VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Pakartotinis kodo panaudojimas pirminio kriptovaliutų platinimo išmaniuosiuose kontraktuose

Code Reuse in Initial Coin Offering Smart Contracts

Kursinis darbas

Atliko: 3 kurso 1 grupės studentė
Agnė Mačiukaitė (parašas)

Darbo vadovas: lekt. Gediminas Rimša (parašas)

Vilnius
2018

TURINYS

Įvadas

Kodo pernaudojimas leidžia naudoti programas keliuose projektuose. Tai yra svarbi strategija kodui, norint padidinti sistemos 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 naudojama ir kaip būtų valdomas jo pernaudojamumas. Tai padeda programuotojams nuspręsti ar kodas atitinka reikalavimus ir gali būti kuriamas, jei taip, tai ką reikia parametrizuoti ir kaip struktūrizuoti kodą, kad vėliau būtų galima ją pritaikyti skirtingiems kontekstams [KCH+90].

Pirminio kriptovaliuto platinimo (angl. initial coin offering, toliau ICO) metu įmonė parduoda specializuotus kripto-žetonus žadėdami, kad žetonai veiks kaip mainų priemonė gaunat paslaugas įmonės platformoje. Žetonų pardavimas kuria kapitalą pradiniam įmonės platformos kūrimui, nors nėra įsipareigojimo dėl būsimos paslaugos kainos (žetonais ar kitaip). Neseniai iškylęs ICO populiarumas yra paveiktas Bitcoin ir Ethereum sukurtos 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 complete programą, kuri leidžia rašyti decentralizuotas aplikacijas [But14]. Solidity - populiariausia kalba, naudojama rašyti išmaniesiems kontraktams [Dan17]. Problema - išmaniųjų kontraktų technologijos yra pakankamai jaunos, dėl to pakartotinio kodo panaudojimo bazė dar tik formuojasi. Šiuo metu ICO išmanieji kontraktai yra tiražuojami kopijavimo su modifikacijos būdu.

Produktų linijos programinė įranga (angl. product line software engineering, toliau PLSE) naudojama įmonėse pakartojamumui susijusiuose programinės įrangos produktuose numatyti. PLSE suteikia bendrą architektūrą ir pernaudojamą kodą programinės įrangos kūrėjams [SVB01]. Toks kūrimas susideda iš savybių išskyrimo ir jų įgyvendinimo produkte. Gerai išskirtos produkto ypatybės padeda sukurti lengvai pernaudojamą programą. Savybės turi būti atrinktos atsižvelginat į jų paplitimą bei kintamumą srityje [LKL15]. Naudojantis PLSE produkto kūrėjai gali fokusuotis produkto specifikacijoje, o ne bendrose savybėse [SVB01].

Savybių modeliavimas yra pagrindinis metodas atrinkti bei valdyti bendrąsias ir kintamas savybes produktų linijoje. Programinės įrangos 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, kokia yra visos programos ar atskirų savybių kaina. Vėliau savybių modeliavimas padeda išskirti variacijos taškus programinės įrangos architektūroje [CHE04]. Savybių modeliavimas yra populiariausias PLSE kūrime nuo pat pirmojo jo pristatymo [KCH+90]. Kadangi savybės yra pakankamai abstraktus konceptas, padedantis efektyviai bendrauti suinterasuotoms šalims. Savybių modeliavimas yra intuityvus ir efektyvus būdas, žmonėms išreikšti savybių paplitimą ir kintamumą programinės įrangos šeimoje [KL13].

Šio darbo tikslas - ištirti pirminio finansavimo kriptovaliutomis išmaniuosius kontraktus, nustatyti, kokios savybės yra pastavios, o kokios - kintamos bei pasiūlyti būdus kodo pernaudojamu-

mui didinti.

Tikslui pasiekti išsikelti uždaviniai:

1. Apžvelgti savybių modeliavimą programinės įrangos produktų linijos sričiai;
2. Surinkti virš 100 išmaniųjų kontraktų skirtų ICO;
3. Išskirti surinktų išmaniųjų kontraktų savybes;
4. Sukurti ICO savybių modelį ir jį validuoti.

1. Savybių modeliavimas

Savybių modeliavime bendri ir kintami bruožai yra modeliuojami iš produkto savybių perspektyvos PLSE. Originalus savybių modeliavimas - FODA (angl. Feature-Oriented Domain Analysis (FODA), toliau FODA) [KCH+90] - paprastas modelis, kuris savybes skirsto pagal „susideda iš" santykį bei pagal bendrumą ir specializaciją naudojant IR/AR (angl. AND/OR) diagramas. Savybės yra suskirstytos į būtinas, alternatyvias ir pasirenkamas pagal bendrus ir kintamus bruožus [KL13].

1.1. Savybė

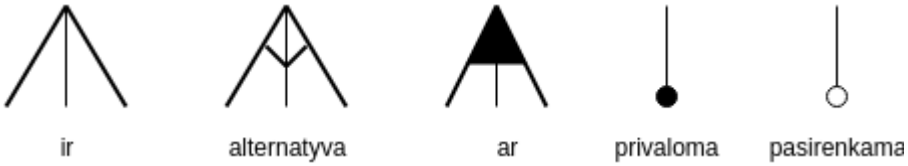
Keli tos pačios srities programinės įrangos produktai turi daug bendrų galimybių, bet taip pat kiekviena programinė įranga turi ir savo išskirtinumų. Tos galimybės iš naudotojo perspektyvos yra savybės [KCH+90]. Savybės yra pagrindinis produkto skiriamasis bruožas. Skirtingi srities analizės metodai 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 ties kliento perspektyva, tai yra 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, objektais orientuotas modeliavimas). Suprantant produkto bendrus ir kintamus bruožus galima sukurti pernaudojamas funkcijas ir objektus [LKL15].

