

# **Definování záměru a rozsahu produktu**

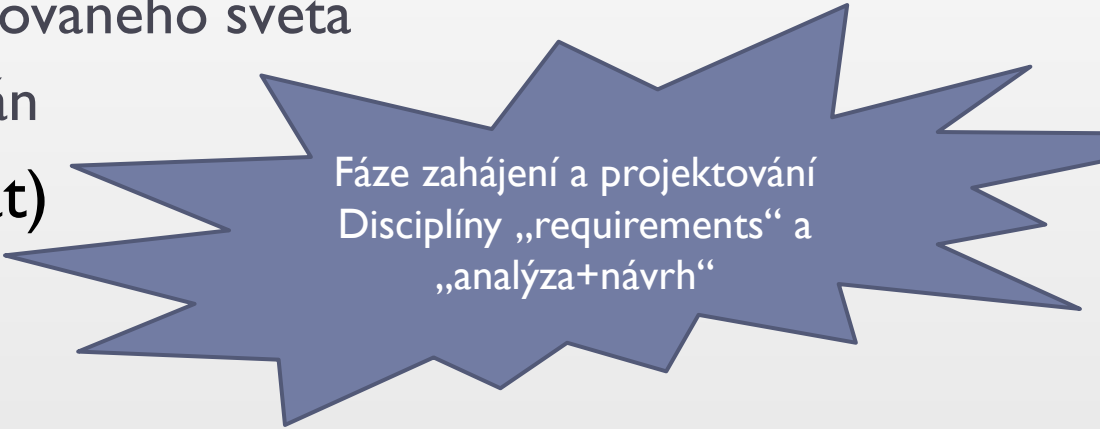
KIV/ASWI 2017/2018

# Požadavky obecně

# ► Požadavky: co, proč, kdy

---

- Sběr požadavků, analýza: co se chce
  - pochopení problému
  - model reálného/projektovaného světa
  - podklady – realizace, plán
- (Návrh: jak to realizovat)



Fáze zahájení a projektování  
Disciplíny „requirements“ a  
„analýza+návrh“

- Úrovně detailu
  - zahájení projektu: klíčové, obrysy
  - projektování: podstatné, úplnost
  - konstrukce: podrobnosti

## ► (software requirements)

---

- Požadavek = schopnost nebo vlastnost, kterou má software mít, aby jej uživatel mohl používat k vyřešení problému nebo dosažení cíle, který vedl k zadání, nebo aby splnil podmínky stanovené smlouvou, standardem, nebo jinou specifikací.

[dle IEEE Std 610.12-1990]

- nepotřebuje-li uživatel X, není X požadavkem
  - požadavky jsou omezeny vnějšími podmínkami
- 
- Účel specifikace požadavků:
    - popsat zadání tak, aby se z toho dalo vycházet pro implementaci =>
    - srozumitelnost pro zákazníka, jednoznačnost+struktura pro vývojáře



# ► Typy požadavků

---

- Business reqts
  - Vize a rozsah projektu
- User (funkční) reqts
- Business rules, Constraints
- Extra-functional
  - vlastnosti
- System reqts
- Contractual, legal, ...

Příklad:  
business rules

# ► Kategorie požadavků (I)

---

- Klasifikace funkčnosti do čtyř kategorií
  - jaké **informace** systém obsahuje, udržuje
  - jaké **funkce** poskytuje uživatelům
  - jaké **analýzy** dat provádí
  - jaké jsou **interakce** s jinými systémy
- Pro každý druh příslušné vlastnosti
  - stačí jednoduché seznamy
- + not this time (až příště)
  - **mimo rozsah** zadání
  - doplňková funkčnost

## ► Kategorie požadavků (2)

---

The basic issues that the SRS writer(s) shall address are the following:

- a) *Functionality*. What is the software supposed to do?
- b) *External interfaces*. How does the software interact with people, the system's hardware, other hardware, and other software?
- c) *Performance*. What is the speed, availability, response time, recovery time of various software functions, etc.?
- d) *Attributes*. What are the portability, correctness, maintainability, security, etc. considerations?
- e) *Design constraints imposed on an implementation*. Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

## ► IEEE Std 830-1998

## ► Lidé v analýze

---

### ► Zákazník

- externí, interní
- doménový expert

### ► Zainteresovaný hráč (stakeholder)

- ředitel / investor / standardizační orgán / daňový poplatník
- vliv na úspěch projektu

