Informacijos transformacija programų inžinerijoje: nuo verslo intencijos iki veikiančios sistemos

Gytis Gurklys

2026-08-12

https://gytisgurklys.eu/lt/posts/informacijos-transformacija-programu-inzinerijoje/

ANOTACIJA

Šiame darbe nagrinėjamas programų inžinerijos procesas kaip nuosekli informacijos transformacija: nuo neišsakytos verslo intencijos iki produkcinėje aplinkoje veikiančios sistemos. Pasiūlytas šešių lygių transformacijos modelis, kiekvienam lygiui apibrėžiant įvestį, transformaciją, išvestį ir jam būdingą klaidų tipą. Pirmieji trys lygiai išskaidyti į smulkesnius žingsnius, sudarytas vientisas artefaktų registras su priklausomybių tinklu, iš registro išvestas rolių modelis ir suformuluoti bendrieji proceso projektavimo principai. Detaliai išnagrinėta veiklos analitiko (reikalavimų inžinieriaus) veikla pirmajame transformacijos lygyje. Darbo metodas – konceptuali dedukcinė analizė: modelis pateikiamas kaip vidiniu nuoseklumu grindžiama hipotezė, iliustruota sąskaitų faktūrų valdymo srities pavyzdžiais; visa grandinė pristatoma modelio granuliacija, detaliai nagrinėjamas pirmasis lygis. Nagrinėjimo kontekstas – grafinę vartotojo sąsają turinti verslo aplikacija, veikianti globaliu mastu ir aptarnaujanti tūkstančius vartotojų.

Raktiniai žodžiai: reikalavimų inžinerija, veiklos analizė, informacijos transformacija, programų inžinerijos procesas, artefaktų atsekamumas, validacija.

TURINYS

1. ĮVADAS

1.1. Tyrimo objektas ir aktualumas

Programų kūrimas tradiciškai aprašomas kaip veiklų seka: analizė, projektavimas, programavimas, testavimas, diegimas. Šiame darbe pasirinktas kitas pjūvis – programų kūrimas nagrinėjamas kaip informacijos transformacijos procesas, kurio metu natūralios kalbos teiginiai apie verslo tikslus laipsniškai paverčiami vykdomu kodu ir galiausiai veikiančia sistema.

Toks pjūvis aktualus dėl kelių priežasčių. Pirma, jis leidžia tiksliai lokalizuoti, kur ir kokio pobūdžio informacija prarandama arba iškraipoma, ir atitinkamai projektuoti prevencijos mechanizmus. Antra, jis paaiškina, kodėl tam tikri proceso žingsniai yra būtini, o ne konvenciniai: kiekvienas žingsnis atlieka apibrėžtą informacinį darbą, kurio neatlikus trūkstamą informaciją kas nors pagamins vėliau, dažniausiai nekontroliuojamu būdu. Trečia, jis suteikia vienijantį pagrindą artefaktų, rolių ir kokybės užtikrinimo praktikos analizei.

Reikalavimais šiame darbe vadinami teiginiai, aprašantys tikslus, kuriuos turi pasiekti iš kodo sukompiliuota ir produkcinėje aplinkoje veikianti programinė įranga.

1.2. Darbo tikslas ir uždaviniai

Darbo tikslas – pateikti nuoseklų informacijos transformacijos modelį, apimantį visą kelią nuo verslo intencijos iki veikiančios sistemos, ir jo pagrindu suformuluoti praktines gaires veiklos analitiko darbui. Aprėpties gylis sąmoningai netolygus: visa grandinė pristatoma modelio granuliacija, o detali analizė skiriama pirmajam lygiui; ketvirto–šešto lygių vidinis skaidymas paliktas tolesniam darbui.

Uždaviniai:

1. Apibrėžti transformacijos lygius, kiekvienam nurodant įvestį, transformaciją, išvestį ir būdingą klaidų tipą.
2. Išskaidyti pirmuosius tris lygius į žingsnius pagal vienodą išskyrimo kriterijų.
3. Sudaryti artefaktų registrą su unikaliais identifikatoriais ir priklausomybių tinklu.
4. Iš registro išvesti projekto rolių modelį.
5. Suformuluoti bendruosius proceso projektavimo principus.
6. Detaliai aprašyti veiklos analitiko veiklą pirmajame lygyje, įskaitant darbo technikas, kokybės kriterijus ir tipinius nesėkmių scenarijus.
7. Pozicionuoti modelį esamų programų inžinerijos teorijų atžvilgiu ir įvardyti jo patikrinimo kryptis.

1.3. Metodas ir apimties ribos

Darbas yra konceptualioji analizė: modelis konstruojamas dedukciškai, iš informacijos transformacijos prigimties, ir iliustruojamas pavyzdžiais iš sąskaitų faktūrų valdymo srities. Nagrinėjimo kontekstas – grafinę sąsają turinti verslo aplikacija globaliu mastu su tūkstančiais vartotojų; šis kontekstas lemia atskirų vietų akcentus (lokalizacija, lygiagretumas, regioniniai skirtumai), bet pats modelis nuo jo nepriklauso.

Darbo statusas – konceptuali hipotezė: modelio pagrindimas yra vidinis nuoseklumas ir dedukcija, o ne empirinis patikrinimas. Pavyzdžiai yra iliustracijos, ne įrodymai; vientisas perėjimas per pirmąjį lygį pateikiamas priede A. Kartu modelis formuluoja patikrinamas prognozes: pavyzdžiui, kad praleidimo tipo klaidos (praleista šalis, praleistas scenarijus) su vėlai aptinkamais ir brangiais defektais koreliuoja stipriau negu iškraipymo tipo klaidos, ir kad projektai be sisteminių generavimo procedūrų (1.1 žemėlapis, 2.4 klausimynas) turi didesnę tylių klaidų dalį. Tokias prognozes galima tikrinti retrospektyvine defektų analize, nesikišant į vykstantį procesą.

Darbe sąmoningai nenagrinėjama: verslo strategijos formavimas (jis vyksta iki pirmojo lygio ir už programų inžinerijos ribų), projektų valdymo metodikos (jos organizuoja procesą laike, bet nekeičia informacijos srauto struktūros) ir ketvirto–šešto lygių vidinis skaidymas (jis paliktas tolesniam darbui).

1.4. Darbo struktūra

Darbą sudaro dvi dalys. Pirmoje dalyje (2-8 skyriai) pateikiamas visas modelis: teorinis pagrindas, santykis su esamomis teorijomis, šešių lygių struktūra, projektavimo principai, lygių vidinis skaidymas, artefaktų registras, rolių modelis ir proceso topologija. Antroje dalyje (9-14 skyriai) detaliai nagrinėjama veiklos analitiko veikla: rolės apibrėžtis, pagrindinės darbo sąvokos, septynių žingsnių analizė, tylių klaidų prevencija ir kokybės vartai. Priede A vienas reikalavimas pravedamas per visus pirmojo lygio žingsnius; darbo pabaigoje pateikiamas literatūros sąrašas.

PIRMOJI DALIS. TRANSFORMACIJOS MODELIS

2. TEORINIS PAGRINDAS

2.1. Informacijos transformacijos samprata

Programų kūrimą galima aprašyti kaip nuoseklų informacijos kondensavimą: kiekviename lygyje mažėja neapibrėžtumas ir auga formalumas, kol natūralios kalbos teiginiai virsta vykdomu kodu, o kodas – veikiančia sistema. Informacija juda nuo atsakymo į klausimą „kodėl“ (tikslai) per „ką“ (reikalavimai, elgsena) ir „kaip“ (projektiniai sprendimai) iki vykdymo (kodas, veikianti sistema).

Terminas „informacija“ šiame darbe vartojamas inžinerine, o ne informacijos teorijos prasme: informacija čia yra priimti sprendimai ir pašalintas neapibrėžtumas dėl kuriamos sistemos, ne Shannono kiekybinis matas [14]. Teiginys, kad lygis „pagamina naują informaciją“, reiškia, kad jame priimami sprendimai, kurių ankstesniuose artefaktuose nebuvo, o ne kad didėja perduodamo pranešimo bitų kiekis. Ši skirtis esminė: grandinė gamina ne bitus, o sprendimus.

Kiekviena transformacija kartu yra praradimo rizika: galima dviprasmybė, praleistas scenarijus, klaidinga interpretacija. Iš to kyla atsekamumo reikalavimas – galimybė bet kurį žemesnio lygio artefaktą atsekti iki jį pagrindžiančio aukštesnio lygio artefakto.

Apibendrinta transformacijos seka per visą grandinę: tyli (neišsakyta) žinia → eksplicitinė nestruktūruota → struktūruota → formalizuota → patikrinta → sutarta → vykdoma. Kiekvienas etapas informacijai prideda po vieną savybę: užrašyta, struktūruota, patikrinama, sutarta.

2.2. Formalizavimas kaip kalbos siaurinimas

Formalizavimas techniškai yra kalbos siaurinimas: laisva natūrali kalba keičiama vis labiau ribota (šablonai, privalomi laukai, kontroliuojamas žodynas), kol pasiekiama visiškai determinuota kalba – kodas.

Esminis formalizavimo mechanizmas yra ne užrašymo forma, o priverstiniai pasirinkimai. Natūrali kalba leidžia neapibrėžtumui išlikti: formuluotės „protingas terminas“ ar „tinkamas vartotojas“ atrodo kaip informacija, bet iš tikrųjų yra atidėti sprendimai. Šablonas su privalomu lauku atidėjimo neleidžia. Todėl didžioji formalizavimo darbo dalis yra ne rašymas, o atidėtų sprendimų priėmimas arba jų perdavimas tam, kas turi įgaliojimus juos priimti.

Formalizavimo teikiama nauda:

1. Vienos interpretacijos garantija. Formalų teiginį skirtingi žmonės skirtingu laiku perskaito vienodai, autoriui nedalyvaujant; informacija tampa savarankiška, nebepririšta prie ją sukūrusio asmens. Tai kritiška, nes tarp lygių informacija keičia šeimininką.
2. Patikrinamumas. Formalus teiginys turi tiesos sąlygas, todėl iš jo galima išvesti testą dar neegzistuojant kodui.
3. Ankstesnis klaidų aptikimas. Prieštaravimas tarp formalių teiginių matomas mechaniškai specifikavimo etape, kur taisymo kaina nepalyginamai mažesnė nei produkcinėje aplinkoje.
4. Mechaninis apdorojamumas. Formalius artefaktus galima tikrinti įrankiais, sekti atsekamumą, generuoti iš jų kitus artefaktus.

Formalizavimo kaina – netikro tikslumo rizika: šablonas verčia rinktis tikslumą ir ten, kur turimos žinios jo nepagrindžia. Įrašyta konkreti reikšmė atrodo patikima vien dėl formos. Šios rizikos valdymo priemonė yra prielaidų disciplina (žr. 4.5 poskyrį). Apibendrinant: formalizavimas žinių nekuria, jis tik parodo, kur jų trūksta, ir tai laikytina vertingiausiu jo šalutiniu efektu.

2.3. Klasifikavimo funkcija

Klasifikavimas ir formalizavimas atlieka skirtingą darbą: klasifikavimas tvarko erdvę tarp teiginių, formalizavimas – erdvę kiekvieno teiginio viduje.

Surinkta žaliava natūraliai būna sutvarkyta pagal kilmę (kas pasakė, kuriame dokumente rasta), o naudojama ji pagal paskirtį (kurioje sistemos vietoje gyvens). Klasifikavimas yra perindeksavimas iš kilmės į paskirtį. Jis suteikia keturias savybes:

1. Palyginamumą. Tik tai pačiai klasei ir tam pačiam inkarui priskirti teiginiai tampa lyginami tarpusavyje; dublikatai ir prieštaravimai matomi tik bendroje dimensijoje.
2. Pilnumo klausimo galimybę. Klausimą „ar turime visas tam tikros rūšies taisykles“ galima užduoti tik tada, kai ta rūšis yra apibrėžta klasė; klasifikavimas nežinomus nežinomuosius paverčia suskaičiuojamais tuščiais langeliais.
3. Ankstyvą maršrutizavimą. Teiginio tipas nulemia, kurioje būsimos sistemos vietoje jis materializuosis (duomenų apribojimas – validavimo sluoksnyje, būsenų taisyklė – būsenų mašinoje, prieigos taisyklė – autorizacijos sluoksnyje); klasifikavimas yra ankstyvas architektūrinis sprendimas.
4. Formalizavimo kalbos parinkimą. Kiekviena klasė turi savo šabloną; klasifikavimas yra formalizavimo prielaida, nes teiginio negalima teisingai formalizuoti nežinant, kas jis yra.

