Princípy softvérového inžinierstva
http://www2.fiit.stuba.sk/~lang/PSI/

Softvérové procesy

doc. Ing. Ján Lang, PhD.

Ústav informatiky, informačných systémov a softvérového inžinierstva (UISI)
Fakulta informatiky a informačných technológií
Slovenská technická univerzita v Bratislave
Ilkovičova 2, 84216 Bratislava 4
jan.lang@stuba.sk

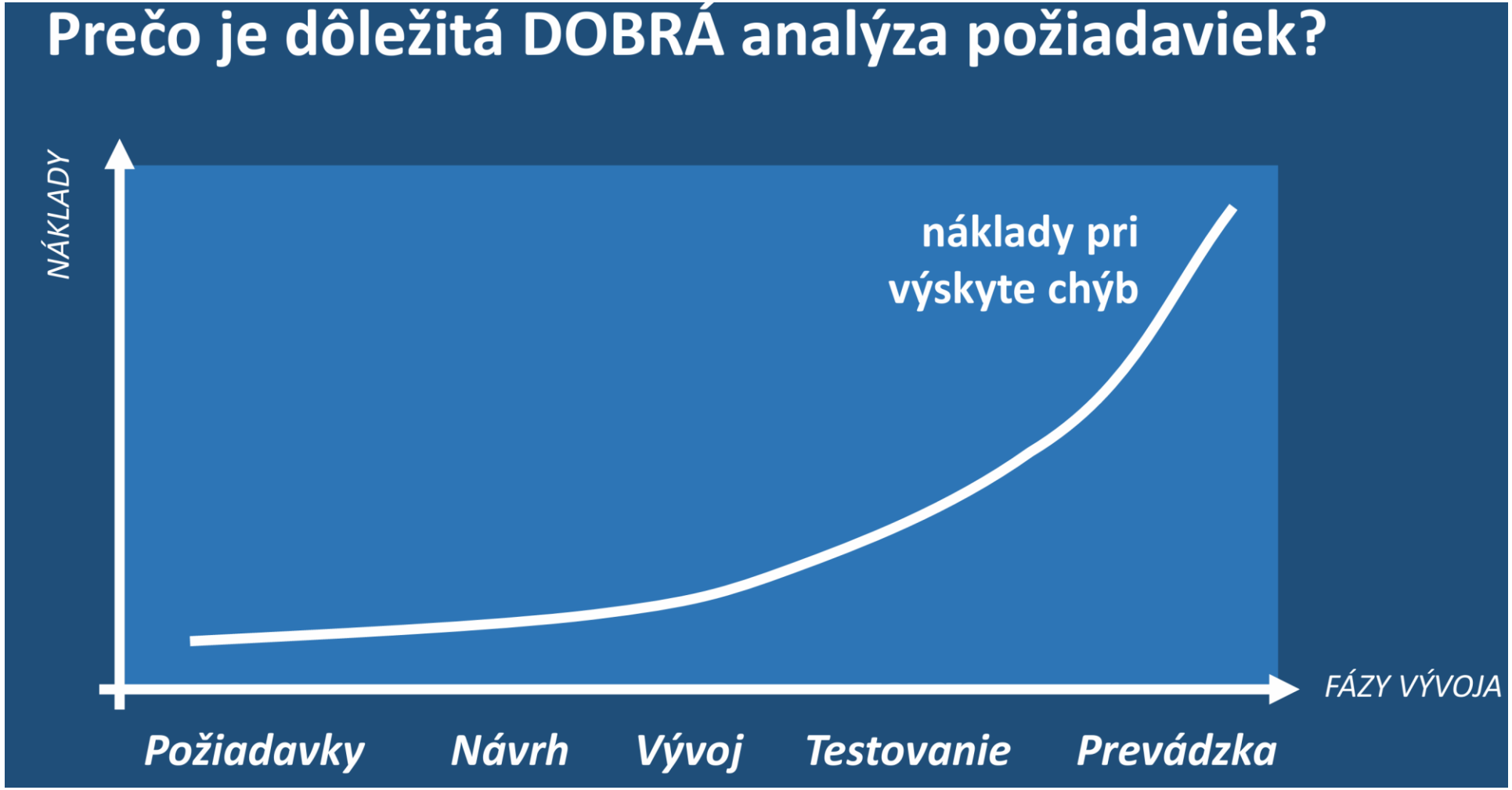
26.02.2026

Prípad použitia: Zadaj objednávku ?

Tu sme sa dostali naposledy...

Koľko požiadaviek ste si stihli zaznamenať? SLIDO

Životný cyklus softvéru:
Analýza, špecifikácia požiadaviek



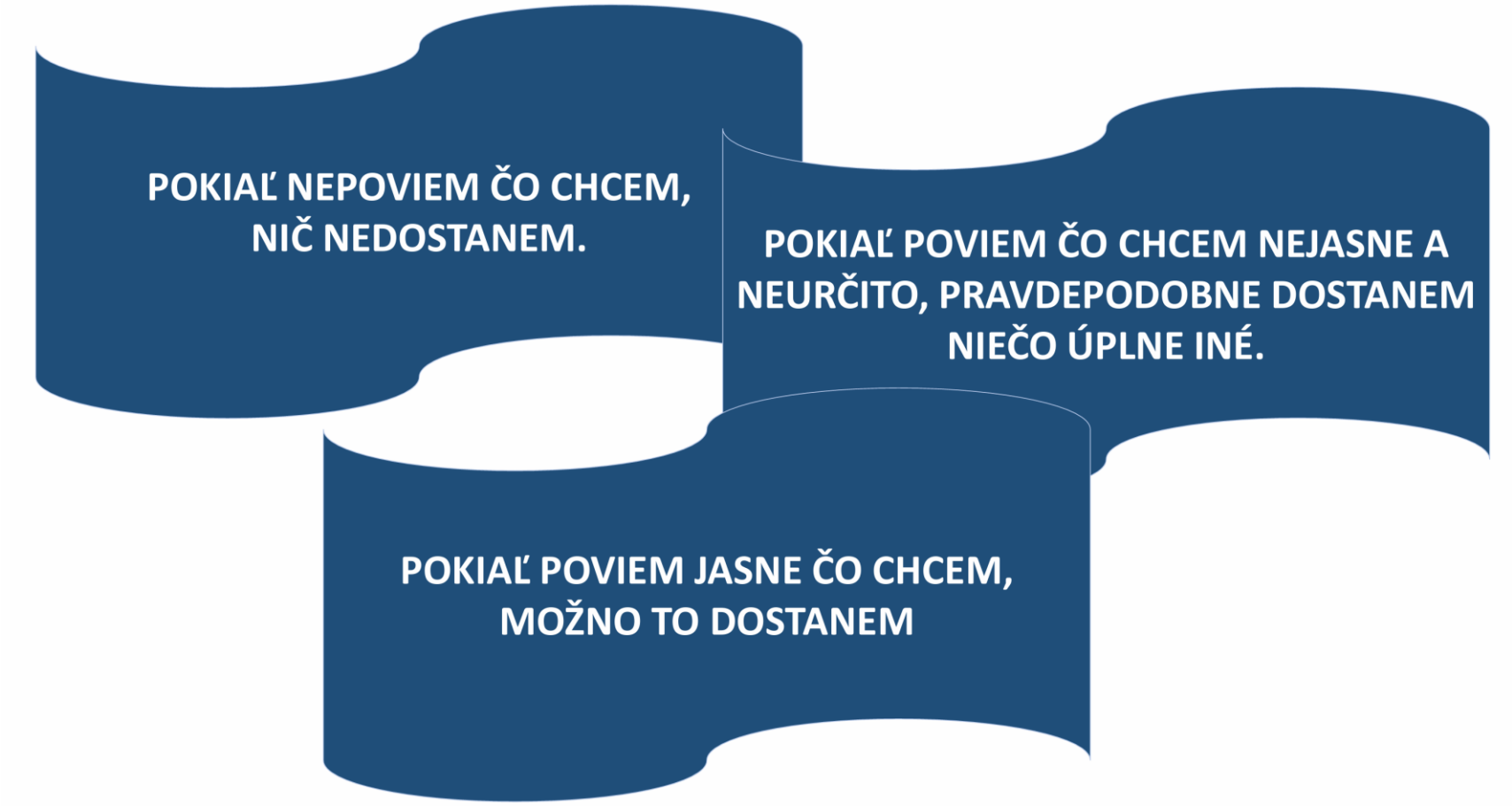
Životný cyklus softvéru:
Analýza, špecifikácia požiadaviek

Chyby v rámci projektov – odkiaľ sa berú?

Aké sú dôvody pre tieto chyby?

- Nejasné alebo nejednoznačné požiadavky
- Niektoré požiadavky nie sú známe Aké sú dôvody pre tieto chyby?
- Nejasná kompetencia zadávateľov
- Zabudnutie alebo ignorovanie na používateľov
- Požiadavky nie sú prioritizované
- Požiadavky nie sú implementované
- Požiadavky sú zle implementované
- Požiadavky sú neskoro implementované

Príklad hojdačky z minulej prednášky...



Kto je stakeholder?

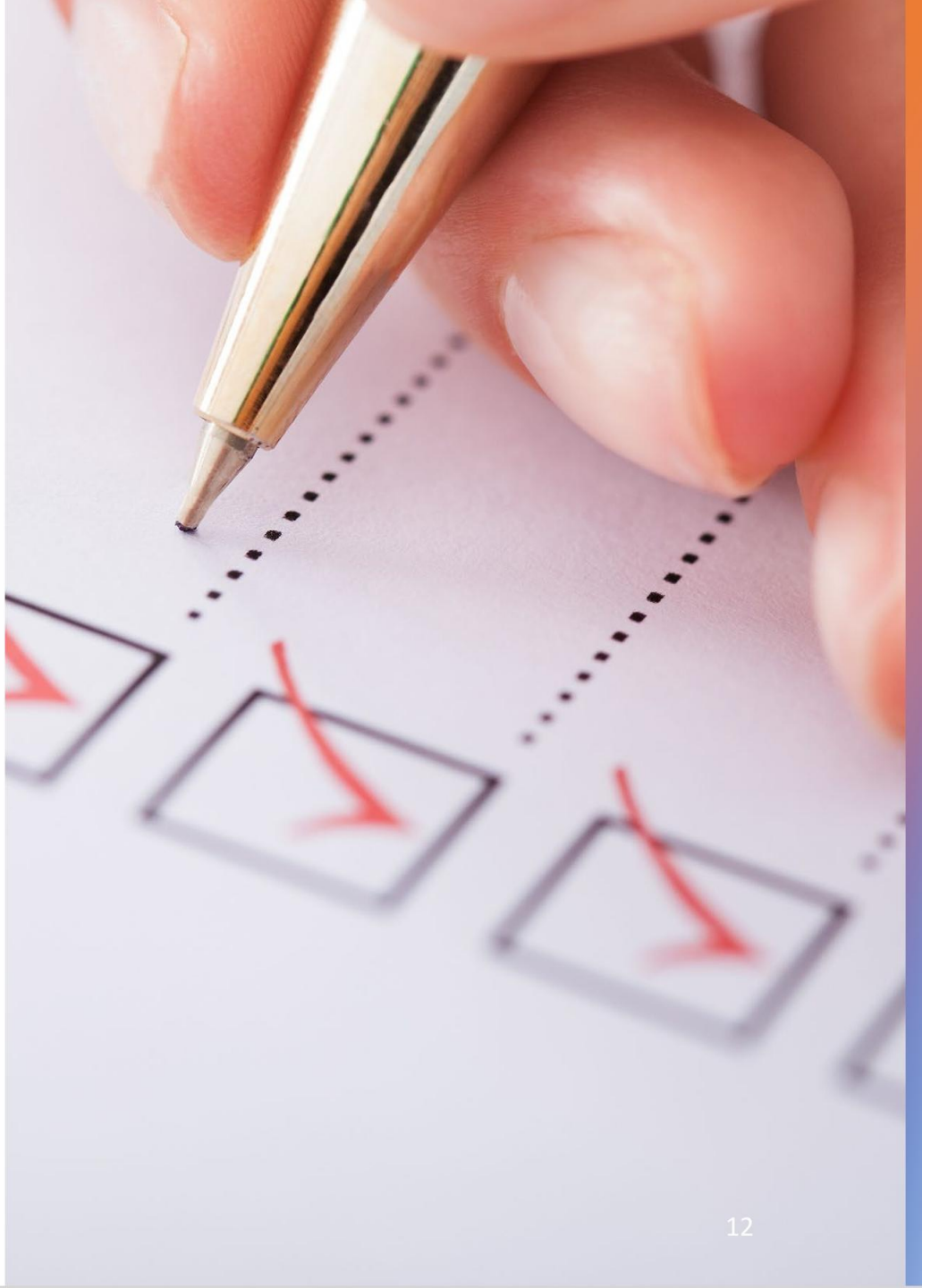
Kto je stakeholder?

zúčastnená osoba alebo organizácia, ktorá má na produkt vplyv alebo jej účasť je pri vývoji produktu nevyhnutná.

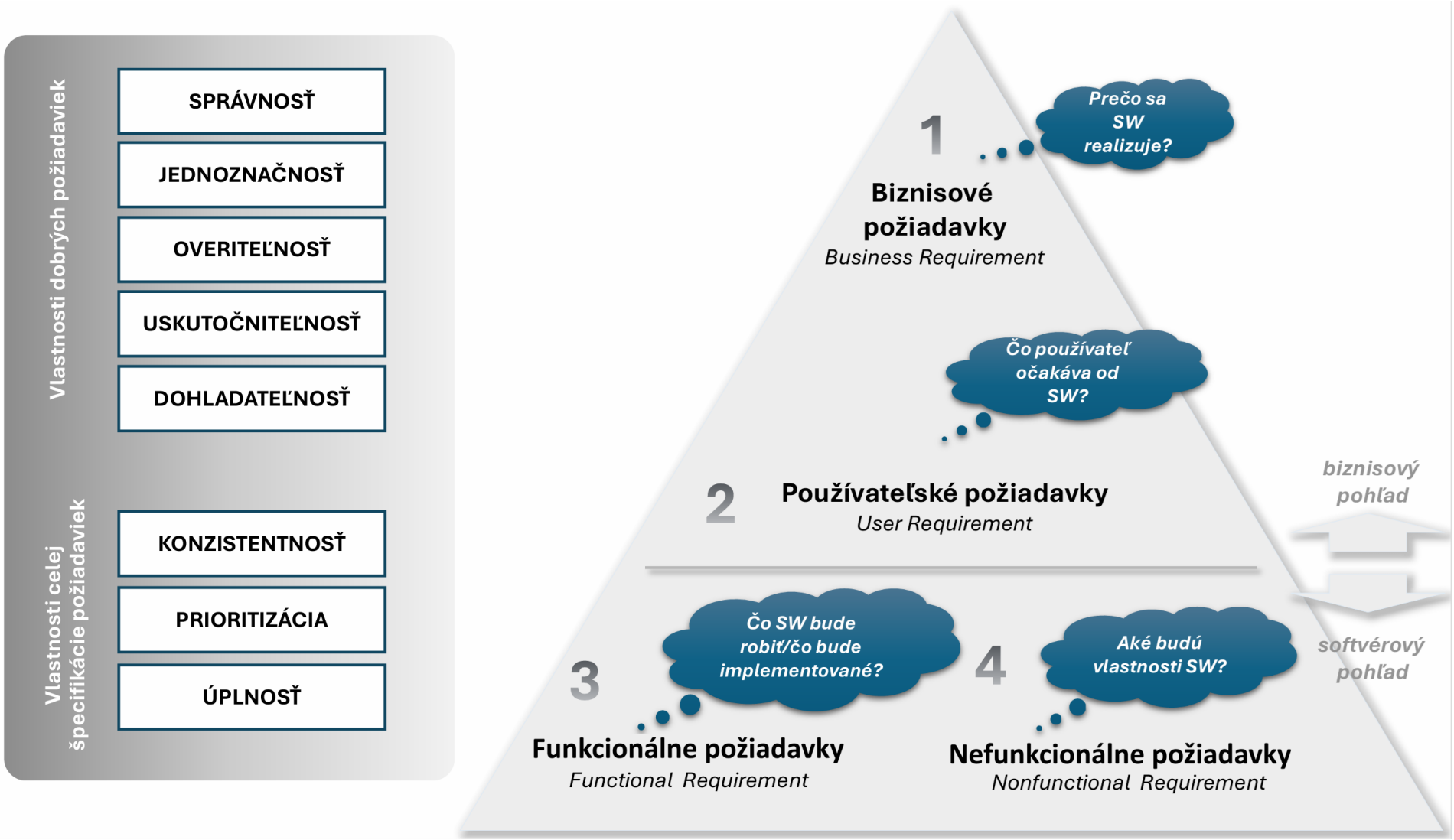
Bez identifikácie zúčastnených osôb Vám budú chýbať jednotlivé požiadavky.

