VILNIAUS UNIVERSITETAS
INFORMATIKOS INSTITUTAS
PROGRAMŲ SISTEMŲ KATEDRA

Pakartotinis kodo panaudojimas pirminio kriptovaliutų platinimo išmaniuosiuose kontraktuose

Code Reuse in Initial Coin Offering Smart Contracts

Bakalauro baigiamasis darbas

Atliko:	Agnė Mačiukaitė	(parašas)
Darbo vadovas:	lekt. Gediminas Rimša	(parašas)
Darbo recenzentas:	partn. doc. Vaidas Jusevičius	(parašas)

Vilnius – 2019

Santrauka

Per neilgą laikotarpį Ethereum blokų grandinėje buvo sukurta daug pirminio kriptovaliutų platinimo išmaniųjų kontraktų, kurių kodas kartojasi. Kodo pakartojamumo problemą bandoma spręsti bibliotekomis. Šiame darbe kodo pakartojamumo problema tiriama pasitelkiant savybių modeliavimą. Pirmiausia interneto skaitytuvu buvo surinkti 161 pirminiam kriptovaliutų platinimui skirti išmanieji kontraktai, patalpinti Ethereum blokų grandinėje, iš kurių 40 buvo panaudoti sudarant savybių modelį. Tuomet, darbe sudaromi savybių modeliai atspindintys populiariausias Ethereum bibliotekas OpenZeppelin ir TokenMarket. Atlikus savybių modelių palyginimą tarp blokų grandinėje esančių išmaniųjų kontraktų ir bibliotekų, nustatyta, kad bibliotekos nepilnai padengia reikalingą savybių aibę. OpenZeppelin bibliotekai trūksta 9 savybių bei turi vieną perteklinę, TokenMarket bibliotekai trūksta trijų savybių. Tam, kad būtų padidinamas kodo pakartojamumas, nuspręsta pasiūlyti pakeitimus TokenMarket bibliotekai.

Raktiniai žodžiai: pirminis kriptovaliutų platinimas, blokų grandinė, išmanieji kontraktai, savybių modeliavimas, savybių pernaudojimo metodas

Summary

A lot of initial coin offering smart contracts were created in a short period. However, the code of those smart contracts often repeats across multiple smart contracts. Some Ethereum smart contract libraries are trying to solve this code reusability problem. In this thesis, the problem of repetitive code is being looked at from the perspective of feature modelling. Firstly, using a web crawler, 161 initial coin offering smart contracts were collected and 40 of them have been used composing a feature model. Then, feature models were composed for the most popular Ethereum libraries: OpenZeppelin and TokenMarket. After performing the comparison between the smart contracts from Ethereum blockchain and libraries, it was found that the Ethereum libraries do not exhibit all the required features. Namely, OpenZeppelin library is missing 9 features and has one redundant feature. Whereas, TokenMarket library is missing 3 features. As a contribution to code reuse improvement, changes to TokenMarket library were suggested.

Keywords: initial coin offering, blockchain, smart contracts, feature-oriented domain analysis, feature-oriented reuse method

TURINYS

Įvadas

Kodo pernaudojimas suteikia galimybę naudoti programas keliuose projektuose. Tai yra svarbi strategija kodui, norint padidinti sistemos kūrimo efektyvumą ir kokybę. Taikant pernaudojamumą programuotojai naudojasi jau įgyvendintu kodu, kurį keičia taip, kad jis atitiktų naujo projekto reikalavimus [RR03]. Viena iš būtinų sąlygų, kuriant pernaudojamą kodą yra supratimas skirtingų kontekstų, kuriuose pernaudojamas kodas galėtų būti naudojamas ir kaip būtų valdomas jo pernaudojamumas. Tai padeda programuotojams nuspręsti ar kodas atitinka reikalavimus ir ar gali būti kuriamas, jei taip, tai ką reikia parametrizuoti ir kaip struktūrizuoti kodą, kad vėliau būtų galima jį pritaikyti skirtingiems kontekstams [KCH+90].

Pirminio kriptovaliutos platinimo (angl. initial coin offering, toliau ICO) metu įmonė parduoda specializuotus kripto-žetonus žadėdami, kad žetonai veiks kaip mainų priemonė atsiskaitant už paslaugas įmonės platformoje. Žetonų pardavimas kuria kapitalą pradiniam įmonės platformos kūrimui, nors nėra įsipareigojimo dėl būsimos paslaugos kainos (žetonais ar kitaip) [CG18]. Tarp 2014 m. sausio ir 2018 m. birželio, ICO surinko daugiau nei 18 milijardų dolerių. Lėšų surinkimas ICO būdu pritraukė verslininkų, investuotojų ir reguliavimo įstaigų dėmesį. Dauguma ICO kuria išmaniuosius kontraktus (automatizuotą programinę įrangą) ir juos talpina Ethereum blokų grandinėje [HNY18].

Neseniai iškilęs ICO populiarumas yra paveiktas Bitcoin kriptovaliutos ir Ethereum kriptovaliutos su papildoma programavimo galimybe [CG18]. Ethereum ir Bitcoin šiuo metu dvi populiariausios blokų grandinės (angl. blockchain, toliau blockchain) [LCO+16]. Bitcoin - skaitmeniniai pinigai, kurių pavedimai vyksta internete naudojantis decentralizuota vieša duomenų baze - blockchain [Swa15]. Ethereum be savo kriptovaliutos turi ir kitą svarbų funkcionalumą - išmaniuosius kontraktus - Turing pilną (angl. complete) programą, kuri leidžia rašyti decentralizuotas aplikacijas [But14]. Solidity - populiariausia kalba, naudojama rašyti išmaniesiems kontraktams [Dan17].

Problema - per trumpą laiką buvo sukurta daug panašių išmaniųjų kontraktų, kurie dažnai kuriami kopijuojant jau esamus ICO išmaniuosius kontraktus bei pridedant reikiamus pakeitimus. Taip sukurta didelė aibė ICO išmaniųjų kontraktų, kurių kodas yra labai panašus arba kartojasi. Pernaudojamumo problemą spręsti bandoma Ethereum išmaniųjų kontraktų bibliotekomis, tačiau jos ne visada padengia visą ICO išmaniųjų kontraktų savybių aibę.

Produktų linijos programų inžinerija (angl. product line software engineering, toliau PLSE) naudojama įmonėse pakartojamumui susijusiuose aplikacijose numatyti. Kodas yra kartojamas skirtinguose produktuose tam, kad būtų užtikrintas jo pernaudojamumas. PLSE suteikia bendrą

architektūrą ir pernaudojamą kodą aplikacijos kūrėjams [SVB01]. PLSE kūrimas susideda iš savybių išskyrimo ir jų įgyvendinimo produkte. Gerai išskirtos produkto savybės padeda sukurti lengvai pernaudojamą programą. Savybės turi būti atrinktos atsižvelgiant į jų paplitimą bei kintamumą dalykinėje srityje [LKL15].

Savybių modeliavimas yra pagrindinis metodas atrinkti bei valdyti bendrąsias ir kintamas savybes produktų linijoje. Programų inžinerijos šeimos gyvavimo pradžioje savybių modelis padeda išskirti pagrindines savybes, kurios gelbsti kuriant naują rinką ar norint išlikti jau esamoje. Taip pat savybių modelis leidžia išskirti rizikingas savybes, nuspėti visos programos ar atskirų savybių kainą. Vėliau savybių modeliavimas padeda išskirti variacijos taškus programos architektūroje [CHE04]. Savybių modeliavimas yra populiariausias metodas PLSE kūrime nuo pat pirmojo jo pristatymo [KCH+90]. Savybės yra pakankamai abstraktus konceptas, padedantis efektyviai bendrauti suinteresuotoms šalims. Savybių modeliavimas yra intuityvus ir efektyvus būdas, žmonėms išreikšti savybių paplitimą ir kintamumą programos inžinerijos šeimoje [KL13].

Savybių pernaudojimo metodas (angl. feature-oriented reuse method, toliau - FORM) praplečia savybių modeliavimą bei nusako, kaip remiantis savybių modeliu sukurti programos inžinerijos architektūrą ir komponentus. FORM teigia, kad savybės apibūdina galimus galutinio produkto variantus, o kodas, kuris įgyvendina savybes, turi būti sukurtas pakartotiniam panaudojimui. Savybių modelis naudojamas pagelbėti srities architektūros kūrime bei programavime naudojant srities artefaktus. Išanalizavus bendrumą naudojantis savybių analize, savybių modelis yra naudojamas sukurti architektūrą bei komponentus [KKL+98].

Šio darbo tikslas - patikrinti ar Ethereum bibliotekos padengia visas savybes, kurias turi ICO išmanieji kontraktai patalpinti Ethereum blockchain, bei pasiūlyti būdą ICO išmaniųjų kontraktų kodo pernaudojamumui didinti.

Tikslui pasiekti išsikelti uždaviniai:

1. Išskirti savybių modelį ICO išmaniesiems kontraktams, patalpintiems Ethereum blockchain, ir jį validuoti;
2. Išskirti ICO savybių modelius OpenZeppelin ir TokenMarket bibliotekoms;
3. Palyginti savybių modelį, išskirtą iš Ethereum blockchain esančių ICO išmaniųjų kontraktų, su OpenZeppelin ir TokenMarket savybių modeliais;
4. Pasiūlyti pakeitimus vienai iš bibliotekų (tai, kuri yra labiau išvystyta), remiantis savybių modelių palyginimu.

1. Savybių modeliavimas

Savybių modeliavime bendri ir kintami bruožai yra modeliuojami iš produkto savybių perspektyvos dalykinėje srityje. Savybės yra pagrindinis produkto skiriamasis bruožas [LKL15]. Keli tos pačios srities programų inžinerijos produktai turi daug bendrų bruožų, bet taip pat kiekvienas produktas turi ir savo išskirtinumų. Tie bruožai iš naudotojo perspektyvos yra savybės [KCH+90]. Savybės yra suskirstytos į būtinas, alternatyvias ir pasirenkamas pagal bendrus ir kintamus bruožus [KL13]. Originalus savybių modeliavimas - FODA (angl. Feature-Oriented Domain Analysis (FODA), toliau FODA) [KCH+90] - paprastas modeliavimas, kuris savybes skirsto pagal „susideda iš" santykį bei pagal bendrumą ir specializaciją naudojant IR/AR (angl. AND/OR) diagramas.

1.1. Savybė

Skirtingi srities analizės metodai terminą savybė apibūdina šiek tiek kitaip [LKL15]. FODA [KCH+90] savybę apibūdina kaip pastebimą ir skiriamą sistemos charakteristiką, kuri yra matoma įvairioms suinteresuotoms šalims.

FODA fokusuojasi į kliento perspektyvą - ties paslaugomis, kurias teikia aplikacija ir aplinka, kurioje dirbama. Savybės yra programinės įrangos atributai, kurie tiesiogiai paveikia naudotoją [KCH+90]. Skirtumas tarp savybės ir konceptualios abstrakcijos (pvz.: funkcijos, objekto) yra tai, kad funkcijos ir objektai yra naudojami specifikuojant vidines programinės įrangos detales. Kitaip, funkcijos ir objektai yra konceptualios abstrakcijos, kurios yra identifikuojamos iš vidinės programinės įrangos pusės. Savybė - aiškiai matoma pagal charakteristiką, kuri gali išskirti produktą iš kitų. Todėl savybių modeliavimas turi išskirti iš išorės matomas charakteristikas produktuose bendrumo ir kintamumo atžvilgiu, o ne apibūdinti visas produkto modeliavimo detales (pvz.: funkcinis, objektinis modeliavimas). Suprantant produkto bendras ir kintamas savybes galima sukurti pernaudojamas funkcijas ir objektus [LKL15].