2.4. Grandinės nesutraukiamumas

Hipotetinis atvejis, kai užsakovas formuluotų poreikius tiesiogiai programos kodu, leidžia atskleisti grandinės prigimtį. Kodas yra visiškai determinuota kalba: jame kiekvienas kraštinis atvejis, kiekviena klaidos šaka ir kiekvienas lygiagretumo scenarijus turi apibrėžtą atsakymą. Užsakovo turimoje informacijoje šių atsakymų nėra – atitinkami klausimai jo veikloje niekada nekilo. Iš to išplaukia esminė išvada: transformacijos grandinė egzistuoja ne tam, kad turimą informaciją išverstų į kitą notaciją, o tam, kad sukurtų trūkstamą – kiekvienas lygis generuoja sprendimus, kurių ankstesniame lygyje nebuvo. Kliūtis yra ne kalba, o trūkstami sprendimai.

Tą pačią mintį išreiškia klasikinis akcidentinio ir esminio sudėtingumo skyrimas [1]: notaciją ir įrankius galima tobulinti neribotai, bet esminis sudėtingumas (tiksliai nuspręsti, ko norima) niekur nedingsta. Tikslus specifikavimas ir yra programavimas, nepriklausomai nuo naudojamos kalbos.

Sprendimų požiūriu: jei išvestyje yra sprendimų, kurių įvestyje nebuvo, skirtumą privalo pagaminti žmogus, procesas arba spėjantis mechanizmas (informacijos sąvokos prasmė apibrėžta 2.1 poskyryje). Natūralios kalbos programavimas, įskaitant didžiuosius kalbos modelius, grandinės nepanaikina, o suspaudžia ją į dialogą: dviprasmybę arba išsprendžia tikslinamieji klausimai (reikalavimų išgavimo analogas), arba užpildo spėjimai – ir tuomet sistemos elgsenos kraštiniais atvejais niekas nėra sąmoningai nusprendęs. Ši tema plėtojama 2.6 poskyryje.

2.5. Santykis su esamomis teorijomis

Modelis nekuriamas tuščioje vietoje: dauguma jo statybinių blokų perimta iš esamų programų inžinerijos tradicijų. Sąžininga tiksliai įvardyti, kas skolinta, o kas nauja, nes originalumas matomas tik ant fono.

Perimta. Akcidentinio ir esminio sudėtingumo skirtis, kuria grindžiamas grandinės nesutraukiamumo argumentas (2.4), yra Brookso [1]. Reikalavimų kokybės kriterijai (atomiškumas, vienareikšmiškumas, patikrinamumas), validacijos ir verifikacijos skirtis bei funkcinių ir nefunkcinių reikalavimų atskyrimas ateina iš reikalavimų inžinerijos literatūros [3, 4, 5, 6]; išgavimo technikos (interviu, stebėjimas, situaciniai klausimai) detaliai aprašytos Gause ir Weinbergo [7]. Tikslų hierarchija ir tikslo ryšys su reikalavimu (1.3-1.5 žingsniai) atitinka tikslais grįstos reikalavimų inžinerijos tradiciją [15, 17]. Atsekamumas kaip savarankiška problema suformuluotas Gotel ir Finkelstein [8], o atskaitos linijos ir pakeitimų valdymo procedūros yra standartinė konfigūracijų

valdymo medžiaga [9]. Sprendimo įrašo forma (4.4) atitinka architektūrinių sprendimų įrašų praktiką [10]. Žodynas su įtvirtintais terminų apibrėžimais (2.1 žingsnis) atlieka tą patį darbą kaip vieninga kalba dalykinės srities projektavime [11], o sąveikos vienetų inventorius (2.2 žingsnis) remiasi panaudos atvejų metodika [12]. MVP mokymosi kriterijus (8.3) perimtas iš Lean Startup [16].

Nauja. Darbo originalumo pretenzija apima keturis dalykus. Pirma, klaidų tipas kaip žingsnio išskyrimo kriterijus (4.1): žingsnis pateisinamas tik tada, kai jame galima suklysti taip, kaip niekur kitur; literatūroje žingsniai dažniausiai skiriami pagal veiklą arba artefaktą. Antra, rolių modelio ir kokybės vartų išvedimas iš artefaktų registro sąsmaukų (6-7 skyriai), o ne postulavimas. Trečia, grandinės paskirties performulavimas: ne versti turimą informaciją, o gaminti trūkstamus sprendimus; ši mintis artima Nauro tezei, kad programavimas yra teorijos kūrimas [2], bet čia ji operacionalizuojama artefaktų ir žingsnių lygmeniu. Ketvirta, vieninga fraktalinė transformacijos seka (4.8), taikoma ir visai grandinei, ir kiekvieno lygio vidui.

Skirtumai. Nuo krioklio interpretacijos modelis skiriasi tuo, kad lygius laiko loginiais, o ne kalendoriniais etapais (žr. 3.1 ir 8.1-8.3 poskyrius). Nuo dalies lanksčiųjų metodikų praktikos skiriasi tuo, kad atskaitos linijų ir kokybės vartų nelaiko biurokratija: jie kyla iš informacijos šeimininko keitimosi sąsmaukose, todėl išlieka ir mažose iteracijose, tik mažesne apimtimi.

2.6. Didžiųjų kalbos modelių vieta grandinėje

Didieji kalbos modeliai keičia grandinės ekonomiką, bet ne jos struktūrą; modelis leidžia šį teiginį suformuluoti tiksliai.

Kas suspaudžiama. Kalbos modeliai pigina formalizavimo darbą, tai yra akcidentinį sudėtingumą: vertimą tarp notacijų, juodraščių generavimą, penktojo lygio transformaciją projektas → kodas ir dalį antrojo lygio mechanikos (scenarijų išrašymą pagal šabloną). Ten, kur transformacija yra vertimas iš jau turimos informacijos, kalbos modelis ją atlieka greitai ir dažnai gerai.

Kas išlieka. Esminis darbas pagal 2.4 yra trūkstamų sprendimų gamyba, ir jis niekur nedingsta – keičiasi tik vieta, kurioje vyksta. Dialogas su modeliu yra suspausta grandinė: tikslinamieji klausimai atitinka išgavimą, o kiekvienas neužduotas klausimas reiškia, kad spragą užpildė spėjimas. Spėjimas yra praleidimo tipo klaida: jo nesimato peržiūroje, nes sugeneruotas tekstas atrodo išbaigtas ir įtikinamas. Iš čia modelio prognozė: kalbos modelių naudojimas mažina iškraipymo klaidų dalį (mechaninis vertimas tikslus), bet be sisteminių padengimo procedūrų didina tylių praleidimų dalį, nes spėjimai gaminami mašinos greičiu ir įtikinama forma.

Ko kalbos modelis negali iš principo. Validacija (1.6 žingsnis) lygina artefaktą su etalonu, kuris egzistuoja tik šalių atmintyje (11.6 poskyris), todėl jos negalima deleguoti jokiam įrankiui, neturinčiam prieigos prie šalių intencijų. Mandato klausimai (4.3) – konfliktai, prioritetai, apimties ribos – yra vertės, ne žinojimo klausimai, todėl jų negalima deleguoti niekam, kas neturi įgaliojimų. Praktinė išvada: kalbos modeliai perkelia analitiko darbo svorio centrą nuo formulavimo (1.5) prie validacijos (1.6) ir suderinimo (1.7), o prielaidų disciplina (4.5) tampa dar svarbesnė, nes spėjimų šaltinių atsiranda daugiau.

3. ŠEŠIŲ LYGIŲ MODELIS

3.1. Lygių struktūra

1 lentelė. Transformacijos lygiai ir jų kalbos

Lygis	**Transformacija**	**Kalba**
1	Verslo intencija → reikalavimų atskaitos linija	Problemos
2	Reikalavimai → funkcinė specifikacija	Problemos
3	Specifikacija → architektūra	Sprendimo
4	Architektūra → detalusis projektas	Sprendimo
5	Projektas → kodas	Sprendimo
6	Kodas → veikianti sistema	Eksploatacijos

Informacijos pobūdis grandinėje keičiasi dviem šuoliais: tarp antro ir trečio lygio (nuo „ką sistema daro“ prie „iš ko ji sudaryta“) ir tarp penkto ir šešto (nuo teksto prie vykstančio proceso). Pagal tai lygiai grupuojasi į tris kalbas: pirmieji du kalba problemos kalba, trečias–penktas – sprendimo, šeštas – eksploatacijos.

Dvi pastabos dėl struktūros. Pirma, lygiai yra loginiai, o ne kalendoriniai etapai: modelis aprašo informacijos srauto struktūrą, o ne darbų seką laike. Per lygius einama mažomis atkarpomis ir kartotinai (8.1-8.3 poskyriai), o atskaitos linija galioja iteracijos apimčiai, ne visam produktui iš anksto (8.4). Antra, du kalbos šuoliai fiksuoja tik ribas tarp trijų kalbų; vidinė granuliacija tarp jų (pavyzdžiui, trečio ir ketvirto lygių atskyrimas) yra projektinis pasirinkimas, pagrįstas žingsnio išskyrimo kriterijumi (4.1), bet ne vienintelis galimas.

3.2. Lygių charakteristika

Pirmas lygis. Verslo intencija → reikalavimų atskaitos linija. Įvestis: verslo tikslai, teisinis kontekstas, suinteresuotų šalių poreikiai – natūrali kalba, dažnai prieštaringa ir nepilna. Transformacija: neapibrėžtumo mažinimas per išgavimą, tikslų išgryninimą, formulavimą, validaciją ir suderinimą. Išvestis: patvirtinta funkcinių ir nefunkcinių reikalavimų atskaitos linija (angl. baseline) su apimties ribomis.

Antras lygis. Reikalavimai → funkcinė specifikacija. Transformacija: dekompozicija į konkrečią sistemos elgseną. Vieninga reikalavimų informacija išskaidoma į keturias viena kitą papildančias projekcijas: struktūrą (dalykinės srities modelis), dinamiką (panaudos atvejai), apribojimus (taisyklės) ir pateikimą (sąsaja). Lygis generuoja naują informaciją – alternatyvius ir klaidų scenarijus, kraštinius atvejus, kurių reikalavimuose nebuvo ir negalėjo būti, nes reikalavimai kalba apie sėkmės tikslą, o specifikacija privalo aprėpti ir nesėkmes. Kartu atsiranda nauja rizikos rūšis – projekcijų tarpusavio nesuderinamumas, todėl lygis baigiasi kryžminės konsistencijos patikra.

Trečias lygis. Specifikacija → architektūra. Transformacija: sisteminiai, visus komponentus kertantys sprendimai – struktūra, technologijos, duomenų pasiskirstymas, saugumo modelis. Šio lygio ritmas skiriasi nuo pirmųjų dviejų: ne „surinkti ir formalizuoti“, o „generuoti alternatyvas ir rinktis“, nes architektūra kuria informaciją, kurios pasaulyje dar nėra – sprendimus. Klaidų tipas: brangiai grįžtamas ir sisteminis.

Ketvirtas lygis. Architektūra → detalusis projektas. Transformacija: elgsenos atvaizdavimas į konkrečius komponentus – sąsajų kontraktai, duomenų bazės schema, modulių vidus, darbo skaidymas į įgyvendinimo vienetus su priėmimo kriterijais. Klaidų tipas: lokalus ir palyginti pigiai taisomas. Trečias ir ketvirtas lygiai atskirti būtent dėl skirtingos klaidos kainos ir grįžtamumo: sprendimai juose priimami skirtingu mastu, skirtingų kompetencijų ir su skirtinga klydimo kaina.

Penktas lygis. Projektas → kodas. Transformacija: įgyvendinimas – formalizavimas iki galo, kai interpretacijos laisvės nebelieka, nes kodą vykdo mašina. Išvestis: kodas, automatiniai testai, diegiamas artefaktas.