Identifikácia zúčastnených osôb výrazne ovplyvňuje úspech projektu!

„Kto je za požiadavkou na tento produkt?“
„Kto bude riešenie používať?“



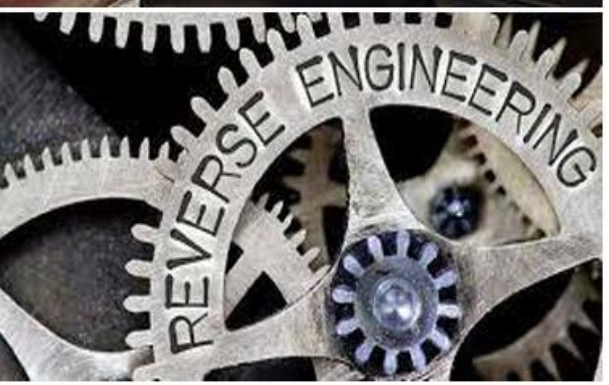
Model požiadaviek (Requirement model)



Spôsob získavania požiadaviek

Dokumentácia existujúceho stavu resp. konkurenčné systémy

Sledovanie používateľov pri práci



Reverse engineering existujúcich systémov

Rozhovory a diskusie

Prieskum trhu a používateľské dotazníky

Prototypovanie

Diskusia „umenie viesť ľudí ku spoločným cieľom spôsobom, ktorý podporuje zapojenie, účasť a produktivitu všetkých zúčastnených“ (Sibbet)
Kniha: Requirements by Collaboration. Ellen Gottesdiener 2002

Spôsob získavania požiadaviek

Najzložitejšia, najdôležitejšia a komunikačne najnáročnejšia časť vývoja softvéru!

Pýtať sa „Prečo?“ nestačí. Je potrebné:

- Klásť ***doplňovacie otázky*** pomôžu lepšie zistiť čo má systém robiť.
- Zistiť rôzne ***varianty*** užívateľských postupov.
- Zistiť ***výnimky*** (čoby sa stalo keby... Odkiaľ máte .... Prečo práve takto ...robí to niekto takto...)
- Hľadať funkcie, ktoré používatelia ***očakávajú, ale nepovedia.***
- Zapisovať si zdroj (zákazníka) požiadaviek.

Analýza požiadaviek

Analýzou požiadaviek sa rozumie upresňovanie požiadaviek, hľadanie chýb, nedostatkov a rozpracovanie požiadaviek do podrobností.

Súčasťou je:

- analyzovanie realizovateľnosti požiadaviek
- rozdelenie požiadaviek podľa priorít
- grafický model (stavové diagramy, diagramy aktivít, rozhodovacie stromy,...)
- textový popis

Cieľ – vyvinúť dostatočne kvalitné a podrobné požiadavky, podľa ktorých je možné sa pustiť do vývoja a testovania SW.

Vlastnosti dobrých požiadaviek

SPRÁVNOSŤ

Funkcionalita musí byť popísaná presne, tak aby spĺňala potreby používateľa.

Správnosť overuje iba používateľ a preto musia prejsť kontrolou.

JEDNOZNAČNOSŤ

Pri čítaní požiadavky by každý čitateľ mal prísť k jedinému výkladu.

Preto je potrebné požiadavky popisovať jednoduchým, stručným a zrozumiteľným jazykom.

Adekvátny
Mal by
Môže
Napríklad
Alebo
Prinajmenšom
Praktický
Úsporný
Rýchly
Lepší

SW by mal pravidelne report o transakciách

Vytvorte prihlásenia do systému.

Toto je testovacia správa. Pokiaľ ste túto správu nedostali, kontaktujte nás na telefónnom čísle xxx

Vlastnosti dobrých požiadaviek

OVERITEĽNOSŤ	USKUTOČNITEĽNOSŤ	DOHLADATEĽNOSŤ
Správna implementácia požiadavky je overiteľná nejakým testom, nástrojom. Overiť sa nedajú neúplne, nekonzistentné, neuskutočniteľné a nejednoznačné požiadavky	Každá požiadavka sa musí dať realizovať v rámci známych možností a obmedzení. Vhodne sa dá predchádzať inkrementálnym vývojom alebo agilným.	Vysledovanie zdroja požiadavky, cez zdrojový kód až po testovací scenár. Každá požiadavka má mať jedinečný identifikátor, popísaná štruktúrovaným a čitateľným textom.

SW musí byť užívateľsky prívetivý.

SW musí odpovedať na všetky používateľské požiadavky okamžite a bez oneskorenia.

Vlastnosti celej špecifikácie požiadaviek

ÚPLNOSŤ

Žiadne požiadavky a ani informácie by nemali chýbať.

Požiadavky pokrývajú všetky potreby a ciele zákazníka, ktoré chce SW dosiahnuť.

Vyhnúť sa dá sústredením sa na používateľské úlohy.

KONZISTENTNOSŤ

Požiadavky neprotirečia iným. Nezhody je potrebné vyriešiť.

„Používateľ si môže zmeniť heslo v sekcii Nastavenia “

„Heslo používateľa je pevne nastavené a nemeniteľné.“

PRIORITIZÁCIA

Nie všetko ma najvyššiu prioritu.

Porozumieť dôležitosti a urgentnosti.

Priorita indikuje dôležitosť pre zákazníka a umožňuje:

- pracovať na správnych veciach
- prijať rozhodnutia

Ešte k požiadavkám

Špecifikácia požiadaviek NIE JE návrh systému.
Obsahuje **ČO** by mal IS robiť a nie **AKO.**

Vyhnúť sa **IT žargónu.**

Vysvetlite **PREČO** je požiadavka nevyhnutná.

Prípady použitia (Use Cases) Slido

Prípady použitia sú spôsobom zaznamenania (špecifikácie) funkcionálnych požiadaviek, ale nie sú úplnou špecifikáciou všetkých požiadaviek systému.

Prípady použitia zachytávajú:

- Funkcionálne požiadavky z pohľadu účastníka, používateľa, aktéra, ciele používateľa voči systému
- Interakcie medzi aktérom a systémom, hlavný scenár a alternatívne priebehy, Reakcie systému na vstupy

Prípady použitia nezachytávajú úplne

- Nefunkcionálne požiadavky (výkon, bezpečnosť, dostupnosť...)
- Architektúru systému, detailný návrh používateľského rozhrania
- Interné algoritmy a implementáciu

Prípady použitia (Use Cases)

Use Case je špecifikáciou správania (napr. Zadaj objednávku PSI/P1).

A **Use Case specifies a set of actions performed by its subjects, which yields an observable result that is of value** for one or more Actors or other stakeholders of each subject – UML 2.5.1 Specification (https://www.omg.org/spec/UML).

Napriek existujúcemu štandardu, detailný opis len na úrovni **notácií**.

Prípady použitia (Use Cases) - Notácia

Notácia: Konvencie, prvky, **systém formálnych znakov, symbolov a pravidiel**, pomocou ktorých sa vyjadrujú informácie, pojmy alebo vzťahy.

Ivar Jacobson: zaviedol koncept, metodický rámec, staršia.

Alistair Cockburn: metodicky prepracoval a formalizoval, *„one column of text (not a table), numbered steps, no if statements, the numbering convention in the extensions“*, budeme používať.

ABOUT THE UNIFIED MODELING LANGUAGE SPECIFICATION VERSION 2.5.1

UML® — Unified Modeling Language

A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems.

PDF Specification

Title: Unified Modeling Language
Acronym: UML®
Version: 2.5.1
Document Status: formal ⓘ
Publication Date: December 2017
Categories: *Modeling* *Software Engineering* *Platform*
IPR Mode ⓘ RF-Limited ⓘ

TABLE OF CONTENTS

UC: BUY SOMETHING (CASUAL VERSION)

The Requestor initiates a request and sends it to her or his Approver. The Approver checks that there is money in the budget, check the price of the goods, completes the request for submission, and sends it to the Buyer. The Buyer checks the contents of storage, finding best vendor for goods. Authorizer: validate Approver's signature. Buyer: complete request for ordering, initiate Purchase Order with Vendor. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design). Receiver: register delivery, send goods to Requestor. Requestor: mark request delivered. At any time prior to receiving goods, Requestor can change or cancel the request. Canceling it removes it from any active processing. (delete from system?) Reducing the price leaves it intact in process. Raising the price sends it back to Approver (Cockburn, 2000).

UC: TEMPLATE

Use Case: <number> <the name should be the goal as a short active verb phrase>

Characteristic Information
Context of use: <a longer statement of the goal, if needed, its normal occurrence conditions>
Scope: <design scope, what system is being considered black-box under design>
Level: <one of: Strategic, User-goal, Subfunction>
Primary Actor: <a role name for the primary actor, or description>
Stakeholders & Interests: <list of stakeholders and key interests in the use case>
Precondition: <what we expect is already the state of the world>
Success End Condition: <the state of the world upon successful completion>
Failed End Protection: <the state of the world if goal abandoned>
Trigger: <what starts the use case, may be time event>

Main Success Scenario
<put here the steps of the scenario from trigger to goal delivery, and any cleanup after>
<step #> <action description>

Extensions
<put here there extensions, one at a time, each referring to the step of the main scenario>
<step altered> <condition> : <action or sub-use case>
<step altered> <condition> : <action or sub-use case>

I/O Variations
<put here the variations that will cause eventual bifurcation in the scenario>
<step or variation # > <list of variations>
<step or variation # > <list of variations>

Related Information (Optional)
<whatever your project needs for additional information>

UC: FULLY DRESSED VERSION

USE CASE 5: BUY SOMETHING

Primary Actor: Requestor
Main Context: Requestor buys something through the system, gets it. Does not include pay-for it.
Scope: Business - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests:
Requestor: wants what he/she ordered, easy way to do that.
Company: wants to control spending but allow needed purchases.
Vendor: wants to get paid for any goods delivered.
Precondition: none
Minimal guarantees: Every order sent out has been approved by a valid authorizer. Order was tracked so that company can only be billed for valid goods received.
Success guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main success scenario:
1. Requestor: *initiate a request*
2. Approver: check money in the budget, check price of goods, *complete request for submission*
3. Buyer: check contents of storage, find best vendor for goods
4. Authorizer: *validate Approver's signature*
5. Buyer: *complete request for ordering, initiate PO with Vendor*
6. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design)
7. Receiver: *register delivery*, send goods to Requestor
8. Requestor: *mark request delivered.*

Extensions:
1a. Requestor does not know vendor or price: leave those parts blank and continue.
1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.
2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.
2b. Approver is not Requestor's manager: still ok, as long as approver signs
2c. Approver declines: send back to Requestor for change or deletion
3a. Buyer finds goods in storage: send those up, reduce request by that amount and carry on.
3b. Buyer fills in Vendor and price, which were missing; gets resent to Approver.

UC: TEMPLATE

Use Case: <number> <the name should be the goal as a short active verb phrase>

Characteristic Information

Context of use: <a longer statement of th[illegible]d, if needed, its normal occurrence conditions>
Scope: <design scope, what system is [illegible] considered black-box under design>
Level: <one of: Strategic, User-goal, Subfunction>
Primary Actor: <a role name for the primary actor, or description>
Stakeholders & Interests: <list of stakeholders and key interests in the use case>
Precondition: <what we expect is already the state of the world>
Success End Condition: <the state of the world upon successful completion>
Failed End Protection: <the state of the world if goal abandoned>
Trigger: <what starts the use case, may be time event>

Main Success Scenario

<put here the steps of the scenario from trigger to goal delivery, and any cleanup after>
<step #> <action description>

Extensions

<put here there extensions, one at a time, each referring to the step of the main scenario>
<step altered> <condition> : <action or sub-use case>
<step altered> <condition> : <action or sub-use case>

I/O Variations

<put here the variations that will cause eventual bifurcation in the scenario>
<step or variation # > <list of variations>
<step or variation # > <list of variations>

Related Information (Optional)

<whatever your project needs for additional information>

UC: FULLY DRESSED VERSION

USE CASE 5: BUY SOMETHING

Primary Actor: Requestor
Goal in Context: Requestor buys something through the system, gets it. Does not include paying for it.
Scope: Business - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests:
Requestor: wants what he/she ordered, easy way to do that.
Company: wants to control spending but allow needed purchases.
Vendor: wants to get paid for any goods delivered.
Precondition: none
Minimal guarantees: Every order sent out has been approved by a valid authorizer. Order was tracked so that company can only be billed for valid goods received.
Success guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main success scenario:
1. Requestor: *initiate a request*
2. Approver: check money in the budget, check price of goods, *complete request for submission*
3. Buyer: check contents of storage, find best vendor for goods
4. Authorizer: *validate Approver's signature*
5. Buyer: *complete request for ordering, initiate PO with Vendor*
6. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design)
7. Receiver: *register delivery*, send goods to Requestor
8. Requestor: *mark request delivered.*

Extensions:
1a. Requestor does not know vendor or price: leave those parts blank and continue.
1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.
2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.
2b. Approver is not Requestor's manager: still ok, as long as approver signs
2c. Approver declines: send back to Requestor for change or deletion
3a. Buyer finds goods in storage: send those up, reduce request by that amount and carry on.
3b. Buyer fills in Vendor and price, which were missing; gets resent to Approver.

