Baltijos Programinė Įranga

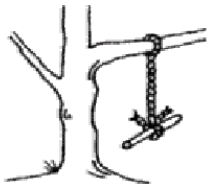
Reikalavimų analizė

Trumpas įvadas į praktinius metodus, naudojamus kuriamos programinės įrangos reikalavimų analizei, efektyviam jų dokumentavimui bei reikalavimų realizavimo ir pakeitimų valdymui viso projekto metu

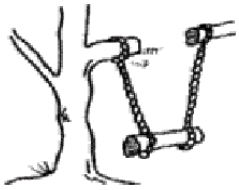
Reikalavimų inžinerija

- Reikalavimų procesas nustato servisus, kurių jūsų klientai reikalauja iš sistemos, ir taikomus sistemos veikimo ir kūrimo apribojimus
- Reikalavimai yra sistemos servisų ir apribojimų aprašymai
- Reikalavimai nurodo *KĄ* sistema turi daryti, bet ne *KAIP* sistema turi atlikti užduotis

Tipinio programinės įrangos projekto iliustracija

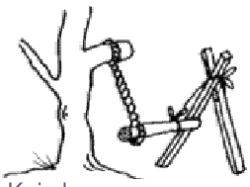


Ko prašė vartotojas

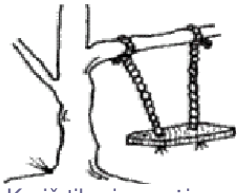


Kaip suprato analitikas

Kaip buvo suprojektuota



Kaip buvo suprogramuota



Ko iš tikrųjų norėjo vartotojas

Pagaliau veikia …

Dažniausiai nurodomos reikalavimų problemos

- Nepakankamas vartotojo įtraukimas
- Nepilni ir neaiškūs reikalavimai
- Reikalavimai pasikeitimai
- Neefektyvios komunikacijos priemonės
- …

Reikalavimų valdymo privalumai (pagal RUP)

- Reikalavimų valdymas tampa disciplinuota veikla
- Komunikacijos yra grindžiamos apibrėžtais reikalavimais
- Reikalavimai yra prioritetizuojami, filtruojami ir sekami
- Galimas objektyvus funkcionalumo ir veikimo įvertinimas
- Netikslumai yra lengviau nustatomi
- Galima saugoti reikalavimus, jų atributus ir ryšius, susieti juos su specifikacijomis, išoriniais dokumentais
  - Čia gali praversti specializuotas reikalavimų valdymo įrankis

Sistemos analitiko atsakomybės (pagal RUP)

- Suprasti vartotojo poreikius
- Suprasti suinteresuotų asmenų poreikius
- Dokumentuoti, prioritetizuoti ir komunikuoti reikalavimus
- Tvirtinti reikalavimus ir siekti, kad vartotojas priimtų sukurtą programinę įrangą

Kas yra reikalavimai?

Reikalavimas – funkcionalumas arba sąlyga, kurią turi įgyvendinti sistema

FURPS+ yra reikalavimų modelis, apibrėžiantis, ką vartotojas gali pasakyti apie sistemą

- Funkcionalumas (Functionality)
- Vartojimo patogumas (Usability)
- Patikimumas (Reliability)
- Veikimas (Performance)
- Palaikomumas (Supportability)

Reikalavimų vaidmuo programinės įrangos kūrimo procese

Pagal reikalavimus vertinama:

- projekto apimtis,
- nustatomas biudžetas,
- planuojami resursai,
- darbų tvarkaraščiai,
- ruošiami testavimo ir produkto priėmimo planai,
- dalis vartotojo dokumentacijos.

Reikalavimai yra pagrindas, kuriuo remiantis atliekamas projekto sekimas ir kontrolė

Prasto reikalavimų inžinerijos proceso rezultatai

Nepakankamas vartotojo įtraukimas sąlygoja nepriimtino produkto sukūrimą

- Nėra jokios alternatyvos, galinčios pakeisti vartotojų įtraukimą į projekto kūrimą nuo pat pradžių ir pastovų bendradarbiavimą viso projekto metu

Reikalavimų pakeitimai sąlygoja apimties padidėjimą, biudžeto viršijimus, vėlavimą ir kokybės prastėjimą

Neaiškūs reikalavimai sąlygoja neefektyvų darbą ir perdarymus

Nereikalingų savybių įgyvendinimas neprideda esminio funkcionalumo arba patogumo, tačiau padidina projekto apimtį

Minimaliai ruošiamos specifikacijos sąlygoja pamirštus reikalavimus

- Šios problemos simptomas yra reikalavimų ruošimas, kada sistemos įgyvendinimas jau yra įpusėjęs

Nepakankamas atskirų vartotojų klasių poreikių išsiaiškinimas sąlygoja vartotojo nusivylimą ir jiems nepriimtiną produkto funkcionalumą

Nepilnai apibrėžti reikalavimai neleidžia tiksliai suplanuoti ir sekti projekto

Pagrindiniai reikalavimų valdymo dalyviai ir priemonės

Galutiniai vartotojai ir užsakovai

- pateikia savo poreikius programinei įrangai;

Sistemų analitikai

- tai specialistai, kurie sugeba kvalifikuotai apibrėžti, aprašyti reikalavimus.