### ► Analytik

- zprostředkovatel mezi zákazníkem a programátory





# ► Postup práce s požadavky

---

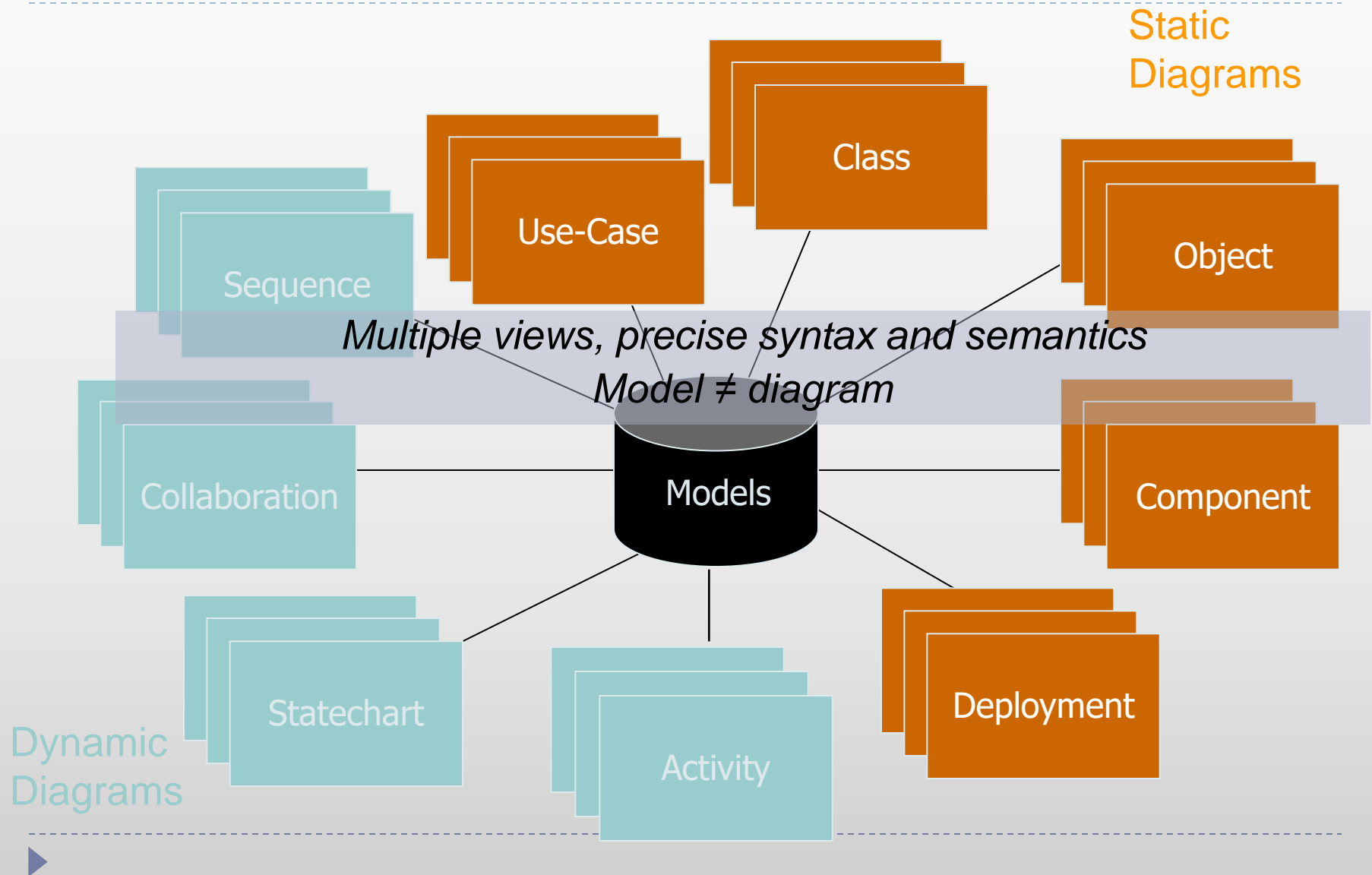
## ► Reqts **development**

- Elicit
- Analyze, Negotiate
  - potential => stable requirements
- Document
- Review
  - baselined requirements

## ► Reqts **management**

- Change management

# ► UML modely





# **Definice systému**



# ► Účel definice systému

---

- Základní, stručný popis účelu projektovaného systému
- Vyjádřit cíl projektu
  - soulad zákazník - dodavatel
  - zabránit divergování během vývoje
  - prevence nárůstu požadavků (*feature creep*)
- Etalon pro zhodnocení úspěšnosti projektu

# ► Stručný popis problému

---

## ► 25 slov / jeden odstavec

- k čemu má systém sloužit
- jaké informace bude udržovat
- kdo jej bude používat
- co přinese, čemu pomůže

## ► Šablona RUP

- problém ...
- postihuje ... [koho]
- což vede k ... [důsledky]
- řešení bude ... [cílový stav]

## ► Tisková zpráva

- jak si představujeme výsledné řešení

## ► Kontextový model

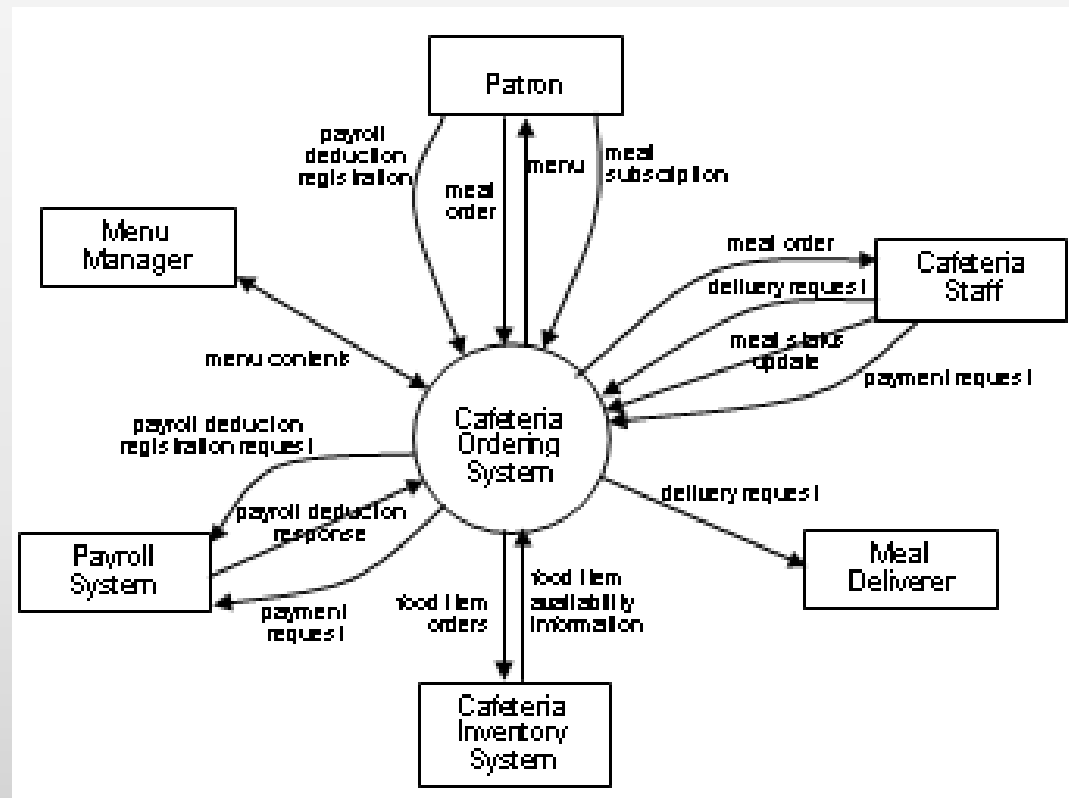
### ► Zobrazení vztahů systému s okolními entitami

- systém jako černá skříňka
- aktéři, stakeholders
- ostatní systémy

### ► Rozsah systému

### ► Rozhraní na okolí

- HCI, API, data



# ► Vize produktu a rozsah projektu

---

- Popis cíle, který má vzniknout
  - esenciální, klíčové, vysoko-úrovňové požadavky
  - „problem behind the problem“

Související artefakty:

- + Business case: *proč se to chce*
  - zdůvodnění výhodnosti-návratnosti projektu
- ☐ Požadavky: co to má *dělat*
- Též např. Concept of Operations, Definice systému, ...

Definice / příklad  
Business Case



# ► Vize produktu: kostra

---

## ► Popis **problému** a účelu

- smysl a účel cílového produktu
- obchodní příležitost, důvod ekonomické návratnosti

## ► Přehled **stakeholders**

- kdo jsou zájemci o systém, typy uživatelů
- (potenciální konkurence)

## ► Přehled očekávaných **schopností** a funkcí produktu

- popis, kvalitativní charakteristiky, priority
- stručný výčet bez detailů

## ► **Omezení, standardy, závislosti**

- vztahující se k projektu

## ► (Rámec **plánu** projektu)

- časový rozsah, plánované verze / vydání

Příklad: Wieggers,  
IEEE ConOps



# **Popis základních požadavků s UML**

(Definice systému s použitím UML)

# ► Přehled

---

- Popis požadavků na (vnější) **funkčnost** systému
  - Jací uživatelé k systému přistupují?
  - Co pro ně software dělá? Jak systém zpracovává požadavky?
  - *Model užití* = aktéři + případy užití + dokumentace
- Zachycení **informací**, které systém zpracovává
  - Data udržovaná systémem
  - Informace zadávané / poskytované
  - *Doménový model*
- Popis **prostředí**
  - Hardwarová stránka systému (fyzické řešení)
  - Nasazení softwarových částí do prostředí, jejich komunikace
  - *Model nasazení*

Společné: Kde je hranice systému (co je předmětem řešení)?