1.2. Savybių modelis

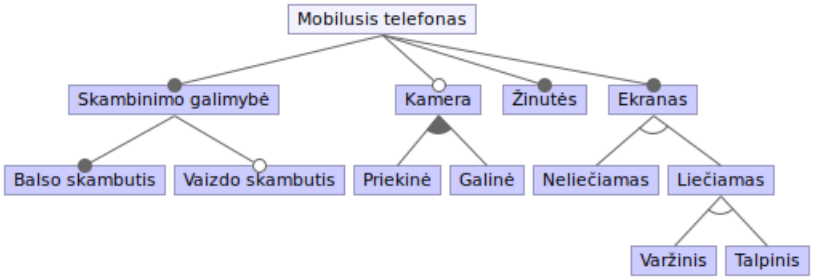
FODA [KCH+90] autoriai apibrėžia savybių modelį, kaip modelį, kuris turi pavaizduoti standartines savybes dalykinėje srityje ir santykius tarp jų. Trumpiau - savybių modelis yra hierarchiškai išskirstytų savybių rinkinys - „susideda iš" santykių rinkinys [Bat05; KCH+90]. Santykiai tarp savybių modelyje yra kategorizuojami į (1 pav.):

- Ir - visos vaikinės savybės turi būti pasirinktos;

- Alternatyva - tik viena vaikinė savybė gali būti pasirinkta;
- Ar - viena ar daugiau gali būti pasirinkta;
- Privaloma - savybė yra privaloma;
- Pasirenkama - savybė gali būti pasirenkama.

1 pav. Santykių kategorijos tarp savybių [Bat05]

Savybių diagrama yra grafinė savybių modelio reprezentacija. Tai modelis - medis, kur primityvios savybės - lapai, pagrindinės - mazgai (modelio pavyzdys - 2 pav.). Būtinos savybės tarp skirtingų produktų yra modeliuojamos kaip privalomos, kai skirtingos savybės tarp jų žymimos kaip alternatyvios ar pasirenkamos (angl. optional) [Bat05]. Pagrindinės savybės atributai yra paveldimi pagal visą jos specifikaciją. Visos pasirenkamos ar alternatyvios savybės, kurios negali būti pasirinktos, kai yra bendra savybė pasirinkta turi būti pažymėtos kaip tarpusavyje nesuderinamos (angl. mutually exclusive with). Visos pasirenkamos ir alternatyvios savybės, kurios turi būti pasirinktos, kai bendra yra pasirinkta, turi būti pažymėtos kaip privalomos. Savybių modelio dokumentacija susideda iš struktūrinės diagramos hierarchiškai suskaidančios savybes identifikuojančias pasirenkamas ir alternatyvias savybes, savybių apibūdinimo ir taisyklių kompozicijos savybėms [KCH+90].



2 pav. Savybių modelis mobiliajam telefonui [KL13]

Pasirenkamos ir alternatyvios savybės negali būti atrinktos savavališkai. Įprastai jos parenkamos pagal galutinio naudotojo (kliento) tikslus ar interesus [KCH+90]. Reikalavimai ir funkcijos yra apibūdinami kaip savybės, kadangi jos yra aiškiai atpažįstamos abstrakcijos (plačiau 1.1. Savybė) [LKL15]. Savybių modelis taip pat tarnauja kaip komunikacija tarp naudotojų ir kūrėjų. Naudotojui savybių modelis teikia informaciją, kokios yra savybės, iš kurių gali rinktis ir kada. Kūrėjams savybių modelis identifikuoja, ką reikėtų parametrizuoti kituose modeliuose bei programinės įrangos architektūroje ir kaip parametrizacija turi būti atlikta [KCH+90]. Kitaip

tariant, savybių modelis gelbsti ne tik pernaudojamų komponentų kūrime, bet ir valdant produktų konfigūraciją srityje [LKL15].

1.3. Procesas ir gairės

Savybių analizė susideda iš reikalingų dokumentų surinkimo, savybių išskyrimo, savybių abstrakcijos ir identifikavimo modelyje, savybių apibrėžimo, modelio validacijos [KCH+90]. Tačiau, prieš atliekant savybių modelį pirma turėtų būti išskirta sritis, kurios savybių analizė bus atliekama [LKL15].

1.3.1. Srities identifikavimas

Srities identifikavimas prasideda nustatant sritį su kuria bus dirbama. Pasirinkus sritį turi būti nubrėžtos ribos ir santykiai tarp srities elementų ir kitų esybių, esančių už srities ribų bei apibrėžti informacijos dalinimasis vieni tarp kitų. Srities modeliavimo tikslas yra nustatyti bendrus ir skirtingus konceptus ar charakteristikas sistemos kūrime [LKL15].

1.3.2. Savybių identifikavimas

Savybių identifikavimas susideda iš išskyrimo srities žinių gautų iš srities ekspertų ir kitų dokumentų tokių kaip knygos, naudotojo vadovo, projektavimo dokumentų ir jau parašytų programų [LKL15]. Aplikacijos savybes galima išskirti į keturias kategorijas:

- darbo aplinka, kurioje aplikacijos yra naudojamos;
- galimybės iš naudotojo perspektyvos;
- srities technologija - kokiais reikalavimais remiantis sprendimas yra padaromas;
- įgyvendinimo technika.

Visos identifikuotos savybės turi būti pavadintos ir konfliktai susiję su vardais turi būti išspręsti. Savybių sinonimai taip pat turi būti įtraukti į srities terminologijos žodyną [KCH+90].

1.3.3. Savybių abstrakcija, klasifikacija, modeliavimas

Sekantis žingsnis identifikavus savybes turėtų būti hierarchinio modelio sukūrimas pagal savybių klasifikavimą, struktūrizavimą naudojant „susideda iš" santykį. Ar savybė yra būtina, alternatyvi, pasirenkama turi būti identifikuojama modelyje [KCH+90].

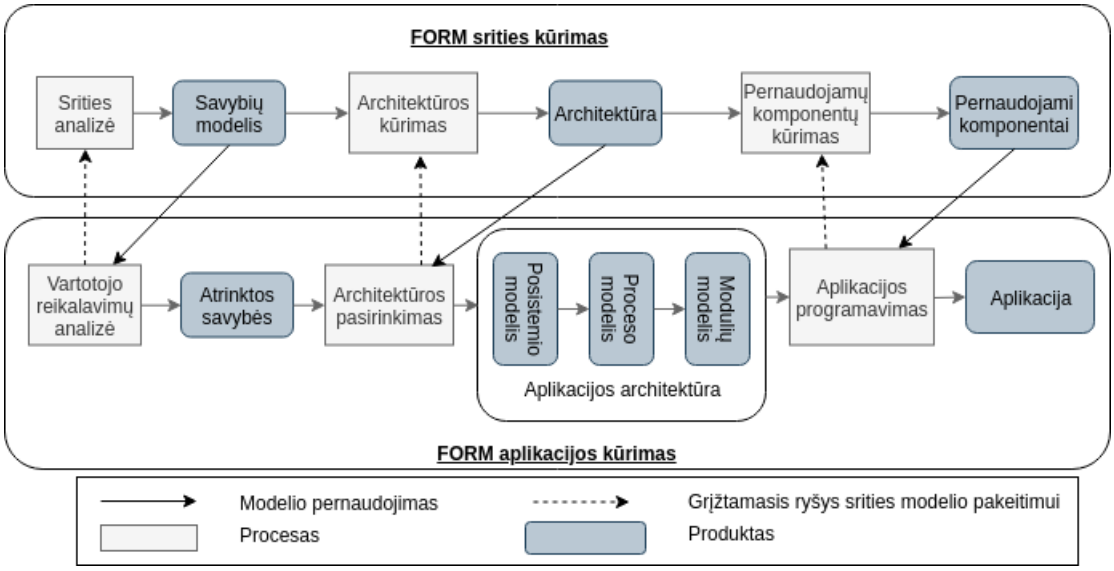
1.3.4. Savybių modelio validacija

Ar savybių modelis gerai reprezentuoja srities savybes turi būti validuota srities ekspertų ir jau egzistuojančias aplikacijas. Srities ekspertai, kurie konsultavo analizės metu, neturi dalyvauti validacijoje. Taip pat bent viena aplikacija, kuri nebuvo naudota analizėje, turi būti panaudota, kad būtų nustatytas modelio bendrumas ir pritaikomumas. Jei įmanoma, validuojant turi būti panaudotas naujas aplikacijų rinkinys [KCH+90].

2. Savybių pernaudojimo metodas

Skirtingai nei savybių pernaudojimo metodas dauguma metodų buvo orientuoti į vienos programos kūrimą. FORM yra skirtas analizuoti ir modeliuoti tam tikros srities aplikacijų bendrumus ir skirtumus, kad būtų sukurta srities architektūra ir komponentai [KKL+99]. FORM praplečia savybių modeliavimą bei nusako, kaip remiantis savybių modeliu sukurti programinės įrangos architektūrą ir komponentus [KKL+98]. FORM analizuoja ir modeliuoja bendrumus bei skirtumus srities aplikacijose remiantis veikimo, operacine aplinka, srities technologija ir įgyvendinimo metodu srityje. Savybės yra naudojamos sukurti objektus, kurie atspindėtų savybes ir sukurtų pernaudojamą srities architektūrą [LKC+00].

FORM susideda iš dviejų pagrindinių procesų: srities ir aplikacijos kūrimo (kaip parodyta 3 paveikslėlyje). Srities kūrimo procesas susideda iš sistemos analizavimo srityje ir standartinės architektūros ir komponentų kūrimo, remiantis analizės rezultatais. Aplikacijos kūrimas susideda iš veiklos skirtos kurti programas naudojant srities kūrime sukurtus artefaktus [KKL+98].



3 pav. FORM kūrimo procesai [KKL+98]

2.1. Srities architektūros ir komponentų kūrimas

FORM metodas suteikia sluoksninį architektūros karkasą. Remiantis savybių modeliu, hierarchija turi atskirti paslaugas (angl. service), operacijas, srities technologiją ir įgyvendinamus modulius. Pernaudojami artefaktai yra apibūdinami keturiais lygiais: 1) posistemio (angl. subsystem), 2) proceso, 3) modulių, 4) modulių specifikacijos ir įgyvendinimo [KKL+99]. Architektūra yra kuriama atsižvelgiant į tai, ar savybė yra funkcinė ir savybių suskirstymą į keturis lygius pagal jų tipus. Funkcinės savybės yra naudojamos nustatyti reikalingus komponentus,

nefunkcinės suskirsto komponentus bei nustato sąryšius tarp jų. Loginės ribos sukurtos savybių modelio turi atitikti fizines ribas architektūroje [KKL+98].

2.1.1. Srities analizė ir savybių modeliavimas

Srities analizė susideda iš: planavimo, savybių analizės ir validavimo. Analizės tikslas yra identifikuoti sistemos bendrumus ir skirtumus srityje. Tai padeda identifikuoti savybes, jas klasifikuoti, organizuoti į modelius (plačiau 1 skyriuje) [KKL+98].

2.1.2. Posistemio modelis

Posistemio modelis parodo pagrindines funkcionalumo grupes programų sistemoje, kiekviena jų gali būti išdėstoma skirtingose mašinose. Savybių pasiskirstymas skirtingose posistemiuose yra aukšto lygio (angl. high-level) veikimo bei operacinėse aplinkose [LKC+00]. Duomenų perdavimas tarp posistemių yra modeliuojamas neužsiblokuojančiais komunikacijos kanalais, laisvai sujungtais pranešimų eile, arba per blokuojamą pranešimų/atsakymų mechanizmą [KKL+98].

2.1.3. Proceso modelis

Kiekvienas posistemis yra dalinamas į procesus, kuriuose nusakomas procesų veikimas ir sąveika tarp jų [LKC+00]. Posistemis yra išskirstomas į procesus, perskirstančius operacines savybes į paslaugų savybes. Kiekvienas procesas gali būti skirstomas į ilgalaikį arba trumpalaikį bei į daugkartinį arba ne [KKL+98]. Posistemio ir proceso modeliai paslaugų savybes paskirsto į lengvai suskirstomus komponentus. Šios paslaugų savybės turi būti įgyvendintos kaip modulių rinkinys [KKL+99].