UC: TEMPLATE

Use Case: <number> <the name should be the goal as a short active verb phrase>

Characteristic Information

Context of use: <a longer statement of the goal, if needed, its normal occurrence conditions>
Scope: <design scope, what system is being considered black-box under design>
Level: <one of: Strategic, User-goal, Subfunction>
Primary Actor: <a role name for the primary actor, or description>
Stakeholders & Interests: <list of stakeholders and key interests in the use case>
Precondition: <what we expect is already the state of the world>
Success End Condition: <the state of the world upon successful completion>
Failed End Protection: <the state of the world if goal abandoned>
Trigger: <what starts the use case, may be time event>

Main Success Scenario

<put here the steps of the scenario from trigger to goal delivery, and any cleanup after>
<step #> <action description>

Extensions

<put here there extensions, one at a time, each referring to the step of the main scenario>
<step altered> <condition> : <action or sub-use case>
<step altered> <condition> : <action or sub-use case>

I/O Variations

<put here the variations that will cause eventual bifurcation in the scenario>
<step or variation # > <list of variations>
<step or variation # > <list of variations>

Related Information (Optional)

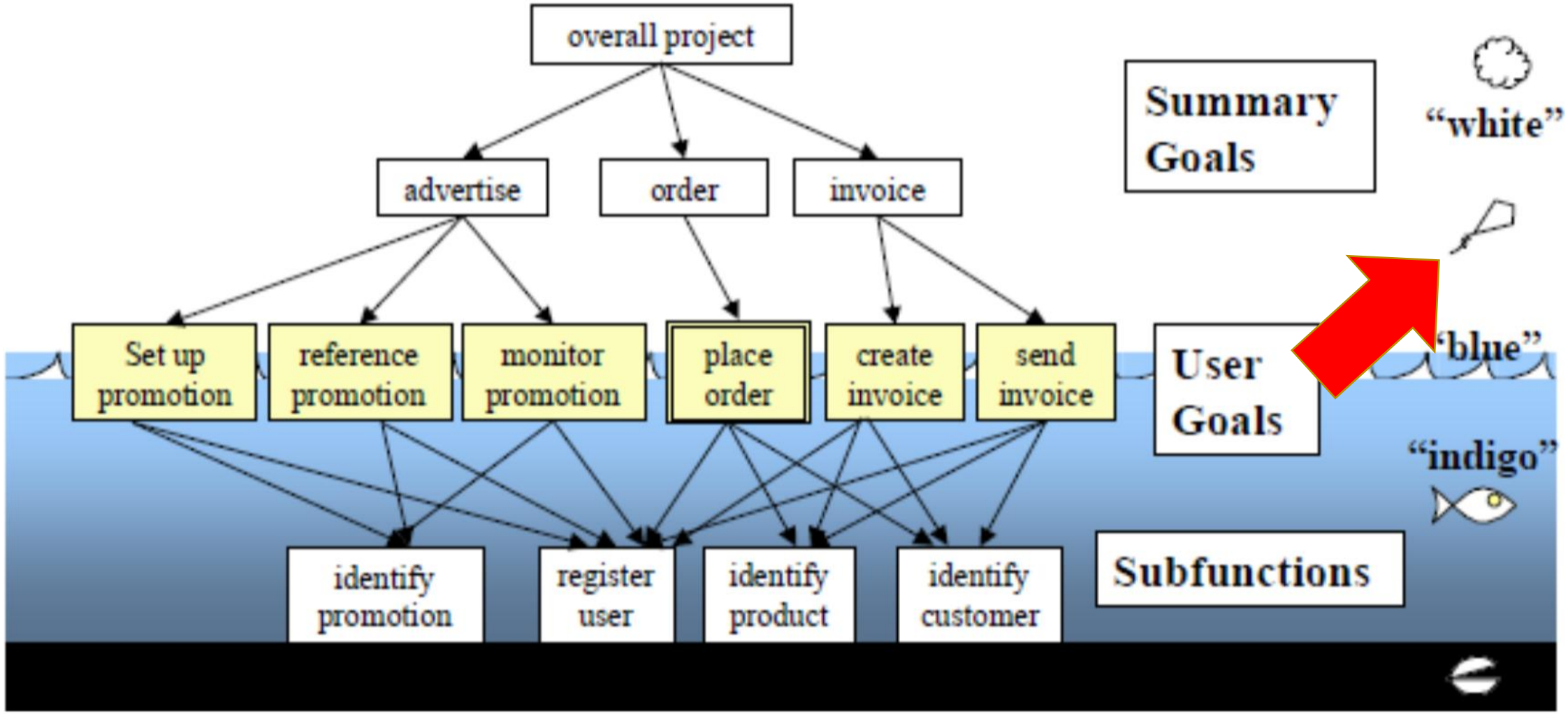
<whatever your project needs for additional information>

UC: FULLY DRESSED VERSION

USE CASE 5: BUY SOMETHING

Primary Actor: Requestor
Goal in Context: Requestor buys something through the system, gets it. Does not include paying for it.
Scope: Business - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests:
Requestor: wants what he/she ordered, easy way to do that.
Company: wants to control spending but allow needed purchases.
Vendor: wants to get paid for any goods delivered.
Precondition: none
Minimal guarantees: Every order sent out has been approved by a valid authorizer. Order was tracked so that company can only be billed for valid goods received.
Success guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main success scenario:
1. Requestor: *initiate a request*
2. Approver: check money in the budget, check price of goods, *complete request for submission*
3. Buyer: check contents of storage, find best vendor for goods
4. Authorizer: *validate Approver's signature*
5. Buyer: *complete request for ordering, initiate PO with Vendor*
6. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design)
7. Receiver: *register delivery*, send goods to Requestor
8. Requestor: *mark request delivered.*

Extensions:
1a. Requestor does not know vendor or price: leave those parts blank and continue.
1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.
2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.
2b. Approver is not Requestor's manager: still ok, as long as approver signs
2c. Approver declines: send back to Requestor for change or deletion
3a. Buyer finds goods in storage: send those up, reduce request by that amount and carry on.
3b. Buyer fills in Vendor and price, which were missing; gets resent to Approver.



A. Cockburn. Writing Effective Use Cases. Addison–Wesley, 2000.

Prípady použitia (Use Cases)

Focuses on description the essential features of the system, in a text form. Use cases **are fundamentally a text** form.

UC opisom toho ako používať systém. Nie toho ako systém pracuje, ako rozmýšľa, uvažuje...

A use case describes how the system responds to a request from one of the stakeholders, called the **primary actor**.

The **primary actor** initiates an interaction with the system to accomplish some goal.

E CASE 5: BUY SOMETHING

Primary Actor: Requestor
Goal in Context: Requestor buys something through the system, gets it. Does not include paying for it.
Scope: Business - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests:
Requestor: wants what he/she ordered, easy way to do that.
Company: wants to control spending but allow needed purchases.
Vendor: wants to get paid for any goods delivered.
Precondition: none
Minimal guarantees: Every order sent out has been approved by a valid authorizer. Order was tracked so that company can only be billed for valid goods received.
Success guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main success scenario:
1. Requestor: *initiate a request*
2. Approver: check money in the budget, check price of goods, *complete request for submission*
3. Buyer: check contents of storage, find best vendor for goods
4. Authorizer: *validate Approver's signature*
5. Buyer: *complete request for ordering, initiate PO with Vendor*
6. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design)
7. Receiver: *register delivery*, send goods to Requestor
8. Requestor: *mark request delivered.*

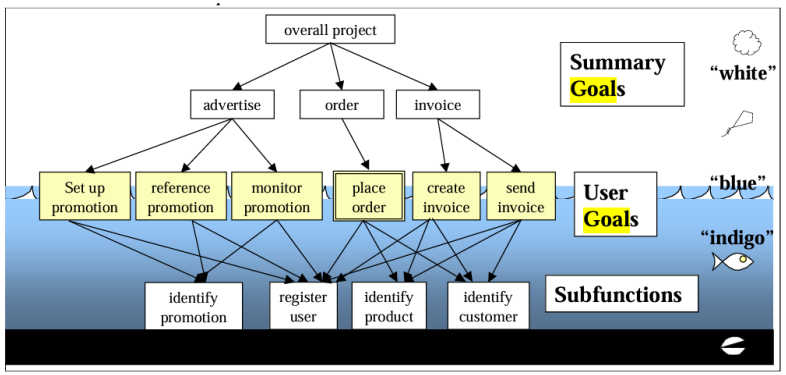
Extensions:
1a. Requestor does not know vendor or price: leave those parts blank and continue.
1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.
2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.
2b. Approver is not Requestor's manager: still ok, as long as approver signs
2c. Approver declines: send back to Requestor for change or deletion
3a. Buyer finds goods in storage: send those up, reduce request by that amount and carry on.
3b. Buyer fills in Vendor and price, which were missing; gets resent to Approver.

Prípady použitia (Use Cases)

Actors have **goals.** Users of the system.

That is why: Alistar's **Three named goal levels**.

The **user goal** is the primary actor has in using the system.



The use case set reveals a **hierarchy of goals**, the ever unfolding story an the Scope.

SE CASE 5: BUY SOMETHING

Primary Actor: Requestor
Goal in Context: Requestor buys something through the system, gets it. Does not include paying for it.
Scope: Business - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests:
Requestor: wants what he/she ordered, easy way to do that.
Company: wants to control spending but allow needed purchases.
Vendor: wants to get paid for any goods delivered.
Precondition: none
Minimal guarantees: Every order sent out has been approved by a valid authorizer. Order was tracked so that company can only be billed for valid goods received.
Success guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main success scenario:
1. Requestor: *initiate a request*
2. Approver: check money in the budget, check price of goods, *complete request for submission*
3. Buyer: check contents of storage, find best vendor for goods
4. Authorizer: *validate Approver's signature*
5. Buyer: *complete request for ordering, initiate PO with Vendor*
6. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design)
7. Receiver: *register delivery*, send goods to Requestor
8. Requestor: *mark request delivered.*

Extensions:
1a. Requestor does not know vendor or price: leave those parts blank and continue.
1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.
2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.
2b. Approver is not Requestor's manager: still ok, as long as approver signs
2c. Approver declines: send back to Requestor for change or deletion
3a. Buyer finds goods in storage: send those up, reduce request by that amount and carry on.
3b. Buyer fills in Vendor and price, which were missing: gets resent to Approver.

Prípady použitia (Use Cases)

Scope (rozsah) určuje, čo je vnútri systému, ktorý navrhujeme, a čo je mimo systému
Scope určuje hranice modelovania.

„Black-box“: systém sa posudzuje zvonku, opisujeme jeho správanie (reakcie na vstupy), neriešime vnútornú implementáciu.

Používame pohľad:

„Čo systém robí?“
nie
„Ako to interne robí?“

USE CASE 5: BUY SOMETHING

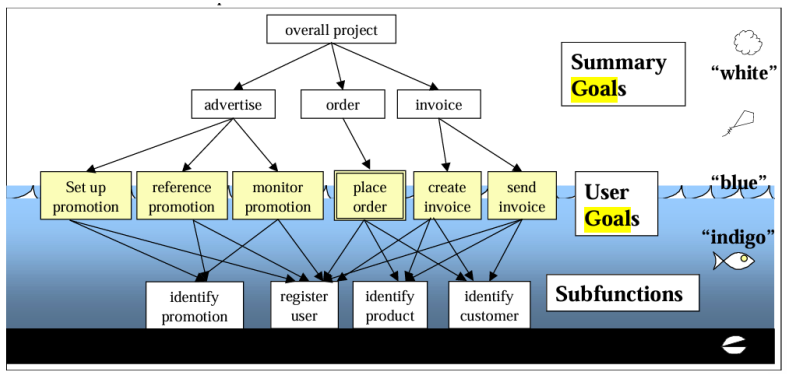
Primary Actor: Requestor
Goal in Context: Requestor buys something through the system, gets it. Does not include paying for it.
Scope: Business - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests:
Requestor: wants what he/she ordered, easy way to do that.
Company: wants to control spending but allow needed purchases.
Vendor: wants to get paid for any goods delivered.
Precondition: none
Minimal guarantees: Every order sent out has been approved by a valid authorizer. Order was tracked so that company can only be billed for valid goods received.
Success guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main success scenario:
1. Requestor: *initiate a request*
2. Approver: check money in the budget, check price of goods, *complete request for submission*
3. Buyer: check contents of storage, find best vendor for goods
4. Authorizer: *validate Approver's signature*
5. Buyer: *complete request for ordering, initiate PO with Vendor*
6. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design)
7. Receiver: *register delivery*, send goods to Requestor
8. Requestor: *mark request delivered.*

Extensions:
1a. Requestor does not know vendor or price: leave those parts blank and continue.
1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.
2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.
2b. Approver is not Requestor's manager: still ok, as long as approver signs
2c. Approver declines: send back to Requestor for change or deletion
3a. Buyer finds goods in storage: send those up, reduce request by that amount and carry on.
3b. Buyer fills in Vendor and price, which were missing: gets resent to Approver.

Prípady použitia (Use Cases)

A level communicates the level of abstraction

That is why: Alistar's **Three named goal levels**.



USE CASE 5: BUY SOMETHING

Primary Actor: Requestor
Goal in Context: Requestor buys something through the system, gets it. Does not include paying for it.
Scope: Business - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests:
Requestor: wants what he/she ordered, easy way to do that.
Company: wants to control spending but allow needed purchases.
Vendor: wants to get paid for any goods delivered.
Precondition: none
Minimal guarantees: Every order sent out has been approved by a valid authorizer. Order was tracked so that company can only be billed for valid goods received.
Success guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main success scenario:
1. Requestor: *initiate a request*
2. Approver: check money in the budget, check price of goods, *complete request for submission*
3. Buyer: check contents of storage, find best vendor for goods
4. Authorizer: *validate Approver's signature*
5. Buyer: *complete request for ordering, initiate PO with Vendor*
6. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design)
7. Receiver: *register delivery*, send goods to Requestor
8. Requestor: *mark request delivered.*

Extensions:
1a. Requestor does not know vendor or price: leave those parts blank and continue.
1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.
2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.
2b. Approver is not Requestor's manager: still ok, as long as approver signs
2c. Approver declines: send back to Requestor for change or deletion
3a. Buyer finds goods in storage: send those up, reduce request by that amount and carry on.
3b. Buyer fills in Vendor and price, which were missing: gets resent to Approver.

Prípady použitia (Use Cases)

Depicts secondary actors, Vendors and their interests.

USE CASE 5: BUY SOMETHING

Primary Actor: Requestor
Goal in Context: Requestor buys something through the system, gets it. Does not include paying for it.
Scope: Business - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests:
Requestor: wants what he/she ordered, easy way to do that.
Company: wants to control spending but allow needed purchases.
Vendor: wants to get paid for any goods delivered.
Precondition: none
Minimal guarantees: Every order sent out has been approved by a valid authorizer. Order was tracked so that company can only be billed for valid goods received.
Success guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main success scenario:
1. Requestor: *initiate a request*
2. Approver: check money in the budget, check price of goods, *complete request for submission*
3. Buyer: check contents of storage, find best vendor for goods
4. Authorizer: *validate Approver's signature*
5. Buyer: *complete request for ordering, initiate PO with Vendor*
6. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design)
7. Receiver: *register delivery*, send goods to Requestor
8. Requestor: *mark request delivered.*

Extensions:
1a. Requestor does not know vendor or price: leave those parts blank and continue.
1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.
2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.
2b. Approver is not Requestor's manager: still ok, as long as approver signs
2c. Approver declines: send back to Requestor for change or deletion
3a. Buyer finds goods in storage: send those up, reduce request by that amount and carry on.
3b. Buyer fills in Vendor and price, which were missing; gets resent to Approver.

Prípady použitia (Use Cases)

The use case describes the system's behavior under various conditions as it responds to a request from one of the stakeholders.

Precondition, vstupná podmienka, minimal guarantees...

USE CASE 5: BUY SOMETHING

Primary Actor: Requestor
Goal in Context: Requestor buys something through the system, gets it. Does not include paying for it.
Scope: Business - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests:
Requestor: wants what he/she ordered, easy way to do that.
Company: wants to control spending but allow needed purchases.
Vendor: wants to get paid for any goods delivered.
Precondition: none
Minimal guarantees: Every order sent out has been approved by a valid authorizer. Order was tracked so that company can only be billed for valid goods received.
Success guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main success scenario:
1. Requestor: *initiate a request*
2. Approver: check money in the budget, check price of goods, *complete request for submission*
3. Buyer: check contents of storage, find best vendor for goods
4. Authorizer: *validate Approver's signature*
5. Buyer: *complete request for ordering, initiate PO with Vendor*
6. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design)
7. Receiver: *register delivery*, send goods to Requestor
8. Requestor: *mark request delivered.*

Extensions:
1a. Requestor does not know vendor or price: leave those parts blank and continue.
1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.
2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.
2b. Approver is not Requestor's manager: still ok, as long as approver signs
2c. Approver declines: send back to Requestor for change or deletion
3a. Buyer finds goods in storage: send those up, reduce request by that amount and carry on.
3b. Buyer fills in Vendor and price, which were missing; gets resent to Approver.

Prípady použitia (Use Cases)

UC sú ucelené úlohy, ktoré si používateľ kladie za cieľ a ktoré **po ich vykonaní spôsobujú zmenu na strane systému alebo používateľa**.

Postcondition, výstupná podmienka, guarantees of success, success guarantees...

Merateľná zmena!

Each requirement must be necessary, verifiable, attainable, and clear in order to be useful for development.