## ► Aktér

- Co to je: uživatel nebo jiný systém, který analyzovaný systém používá
  - typový uživatel, nachází se vně systému
  - spouští případy použití
  - primární / sekundární aktéři
- Popis aktéra
  - název: role, ne jména
  - jak a k čemu používá systém
  - seznam cílů
- Vazby mezi aktéry
  - generalizace – hierarchie rolí



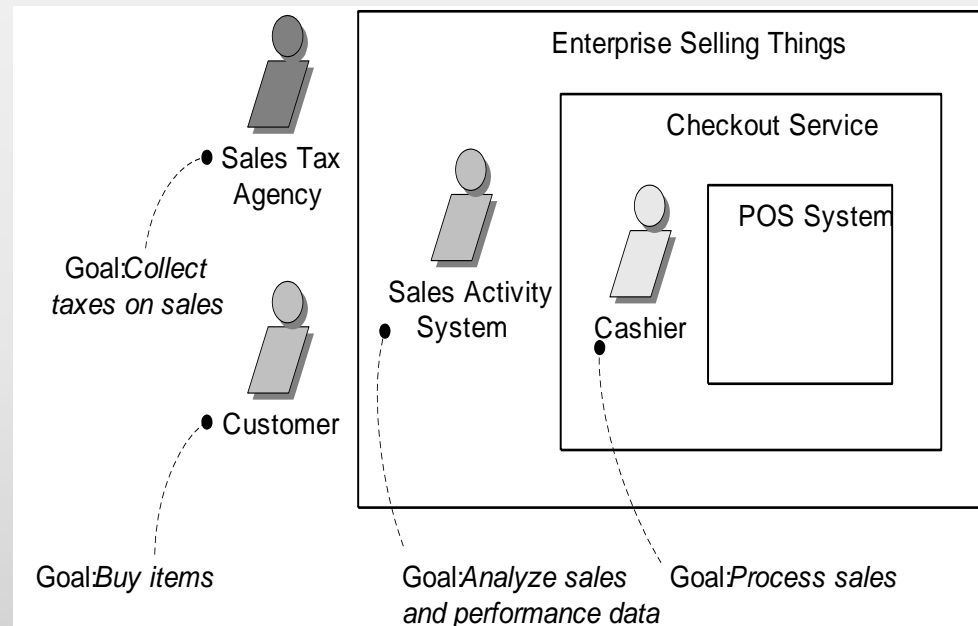
# ► Jak najít aktéry

## ► Poznají se tak, že

- odrážejí způsoby používání aplikace
- jsou podstatní pro určení hranice systému

## ► Pozor

- primární × sekundární
- vytrždit uvnitř systému a not this time
- některé budou nalezeny až později



## ► Funkcionalita: Případy užití

---

Co to je případ užití (PU):

sekvence akcí, které systém provádí  
v důsledku nějakého vnějšího podnětu  
a které vedou k výsledkům  
viditelným pro některého jeho uživatele.

- též “prototypová úloha” či „**elementární business proces**“
- poprvé Jacobson 1992 (OOSE)

# ► Jak najít případy užití

---

- Hledáme dialogy aktér–systém
- Začít od aktérů
  - popis problému z pohledu 1 aktéra
  - seznam cílů/potřeb aktéra => první diagram PU
  - “Jaké jsou hlavní akce, které provádí?”
  - “Jak vkládá / získává informace?”
  - “Potřebuje vědět o stavu systému?”
- Výsledek
  - seznam případů užití (názvy)

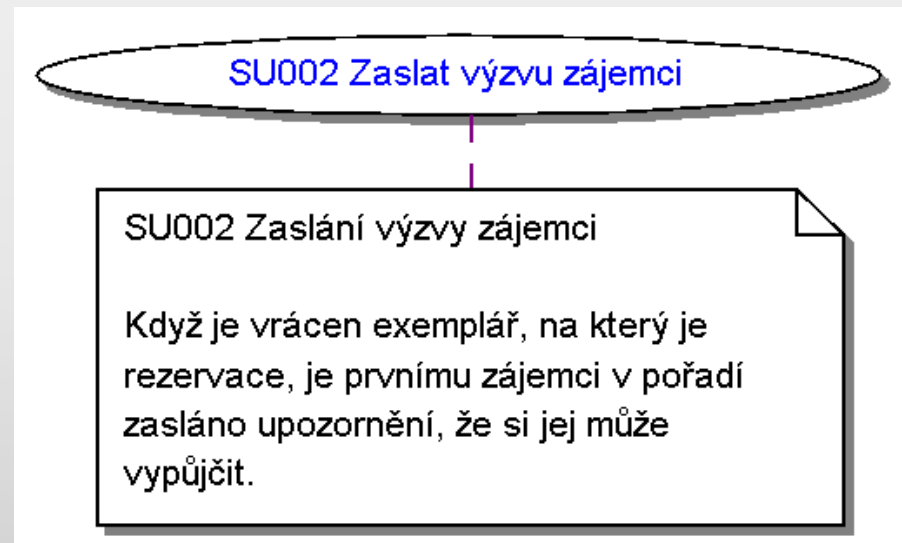
Viz OpenUP „Guideline: Identify and Outline Actors and Use Cases“