Šeštas lygis. Kodas → veikianti sistema. Transformacija: komponavimas, konfigūracija, aplinkos, duomenų migracijos, lokalizacijos turinys, diegimo strategija. Klaidų tipas savitas: „kodas teisingas, bet sistema neveikia“ – konfigūracijos, apkrovos ir aplinkų skirtumų klaidos, kurių ankstesniuose lygiuose pamatyti neįmanoma. Šio lygio būtinybė kyla iš pačios reikalavimų apibrėžties: reikalavimai aprašo tikslus, kuriuos turi pasiekti veikianti programinė įranga, o kodo tekstas pats savaime jokio tikslo nepasiekia. Tikrasis grandinės galutinis taškas yra elgsena produkcinėje aplinkoje ir įvykęs pokytis verslo procese.

3.3. Modelio ribos

Du potencialūs lygiai į modelį sąmoningai neįtraukti.

Nulinis lygis (verslo strategija → intencija) egzistuoja, bet vyksta už programų inžinerijos ribų: jo transformacijas atlieka rinka ir organizacija, o ne informacijos apie kuriamą sistemą apdorojimas.

Verifikacija nėra grandinės grandis, nors jos išvestis („patikrinta sistema“) tokia atrodo. Verifikacija yra kiekvienos grandies palydovas: reikalavimai validuojami, specifikacija tikrinama dėl konsistencijos, projektas peržiūrimas, kodas testuojamas. Įtraukus ją kaip lygį, modelis klaidingai teigtų, kad tikrinama vieną kartą pabaigoje.

4. PROCESO PROJEKTAVIMO PRINCIPAI

Šiame skyriuje formuluojami principai, kuriais grindžiamas lygių skaidymas ir kurie galioja visai grandinei.

4.1. Žingsnio išskyrimo kriterijus

Atskiras žingsnis (arba lygis) pateisinamas tik tada, kai turi savitą transformaciją, savitą išvesties rūšį ir savitą klaidų tipą – tai yra, jame galima suklysti taip, kaip niekur kitur. Skaidymas be šio kriterijaus kuria biurokratiją, o ne kokybę.

4.2. Klaidų matomumo principas

Proceso projektavimas iš esmės yra klaidų matomumo projektavimas. Pavojingiausios klaidos yra tylios: praleista suinteresuota šalis, nepastebėtas tikslų konfliktas, praleistas klaidos scenarijus, patvirtinimas be perskaitymo. Visi proceso mechanizmai (registrai, radinių sąrašai, inventoriai, kryžminės patikros, priverstinis rikiavimas) tarnauja vienam uždaviniui – paversti nematomą klaidą matoma.

Svarbi šio principo išvada: iškraipymo ir praleidimo klaidos reikalauja skirtingų patikros strategijų. Iškraipymą (neteisingai aprašytą eigą) suinteresuotos šalys pastebi peržiūroje, nes turi etaloną atmintyje. Praleidimo (trūkstamo scenarijaus, trūkstamos šalies) peržiūra nepagauna, nes trūkumas nematomas – jam reikalingos sisteminės generavimo ir padengimo procedūros.

Darbe vartojami klaidų tipai apibendrinami lentelėje: kiekvienas tipas pririšamas prie žingsnio, kuriame tipiškai kyla, ir prie jį aptinkančios strategijos.

2 lentelė. Klaidų tipai, jų kilmės vietos ir aptikimo strategijos

Klaidų tipas	**Esmė**	**Tipinė kilmės vieta**	**Aptikimo strategija**
Nepilnumas	Užfiksuota ne visa šaltinio turima informacija	1.2 išgavimas	Grįžimas pas šaltinį; šaltinių padengimo patikra pagal šalių žemėlapį
Iškraipymas	Informacija užrašyta arba išversta pakeičiant prasmę	1.2 fiksavimas, 1.5-1.6 vertimas, 2.3 scenarijai	Peržiūra su šalimis; vertimo atgal metodas
Praleidimas	Trūksta elemento, kurio niekas neužfiksavo (šalis, scenarijus, veiksnys)	1.1, 2.4, 3.1	Sisteminis generavimas ir padengimo patikros; peržiūra neaptinka
Dviprasmybė ir netikras tikslumas	Forma tvarkinga, bet reikšmė daugiaprasmė arba nepagrįsta	1.5 formulavimas	Formalizavimo šablonai; prielaidų registras
Sutarimas be įsipareigojimo	Statusas suteiktas be tikro įsitraukimo	1.7 suderinimas	Įsitraukimo įrodymas vietoj parašo

Nepilnumas ir praleidimas yra giminingi (abu yra trūkumas), skiriasi atskaitos taškas: nepilnumas matuojamas prieš šaltinio turimą informaciją (šaltinis žinojo, bet neužfiksuota), praleidimas – prieš dalykinę tikrovę (niekas net nepaklausė, nes nebuvo žinoma, kad yra ko klausti). Nepilnumą galima aptikti grįžus pas šaltinį; praleidimui šaltinio nėra, todėl jį dengia tik sisteminės procedūros.

4.3. Mandato ir kompetencijos skyrimas

Dalis transformacijų reikalauja žinojimo (analizė, formulavimas, modeliavimas), dalis – įgaliojimo (konfliktų sprendimas, prioritizavimas, apimties ribos). Konflikto tarp dviejų šalių tikslų negalima išspręsti analize, nes tai ne fakto, o vertės klausimas. Kompetencijos klausimus sprendžia specialistai, mandato klausimus – įgaliojimus turintis asmuo.

4.4. Sprendimo fiksavimas su pagrindimu

Sprendimas be pagrindimo yra informacijos praradimas: po laiko neįmanoma atskirti „taip nuspręsta dėl priežasties“ nuo „taip susiklostė“. Fiksuojama: kontekstas, svarstytos alternatyvos, atmetimo priežastys, sąmoningai prisiimtos pasekmės. Ši forma (sprendimo įrašas) galioja ir architektūriniams, ir verslo lygmens sprendimams; ji atitinka architektūrinių sprendimų įrašų praktiką [10]. Šiame modelyje laikomasi nuostatos: sprendimas be svarstytų alternatyvų traktuojamas ne kaip sprendimas, o kaip įprotis, nes be alternatyvų neįmanoma atskirti sąmoningo pasirinkimo nuo inercijos.

4.5. Prielaidų disciplina

Kai formalizavimas reikalauja reikšmės, kurios niekas nežino, reikšmė ne išgalvojama, o registruojama kaip prielaida su savininku ir patvirtinimo planu. Validacijos metu nepatvirtinta prielaida perkeliama į rizikų registrą, o ne pašalinama. Taip tikslumo iliuzija atskiriama nuo žinojimo.

4.6. Autoriaus aklumo principas

Praktika rodo, kad autorius savo teksto spragų sistemingai nepastebi: skaitydamas jis mato tai, ką norėjo pasakyti, o ne tai, kas parašyta; šis efektas gerai žinomas iš korektūros praktikos, ir modelyje jis priimamas kaip darbinis principas. Todėl tam tikrų rolių porų negalima jungti viename asmenyje: formuluotojas negali pats vesti savo teiginių validacijos, pagrindinio scenarijaus autoriui reikia kitų akių alternatyvoms generuoti.

4.7. Sąsmaukos principas

Lygių sandūros yra siauros: iš pirmo lygio į antrą pereina trys artefaktai, iš antro į trečią – iš esmės du. Šios sąsmaukos yra natūralūs kokybės vartai, nes per jas informacija keičia šeimininką, o šeimininko keitimas yra didžiausias praradimo taškas. Rolių ribos turi sutapti su sąsmaukomis. Jei žemesniam lygiui prireikia aukštesnio lygio vidinių artefaktų, tai signalas, kad sąsmaukos artefaktas nepilnas.

4.8. Fraktalinė struktūra

Lygių viduje kartojasi tas pats raštas: paruošti atskaitos sistemą → surinkti arba inventorizuoti → generuoti turinį → formalizuoti → patikrinti → įtvirtinti. Tai ta pati transformacijos seka, taikoma vis siauresnei informacijos rūšiai. Išimtis – trečias lygis, kur vidurinioji dalis iš „surinkti ir formalizuoti“ virsta „generuoti alternatyvas ir rinktis“.

5. LYGIŲ VIDINĖ STRUKTŪRA

Artefaktų identifikavimo schema: A<lygis>.<žingsnis>.<eilės numeris>.

Vidinis skaidymas pateikiamas pirmiesiems trims lygiams. Ketvirto–šešto lygių skaidymas paliktas tolesniam darbui; jam galioja tie patys principai: žingsnio išskyrimo kriterijus (4.1), fraktalinis raštas (4.8) ir sąsmaukų principas (4.7).

5.1. Pirmojo lygio žingsniai (apžvalga)

Pirmasis lygis skaidomas į septynis žingsnius (detali analizė pateikiama antrojoje dalyje):

3 lentelė. Pirmojo lygio žingsniai ir jiems būdingi klaidų tipai

Žingsnis	**Esmė**	**Būdingas klaidų tipas**
1.1 Šalių ir šaltinių identifikavimas	Kas turi informacijos ir interesų	Praleista šalis, nematoma tolesnėms patikroms
1.2 Išgavimas	Tyli žinia → užrašyta žaliava	Nepilnumas, iškraipymas fiksuojant
1.3 Tikslų išgryninimas	Žaliava → tikslai, atskirti nuo sprendimų	Sprendimas užfiksuotas vietoj tikslo
1.4 Struktūrizavimas ir konfliktai	Hierarchija, konfliktų ir prielaidų registrai	Nepastebėtas konfliktas
1.5 Reikalavimų formulavimas	Tikslai → patikrinami teiginiai apie sistemą	Dviprasmybė, netikras tikslumas
1.6 Validacija	Tiesos klausimas: ar teisingai suprasta	Nepagautas iškraipymas vertime
1.7 Suderinimas ir atskaitos linija	Valios klausimas: ar įsipareigojama	Sutarimas be įsipareigojimo; apimtis be aiškių ribų

Žingsnių grupės pagal informacijai suteikiamą savybę: 1.1-1.2 užtikrina informacijos egzistavimą, 1.3-1.4 – struktūrą, 1.5 – formą, 1.6-1.7 – statusą.

5.2. Antrojo lygio žingsniai

2.1. Dalykinės srities modelis ir žodynas. Įvestis: A1.7.1. Iš reikalavimų tekstų ištraukiamos sąvokos, suliejami sinonimai, kiekvienai sąvokai nustatoma, ar ji esybė, atributas, ar rolė; nubrėžiami ryšiai su kardinalumais; reikšmingoms esybėms apibrėžiami gyvavimo ciklai; įtvirtinami terminų apibrėžimai. Išvestis: A2.1.1 konceptualus modelis, A2.1.2 gyvavimo ciklai, A2.1.3 rolių modelis, A2.1.4 žodynas. Klaidų tipas: neteisinga abstrakcija – brangiausia lygio klaida, nes modeliu remiasi visi tolesni artefaktai.

2.2. Sąveikos žemėlapis. Įvestis: A1.7.1, A2.1.3. Kiekvienam reikalavimui nustatoma, kokie aktoriai su kokiais baigtiniais tikslais sąveikauja su sistema; sąveikos vienetai apibrėžiami pora „aktorius + tikslas“ su ribomis ir inicijuojančiais įvykiais. Išvestis: A2.2.1 panaudos atvejų inventorius su atsekamumu į reikalavimus. Klaidų tipas: aprėpties spraga – pamirštas sąveikos vienetas reiškia pamirštą funkcionalumą; tolesni žingsniai detalizuoja tik tai, kas inventoriuje yra.

2.3. Pagrindinių scenarijų išrašymas. Įvestis: A2.2.1, A2.1.1, A2.1.4. Kiekvienam vienetui surašoma sėkmės eiga žingsnis po žingsnio, kaitaliojant vartotojo ir sistemos perspektyvas, vartojant tik žodyno terminus. Išvestis: A2.3.1 pagrindinių scenarijų rinkinys. Klaidų tipas: iškraipyta eiga – palyginti nepavojinga, nes matoma peržiūroje.