USE CASE 5: BUY SOMETHING

Primary Actor: Requestor
Goal in Context: Requestor buys something through the system, gets it. Does not include paying for it.
Scope: Business - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests:
Requestor: wants what he/she ordered, easy way to do that.
Company: wants to control spending but allow needed purchases.
Vendor: wants to get paid for any goods delivered.
Precondition: none
Minimal guarantees: Every order sent out has been approved by a valid authorizer. Order was tracked so that company can only be billed for valid goods received.
Success guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main success scenario:
1. Requestor: *initiate a request*
2. Approver: check money in the budget, check price of goods, *complete request for submission*
3. Buyer: check contents of storage, find best vendor for goods
4. Authorizer: *validate Approver's signature*
5. Buyer: *complete request for ordering, initiate PO with Vendor*
6. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design)
7. Receiver: *register delivery*, send goods to Requestor
8. Requestor: *mark request delivered.*

Extensions:
1a. Requestor does not know vendor or price: leave those parts blank and continue.
1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.
2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.
2b. Approver is not Requestor's manager: still ok, as long as approver signs
2c. Approver declines: send back to Requestor for change or deletion
3a. Buyer finds goods in storage: send those up, reduce request by that amount and carry on.
3b. Buyer fills in Vendor and price, which were missing: gets resent to Approver.

Prípady použitia (Use Cases)

Main Success Scenario, Happy Day Scenario, Main Flow, Základný tok, **Basic flow,...**

Základný, ideálny priebeh prípadu použitia, v ktorom všetko prebehne bez chýb a výnimiek a cieľ aktéra je úspešne dosiahnutý.

Je uvedený/napísaný ako sekvencia krokov medzi aktérom a systémom.

USE CASE 5: BUY SOMETHING

Primary Actor: Requestor
Goal in Context: Requestor buys something through the system, gets it. Does not include paying for it.
Scope: Business - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests:
Requestor: wants what he/she ordered, easy way to do that.
Company: wants to control spending but allow needed purchases.
Vendor: wants to get paid for any goods delivered.
Precondition: none
Minimal guarantees: Every order sent out has been approved by a valid authorizer. Order was tracked so that company can only be billed for valid goods received.
Success guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main success scenario:
1. Requestor: *initiate a request*
2. Approver: check money in the budget, check price of goods, *complete request for submission*
3. Buyer: check contents of storage, find best vendor for goods
4. Authorizer: *validate Approver's signature*
5. Buyer: *complete request for ordering, initiate PO with Vendor*
6. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design)
7. Receiver: *register delivery*, send goods to Requestor
8. Requestor: *mark request delivered.*

Extensions:
1a. Requestor does not know vendor or price: leave those parts blank and continue.
1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.
2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.
2b. Approver is not Requestor's manager: still ok, as long as approver signs
2c. Approver declines: send back to Requestor for change or deletion
3a. Buyer finds goods in storage: send those up, reduce request by that amount and carry on.
3b. Buyer fills in Vendor and price, which were missing; gets resent to Approver.

Prípady použitia (Use Cases)

UC ponúka aj odpovede na otázku čo v prípade, že Main success scenario nenastane.

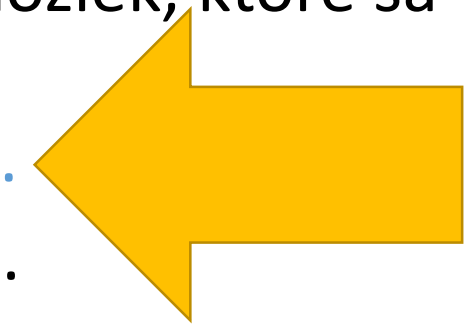
USE CASE 5: BUY SOMETHING

Primary Actor: Requestor
Goal in Context: Requestor buys something through the system, gets it. Does not include paying for it.
Scope: Business - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests:
Requestor: wants what he/she ordered, easy way to do that.
Company: wants to control spending but allow needed purchases.
Vendor: wants to get paid for any goods delivered.
Precondition: none
Minimal guarantees: Every order sent out has been approved by a valid authorizer. Order was tracked so that company can only be billed for valid goods received.
Success guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main success scenario:
1. Requestor: *initiate a request*
2. Approver: check money in the budget, check price of goods, *complete request for submission*
3. Buyer: check contents of storage, find best vendor for goods
4. Authorizer: *validate Approver's signature*
5. Buyer: *complete request for ordering, initiate PO with Vendor*
6. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design)
7. Receiver: *register delivery*, send goods to Requestor
8. Requestor: *mark request delivered.*

Extensions:
1a. Requestor does not know vendor or price: leave those parts blank and continue.
1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.
2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.
2b. Approver is not Requestor's manager: still ok, as long as approver signs
2c. Approver declines: send back to Requestor for change or deletion
3a. Buyer finds goods in storage: send those up, reduce request by that amount and carry on.
3b. Buyer fills in Vendor and price, which were missing; gets resent to Approver.

Use Case Modularization and Expressing It in UML

1. Systém musí umožniť zákazníkovi vyhľadať výrobok.
2. Systém musí umožniť zákazníkovi nastaviť počet položiek, ktoré sa naraz zobrazujú. . .
3. Systém musí umožniť zákazníkovi objednať výrobok.
4. Systém musí umožniť zákazníkovi zrušiť objednávku.



...

99. Systém musí umožňovať expedovať objednávku.

...

125. Do systému musí byť možné zadať nové druhy výrobku.

Use Case Modularization

Ak by sme mali e-obchod rýchlo sprevádzkovať, ktorý z prípadov použitia by sme mali realizovať ako prvý?

Bez možnosti objednávania výrobku profit neprinesie.

Výrobky zo začiatku môžu byť prezentované v zozname, ktorý zadáme napevno.

Vyhľadávanie vynecháme, expedovanie si poznačíme do externej tabuľky...

Určite to nie je ten najlepší e-obchod, ale môže začať zarábať!

Prípad použitia: Zadaj objednávku

Zákazník vyberá výrobok do košíka. Ten sa stáva súčasťou jeho objednávky, ktorú nakoniec potvrdí, čím sa objednávka presunie na expedovanie.

1. Zákazník zvolí zadanie objednávky.
2. Systém zobrazí možnosti vyhľadávania.
3. Zákazník nastaví možnosti vyhľadávania a spustí vyhľadávanie.
4. Systém zobrazí vyhľadané položky.
5. Zákazník vyberie z vyhľadaných položiek a potvrdí výber.
6. Systém vloží zvolený výrobok do košíka.
7. Zákazník môže pokračovať vo výbere výrobkov – prípad použitia pokračuje krokom 2.
8. Zákazník objedná výrobky v košíku.
9. Systém vyžiada údaje potrebné na realizáciu objednávky vrátane spôsobu platby.
10. Zákazník zadá požadované platobné údaje.
11. Kedykoľvek počas objednávania, zákazník môže vzdať tento proces.
12. Systém uloží objednávku do zoznamu objednávok na expedovanie.
13. Pre každý výrobok v objednávke, systém zistí stav zásob.
14. Systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu každého objednaného výrobku, ak by jeho stav po expedovaní poklesol pod stanovený limit.
15. Prípad použitia končí.

Predpoklady: zákazník je prihlásený

Dôsledky:

- Minimálne: výrobok, ktoré pôvodne boli súčasťou objednávky, sú v nej naďalej
- V prípade úspechu: výrobky, ktoré zákazník chce objednať sú súčasťou objednávky

Use Case Modularization

Názov prípadu použitia je v imperatíve (prikazovací spôsob), aby sme nezabudli, na to, že sledujeme dosiahnutie určitého cieľa.

Nech je akokoľvek výstižný, názov je dobré rozviesť v krátkom opise prípadu použitia.

Podstatou prípadu použitia je interakcia účastníka (angl. actor) so systémom, a tu je najjednoduchšie vyjadriť vo forme postupnosti krokov.

Prípad použitia: Zadaj objednávku

Zákazník vyberá výrobok do košíka. Ten sa stáva súčasťou jeho objednávky, ktorú nakoniec potvrdí, čím sa objednávka presunie na expedovanie.

1. Zákazník zvolí zadanie objednávky.
2. Systém zobrazí možnosti vyhľadávania.
3. Zákazník nastaví možnosti vyhľadávania a spustí vyhľadávanie.
4. Systém zobrazí vyhľadané položky.
5. Zákazník vyberie z vyhľadaných položiek a potvrdí výber.
6. Systém vloží zvolený výrobok do košíka.
7. Zákazník môže pokračovať vo výbere výrobkov – prípad použitia pokračuje krokom 2.
8. Zákazník objedná výrobky v košíku.
9. Systém vyžiada údaje potrebné na realizáciu objednávky vrátane spôsobu platby.
10. Zákazník zadá požadované platobné údaje.
11. Kedykoľvek počas objednávania, zákazník môže vzdať tento proces.
12. Systém uloží objednávku do zoznamu objednávok na expedovanie.
13. Pre každý výrobok v objednávke, systém zistí stav zásob.
14. Systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu každého objednaného výrobku, ak by jeho stav po expedovaní poklesol pod stanovený limit.
15. Prípad použitia končí.

Predpoklady: zákazník je prihlásený

Dôsledky:

- Minimálne: výrobok, ktoré pôvodne boli súčasťou objednávky, sú v nej naďalej
- V prípade úspechu: výrobky, ktoré zákazník chce objednať sú súčasťou objednávky

Use Case Modularization

Táto postupnosť krokov sa označuje ako tok udalostí alebo jednoducho iba tok.

Prvý krok vystihuje, čím prípad použitia začína.

Posledný krok predstavuje formálne ukončenie prípadu použitia.

V ostatných krokoch sa striedajú akcie na strane účastníka s akciami na strane sys tému.

Krok 7 (skok) a krok 11 (nadčasový charakter)

Prípad použitia: Zadaj objednávku

Zákazník vyberá výrobok do košíka. Ten sa stáva súčasťou jeho objednávky, ktorú nakoniec potvrdí, čím sa objednávka presunie na expedovanie.

1. Zákazník zvolí zadanie objednávky.
2. Systém zobrazí možnosti vyhľadávania.
3. Zákazník nastaví možnosti vyhľadávania a spustí vyhľadávanie.
4. Systém zobrazí vyhľadané položky.
5. Zákazník vyberie z vyhľadaných položiek a potvrdí výber.
6. Systém vloží zvolený výrobok do košíka.
7. Zákazník môže pokračovať vo výbere výrobkov – prípad použitia pokračuje krokom 2.
8. Zákazník objedná výrobky v košíku.
9. Systém vyžiada údaje potrebné na realizáciu objednávky vrátane spôsobu platby.
10. Zákazník zadá požadované platobné údaje.
11. Kedykoľvek počas objednávania, zákazník môže vzdať tento proces.
12. Systém uloží objednávku do zoznamu objednávok na expedovanie.
13. Pre každý výrobok v objednávke, systém zistí stav zásob.
14. Systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu každého objednaného výrobku, ak by jeho stav po expedovaní poklesol pod stanovený limit.
15. Prípad použitia končí.

Predpoklady: zákazník je prihlásený

Dôsledky:

- Minimálne: výrobok, ktoré pôvodne boli súčasťou objednávky, sú v nej naďalej
- V prípade úspechu: výrobky, ktoré zákazník chce objednať sú súčasťou objednávky

Use Case Modularization

Predpoklady a dôsledky.

Účastníci (zákazník, obchodník, používateľ,...) - primárny actor

Prostredníc tvom účastníkov prípadov použitia si ujasňujeme, kto a ako bude používať systém, ale nemodelujeme tým priamo používateľské práva.

V prípadoch použitia ako účastník vystupuje aj systém alebo jeho časti - sekundárny actor.

Prípad použitia: Zadaj objednávku

Zákazník vyberá výrobok do košíka. Ten sa stáva súčasťou jeho objednávky, ktorú nakoniec potvrdí, čím sa objednávka presunie na expedovanie.

1. Zákazník zvolí zadanie objednávky.
2. Systém zobrazí možnosti vyhľadávania.
3. Zákazník nastaví možnosti vyhľadávania a spustí vyhľadávanie.
4. Systém zobrazí vyhľadané položky.
5. Zákazník vyberie z vyhľadaných položiek a potvrdí výber.
6. Systém vloží zvolený výrobok do košíka.
7. Zákazník môže pokračovať vo výbere výrobkov – prípad použitia pokračuje krokom 2.
8. Zákazník objedná výrobky v košíku.
9. Systém vyžiada údaje potrebné na realizáciu objednávky vrátane spôsobu platby.
10. Zákazník zadá požadované platobné údaje.
11. Kedykoľvek počas objednávania, zákazník môže vzdať tento proces.
12. Systém uloží objednávku do zoznamu objednávok na expedovanie.
13. Pre každý výrobok v objednávke, systém zistí stav zásob.
14. Systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu každého objednaného výrobku, ak by jeho stav po expedovaní poklesol pod stanovený limit.
15. Prípad použitia končí.

Predpoklady: zákazník je prihlásený

Dôsledky:

- Minimálne: výrobok, ktoré pôvodne boli súčasťou objednávky, sú v nej naďalej
- V prípade úspechu: výrobky, ktoré zákazník chce objednať sú súčasťou objednávky

Use Case Modularization

Prípad použitia je koncept vedomia koncového používateľa.

Korektná percepcia vývojárom - objednávky sa majú expedovať jednotlivo vs. expedovať viac objednávok naraz, aby mohol prioritne priradiť deficitárny výrobok k hodnotnejším objednávkam alebo význam nejším klientom.

Prípady použitia sú skutočne veľmi lacnou, avšak mimoriadne účinnou technikou modelovania softvéru.

Prípad použitia: Zadaj objednávku

Zákazník vyberá výrobok do košíka. Ten sa stáva súčasťou jeho objednávky, ktorú nakoniec potvrdí, čím sa objednávka presunie na expedovanie.

1. Zákazník zvolí zadanie objednávky.
2. Systém zobrazí možnosti vyhľadávania.
3. Zákazník nastaví možnosti vyhľadávania a spustí vyhľadávanie.
4. Systém zobrazí vyhľadané položky.
5. Zákazník vyberie z vyhľadaných položiek a potvrdí výber.
6. Systém vloží zvolený výrobok do košíka.
7. Zákazník môže pokračovať vo výbere výrobkov – prípad použitia pokračuje krokom 2.
8. Zákazník objedná výrobky v košíku.
9. Systém vyžiada údaje potrebné na realizáciu objednávky vrátane spôsobu platby.
10. Zákazník zadá požadované platobné údaje.
11. Kedykoľvek počas objednávania, zákazník môže vzdať tento proces.
12. Systém uloží objednávku do zoznamu objednávok na expedovanie.
13. Pre každý výrobok v objednávke, systém zistí stav zásob.
14. Systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu každého objednaného výrobku, ak by jeho stav po expedovaní poklesol pod stanovený limit.
15. Prípad použitia končí.

Predpoklady: zákazník je prihlásený

Dôsledky:

- Minimálne: výrobok, ktoré pôvodne boli súčasťou objednávky, sú v nej naďalej
- V prípade úspechu: výrobky, ktoré zákazník chce objednať sú súčasťou objednávky

Use Case Modularization

Prípad použitia nie je opis používateľského rozhrania. Opisuje funkcionalitu, nie používateľské rozhranie.

V texte UC neuvádzať prvky UI (okná, tlačidlá, menu, formuláre, označenia atď.). Popisovať činnosti na úrovni zámeru používateľa, nie na úrovni interakcie s GUI. Namiesto „klikne tlačidlo OK“ písať napr. „potvrdí akciu“.

Jeden prípad použitia môže byť realizovaný viacerými formulármi. Jeden formulár môže podporovať viacero prípadov použitia.

Prípady použitia reprezentujú mentálny model používateľa, nie dizajn obrazoviek.

Občasná zmienka o UI prvku nie je fatálna chyba, ale systematicky sa jej treba vyhýbať.

Prípad použitia: Zadaj objednávku

Zákazník vyberá výrobok do košíka. Ten sa stáva súčasťou jeho objednávky, ktorú nakoniec potvrdí, čím sa objednávka presunie na expedovanie.