2.1.4. Modulių modelis

Kiekvienas procesas proceso modelyje yra toliau tobulinamas taip, kad būtų galima sukurti daugkartinio pernaudojimo komponentus - modulius. Modulių hierarchija atitinka savybių hierarchiją savybių modelyje [LKC+00]. Svarbu paminėti, kad hierarchija atskiria modulius, kurie įgyvendina savybes, kurios dažnai keičiasi, nuo žemesnio lygio operacijų ir įgyvendinimo metodų [KKL+99]. Pasirenkamos savybės turi būti įgyvendintos kaip šablono (angl. template) modulis ar aukštesnio lygio modulis su sąsaja, kuri padengtų visus galimus pasirinkimus [KKL+98].

2.2. Aplikacijos kūrimas

FORM aplikacijos kūrimas yra procesas, kurio metu naudojantis žiniomis apie sritį bei srities architektūrą, sukuriama aplikacija. Aplikacijos kūrimas prasideda nuo vartotojo reikalavimo surinkimo, pagal tai pasirenkant savybes iš savybių modelio, nustatant atitinkamus modelius bei kuriama aplikacija pasinaudojus pernaudojamais srities komponentais [KKL+98].

2.2.1. Reikalavimų analizė ir savybių pasirinkimas

Aplikacijos kūrimas prasideda nuo vartotojo reikalavimų analizavimo bei atitinkamų savybių suradimo modelyje. Savybių modelis ne tik nurodo galimas sistemos charakteristikas, bet ir apibrėžia tarpusavio santykius bei pasirinkimo kriterijus. Efektyvus metodas rasti tinkamoms savybėms yra pirmiausia apsvarstyti veikimui reikalingas savybes, tada operacinę aplinką ir srities technologijas bei jų įgyvendinimo būdus. Kiekvienas lygis suteikia daugiau detalumo, taip atspindi laipsnišką tobulinimą [KKL+98].

2.2.2. Architektūrinio modelio pasirinkimas ir programavimas

Savybių pasirinkimo etape automatiškai yra gaunamas atitinkamas architektūrinis modelis. Kai architektūra yra žinoma, pernaudojami komponentai yra lengvai randami (galima pasinaudoti prieš tai suprogramuotais komponentais, užpildyti skeletą ar parametrizuoti šablonus) [KKL+98]. Taip pasinaudojus srities architektūra lengvai gaunama aplikacijos architektūra bei jos kodas.

3. Savybių modeliavimas ICO išmaniesiems kontraktams

Buvo pasirinkta atlikti savybių modeliavimą remiantis procesu aprašytu 1.3. skyriuje. Savybių modeliavimas pradėtas nuo srities nusistatymo (plačiau 3.1. Srities identifikavimas). Toliau, buvo sudaryti trys savybių modeliai: 1) ICO išmaniesiems kontraktams, patalpintiems Ethereum blockchain, 2) OpenZeppelin bibliotekai, 3) TokenMarket bibliotekai. OpenZeppelin ir TokenMarket - atviro kodo Ethereum išmaniųjų kontraktų bibliotekos, populiariausios GitHub[1] versijų valdymui skirtame tinklapyje. Kiekvieno savybių modelio sudarymas susideda iš: savybių identifikavimo, jų abstrakcijos, klasifikacijos ir modeliavimo bei savybių modelio validacijos.

3.1. Srities identifikavimas

Šio savybių modeliavimo sritis - išmanieji kontraktai pirminiam kriptovaliutų platinimui. Kaip jau buvo minėta įvade: „Per trumpą laiką buvo sukurta daug panašių išmaniųjų kontraktų, kurie dažnai kuriami kopijuojant jau esamus ICO išmaniuosius kontraktus bei pridedant reikiamus pakeitimus. Taip sukurta didelė aibė ICO išmaniųjų kontraktų, kurių kodas yra labai panašus arba kartojasi. Pernaudojamumo problemą išspręsti bandoma Ethereum išmaniųjų kontraktų bibliotekomis, tačiau jos ne visada padengia visą ICO išmaniųjų kontraktų savybių aibę.” Todėl norint apžvelgti, kurias ICO išmaniųjų kontraktų savybes bibliotekos padengia, tiriami ICO išmanieji kontraktai, patalpinti Ethereum blockchain bei OpenZeppelin ir TokenMartket bibliotekose.

3.2. Savybių modeliavimas ICO išmaniesiems kontraktams, patalpintiems Ethereum blockchain

Pirmiausia tirtos savybės, kurias turi ICO išmanieji kontraktai, patalpinti Ethereum blockchain. Išmaniųjų kontraktų savybės tirtos surinkus aibę ICO išmaniųjų kontraktų. Išanalizavus jų savybes sudarytas savybių modelis bei patikrintas jo validumas.

3.2.1. Savybių identifikavimas

Savybių modelio sukūrimui yra būtinas savybių identifikavimas. Remiantis 1.3.2 skyriuje aprašytais savybių identifikavimo būdais buvo pasirinktas jau parašytų programų (šiuo atveju išmaniųjų kontraktų) surinkimas, analizavimas ir savybių juose identifikavimas. Išmaniųjų kontraktų surinkimo procesas:

[1] `https://github.com/`

1. Sukurtas interneto skaitytuvas[2] (angl. web crawler) naudojantis Node.js karkasu. Jis nuskaito visus išmaniuosius kontraktus, patalpintus ethrescan.io su žyma Token Sale[3](liet. žetonų pardavimas) bei turinčius daugiau nei 700 transakcijų;
2. Visi surinktų išmaniųjų kontraktų Solidity kodai buvo surašyti į failus. Failo pavadinimo struktūra: contract + identifikacijos numeris + .sol plėtinys;

Tokiu būdu buvo surinkta 161 pirminio kriptovaliutų platinimo išmaniųjų kontraktų. Iš jų atrinkta 40 ICO išmaniųjų kontraktų, kurie realizuoja tik ICO funkcionalumą (nerealizuoja žetono funkcionalumo). Išanalizavus atrinktus išmaniuosius kontraktus buvo išskirtos savybės ir apibūdintos dalinai naudojantis forma pateikta FODA [KCH+90]. Visos savybės pateiktos 3-oje lentelėje, pilna lentelės versija pateikta priede nr. 1. 3-oje lentelėje nurodytas tik skaičius ICO išmaniųjų kontraktų, patalpintų Ethereum, įgyvendinusių savybę. Prieduose numeriai atitinka surinktų ICO išmaniųjų kontraktų numerius.

3.2.2. Savybių abstrakcija, klasifikacija, modeliavimas

Savybių modelis ICO išmaniesiems kontraktams, patalpintiems Ethereum blockchain (4 pav.), buvo sukurtas naudojantis savybėmis, kurias turi šie kontraktai. Visos savybės yra pateiktos 3-ioje lentelėje. Savybių pavadinimai modelyje gali nesutapti su pavadinimais lentelėje, tačiau numeracija sutampa. Jei savybė lentelėje neturėjo sau priešingos, tai modelyje priešinga savybė jau yra įvardinama ir tai priimtina kaip savaime suprantamas reiškinys. Tokioms savybėms taip pat suteikta numeracija. Savybės, kurios yra tarp 90% ir daugiau išmaniųjų kontraktų, yra laikomas privalomomis. Modelyje nėra savybių, kurias turi tik 5% ir mažiau išmaniųjų kontraktų, tokios savybės laikomos pavienėmis ir dėl tos priežasties nėra įtrauktos į bendrą modelį. Visos savybės modelyje buvo klasifikuotos ir struktūrizuotos naudojant „susideda iš" santykį. Savybės modelyje identifikuotos naudojantis santykių kategorijomis tarp savybių (1 pav.).

Modelio kompozicijos taisyklės:

1. Žetonų susigrąžinimui (S2) reikalingas savininkas (S1);
2. Platinimo periodo keitimui bei pradėjimui (S7, S9, S8) reikalingas savininkas (S1);
3. Platinimo periodo laikinam sustabdymui (S3) reikalingas savininkas (S1);
4. Kainos pakeitimui, iškvietus funkciją, (S20) reikalingas savininkas (S1);
5. Žetonus atsiimti po ICO (S29) gali tik investuotojas (S13);
6. ETH susigrąžinti nepasiekus tikslo (S33) gali tik investuotojas (S13);

[2]Interneto skaitytuvas patalpintas: `https://github.com/maciukaite/bakalauro-darbas/tree/master/crawler`

[3]`https://etherscan.io/accounts/label/token-sale`

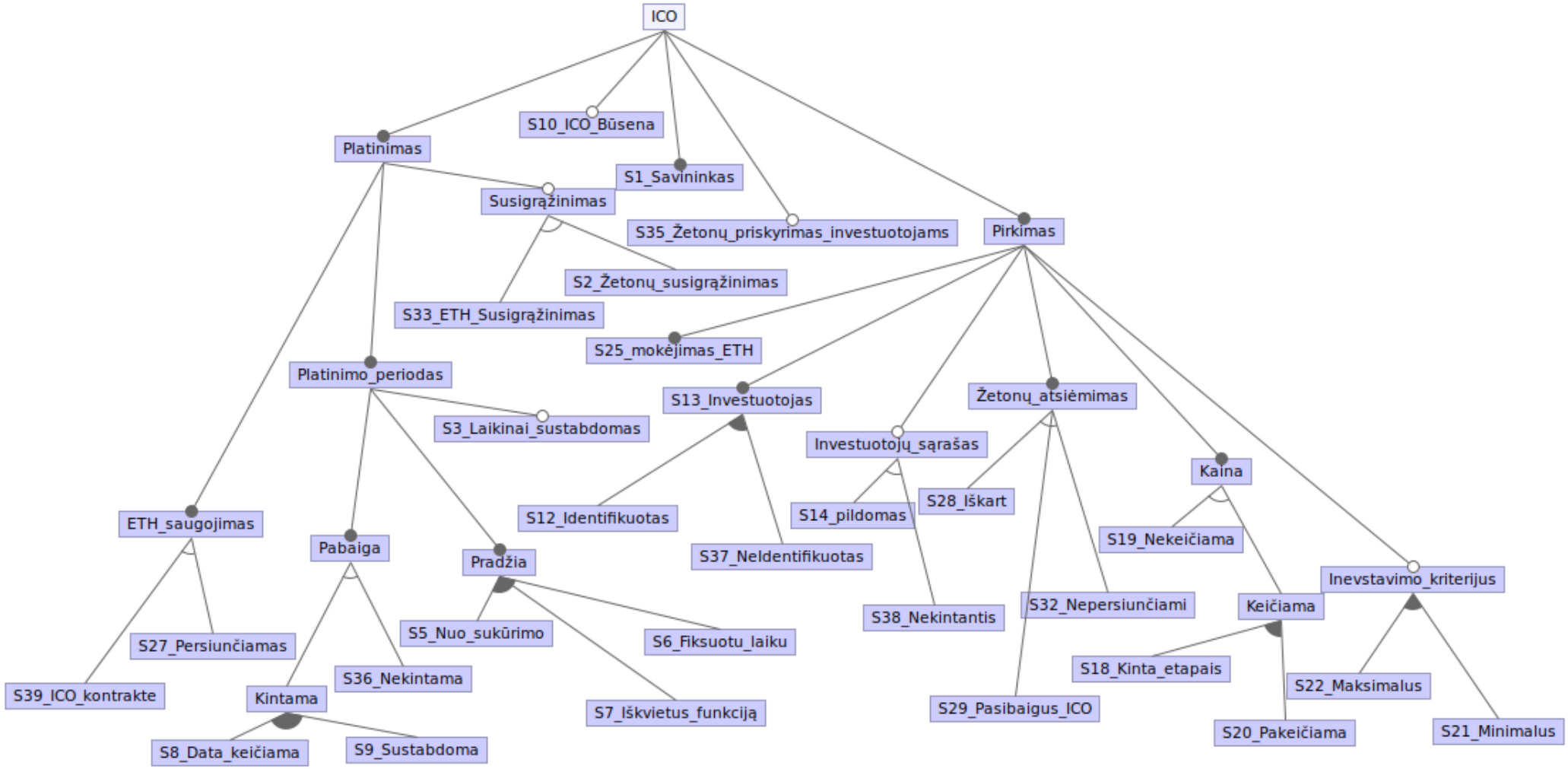
7. Investuotojų sąrašas egzistuoja tik tada (S38, S14), kai yra identifikuoti investuotojai (S12).