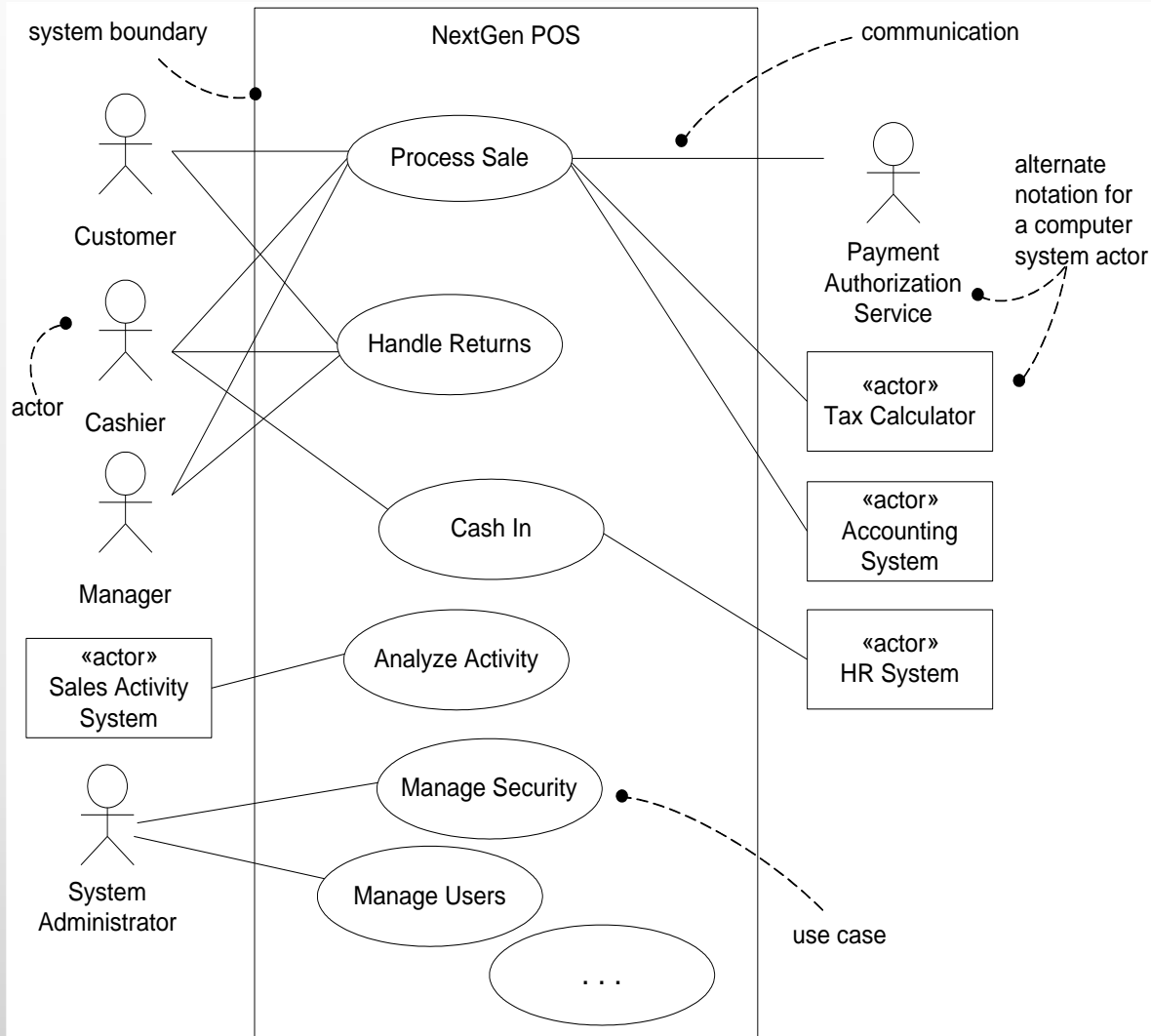
# ► Základní popis případu užití

... ve fázi shromažďování požadavků:  
základní popis dané funkce aplikace

- Název (+ ID)
- Stručný popis – 1 věta
- Případně
  - Základní kroky postupu pro klíčové PU
  - Odkazy na zdrojové informace



# ► UML – diagram případů užití

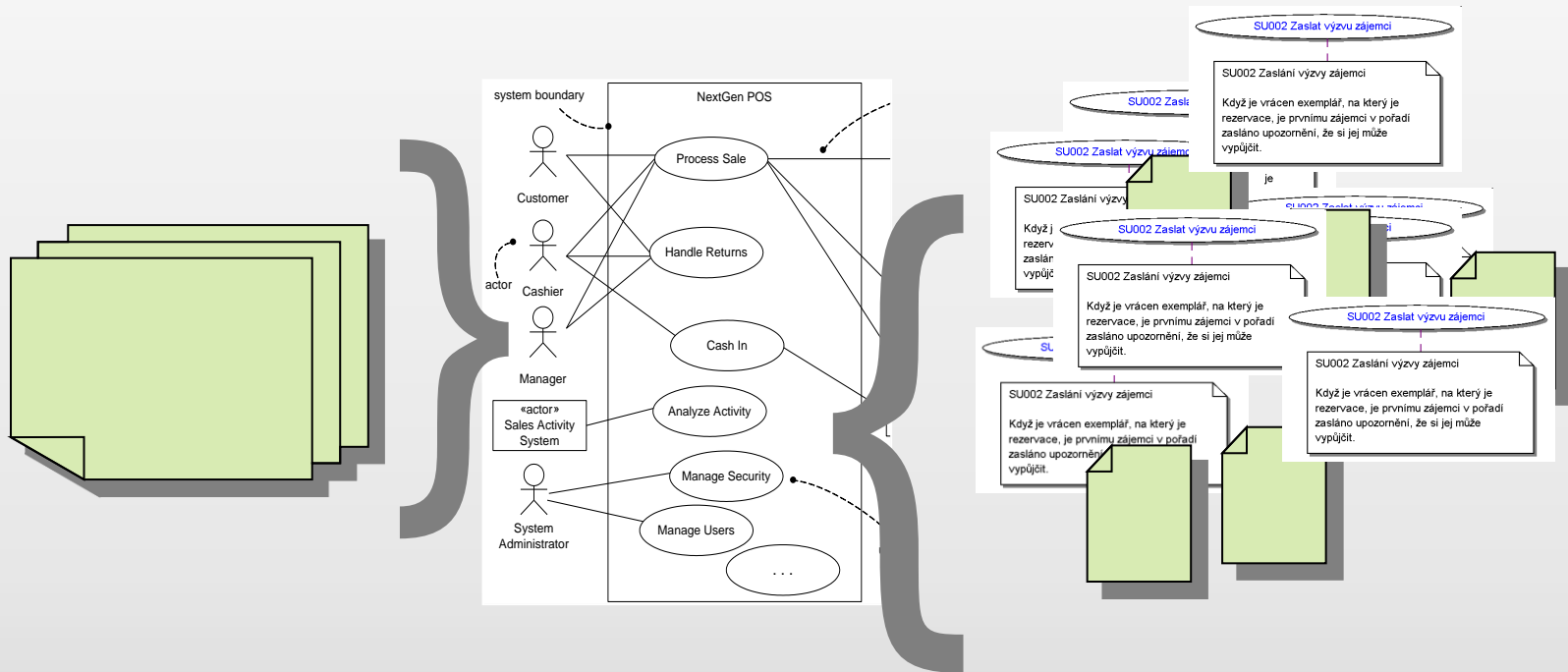


Soubor  
diagramů +  
textů =  
model užití



# ► Definice systému s případy užití

- Charakteristiky aktérů
- Model užití



- Jeden dokument (diagram jako „obsah“) nebo informace v UML nástroji

## ► **Příklad**

---

- ECP Use Case Model (Outlined)

# ► Data: Glosář

---

- Seznam důležitých pojmů
  - klíčové
  - nejasné
  - sporné
- Stručný, všemi odsouhlasený popis = společný slovník, prevence nedorozumění
- Formát různý
  - Word
  - Excel
  - databáze