Specializuoti reikalavimų valdymo įrankiai

- leidžia suteikti prioritetus, saugoti reikalavimus, jų atributus ir ryšius;

Reikalavimų pakeitimų valdymo procedūros

- leidžia skaidriai, pagrįstai, objektyviai ir operatyviai valdyti besikeičiančius reikalavimus.

Požiūriai į reikalavimų valdymą

Tradicinis požiūris:

akcentuojamas išsamus reikalavimų dokumentavimas ir formalus reikalavimų valdymas.

Agile požiūris:

minimalus reikalavimų dokumentavimas, tiesioginis bendravimas su vartotojais, iteratyvus kūrimas, greitas reagavimas į pakeitimus.

Mūsų praktika – balansavimas tarp tradicinio ir Agile požiūrio

Skirtingi metodai MagicDraw ir užsakomuosiuose projektuose, bei sistemų analitikų specializacija.

Reikalavimų kategorijos

Funkciniai reikalavimai nusako veiksmus, kuriuos sistema turi galėti atlikti, neįvertinant galimų apribojimų

Kiti reikalavimai yra nefunkciniai, aprašantys tiktai sistemos ir sistemos aplinkos apribojimus ir atributus

Funkciniai reikalavimai

Nurodo sistemos įėjimo ir išėjimo veikimą

Savybių rinkiniai (feature sets)

Sistemos administravimas

Vartotojų identifikavimas ir autorizavimas

Pavyzdžiai:

- Bibliotekininkas gali užregistruoti paskolinimus ir grąžinimus
- Vartotojai (tiek bibliotekininkai, tiek klientai) turi prisijungti prie sistemos

Nefunkciniai reikalavimai

Nefunkciniai reikalavimai yra apribojimai kuriamai sistemai:

- Naudojimo patogumas (žmogiškieji faktoriai, estetika, pagalba, vedliai, vartotojo dokumentacija);
- Patikimumas (MTBF, atstatomumas, tikslumas);
- Veikimas (greitis, pasiekiamumas, pralaidumas, atsakymo laikas, resursų naudojimas);
- Palaikomumas (testuojamumas, praplečiamumas, konfigūravimo galimybės, instaliavimas, pritaikomumas, pagalba vartotojams);
- Projektavimo reikalavimai (techniniai nurodymai ir apribojimai);
- Realizacijos reikalavimai (programavimo kalbos ir technologijos, sąsajos standartai, palaikomos operacinės sistemos);
- Sąsajos su kitomis sistemomis.

Nefunkcinių reikalavimų pavyzdžiai

Sistema turi būti realizuota Java kalba, naudojant J2EE technologijas

Bibliotekininko vartotojos sąsaja turi būti realizuota pagal *Java Look and Feel Design Guidelines* stiliaus nurodymus

Skaitytojo vartotojo sąsaja turi būti prieinama per Internet naršykles, naudojant HTTP protokolą

Sistema turi būti pritaikoma įvairiam inventoriui – knygoms, muzikiniams ir video įrašams, elektroniniams dokumentams – ir turi turėti galimybę apibrėžti naujus inventoriaus tipus

Sistema turi galėti dirbti su iki 1 000 000 inventoriaus įrašų

Sistema turi palaikyti 100 000 paskolinimų vienu metu

Sistema turi palaikyti iki 100 lygiagrečiai prisijungusių vartotojų vienu metu

Kiekvienam vartotojo veiksmui turi būti prieinama kontekstinė pagalba ir patarimai

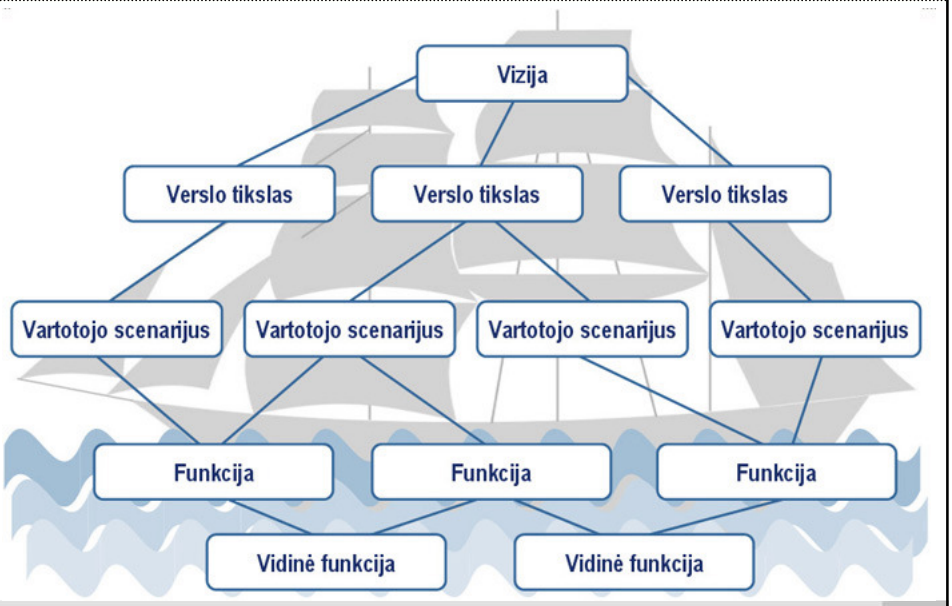
Reikalavimų lygiai

Reikalavimai gali būti skirtingų lygių:

- Verslo reikalavimai;
- Vartotojų reikalavimai;
- Funkciniai reikalavimai.