3.2.3. Savybių modelio validacija

Remiantis 1.3.4. skyriuje aprašytu procesu validuoti modelį turėtų srities ekspertai ar naujas dokumentų rinkinys. Kadangi šiame darbe negalima pasinaudoti srities ekspertais, tai validacijai naudojami dokumentai - išmanieji kontraktai, kurie buvo surinkti anksčiau (plačiau 3.2.1. skyriuje). Iš dar nenaudotų dokumentų buvo atsitiktinai atrinkta 10 išmaniųjų kontraktų ir patikrinta ar savybių modelis (4 pav.) yra validus. Validacijos rezultatas pateiktas lentelėje:

1 lentelė. ICO savybių modelio validacija

ICO išmanaus kontrakto numeris	**Savybės**	**Validumas**
1	S10, S1, S25, S18, S7, S3, S9, S2, S39, S28, S37	validus
20	S10, S1, S3, S6, S12, S18, S2, S27, S14, S28, S25, S8	validus
33	S1, S25, S6, S8, S21, S22, S28, S27,S37, S19	validus
67	S1, S25, S21, S18, S20, S3, S27, S6, S36, S37, S28	validus
87	S1, S25, S6, S36, S18, S21, S3, S27, S37, S28	validus
99	S1, S25, S21, S18, 36, S6, S3, S27, S28, S9	validus
126	S1, S25, S27, S36, S6, S37, S28, S19	validus
130	S1, S25, S27, S36, S6, S37, S28, S19	validus
145	S1, S25, S27, S36, S6, S37, S28, S19	validus
160	S1, S25, S39, S36, S6, S3, S12, S14, S29, S18, S22, S21	validus



4 pav. Savybių modelis ICO išmaniesiems kontraktams, patalpintiems Ethereum blockchain

3.3. Savybių modeliavimas OpenZeppelin ICO išmaniesiems kontraktams

OpenZeppelin - atviro kodo Ethereum išmaniųjų kontraktų biblioteka, populiariausia GitHub tinklapyje. OpenZeppelin biblioteka suteikia daug įvairių Ethereum išmaniųjų kontraktų. Šiam darbui pasirinkti tik tie išmanieji kontraktai, kurie realizuoja ICO arba yra tiesiogiai susiję su ICO išmaniaisiais kontraktais. Išanalizuotos tų kontraktų savybės bei sudarytas savybių modelis. Kadangi analizuojamos bibliotekos savybės, tai savybių modelio validacija nėra atliekama.

3.3.1. Savybių identifikavimas

Savybių identifikavimas OpenZeppelin bibliotekai buvo atliktas pasirinkus išmaniuosius kontraktus, kurie įgyvendina ICO funkcionalumą arba yra tiesiogiai su juo susiję. OpenZeppelin biblioteka yra patalpinta versijavimo sistemoje, pasirinkta specifinė kontraktų versija - darbo rašymo metu naujausia versija. Pasirinkta versija - 81e36d2e74ee8218339028bdf97a4e9116a5da6a[4]. Šios savybės taip pat pateiktos 3-oje lentelėje, pilna lentelės versija pateikta priede nr. 1. 3-oje lentelėje OpenZeppelin stulpelyje pateiktas faktas, ar biblioteka savybę įgyvendina. Prieduose pateikti išmanieji kontraktai turintys atitinkamas savybes. Pilna nuoroda iki išmanaus kontrakto: nuoroda į bibliotekos versiją + /contracts/crowdsale + /OpenZeppelin išmanusis kontraktas + .sol[5].

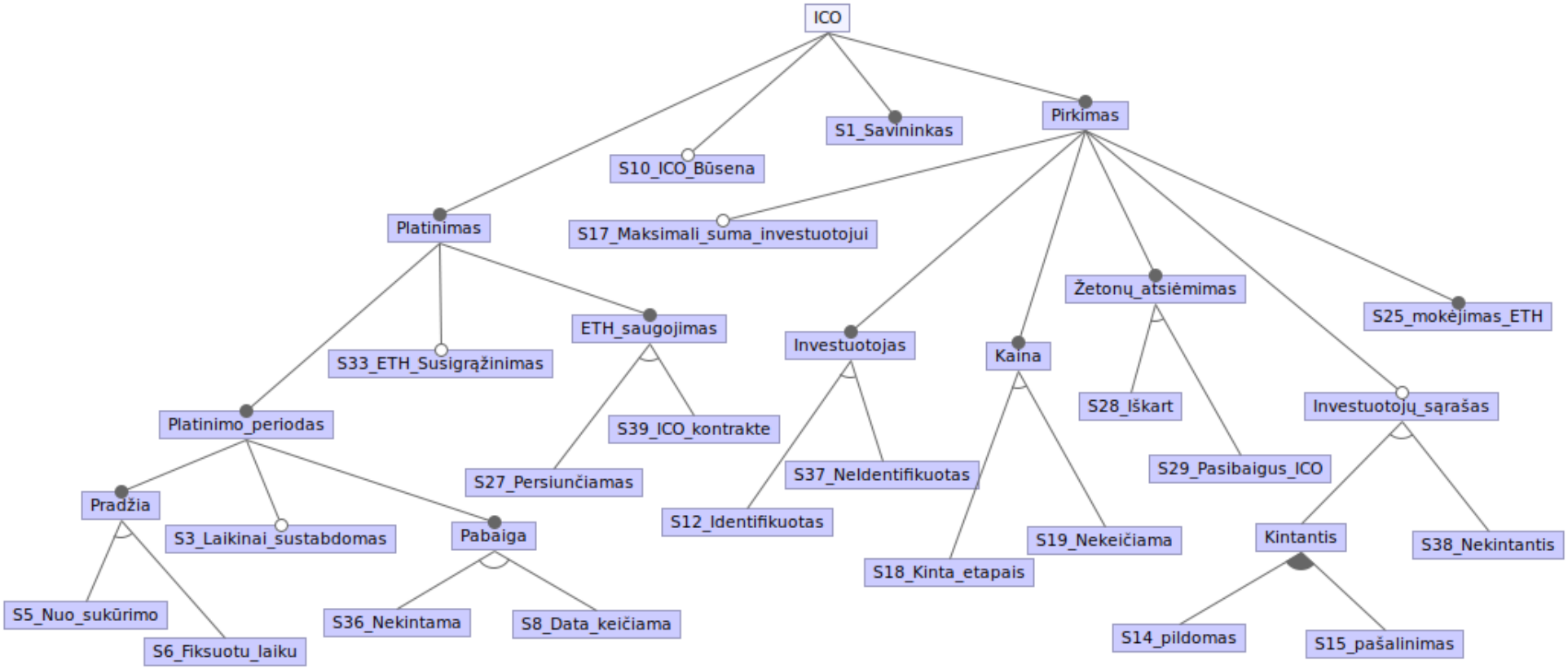
3.3.2. Savybių abstrakcija, klasifikacija, modeliavimas

Savybių modelis OpenZeppelin ICO išmaniesiems kontraktams (5 pav.) buvo sukurtas naudojantis savybėmis, kurias turi šie kontraktai. Visos savybės yra pateiktos 3-je lentelėje. Savybių abstrakcijos, klasifikacijos ir modeliavimo procesas nesiskyrė nuo proceso pateikto 3.2.2 skyriuje. Modelio kompozicijos taisyklės:

1. Platinimo periodo pabaigos datos keitimui (S8) reikalingas savininkas (S1);
2. Platinimo periodo laikinam sustabdymui (S3) reikalingas savininkas (S1);
3. Žetonus atsiimti po ICO (S29) gali tik investuotojas (S12, S37);
4. ETH susigrąžinti nepasiekus tikslo (S33) gali tik investuotojas (S12, S37);
5. Investuotojų sąrašas egzistuoja tik tada (S38, S14, S15), kai yra identifikuoti investuotojai (S12).

[4]Nuoroda į pasirinktą OpenZeppelin bibliotekos versiją: `https://github.com/OpenZeppelin/openzeppelin-solidity/tree/81e36d2e74ee8218339028bdf97a4e9116a5da6a`

[5]Pavyzdžiui, `https://github.com/OpenZeppelin/openzeppelin-solidity/blob/81e36d2e74ee8218339028bdf97a4e9116a5da6a/contracts/crowdsale/Crowdsale.sol`



5 pav. Savybių modelis OpenZeppelin ICO išmaniesiems kontraktams

3.4. Savybių modeliavimas TokenMarket ICO išmaniesiems kontraktams

TokenMarket - antra pagal populiarumą Ethereum išmaniųjų kontraktų biblioteka GitHub. Savybių modeliavimui pasirinkti tik tie išmanieji kontraktai, kurie įgyvendina ICO funkcionalumą arba yra tiesiogiai susiję su ICO išmaniaisiais kontraktais. Išanalizavus savybes, sudarytas bibliotekos savybių modelis. Kadangi analizuojamos bibliotekos savybės, tai savybių modelio validacija nėra atliekama. Nėra būdo atlikti validaciją bibliotekoms, nes nebuvo galimybės pasitelkti srities ekspertais, kurie nedalyvavo modeliavime, taip pat nėra naujo aplikacijų rinkinio, kuris galėtų būti naudojamas validacijai.

3.4.1. Savybių identifikavimas

Pasirinkti visi TokenMarket patalpinti išmanieji kontraktai, kurie įgyvendina ICO funkcionalumą arba yra tiesiogiai su juo susiję. TokenMarket bibliotekos analizavimui pasirinkta 067e241dffb7c7865b0f7628a875d285c7cd894c [6] versija. Visos savybės pateiktos 3-oje lentelėje (pilna lentelės versija pateikta priede nr. 1). 3-oje lentelėje pateiktas faktas apie savybės įgyvendinimą bibliotekoje. Prieduose TokenMarket stulpelyje pateikti išmanieji kontraktai turintys atitinkamas savybes. Pilna nuoroda iki išmanaus kontrakto: nuoroda į bibliotekos versiją + /contracts + /TokenMarket išmanusis kontraktas + .sol[7].