# ► **Příklad glosáře**

---

## ► ECP Project Glossary

# ► Doménový model

---

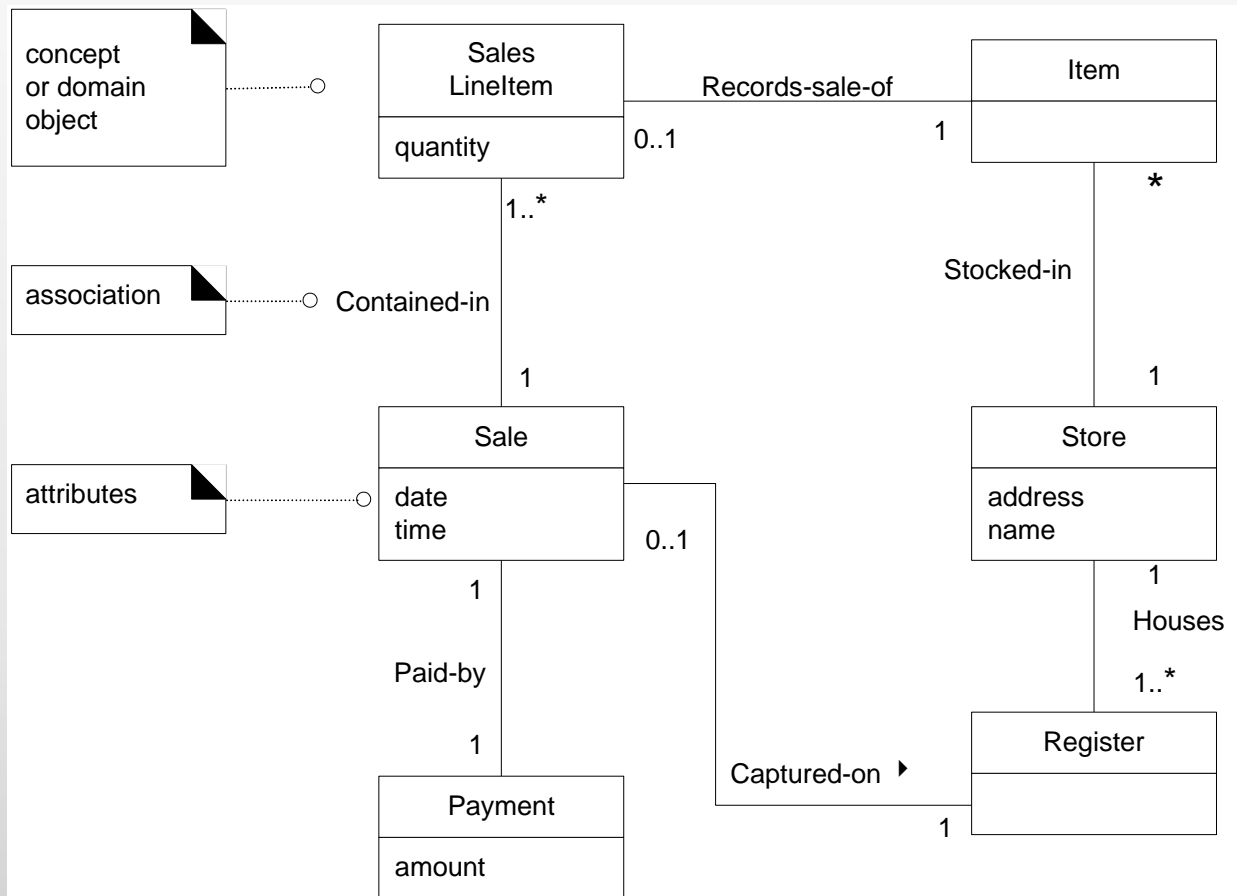
- Popis struktury problémové oblasti
  - Jaké jsou základní abstrakce používané v oblasti aplikace?
  - Jaké mají názvy, vzájemné vztahy a vlastnosti?
  - Jakým postupem je získáme?
  - Podle čeho si máme vymyslet [stabilní] třídy pro realizaci?
- Východisko = glosář
- Model = doménové objekty (diagram tříd)

# ► Doménový model – prvky

---

- Doménové objekty  $\Rightarrow$  třídy
  - “věci” vyskytující se v problémové doméně
  - klíčové pro fungování systému
  - systém udržuje informace
- Podstatné aspekty
  - terminologie uživatele, pojmy  $\rightarrow$  názvy tříd
  - jen základní obrysy
  - vztahy mezi třídami (asociace, kardinality)
  - nezávislost na implementaci