✓**Reikalavimų lygius puikiai iliustruoja Cockburn laivo modelis**

Cockburn laivo modelis



Gerų reikalavimų atributai

- Išsamūs
- Teisingi (*correct*)
- Nesunkiai įgyvendinami (*feasible*)
- Reikalingi
- Turi priskirtus prioritetus
- Aiškūs (nedviprasmiški)
- Patikrinami

Reikalavimų specifikacijos charakteristikos

Pilnumas

- Neturi būti trūkstamos informacijos

Suderinamumas

- Reikalavimai turi būti tarpusavyje suderinti ir neprieštarauti viens kitam arba aukštesnio lygio sistemos reikalavimams

Keičiamumas

- Turi būti galimybė keisti reikalavimus ir sekti pakeitimų istoriją

Atsekamumas

- Turi būti galimybė susieti reikalavimus tarpusavyje bei su projektavimo, realizavimo, testavimo planų elementais

Baltijos Programinė Įranga

Verslo modeliavimas, naudojant UML

Supažindinimas su statiniu ir dinaminiu dalykinės verslo srities modeliavimu, naudojant UML klasių, veiklos ir būsenų diagramas. Pristatoma naudojamų UML diagramų elementų sintaksė

Kas yra UML?

UML (*Unified Modeling Language*) yra populiariausia grafinė notacija, skirta objektiškai orientuotų programinės įrangos sistemų modeliavimui

UML yra naudojama ir kitų sričių modeliavimui

- Verslo procesų
- Duomenų bazių
- XML schemų
- Programinės įrangos kūrimo procesų
- Mokslinių eksperimentų
- ...

"Paveikslas yra vertas tūkstančio žodžių, o UML modelis – dar daugiau"

UML savybės

13 rūšių diagramos

Grafinė modeliavimo kalba

Industrinis notacijos standartas

Unifikuota terminologija

Efektyvus būdas

- Vizualizuoti
- Specifikuoti
- Dokumentuoti

Dalykinės srities modeliavimas

Dalykinės srities esybių ir jų tarpusavio ryšių modeliavimas naudojant UML klasių diagramas

Veiksmų sekų, verslo procesų modeliavimas, duomenų srautų modeliavimas naudojant UML veiklos diagramas

Svarbių esybių būsenų kaita naudojant UML būsenų diagramas

UML klasių diagramos

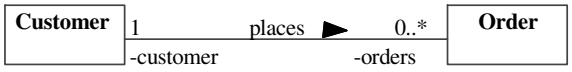
Modeliuoja kas sudaro sistemą, kurią mes norime aprašyti

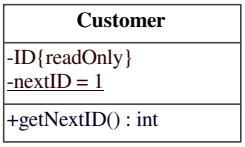
Klasifikacija yra naudoja supaprastinti sudėtingų sistemų aprašymus

Objektai vaizduoja atskirus egzempliorius, sukurtus pagal bendrą šabloną, kurį aprašo klasė

Klasės aprašas įtraukia atributus (duomenys) ir operacijas (veiksmai), kurios gali būti atliekamos naudojant atributus

Klasės diagramos pavyzdys





Klasių diagramų lygiai

Sąvokų (analitinė) klasių diagrama

- Verslo analitikai naudoja klasių diagramas sukurti dalykinės srities sąvokų modelį
- Neturi jokių realizacijos detalių, tokių kaip duomenų tipai arba savybių matomumas
- Modeliuoja tiktai “realaus pasaulio” klases – jokių programavimo klasių!

Projektavimo klasių diagrama

- Projektuotojai naudoja klasių diagramas sumodeliuoti realizacijas, atliekančias reikalaujamas užduotis
- Atvaizduojami architektūros elementai, tokie kaip projektavimo šablonai, karkasai
- Gali turėti realizacijos detalias, pvz. duomenų tipus ir matomumo modifikatorius

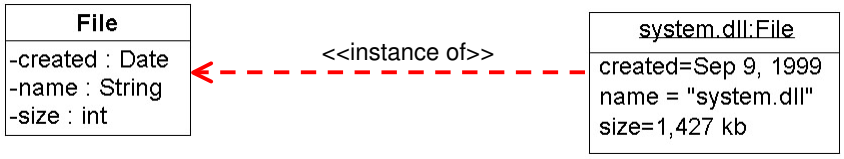
Realizacijos klasių diagrama

- Nuo platformos (OS, DBMS, programų serverio) priklausomos detalės, sąvybės, žyminės reikšmės
- Gali būti naudojamos sugeneruoti kodą
- Gali būti atsatomos iš parašyto kodo

Klasės ir objektai

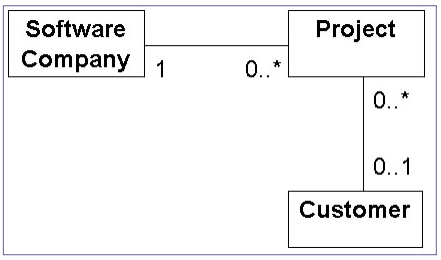
Objektas yra daiktas, kuriuo mes manipuliuojame

Klasė yra objektų tipo aprašymas



Klasių diagramos

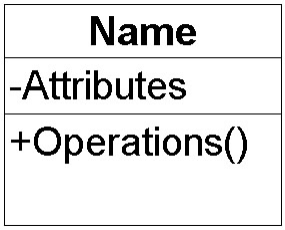
Klasių diagramos atvaizduoja statinę sistemos struktūrą, naudojamos klases ir ryšius tarp jų