2.4. Alternatyvų ir klaidų scenarijų generavimas. Įvestis: A2.3.1. Kiekvienam scenarijaus žingsniui sistemingai keliamas klausimas „kas, jeigu ne“: netinkami duomenys, nutraukimas, kitos šalies neatsakymas, lygiagretus pakeitimas. Tai našiausias naujos informacijos generavimo taškas visame lygyje. Išvestis: A2.4.1 alternatyvių ir klaidų scenarijų rinkinys, susietas su konkrečiais pagrindinio scenarijaus žingsniais. Klaidų tipas: praleidimas – priešingai nei 2.3, nematomas, nes suinteresuotos

šalys nesėkmės scenarijų atmintyje neturi. Skirtingas klaidų matomumas (iškraipymas prieš praleidimą) yra priežastis, dėl kurios 2.3 ir 2.4 yra du žingsniai, o ne vienas.

2.5. Taisyklių išgryninimas. Įvestis: A1.7.1, A2.1.1, A2.1.2, A2.1.3, A2.3.1, A2.4.1. Kandidatai renkami skenuojant šaltinius dėl sąlyginės kalbos („privalo“, „negali“, „tik jei“), klasifikuojami, inkaruojami ir formalizuojami iki patikrinamos formos su apibrėžtu elgesiu pažeidimo atveju. Išvestis: A2.5.1 taisyklių katalogas. Klaidų tipas: taisyklė be inkaro (galiojanti „apskritai“, todėl niekur) arba tarpusavyje prieštaraujančios taisyklės.

Taisyklė apibrėžiama kaip invariantas – sąlyga, privalanti galioti visada, nepriklausomai nuo kelio, kuriuo vartotojas atėjo. Scenarijus aprašo eigą, taisyklė – sąlygą, galiojančią daugelyje eigų. Esminis apribojimas: taisyklė gali apriboti tik įvesties artefaktuose jau egzistuojančius objektus. Iš šio apribojimo išvedami lygiai keturi taisyklių tipai – po vieną kiekvienai iki 2.5 žingsnio egzistuojančiai objektų rūšiai:

4 lentelė. Taisyklių tipai ir jų inkarai

Tipas	**Apriboja**	**Inkaras**
Duomenų apribojimas	Atributo reikšmių aibę	A2.1.1 atributai
Duomenų konsistencija	Predikatą tarp atributų ar esybių	A2.1.1 ryšiai
Būsenų mašinos taisyklė	Leidžiamas būsenų perėjas	A2.1.2 perėjos
Prieigos taisyklė	Trejetą rolė-veiksmas-esybė	A2.1.3 rolės, A2.3.1/A2.4.1 veiksmai

Taisyklių tipologija nėra konvencija – ji atkartoja ankstesnių išvesčių ontologiją. Iš to paties išplaukia ir tipologijos riba: taisyklės, kuriai reikia visiškai naujo objektų tipo, tipologija aptikti negali, nes tokia taisyklė nepatenka į jokį tipą. Šiuo atveju tipologija veikia ne kaip aptikimo, o kaip signalizavimo mechanizmas: jei formuluojant taisyklę prireikia objekto, kurio įvestyse nėra, tai signalas apie spragą 2.1 žingsnyje – taisoma pirmiausia modelio išvestis, o tik paskui katalogas.

2.6. Sąsajos specifikacija. Įvestis: A2.1.1, A2.1.4, A2.3.1, A2.4.1, A2.5.1. Abstrakti elgsena atvaizduojama į ekranus, laukus, navigaciją ir būsenas (tuščia, užpildyta, klaida, laukimas); kiekvienas laukas siejamas su modelio atributu, kiekviena validacija – su taisykle; sprendžiami masiškumo aspektai (puslapiavimas, paieška, filtrai, lokalizuojami tekstai). Išvestis: A2.6.1 ekranų specifikacijos, A2.6.2 navigacijos žemėlapis. Klaidų tipas: elementas be atitikmens modelyje arba scenarijaus žingsnis be vietos sąsajoje.

2.7. Kryžminė konsistencija ir konsolidavimas. Įvestis: visi antrojo lygio artefaktai, A1.7.1, A1.7.3. Tikrinama visomis kryptimis: kiekvienas scenarijaus žingsnis turi ekraną, kiekviena taisyklė – suveikimo vietą, kiekviena būsena pasiekiama scenarijumi, kiekvienas reikalavimas padengtas bent vienu panaudos atveju ir nėra panaudos atvejų be reikalavimo (tikrinant prieš apimties ribas). Išvestis: A2.7.1 suderinta funkcinė specifikacija, A2.7.2 atsekamumo matrica. Klaidų tipas: lokaliai teisingos, bet tarpusavyje nesuderintos projekcijos – klaidos rūšis, atsirandanti tik šiame lygyje, nes pirmajame informacija dar gyveno vienoje projekcijoje.

5.3. Trečiojo lygio žingsniai

3.1. Architektūrinių veiksnių išgryninimas. Įvestis: A1.7.2, A2.7.1. Atrenkama, kas architektūriškai reikšminga: apkrovos ir augimo scenarijai, prieinamumas, saugumas, integracijos, organizaciniai apribojimai. Kokybės atributai verčiami matuojamais scenarijais. Išvestis: A3.1.1 prioritizuotas veiksnių sąrašas. Klaidų tipas: praleistas ar neteisingai prioritizuotas veiksnys – architektūra optimizuojama ne tam uždaviniui.

3.2. Variantų generavimas ir kompromisų analizė. Įvestis: A3.1.1. Kiekvienam esminiam klausimui generuojami bent du realūs variantai ir vertinami prieš veiksnius; rizikingiausioms hipotezėms atliekami prototipai. Išvestis: A3.2.1 variantų ir kompromisų analizė, A3.2.2 prototipų rezultatai. A3.2.2 yra pirmasis empirinis artefaktas grandinėje: iki šiol visa informacija buvo išgauta iš žmonių

arba išvesta samprotavimu, čia ji pirmą kartą gaunama iš eksperimento. Klaidų tipas: tunelinis matymas – vertinamas vienintelis iš anksto pasirinktas variantas.

3.3. Sprendimų priėmimas ir fiksavimas. Įvestis: A3.2.1, A3.2.2. Pasirenkama ir fiksuojama su pilnu pagrindimu pagal 4.4 principą. Išvestis: A3.3.1 sprendimų įrašų rinkinys. Klaidų tipas: sprendimas be pagrindimo – jo neįmanoma nei peržiūrėti, nei saugiai pakeisti.

3.4. Sistemos struktūros apibrėžimas. Įvestis: A3.3.1, A2.7.1. Sistema skaidoma į komponentus su atsakomybėmis ir ribomis; specifikacija atvaizduojama ant struktūros. Išvestis: A3.4.1 komponentų modelis, A3.4.2 atsakomybių žemėlapis (atsekamumo artefaktas: specifikacijos elementai → komponentai). Klaidų tipas: neteisingai nubrėžta komponento riba – pasireiškia lėtai, kaip nuolatinė trintis kiekviename pakeitime.

3.5. Skerspjūviniai mechanizmai. Įvestis: A3.1.1, A3.4.1, A3.3.1. Vienkartiniai standartiniai atsakymai į klausimus, kylančius kiekviename komponente: autentikacija ir autorizacija, klaidų apdorojimas, auditas, lokalizacija ir laiko juostos, lygiagretumo valdymas, versijavimas. Išvestis: A3.5.1 mechanizmų specifikacijos. Klaidų tipas: mechanizmo nebuvimas – kiekvienas komponentas tą pačią problemą sprendžia savaip, ir sistema tampa nevienalytė ten, kur privalo būti vienalytė.

3.6. Architektūros įvertinimas. Įvestis: A3.1.1, A3.4.1, A3.4.2, A3.5.1. Architektūra tikrinama prieš veiksnių scenarijus: piko apkrova, regiono kritimas, saugumo padengimas; atliekami skaičiavimai ir matavimai. Išvestis: A3.6.1 įvertinimo ataskaita, A3.6.2 rizikų registras. A3.6.2 yra pirmasis artefaktas, gyvenantis ilgiau už savo lygį: jis pildomas per ketvirtą, penktą ir šeštą lygius. Klaidų tipas: „popierinė“ architektūra – viduje konsistentiška, bet netikrinta prieš realius scenarijus; praleidus šį žingsnį, patikra vis tiek įvyks, tik produkcinėje aplinkoje.

6. ARTEFAKTŲ REGISTRAS

5 lentelė. Artefaktų registras

ID	Artefaktas	Gamina	Naudoja
A1.1.1	Šalių ir šaltinių žemėlapis	1.1	1.2, 1.6, 1.7
A1.2.1	Žaliava	1.2	1.3
A1.3.1	Išgrynintų tikslų sąrašas	1.3	1.4
A1.4.1	Tikslų hierarchija	1.4	1.5
A1.4.2	Konfliktų registras	1.4	1.7
A1.4.3	Prielaidų registras	1.4	1.5, 1.6, 1.7
A1.5.1	Funkcinių reikalavimų juodraštis	1.5	1.6
A1.5.2	Nefunkcinių reikalavimų juodraštis	1.5	1.6
A1.6.1	Validuotas reikalavimų rinkinys	1.6	1.7
A1.6.2	Validacijos radinių sąrašas	1.6	1.6 (kartojimo kriterijus)
A1.7.1	Funkcinių reikalavimų atskaitos linija	1.7	2.1, 2.2, 2.5, 2.7
A1.7.2	Nefunkcinių reikalavimų atskaitos linija	1.7	3.1
A1.7.3	Apimties ribos (dabar/vėliau/niekada)	1.7	2.7, pakeitimų valdymas
A2.1.1	Konceptualus modelis	2.1	2.3, 2.5, 2.6, 2.7
A2.1.2	Gyvavimo ciklai	2.1	2.5, 2.7
A2.1.3	Rolių modelis	2.1	2.2, 2.5, 2.7
A2.1.4	Žodynas	2.1	2.3, 2.6, visi lygiai
A2.2.1	Panaudos atvejų inventorius	2.2	2.3, 2.7
A2.3.1	Pagrindinių scenarijų rinkinys	2.3	2.4, 2.5, 2.6, 2.7
A2.4.1	Alternatyvių ir klaidų scenarijų rinkinys	2.4	2.5, 2.6, 2.7
A2.5.1	Taisyklių katalogas	2.5	2.6, 2.7
A2.6.1	Ekranų specifikacijos	2.6	2.7
A2.6.2	Navigacijos žemėlapis	2.6	2.7
A2.7.1	Suderinta funkcinė specifikacija	2.7	3.1, 3.4
A2.7.2	Atsekamumo matrica	2.7	pakeitimų valdymas, visi lygiai
A3.1.1	Architektūrinių veiksnių sąrašas	3.1	3.2, 3.5, 3.6
A3.2.1	Variantų ir kompromisų analizė	3.2	3.3
A3.2.2	Prototipų rezultatai	3.2	3.3
A3.3.1	Sprendimų įrašų rinkinys	3.3	3.4, 3.5, 4 lygis
A3.4.1	Komponentų modelis	3.4	3.5, 3.6, 4 lygis
A3.4.2	Atsakomybių žemėlapis	3.4	3.6, 4 lygis
A3.5.1	Skerspjūvinių mechanizmų specifikacijos	3.5	3.6, 4 lygis
A3.6.1	Įvertinimo ataskaita	3.6	4 lygis
A3.6.2	Rizikų registras	3.6	4, 5, 6 lygiai (pildomas)

Registro analizė atskleidžia tris artefaktų savybes, kurių nematyti tekstiniame aprašyme:

1. Gyvavimo trukmė. A1.2.1 žaliava sunaudojama vieno žingsnio; A1.7.1 atskaitos linija maitina penkis žingsnius dviejuose lygiuose. Ilgiausiai gyvenančių artefaktų kokybė svarbiausia.
2. Sertifikuojantys artefaktai (A2.7.1) ne kuria naują informaciją, o suteikia rinkiniui suderinamumo statusą; jų priklausomybių semantika skiriasi nuo generuojančių artefaktų.

3. Atsekamumo artefaktai (A2.7.2, A3.4.2) yra atvaizdžiai tarp dviejų pasaulių ir pagrindinis pakeitimų poveikio analizės įrankis.

7. ROLIŲ MODELIS

Rolė apibrėžiama kaip kompetencija transformuoti vieną artefaktų rūšį į kitą – vertėjas tarp dviejų gretimų kalbų. Rolės išvedamos iš artefaktų registro, o ne postuluojamos.