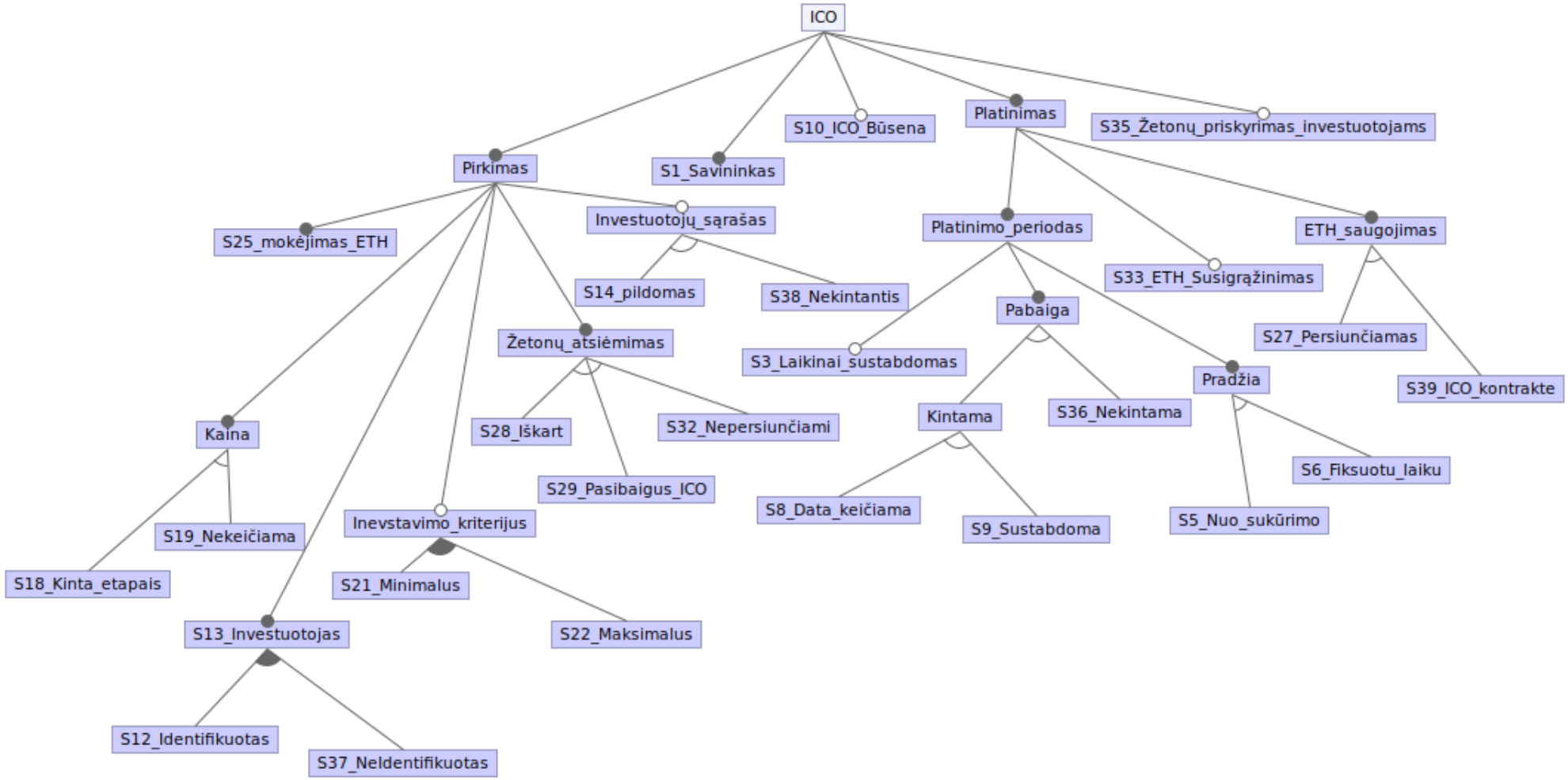
3.4.2. Savybių abstrakcija, klasifikacija, modeliavimas

Savybių modelis TokenMarket ICO išmaniesiems kontraktams (6 pav.) buvo sukurtas naudojantis savybėmis, kurias turi šie kontraktai. Visos savybės yra pateiktos 3-ioje lentelėje. Savybių abstrakcijos, klasifikacijos ir modeliavimo procesas nesiskyrė nuo proceso pateikto 3.2.2 skyriuje. Modelio kompozicijos taisyklės:

1. Platinimo periodo keitimui (S9, S8) reikalingas savininkas (S1);
2. Platinimo periodo laikinam sustabdymui (S3) reikalingas savininkas (S1);
3. Žetonus atsiimti po ICO (S29) gali tik investuotojas (S13);
4. ETH susigrąžinti nepasiekus tikslo (S33) gali tik investuotojas (S13);
5. Investuotojų sąrašas egzistuoja tik tada (S38, S14), kai yra identifikuoti investuotojai (S12).

[6] Nuoroda į pasirinktą TokenMarket bibliotekos versiją: `https://github.com/TokenMarketNet/smart-contracts/tree/067e241dffb7c7865b0f7628a875d285c7cd894c`

[7] Pavyzdžiui, `https://github.com/TokenMarketNet/smart-contracts/blob/067e241dffb7c7865b0f7628a875d285c7cd894c/contracts/CrowdsaleBase.sol`



6 pav. Savybių modelis TokenMarket ICO išmaniesiems kontraktams

3 lentelė. ICO išmaniųjų kontraktų savybių įgyvendinimas

Savybės pavadinimas	**Ethereum ICO išmaniųjų kontraktų skaičius**	**Įgyvendinta OpenZeppelin bibliotekoje**	**Įgyvendinta TokenMarket bibliotekoje**
S1. Savininkas	36	+	+
S2. Žetonų susigrąžinimas	8	-	-
S3. Laikinas sustabdymas	17	+	+
S4. Pradžios ir pabaigos datos nustatymas tik sukūrus išmanųjį kontraktą [8]	2	-	-
S5. ICO pradžia nuo sukūrimo	5	+	+
S6.ICO pradžia fiksuotu laiku	25	+	+
S7. ICO prasideda iškvietus funkciją	6	-	-
S8. Kintama ICO pabaiga	10	+	+
S9. Rankinis ICO sustabdymas	4	-	+
S10. ICO būsena	12	+	+
S11. Rankinis ICO būsenos pakeitimas[8]	2	-	-
S12. Identifikuotas investavimas	3	+	+
S13. Neidentifikuotas ir identifikuotas investavimas	10	-	+
S14. Pildomas identifikuotų investuotojų sąrašas	7	+	+
S15. Galimybė pašalinti investuotoją[8]	1	+	-
S16. Galimybė investuoti kitu adresu[8]	1	-	-
S17. Maksimali investavimo suma investuotojui	0	+	-

[8] Ši savybė laikoma paviene ir nėra įtraukta į modelį, nes jos neįgyvendina daugiau nei 5% išmaniųjų kontraktų

Savybės pavadinimas	**Ethereum ICO išmaniųjų kontraktų skaičius**	**Įgyvendinta OpenZeppelin bibliotekoje**	**Įgyvendinta TokenMarket bibliotekoje**
S18. Kintanti kaina	19	+	+
S19. Nekintanti kaina	12	+	+
S20. Pakeičiama kaina	8	-	-
S21. Minimalus investavimas	13	-	+
S22. Maksimalus investavimas	8	-	+
S23. Kintanti bendra investavimo suma[8]	1	-	-
S24. Žetonų pirkimas netiesiogiai[8]	2	-	-
S25. Žetonų pirkimas tiesiogai	37	+	+
S26. Žetonų pirkimas investavus BTC ar ETH[8]	1	-	-
S27. Visi ETH yra iš karto persiunčiami	13	+	+
S28. Žetonų persiuntimas iš karto	24	+	+
S29. Žetonų atsiėmimas pasibaigus ICO	9	+	+
S30. Žetonų atsiėmimas pasibaigus ICO, su savininko atrakinimu[8]	1	-	-
S31. Žetonai persiunčiami savininko[8]	1	-	-
S32. Žetonai nepersiunčiami	8	-	+
S33. ETH susigrąžinimo galimybė	14	+	+
S34. Žetonų padalijimas partneriams[8]	1	-	-
S35. Žetonų priskyrimas investuotojams	4	-	+

4. ICO savybių modelių palyginimas

Savybių modeliai palyginti remiantis sukurtais modeliais (modeliai pavaizduoti 4, 5, 6 paveikslėliuose). OpenZeppelin ir TokenMarket modeliai lyginti su ICO išmaniųjų kontraktų, patalpintų Ethereum blockchain, modeliu. Palyginti savybių skirtumai tarp šių modelių, išskirtos savybės, kurios modeliuose nesutampa.

4.1. Palyginimas su OpenZeppelin

Savybės, kurias turi ICO išmanieji kontraktai, patalpinti Ethereum blockchain, bet neturi OpenZeppelin ICO išmanieji kontraktai:

1. S2. Žetonų susigrąžinimas;
2. S7. ICO prasideda iškvietus funkciją;
3. S9. Rankinis ICO sustabdymas;
4. S13. Neidentifikuotas ir identifikuotas investavimas;
5. S20. Pakeičiama kaina;
6. S21. Minimalus investavimas;
7. S22. Maksimalus investavimas;
8. S32. Žetonai nepersiunčiami;
9. S35. Žetonų priskyrimas investuotojams.

Savybės, kurias turi OpenZeppelin ICO išmanieji kontraktai, bet neturi ICO išmanieji kontraktai, patalpinti Ethereum blockchain:

1. S17. Maksimali investavimo suma investuotojui.

4.2. Palyginimas su TokenMarket

Savybės, kurias turi ICO išmanieji kontraktai, patalpinti Ethereum blockchain, bet neturi TokenMarket ICO išmanieji kontraktai:

1. S2. Žetonų susigrąžinimas;
2. S7. ICO prasideda iškvietus funkciją;
3. S20. Pakeičiama kaina.

Nėra savybių, kurias turi TokenMarket ICO išmanieji kontraktai, bet neturi ICO išmanieji kontraktai, patalpinti Ethereum blockchain.

5. Pasiūlymai TokenMarket bibliotekai

Atsižvelgus į 4 skyrių buvo nuspręsta daryti palyginimus TokenMarket bibliotekai. Ši biblioteka pasirinkta, nes ji turi mažiau savybių nesutampančių su ICO išmaniaisiais kontraktais, patalpintais Ethereum blockchain - biblioteka labiau padengianti realų savybių poreikį.

Pasiūlymų teikimo procesas susideda iš: savybių surikiavimo pagal svarbumą, patikrinimo, ar savybės nebuvo įgyvendintos nuo to laiko, kai savybių modeliavimas buvo atliktas arba ar nėra pateikti pasiūlymai šių savybių įgyvendinimui, pasiūlymų įgyvendinti reikiamas savybes.

5.1. Savybių svarbumas

Visos trūkstamos savybės (jos yra pateiktos 4.2 skyriuje) pirmiausiai surikiuotos pagal svarbumą (svarbumas skirstomas pagal išmaniuosius kontraktus, kurie yra įgyvendinę šias savybes (informacija apie savybių įgyvendinimą išmaniuose kontraktuose pateikta 3 lentelėje)), svarbiausia savybė - įgyvendinta daugiausiai išmaniųjų kontraktų). Savybės pagal svarbumą:

1. S2. Žetonų susigrąžinimas (įgyvendinta 8-iuose išmaniuosiuose kontraktuose);
2. S20. Pakeičiama kaina (įgyvendinta 8-iuose išmaniuosiuose kontraktuose);
3. S7. ICO prasideda iškvietus funkciją (įgyvendinta 6-iuose išmaniuosiuose kontraktuose).

5.2. Savybių statuso patikrinimas naujausioje bibliotekos versijoje

Savybių modeliavimui buvo pasirinkta specifinė šios bibliotekos versija. Nuo to laiko, kai savybių modeliavimas buvo atliktas, atsirado ne viena nauja bibliotekos versija. Tam, kad būtų užtikrintas savybių ir kodo nesikartojimas, pirmiausia patikrinta naujausia bibliotekos versija darbo rašymo metu - 2c5b78d8e4f5d272f0eacba487f00408a36dfb95[9]. Taip pat patikrinti atviri prašymai pakeitimams (angl. pull request) ir iškeltos problemos (angl. issue) pateikti iki 2019-05-08 dienos.

Pirmiausia tikrinti pasikeitimai pagrindinėje (angl. master) bibliotekos šakoje (angl. branch). Patikrinti pasikeitimai pagrindinėje šakoje nuo tos versijos, kuri naudota savybių modeliavimui. Nustatyta, kad pasikeitimų, kurie galėtų įgyvendinti trūkstamas savybes, neįvyko.

