

Modelovanie use-casov

Bittner, R., Spence, I.: Use Case Modeling,
Addison-Wesley 2002,
ISBN 0-201-70913-9

Zmysel use-casov

- ako zachytiť, preskúmať a zadokumentovať požiadavky na systém?
 - to, čo má systém robiť?
- ako prehľadne znázorniť a popísať ako bude systém fungovať?
- ako vytvoriť mentálny model spôsobu, ktorým systém pracuje na konceptuálnej úrovni?
 - ...tak, aby to pochopili aj zadávatelia a používatelia?

Use-Case Model

- množina use-caseov, aktorov a vzťahov medzi nimi, ktorá popisuje konkrétny systém
- **use-case model** opisuje všetky rozličné spôsoby využitia systému
 - voľnejšie: najtypickejšie spôsoby používania systému

Model (def. z UML)

- sémanticky uzavretá abstrakcia systému.
- uzavretý [bez potreby ďalších informácií] popis systému z danej perspektívy

Use case je príbeh

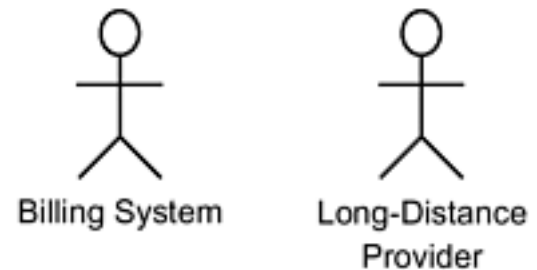
- popis use-casu = príbeh
 - „ako systém a aktori pospolu dosiahli konkrétny cieľ“
 - dialóg medzi systémom a aktormi
- úvod – ako aktor začne používať UC
 - aktor niečo urobí, systém odpovie
- jadro – ako systém a aktor spolupracovali
 - dialóg medzi aktormom a systémom
- záver – ako sa uzavrie UC
 - vo chvíli, keď systém vykonal niečo „dobré“ pre jedného či viacerých aktovov

Use case je príbeh

- UC nie je opisom všetkých možných spôsobov, ktorými možno dosiahnuť cieľ
- nehovorí nič o návrhu či implementácii systému
- je to typický spôsob (prípado) použitia systému

Základné stavebné bloky: aktor

- definuje rolu, ktorú môže používateľ zaujať pri práci so systémom.
 - používateľ: fyzická osoba ale aj iný systém
- UML: koherentná sada rolí, ktoré používatelia v use-case hrajú pri interakcii s use-caseom
- reprezentujú ľudí či iné systémy
- nachádzajú sa mimo systému
 - typicky nad nimi systém nemá kontrolu
- stanovujú požiadavky na to, čo má systém robiť



Základné stavebné bloky: use-case

- ako aktor využíva systém na to, aby dosiahol svoj cieľ?
- čo robí systém pre aktora na to, aby sa jeho cieľ naplnil?
- ako spolupracuje systém a aktori v spoločnom úsilí dodať aspoň jednému z aktorov niečo hodnotné?

Use-Case: definícia v UML

Popis sady postupností akcií (a ich variantov), ktoré systém vykonáva preto, aby poskytol konkrétnemu aktorovi pozorovateľný výsledok.