Klasė – notacija

Klasė yra keturkampis padalintas į tris skyrius

- Pavadinimo
- Atributų
- Operacijų



Klasių identifikavimas

Kandidatai į klases

- Informacija, kuri gali būti saugoma arba analizuojama
- Daiktavardžiai užduoties apibrėžime
- Sistemos vartotojų grupės (rolės)

Klasės vardas

- Privaloma savybė
- Turi atitikti pavadinimą dalykinėje srityje
- Nedviprasmiškas, paprastai daiktavardis
- Vienaskaitai

Klasių radimas – pavyzdys

Librarian can register loans and returns. Librarian can see customer status report, which shows what books are loaned to him. Customer can also see his status report. For accessing system functionality users (both librarians and customers) have to login to the system.

Asociacijos ryšys

Ryšys tarp dviejų klasių

Nurodo, kad vienos klasės objektas gali turėti nuorodas į kitų klasių objektus

Leidžia vienos klasės objektams naudoti pasiekiamus kitos klasės objekto metodus ir atributus

Pagal nutylėjimą dvipusis

Paprasta asociacija

Dažniausiai pasitaikanti asociacija

Gali turėti krypties apribojimą

Abu asociacijos galai turi kiekybiškumo rodiklius

Kiekybiškumas

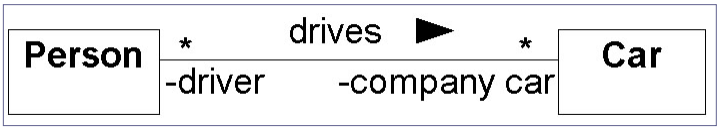
Nurodo, kiek esybių dalyvauja kiekvieno ryšio gale

- 1 – vienas ir tiktai vienas
- 0..1 – nulis arba vienas
- *, 0..* – nuo nulio iki bet kokio teigiamo skaičiaus
- 1..* – nuo vieno iki bet kokio teigiamo skaičiaus
- m..n – nuo m iki n (natūralūs sveikieji skaičiai)

Asociacijos galas

Asociacijos galas nurodo rolę, kurią klasė atlieka asociacijoje

Svarbu nurodyti vardą bent vienoje pusėje



Verslo procesų modeliavimas

Analizuojant verslo sritį reikia modeliuoti ne tik esybes ir jų statinius ryšius, bet ir kokius veiksmus įtraukia verslo procesai, kokiais duomenimis operuojama

Verslo procesai dažniausiai modeliuojami naudojant UML veiklos diagramas

Baltijos Programinė Įranga

Funkcinių reikalavimų aprašymas, taikant panaudojimo atvejus

Supažindinimas su panaudojimo atvejų metodika, skirta funkcinių reikalavimų rinkimui, analizei ir dokumentavimui, pristatant UML panaudojimo atvejų diagramos elementus

Panaudojimo atvejų modelis

Panaudojimo atvejai yra modeliavimo technika, leidžianti aprašyti, ką kuriama sistema turi daryti arba ką daro jau egzistuojanti sistema

Funkcinių reikalavimų rinkimas

Renkant funkcinius reikalavimus reikia atsakyti į tokius klausimus:

- Ką sistema turi galėti daryti?
- Ką reikia atlikti vartotojams su sistema?

Panaudos atvejų modelis leidžianti atsakyti į šiuos klausimus reikalavimų rinkimo veikloje ir vizualizuoti atsakymus diagramose

Panaudojimo atvejų modelis

Modeliuoja, ką vartotojai turi galėti atlikti su sistema

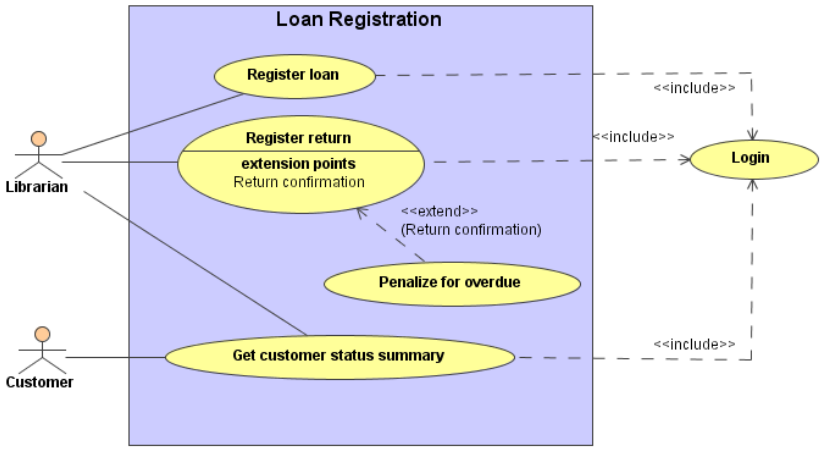
Leidžia analizuoti, apibrėžti ir suprasti funkcionalumą

Panaudojimo atvejų diagramos tiesiogiai įtraukiamos į funkcinių reikalavimų specifikaciją ir ten detalizuojamos

Panaudojimo atvejai yra netiesiogiai naudojami nefunkcinių reikalavimų apibrėžimui – nefunkciniai reikalavimai gali būti susiejami su panaudojimo atvejais