1. Zákazník zvolí zadanie objednávky.
2. Systém zobrazí možnosti vyhľadávania.
3. Zákazník nastaví možnosti vyhľadávania a spustí vyhľadávanie.
4. Systém zobrazí vyhľadané položky.
5. Zákazník vyberie z vyhľadaných položiek a potvrdí výber.
6. Systém vloží zvolený výrobok do košíka.
7. Zákazník môže pokračovať vo výbere výrobkov – prípad použitia pokračuje krokom 2.
8. Zákazník objedná výrobky v košíku.
9. Systém vyžiada údaje potrebné na realizáciu objednávky vrátane spôsobu platby.
10. Zákazník zadá požadované platobné údaje.
11. Kedykoľvek počas objednávania, zákazník môže vzdať tento proces.
12. Systém uloží objednávku do zoznamu objednávok na expedovanie.
13. Pre každý výrobok v objednávke, systém zistí stav zásob.
14. Systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu každého objednaného výrobku, ak by jeho stav po expedovaní poklesol pod stanovený limit.
15. Prípad použitia končí.

Predpoklady: zákazník je prihlásený

Dôsledky:

- Minimálne: výrobok, ktoré pôvodne boli súčasťou objednávky, sú v nej naďalej
- V prípade úspechu: výrobky, ktoré zákazník chce objednať sú súčasťou objednávky

Use Case Modularization and Expressing It in UML

Nezahŕňať všetky možnosti do jedného sledu krokov – vedie to k neprehľadnosti a strate podstaty prípadu použitia.

Modularizovať prípady použitia – organizovať ich:na vnútornej úrovni (štruktúra scenárov, rozšírenia), na vonkajšej úrovni (vzťahy medzi prípadmi použitia).

Budovať celkový obraz systému cez prepojenie modulov, nie cez jeden rozsiahly opis.

Pri väčšom počte prípadov použitia hrozí strata prehľadnosti.

Použiť diagram prípadov použitia (UML Use Case Diagram) na zobrazenie štruktúry a vzťahov.

Cieľ: Zachovať jasnosť, prehľadnosť a systematickosť modelu.

Use Case Modularization P1

PROBLÉM 1

Prípad použitia Zadaj objednávku zahŕňa aj vyhľadanie výrobku.

Niekoľko krokov navyše, narúšajú plynulosť vnímania podstaty tohto prípadu použitia, ktorou je proces objednávania výrobku.

Tieto kroky by sme mohli vyčleniť do určitého podtoku, ktorý aktivujeme na príslušnom mieste základného toku.

Prípad použitia: Zadaj objednávku

Zákazník vyberá výrobok do košíka. Ten sa stáva súčasťou jeho objednávky, ktorú nakoniec potvrdí, čím sa objednávka presunie na expedovanie.

1. Zákazník zvolí zadanie objednávky.
2. Systém zobrazí možnosti vyhľadávania.
3. Zákazník nastaví možnosti vyhľadávania a spustí vyhľadávanie.
4. Systém zobrazí vyhľadané položky.
5. Zákazník vyberie z vyhľadaných položiek a potvrdí výber.
6. Systém vloží zvolený výrobok do košíka.
7. Zákazník môže pokračovať vo výbere výrobkov – prípad použitia pokračuje krokom 2.
8. Zákazník objedná výrobky v košíku.
9. Systém vyžiada údaje potrebné na realizáciu objednávky vrátane spôsobu platby.
10. Zákazník zadá požadované platobné údaje.
11. Kedykoľvek počas objednávania, zákazník môže vzdať tento proces.
12. Systém uloží objednávku do zoznamu objednávok na expedovanie.
13. Pre každý výrobok v objednávke, systém zistí stav zásob.
14. Systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu každého objednaného výrobku, ak by jeho stav po expedovaní poklesol pod stanovený limit.
15. Prípad použitia končí.

Predpoklady: zákazník je prihlásený

Dôsledky:

- Minimálne: výrobok, ktoré pôvodne boli súčasťou objednávky, sú v nej naďalej
- V prípade úspechu: výrobky, ktoré zákazník chce objednať sú súčasťou objednávky

Use Case Modularization

Podtok sa samostatne neaktivuje, ani neukončuje prípad použitia, a preto neobsahuje aktivačný a ukončovací krok ako základný tok. Pomenovanie základného toku možno vynechať.

Prípad použitia: Zadaj objednávku

Základný tok: Zadaj objednávku

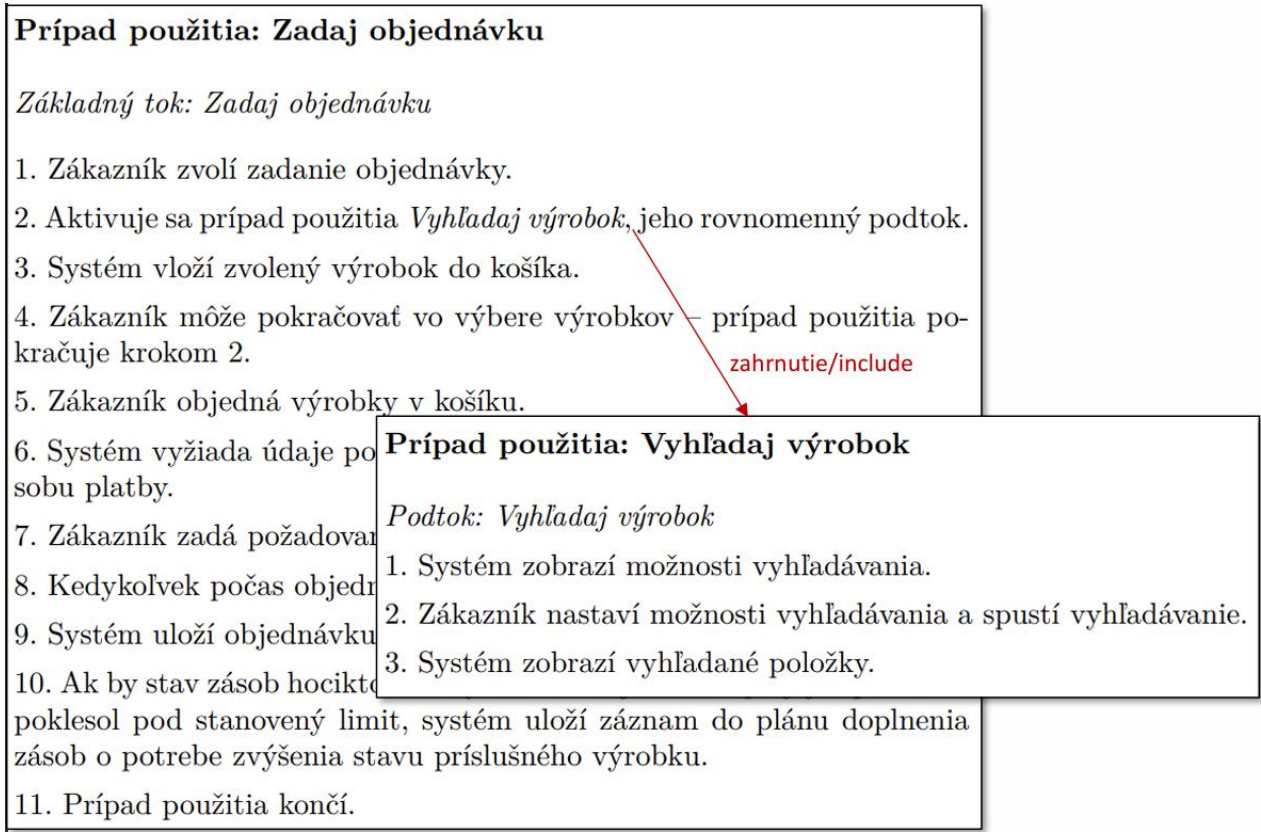
1. Zákazník zvolí zadanie objednávky.
2. Aktivuje sa podtok *Vyhľadaj výrobok*.
3. Systém vloží zvolený výrobok do košíka.
4. Zákazník môže pokračovať vo výbere výrobkov – prípad použitia pokračuje krokom 2.
5. Zákazník objedná výrobky v košíku.
6. Systém vyžiada údaje potrebné na realizáciu objednávky vrátane spôsobu platby.
7. Zákazník zadá požadované platobné údaje.
8. Kedykoľvek počas objednávania, zákazník môže vzdať tento proces.
9. Systém uloží objednávku do zoznamu objednávok na vybavenie.
10. Ak by stav zásob hociktorého výrobku v objednávke po jej expedovaní poklesol pod stanovený limit, systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu príslušného výrobku.
11. Prípad použitia končí.

Podtok: Vyhľadaj výrobok

1. Systém zobrazí možnosti vyhľadávania.
2. Zákazník nastaví možnosti vyhľadávania a spustí vyhľadávanie.
3. Systém zobrazí vyhľadané položky.

Use Case Modularization - include

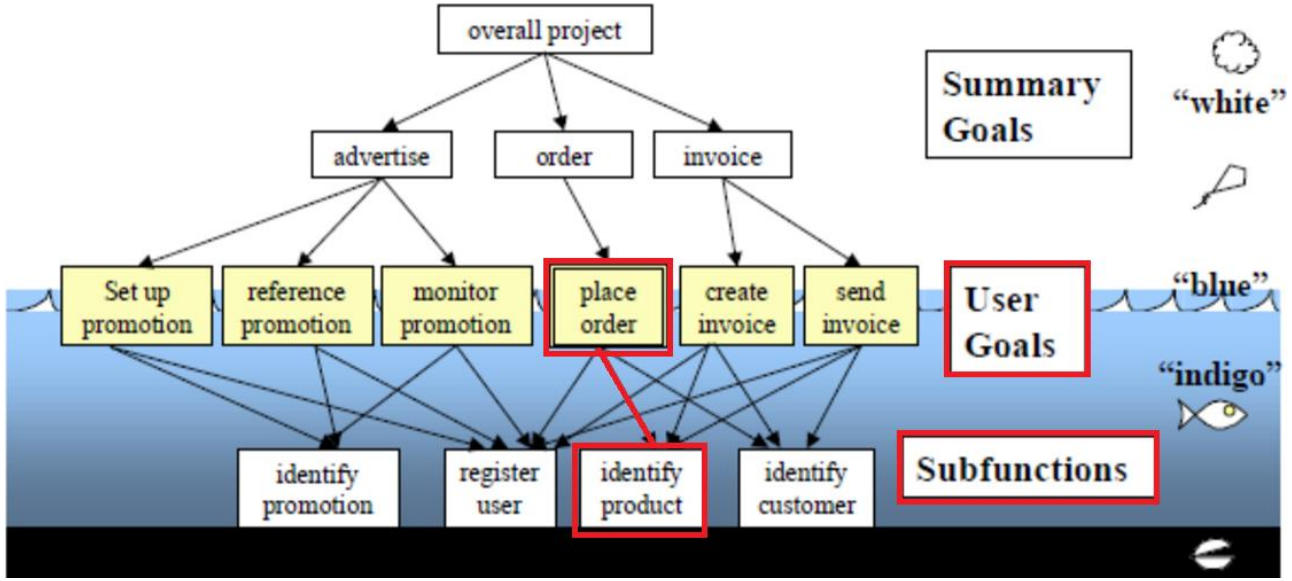
Podtok je možné vyčleniť do separátneho prípadu použitia a odvolávať sa naň. V tom prípade ide o zahrnutie.



Use Case Modularization - include

Zahrnutie (Cockburn) Sub use case.

Vyčlenený podtok môže byť užitočný aj pre iné prípady použitia.



Use Case 7: Vyhľadaj výrobok

Level: User-goal
Primary Actor: Zákazník

Main Success Scenario:

1. Zákazník zvolí zadanie objednávky.
2. Zákazník vyhľadá a zvolí výrobok (UC 35 Vyhľadaj výrobok).
3. Systém vloží zvolený výrobok do košíka.
4. Zákazník môže pokračovať vo výbere výr… prípad použitia pokračuje krokom 2.
5. Zákazník objedná výrobky v košíku.
6. Systém vyžiada údaje potrebné na realiz… objednávky vrátane spôsobu platby.
7. Zákazník zadá požadované platobné úda…
8. Kedykoľvek počas objednávania, zákazní… vzdať tento proces.
9. Systém uloží objednávku do zoznamu ob… na vybavenie.
10. Ak by stav zásob hociktorého výrobku v … po jej expedovaní poklesol pod stanover… systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu príslušného výrobku.
11. Prípad použitia končí.

zahrnutie/include

Sub use case

Use Case 35: Vyhľadaj výrobok

Level: Subfunction
Primary Actor: Zákazník

Main Success Scenario:

1. Zákazník pristúpi k vyhľadávaniu.
2. Systém zobrazí možnosti vyhľadávania.
3. Zákazník nastaví možnosti vyhľadávania a spustí vyhľadávanie.
4. Systém zobrazí vyhľadané položky.

Use Case Modularization - include

```java
public class Objednavanie {

    public void objednaj(Vyrobok vyrobok, int mnozstvo) {

        // Vyhľadanie výrobku
        new VyhladavanieVyrobkov().vyhladaj(vyrobok);

        // Kontrola stavu zásob
        if (zistiStavZasob(vyrobok) >= mnozstvo) {

            // Dostatočné množstvo na sklade
            // ... realizácia objednávky

        } else {

            // Nedostatočné množstvo na sklade
            // ... alternatívne spracovanie

        }
    }

    private int zistiStavZasob(Vyrobok vyrobok) {
        // ... implementácia zisťovania stavu zásob
        return 0;
    }
}
```

Prípad použitia: Zadaj objednávku

Základný tok: Zadaj objednávku

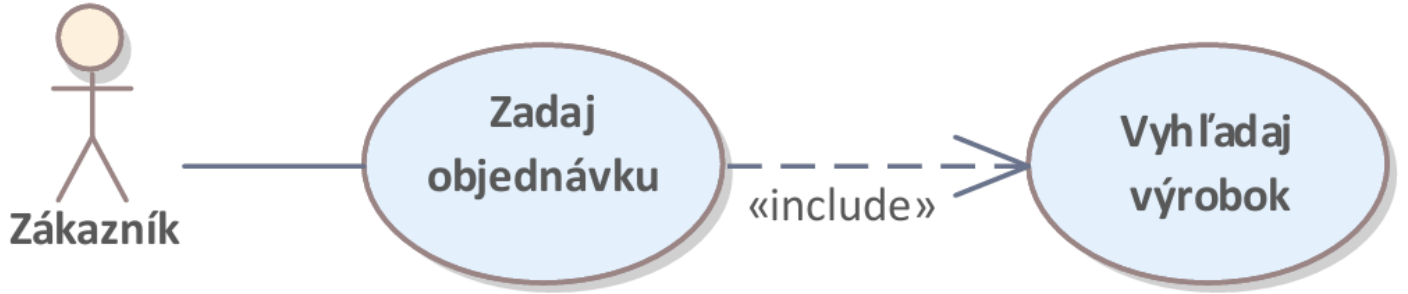
1. Zákazník zvolí zadanie objednávky.
2. Aktivuje sa prípad použitia *Vyhľadaj výrobok*, jeho rovnomenný podtok.
3. Systém vloží zvolený výrobok do košíka.
4. Zákazník môže pokračovať vo výbere výrobkov – prípad použitia pokračuje krokom 2.
5. Zákazník objedná výrobky v košíku.
6. Systém vyžiada údaje po
sobu platby.
7. Zákazník zadá požadovar
8. Kedykoľvek počas objedr
9. Systém uloží objednávku
10. Ak by stav zásob hocikt
poklesol pod stanovený limit, systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu príslušného výrobku.
11. Prípad použitia končí.

zahrnutie/include

Prípad použitia: Vyhľadaj výrobok

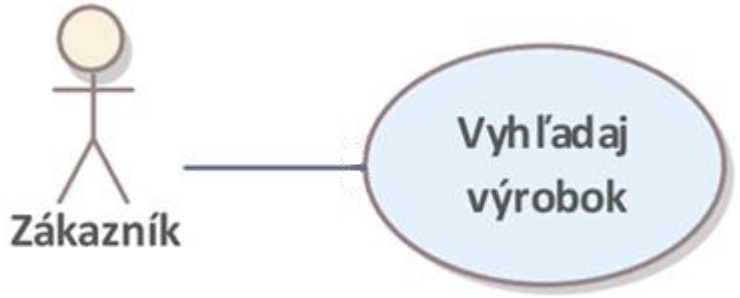
Podtok: Vyhľadaj výrobok

1. Systém zobrazí možnosti vyhľadávania.
2. Zákazník nastaví možnosti vyhľadávania a spustí vyhľadávanie.
3. Systém zobrazí vyhľadané položky.



