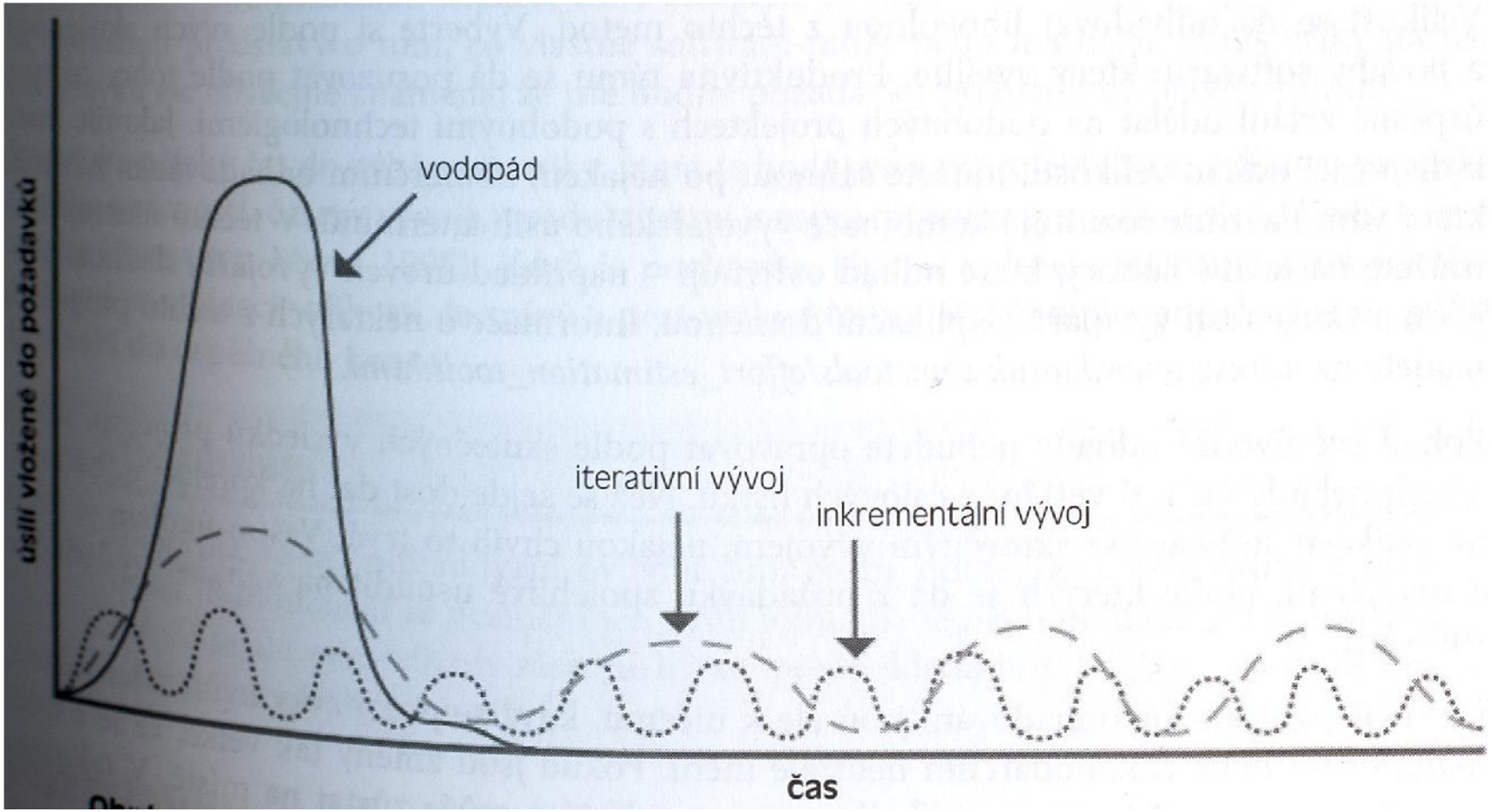
- Projekty v EU, které vyvíjely rychleji než ostatní sledované, investovaly do požadavků 14% zdrojů a 17% času

Motivace pro sběr požadavků

Nemá smysl být precizní, když ani nevíš o čem mluvíš

(von Neuman)

Požadavky a vývoj SW



Požadavky jsou v nějaké doméně

- Kontextový model

- Doménový model

Datový slovník – mluvit stejným jazykem

- Vztah mezi zákazníkem a analytikem

- Např. DSL

Koho se ptát?

Uživatel vs. Zákazník

Všichni vs. Produktový šampion

Jak se ptát?

- Workshop/Interview
- Sledování uživatelů
- Stávající dokumentace
- Dotazníky
- Šampion přímo v týmu (agilní přístup)

Na co se ptát?