Kiekviena kūrimo proceso fazė turėtų būti pagrįsta panaudojimo atvejų realizacija

Panaudojimo atvejų modelis – pavyzdys



Panaudojimo atvejų kūrimo žingsniai

Nuspręskite, kokie bus sistemos aktoriai (vartotojų tipai)

Apibrėžkite panaudojimo atvejus kiekvienam aktoriui

Sukurkite ryšius

- Tarp aktorių
- Tarp aktorių ir panaudos atvejų
- Tarp panaudos atvejų

Panaudojimo atvejų modelis nebūtinai turi būti grafinis – tai gali būti ir tekstinis dokumentas, lentelė

Aktorius

Aktorius yra rolė, kurią atlieka žmogus arba kita sistema, sąveikaudamas (-a) su analizuojama sistema

Aktorių kategorijos

- Pirminiai aktoriai
- Antriniai aktoriai
- Išorinės sistemos

Aktorių identifikavimas

Identifikuojant aktorius, reikia atsakyti, kas

- … naudoja sistemą?
- … gauna iš sistemos informaciją?
- … teikia sistemai informaciją?
- … paleidžia ir sustabdo sistemą?
- … instaliuoja ir palaiko sistemą?
- Kokios kitos sistemos naudoja ar naudos sistemą

Praktiniai patarimai aktorių modeliavimui

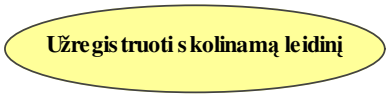
- Aktoriaus pavadinimas turėtų būti daiktavardis arba daiktavardinė grupė
- Aktoriaus pavadinimas turi būti unikalus
- Aktorius nebūtinai turi būti žmogus, tai gali būti ir kita sistema, tačiau aktorius turi būti sistemos išorėje
- Jeigu yra grupė bendrų funkcijų, kurias gali atlikti visi arba dalis aktorių, reikėtų išskirti bendrą abstraktų aktorių, iš kurio paveldėti konkrečius aktorius, o bendrus panaudojimo atvejus priskirti abstrakčiam aktoriui
- Kiekvienas aktorius turi turėti bent vieną su juo susietą panaudojimo atvejį
- …

Panaudojimo atvejis

- Panaudojimo atvejis yra tipinis bendradarbiavimas tarp aktoriaus ir sistemos, kurie padeda aktoriui pasiekti konkretų tikslą
- Panaudojimo atvejis apibrėžia vartotojui matomą funkciją, kuri yra reikšminga, duoda apčiuopiamą rezultatą, yra diskreti ir pilna

Pavyzdžiai

- Prisijungti prie sistemos
- Pasikeisti slaptažodį
- Užregistruoti skolinamą leidinį
- Peržiūrėti skaitytojo istoriją
- Atlikti leidinio paiešką
- …



Panaudojimo atvejų radimas

Kiekvienam aktoriui reikėtų paklausti:

- Ką aktorius gali daryti su sistema?
- Ar aktoriui reikia nuskaityti, sukurti, ištrinti, modifikuoti arba saugoti kokią nors informaciją naudojantis sistema?

Asociacijos ryšys

Nurodo, kad aktorius sistemoje gali atlikti panaudojimo atvejo apibrėžiamą funkcionalumą

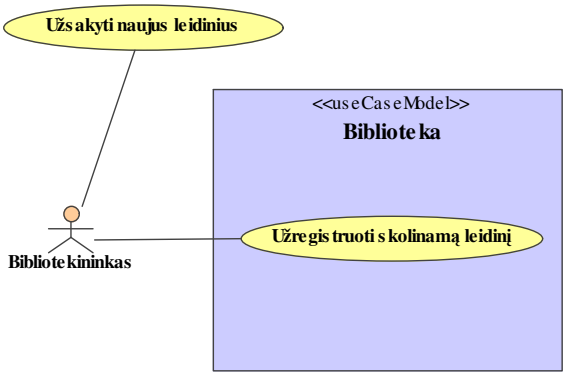
- aktorius ir panaudojimo atvejis bendrauja – siunčia viens kitam informaciją

Sistema ir jos ribos

Sistema yra tai, ką jūs planuojate sukurti

Galima modeliuoti kaip sistemas tiek verslo organizacijas, tiek programinės įrangos sistemas

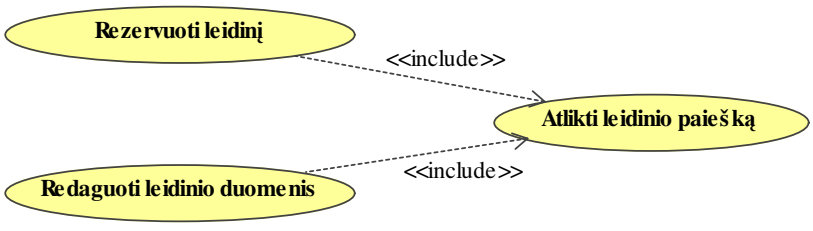
Sistemos ribos apibrėžia, kas yra sistemoje ir ko ten nėra



Panaudojimo atvejų asociacija <<include>>

Asociacijos <<include>> ryšys tarp dviejų panaudojimo atvejų naudojamas tada, kai vieno panaudojimo atvejo funkcionalumas visada įtraukia kitą panaudojimo atvejį

Šį ryšį naudoti tada, kai tam tikras funkcionalumas gali būti pakartotinai naudojamas (*reused*)