Use Case Modularization - include

Ak by prípad použitia Vyhľadaj výrobok malo význam aktivovať aj samostatne, po trebné je pridať základný tok.



Prípad použitia: Vyhľadaj výrobok

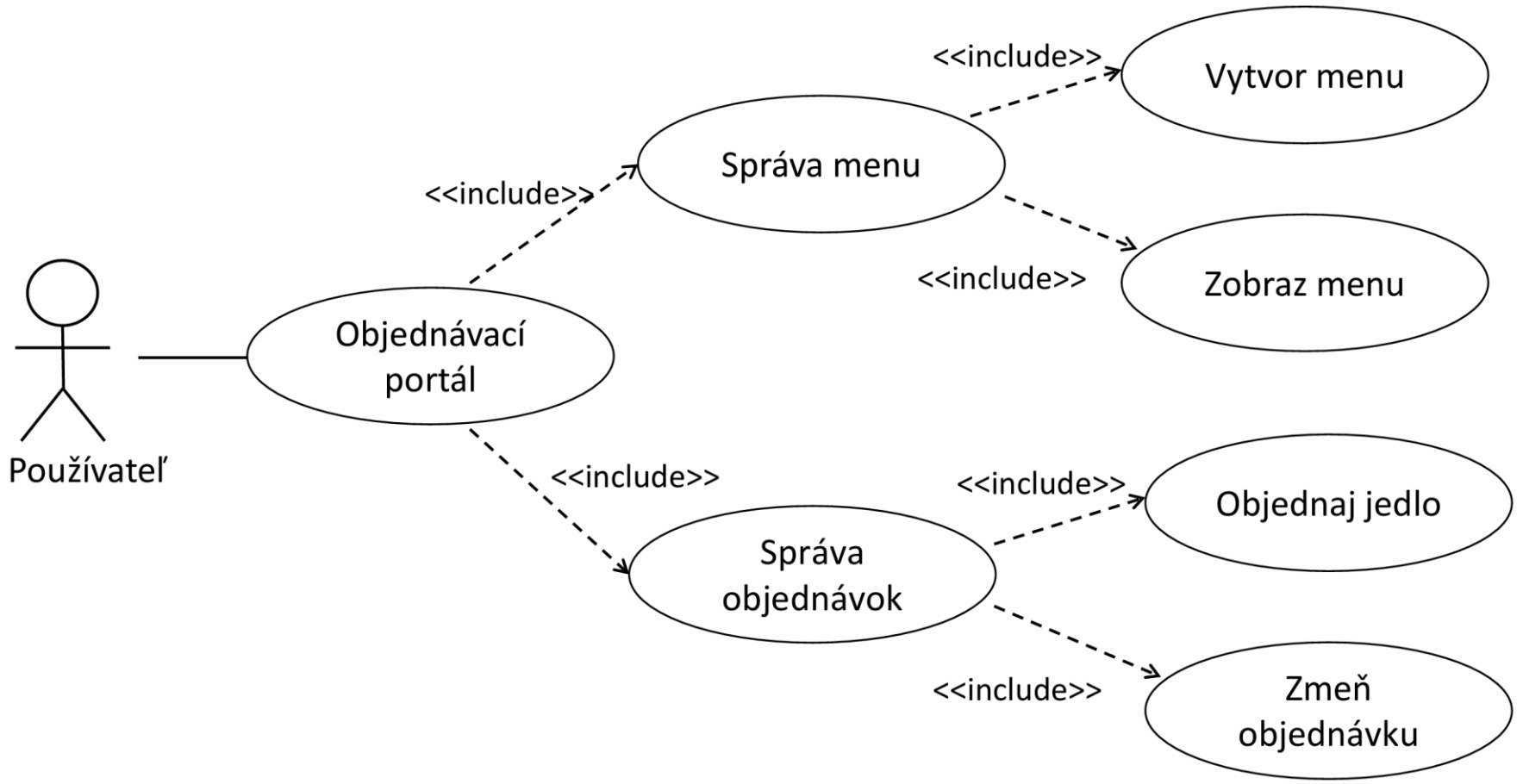
Základný tok: Vyhľadaj výrobok

1. Zákazník zvolí zadanie objednávky.
2. Aktivuje sa podtok *Vyhľadaj výrobok.*
3. Prípad použitia končí.

Podtok: Vyhľadaj výrobok

1. Systém zobrazí možnosti vyhľadávania.
2. Zákazník nastaví možnosti vyhľadávania a spustí vyhľadávanie.
3. Systém zobrazí vyhľadané položky.

POZOR, include, ale nie dekompozíciu systému!



Use Case Modularization P2

PROBLÉM 2

Krok č. 10 = kontrola stavu zásob výrobkov v objednávke.

Taktiež narúša plynulosť vnímania podstaty tohto prípadu použitia.

Iná povaha súvislosti toku - Alternatíva (Alternatívny tok)

Umiestnenie mimo základného toku vrátane podmienky aktivácie

Prípad použitia: Zadaj objednávku

Základný tok: Zadaj objednávku

1. Zákazník zvolí zadanie objednávky.
2. Aktivuje sa prípad použitia *Vyhľadaj výrobok*, jeho rovnomenný podtok.
3. Systém vloží zvolený výrobok do košíka.
4. Zákazník môže pokračovať vo výbere výrobkov – prípad použitia pokračuje krokom 2.
5. Zákazník objedná výrobky v košíku.
6. Systém vyžiada údaje potrebné na realizáciu objednávky vrátane spôsobu platby.
7. Zákazník zadá požadované platobné údaje.
8. Kedykoľvek počas objednávania, zákazník môže vzdať tento proces.
9. Systém uloží objednávku do zoznamu objednávok na vybavenie.
10. Ak by stav zásob hociktorého výrobku v objednávke po jej expedovaní poklesol pod stanovený limit, systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu príslušného výrobku.
11. Prípad použitia končí.

Use Case Modularization P2

PROBLÉM 2

Krok č. 10 = kontrola stavu zásob výrobkov v objednávke.

Taktiež narúša plynulosť vnímania podstaty tohto prípadu použitia.

Iná povaha súvislosti toku - Alternatíva (Alternatívny tok - Modifikuj plán doplnenia zásob)

Umiestnenie mimo základného toku vrátane podmienky aktivácie

Na alternatívny tok sa v hlavnom toku neodvolávame

Prípad použitia: Zadaj objednávku

Základný tok: Zadaj objednávku

1. Zákazník zvolí zadanie objednávky.

2. Aktivuje sa prípad použitia *Vyhľadaj výrobok*, podtok *Vyhľadaj výrobok*.

3. Systém vloží zvolený výrobok do košíka.

4. Zákazník môže pokračovať vo výbere výrobkov – prípad použitia pokračuje krokom 2.

5. Zákazník objedná výrobky v košíku.

6. Systém vyžiada údaje potrebné na realizáciu objednávky vrátane spôsobu platby.

7. Zákazník zadá požadované platobné údaje.

8. Kedykoľvek počas objednávania, zákazník môže vzdať tento proces.

9. Systém uloží objednávku do zoznamu objednávok na vybavenie.

10. Prípad použitia končí.

Alternatívny tok: Modifikuj plán doplnenia zásob

V kroku 9 základného toku, ak by stav zásob hociktorého výrobku v objednávke po jej expedovaní poklesol pod stanovený limit:

1. Systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu každého výrobku v objednávke, ktorého stav zásob by po jej expedovaní poklesol pod stanovený limit.

Use Case Modularization - extend

Alternatívny tok - Modifikuj plán doplnenia zásob vyjadrený ako UC.

V tomto rozširujúcom prípade použitia je nutné uviesť, ktorý prípad použitia rozširujeme a v ktorom konkrétnom bode rozšírenia.

V rozšírenom prípade použitia je teda nutné uviesť body rozšírenia, uvádza sa ich názov (ľubovoľný) a číslo kroku, kde dochádza k rozšíreniu daného toku.

Prípad použitia: Zadaj objednávku Rozšírený prípad použitia

Základný tok: Zadaj objednávku

1. Zákazník zvolí zadanie objednávky.
2. Aktivuje sa prípad použitia *Vyhľadaj výrobok*, jeho rovnomenný podtok.
3. Systém vloží zvolený výrobok do košíka.
4. Zákazník môže pokračovať vo výbere výrobkov – prípad použitia pokračuje krokom 2.
5. Zákazník objedná výrobky v košíku.
6. Systém vyžiada údaje potrebné na realizáciu objednávky vrátane spôsobu platby.
7. Zákazník zadá požadované platobné údaje.
8. Kedykoľvek počas objednávania, zákazník môže vzdať tento proces.
9. Systém uloží objednávku do zoznamu objednávok na vybavenie.
10. Prípad použitia končí.

Body rozšírenia:

- Vloženie výrobku do košíka: krok 3
- Uloženie objednávky: krok 9

Prípad použitia: Modifikuj plán doplnenia zásob

Alternatívny tok: Modifikuj plán doplnenia zásob

rozšírenie/extend

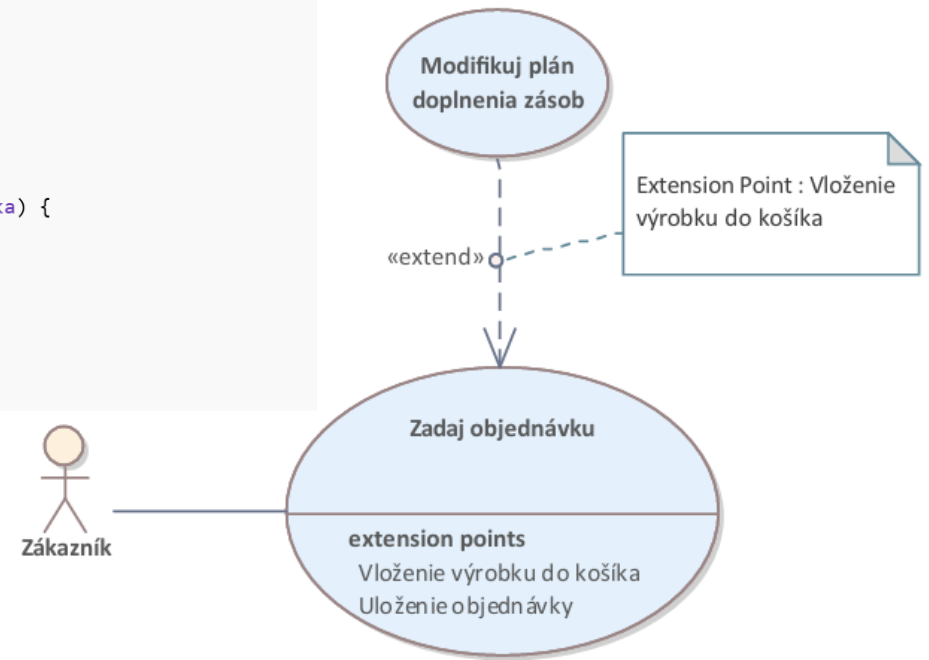
V bode rozšírenia *Uloženie objednávky* základného toku prípadu použitia *Zadaj objednávku*, ak by stav zásob hociktorého výrobku v objednávke po jej expedovaní poklesol pod stanovený limit:

1. Systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu každého výrobku v objednávke, ktorého stav zásob by po jej expedovaní poklesol pod stanovený limit.

Rozširujúci prípad použitia

Use Case Modularization - extend

```java
public class Objednavanie {

    // ...

    public void objednaj(Objednavka objednavka) {

        // Uloženie objednávky
        ulozObjednavku(objednavka);

        // Kontrola zásob pre každú objednanú položku
        for (Vyrobok vyrobok : objednavka.objednanePolozky) {

            if (planZasob.zistiStavZasob(vyrobok)
                    < planZasob.minimalneZasoby(vyrobok) + vyrobok.mnozstvo) {

                planZasob.dopln(vyrobok);
            }
        }

        // ...
    }

    private void ulozObjednavku(Objednavka objednavka) {
        // ... implementácia uloženia objednávky
    }

    private PlanZasob planZasob;

}
```



Prípad použitia: Zadaj objednávku Rozšírený prípad použitia

Základný tok: Zadaj objednávku

1. Zákazník zvolí zadanie objednávky.
2. Aktivuje sa prípad použitia *Vyhľadaj výrobok*, jeho rovnomenný podtok.
3. Systém vloží zvolený výrobok do košíka.
4. Zákazník môže pokračovať vo výbere výrobkov – prípad použitia pokračuje krokom 2.
5. Zákazník objedná výrobky v košíku.
6. Systém vyžiada údaje potrebné na realizáciu objednávky vrátane spôsobu platby.
7. Zákazník zadá požadované platobné údaje.
8. Kedykoľvek počas objednávania, zákazník môže vzdať tento proces.
9. Systém uloží objednávku do zoznamu objednávok na vybavenie.
10. Prípad použitia končí.

Body rozšírenia:

- Vloženie výrobku do košíka: krok 3
- Uloženie objednávky: krok 9

Prípad použitia: Modifikuj plán doplnenia zásob

Alternatívny tok: Modifikuj plán doplnenia zásob

rozšírenie/extend

V bode rozšírenia *Uloženie objednávky* základného toku prípadu použitia *Zadaj objednávku*, ak by stav zásob hociktorého výrobku v objednávke po jej expedovaní poklesol pod stanovený limit:

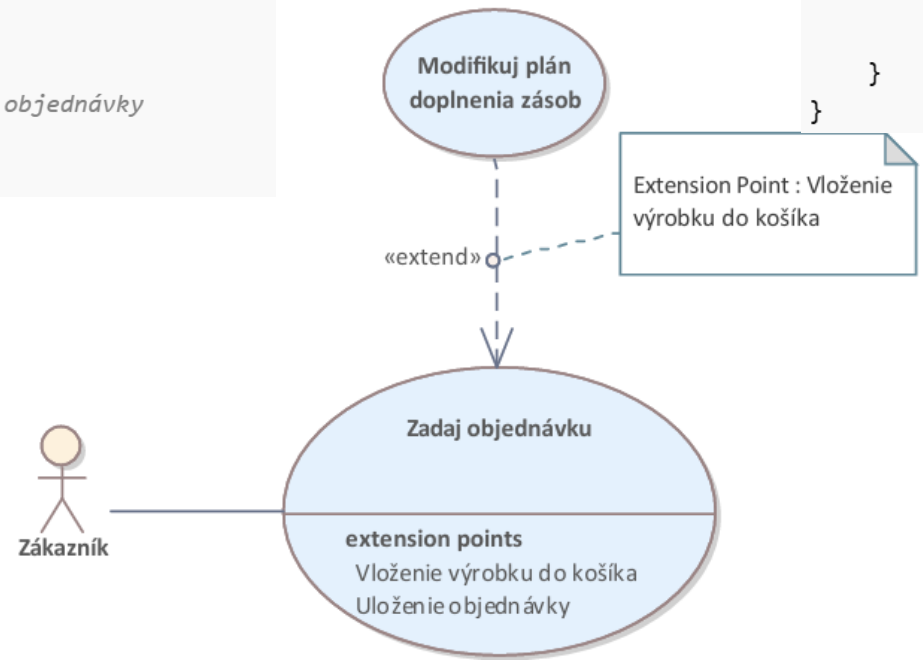
1. Systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu každého výrobku v objednávke, ktorého stav zásob by po jej expedovaní poklesol pod stanovený limit.