2019-05-08 buvo 8 prašymai pakeitimams ir 20 problemų. Peržiūrėjus visus prašymus pakeitimams ir problemas, nustatyta, kad jie neįgyvendina trūkstamų savybių.

[9]Nuoroda į pasirinktą TokenMarket bibliotekos versiją: `https://github.com/TokenMarketNet/smart-contracts/tree/2c5b78d8e4f5d272f0eacba487f00408a36dfb95`

5.3. Siūlomi pakeitimai TokenMarket bibliotekai

Siūlomi pakeitimai įgyvendinti dalinai naudojantis FORM (plačiau 2 skyriuje). Posistemio ir proceso modeliai šiuo atveju yra neaktualūs, nes nebuvo tirta aplinka, su kuria išmanieji kontraktai sąveikauja, tirtas tik jų veikimas Ethereum blockhain. Nubraižytas TokenMarket ICO išmaniųjų kontraktų modulių modelis, jis papildytas naujai pridėtais išmaniaisiais kontraktais po pasiūlymų pateikimo (10 paveikslėlis). Taip pat papildomai nubraižytas savybių pasiskirstymas TokenMarket ICO išmaniuosiuose kontraktuose po pasiūlymų (11 paveikslėlis).

5.3.1. S2. Žetonų susigrąžinimas

Žetonų susigrąžinimo savybę nuspręsta įgyvendinti CrowdsaleBase išmaniajame kontrakte. Remiantis savybių modeliu ši savybė neprieštaraus jau esančiomis savybėmis, o tik praplės CrowdsaleBase funkcionalumą. Taip pat didelė dalis išmaniųjų kontraktų, kurie įgyvendino savybes esančias CrowdsaleBase, turėjo žetonų susigrąžinimo funkcionalumą.

TokenMarket GitHub saugykloje sukurta problema numeris 159[10]. Problema apibūdina atliktą tyrimą ir tai, kad jo metu nustatyta - žetonų susigrąžinimo savybė nėra įgyvendinta bibliotekoje. Taip pat pasiūlytas galimas kodas žetonų susigrąžinimo savybei. Kodas realizuotas bibliotekoje ir pasiūlytas pakeitimas bibliotekai[11] kartu su kitomis trūkstamomis savybėmis.

Žetonų susigrąžinimo savybės įgyvendinimui buvo sukurta viena funkcija (ji pateikta 7 paveikslėlyje). Ji sukurta naudojantis kodu, kuris buvo naudojamas tai pačiai savybei kituose išmaniuosiuose kontraktuose. Ši funkcija pridėta prie CrowdsaleBase išmanaus kontrakto.

```
1  function withdrawTokens ( address _token , address to , uint256 amount )
2  public
3  onlyOwner
4  {
5    FractionalERC20 token = FractionalERC20(_token);
6
7    if(!token.transfer(to , amount )) throw ;
8  }
```

7 pav. Žetonų susigrąžinimo savybės įgyvendinimas CrowdsaleBase išmaniajame kontrakte

[10]Nuoroda į problemą 159 TokenMarket bibliotekoje: https://github.com/TokenMarketNet/smart-contracts/issues/159

[11]Nuoroda į pasiūlymą TokenMarket bibliotekoje: https://github.com/TokenMarketNet/smart-contracts/pull/1629

5.3.2. S20. Pakeičiama kaina

Pakeičiamos kainos savybę nuspręsta pridėti prie MilestonePricing išmaniojo kontrakto. Toks pasirinkimas priimtas, nes šios savybės turi bendrą tėvinį mazgą ir 7/8 išmaniųjų kontraktų, kurie turi šią savybę, taip pat turi ir savybę S18. Kintanti kaina.

Šiai savybei sukurta dar viena problema - 160[12]. Problemas struktūra sutampa su 5.3.1 skyriuje apibūdinta problema. Savybės įgyvendinimui buvo sukurta funkcija (ji pateikta 8 paveikslėlyje). Ji sukurta naudojantis kodu, kuris buvo naudojamas tai pačiai savybei kituose išmaniuosiuose kontraktuose bei pritaikyta MilestonePricing. Funkcija pridėta prie MilestonePricing išmanaus kontrakto.

```
1  function setCurrentPrice (uint price) public onlyOwner {
2     uint i;
3
4    for(i=0; i<milestones.length; i++) {
5      if(now < milestones[i].time) {
6        milestones[i-1].price = price;
7      }
8    }
9  }
```

8 pav. Pakeičiamos kainos savybės įgyvydendinimas MilestonePricing išmaniajame kontrakte

5.3.3. S7. ICO prasideda iškvietus funkciją

ICO prasideda iškvietus funkciją savybei sukurtas naujas išmanusis kontraktas - StartableCrowdsale. Kadangi yra tik vienas kontraktas, kuris turi abi savybes: S7. ICO prasideda iškvietus funkciją ir S6. ICO pradžia fiksuotu laiku. Sudėti šias savybes į vieną išmanųjį kontraktą yra neaktualu, nes dauguma išmaniųjų kontraktų paveldi CrowdsaleBase savybę - S6. ICO pradžia fiksuotu laiku.

Savybei sukurta trečia problema - 161[13]. Problemos struktūra yra tokia pati, kaip kitų dviejų (plačiau 5.3.1 skyriuje). Problemai išspręsti sukurtas išmanusis kontraktas - StartableCrowdsale. Šis išmanusis kontraktas praplečia CrowdsaleBase funkcionalumą bei pridedą funkciją, kuri įgyvendiną ICO prasideda iškvietus funkciją savybę (išmanusis kontraktas pateiktas

[12]Nuoroda į problemą 160 TokenMarket bibliotekoje: https://github.com/TokenMarketNet/smart-contracts/issues/160

[13]Nuoroda į problemą 161 TokenMarket bibliotekoje: https://github.com/TokenMarketNet/smart-contracts/issues/161

9 paveikslėlyje).

```
import "./CrowdsaleBase.sol";

contract StartableCrowdsale is CrowdsaleBase{

  event StartAtChanged(uint newStartAt);

  constructor  (address _token, PricingStrategy _pricingStrategy,
  address _multisigWallet, uint _start, uint _end, uint _minimumFundingGoal)
    CrowdsaleBase (_token, _pricingStrategy, _multisigWallet,
    _start, _end, _minimumFundingGoal) public{}
    /**
    * @dev Start Crowdsale
    */
    function startCrowdsale () public onlyOwner inState(State.PreFunding){
      startsAt = now;
      StartAtChanged(startsAt);
    }
}
```

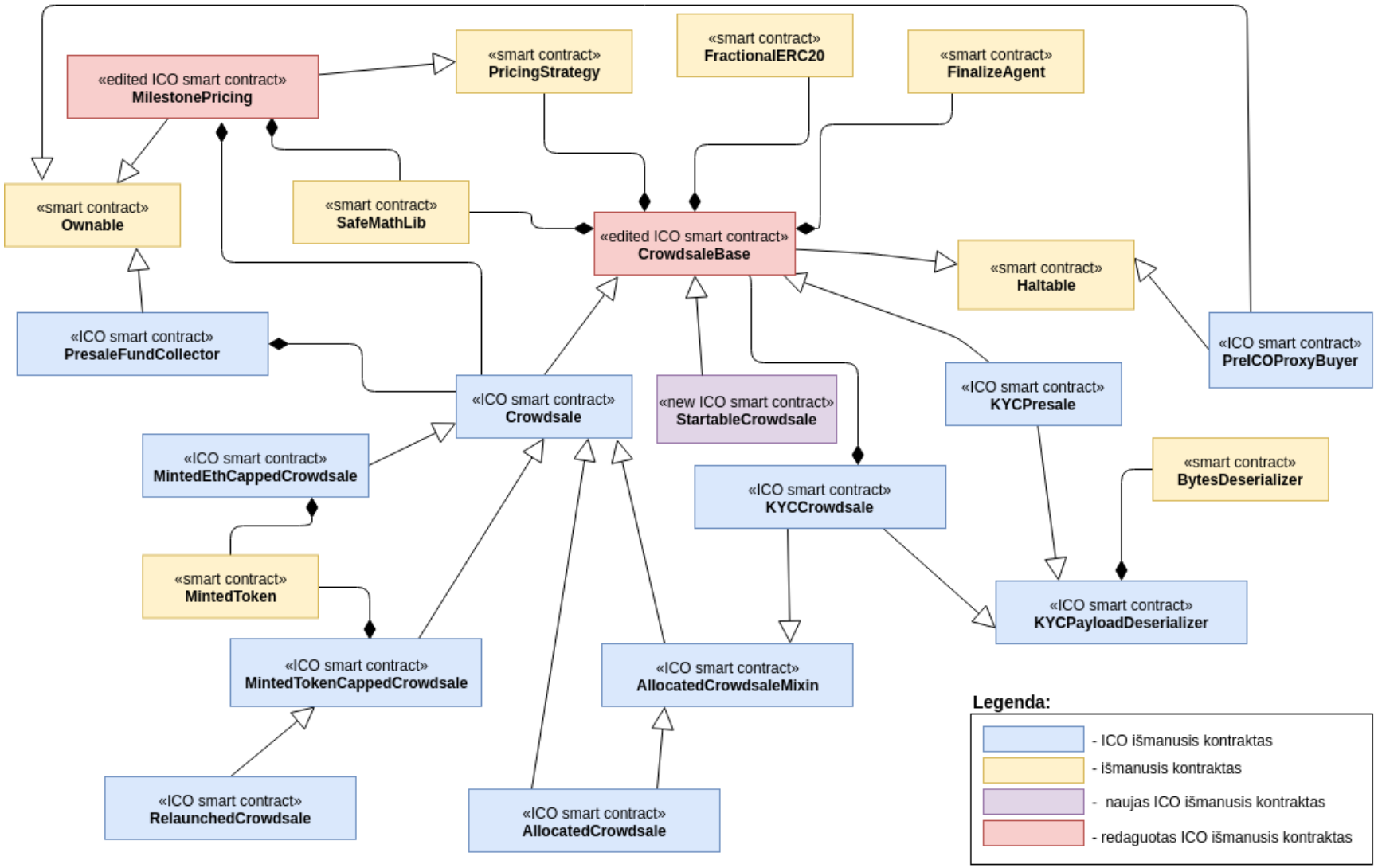
9 pav. Pakeičiamos kainos savybės įgyvendinimas MilestonePricing išmaniajame kontrakte