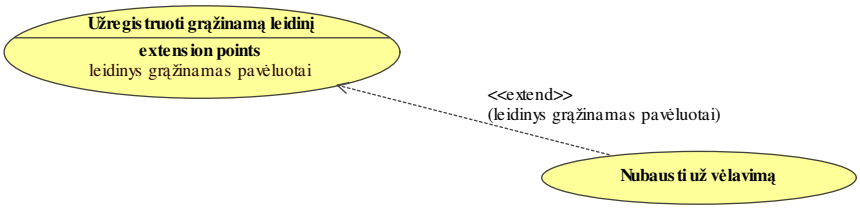


Panaudojimo atvejų asociacija <<exclude>>

Asociacijos <<*exclude*>> ryšys tarp dviejų panaudojimo atvejų naudojamas tada, kai vieno panaudojimo atvejo funkcionalumas specialiais atvejais įtraukia kitą panaudojimo atvejį

Veiksmas, prieš kurį gali būti įtraukiamas praplėtimo panaudojimo atvejis, vadinamas *extension point*

Taip pat nurodoma sąlyga, kuri leidžia įtraukti praplečiantį panaudojimo atvejį



51

Panaudojimo atvejų specifikacija

Kadangi kiekvienas panaudojimo atvejis yra pagrindas sistemos kūrimui, reikia detaliau aprašyti ir sumodeliuoti panaudojimo atvejus

Panaudojimo atvejai nurodo, ką sistema turi daryti, bet neaprašo kaip

Kiekvienam panaudojimo atvejui turi būti paruošta jo detalaus aprašymo specifikacija

Panaudojimo atvejų specifikacijoms naudojami

- Specifikacijų dokumentai, kuriuose aprašomos panaudojimo atvejų savybės
- Viena ar daugiau veiklos diagramų, kurios iliustruoja panaudojimo atvejo realizacijos veiksmų sekas pagal pirminius ir šalutinius scenarijus

Praktiniai patarimai panaudojimo atvejų modeliavimui

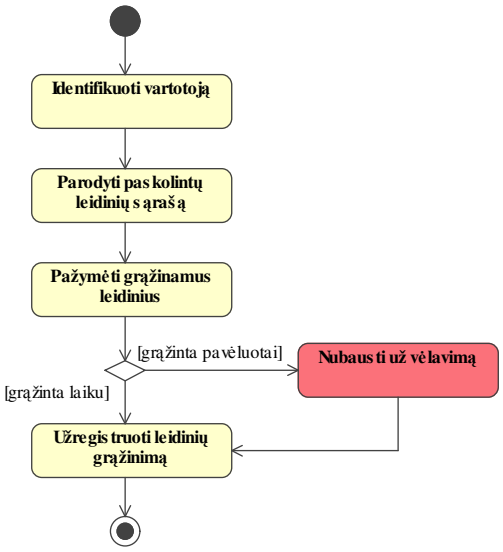
- Panaudojimo atvejo vardas turi prasidėti veiksmažodžio bendratimi ir nurodyti objektą su kuriuo atliekamas veiksmas
  - Rezervuoti leidinį
  - Užregistruoti leidinio grąžinimą
- Kiekvienas panaudojimo atvejis turėtų būti susietas bent su vienu aktoriumi arba kitu panaudojimo atveju
- Panaudojimo atvejis turi būti apibrėžiamas pakankamai konkrečiai – turi būti galima apibrėžti pirminį scenarijų
  - Panaudojimo atvejis "tvarkyti knygas" pernelyg abstraktus, nes apima pridėjimą, redagavimą, šalinimą, kuriems visiškai skirtingi scenarijai
- Panaudojimo atvejai turi būti vidinės sistemos funkcijos
- Jeigu tarp aktoriaus ir panaudojimo atvejo yra ryšys, turi būti galima pasakyti
  - *aktoriaus_pavadinimas* gali *panaudojimo_atvejo_pavadinimas*

Veiklos diagrama

- Veiklos diagrama parodo proceso veiksmų seką
- Veiklos diagramos paprastai yra naudojamos analizuojant dalykinės srities procesus ir detalizuojant panaudojimo atvejų funkcionalumą

Veiklos diagramos pavyzdys

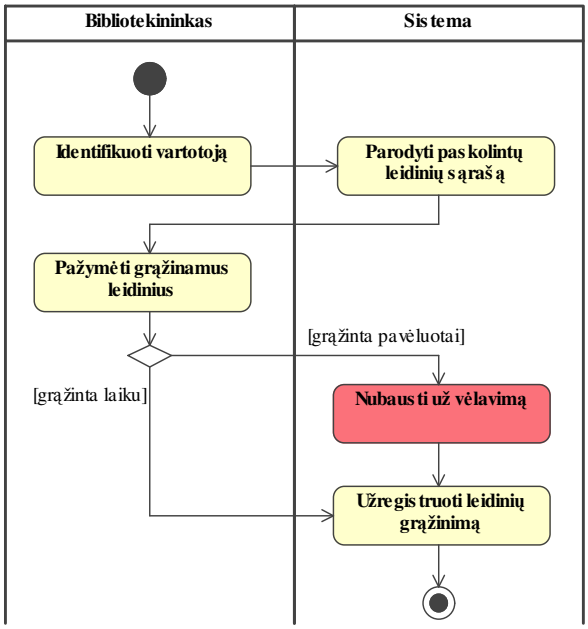
Panaudojimo atvejo *Užregistruoti leidinio grąžinimą* funkcionalumą detalizuojanti veiklos diagrama