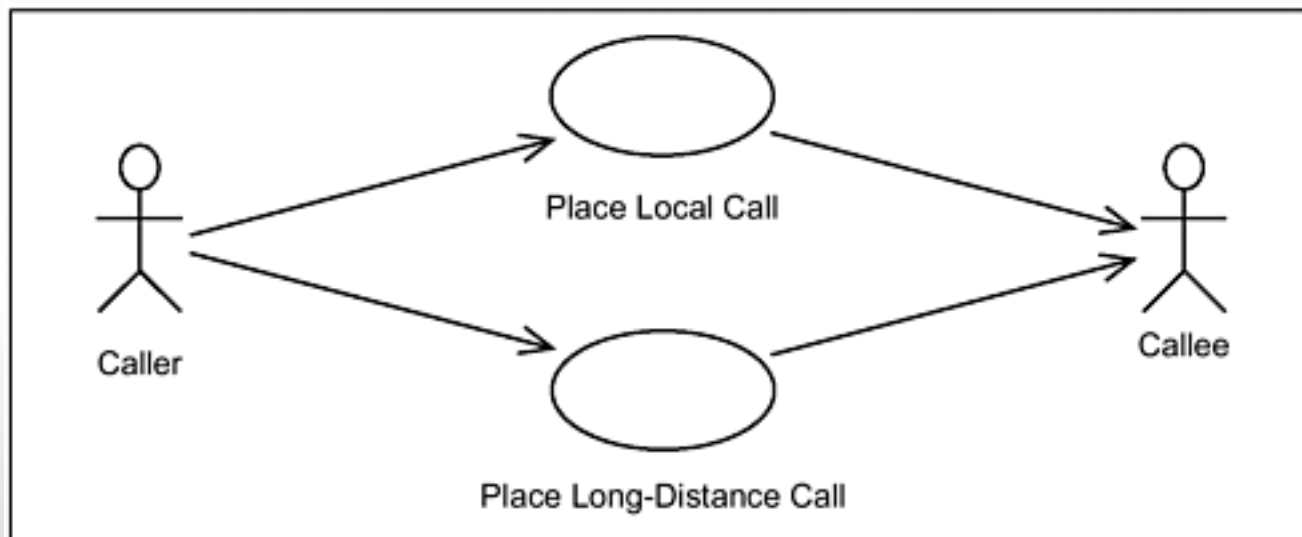
Sadu postupností vnímame ich ako "transakciu": buď sa uskutoční úplne alebo vôbec

Diagramy vs obkec

- **UC má popis** – text, ktorý opisuje príbeh
 - príbeh: čo robí systém pre konkrétneho aktora?
- aktor využíva systém
 - systém poskytuje use-case
 - aktor spúšťa use-case
- UC opisuje, ako systém prináša aktoram nejakú hodnotu
 - ciele aktorov a dodanie hodnoty sú fundamentálnou súčasťou odhalenia, definície a aplikácie UC

UC v diagramoch

- čiara: komunikačná asociácia [communicate association]
- šípka: iniciátor komunikácie je na strane bez šípky



Stručný popis UC

- každý aktor a každý UC **musí** mať stručný popis
 - obsahuje krátku charakteristiku
 - zmysel existencie
- stručnosť = cca 2 vety

„Tento use-case popisuje, ako zákazník banky využíva bankomatový terminál na výber peňazí z vlastného účtu.“

Scenár – Flow of Events

- rozpráva samotný príbeh
- príklad: „ako sa dostať k Jožovi na žúrku“
 - opis úspešnej cesty (**basic flow**)
 - alternatívne spôsoby (**alternate flows**)

Hlavný scenár / The Basic Flow / Happy Day Scenario

- bežný prípad / očakávané použitie

Use Case – Uskutočniť miestny hovor

- *Volajúci* zdvihne slúchadlo
- *Volajúci* vytočí číslo
- Systém pripojí telefón *volajúceho* k požadovanému zariadeniu
- Uskutoční sa hovor
- Spojenie sa ukončí
- Zaznamenajú sa podrobnosti o hovore
- UC končí

Alternatívne použitie

- voliteľné správanie / varianty
- obchádzky v bežnej ceseťe

Use Case – Uskutočniť miestny hovor

- Žiadna odpoveď
 - ak volaný neodpovedá, volajúci zloží slúchadlo a UC končí
- Obsadené
 - Ak je u volaného obsadené, systém to indikuje pípaním. Volajúci zloží slúchadlo a UC končí

Alternatívne použitie: Chybové stavy

- alternatívne toky plynúce z výskytu chýb a ich obsluhy

Use Case – Uskutočniť miestny hovor

- Neznáme volané číslo
 - systém nedokáže identifikovať koncové zariadenie, ktoré je volané
- Strata signálu
 - počas hovoru nastane výpadok na linke

Subflows: vnorené toky

- niekedy treba rozdeliť tok udalostí na čiastkové úlohy

- Systém pripojí telefón *volajúceho* k požadovanému zariadeniu

- systém analyzuje cifry zadaného čísla a určí sieťovú adresu *volaného*
- systém určí, či možno ustanoviť spojenie medzi *volajúcim* a *volaným*
- systém ustanoví spojenie
- systém *prezvoní* volaného mobil