5.4. TokenMarket atsakymas į pasiūlytus pakeitimus

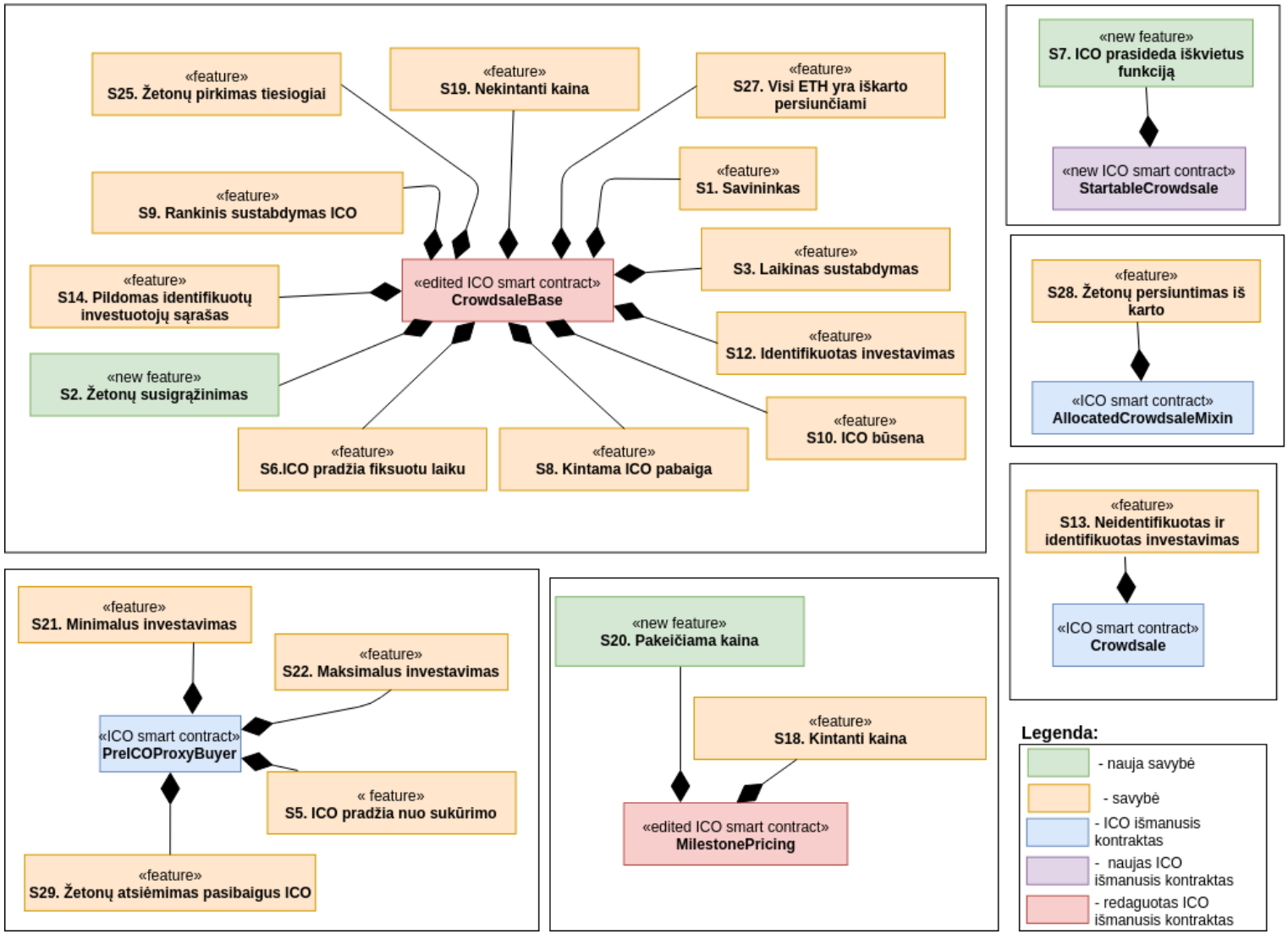
TokenMarket pateikė atsakymą į prašymą pakeitimui[14]. Jie atsakė, kad šiuo metu netęsia ICO išmaniųjų kontraktų kūrimo dėl pasikeitusios rinkos, šiuo metu jie fokusuojasi ties vetybinių popierių žetonų platinimu (angl. security token offering, toliau STO). STO - naujas žetonų platinimo būdas. TokenMarket turi STO kūrimo įrankį[15], kuris leidžia lengvai kurti vertybinių popierių žetonus bei juos platinti keliuose blockchain. Šis įrankis leidžia neturint daug programavimo žinių susikurti žetoną ir jį platinti.

[14]Nuoroda į atsakymą: https://github.com/TokenMarketNet/smart-contracts/pull/162#issuecomment-494827821

[15]Nuoroda į TokenMarket STO įrankį: https://github.com/TokenMarketNet/sto



10 pav. TokenMarket ICO išmaniųjų kontraktų modulių modelis po pasiūlymų pateikimo



11 pav. Savybių pasiskirstymas TokenMarket ICO išmaniuosiuose kontraktuose po pasiūlymų pateikimo

Rezultatai

Šiame darbe pasiekti rezultatai:

1. Surinkti 161 ICO išmanieji kontraktai, patalpinti Ethereum blokų grandinėje. Iš jų 40 panaudota išskiriant savybių modelį;
2. Išskirti OpenZeppelin ir TokenMarket ICO išmaniųjų kontraktų savybių modeliai;
3. Palyginti OpenZeppelin ir TokenMarket savybių modeliai su ICO išmaniųjų kontraktų, patalpintų Ethereum blockchain, savybių modeliu;
4. Sukurtos trys problemos TokenMarket bibliotekos GitHub talpykloje. Pasiūlyta, kaip trūkstamas savybes įgyvendinti bibliotekoje.

Darbo tikslas - patikrinti ar Ethereum bibliotekos padengia visas savybes, kurias turi ICO išmanieji kontraktai patalpinti Ethereum blockchain, bei pasiūlyti būdą ICO išmaniųjų kontraktų kodo pernaudojamumui didinti - pasiektas. Pernaudojamumą bandoma užtikrinti bibliotekomis. Pateikti pasiūlymai TokenMarket bibliotekai leistų padidinti pernaudojamumą ICO išmaniuosiuose kontraktuose.

Išvados

Išvados

Atlikus darbą daromos išvados:

1. Savybių modeliavimas yra geras būdas ICO išmaniųjų kontraktų kintamumui ir bendrumui nustatyti. Tačiau savybių pernaudojimo metodas šiai sričiai nevisiškai tinka. Posistemio ir proceso modeliai neaktualūs ICO išmaniųjų kontraktų sričiai, nors savybių pernaudojamumo metodas nurodo kurti šiuos modelius;
2. Savybių pernaudojimo metodas tyrinėjamas mažai: nedaug susijusių šaltinių, dažniausiai jie seni, dauguma autorių šaltiniuose sutampa;
3. TokenMarket ir OpenZeppelin bibliotekos savybėmis nepilnai padengia savybių, kurias turi ICO išmanieji kontraktai, patalpinti Ethereum blockchain. TokenMarket padengia didesnį savybių kiekį nei OpenZeppelin;
4. ICO evoliucionavo, STO yra naujas žetonų platinimo būdas, kuris galėtų pakeisti ICO. TokenMarket perėjimas prie šio žetonų platinimo būdo, rodo jo potencialą. Toliau plečiant TokenMarket STO įrankį, gali būti, kad ateityje nebereiks turėti daug programavimo žinių sukurti ir platinti žetonus.

Darbo tęsimas

Darbą toliau galima tęsti šiomis kryptimis:

1. Būtų galima išplėsti sritį bei tirti ne tik ICO savybes, bet ir žetonų savybes. Taip pat galima tirti sąryšius tarp žetono ir ICO išmaniųjų kontraktų. Taip būtų galima ištirti ir išmaniuosius kontraktus, kurie turi žetono ir ICO funkcionalumą. Kadangi didžioji dauguma išmaniųjų kontraktų šias sritis sujungia kartu (šiame darbe 121 išmanusis kontraktas iš 161 apėmė žetono ir ICO sritis);
2. Būtų galima pasiūlyti naują biblioteką, kuri pilnai remtųsi tik išmaniųjų kontraktų, patalpintų Ethereum blockchain ir savybėmis. Modulių modelis turėtų išplaukti iš savybių modelio, taip būtų galima sukurti biblioteką, kuri būtų visiškai paremta savybių modeliavimu ir savybių pernaudojimo metodu;
3. Būtų galima tęsti darbą STO tema bei palyginti jį su ICO iš savybių modeliavimo pusės. Galima sudaryti STO savybių modelį, palyginti jį su ICO savybių modeliu, nustyti žetonų platinimo būdų skirtumus iš savybių modeliavimo perspektyvos.

Literatūra

[Bat05] Don Batory. Feature Models, Grammars, and Propositional Formulas. *LNCS*, tom. 3714, p. 7–20, 2005. URL: http://www.cs.utexas.edu/ftp/predator/splc05.pdf (tikrinta 2019-05-08).

[But14] Vitalik Buterin. A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM. 2014. URL: https://www.weusecoins.com/assets/pdf/library/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf (tikrinta 2019-05-08).

[CG18] Christian Catalini ir Joshua S Gans. Initial Coin Offerings and the Value of Crypto Tokens, 2018. URL: https://ssrn.com/abstract=3137213 (tikrinta 2019-05-08).

[CHE04] Krzysztof Czarnecki, Simon Helsen ir Ulrich Eisenecker. Staged Configuration Using Feature Models. 2004. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.1586&rep=rep1&type=pdf (tikrinta 2019-05-08).

[Dan17] Chris Dannen. *Introducing Ethereum and Solidity*. 2017. ISBN: 978-1-4842-2535-6. DOI: 10.1007/978-1-4842-2535-6. URL: https://www.ikamy.ch/public/img/books/Introducing+Ethereum+and+Solidity.pdf (tikrinta 2019-05-08).

[HNY18] Sabrina T Howell, Marina Niessner ir David Yermack. Initial Coin Offerings: Financing Growth with Cryptocurrency Token Sales. Tech. atask., 2018. URL: http://www.nber.org/papers/w24774 (tikrinta 2019-05-08).

[KCH+90] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak ir A. Spencer Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, tom. 18 numeris 3-4. 1990. DOI: 10.1080/10629360701306050.

[KKL+98] Kyo C Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Gerard Jounghyun Kim ir Euiseob Shin. FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. Tech. atask., 1998. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.7568%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf (tikrinta 2019-05-08).

[KKL+99] Kyo C Kang, Sajoong Kim, Jaejoon Lee ir Kwanwoo Lee. Feature-Oriented Engineering of PBX Software for Adaptability and Reuseability. Tech. atask. 10, 1999, p. 875–896. URL: https://onlinelibrary.wiley.com/doi/pdf/10.1002/

(SICI)1097-024X(199908)29:10%7B%5C%%7D3C875::AID-SPE262%7B%5C%%7D3E3.0.CO;2-W (tikrinta 2019-05-08).

[KL13] Kyo C. Kang ir Hyesun Lee. Variability Modeling. *Systems and Software Variability Management*, p. 25–42. 2013. ISBN: 978-3-642-36582-9. DOI: 10.1007/978-3-642-36583-6. URL: http://link.springer.com/10.1007/978-3-642-36583-6 (tikrinta 2019-05-08).

[LCO+16] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena ir Aquinas Hobor. Making Smart Contracts Smarter, 2016. DOI: 10.1145/2976749.2978309. URL: https://loiluu.com/papers/oyente.pdf (tikrinta 2019-05-08).

[LKC+00] Kwanwoo Lee, Kyo C Kang, Wonsuk Chae ir Byoung Wook Choi. Feature-based approach to object-oriented engineering of applications for reuse. Tech. atask., 2000, p. 1025–1046. URL: https://onlinelibrary.wiley.com/doi/pdf/10.1002/1097-024X%2820000725%2930%3A9%3C1025%3A%3AAID-SPE323%3E3.0.CO%3B2-W (tikrinta 2019-05-08).

[LKL15] Kwanwoo Lee, Kyo C. Kang ir Jaejoon Lee. Concepts and Guidelines of Feature Modeling for Product Line Software Engineering. 2015. DOI: 10.1007/3-540-46020-9_5. URL: https://www.researchgate.net/profile/Kwanwoo_Lee/publication/221553200_Concepts_and_Guidelines_of_Feature_Modeling_for_Product_Line_Software_Engineering/links/558bdbde08ae591c19d8d3ce.pdf (tikrinta 2019-05-08).

[RR03] T. Ravichandran ir Marcus A. Rothenberger. Software reuse strategies and component markets. *Communications of the ACM*, 46(8), 2003. DOI: 10.1145/859670.859678. URL: https://www.researchgate.net/profile/T_Ravichandran/publication/220423696_Software_reuse_strategies_and_component_markets/links/54201aa60cf2218008d43cdb.pdf (tikrinta 2019-05-08).

[SVB01] Mikael Svahnberg, Jilles Van Gurp ir Jan Bosch. On the Notion of Variability in Software Product Lines, 2001. URL: http://www.diva-portal.org/smash/get/diva2:837870/FULLTEXT01.pdf (tikrinta 2019-05-08).