- Podnikatelské požadavky
- Podnikatelská pravidla (business rules)
- Funkční požadavky
- Nefunkční požadavky
- Případy a scénáře užití
- Kvalitativní parametry
- Požadavky na rozhraní
- Omezení
- Definice dat

Obecné požadavky

Zákaznické

- Dostupnost
- Efektivita
- Flexibilita
- Integrita
- Kompatibilita
- Spolehlivost
- Odolnost
- Použitelnost

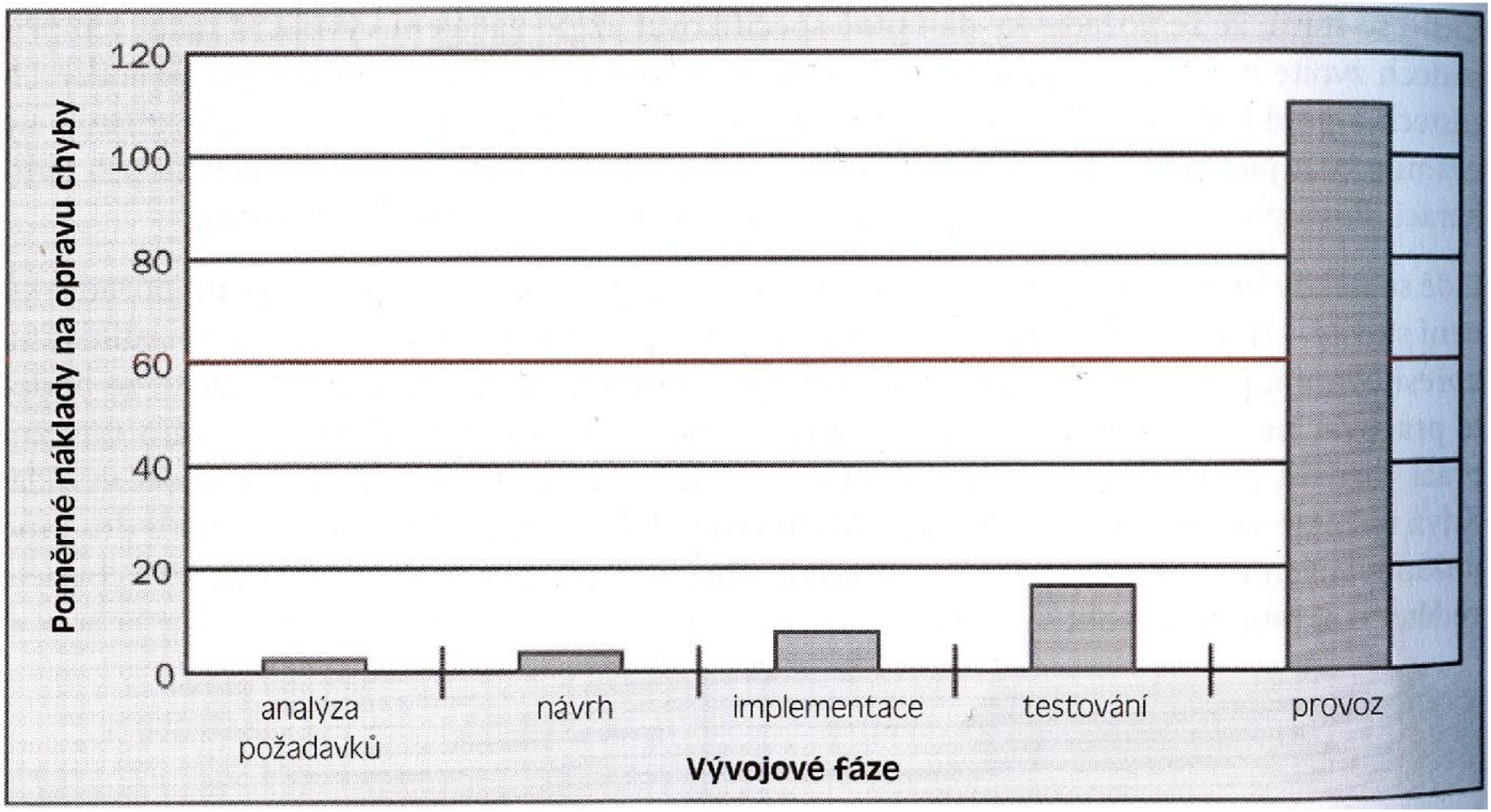
Programátorské

- Udržovatelnost
- Přenositelnost
- Znovupoužitelnost
- Testovatelnost

Vlastnosti dobrého požadavku

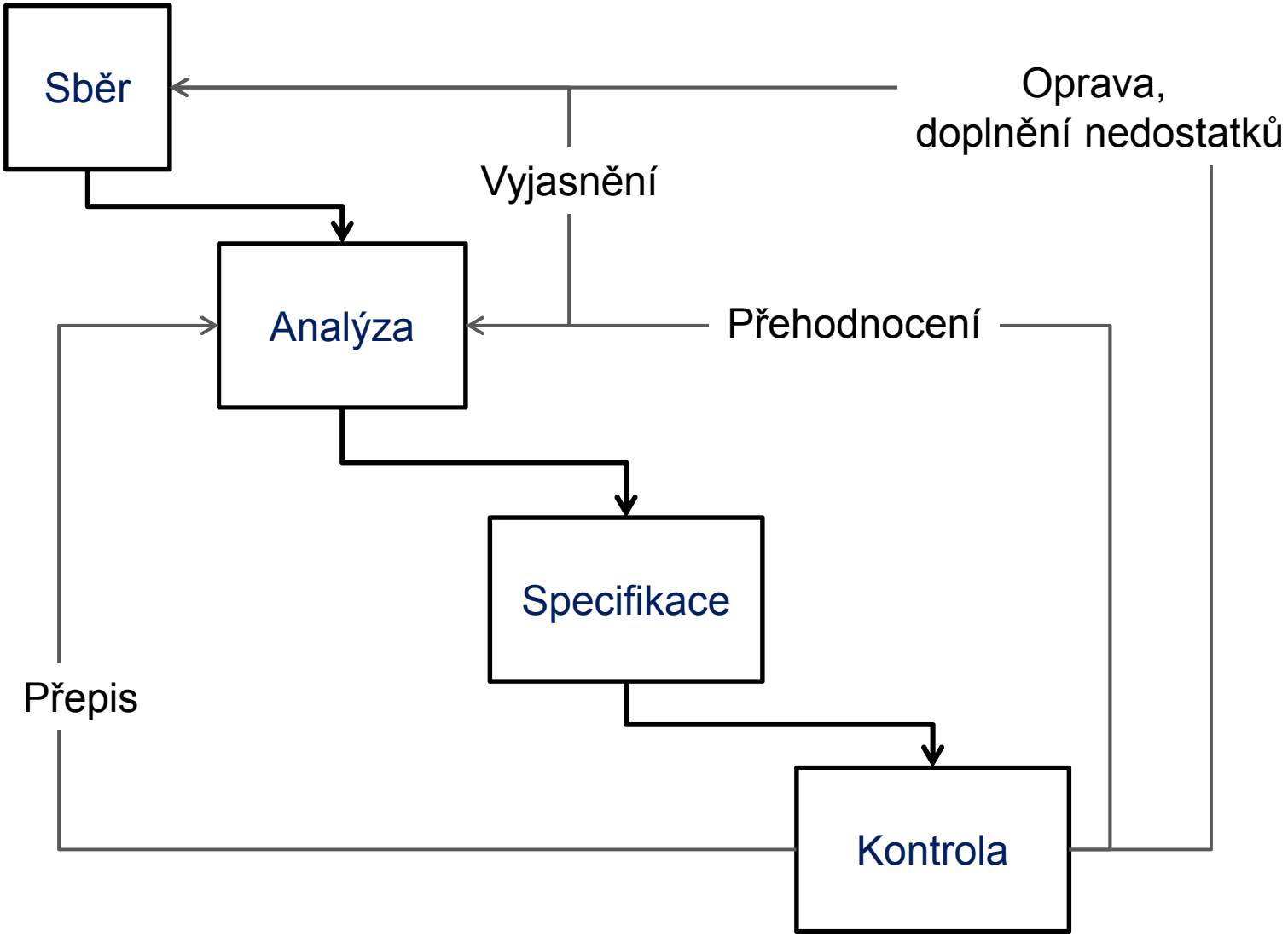
- Úplnost
- Jednoznačnost
- Správnost
- Priorita
  - must/should have, nice to have
- Proveditelnost
- Ověřitelnost
  - akceptační test

Poměrné náklady na opravu chyby podle jejího odhalení



Pozn.: chybou může být i špatná počáteční specifikace

Životní cyklus požadavku



Model požadavků

- (Hierarchické) číslování
- Dělení do skupin
- Jednotná šablona

Nešvary (sběru) požadavků

- Nedostatečné zapojení nebo přehlédnutí uživatelů
- Nabobtnávání požadavků
- Nejednoznačné požadavky
- Šperkování
- Malá specifikace
- Definují se jen uživatelské požadavky
- Ignorují se právní normy/standardy pro danou oblast

Příklady požadavků

- Generování matice zodpovědnosti

- Systém bude umožňovat generování RACI matic podle kritérií zadaných uživatelem

- Přihlášení správce se provede vložením servisní karty. Dojde k uzamčení uživatelské části terminálu po odhlášení případného aktuálního uživatele a odeslání informace o přihlášení správce na dispečink. Technikovi se následně zobrazí nabídka dostupných akcí.