Ďalší príklad

S1 Login

1. ak používateľ zadáva heslo po prvýkrát, systém ho vyzve na zadanie hesla
2. používateľ zadá heslo (systém vypisuje namiesto znakov hviezdinky). Ak používateľ naznačí dokončenie zadávania hesla, systém porovná zadané heslo s heslom uloženým v používateľskom profile
3. Ak sa heslá zhodujú, používateľovi je poskytnutý prístup do systému a UC pokračuje
 1. ak používateľ nezadá správne heslo, systém ohlási, že heslo je nesprávne
 1. používateľovi sú dané dva dodatočné pokusy na zadanie hesla
 2. ak používateľ nezadá heslo ani po troch pokusoch, systém zaznamená dátum a čas pokusu spolu s profilom používateľa a odhlási ho zo systému
 2. ak sa heslá zhodujú, používateľovi sa poskytne prístup do systému a UC pokračuje. v opačnom prípade systém ohlási nesprávnosť hesla

...Vykonaj vnorený tok **Login...**

Použitie vnoreného toku

- niekedy treba rozdeliť tok udalostí na čiastkové úlohy

Vnorený tok: prepoj volaného s volajúcim

- systém analyzuje cifry zadaného čísla a určí sieťovú adresu *volaného*
- systém určí, či možno ustanoviť spojenie medzi *volajúcim* a *volaným*
- systém ustanoví spojenie
- systém prezvoní *volaného* mobil

- volajúci zadá číslo *volaného*
- vykonaj Prepoj volaného s volajúcim
- systém drží spojenie až dokiaľ *volaný* či *volajúci* neukončia hovor

Písanie voliteľných tokov

- ak možno alternatívu popísať tak, že neodvádza pozornosť od hlavného toku, uveďme ju rovno doň
 - kurzívou indikujeme chybový stav

- Systém načíta z **karty** všetky **detaily** o nej.
- *Ak systém nemôže načítať všetky **detaily**, potom systém informuje Zákazníka o tom, že karta nemôže byť prečítaná. Karta je vrátená Zákazníkovi a UC končí.*
- V opačnom prípade systém nakontaktuje Zákazníkovu banku.

Písanie voliteľných tokov

- Systém načíta z **karty** všetky **detaily** o nej.
- *Ak systém nemôže načítať všetky **detaily**, potom systém informuje Zákazníka o tom, že karta nemôže byť prečítaná. Karta je vrátená Zákazníkovi a UC končí.*
- Systém nakontaktuje **Zákazníkovu banku**, aby overil, že **informácia o zákazníkovi** je správna.
- Ak Zákazníková banka ohlási, že karta bola odcudzená :
 - systém skonfiškuje kartu a oznámi konfiškáciu Zákazníkovej banke
 - Uloží videozáznam zákazníka pre budúce účely
 - Ukončí transakciu
 - Oznámi zákazníkovi, že
 - karta bola nahlásená ako odcudzená
 - karta bola skonfiškovaná
 - Zákazník má nakontaktovať banku
- UC končí
- V opačnom prípade [...]

Ošetrenie výnimočnej situácie odkláňa pozornosť od hlavného toku.

Písanie voliteľných tokov

Základný scenár

...

- {Načítanie karty} Systém načíta z **karty** všetky **detaily** o nej.
- {Overenie údajov na karte} Systém nakontaktuje **Zákazníkovu banku**, aby overil, že **informácia o zákazníkovi** je správna.

....

Alternatívne scenáre:

- A1: Vrátene nečitateľnej karty. Ak systém nemôže načítať všetky **detaily**, potom systém informuje Zákazníka o tom, že karta nemôže byť prečítaná. Karta je vrátená Zákazníkovi a UC končí.
- 2: Spracovanie odcudzenej karty
Pri {Overovaní údajov na karte} ak Zákazníkova banka ohlási, že karta bola odcudzená :
 - systém skonfiškujú kartu a oznámi konfiškáciu Zákazníkovej banke
 -

UC končí.