7.1. Transformuojančios rolės

6 lentelė. Transformuojančios rolės

Rolė	**Lygis**	**Kalbų pora**	**Esminė kompetencija**
Veiklos analitikas (reikalavimų inžinierius)	1	Šalių pasaulis ↔ reikalavimai	Reikalavimų išgavimas, tikslų atskyrimas nuo sprendimų
Veiklos analitikas (reikalavimų inžinierius, sistemos analitikas)	2	Reikalavimai ↔ elgsenos modelis	Dekompozicija, formalizavimas, sisteminis klaidų scenarijų generavimas
Sąsajos (UX) projektuotojas (veiklos analitikas, sistemų analitikas, reikalavimų inžinierius)	2.6	Elgsenos modelis ↔ vartotojo patirtis	Sąsaja, lokalizacija, prieinamumas
Architektas (prireikus – saugumo architektas)	3	Elgsena ir kokybės atributai ↔ sprendimo struktūra	Alternatyvų generavimas, kompromisai, negrįžtami sprendimai
Vyresnysis inžinierius (techninis vadovas)	4	Architektūra ↔ kontraktai ir darbo vienetai	Struktūros vertimas į įgyvendinimo vienetus
Programuotojas	5	Projektas ↔ kodas	Galutinis formalizavimas
Diegimo ir eksploatavimo (DevOps) inžinierius	6	Kodas ↔ veikianti sistema	Konfigūracija, migracijos, diegimo strategija

Dvi pastabos dėl šio modelio. Pirma, rolių taksonomija sąmoningai sutampa su tradicine: naujumo pretenzija yra ne naujos rolės, o išvedimo būdas – rolės čia ne postuluojamos pagal profesijų sąrašą, o išplaukia iš artefaktų registro sąsmaukų. Antra, realiose organizacijose rolės kerta lygius: UX projektuotojas dalyvauja ir pirmojo lygio validacijoje, saugumo perspektyva prasideda nefunkciniuose reikalavimuose, o ne trečiajame lygyje. Lentelė rodo kiekvienos rolės svorio centrą, ne veiklos ribas.

7.2. Mandato ir šaltinio rolės

Produkto savininkas – mandato rolė: sprendžia ten, kur reikia įgaliojimo, o ne žinojimo (1.7 konfliktai, prioritetai, apimties ribos, vėlesni kompromisai). Analitikas gali konfliktą suformuluoti, bet ne išspręsti: sprendimas yra atsakomybės, ne žinojimo aktas.

Dalykinės srities ekspertas – šaltinio rolė su autoritetu: informacijos šaltinis išgavime, validatorius validacijoje, recenzentas antrojo lygio peržiūrose.

7.3. Skerspjūvinė rolė

Testavimo inžinierius sąmoningai nepriskirtas jokiam lygiui, nes verifikacija yra kiekvienos grandies palydovas (3.3 poskyris). Jo darbas prasideda antrajame lygyje: iš taisyklių katalogo ir scenarijų testai išvedami dar neegzistuojant kodui; adversarinis mąstymas yra vertingiausias indėlis į 2.4 žingsnį. Šios rolės nelokalizuojamumas yra sąmoninga 3.3 poskyrio sprendimo pasekmė: verifikaciją laikant palydovu, o ne lygiu, verifikacijos kompetencija taip pat tampa skerspjūvinė. Tai kompromisas: modelis išlošia tikslumą (tikrinama visur), bet praranda galimybę šią rolę priskirti vienai sąsmaukai.

Projektų vadovas valdo ne informacijos transformaciją, o jos vykdymą laike, todėl į šį modelį nepatenka.

7.4. Struktūrinės išvados

1. Rolių ribos sutampa su sąsmaukomis: perdavimai tarp rolių vyksta per konsoliduotus artefaktus (A1.7.x, A2.7.1). Tai ne konvencija, o pasekmė: sąsmaukos yra kokybės vartai būtent todėl, kad per jas informacija keičia šeimininką.
2. Pagal autoriaus aklumo principą (4.6) tam tikros rolių poros nejungtinos viename asmenyje: formuluotojas ir validacijos vedėjas (1.5/1.6), pagrindinio scenarijaus autorius ir alternatyvų generuotojas (2.3/2.4).
3. Mažoje komandoje vienas asmuo neišvengiamai atlieka kelias roles; tokiu atveju būtina eksplicitiškai skirti, kuri rolė atliekama konkrečiu momentu, nes kiekviena rolė ieško skirtingo klaidų tipo.

8. PROCESO TOPOLOGIJA

8.1. Iteratyvumas

Žingsniai nėra griežtai nuoseklūs: formulavimas atskleidžia spragas, grąžinančias į išgavimą; taisyklės formulavimas atskleidžia trūkstamą modelio elementą. Informacijos srauto kryptis išlieka pastovi, bet judėjimas per žingsnius iteratyvus. Grįžimas atgal yra normalus; nenormalus yra nepastebėtas grįžimas – ankstesnio artefakto keitimas neatnaujinus išvestinių.

8.2. Partijos dydžio mažinimas

Didelės partijos yra tylių klaidų terpė: didelės apimties dokumento atidus perskaitymas viršija kognityvinę talpą, nepriklausomai nuo disciplinos. Partija mažinama dviem lygmenimis: validacijos partija (tvirtinama dalimis, rizikingiausios pirmiausia) ir grandinės partija (mažas rinkinys pervaromas per visus šešis lygius greitai). Iteracinis vystymas šiuo požiūriu yra būdas apriboti nevaliduoto įsipareigojimo žalą iteracijos dydžiu.

8.3. Minimalus gyvybingas produktas ir grįžtamojo ryšio kilpa

Minimalus gyvybingas produktas (MVP) yra prioritizavimas su dviem papildomais apribojimais:

1. Rišlumas. MVP yra ne prioritetų sąrašo viršūnė, o rišli atkarpa: mažiausias reikalavimų poaibis, kartu sudarantis pilną vertės kilpą nuo pradžios iki galo. Kadangi priklausomybės matomos tik antrojo lygio artefaktuose, MVP riba nustatoma iteruojant tarp 1.7 žingsnio ir antrojo lygio.
2. Mokymosi kriterijus. Atranka vykdoma ne pagal klausimą „kas svarbiausia“, o pagal klausimą „ko mažiausiai reikia, kad sužinotume, ar mūsų hipotezė veikia“ [16]. Kadangi kiekvienas reikalavimas yra priežastinė hipotezė (žr. 10.2 poskyrį), MVP yra būdas rizikingiausią hipotezę pervaryti per visą grandinę iki šeštojo lygio matavimo mažiausia kaina.

MVP pakeičia grandinės topologiją iš linijos į kilpą: šeštojo lygio rodiklių duomenys grįžta į 1.3 žingsnį kaip nauja, šįkart empirinė žaliava; hipotezės koreguojamos; kita atkarpa eina per grandinę jau patikslinta. Esminė sąlyga: MVP be šeštojo lygio matavimo nėra MVP, o tik mažas diegimas – skirtumas ne apimtyje, o tame, ar kas nors laukia atsakymo į suformuluotą klausimą.

8.4. Pakeitimų valdymas

Atskaitos linija užšaldo informaciją ne todėl, kad ji tobula, o todėl, kad tolesni lygiai negali dirbti ant judančio pagrindo. Atskaitos linija yra kontraktas tarp lygių. Užšaldymas nereiškia draudimo keistis: pakeitimas tampa matomu įvykiu su procedūra – prašymas, poveikio analizė per atsekamumo artefaktus (A2.7.2, A3.4.2), sprendimas tuo pačiu mandatu. Skirtumas yra ne tarp „galima keisti“ ir „negalima“, o tarp tylaus slinkimo ir apskaitomo keitimosi.

ANTROJI DALIS. VEIKLOS ANALITIKO VEIKLA

9. ROLĖS APIBRĖŽTIS

Veiklos analitikas (reikalavimų inžinierius) yra pirmojo lygio transformuojanti rolė. Jo kalbų pora: suinteresuotų šalių pasaulis ↔ reikalavimų kalba. Analitikas veda visus septynis pirmojo lygio žingsnius: 1.1-1.5 atlieka pats, 1.6-1.7 fasilituoja, nes juose sprendžia suinteresuotos šalys ir mandatą turintis asmuo.

Esminės kompetencijos: reikalavimų išgavimas (tylių žinių išsiaiškinimas), tikslų atskyrimas nuo atsineštų sprendimų, formalizavimas (kalbos siaurinimas iki patikrinamų teiginių) ir fasilitavimas (validacijos bei suderinimo vedimas nedarant įtakos turiniui).

Analitiko darbo objektas yra minkščiausia grandinės informacija: išsibarsčiusi po žmonių atmintį ir dokumentus, dažnai neišsakyta. Darbo esmė – suteikti šiai informacijai keturias savybes, po vieną per žingsnių grupę: egzistavimą (1.1-1.2), struktūrą (1.3-1.4), formą (1.5) ir statusą (1.6-1.7).

10. PAGRINDINĖS DARBO SĄVOKOS

10.1. Tikslas

Apibrėžtis. Tikslas yra verslo būsenos rodiklis, jo norima reikšmė ir laiko profilis. Sutrauktai: iš principo pamatuojamas norimas verslo būsenos skirtumas, užrašytas kaip rodiklis, reikšmė ir laikas.

Komponentų analizė:

1. Rodiklis – matuojama verslo būsenos savybė su savo skale, nuo binarinės (atitiktis: taip/ne) iki kiekybinės (vėlavimų dalis procentais). Binarinė skalė yra pilnavertė matavimo skalė: matavimo skalių teorijoje ji atitinka nominaliąją skalę su dviem reikšmėmis [13]. Verslo būsena apima keturias sritis: procesų eigą, jų rezultatus, įsipareigojimus ir gebėjimus; proceso rodiklis yra dažniausia, bet ne vienintelė atmaina.
2. Norima reikšmė – viena iš trijų formų:
   - taikinys: rodiklis lygus reikšmei arba patenka į intervalą;
   - riba: rodiklis neviršija maksimumo arba nenukrinta žemiau minimumo;
   - kryptis: rodiklį mažinti arba didinti.

   Kryptis reikalauja skalės su tvarka, todėl binariniai rodikliai jos turėti negali – dėl to atitikties tikslai visada užrašomi taikiniu. Kryptis yra leistina laikina forma 1.3 žingsnyje, tačiau ji yra skola: formulavimo žingsnis privalo ją paversti taikiniu ar riba arba užregistruoti prielaidą, nes iš krypties formos neįmanoma išvesti testo (klausimas „ar pasiekta“ neturi atsakymo). Praktikoje verslas retai siekia maksimizuoti – jis siekia pakankamo lygio, todėl riba yra natūrali galutinė forma.
3. Laiko profilis – kada rodiklis turi turėti norimą reikšmę: pasiekti iki momento (matuojama kartą), gerinti nuolat (matuojama periodiškai), išlaikyti visada (matuojama nuolat arba audituojama). Tikslų rūšys skiriasi ne matuojamumu, o laiko profiliu ir skalės grubumu.

Trys komponentai veikia kaip formalizavimo šablonas: neužpildytas laukas parodo, ko dar nežinoma, ir registruojamas prielaidų registre A1.4.3.

Tikslo atpažinimo testai:

- Klausimas „kodėl“ turi vesti į aukštesnį tikslą; jei atsakymo nėra, užfiksuotas įprotis.
- Klausimas „kaip“ turi turėti daugiau nei vieną atsakymą; jei atsakymas vienas, užfiksuotas sprendimas.

Pavyzdžiai (sąskaitų faktūrų valdymo sritis):

7 lentelė. Tikslų rūšių pavyzdžiai sąskaitų faktūrų valdymo srityje

Rūšis	**Pavyzdys**
Pasiekimo	Sąskaita išsiunčiama ne vėliau kaip kitą darbo dieną po paslaugos suteikimo
Optimizavimo	Pavėluotų apmokėjimų dalis sumažinama nuo 30 iki 10 procentų iki metų pabaigos
Palaikymo	Bet kuriuo momentu įmanoma nustatyti, kas ir kada patvirtino kiekvieną sąskaitą
Atitikties	Sąskaitų numeracija ir turinys atitinka PVM įstatymą (taikinys „taip“, profilis „visada“)
Kontrolės	Niekas negali patvirtinti savo paties sukurtos sąskaitos

Pastaba: kontrolės tikslas dar nėra taisyklė, nors formuluotė panaši. Taisykle (A2.5.1, su inkaru į roles ir veiksmus) jis taps antrajame lygyje, kai egzistuos objektai, kuriuos galima apriboti; pirmajame lygyje tai verslo intencija.