[Swa15] Melanie Swan. *Blockchain. Blueprint for a new economy*. 2015, p. 149. ISBN: 978-1-491-92049-7. URL: https://isidore.co/calibre/legacy/get/PDF/5503/CalibreLibrary/Blockchain_%20Blueprint%20for%20a%20Ne%20-%20Swan%2C%20Melanie_5503.pdf (tikrinta 2019-05-08).

[Tel94] Astro Teller. Turing Completeness in the Language of Genetic Programming with Indexed Memory, 1994. URL: http://www.astroteller.net/content/3-work/2-papers/17-turing-completeness-in-the-language-of-genetic-programming-with-indexed-memory/turing.pdf (tikrinta 2019-05-08).

[The01] Mike Thelwall. A web crawler design for data mining. *Journal of Information Science*, 27(319), 2001. DOI: 10.1177/016555150102700503. URL: https://journals.sagepub.com/doi/pdf/10.1177/016555150102700503?casa_token=rC5ToiWjm6EAAAAA%3Al5uBYBPvbLgOLwpGp0W1uU9ShpyukeeMSxGhhNwrlpCiLe0YIRme24ZHQ6VtQH_6lX3mt1db_Xzo& (tikrinta 2019-05-08).

Sąvokų apibrėžimai

Turing pilna (angl. complete) - bet kuri sistema, kuri yra pakankamai universali atpažinti visus galimus algoritmus [Tel94].

Interneto skaitytuvas (angl. web crawler) - programa, kuri nuskaito svetainę ar svetainių rinkinį, analizuoja surinktus puslapius ir praneša apie rezultatus. [The01].

Santrumpos

PLSE - produktų linijos programų inžinerija (angl. product line software engineering).

ICO - pirminis kriptovaliutų platinimas (angl. initial coin offering).

FODA - Feature-Oriented Domain Analysis [KCH+90].

FORM - savybių pernaudojimo metodas (angl. feature-oriented reuse method).

STO - vertybinių popierių žetonų platinimas (angl. security token offering).

Priedas nr. 1

ICO išmaniųjų kontraktų savybės

Savybės pavadinimas	**Apibūdinimas**	**Ethereum ICO išmaniųjų kontraktų numeriai**	**OpenZeppelin išmanieji kontraktai**	**TokenMarket išmanieji kontraktai**
S1. Savininkas	Išmanusis kontraktas turi priviligijuotą adresą - kontrakto savininką, kuris turi daugiau funkcionalumo nei įprastas naudotojas	0, 5, 8, 9, 10, 11, 15, 17, 19, 23, 28, 31, 34, 37, 41, 47, 49, 58, 59, 64, 68, 71, 77, 86, 90, 96, 106, 109, 110, 111, 113, 115, 139, 141, 153, 157	validation/PausableCrowdsale, validation/IndividuallyCappedCrowdsale	CrowdsaleBase
S2. Žetonų susigrąžinimas	Savininkas gali persisiųsti sau žetonus, kurie yra išmaniajame kontrakte	10, 11, 15, 28, 59, 76, 113, 141		
S3. Laikinas sustabdymas	ICO sustabdymo metu nebūtų galima nusipirkti žetonų	5, 8, 9, 17, 19, 23, 31, 41, 68, 77, 86, 90, 96, 113, 141, 153, 157	validation/PausableCrowdsale	CrowdsaleBase
S4. Pradžios ir pabaigos datos nustatymas tik sukūrus išmanųjį kontraktą [16]	ICO pradžios ir pabaigos datos gali būti nustatytos tik tada, kai išmanusis kontraktas jau yra sukurtas	11, 37		
S5. ICO pradžia nuo sukūrimo	ICO prasideda iš karto sukūrus išmanųjį kontraktą	0, 23, 68, 71, 96	Crowdsale	PreICOProxyBuyer
S6.ICO pradžia fiksuotu laiku	ICO yra pradedamas nuo laiko, kuris buvo priskirtas kaip ICO pradžios laikas	5, 8, 9, 10, 16, 19, 31, 34, 41, 49, 58, 59, 60, 77, 88, 90, 106, 110, 111, 113, 115, 139, 141, 153, 157	validation/TimedCrowdsale	CrowdsaleBase
S7. ICO prasideda iškvietus funkciją	Savininkas iškviečia ICO pradžios funkciją, nuo to momento prasideda ICO	47, 59, 64, 76, 86, 109		

[16] Ši savybė laikoma paviene ir nėra įtraukta į modelį, nes jos neįgyvendina daugiau nei 5% išmaniųjų kontraktų

Savybės pavadinimas	**Apibūdinimas**	**Ethereum ICO išmaniųjų kontraktų numeriai**	**OpenZeppelin išmanieji kontraktai**	**TokenMarket išmanieji kontraktai**
S8. Kintama ICO pabaiga	Savininkas gali pakeisti ICO pabaigos datą	5, 8, 9, 19, 31, 34, 37, 41, 47, 58	validation/TimedCrowdsale	CrowdsaleBase
S9. Rankinis ICO sustabdymas	ICO baigiasi savininkui iškvietus pabaigos funkciją	71, 86, 109 , 139		CrowdsaleBase
S10. ICO būsena	skirtingos ICO pakopos turi skirtingas būsenas	0, 28, 47, 49, 58, 64, 77, 90, 113, 115, 139, 141	distribution/RefundableCrowdsale	CrowdsaleBase
S11. Rankinis ICO būsenos pakeitimas[16]	Savininkas gali pakeisti ICO būseną iškviesdamas funkciją	0, 64		
S12. Identifikuotas investavimas	Pirkti žetonus gali tik prieš tai išmaniajame kontrakte įvardinti investuotojai	5, 8, 110	validation/WhitelistCrowdsale	CrowdsaleBase
S13. Neidentifikuotas ir identifikuotas investavimas	ICO metu gali pirkti žetonus tiek identifikuoti, tiek neidentifikuoti investuotojai. Identifikuoti investuotojai paprastai turi pirmumo teisę arba geresnę kainą	9, 19, 31, 41, 77, 111, 113, 115, 139, 141		Crowdsale
S14. Pildomas identifikuotų investuotojų sąrašas	Savininkas gali papildyti identifikuotų investuotojų sąrašą, kuriame nurodyti investuotojai, turintis teisę dalyvauti ICO	5, 8, 15, 110, 113, 115, 139	validation/WhitelistCrowdsale	CrowdsaleBase
S15. Galimybė pašalinti investuotoją[16]	Savininkas gali pašalinti investuotoją iš investuotojų sąrašo	15	validation/WhitelistCrowdsale	
S16. Galimybė investuoti kitu adresu[16]	Investavimo metu nurodomą, už kurį adresą yra investuojama ir jam persiunčiami žetonai	110		
S17. Maksimali investavimo suma investuotojui	Investuotojui yra nustatoma maksimali investavimo suma		validation/IndividuallyCappedCrowdsale	

Savybės pavadinimas	**Apibūdinimas**	**Ethereum ICO išmaniųjų kontraktų numeriai**	**OpenZeppelin išmanieji kontraktai**	**TokenMarket išmanieji kontraktai**
S18. Kintanti kaina	ICO metu žetonas gali turėti kelias kainas, skirtinga pakopa žetonų platinime - skirtinga kaina	5, 8, 9, 16, 19, 31, 34, 41, 49, 58, 60, 90, 110, 111, 113, 115, 139, 141, 157	price/IncreasingPriceCrowdsale	MilestonePricing
S19. Nekintanti kaina	Žetono kaina išlieka tokia pati viso ICO metu	10, 59, 64, 68, 71, 76, 77, 86, 96, 106, 109, 153	Crowdsale	CrowdsaleBase
S20. Pakeičiama kaina	ICO metu savininkas iškvietęs funkciją gali pakeisti kainą	5, 8, 9, 11, 19, 31, 41, 110		
S21. Minimalus investavimas	Nustatyta mažiausia reikšmė, kurią galima investuoti	15, 17, 34, 58, 59, 71, 86, 110, 111, 113, 139, 141, 153		PreICOProxyBuyer
S22. Maksimalus investavimas	Investuotojas negali investuoti daugiau nei yra nustatyta maksimali investavimo suma	64, 77, 86, 110, 111, 113, 141, 153		PreICOProxyBuyer
S23. Kintanti bendra investavimo suma[16]	Savininkas gali pakeisti bendrą maksimalaus investavimo sumą, tai yra sumą, kuri maksimaliai gali būti surenkama ICO metu	34		
S24. Žetonų pirkimas netiesiogiai[16]	Žetonai nėra perkami nusiuntus ETH į ICO išmanųjį kontraktą	0, 49		
S25. Žetonų pirkimas tiesiogiai	Mainas į atsiųstą ETH gaunama atitinkama dalis žetonų	5, 8, 9, 10, 11, 15, 16, 17, 19, 23, 28, 31, 34, 37, 41, 47, 58, 59, 60, 64, 68, 71, 76, 77, 86, 88, 90, 96, 109, 110, 111, 113, 115, 139, 141, 153, 157	Crowdsale	CrowdsaleBase
S26. Žetonų pirkimas investavus BTC ar ETH[16]	Galima investuoti tiesiogiai - ETH arba netiesiogiai - BTC	10		

Savybės pavadinimas	**Apibūdinimas**	**Ethereum ICO išmaniųjų kontraktų numeriai**	**OpenZeppelin išmanieji kontraktai**	**TokenMarket išmanieji kontraktai**
S27. Visi ETH yra iš karto persiunčiami	Investavus į žetoną, visi gauti ETH yra iškart pervedami į atskirą adresą	5, 8, 9, 15, 16, 19, 23, 31, 41, 60, 64, 110, 113	Crowdsale	CrowdsaleBase
S28. Žetonų persiuntimas iš karto	Žetonai yra nusiunčiami pirkėjui iškart atlikus investiciją	5, 8, 9, 10, 11, 15, 16, 17, 19, 31, 34, 41, 47, 60, 71, 76, 77, 88, 90, 110, 113, 115, 141, 157	Crowdsale	AllocatedCrowdsaleMixin
S29. Žetonų atsiėmimas pasibaigus ICO	Pirkėjas žetonus gali atsiimti tik pasibaigus ICO	0, 23, 49, 58, 59, 64, 86, 111, 139	distribution/PostDeliveryCrowdsale	PreICOProxyBuyer
S30. Žetonų atsiėmimas pasibaigus ICO, su savininko atrakinimu[16]	Savininkui iškvietus funkciją pradedamas nupirktų žetonų išdalinimas	0		
S31. Žetonai persiunčiami savininko[16]	Savininkas pasibaigus ICO visiems investuotojams perveda jų nusipirktus žetonus	15		
S32. Žetonai nepersiunčiami	Investuotojas negauna žetonų tiesiogiai iš ICO kontrakto. ICO kontraktas naudojamas tik susirinkti ETH ir apskaičiuoti, kiek žetonų investuotojas gaus	37, 58, 64, 96, 106, 109, 139, 153		KYCPresale
S33. ETH susigrąžinimo galimybė	Jei ICO nepasiekia nustatyto minimalaus suinvestuotų pinigų ribos, tai visi investuoti pinigai yra grąžinami investuotojams	5, 8, 9, 15, 16, 19, 31, 41, 86, 90, 115, 139, 141, 157	distribution/RefundableCrowdsale	CrowdsaleBase
S34. Žetonų padalijimas partneriams[16]	Pasibaigus ICO partneriai gali atsiimti priklausančius žetonus iš ICO išmaniojo kontrakto	0		

Savybės pavadinimas	**Apibūdinimas**	**Ethereum ICO išmaniųjų kontraktų numeriai**	**OpenZeppelin išmanieji kontraktai**	**TokenMarket išmanieji kontraktai**
S35. Žetonų priskyrimas investuotojams	Savininkas gali priskirti tam tikrą skaičių žetonų investuotojams, pirkusiems žetonus dar neprasidėjus ICO	9, 19, 31, 41		Crowdsale