Písanie voliteľných tokov

- Kučeravé zátvorky s menom sú lepšie než čísla
 - nazývané extension points
 - udalosti v scenári môžeme totiž podľa potreby rozširovať
 - čísla sa menia, často nutné prečíslovať
 - mená by mali byť popisné
 - zlý príklad: {Pokračuj v spracovaní}
- Alternatívy identifikované predponou "A"
- očíslované v takom poradí, v akom boli odhalené

Veľkosť popisov v UC

- obvykle 5 až 15 strán
 - každý popis musí byť dostatočne dlhý na to, aby popisoval príbeh
- prepodmienky
 - očakávaný stav aktorov a systému pred začatím UC
 - príklad: zariadenie *volajúceho* je aktívne pripojené k systému
- postpodmienky
 - stav systému po skončení UC
 - príklad: spojenie medzi volajúcim a volaným bolo skončené. Detaily o hovore boli uložené.

Zhrnutie k popisom UC

- diagramy sú len prehľadným zobrazením správania sa systému
 - 90% obsahu sa skrýva pod povrchom
- najdôležitejšia časť UC je podrobný popis
- najdôležitejšia časť podrobného popisu je tok udalostí

Vytváranie popisov UC

- častý prístup k UC
 - identifikujme aktorov
 - identifikujme UC
 - napíšme dajaký stručný obkec
 - namaľujme dajaké diagramy, aby bolo
- dôsledok: nad podrobným tokom udalostí sa často uvažuje až pri písaní kódu
- často sa však pletie **čo** treba urobiť s tým **ako** sa to dosiahne
 - vyvstávajú otázky: vyplýva konkrétne správanie z používateľských požiadaviek alebo je to vedľajší dôsledok návrhu systému?

Vlastnosti písania UC

- vytváranie popisov k UC je spisovateľstvo
 - najlepšie je ich vytvárať vo dvojici / v tíme
- programátori majú niekedy problém spisovať UC
 - kód = **ako** systém vykoná konkrétnu vec
 - UC = **čo** má systém robiť
- UC = detektívka = záhada toho, čo má systém urobiť preto, aby bol užitočný ľuďom

Vlastnosti písania UC

- písať v činnom rode
 - „systém overí zadanú hodnotu“
 - protipríklad: zadaná hodnota by mala byť skontrolovaná systémom
- písať v prítomnom čase
 - nepísať v budúcom čase: „systém bude overovať“
- jednoduché priame veci
- organizovať informácie zhora nadol

UC považujte za príbeh

- vhodné začať "infantilne"
 - „UC začína vo chvíli keď aktor X urobí [niečo]"
- vhodné používať:
 - „Aktor X urobí [blabla], a následne systém zareaguje tým, že urobí [yadda, yadda]."
- miera detailov záleží od systému

UC: opis interakcie medzi aktorm a systémom

- účel UC modelu je zachytiť interakciu medzi aktormi a systémom
 - čo urobí aktor pre systém
 - a čo urobí systém v rámci odpovede
- UC má opisovať **čo** systém urobí
- ale neopisuje činnosť používateľského rozhrania ani spoluprácu komponentov systému na to, aby dosiahli cieľ

Príklady a protipríklady

- zlý: systém zobrazí dialógové okno na to, aby doň používateľ zadal detaily
- zlý: systém zobrazí listbox so zoznamom produktov
- dobrý: systém určí, či je zadaný PIN korektný
- dobrý: systém zaznamená objem vložených peňazí
- zlý: komponent uloží objem peňazí ako reťazec do transakčnej databázy

Vysporiadanie sa s detailami

- dobré UC nemajú úrovne
 - túžba vidieť úrovne často mení UC na návrhársky nástroj
- UC zachytáva popis niečoho, čo má systém robiť
 - vyjadrenie požadovaného správania sa systému
 - systém musí robiť TO a TO bez ohľadu na to, kde a ako je implementovaný

Vysporiadanie sa s detailami

- zásada: uistite sa, že UC poskytuje používateľovi systému niečo hodnotné
- zlý príklad: prihlásenie do bankomatu
 - prečo? používateľ zadá PIN, systém vypíše „blahoželám, zadali ste úspešne PIN“
 - toto nie je ktovieaká pridaná hodnota
- dobrý príklad UC: vybrať peniaze
 - prihlásenie je len jeden z krokov, ktoré treba vykonať na dosiahnutie cieľa