Rozširujúci prípad použitia

Use Case Modularization – extend ()

```java
public class Objednavanie {

    // ...

    public void objednaj(Vyrobok vyrobok, int mnozstvo) {

        // ... spracovanie objednávky

        ulozObjednavku();
    }

    // ...

    private void ulozObjednavku() {
        // ... implementácia uloženia objednávky
    }
}
```

```java
public aspect PlanDoplneniaZasob {

    after(Objednavanie objednavanie, Objednavka objednavka) :
        this(objednavanie)
        && call(* Objednavanie.ulozObjednavku(Objednavka))
        && args(objednavka) {

        for (Vyrobok vyrobok : objednavka.objednanePolozky) {

            if (objednavanie.planZasob.zistiStavZasob(vyrobok)
                    < vyrobok.minimalneZasoby() + vyrobok.mnozstvo) {

                objednavanie.planZasob.dopln(vyrobok);
            }
        }
    }
}
```

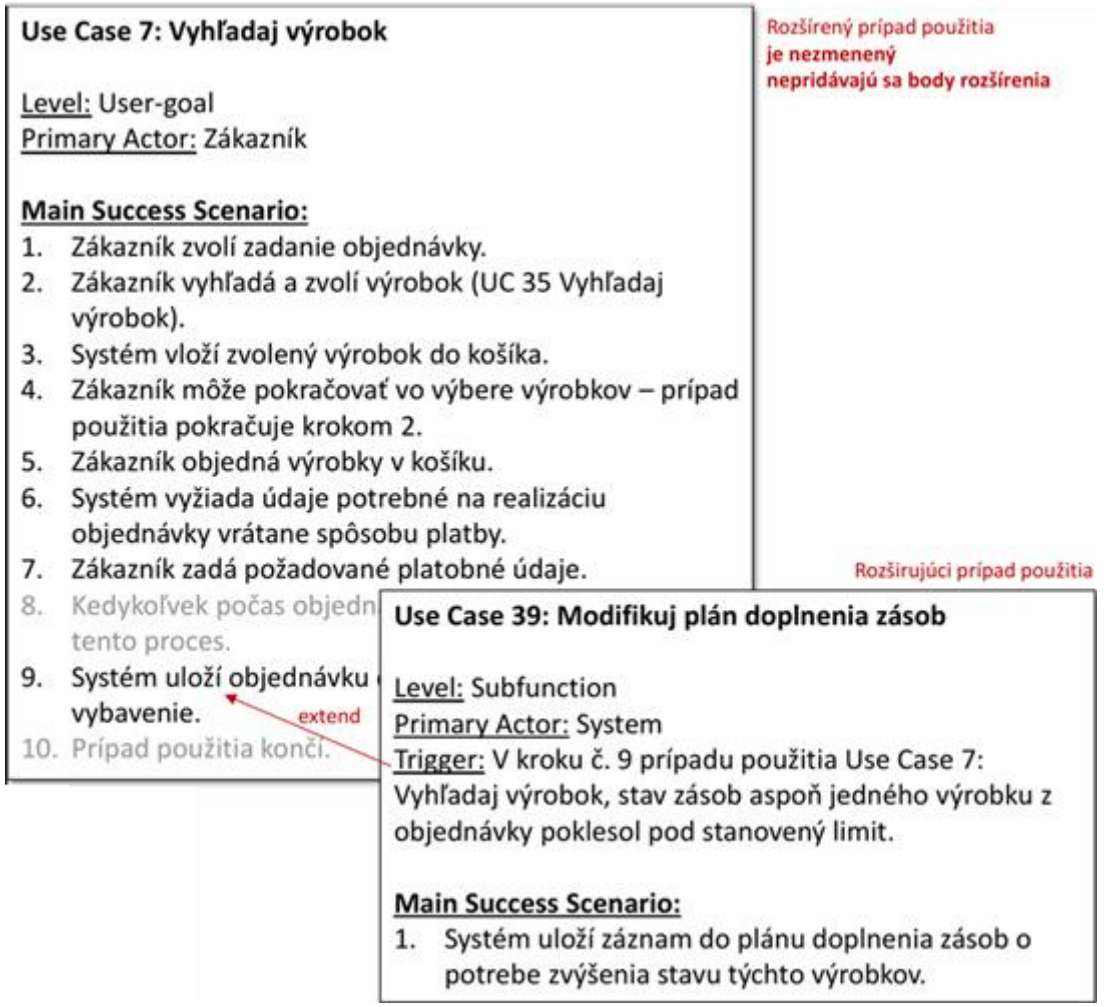


Use Case Modularization - extend

Rozšírenie (Cockburn) realizácia Extension use case vlastnosťou Trigger.

Nemodelujeme: Podtok, Alternatívny tok ako súčasť prípadu použitia

Modelujeme: Zahrnutie (a teda aj odkazovanie na sub use case) ako include, Rozšírenie (a teda aj extension use cases) ako extend. U Cockburna neuvádzame body rozšírenia (Trigger)



Use Case Modularization and Expressing It in UML

Nemodelujeme: Podtok, Alternatívny tok ako súčasť prípadu použitia.

Modelujeme: Zahrnutie (a teda aj odkazovanie na sub use case) ako include, Rozšírenie (a teda aj extension use cases) ako extend. U Cockburna neuvádzame body rozšírenia (Trigger).

It makes it comprehensible to all the stakoholders, it helps in source code orientation.

Use case diagram is useful as an overview, but is not a major part of the use case model

Use case modularization help to cope with complexity

Use Case Modularization - inheritance

Prípad použitia môže špecializovať už definovaný prípad použitia, t. j. môže od neho dediť. V opise sa uvedie, ktoré kroky prekonáva.

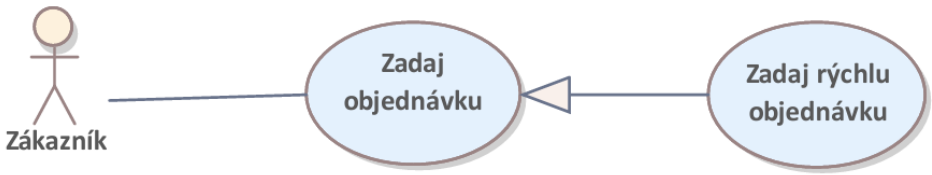
V diagramoch prípadov použitia sa tento vzťah značí ako generalizácia/špecializácia (angl. generalization/specialization), čo je UML termín pre vzťah vo vývoji softvéru známy ako dedenie (angl. inheritance).

Zadaj rýchlu objednávku

Prípad použitia špecializuje prípad použitia *Zadaj objednávku*. Kroky sú prekonané nasledovne:

1. Zákazník zvolí urýchlené objednanie výrobku.
2. Zákazník priamo zadá kód výrobku.

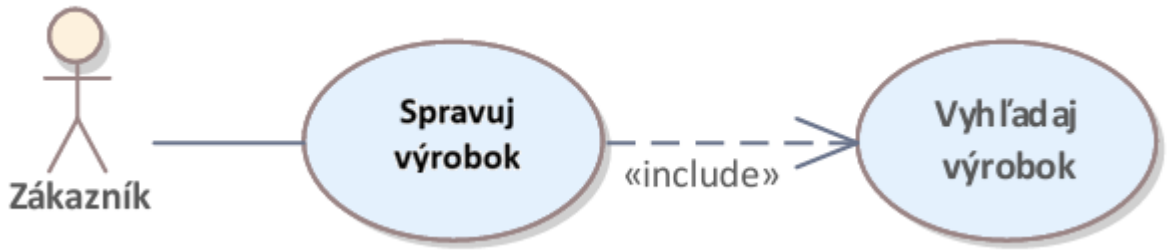
Ostatné kroky zostávajú v platnosti.



Use Case Modularization - CRUD

CRUD prípad použitia vystupuje ako jeden prípad použitia.

CRUD prípad použitia nemusí zahŕňať všetky štyri operácie.



Prípad použitia: Spravuj výrobok

Základný tok: Vytvor výrobok

1. Obchodník zvolí vytvorenie výrobku.
2. Systém vyžiada údaje o výrobku.
3. Obchodník zadá názov, typ a obrázok výrobku.
4. Systém prispôsobí veľkosť obrázku výrobku štandardnej veľkosti.
5. Obchodník zaradí výrobok do kategórie.
6. Ak obchodník potvrdí zadané údaje, systém ich uloží.
7. Prípad použitia končí.

Základný tok: Zobraz výrobok

1. Obchodník zvolí zobrazenie jestvujúceho výrobku.
2. Aktivuje sa prípad použitia *Vyhľadaj výrobok*, jeho rovnomenný tok.
3. Systém zobrazí vybraný výrobok.
4. Obchodník prezrie výrobok.
5. Prípad použitia končí.

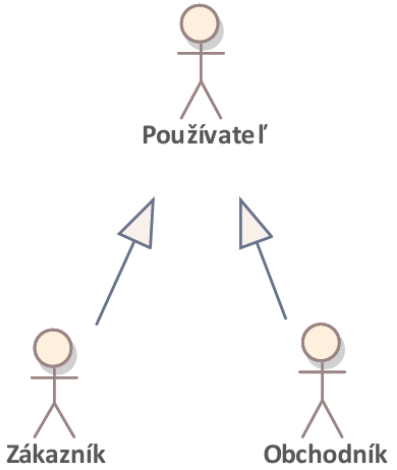
Základný tok: Uprav výrobok

1. Obchodník zvolí úpravu jestvujúceho výrobku.
2. Aktivuje sa prípad použitia Vyhľadaj výrobok.
3. Systém otvorí vybraný výrobok a umožní jeho úpravu.
4. Obchodník upraví údaje o výrobku.
5. Ak obchodník potvrdí zmeny, systém ich uloží.
6. Prípad použitia končí.

Základný tok: Vyraď výrobok

1. Obchodník zvolí vyradenie výrobku.
2. Aktivuje sa prípad použitia *Vyhľadaj výrobok*, jeho rovnomenný tok.
3. Ak obchodník potvrdí vyradenie výrobku, systém na danom výrobku nastaví príznak vyradenia.
4. Prípad použitia končí.

Účastníci, Aktéri, Roly



Zákazník a obchodník sú druhmi používateľa.

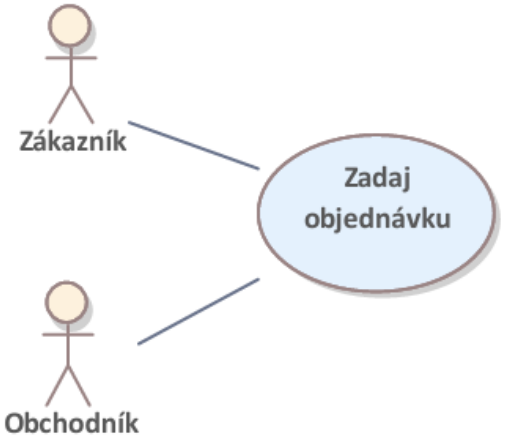
Typy účastníkov môžu pribúdať.

Tak ako pri dedení medzi prípadmi použitia, v diagrame použijeme vzťah generalizácie/špecializácie.

Obchodník = Sekundárny účastník, Zákazník zostáva primárnym účastníkom, ak/lebo prípad použitia aktivuje.

Prostredníctvom účastníkov prípadov použitia si ujasňujeme, kto a ako bude používať systém, ale nemodelujeme tým priamo používateľské práva.

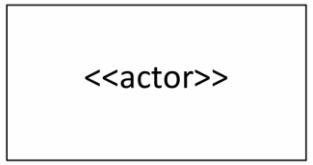
V prípadoch použitia ako účastník vystupuje aj systém alebo jeho časti.

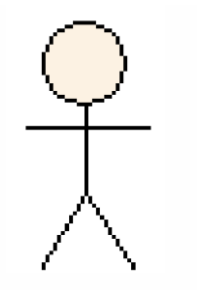


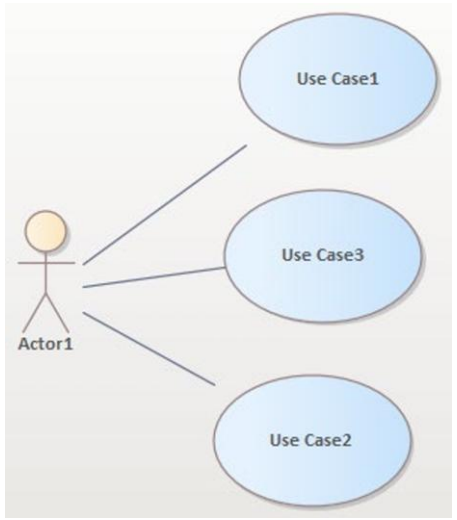
Účastníci, Aktéri, Roly

Aktér, predstavuje **rolu, ktorú niekto alebo niečo zohráva pri interakcii so systémom**.

Kto všetko môže byť Aktérom? (Účastník Používateľ, User; Čas; Externý systém; Interný subsystém.







Otázky pomáhajúce nájsť aktérov:

- Kto alebo čo používa systém?
- Akú rolu v tejto interakcii hrá?
- Kto spúšťa a vypína systém?
- Kto sa stará o údržbu systému?
- Kto systému zadáva informácie a kto ich používa?
- Aké ďalšie systémy spolupracujú so systémom?
- Robí sa niečo v určitú dobu?

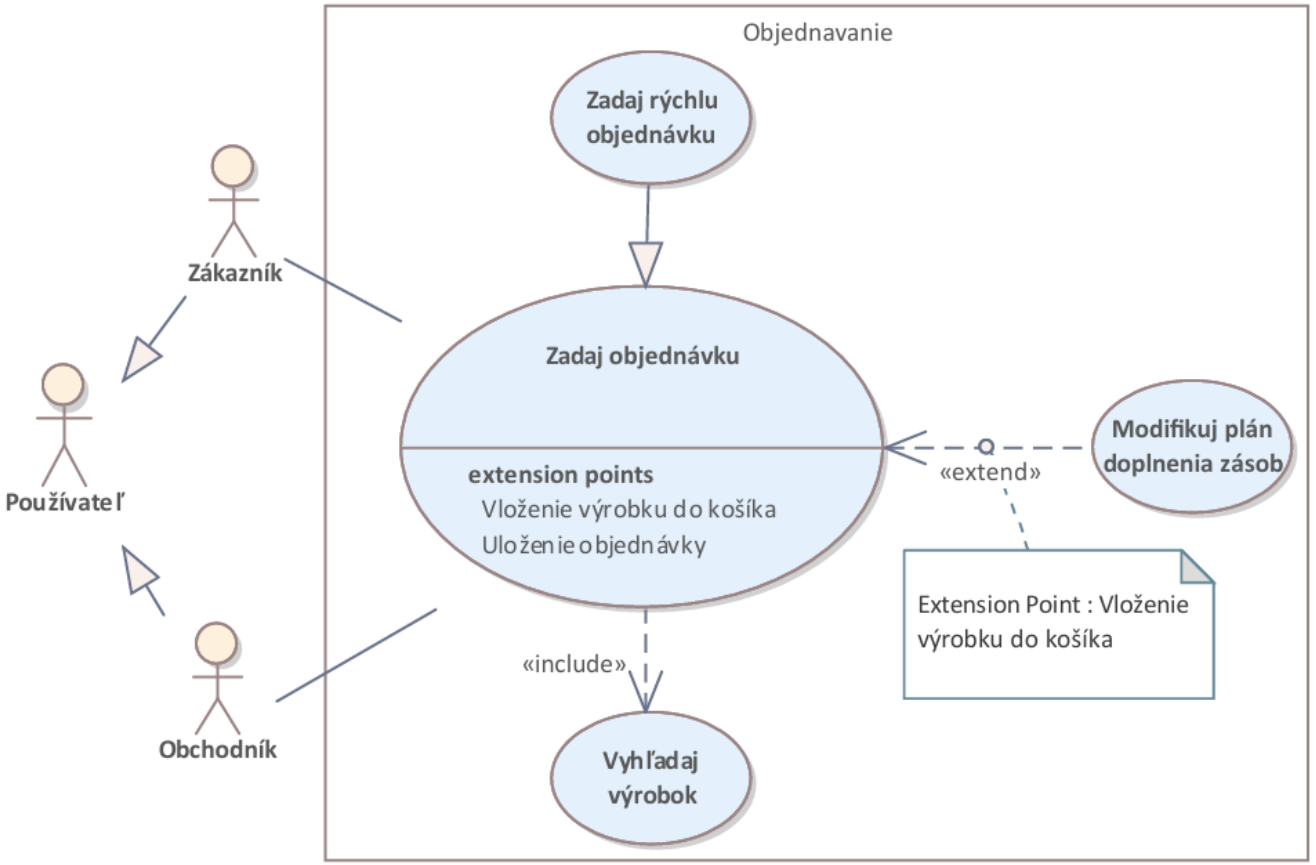
Modularizácia na úrovni prípadov použitia

Prípadov použitia môže byť viac

Je vhodné ich zobrazovať vo viacerých diagramoch organizovaných podľa predmetu, na ktorý sa zameriavajú.