Veiklos diagramos elementai

	Pradžia
	Pabaiga
	Veiksmas Vaizduoja veiklą, kuri duoda rezultatą
	Sprendimo taškas Vaizduoja tašką, kuriame išsiskiria galimos veiksmų sekos, priklausomai nuo tenkinamų sąlygų
	Perėjimas Gali turėti sąlygą, nurodančią, kada perėjimas įvyksta

Veiklos diagrama su sričių padalijimais



Scenarijus

Viena vykdymo metu pereinama veiksmų seka vadinama scenarijumi

Paprastai panaudojimo atvejis apima kelis scenarijus

Normali veiksmų seka, kada viskas vyksta gerai, vadinama *pirminiu scenarijumi*

Kiti (išimtiniai) scenarijai vadinami *šalutiniais* ir įvyksta, kai

- Tam tikri papildomi veiksmai gali būti kartais atliekami kažkuriame pirminio scenarijaus taške
- Įvyksta kokia nors klaida arba netikslumas pirminiame scenarijuje
- Yra funkcionalumas, kuris gali įvykti bet kuriuo metu

Panaudojimo atvejo specifikacija

Panaudojimo atvejį aprašanti specifikacija turėtų turėti tokią informaciją

- Unikalų ID
- Tikslą
- Autorių
- Prioritetą
- Rizikingumą
- Prielaidas
- Išankstines sąlygas (*pre-conditions*) ir aktyvaciją
- Pirminį scenarijų
- Šalutinius scenarijus
- Užbaigimą ir pabaigos sąlygas (*post-conditions*)
- Tenkinamus reikalavimus

Use Case Specification Document – Example(1)

Use Case Name	Place Order
Current Status	Designed, but not implemented
Goal Statement	This use case should allow the user to place orders for tickets.
Author	Victor Peters
Priority	High
Risks	There are some technological risks that we have to be certain the scheme we use is fail proof and works 100% of the time since the orders cannot be lost.
Pre-conditions	The user must already be authenticated.
Post-conditions	The order to send the tickets and to bill the customer must be committed.

Use Case Specification Document – Example(2)

Flow of Events	1. The user selects a show 2. The user is presented with a display of the shows and selects an available seat. 3. The users enters their personal data if it is not already in the system. 4. The system submits the order and displays a thank you message to the user.
Alternative Flows	1) There may be no seats available and the user is unable to place an order. 2) If there is an internal failure in the system, the user must get message that the order has not been submitted (the order cannot be lost while the user thinks it has been placed).
Non-Functional Requirements (Security, Performance, etc.)	The user must not have to wait more than five seconds for the order to be processed.
Notes	

Baltijos Programinė Įranga

Programinės įrangos reikalavimų specifikacijos

Reikalavimų dokumentavimas

Būdai dokumentuoti programinės įrangos reikalavimus

- Tekstiniai dokumentai, kurie naudoja aiškios struktūros ir tiksliai parašytą natūralią kalbą
- Grafiniai modeliai, iliustruojantys pasikeitimų procesus, sistemos būsenas ir pasikeitimus tarp jų, duomenų sąryšius, logines veiksmų sekas, objektų klases ir jų ryšius
- Formalios specifikacijos, apibrėžiančios reikalavimus naudojant tikslias formalias kalbas

Programinės įrangos reikalavimų specifikacija

- PĮRS tiksliai nustato funkcijas ir galimybes, kurias programinės įrangos sistema turi palaikyti, ir apribojimus, kuriuos turi atitikti
- PĮRS yra pagrindai visam tolesniam projekto planavimui, projektavimui, realizacijai, testavimui ir vartotojo dokumentacijos ruošimui
- PĮRS turi aprašyti kiek įmanoma tiksliai išorinį, vartotojui matomą, sistemos funkcionalumą. Joje neturėtų būti projektavimo, realizacijos, testavimo ar projekto valdymo detalių, išskyrus numatytus projektavimo ir realizacijos apribojimus

PĮRS šablonas

Įvadas
- 1.1 Tikslas
- 1.2 Apimtis
- 1.3 Apibrėžimai, sutrumpinimai
- 1.4 Nuorodos
- 1.5 Apžvalga

Bendras aprašymas
- 2.1 Panaudojimo atvejų modelio apžvalga
- 2.2 Prielaidos ir priklausomybės

Reikalavimų specifikacijos
- 3.1 Panaudojimo atvejų aprašymai
- 3.2 Papildomi reikalavimai