Vysporiadanie sa s detailami

- Rada pre zmýšľanie
 - som používateľom a mojim **zámerom** je...
 - vybrať peniaze
 - na to potrebujem:
 - prihlásiť sa do systému
- Zlá rada pre zmýšľanie:
 - som používateľom a mojim **zámerom** je
 - prihlásiť sa do systému
 - prečo?
 - aby som vybral peniaze

Zámerom používateľa je **vybrať peniaze**.
Prihlásenie je len jeden z nutných krokov.

Pozor na CRUD

- CRUD = create, retrieve, update, delete
- UC sa mimoriadne hodí na pochopenie sekvencií správania sa systému
 - resp. na pochopenie toku udalostí
- CRUD je z hľadiska správania nezaujímavé
 - dokola: „systém zobrazí políčka, používateľ ich vyplní, systém overí dáta, používateľ odošle transakciu“
- validáciu obslúži doménový model
- radšej než vytvárať takéto nudné UC, je lepšie vytvoriť prototyp

Vzťahy medzi use-cases

- čo ak vidíme podobné správanie v dvoch rozličných use-caseoch?
- čo ak chceme znížiť zložitosť tým, že odčleníme časti, ktoré sa používajú len za istých okolností?
- vzťahy medzi UC zavádzajte až vtedy, keď máte aspoň predbežnú verziu scenárov

Vzťah <<include>>

- z viacerých UC vytiahneme spoločné sekcie
- predpoklad: rovnaký popisný text v najmenej dvoch popisoch UC
 - zásada: najprv napíšte popisné texty a až potom uvažujte nad <<include>
 - protizásada: nerobte <<include>> nad čistými diagramami!