Opakování SIN

ANALYTICKÉ MODELY

Analytické Modely

- Diagram datových toků
- ER diagram
- Stavové diagram
- Model dialogů
- Use Case model + texty

Dříve zmíněné

- Doménový model
- Kontextový diagram

Kolik je potřeba (typů) modelů?

4+1

PROTOTYPOVÁNÍ A DSL

Prototypování a DSL - motivace

Problém:

- Zákazník nerozumí modelům
- Ne vše jde modelem přesně zachytit
- Společná řeč se zákazníkem

Řešení

- Prototypování
- DSL

Omezit se na doménu

DOMAIN SPECIFIC LANGUAGES (DSL)

Domain Specific Languages

- Language Oriented Programming

- **Doménově** specifický jazyk na vysoké úrovni **abstrakce**
  - abstraktnější než Java, C#

    (tzv. general purpose languages)
  - konkrétnější než UML

    (tzv. general modeling language)

Příklady DSL

- HTML
- OCL
- QVT
- Microsoft XAML
- Apache httpd.conf

Příklad: Validace kreditních karet

```
<validation>
    <validate country="CZE">
        <accept />
    </validate>
    <validate country="USA">
        <securitycode match="exact" />
    </validate>
    <validate country="TKL">
        <fail />
    </validate>
    <validate country="*">
        <securitycode match="exact" />
        <addressmatch match="matchZipStreetNumber" />
    </validate>
</validation>
```

Příklad: XAML

```
<Window ….>
    <StackPanel>

       <Button Content="Drop Shadow Under Me" Width="200" Margin="10">

            <Button.Effect>
                <BlurEffect Radius="10" />
            </Button.Effect>

        </Button>

    </StackPanel>
</Window>
```

Realizace DSL

- Interpreter DSL
  - Vlastní parser DSL
  - DSL je přímo spustitelné
- Kompilátor DSL
  - DSL převedeno do nějakého konkrétního jazyka
  - Generování kódu

Výhody DSL

- Popis způsobem blízkým doméně
- Snadný popis problému v doméně
- Validace už na doménové úrovni
- Kvalita, produktivita, spolehlivost, udržovatelnost, znovupoužitelnost

Nevýhody DSL

- Nutnost naučit se DSL
- Nutnost udržovat DSL a potřebné prostředí
- Možná ztráta výkonu oproti general purpose languages
- Rychlý nárůst jazyků

Analýza na živo

PROTOTYPOVÁNÍ

Prototypování

- Předvádění prototypů aplikace zákazníkům
- Rychlá aktualizace prototypu podle nových požadavků
- Zahazování nepotřebných/špatných prototypů

Přístupy k prototypování

- Rapid prototyping
  - velké množství malých prototypů – mnohé z nich zahozeny
- Evolutionary prototyping
  - vytvoří se masivní základ, který se postupně upravuje
- Incremental protyping
  - víc prototypů, které se později spojí

Prototypování – klady a zápory

+	-
Jasný dorozumívací prostředek	Nedostatečná analýza/návrh
Snižuje náklady a čas	Fixace na prototyp
Zákazník je zapojen do vývoje	Neodhadnutí ceny prototypů

Zajímavá literatura

Wiegers Karl E. ***POŽADAVKY NA SOFTWARE OD ZADÁNÍ K ARCHITEKTUŘE APLIKACE***. Cpress. Brno. 2008

DOMAIN-SPECIFIC LANGUAGE. Wikipedia, the free encyklopedia [online]
http://en.wikipedia.org/wiki/Domain-specific_programming_language [cit . 2010-10-14]

HAAN Johan. ***DSL DEVELOPMENT: 7 RECOMMENDATIONS FOR DOMAIN SPECIFIC LANGUAGE DESIGN BASED ON DOMAIN-DRIVEN DESIGN*** [online]
http://www.theenterprisearchitect.eu/archive/2009/05/06/dsl-development-7-recommendations-for-domain-specific-language-design-based-on-domain-driven-design
[cit . 2010-10-14]

4+1 ARCHITECTURAL VIEW MODEL. Wikipedia, the free encyklopedia [online]
http://en.wikipedia.org/wiki/4%2B1_Architectural_View_Model [cit . 2010-10-14]