# ► UML: doménový diagram



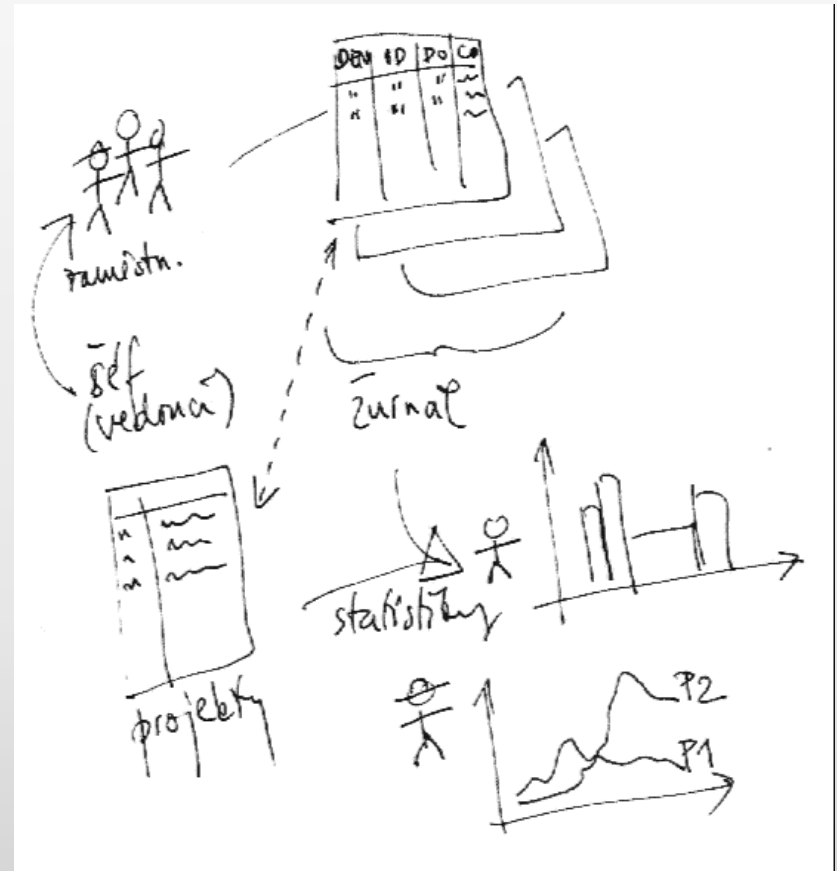
# ► Jak najít doménové objekty

## ► Doménová analýza

- konzultace
- doménový expert
- části systému podstatné z jejich pohledu

## ► Pomůžte

- obrázek
- rozhovor s uživatelem
- pozorování práce



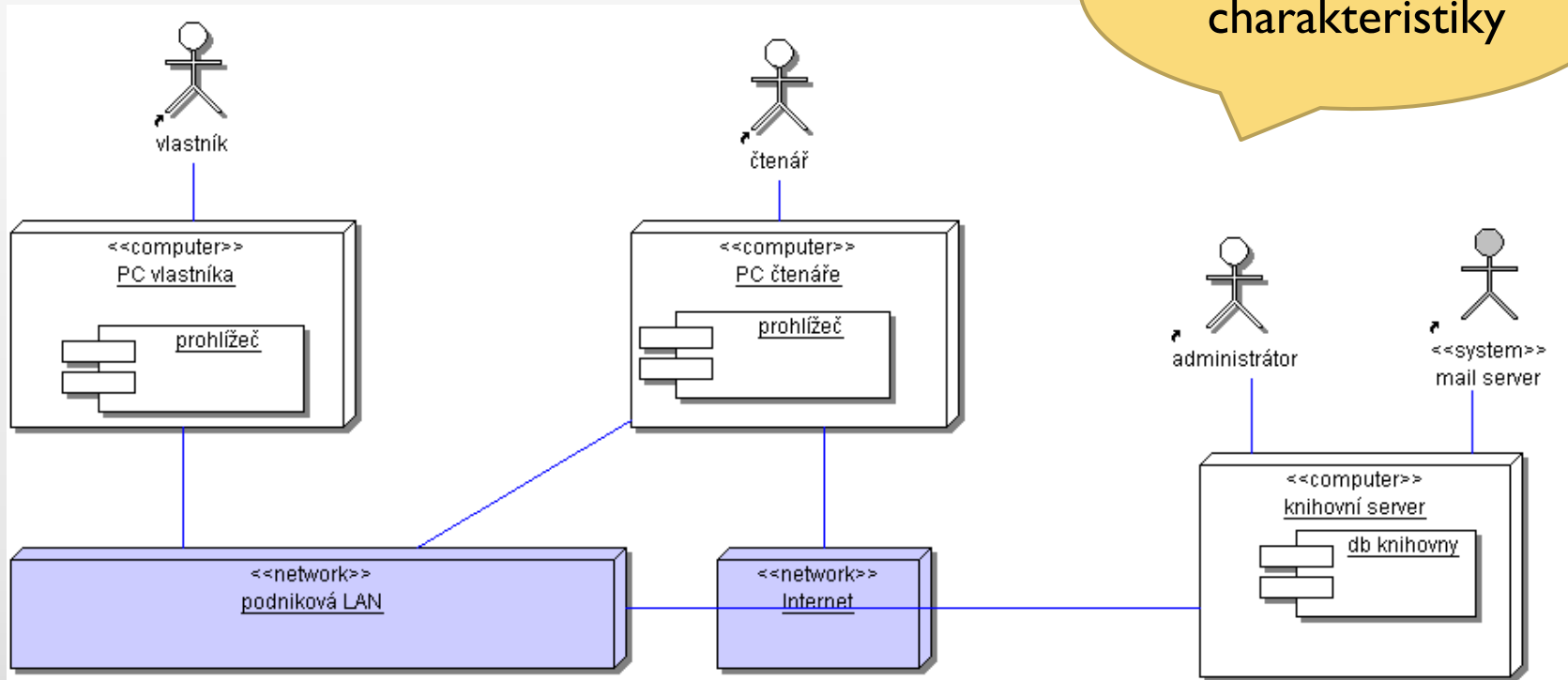


# ► Prostředí: Fyzický rozsah systému

---

- Vztahy produktu a prostředí
  - porozumění run-time a fyzickému prostředí
  - charakteristiky, parametry – mimofunkční požadavky
  - odhad nákladů + podklad pro architekturu
- Varianty
  - „zelená louka“ – součást návrhu architektury (později)
  - „brownfield“ – nutná součást analýzy

## ► Popis prostředí s UML: diagram nasazení





## **Obrysy požadavků agilně: Product backlog**

## ► Cíl agilních specifikací požadavků

---

- Říci „produkt by měl umět X“
- Podpořit (podnítit) budoucí diskusi o detailech
- Nikoli „Funkce X produktu vypadá tak a tak“

The title of my presentation is "Agile Requirements *Collaboration*." I chose that title very carefully. I could have said "Agile Requirements *Gathering*" or "Agile Requirements *Engineering*." I chose "collaboration" because user involvement is a key success factor in software projects... in all software projects, agile or not.

<http://jamesshore.com/Presentations/Beyond%20Story%20Cards.html>



# ► Forma: User Story

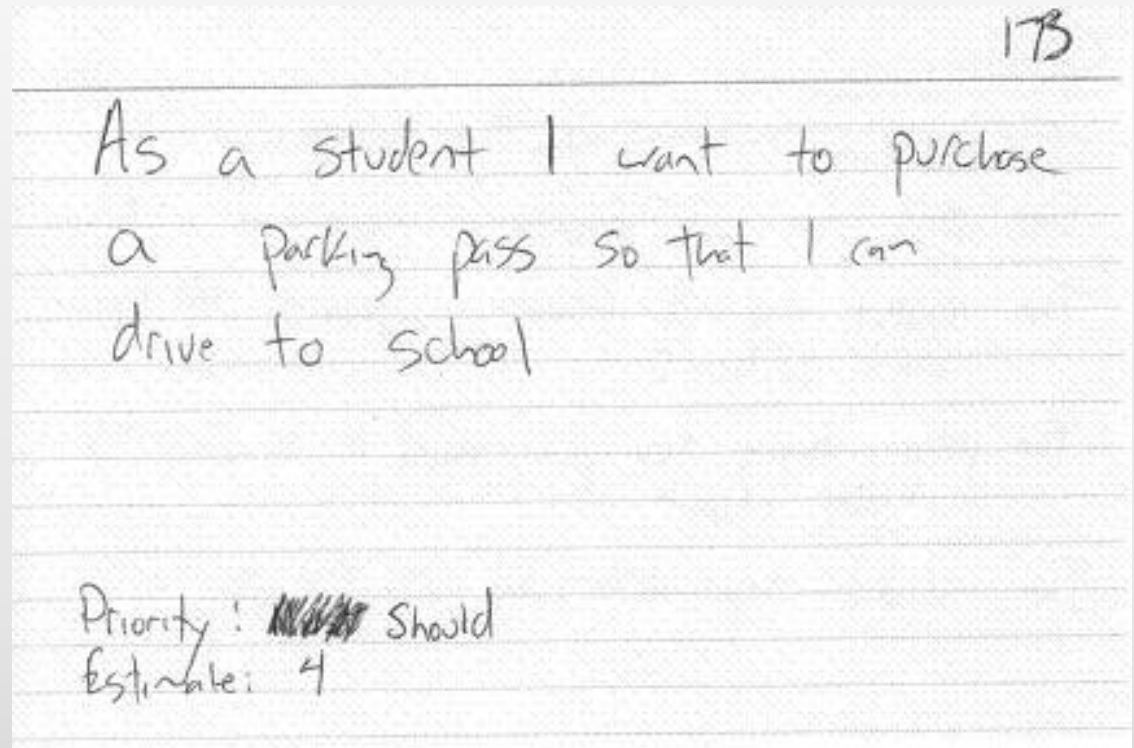
► Co uživatel od systému očekává a proč

► Obsah

- název
- stručný popis
- důležitost

► Způsob zápisu

- karta
- položka v ALM nástroji



<http://www.agileconnection.com/article/how-do-i-write-requirements-using-stories-and-acceptance-criteria-part-one>  
<http://www.agilemodeling.com/artifacts/userStory.htm>