Príklad: vzťah <<include>>

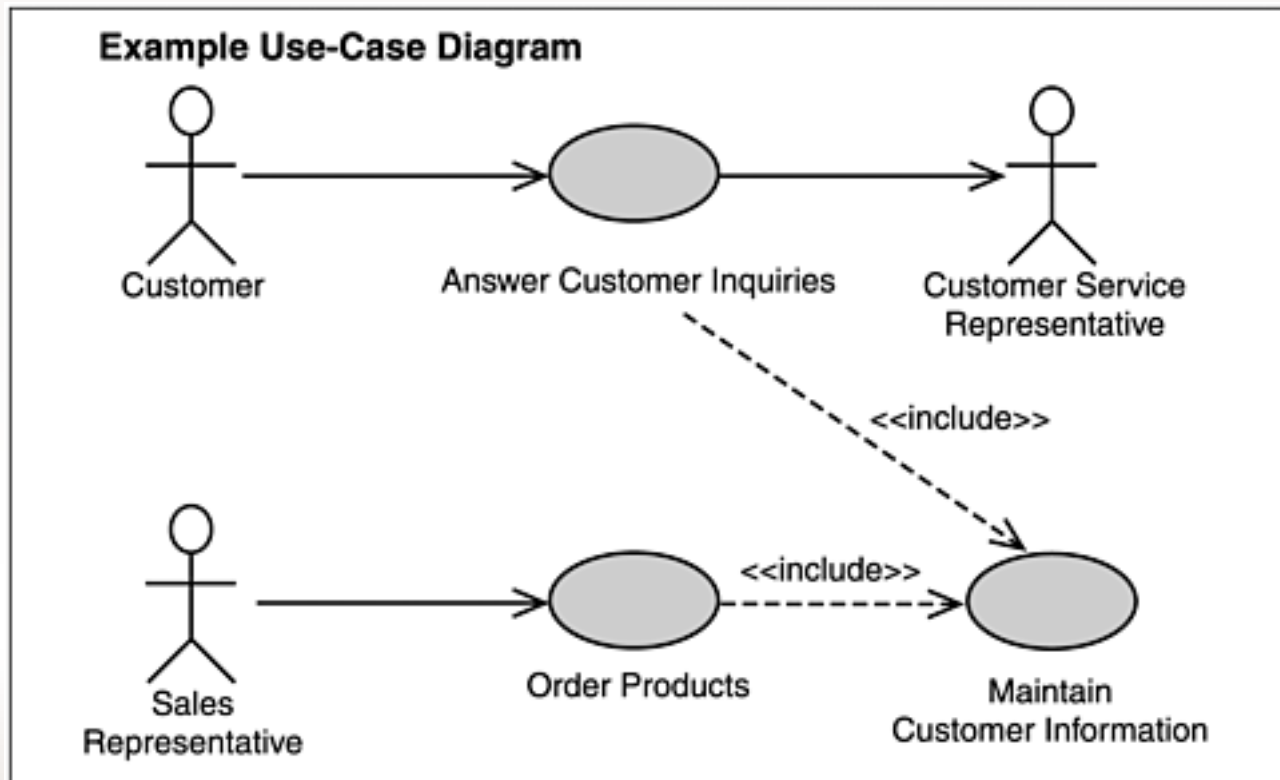
UC: hotline

9. systém požiada zamestnanca podpory (CSR), aby si vyžiadal
 - a) meno zákazníka
 - b) bydlisko
 - c) hodiny, keď ho možno kontaktovať

10. ...

- UC: objednávka tovaru
3. Ak je zákazník novým zákazníkom, díler zaznamená
 - a) meno zákazníka
 - b) bydlisko
 - c) hodiny, keď ho možno kontaktovať

Diagram



Príklad: vzťah <<include>>

UC: hotline

9. zahrňte UC **Pridať informácie o zákazníkovi**, aby mohol zamestnanec hotlinu zaznamenať údaje o zákazníkovi

10. ...

- UC: objednávka tovaru
3. Ak je zákazník novým zákazníkom, zahrňte UC **Pridať informácie o zákazníkovi**, aby mohol diler zaznamenať údaje o zákazníkovi

Rady pre <<include>>

- includeovaný UC nesmie byť použitý len jedným UC
 - v opačnom prípade rozbíjame UC na čiastkové UC a strácame prehľad o hlavnej úlohe
 - nesnažte sa rozbiť UC do čiastkových UC!
- includeovaný UC nemá žiadne vedomosti o UC, v ktorom sa používa
 - vedie to k znovupoužitelnosti UC