10.2. Reikalavimas

Apibrėžtis. Reikalavimas yra patikrinamas teiginys apie sistemą: vienareikšmis, atominis, atsekamas iki tikslo.

Trys esminės savybės:

1. Subjekto pakeitimas. Tikslas kalba apie verslą („vėlavimai iki 10 procentų“), reikalavimas – apie sistemą („sistema siunčia priminimą likus 3 dienoms iki termino“). Iki formulavimo žingsnio grandinės informacija kalba apie pasaulį, nuo jo – apie sistemą.
2. Hipotezės prigimtis. Reikalavimas nėra išvedamas iš tikslo – jis yra statymas dėl tikslo [15]. Iš teiginio „vėlavimai iki 10 procentų“ deduktyviai neišplaukia „siųsti priminimus“: tai priežastinė hipotezė, kad priminimai paskatins mokėti laiku. Reikalavimas gali būti tobulai įgyvendintas, o tikslas nepasiektas. Vienas tikslas paprastai skyla į kelias hipotezes. Atsekamumo ryšys fiksuoja, kurią hipotezę reikalavimas įgyvendina, kad šeštajame lygyje, rodikliui nepajudėjus, būtų matyti, kurios hipotezės žlugo ir ką keisti. Formulavimą galima apibūdinti kaip vertimą iš norų kalbos į įsipareigojimų kalbą.
3. Funkcinių ir nefunkcinių reikalavimų atskyrimas. Funkcinis reikalavimas nusako, ką sistema daro, nefunkcinis – kaip gerai. Jie laikomi atskiruose artefaktuose, nes keliauja skirtingais kanalais: funkciniai maitina antrąjį lygį (elgsenos modelį), nefunkciniai – trečiąjį (architektūros veiksnius). Sujungti jie užkimštų sąsmauką.

10.3. Sąvokų skirtis

8 lentelė. Tikslo, reikalavimo ir sprendimo skirtis

Sąvoka	**Kalba apie**	**Pavyzdys**
Tikslas	Verslą (norima būsena)	Vadovybė kas mėnesį turi pardavimų suvestinę pagal klientus
Reikalavimas	Sistemą (elgsena)	Sistema leidžia suformuoti suvestinę pagal klientą ir laikotarpį
Sprendimas	Įgyvendinimą (mechanizmas)	Skaičiuoklės eksportas

Tipinė klaida: suinteresuotos šalies atsineštas sprendimas užfiksuojamas kaip tikslas ir taip užrakinama sprendimų erdvė visiems tolesniems lygiams.

11. ŽINGSNIŲ ANALIZĖ

11.1. Šalių ir šaltinių identifikavimas (1.1)

Paskirtis: nustatyti, kas apskritai turi informacijos ir interesų, prieš pradedant informaciją rinkti.

– Įvestis: pradinė iniciatyva (išorinė grandinei).

– Transformacija: identifikuojami vaidmenys, paveiktos grupės, dokumentai, teisės aktai, esamos sistemos; kiekvienai šaliai fiksuojama, kokios informacijos ji turi, koks jos interesas ir koks vaidmuo validacijoje.

– Išvestis: A1.1.1 šalių ir šaltinių žemėlapis.

– Klaidų tipas: praleista šalis – brangiausia lygio klaida: praleistos šalies informacija tyliai neegzistuos visuose tolesniuose lygiuose, ir jokia vėlesnė patikra to neparodys, nes tikrinti galima tik užfiksuotą informaciją.

Kokybės kriterijai: žemėlapyje yra ne tik ko nors norinčios, bet ir paveiktos šalys (kurios nieko neprašė, bet kurių darbą sistema pakeis); įtraukti ne tik asmenys, bet ir dokumentiniai šaltiniai (teisės aktai, esamų sistemų elgsena); globalios apimties atveju atstovaujami skirtingi regionai, kurių praktikos gali skirtis.

11.2. Išgavimas (1.2)

Paskirtis: paversti tylią ir išsibarsčiusią žinią užrašyta.

– Įvestis: A1.1.1.

– Transformacija: interviu, seminarai, dokumentų analizė, esamos veiklos stebėjimas [7]; informacija nevertinama ir nefiltruojama – prioritetas teikiamas pilnumui.

– Išvestis: A1.2.1 žaliava (užrašai, citatos, procesų aprašai).

– Klaidų tipas: nepilnumas ir iškraipymas fiksuojant – užrašyta ne tai, kas pasakyta, arba nepaklausta to, ko pašnekovas savo iniciatyva nepasakys.

Darbo technikos:

– Klausiama apie situacijas, ne apie norus: prašymas papasakoti apie paskutinį mėnesio uždarymą suteikia daugiau informacijos negu klausimas, ko norėtųsi iš sistemos.

– Stebėjimas realioje darbo vietoje atskleidžia tai, ko interviu neduoda: žmonės nepasakoja to, kas jiems savaime suprantama.

– Citatos fiksuojamos pažodžiui; interpretacija vyksta kitame žingsnyje.

11.3. Tikslų išgryninimas (1.3)

Paskirtis: iš žaliavos ištraukti tikslus, atskiriant juos nuo atsineštų sprendimų.

– Įvestis: A1.2.1.

– Transformacija: klausimu „kodėl“ kylama aukštyn, kol teiginys nustoja būti mechanizmu ir tampa norima verslo būsena; suliejami to paties tikslo variantai iš skirtingų šalių; kiekvienas tikslas užrašomas šablonu rodiklis + reikšmė + laiko profilis; neužpildyti laukai registruojami A1.4.3.

– Išvestis: A1.3.1 išgrynintų tikslų sąrašas.

– Klaidų tipas: sprendimas, užfiksuotas vietoj tikslo.

Transformacijos pavyzdys. Žaliavoje užfiksuota: buhalterė pageidauja skaičiuoklės eksporto. Klausimas „kodėl“ atskleidžia: ataskaitos vadovybei ruošiamos skaičiuoklėje. Pakartotas klausimas „kodėl“: vadovybei kas mėnesį reikalinga pardavimų suvestinė pagal klientus. Tikslas: „Vadovybė

kas mėnesį turi pardavimų suvestinę pagal klientus“. Skaičiuoklės eksportas lieka vienu iš galimų sprendimų, neužrakinant alternatyvų (integruota ataskaita, automatinis siuntimas).

11.4. Tikslų struktūrizavimas ir konfliktų identifikavimas (1.4)

Paskirtis: sudėti tikslus į struktūrą ir padaryti konfliktus matomus.

– Įvestis: A1.3.1.

– Transformacija: hierarchija pagal „kodėl“ ryšius (kuris tikslas tarnauja kuriam), priklausomybės, šalių tikslų sandūrose fiksuojami konfliktai. Konfliktai tik identifikuojami: jų sprendimas atidedamas į 1.7 žingsnį, nes reikalauja mandato, o ne kompetencijos.

– Išvestis: A1.4.1 tikslų hierarchija, A1.4.2 konfliktų registras, A1.4.3 prielaidų registras.

– Klaidų tipas: nepastebėtas konfliktas – dvi šalys iki pat diegimo gyvena su nesuderinamais lūkesčiais.

Kokybės kriterijai: kiekvienas tikslas turi vietą hierarchijoje (tikslas be aukštesnio tikslo yra arba strateginis, arba abejotinas); konfliktai užrašyti neutraliai, įvardijant abi šalis ir abu tikslus – registras nesprendžia, kas teisus; globalios apimties atveju sistemingai tikrinamas regionų tikslų suderinamumas, nes tai dažniausia konfliktų kilmė.

11.5. Reikalavimų formulavimas (1.5)

Paskirtis: tikslus paversti patikrinamais teiginiais apie sistemą.

– Įvestis: A1.4.1, A1.4.3.

– Transformacija: vienu metu vyksta trys darbai:

  3. Subjekto pakeitimas – iš teiginių apie verslą į teiginius apie sistemos elgseną.

  3. Hipotezių formulavimas – kiekvienas reikalavimas yra priežastinis statymas, kad tam tikra sistemos elgsena pastūmės tikslo rodiklį; ryšys reikalavimas → tikslas fiksuojamas privalomai.

  3. Kalbos siaurinimas trimis operacijomis:

     – Vienareikšmiškumas. Kiekvienas neapibrėžtas žodis („laiku“, „greitai“, „didelis“) keičiamas konkrečia reikšme; nežinoma reikšmė ne išgalvojama, o registruojama prielaidose su savininku ir patvirtinimo planu. Tai skirtumas tarp tikslumo ir netikro tikslumo: reikšmė įrašyta abiem atvejais, bet prielaidos atveju pažymėta, kad nepagrįsta.

     – Atomiškumas. Vienas teiginys – vienas reikalavimas. Testas: ar teiginį galima atmesti arba atidėti nepaliečiant nieko kito; jei ne, jis skaidomas. Neatominis reikalavimas yra neadresuojamas: negali turėti vieno identifikatoriaus, statuso, prioriteto ir testo.

     – Krypčių uždarymas. Tikslo forma „mažinti X“ verčiama taikiniu arba riba; nežinoma riba registruojama prielaidose. Reikalavimo lygmenyje krypties forma neleistina, nes iš jos neįmanoma išvesti testo.

– Išvestis: A1.5.1 funkcinių reikalavimų juodraštis, A1.5.2 nefunkcinių reikalavimų juodraštis.

– Klaidų tipas: dviprasmybė ir netikras tikslumas – forma tvarkinga, bet reikšmė daugiaprasmė arba išgalvota.

Kokybės kriterijai: kiekvienas reikalavimas turi identifikatorių, atsekamumą ir yra testuojamas (testą galima parašyti neegzistuojant kodui); iš to paties tikslo kilę funkciniai ir nefunkciniai reikalavimai atskirti (pavyzdžiui, „siųsti priminimą likus 3 dienoms“ – funkcinis; „priminimas išsiunčiamas ne vėliau kaip per valandą nuo įvykio, įskaitant piko apkrovą“ – nefunkcinis); reikalavimų be tikslo nėra – tokie grąžinami į tyrimą arba atmetami.

11.6. Validacija (1.6)

Paskirtis: atsakyti į epistemologinį klausimą, ar suformuluoti teiginiai atitinka šalių intencijas.

– Įvestis: A1.5.1, A1.5.2, A1.1.1 (kas validuoja), A1.4.3 (prielaidos patvirtinimui).

– Transformacija: trys darbai:

    3. Padengimo patikra pagal A1.1.1 – kiekvienas reikalavimas validuojamas šalies, kurios tikslui tarnauja, ir šalių, kurias paliečia. Jei šalis buvo praleista 1.1 žingsnyje, klaida tyliai praeina ir pro validaciją.

    3. Peržiūra vertimo atgal metodu (žr. toliau).

    3. Prielaidų patvirtinimas – kiekviena prielaida virsta arba žinojimu, arba sąmoningai prisiimta rizika; nepatvirtinta prielaida toliau keliauja kaip neišspręsta rizika.

– Išvestis: A1.6.1 validuotas reikalavimų rinkinys, A1.6.2 radinių sąrašas.

– Klaidų tipas: nepagautas iškraipymas vertime.

Validacijos specifika. Informaciniu požiūriu tai ypatingas žingsnis: artefaktas lyginamas su etalonu, kuris niekur neužrašytas – intencijos egzistuoja tik šalių atmintyje. Todėl validacijos negalima nei automatizuoti, nei atlikti be tų pačių žmonių, nuo kurių grandinė prasidėjo.

Formalizavimo paradoksas. Formulavimo žingsnyje kalba susiaurinta, kad teiginiai taptų tikslūs, tačiau kartu jie tapo sunkiai atpažįstami šalims, kurių intencijas išreiškia. Kuo geriau atliktas formulavimas, tuo sunkesnė validacija: tikslumas ir suprantamumas šalims traukia į priešingas puses.