# ► Vývoj User Stories

- Počátky projektu + dosud neanalyzované oblasti

- (Vision)

- Feature

- „minimal marketable“  
= release level
  - pracnost > 3 iterace

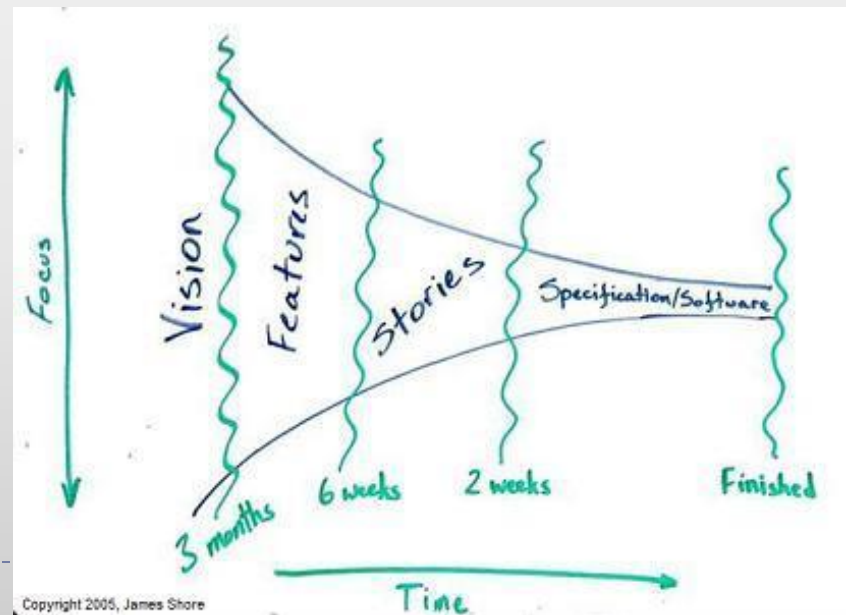
- Epic

- větší funkcionalita
  - pracnost > 1 iterace
  - má akceptační testy

- User story

- (sada = Theme)

Feature
Book search capability (title, author, subject)
Online shopping cart
Credit card processing integration
Book details
Order tracking
Book preview



# ► Definice systému agilně: Backlog

- Product Backlog = základní struktura
  - obsahuje epics, stories
  - počátek projektu: features, epics (~ definice systému)
  - just-in-time  
zpřesňování  
(příští iterace)
- Nejen požadavky
  - viz Plánování
  - viz změnové řízení





# ► Product backlog: příklad

EPICS	
►	As a patron, I want seamless integration with Illiad
►	As a patron, I want a queue of holds that can be used to automatically generate holds for me so I stay at an reasonable level of items out.
►	As a patron, I want improved recommendations for other books that are similar to a title I am looking at or that I may want to read based on my reading history.
►	As a user, I would like to access my library information via Facebook, recommend titles to my friends, add reviews, etc.
►	As a collection development, I would like to add functionality for patron driven acquisitions.
►	As a patron, I want to easily access related data for a title (FRBR)
►	As a librarian, I would like a staff only interface that can be accessed by iPad which would allow increased functionality.

typické Epics



spíše User stories



# **Způsoby získávání požadavků**

I'LL NEED TO KNOW  
YOUR REQUIREMENTS  
BEFORE I START TO  
DESIGN THE SOFTWARE.



E-mail: SCOTTADAMS@AOL.COM

FIRST OF ALL,  
WHAT ARE YOU  
TRYING TO  
ACCOMPLISH?



I'M TRYING TO  
MAKE YOU DESIGN  
MY SOFTWARE.



© 2006 Scott Adams, Inc. / Dist. by UFS, Inc.

I MEAN WHAT ARE  
YOU TRYING TO  
ACCOMPLISH WITH  
THE SOFTWARE?



I WON'T KNOW WHAT  
I CAN ACCOMPLISH  
UNTIL YOU TELL ME  
WHAT THE SOFTWARE  
CAN DO.



1-2-06

TRY TO GET THIS  
CONCEPT THROUGH YOUR  
THICK SKULL: THE  
SOFTWARE CAN DO  
WHATEVER I DESIGN  
IT TO DO!



www.dilbert.com

CAN YOU DESIGN  
IT TO TELL YOU  
MY REQUIREMENTS?



# ► Způsoby získávání požadavků

## ► Neinteraktivní

- studium dokumentace, hlášení problémů
- analýzy trhu
- konkurenční systémy

Specializované metodiky, postupy  
a nástroje – DOORS,  
RequisitePro, on-site customer

## ► Interaktivní

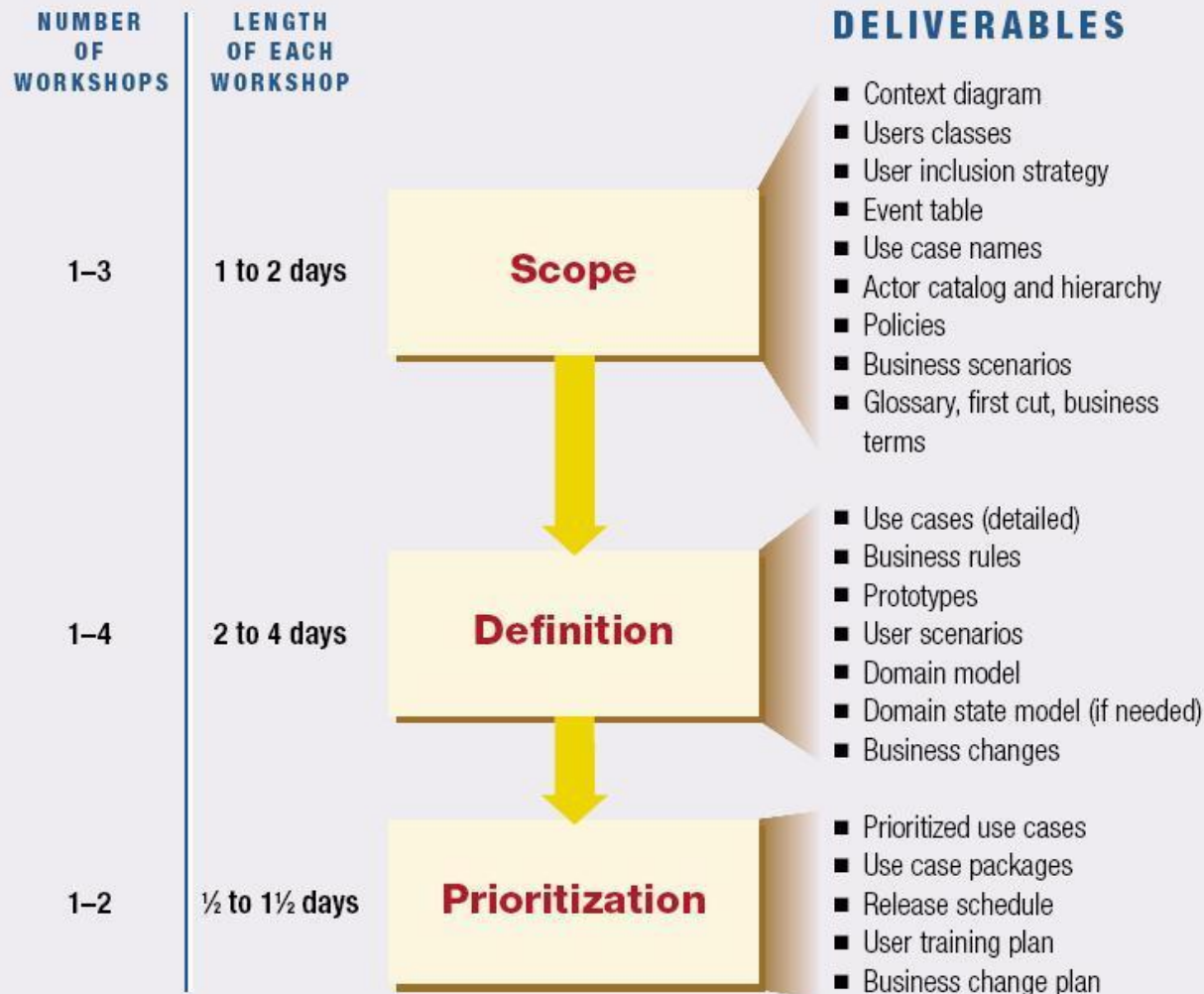
- rozhovory; **requirements workshop**
- pozorování (shadowing), práce s uživateli
- průzkumy (marketingové), dotazníky
- prototypování