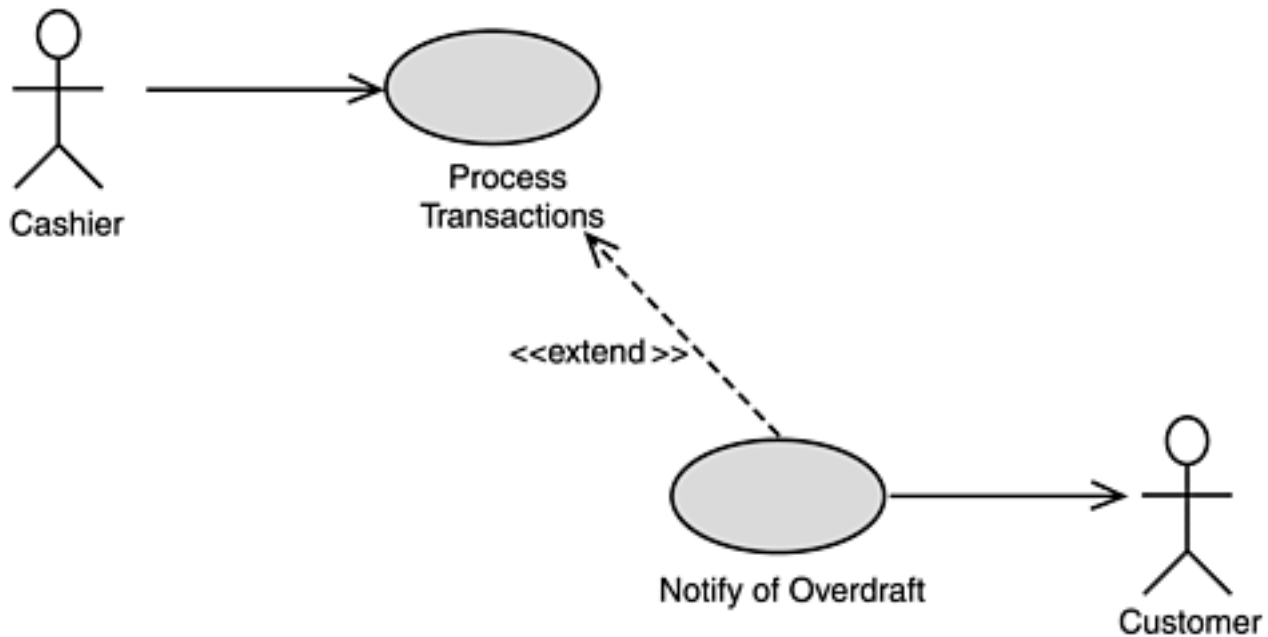
Rady pre <<include>>

- nesnažte sa využiť <<include>> na funkčnú dekompozíciu systému
 - snaha vidieť includeovaný UC ako položku v menu alebo vnímať ho ako funkciu (v zmysle programátorskom)
 - nezabúdajte: každý UC musí prinášať hodnotu!
- chyba: využívať UC mimo kontextu ostatných UC, ktoré ho využívajú
 - chyba: využiť príklad so zákazníkom aj pri UC, kde zákazník aktualizuje dáta
 - ťažko sa udržiava línia pôvodného účelu

<<extend>>

- v prípade, že sa do existujúceho UC vkladá nejaké voliteľné či výnimočné správanie
- umožňuje pridať funkcionality do existujúceho UC bez toho, aby bolo nutné modifikovať pôvodný UC
- vhodné použitie
 - popis vlastností, ktoré sú voliteľné/doplnkové k základnému správaniu systému (dodatočne zakúpiteľné)
 - opis komplexnej obsluhy výnimiek/chýb, ktoré by zahmlievalo primárne správanie systému
 - alternatívne scenáre, ktoré sú dlhšie než primárny scenár
 - prispôbenie modelu požiadaviek konkrétnemu zákazníkovi
 - správanie, ktoré bude k dispozícii až v budúcich verziách systému

Example Use-Case Diagram



<<extend>>

- UC – upozornenie na prečerpanie účtu
- **stručný opis:** tento UC upozorňuje zákazníka na prečerpanie účtu. Táto služba je k dispozícii len vtedy, ak si zákazník objednal službu upozornenia na prečerpanie účtu
- **rozšírenie:** rozširuje UC Spracovanie transakcií v {Sumáre transakcií}, ak si **zákazník** zakúpil službu **upozornenie na prečerpanie účtu** a sada **dokončených transakcií** spôsobila prečerpanie účtu.

<<extend>>

- základný scenár:
 - systém zistí zo zákazníkovoho profilu zákazníkov uprednostňovaný mechanizmus notifikácie
 - systém vytvorí **oznam o prečerpaní**, kde uvedie **informáciu o transakcii**, dátum a čas spracovania **transakcie, údaje o účte**, zostatok pred dokončením **transakcie** a **poplatok za prečerpanie**
 - systém odošle **oznam o prečerpaní** zákazníkovi jeho **uprednostňovaným spôsobom**
 - UC končí [+]