Vertimo atgal metodas. Iš paradokso išplaukia pagrindinė taisyklė: šalims negalima tiesiog pateikti dokumento skaityti – formalūs teiginiai verčiami atgal į jų kalbą ir, dar geriau, į pasekmes. Vietoj klausimo, ar reikalavimas teisingas, pateikiama situacija: sąskaita išsiųsta, klientas apmokėjo pusę sumos, iki termino liko trys dienos – sistema siųs priminimą apie likutį; ar taip ir turi būti. Žmonės pritaria tekstui, bet reaguoja į situacijas: dalinio apmokėjimo atvejo netinkamumą šalis pastebi tik susidūrusi su konkrečia pasekme. Validaciją galima apibūdinti kaip eksperimentą: hipotezė yra „supratome teisingai“, o šalies nustebimas – falsifikuojantis rezultatas.

Technikos (visos veikia pasekmių principu): konkretūs pavyzdžiai su realiais duomenimis, scenarijų perėjimas balsu, popieriniai maketai ir prototipai, testų atvejai, performuluoti į klausimus, prašymas perpasakoti savais žodžiais.

Apverstas sėkmės kriterijus. Validacijos sesija, neradusi nė vieno neatitikimo, laikytina ne sėkme, o įspėjimu: sveika validacija visada ką nors randa. Tuščias radinių sąrašas A1.6.2 reiškia, kad validacija neįvyko, ir ji kartojama kitu metodu. Šis kriterijus neįsitraukimą paverčia matomu.

Tipiniai nesėkmių scenarijai ir priemonės:

9 lentelė. Tipiniai validacijos nesėkmių scenarijai ir priemonės

Nesėkmės scenarijus	**Priemonė**
Mandagus pritarimas (tvirtinama, nes nepatogu prisipažinti nesupratus)	Prašymas perpasakoti; klausimas, kas nutiks konkrečioje situacijoje
Nuovargio validacija (didelė apimtis per trumpą sesiją)	Validuojama dalimis, rizikingiausios pirmiausia
Vedantieji klausimai	Sesiją veda ne autorius; klausimai formuluojami atvirai

11.7. Suderinimas ir atskaitos linija (1.7)

Paskirtis: atsakyti į valios klausimą, ar rinkiniui įsipareigojama; informacija legitimizuojama – įgyja sutarimo statusą.

– Įvestis: A1.6.1, A1.4.2, A1.4.3 (likusios rizikos), A1.1.1 (šalių padengimas).

– Išvestis: A1.7.1 funkcinių reikalavimų atskaitos linija, A1.7.2 nefunkcinių reikalavimų atskaitos linija, A1.7.3 apimties ribos.
– Rolės: produkto savininkas sprendžia (mandatas), analitikas fasilituoja ir fiksuoja.
– Klaidų tipas: formalus sutarimas be realaus įsipareigojimo arba apimtis be aiškių neigiamų ribų.

Konfliktų sprendimas. Konfliktas tarp šalių tikslų yra vertės, o ne fakto klausimas, todėl jį sprendžia mandatas. Kiekvienas sprendimas fiksuojamas sprendimo įrašo forma: kontekstas, alternatyvos, pasirinkimo priežastys, prisiimtos pasekmės. Be įrašo diskusija po laiko prasidės iš naujo; su įrašu – nuo klausimo, ar pasikeitė aplinkybės.

Prioritizavimas. Du principai svarbesni už konkretų metodą. Pirma, prioritetas yra sprendimas, ne savybė: jį priskiria mandatas, ne analizė. Antra, prioritetai privalo būti santykiniai: sąrašas, kuriame kritinių dauguma, informaciškai tuščias, nes prioritizavimo paskirtis yra atsisakymo tvarka – atsakymas į klausimą, kas aukojama pirmiausia, kai pritrūks išteklių. Veiksminga technika yra priverstinis rikiavimas: ne žymėti svarbius, o išrikiuoti visus; rikiavimas verčia lyginti, o lyginimas atskleidžia tikrąsias vertes.

Apimties ribos. A1.7.3 yra trireikšmis: dabar / vėliau / niekada. Aiškus „ne“ yra lygiavertis įsipareigojimas kaip „taip“, saugantis nuo tylių lūkesčių. Neįtrauktų tikslų sąrašas kryžminamas su šalių žemėlapiu: kiekviena šalis turi pamatyti savo tikslų likimą, įskaitant atmestus, dabar, o ne diegimo metu. Riba tarp „dabar“ ir „vėliau“ atitinka MVP sprendimą (8.3 poskyris) ir fiksuojama su pagrindimu: kuri hipotezė tikrinama pirmiausia ir kodėl.

Atskaitos linijos aktas. Užšaldymas yra kontraktas tarp lygių: pirmasis lygis įsipareigoja teiginių stabilumui, antrasis – darbu būtent iš jų. Pakeitimas po atskaitos linijos tampa matomu įvykiu su procedūra (8.4 poskyris).

12. TYLIŲ KLAIDŲ PREVENCIJA

Pavojingiausia pirmojo lygio klaida yra formaliai patvirtinta, bet atidžiai neperskaityta atskaitos linija: ji tyliai nuodija visus tolesnius lygius ir pasireiškia tik šeštajame, kai rodikliai nepajuda. Palyginimui, priešinga klaida (validuoti reikalavimai, dėl kurių neįsipareigojama) sustabdo projektą garsiai ir anksti, todėl yra pigesnė. Trys procesiniai svertai prieš tylųjį variantą:

1. Patvirtinimo turinio pakeitimas. Iš pasyvaus akto (parašo) į įsitraukimo įrodymą: atsakymus į situacinius klausimus, perpasakojimą, scenarijaus perėjimą. Radinių sąrašas A1.6.2 su apverstu sėkmės kriterijumi neįsitraukimą paverčia matomu.
2. Partijos mažinimas. Didelės apimties dokumento atidus perskaitymas viršija kognityvinę talpą nepriklausomai nuo disciplinos, todėl validuojama ir tvirtinama dalimis, rizikingiausios pirmiausia.
3. Konkretumo perkėlimas į priekį. Žmonės reaguoja į konkretų, o konkretumas grandinėje natūraliai auga lygis po lygio. Todėl kiekviena sąsmauka yra pervalidavimo vartai: ekranų maketai (A2.6.1) rodomi toms pačioms šalims, penktojo lygio demonstracijos daromos verslo atstovams, ne komandai. Radikalesnė forma – prototipas dar prieš atskaitos liniją, specialiai validacijos tikslui: informaciniu požiūriu tai išmetamo artefakto gamyba, bet tai pigiausias būdas priversti tylią nežinią pasireikšti.

Sąžiningas apribojimas: procesas gali priversti pademonstruoti supratimą, bet negali priversti nuoširdžiai įsipareigoti – tai organizacijos, ne informacijos transformacijos problema. Proceso galimybė yra žalos ribojimas: abejonės dėl įsipareigojimo registruojamos rizikų registre su savininkais, kad vėliau nebūtų galima teigti, jog niekas negalėjo žinoti.

13. ANALITIKO VEIKLA UŽ PIRMOJO LYGIO RIBŲ

Analitiko darbas nesibaigia atskaitos linijos perdavimu:

– Antrojo lygio peržiūros: tikrinama, ar specifikacija neiškraipė reikalavimų prasmės; dalyvaujama 2.7 padengimo patikroje.

– Pervalidavimo vartai: organizuojamas maketų ir demonstracijų rodymas suinteresuotoms šalims.

– Pakeitimų valdymas: pakeitimų prašymai vertinami prieš apimties ribas, vedama poveikio analizė per atsekamumo artefaktus, sprendimas teikiamas mandatui.

– Šeštojo lygio kilpa: po diegimo stebimi tikslų rodikliai; jų duomenys grįžta į tikslų išgryninimą kaip empirinė žaliava; žlugusios hipotezės identifikuojamos per atsekamumą ir grąžinamos į formulavimą.

14. KOKYBĖS VARTAI: PERDAVIMAS Į ANTRĄJĮ LYGĮ

Atskaitos linija laikoma patikima sąsmauka, jei tenkinami visi kriterijai.

Turinio pilnumas:

1. Kiekvienas tikslas užrašytas pilnu šablonu (rodiklis, reikšmė, laiko profilis) arba trūkumas registruotas prielaidose.
2. Kiekvienas reikalavimas atominis, vienareikšmis, testuojamas, su identifikatoriumi ir atsekamumu į tikslą.
3. Nėra reikalavimų be tikslo ir tikslų be šalies.
4. Funkciniai ir nefunkciniai reikalavimai atskirti.

Sprendimų pilnumas:

1. Kiekvienas konfliktas iš A1.4.2 turi sprendimą su pagrindimu.
2. Kiekviena prielaida iš A1.4.3 patvirtinta arba perkelta į rizikas su savininku.
3. Prioritetai santykiniai ir išrikiuoti.
4. Apimtis trireikšmė, riba tarp „dabar“ ir „vėliau“ pagrįsta.

Proceso pilnumas:

1. Kiekvienas reikalavimas validuotas atsakingos šalies pagal A1.1.1.
2. Radinių sąrašas A1.6.2 netuščias.
3. Kiekviena šalis matė savo tikslų likimą, įskaitant atmestus.

Jei visi kriterijai tenkinami, A1.7.1–A1.7.3 yra patikimas pagrindas, ant kurio antrasis lygis gali statyti be pakartotinių grįžimų. Jei ne, atskaitos linija yra tik dokumentas su parašais, ir skirtumas pasireikš šeštajame lygyje. Pirmojo lygio klaidos antrajame nebepataisomos, tik paveldimos, todėl šie vartai yra privalomi, o ne rekomendaciniai.

15. IŠVADOS

1. Programų kūrimą produktyvu nagrinėti kaip informacijos transformacijos grandinę, kurioje kiekvienas lygis apibrėžiamas įvestimi, transformacija, išvestimi ir savitu klaidų tipu. Grandinę sudaro šeši lygiai su dviem kalbos šuoliais: iš problemos kalbos į sprendimo (tarp specifikacijos ir architektūros) ir iš teksto į vykstantį procesą (tarp kodo ir veikiančios sistemos).
2. Grandinė nesutraukiama, nes jos paskirtis yra ne versti turimą informaciją, o gaminti trūkstamą: kiekvienas lygis priima sprendimus, kurių ankstesniame nebuvo. Priemonės, gerinančios notaciją, mažina tik akcidentinį sudėtingumą; esminis (nuspręsti, ko norima) išlieka. Ši tezė galioja ir kūrimui su dideliais kalbos modeliais: jie suspaudžia formalizavimo darbą, bet trūkstamų sprendimų gamybą perkelia į dialogą, o neužduotus klausimus paverčia tyliais spėjimais (2.6 poskyris).
3. Lygių vidus pasižymi pasikartojančia struktūra: paruošti atskaitą → surinkti → generuoti → formalizuoti → patikrinti → įtvirtinti. Žingsniai išskiriami pagal vienodą kriterijų – savitą klaidų tipą; ypač svarbi skirtis tarp iškraipymo klaidų (matomų peržiūroje) ir praleidimo klaidų (reikalaujančių sisteminių generavimo ir padengimo procedūrų).
4. Artefaktų registras su identifikatoriais ir priklausomybėmis atskleidžia struktūrines savybes, nematomas tekstiniame aprašyme: artefaktų gyvavimo trukmę, sąsmaukas tarp lygių ir skirtingą artefaktų prigimtį (generuojantys, sertifikuojantys, atsekamumo, empiriniai). Rolių modelis išvedamas iš registro: rolė yra vertėjas tarp gretimų kalbų, o rolių ribos sutampa su sąsmaukomis.
5. Veiklos analitiko darbo šerdis yra keturios operacijos: tylios žinios pavertimas užrašyta, tikslų atskyrimas nuo atsineštų sprendimų, kalbos siaurinimas iki patikrinamų teiginių ir informacijos statuso keitimas (validacija ir suderinimas). Tikslo apibrėžtis per rodiklį, reikšmę ir laiko profilį suteikia formalizavimo šabloną, o reikalavimo kaip priežastinės hipotezės samprata susieja pirmąjį lygį su šeštojo lygio matavimu į grįžtamojo ryšio kilpą.
6. Pavojingiausios proceso klaidos yra tylios, o brangiausia iš jų – formaliai patvirtinta, bet neįsisavinta atskaitos linija. Prevencijos svertai (įsitraukimo įrodymas vietoj parašo, partijos mažinimas, konkretumo perkėlimas į priekį) yra vieno principo variantai: paversti nematomą klaidą matoma.
7. Darbo statusas – konceptuali hipotezė (1.3 poskyris), tačiau dalis įrankių taikytini savarankiškai, nelaukiant viso modelio patikrinimo: tikslo šablonas (rodiklis, norima reikšmė, laiko profilis) su prielaidų registru, klaidų tipų lentelė (4.2) ir apverstas validacijos sėkmės kriterijus (11.6). Tolesnio darbo kryptys: ketvirto–šešto lygių vidinis skaidymas ir modelio prognozių retrospektyvinis patikrinimas realių projektų defektų duomenimis.