Papildoma informacija

Papildomi reikalavimai

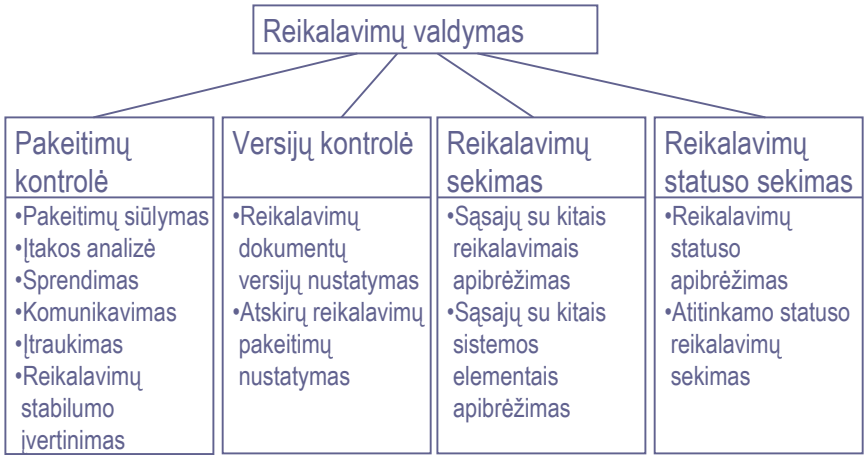
1. **Vartojimo patogumas**
2. **Patikimumas**
3. **Veikimas**
4. **Projektavimo apribojimai**
5. **Vartotojo dokumentacijos ir pagalbos sistemos reikalavimai**
6. **Perkami komponentai**
7. **Sąsajos**
   - 7.1 Vartotojo sąsaja
   - 7.2 Techninės įrangos sąsajos
   - 7.3 Programinės įrangos sąsajos
   - 7.4 Bendravimo sąsajos
8. **Licenzijavimo reikalavimai**
9. **Legalumo, autorinių teisių ir kitos pastabos**
10. **Taikomi standartai**

Baltijos Programinė Įranga

Programinės įrangos reikalavimų valdymas

Pagrindinės reikalavimų valdymo veiklos.
Pakeitimų valdymo procedūra

Pagrindinės reikalavimų valdymo veiklos



Reikalavimų pakeitimų valdymas

Nekontroliuojami pakeitimai yra dažna projekto chaotiškumo, vėlavimo ir kokybės problemų priežastis

Organizacija, kuri rimtai žiūri į programinės įrangos projektus, turi užtikrinti, kad:

- Siūlomi pakeitimai būtų atsakingai įvertinami
- Tinkami asmenys priimtų sprendimus apie pakeitimus
- Pakeitimai būtų perduoti visiems dalyviams
- Projektuose reikalavimų pakeitimų valdymas būtų disciplinuota ir gerai apibrėžta veikla

Pakeitimų kontrolės procesas

Gerai apibrėžtas pakeitimų kontrolės procesas suteikia suinteresuotiems asmenims formalų mechanizmą siūlyti reikalavimų pakeitimus

Pakeitimų kontrolės procesas leidžia projektų vadovams daryti gerai informuotus verslo sprendimus

Pakeitimų kontrolės procesas leidžia kontroliuoti visų pasiūlytų pakeitimų būsenas ir užtikrina, kad siūlomi pakeitimai nėra užmiršta arba neįvertinami

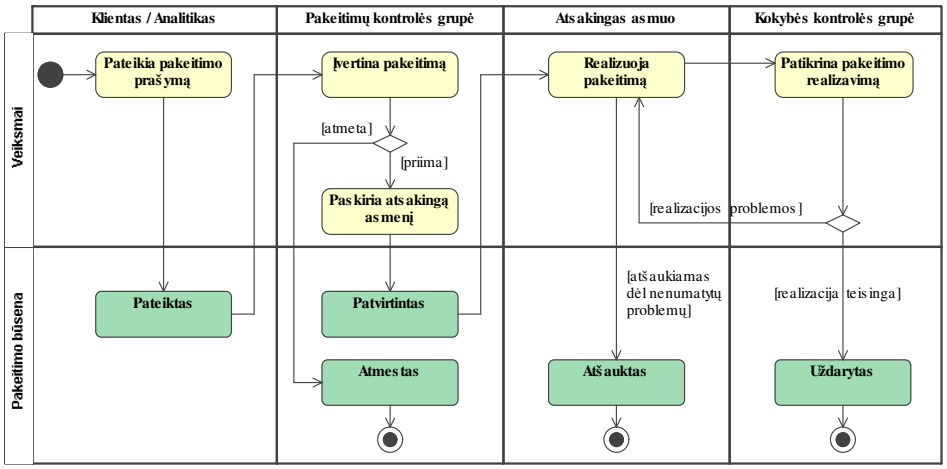
Kaip tik reikalavimai yra patvirtinami, reikia pradėti taikyti reikalavimų pakeitimų kontrolės procesą visiems siūlomiems pakeitimams

Pakeitimų kontrolės procesas yra saugojimo ir filtravimo mechanizmas, leidžiantis užtikrinti, kad tinkami pakeitimai būtų priimami ir kad neigiama įtaka projektui būtų minimizuojama

Pakeitimų kontrolės procesas turi būti gerai dokumentuotas, kiek įmanoma paprastesnis ir, svarbiausia, efektyvesnis

Pakeitimų kontrolė yra susieta su kitais projekto konfigūracijos principais

Pakeitimų kontrolės proceso diagrama



Pakeitimų valdymo santrauka

- Reikalavimų pakeitimai yra realybė, su kuria susiduriame beveik kiekviename programinės įrangos projekte
- Disciplinuotas pakeitimų valdymas gal padėti sumažinti projekto plano ardymą, kurį sąlygoja pakeitimai
- Efektyvesnės reikalavimų dokumentavimo technikos gali padėti sumažinti reikalavimų pakeitimų skaičių ir apimtį
- Efektyvios reikalavimų detalizavimo ir valdymo praktikos padidins galimybes pasiekti užsibrėžtus projekto tikslus