Jaké jsou hlavní kroky?  
Výsledky a výstupy?

Doplňkové info: OpenUp Guideline: Requirements Gathering Techniques

<http://itcoolguy.wordpress.com/2005/08/31/running-a-requirements-workshop/>

## Příklad: requirements workshop



*Note: Some user requirements deliverables are created outside workshops (e.g., the glossary and nonfunctional requirements). As pre-work to each workshop, drafts of some of the deliverables are created.*

<http://itcoolguy.wordpress.com/2005/08/31/running-a-requirements-workshop/>

# **Závěrečné poznámky**

# ► Úvodní obrysy požadavků

---

## ► Obsah je důležitý

- úplnost, pokrytí potřeb skupin uživatelů a stakeholderů
- priority
- srozumitelnost

## ► Formy jsou různé

- funkcionalita, data, pravidla
- use case
- user story
- doménový model

<http://www.stellman-greene.com/2009/05/03/requirements-101-user-stories-vs-use-cases/>

---

# ► Úplnost modelu požadavků

---

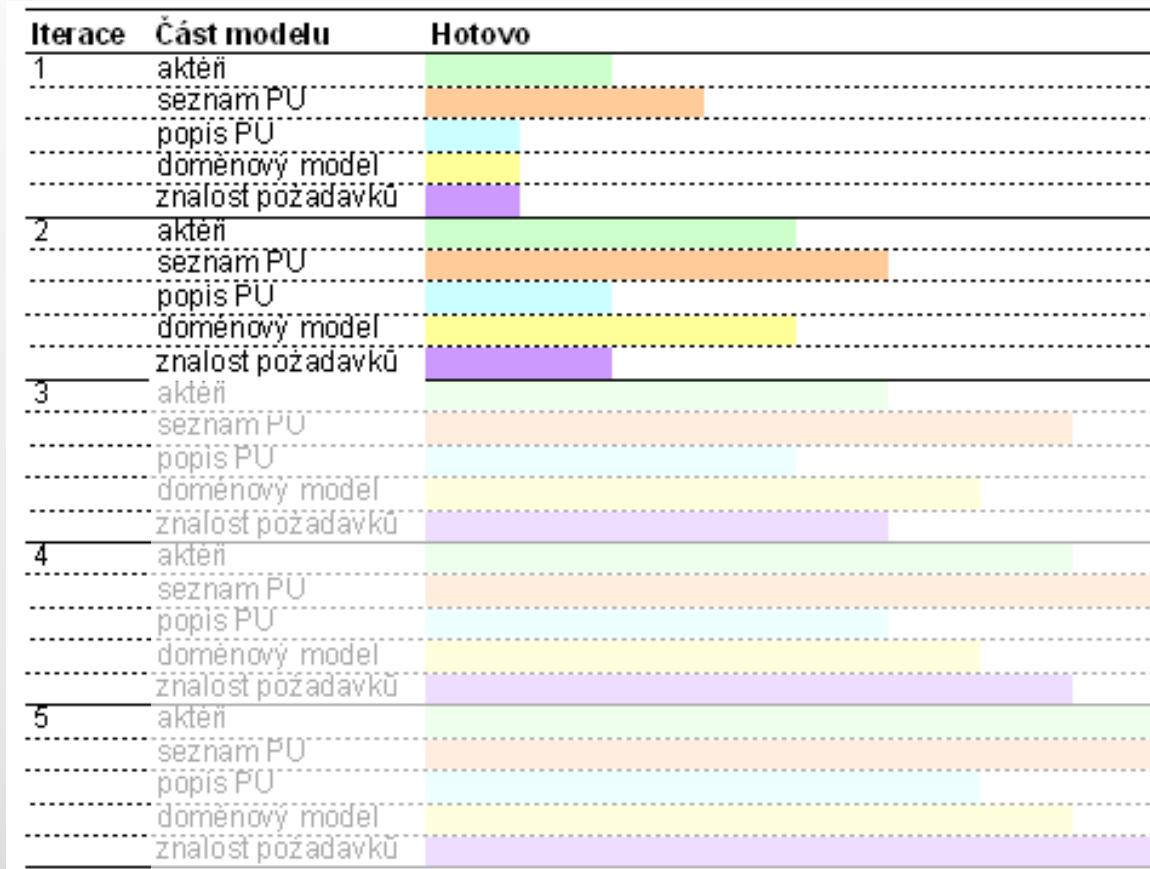
## ► Fáze zahájení projektu

- přesně cíl/vize projektu
- seznam klíčových aktérů, jejich cíle
- seznam/diagram podstatných funkčností (dle cíle)
- stručný popis klíčových funkčností, znalost klíčových vlastností

## ► (Fáze projektování)

- kompletní seznam aktérů, popis důležitých
- přesné diagramy užití, 100% PU
- přesné popisy důležitých PU, stručné u všech
- přesný popis vlastností

## ► Vývoj znalosti požadavků



- V závislosti na iteraci, fázi, release



## ▶ **Prototyp architektury**

---

- ▶ OpenUP: Task > Architecture > Envision the Architecture
- ▶ Architectural Proof-of-Concept