Pre e-shop: objednávanie, expedovanie, správa výrobkov, správa zákazníkov atď.

Predmet sa niekedy vyznačuje orámovaním. Diagram prípadov použitia, ktoré sa vzťahujú na objednávanie.



Artefakty vo vývoji softvéru

Ciele

Čo a prečo:

Zákazník definuje svoje potreby sformulovaním biznis cieľov. Bez cieľov projekt nemá zmysel. Všetky artefakty a činnosti v softvérovom projekte by mali súvisieť s ich plnením.

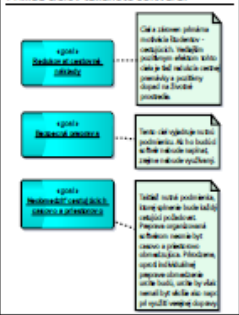
Využívané techniky:

Diagram biznis procesov (non-UML)
Texty

Príklad prvotného zadania:

Pravidelné cestovanie je nákladné. Jednou z možností ako náklady redukovať je spoločné automobilové prepravu (car-pooling). Tá však efektívne funguje iba vtedy, keď ju organizuje veľká komunita. V rámci komunity musí zároveň existovať základná miera dôvery. Univerzita toto identifikovala ako príležitosť a chce pre svojich študentov zaviesť systém zdieľania kapacít áut pri cestovaní. Ten má umožniť študentom sa efektívne organizovať v spoločnom cestovaní a zdieľaní nákladov.

Príklad cieľov takéhoto softvéru:



Zainteresované osoby (aktéri)

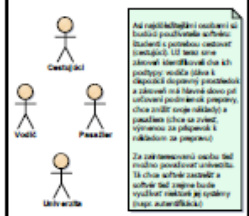
Čo a prečo:

Od prvej chvíle spolu s cieľmi charakterizujeme osoby, ktoré majú na softvéri nejaký záujem (aktérov). Neskôr v analýze sa nezaobídeme bez komunikácie s nimi. Mnohé z nich budú tiež používateľmi vytváraného softvéru.

Využívané techniky:

Diagram biznis procesov (non-UML)
Texty

Príklad identifikovaných aktérov:



Biznis procesy

Čo a prečo:

Postupy, ktoré majú napĺňať stanovené biznis ciele sa nazývajú biznis procesy. Vystupujú v nich tiež aktéri a rôzne ďalšie artefakty. Definovať biznis procesy potrebujeme, aby sme následne mohli identifikovať rolu budúceho softvéru v rámci nich.

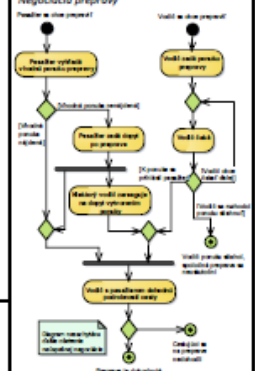
Využívané techniky:

Diagram činností (activity diagram)
Diagram biznis procesov (vzťahy medzi BP)
Texty (dôležité pre opisy diagramov)

Príklady biznis procesov:

Príklad biznis procesu (diagram aktivít):

Negociácia prepravy



Pasažier vyhľadá v existujúcich ponukách prepravy (zadaných predtým vodičmi) nejakú, ktorá mu vyhovuje. Môže tiež zadať vlastný dopyt po preprave (želanú cestu), na ktorú môžu vodiči tiež reagovať. Po nájdení potenciálnych spolucestujúcich je ďalšie vyjednanie podrobností ponechané na nich.

Slovník pojmov

Čo a prečo:

Ako vývojári potrebujeme porozumieť doméne, pre ktorú softvér vyvíjame. Súbežne s biznis modelovaním preto vytvárame slovník pojmov. Definované pojmy sa neskôr často ukážu ako funkcie a dátové entity softvéru. Nevyhýbame sa ani „triviálnym" pojmom. Aj tie môžu rôzni ľudia chápať odlišne, čo môže neskôr spôsobiť problémy.

Využívané techniky:

Zoznam textových definícií

Príklad:

- *Negociácia – Vyjednávanie o podmienkach prepravy.*
- *Trasa – opakovane používaná geografická trajektória z miesta A do miesta B*

Funkcionálne požiadavky

Čo a prečo:

Zmapuje sa budúca funkcionalita softvéru. Snaha je reprezentovať ju najmenšími možnými jednotkami, ktoré ešte majú biznis zmysel. Týmito jednotkami sú prípady použitia (use cases) alebo ich obdoba, napr. odfahčené používateľské príbehy (user stories).

Využívané techniky:

Diagram prípadov použitia (use case diagram)
Textové používateľské príbehy

Príklad diagramu prípadov použitia:

Prípady použitia procesu negociácie prepravy

Príklady dvoch používateľských príbehov:

Forma kto-čo-prečo:
Kto: *Vodič.*
Čo chce spraviť: *Vytvoriť ponuku na prepravu, zadaním časov a miest odchodu a príchodu.*
Prečo: *Aby mohli pasažieri ponuku vyhľadať a prípadne si ju aj vybrať (a znížiť mi náklady).*

Alternatívne aj forma ako-chcem-aby:
Ako pasažier, chcem v existujúcich ponukách prepravy vyhľadať pre mňa vyhovujúcu a prihlásiť sa na ňu, aby som sa mohol s vodičom dohodnúť na podmienkach.

Nefunkcionálne požiadavky

Čo a prečo:

Definujú sa požadované vlastnosti softvéru, ktoré nemajú charakter funkcií softvéru. Mali by byť merateľné.

Využívané techniky:

Textový opis (kritériá)

Príklad:

- *Softvér je prístupný len pre študentov a zamestnancov univerzity*
- *Odozva pri všetkých akciách používateľov softvéru je maximálne 0.5 sekúnd*
- *Aplikácia má responzívny dizajn a funguje aj pre smartfóny a tablety.*
- *Aplikácia musí podporovať rozlíšenia od 720 x 1280 pixelov na zariadeniach s uhlopriečkou od 4.8 palcov vyššie.*

Scenáre

Čo a prečo:

Prípady použitia sa detailne opíšu. Najdôležitejšie sú scenáre, určujúce čo presne sa má v prípadoch použitia diať.

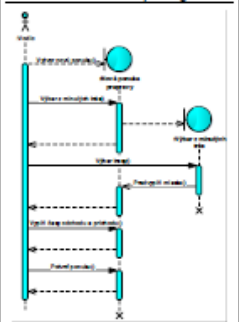
Využívané techniky:

Štruktúrovaný text, číslované zoznamy.
Sekvenčný diagram, diagram činností

Príklad opisu prípadu použitia:

Vytvorenie ponuky na prepravu:
Aktér: *Vodič.*
Krátky opis: *Vodič vytvorí ponuku na prepravu (...).*
Vstupné podmienky: *vodič je prihlásený*
Výstupné podmienky: *ponuka je zverejnená*
Hlavný scenár:
1. V hlavnom menu vodič vyberie možnosť „vytvor"
2. Otvorí sa obrazovka „vytvorte novú cestu"
3. Vodič zadá miesto začiatku trasy (auto-complete)
4. Vodič zadá cieľové miesto trasy (auto-complete)
5. Vodič zadá čas odchodu
6. Aplikácia zobrazí odhadovaný čas príchodu
7. Vodič stlačí „vytvoriť"
8. Aplikácia zobrazí potvrdenie zverejnenia ponuky
Alternatívny scenár 1:
Po kroku 2 sa vodič rozhodne použiť existujúcu trasu
1.1 Vodič klikne na „výber z mojich trás"
1.2 Aplikácia zobrazí zoznam mojich trás
1.3 Vodič zo zoznamu vyberie niektorú z trás
1.4 Prípad použitia pokračuje bodom 5 hl. toku

Príklad: scenár sekvenčným diagramom



Akceptačné testy

Čo a prečo:

Vytvoria sa testovacie scenáre. Ak nimi softvér prejde, je považovaný za dokončený: sú de-facto zmluvou medzi vývojárom a zákazníkom.

Využívané techniky:

Štruktúrovaný text, číslované zoznamy

Príklad akceptačného testu:

Vytvorenie ponuky na prepravu (Test 01):
Predpoklady testu: *Prihlásený používateľ Jožo*
Scenár testu:
1. V hlavnom menu vyberte „vytvor"
2. Očakávaná odozva: otvorí sa obrazovka „nová ponuka prepravy"
3. Ako počiatočné trasy zadajte „Žilina"
3. Očakávaná odozva: auto-complete po prvom znaku
4. Ako cieľ trasy zadajte „Bratislava"
3. Očakávaná odozva: auto-complete po prvom znaku
5. Ako čas odchodu zadajte „5.5.2016 16:00"
6. Ako čas príchodu zadajte „5.5.2016 18:30"
7. Potvrďte vytvorenie ponuky
Očakávaný stav na konci testu:
- *V databáze je vytvorená nová ponuka*
- *Nová ponuka je vyhľadateľná v rámci scenára „Vyhľadanie ponuky na prepravu"*

Logický model údajov

Čo a prečo:

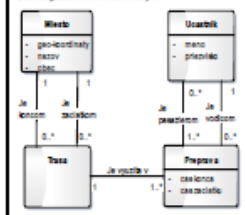
Identifikujú sa údaje, s ktorými softvér pracuje (entity, atribúty, vzťahy, stavy). Nevyhnutný krok pre správne a udržateľné uchovávanie údajov.

Využívané techniky:

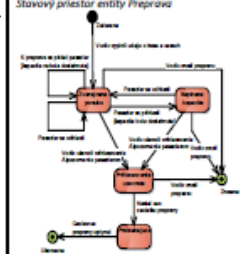
Diagram tried
Stavový diagram

Príklady:

Časť logického modelu údajov



Stavový priestor entity Preprava



Obrazovky

Čo a prečo:

Návrh obrazoviek (predloha pre budúci vývoj). Informačná architektúra (prepojenia obrazoviek).

Využívané techniky:

Náčrty obrazoviek (wireframes)
Orientované grafy (informačná arch.)

Príklad:

Návrh obrazovky vytvárania novej prepravy

Architektúra

Čo a prečo:

Štrukturálnym pohľadom na softvér „zhora" je jeho architektúra tvorená komponentami, ich opismi, rozhraniami a vzťahmi. Potrebujeme poznať úlohy jednotlivých komponentov. Potrebujeme tiež ukotviť softvér „v priestore", robiť rozhodnutia o výbere technológií, ale aj rozdeľovať prácu do menších celkov či rozdeľovať kompetencie vývojárov.

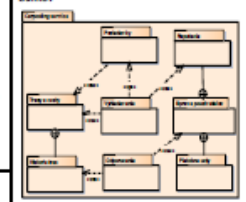
Využívané techniky:

Diagram komponentov
Diagram balíkov
Textové opisy

Príklady:

Architektúra diagramom komponentov

Vnútorná štruktúra súčiastky diagramom balíkov



Integračné testy

Čo a prečo:

Vytvorenie architektúry a rozhraní komponentov softvéru umožňuje napísať prvé automatické testy (integračné). Umožnia priebežne testovať súhru jednotlivých súčastí softvéru.

Využívané techniky:

Programovanie

Príklad:

Test metódy REST rozhrania komponentu CarpoolingService

```
public void
testAnyUpcomingPrepravaVrati()
throws ClientProtocolException, IOException
{
  // Given
  HttpGetRequest request =
    new HttpGet("http://api.cest.sk/search/auto");

  // When
  HttpResponse response =
    HttpClientBuilder.create().build()
    .execute( request );

  // Then
  User resource =
    RetrieveUtil.retrieveResourceFromResponse(
      response, User.class);
  assertThat "jozko",
    Matchers.is( resource.getLogin() ) );
}
```

Rozhrania a kompozícia tried

Čo a prečo:

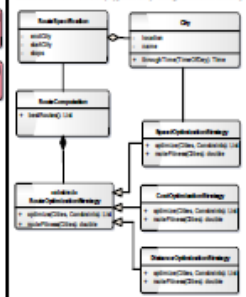
Poznajúc požadovanú funkcionalitu a architektonickú štruktúru softvéru, identifikujú sa programové triedy softvéru, navrhnú sa ich vzťahy a rozhrania (atribúty a deklarácie verejných metód). Niektoré vzniknú ako analógia dátových entít, iné na realizáciu používateľských rozhraní, ďalšie budú držať algoritmy. Identifikáciu tried potrebujeme pre ďalšie dekomponovanie softvéru.

Využívané techniky:

Diagram tried
Diagram objektov
Programovanie

Príklad:

Časť modelu tried (vyjadrený diagramom tried)



Modelová situácia inštancií niektorých tried (vyjadrený diagramom objektov)

Jednotkové testy

Čo a prečo:

Analogicky ako pri integračných testoch, k vytvorenému rozhraniam tried vytvárame jednotkové testy (unit tests). Umožnia sledovať vznik chýb v jednotlivých metódach.

Využívané techniky:

Programovanie

Príklad:

Jednotkový test triedy City, metódy throughTime

```
public class CityTest {
  private City city;
  private Time testing = new Time(0,0,0);

  @Before
  public void setUp() {
    city.distance = 1;
    city.population = 500000;
    city.main_road_type = Road.highway;
  }

  @Test
  public void testThroughTimeNonZero() {
    assertTrue(city.throughTime(testing) > 0);
  }
}
```

Implementácia (nielen tried)

Čo a prečo:

Implementáciou triedy rozumieme programovanie samotného „vnútra" triedy. To zahŕňa predovšetkým programovanie vystavených metód, definície pomocných atribútov a metód a pod. V širšom zmysle implementácia znamená aj ďalšie aktivity spojené s realizáciou softvéru ako napr. programovanie používateľských rozhraní, konfiguračných súborov a pod. Čo všetko? To veľmi záleží od použitých technológií, ale aj typu vytváraného softvéru. Implementáciou sa softvér stáva konkrétnou, použiteľnou vecou.

Využívané techniky:

Programovanie

Fyzický model údajov

Čo a prečo:

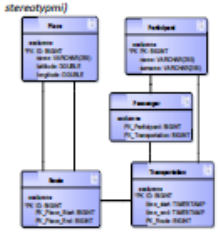
Vytvorí sa fyzický model údajov, ktorým už presne určíme, ako budú v databáze uchovávané údaje. Databázu tak môžeme fyzicky vytvoriť a používať.

Využívané techniky:

Diagram tried
Programovanie

Príklad:

Fyzický model údajov (diagram tried so stereotypmi)



Štruktúra pri nasadení

Čo a prečo:

Uskutočnia sa rozhodnutia o konkrétnom umiestnení (hardvéri, virtuálnom prostredí) nasadeného softvéru, aby mohol byť reálne spustený a používaný.

Využívané techniky:

Diagram rozmiestnenia (deployment diagram)

Príklad:

Rozmiestnenie softvérového systému (diagram rozmiestnenia)

Analýza	Špecifikácia	Návrh	Implementácia