52. ZÍSKÁVÁNÍ A MODELOVÁNÍ POŽADAVKŮ

- získávání, definování, vyjasnění, určení a analýza požadavků tvoří jeden z hlavních a klíčových aspektů vývoje SW
- je nutné přesně odhalit požadavky klienta (zákazníka) na požadovaný vyvíjený SW či systém
- klient musí mít alespoň částečnou představu o tom co chce
- úkolem obchodního analytika je spolu s klientem odhalit a identifikovat jeho potřeby a problémy, které chce pomocí toho SW řešit
  - ⇒ tedy získat požadavky na vyvíjený SW a tyto požadavky analyzovat
- pokud se tato fáze provede špatně, může být realizován SW, který ve výsledku nesplňuje požadavky zákazníka a způsobí to velké ztráty na obou stranách
  - → tato fáze vývoje SW je tedy klíčová
- zároveň se ale jistá velká část požadavků v průběhu času mění (cca 25%) a je tedy potřeba na toto reagovat a měnit požadavky striktně dané (smluvené) aniž by šly měnit a upravovat
- vyžaduje systematický přístup ke zjišťování a sledování měnících se požadavků
- kladen důraz na přímou účast zákazníka
- typy požadavků
  - funkční – na funkčnost systému / SW
    - co to má umět a jak to má fungovat a z čeho to má být složeno
    - schopnosti, bezpečnost
    - vstupy, výstupy
  - nefunkční – spíše požadavky na kvalitu systému
    - ⇒ kvalitativní požadavky

- Model požadavků FURPS+
  - F – funkční požadavky
  - URPS – nefunkční / kvalitativní požadavky

- jiné požadavky mají na SW začátečník jiné pokročilý meziv. a jiné expert

- (F) – Function – funkčnost – viz
- (U) – Usability – použitelnost – jaké to má mít rozhraní (frontend), jak se to bude používat, přívětivý, nenáročný, návody, lidské faktory, použitelnost na mobilu, na PC na tabletu, lze intuitivně;
- (R) – Reliability – spolehlivost – jak je to spolehlivé, jak časté výpadky a jak rychle se to zotaví, kdy je údržba tak se dá pracovat nebo ne?
  - přesnost / spolehlivost výsledků které systém dává
- (P) – Performance – výkonnost – jaký ten systém má výkon, jak to má být rychlé vracení odezvy (200 ms), kolik se toho do něj dá uložit
- (S) – Supportability – podporovatelnost – udržovatelnost, přizpůsobitelnost, konfigurovatelnost, internacionalizace – překlady do více jazyků atd..
- (+) – požadavky na implementaci (prostředí, jazyk, postupy, OO), rozhraní (s čím to má jak propojit..) fyzické požadavky (na jakém HW to má stát, kolik paměti) a právní (licence,..)

Techniky získávání požadavků

A) Tradiční

- **Interview**
  - rozhovor analytika se zákazníkem a odhalení toho co potřebuje
  - nepodsouvat nějaká řešení ale spíše naslouchat
  - strukturované – připravené objektivní otázky a třeba k nim i odpovědi / nestrukturované – neformální
- **Dotazníky**
  - výhodou je čas a to že si to může v klidu zákazník rozmyslet a není do ničeho tlačen
  - nevýhoda je že nemusí rozumět otázce a nemá kdo mu ji vysvětlit rovnou – nemusí pochopit otázku a odpoví na ni omylem špatně
- **Pozorování**
  - analytik rovnou na pracovišti pozoruje jak zákazník pracuje a odhaluje jeho potřeby
  - analytik se může aktivně zapojit a sám si tu práci vyzkoušet a pak mu to může vysvětlovat
  - nevýhodou je že se klienti před někým chovají jinak
- **Studium**
  - dokumentů, knih, článků, standardů, zákaznických standardů ve firmě, … – pracovní postupy ve té firmě

B) Moderní – uživatel neví co přesně chce a požadavky nelze zjistit jinak než mu předkládat prototypy

- **Prototypování** – vytváří se několik prototypů na základě požadavků a ty se ukazují klientovi a ten své požadavky na základě toho upřesňuje a upravuje a mění
  - prototypy se týkají hlavně třeba rozhraní a rozložení a vzhledu atd
  - jsou to jen částečně funkční části systému – funkční / nefunkční – náčrty, obrázky, videa, …
- **Brainstorming**
  - sejde skupina lidí (zákazníci, investoři, analytici, projektový manažeři, vývojáři) a každý předkládá své návrhy a požadavky a předsudky a navzájem se upřesňují
  - moderátor může pokládat otázky
- **JAD (Joint Application Development)** – obdoba brainstormingu, účastní se moderátor který udržuje řád a pokládá otázky, zapisovatel (ale tyto podklady) zákazníci a vývojáři (+ analytici) – upřesňují požadavky a řeší je rovnou, je to ↳ řeší požadavky a rozhodují který je více variant – předvídatelné
- **RAD (Rapid Application Development)** – týká se celého vývoje SW ale i sběru požadavků – rychlý vývoj aplikace, kvalita před kvalitou, kvalita a rychlost, …

Tvorba modelu případů užití → textové dokumenty a případy užití z hlediska uživatele!

- stručný – pouze hlavní scénáři
- neformální – i alternativní scénáře
- úplný – strukturovaný, detailní, všechny scénáře

- nejen diagram ale i textový popis! – není to jen USE CASE diagram
- zaměření na aktéry
- zaměření na to co systém dělá a ne jak to dělá!
- je potřeba i eliminovat duplicitní požadavky a požadavky které nespadají do dané oblasti nebo jsou irelevantní
  - → jednoznačné uspořádání požadavků do dokumentu a hierarchie a každý požadavek svoje ID – lze použít CASE nástroje

⟹ **dokument požadavků** – třeba se může použít i šablona

Části UP (Unified Process nebo RUP) které souvisí s požadavky

- model případů užití – jsou to textové dokumenty
  - předkládají použití systému a hlediska uživatele
- doplňující specifikace
- vize – přínos projektu
- slovník – pojmy které se budou v požadavcích / analýze / návrhu atd.… používat
  - s jejími pojmy budeme operovat
- rozsah systému
- obchodní model použití – use case na vysoké úrovni abstrakce
- obchodní diagram tříd – ~~hierarchie~~ diagram klíčových obchodních objektů systému na vysoké úrovni abstrakce

⋮

⟹ výstup získávání požadavků:

- Dokumenty požadavků (textový soupis, hierarchicky (dokumenty) uspořádané dokumenty, strukturované, podle šablony, lze použít ID, a nad nimi se dají použít CASE nástroje)
- Obchodní model použití (v podstatě use case na velmi vysoké úrovni abstrakce)

  \+ vize, slovník pojmů, rozsah systému

Model případů užití – jsou to textové dokumenty
- zaměřují se na cíle projektu a typické použití SW z pohledu uživatele
- co systém dělá, ne jak
- zaměření na aktéry a jejich úmysly
- textové ale může být doplněno diagramem