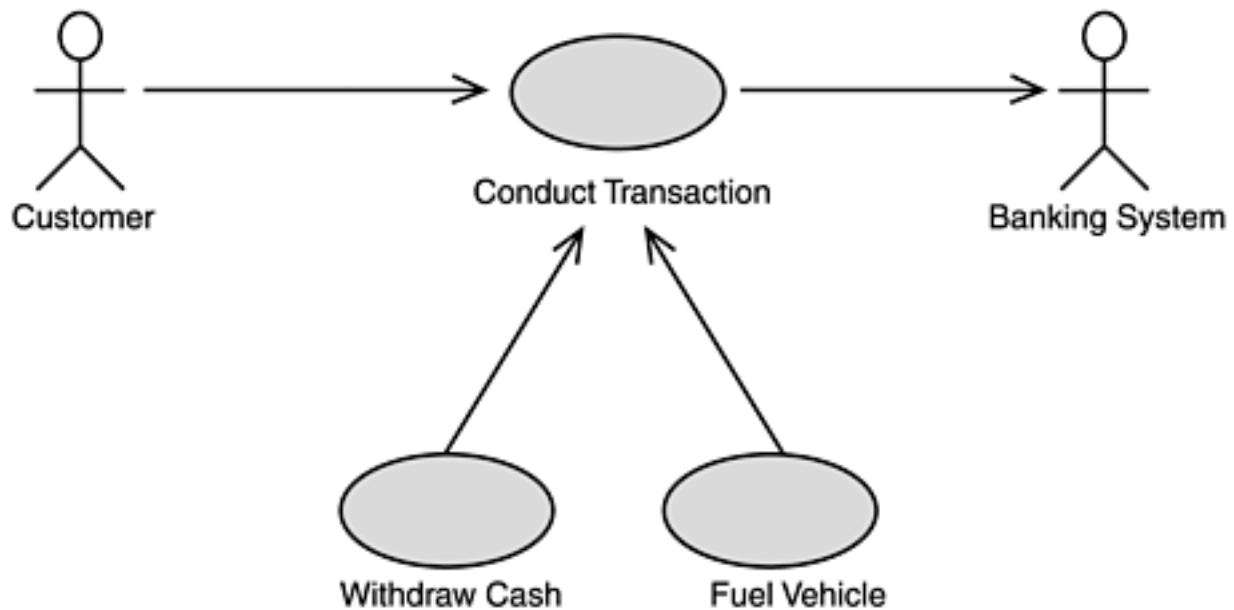
<<extend>>

- [+] koniec potomkovského UC
 - nikdy sa nemôže ukončiť rodičovský UC
- rozširujeme v extension pointoch
 - dodávame novú funkcionálnosť
- uľahčuje sa verzionovanie UC
- obvykle sa bude o každý UC starať iný človek
- ľahšie sa im porozumie

<<extend>> ako špecializácia

- výber z bankomatu:
 - vloženie karty, zadanie PIN, **výber peňazí**, zalogovanie transakcie
- platenie kartou na pumpe
 - vloženie karty, zadanie PIN, **zaplatenie za benzín**, zalogovanie transakcie
- ak máme dva UC, ktoré sú rovnaké v úvode a závere a mení sa len stred, môžeme vytvoriť separátne abstraktný UC obsahujúci spoločné chovanie a potenciálne extension pointy

Example Use-Case Diagram



Príklad abstraktného UC

UC: Vykonanie transakcie [základný scenár]

...

7. systém vyžiada sumu pre transakciu. Zákazník zadá sumu.
8. Systém kontaktuje bankový systém a overí, či má zákazník zostatok na pokrytie transakcie
{Zákazník vykoná transakciu}
9. Systém zaznamená sumu v transakcii.

Príklad konkrétneho UC

UC: Vykonanie transakcie [základný scenár]




V rámci {**Zákazník vykoná transakciu**}

1. Systém overí, či je k dispozícii dostatočný bankoviek na vydanie
2. Systém vydá príslušné bankovky
3. Systém vyzve zákazníka na odobratie bankoviek
4. Zákazník odoberie bankovky z priehradky
5. Pokračuje správanie definované v **UC Vykonanie transakcie**.

<<extends>> má dva účely

- rozšírenie:
 - do UC pridáme ďalšiu funkcionálnosť
 - pôvodný UC musí byť uzavretý a zmysluplný i sám o sebe
 - nemusí sa totiž vykonať žiadne z rozšírení
 - aktor spúšťa pôvodný UC
- špecializácia
 - UC prispôbime konkrétnej situácii
 - pôvodný UC nemusí byť uzavretý a zmysluplný sám o sebe
 - bude mať prázdne miesta, ktoré vyplnia potomkovia
 - aktor spúšťa špecializované UC

Notácia

Vzťah	Notácia	Význam
<<extends>>		UC rozširuje správanie cieľového UC v danom extension pointe Dodatočné správanie do UC.
<<include>>		UC explicitne obsahuje správanie cieľového (vkladaného) UC. Zdieľanie správania medzi UC.
generalizácia		Zdrojový UC špecializuje cieľový UC. Vytváranie všeobecných rámcových UC, ktoré dávajú priestor na prispôbenie systému