PRIEDAS A. PERĖJIMAS PER PIRMĄJĮ LYGĮ: PRIMINIMŲ REIKALAVIMAS

Priedo paskirtis – parodyti visą pirmojo lygio grandinę veikiančią ant vieno nedidelio, bet pilno atvejo. Sritis ta pati kaip pagrindinio teksto pavyzdžiuose: sąskaitų faktūrų valdymas. Pradinė iniciatyva: įmonės vadovybė nepatenkinta, kad klientai vėluoja apmokėti sąskaitas.

A.1. Šalių ir šaltinių identifikavimas (1.1)

A1.1.1 žemėlapio fragmentas:

10 lentelė. Šalių ir šaltinių žemėlapio fragmentas

Šalis / šaltinis	Turima informacija	Interesas	Vaidmuo validacijoje
Buhalterė	Faktinė apmokėjimų sekimo eiga, dabartinis rankinis procesas	Mažiau rankinio darbo	Validuoja proceso eigą
Pardavimų vadovas	Santykiai su klientais	Nepabloginti santykių su klientais	Validuoja komunikaciją klientams
Vadovybė (per produkto savininką)	Verslo prioritetai	Apyvartinių lėšų srautas	Mandatas: konfliktai, apimtis
Klientai	Kaip jų pusėje apdorojamos gaunamos sąskaitos	Nieko neprašė; paveikta šalis	Netiesiogiai (per pardavimų vadovą)
PVM įstatymas	Sąskaitų turinio ir numeracijos reikalavimai	Dokumentinis šaltinis	Atitikties etalonas
Esama apskaitos sistema	Faktinė sąskaitų ir apmokėjimų duomenų struktūra	Dokumentinis šaltinis	Elgsenos etalonas

Pastaba dėl klaidų tipo: klientai yra tipiška praleidžiama šalis – jie nieko neprašo, bet priminimai keis jų patirtį. Jų nepraleidus čia, 1.6 žingsnyje atsiras validacijos klausimas apie komunikacijos toną, kuris kitaip nebūtų kilęs.

A.2. Išgavimas (1.2)

A1.2.1 žaliavos fragmentas (citatos pažodžiui):

– Buhalterė: „Kas antrą savaitę sėdžiu ir rankomis žiūriu, kurios sąskaitos apmokėtos, o kurios ne. Tiems, kur vėluoja, skambinu arba rašau. Norėčiau, kad sistema pati siųstų priminimus.“

– Pardavimų vadovas: „Tik nepradėkit bombarduoti klientų laiškais. Didieji klientai to nemėgsta, su jais tariamės individualiai.“

– Stebėjimas darbo vietoje: buhalterė vėluojančias sąskaitas žymisi atskiroje skaičiuoklėje; dalinių apmokėjimų atvejus sprendžia kiekvieną kartą skirtingai.

– Iš esamos sistemos: apmokėjimo terminas saugomas prie sąskaitos; dalinio apmokėjimo būsena egzistuoja.

Fiksuojama viskas, įskaitant buhalterės atsineštą sprendimą („sistema pati siųstų priminimus“) – jo vertinimas vyksta kitame žingsnyje.

A.3. Tikslų išgryninimas (1.3)

Klausimas „kodėl“ buhalterės pageidavimui: kodėl priminimai? Kad klientai apmokėtų laiku. Kodėl svarbu laiku? Vadovybės atsakymas: vėluojantys apmokėjimai stabdo apyvartines lėšas. Teiginys nustojo būti mechanizmu ir tapo norima verslo būsena.

Tikslas pagal šabloną (rodiklis + reikšmė + laiko profilis): „Pavėluotų apmokėjimų dalis sumažinama nuo 30 iki 10 procentų iki metų pabaigos“ (rodiklis: pavėluotų apmokėjimų dalis; reikšmė: riba 10

procentų; profilis: pasiekti iki momento, toliau išlaikyti). Automatinis priminimų siuntimas lieka vienu iš galimų sprendimų, neužrakinant alternatyvų (išankstinės sąskaitos, mokėjimo sąlygų keitimas).

Į A1.4.3 registruojama prielaida: dabartinė 30 procentų reikšmė paimta iš buhalterės žodinio vertinimo, ją reikia patvirtinti iš apskaitos sistemos duomenų (savininkas: analitikas; planas: ataskaita iš esamos sistemos).

A.4. Struktūrizavimas ir konfliktai (1.4)

A1.4.1: tikslas įstatomas į hierarchiją po aukštesniu tikslu „pagerinti apyvartinių lėšų srautą“ (strateginis, už grandinės ribų).

A1.4.2 konflikto įrašas (neutraliai, abi šalys ir abu tikslai): buhalterės tikslas „apmokėjimai laiku, mažiau rankinio darbo“ ir pardavimų vadovo tikslas „nebloginti santykių su didžiaisiais klientais“ kertasi priminimų dažnio ir aprėpties klausimu. Registras nesprendžia, kas teisus; sprendimas atidedamas į 1.7.

A1.4.3 pildoma antra prielaida: priminimas el. paštu pasiekia asmenį, atsakingą už apmokėjimą kliento pusėje (savininkas: pardavimų vadovas; planas: pasitikrinti su penkiais didžiausiais klientais).

A.5. Reikalavimų formulavimas (1.5)

Subjekto pakeitimas ir kalbos siaurinimas. Iš tikslo formuluojama priežastinė hipotezė: priminimas prieš terminą paskatins apmokėti laiku. Neapibrėžti žodžiai uždaromi: „prieš terminą“ tampa „likus 3 dienoms“ (reikšmė pasiūlyta buhalterės pagal tipinį banko pavedimo ciklą; užregistruota kaip prielaida, kol nepatvirtinta klientų elgsenos duomenimis).

– A1.5.1 fragmentas (funkcinis): „FR-14. Sistema išsiunčia priminimą sąskaitos kontaktiniu adresu likus 3 dienoms iki apmokėjimo termino, jei sąskaita neapmokėta arba apmokėta iš dalies. Atsekamumas: tikslas G-03.“
– A1.5.2 fragmentas (nefunkcinis, atskirtas nuo funkcinio): „NFR-06. Priminimas išsiunčiamas ne vėliau kaip per valandą nuo suplanuoto momento, įskaitant piko apkrovą. Atsekamumas: G-03.“

Atomiškumo testas: FR-14 galima atmesti ar atidėti nepaliečiant kitų reikalavimų; teiginys „sistema siunčia priminimus ir rodo vėluojančių sąskaitų ataskaitą“ būtų skaidomas į du.

A.6. Validacija (1.6)

Vertimo atgal metodas. Buhalterei pateikiama situacija: „Sąskaita išsiųsta, klientas apmokėjo pusę sumos, iki termino liko trys dienos – sistema siųs priminimą apie likutį. Ar taip ir turi būti?“ Buhalterė nustemba: dalinis apmokėjimas dažniausiai reiškia suderintą mokėjimo grafiką, tokiu atveju priminimas nepageidautinas. Tekstui ji buvo pritarusi; į situaciją sureagavo.

A1.6.2 radinių sąrašo fragmentas: „R-01. FR-14 netikslus dalinio apmokėjimo atveju: jei dalinis apmokėjimas turi suderintą grafiką, priminimas nesiunčiamas. Reikalavimas grąžinamas į 1.5 tikslinti.“ Sąrašas netuščias, taigi apversto sėkmės kriterijaus požiūriu validacija įvyko.

Prielaidų patvirtinimas: 30 procentų reikšmė patvirtinta apskaitos duomenimis (prielaida virsta žinojimu); prielaida apie pasiekiamą adresatą kliento pusėje nepatvirtinta iki galo ir perkeliama į rizikas su savininku.

A.7. Suderinimas ir atskaitos linija (1.7)

Konflikto sprendimas mandatu. Produkto savininkas, dalyvaujant abiem šalims, nusprendžia: siunčiamas vienas priminimas iki termino; didiesiems klientams, pažymėtiems kaip „individualiai valdomi“, priminimai nesiunčiami. Sprendimo įrašas: kontekstas (buhalterės ir pardavimų tikslų konfliktas), alternatyvos (priminimai visiems; jokių priminimų; segmentuota schema), pasirinkimo priežastys, prisiimta pasekmė (dalis vėlavimų liks nepadengta priminimais).

Apimties ribos A1.7.3: dabar – priminimas iki termino ir dalinio apmokėjimo išimtis; vėliau – eskalavimo priminimai po termino; niekada – automatiniai skambučiai klientams. Kiekviena šalis mato savo tikslų likimą: buhalterė mato, kad eskalavimas atidėtas, ir tai užfiksuota, o ne numanoma.

Atskaitos linija A1.7.1 su patikslintu FR-14 patvirtinama; riba tarp „dabar“ ir „vėliau“ pagrįsta MVP logika: pirmiausia tikrinama hipotezė, kad priminimas iki termino pajudins rodiklį. Šeštajame lygyje pavėluotų apmokėjimų dalis bus matuojama, ir, rodikliui nepajudėjus, atsekamumas parodys, kuri hipotezė žlugo.

Apibendrinimas. Kiekvienas žingsnis pagamino informaciją, kurios įvestyje nebuvo: žemėlapis (kas apskritai žino), žaliava (kas pasakyta), tikslai (kodėl), hierarchija ir konfliktai (kaip tikslai sąveikauja), reikalavimai (ką sistema darys), radiniai (kur supratimas skyrėsi), atskaitos linija (kam įsipareigojama). Praleidus bet kurį žingsnį, atitinkamą informaciją būtų pagaminęs kas nors vėliau – dažniausiai programuotojas, tyliai, kraštinio atvejo pavidalu.

LITERATŪRA

1. Brooks, F. P. No Silver Bullet: Essence and Accidents of Software Engineering. IEEE Computer, 20(4), 1987.
2. Naur, P. Programming as Theory Building. Microprocessing and Microprogramming, 15(5), 1985.
3. Wiegers, K., Beatty, J. Software Requirements. 3rd ed. Microsoft Press, 2013.
4. Sommerville, I. Software Engineering. 10th ed. Pearson, 2016.
5. Robertson, S., Robertson, J. Mastering the Requirements Process: Getting Requirements Right. 3rd ed. Addison-Wesley, 2012.
6. ISO/IEC/IEEE 29148:2018. Systems and software engineering – Life cycle processes – Requirements engineering.
7. Gause, D. C., Weinberg, G. M. Exploring Requirements: Quality Before Design. Dorset House, 1989.
8. Gotel, O., Finkelstein, A. An Analysis of the Requirements Traceability Problem. Proceedings of the First International Conference on Requirements Engineering, 1994.
9. IEEE Computer Society. SWEBOK Guide: Guide to the Software Engineering Body of Knowledge. v4.0, 2024 (konfigūracijų valdymo ir atskaitos linijų skyriai).
10. Nygard, M. Documenting Architecture Decisions. 2011. Prieiga: cognitect.com/blog/2011/11/15/documenting-architecture-decisions.
11. Evans, E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, 2003.
12. Cockburn, A. Writing Effective Use Cases. Addison-Wesley, 2001.
13. Stevens, S. S. On the Theory of Scales of Measurement. Science, 103(2684), 1946.
14. Shannon, C. E. A Mathematical Theory of Communication. Bell System Technical Journal, 27, 1948.
15. Zave, P., Jackson, M. Four Dark Corners of Requirements Engineering. ACM Transactions on Software Engineering and Methodology, 6(1), 1997.
16. Ries, E. The Lean Startup. Crown Business, 2011.
17. van Lamsweerde, A. Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley, 2009.