1.2. Savybių modelis

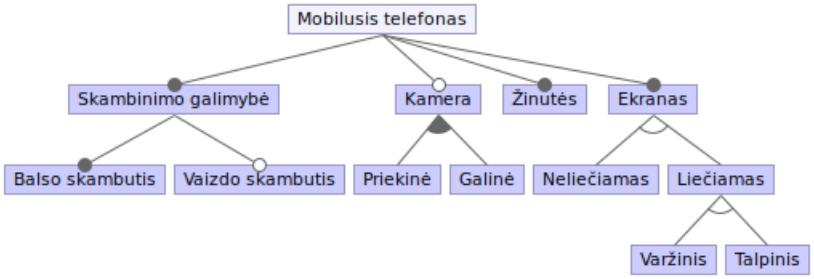
FODA [KCH+90] autoriai apibrėžia savybių modelį, kaip modelį, kuris turi pavaizduoti standartines sistemos šeimos savybes srityje ir santykius tarp jų. Trumpiau - savybių modelis yra hierarchiškai išskirstytų savybių rinkinys - „susideda iš" santykių rinkinys [Bat05; KCH+90]. Santykiai tarp savybių modelyje 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]. Bendros savybės atributai yra paveldimi pagal visą jos specifikaciją. Visos pasirenkamos ar alternetyvios 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 stuktūrinės diagramos hierarchiškai suskaidančios savybes indentifikuojančias pasirenkamas ir alternatyves 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]. Labai naudinga yra tai, kad savybė yra efektyvus komunikavimo būdas tarp suinteresuotų šalių. Dažnai klientai ir inžinieriai kalba apie produkto charakteristiką savybių pavidalu. 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, 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 nustantant sritį su kuria bus dirbama. Pasirinkus sritį turi būti nubrėžtos ribos ir santykiai tarp srities elementų ir kitų esybių, esančių už srities ribų bei 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, nadotojo 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 sinomimai 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, strūkturizavimą naudojant „susideda iš" santykį. Ar savybė yra būtina, alternatyvi, pasirenkama turi būti identifikuojama modelyje. Kiekviena savybė modelyje turi būti apibrėžta [KCH+90].

1.3.4. Savybių modelio validacija

Ar savybių modelis gerai reprezentuoja srities savybes turi būti validuota prieš srities ekspertus ir jau egzituojančias 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 modeliavimas ICO išmaniesiems kontraktams

Buvo pasirinkta atlikti savybių modeliavimą remiantis procesu aprašytu 1.3. skyriuje. Savybių modeliavimas buvo pradėtas nuo srities nusistatymo (plačiau 2.1. Srities identifikavimas), tada buvo surinkta informacija reikalinga srities modelio sudarymui ir savybių išskyrimui bei atrinktos savybės (plačiau 2.2. Savybių identifikavimas), sekantis žingsnis buvo modelio sudarymas (plačiau 2.3. Savybių modelis) ir jo validacija (plačiau 2.4. Savybių modelio validacija).

2.1. Srities identifikavimas

Šio savybių modeliavimo sritis - išmanieji kontraktai pirminiam kriptovaliutų platinimui. Kaip jau buvo minėta įvade: „Išmaniųjų kontraktų technologijos yra pakankamai jaunos, dėl to pakartotinio kodo panaudojimo bazė dar tik formuojasi. ICO kontraktai yra tiražuojami kopijavimo su modifikacijos būdu.” Todėl šiame savybių modeliavime bus apsirobojama tik ICO naudojamais kontraktais ir nebus analizuojami santykiai su kitais išmaniaisiais kontraktais ar sistemomis.

2.2. Savybių identifikavimas

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

1. Iš Ethereum blockchain blokų naršyklės - etherscan.io[1], buvo atrinkti išmaniųjų kontraktų adresai ir nuorodos į juos platformoje;
2. Parašytas interneto skaitytuvas[2] (angl. web crawler) naudojantis Node.js karkasu;
3. Visi surinktų išmaniųjų kontraktų kodai buvo surašyti į failus[3]. Failo pavadinino struktūra: contract + identifikacijos numeris + .sol plėtinys;

Tokiu būdu buvo surinkta 128 pirminio kriptovaliutų platinimo išmanieji kontraktai. Savybės iš išmaniųjų kontraktų buvo išskirtos analizavimo būdu - pirmiausia atrinkta 21 išmanusis kontraktas, kuris neturi savybės - žetono sukūrimas ir disponavimas (priežastys šiam pasirinkimui yra pateiktos 2.1 skyriuje). Išanalizavus atrinktus kontraktus buvo išskirtos savybės ir apibūdintos dalinai naudojantis forma pateikta FODA [KCH+90]:

[1] `https://etherscan.io/`

[2] Interneto skaitytuvas patalpintas: `https://github.com/maciukaite/kursinis/tree/crawler-etherscan`

[3] Visi failai patalpinti: `https://github.com/maciukaite/kursinis/tree/crawler-etherscan/res`

1 lentelė. Atrinktos aibės ICO išmaniųjų kontraktų savybės

Savybės pavadinimas	**Apibūdinimas**	**ICO išmaniųjų kontraktų numeriai**
Žetono kainos nustatymas skirtingomos pakopomis	ICO metu žetonas gali turėti kelias kainas, skirtinga pakopa žetonų platinime - skirtinga kaina	2, 23, 43, 46, 55, 62, 85, 120
Minimalaus investavimo kriterijus	Naudotojas turi būtinai ivestuoti sumą nemažense nei 15	2, 16, 46, 106
ETH susigrąžinimo galimybė nepasiekus tikslo	Jei ICO nepasiekia nusistatyto minimalaus suinvestuotų pinigų ribos, tai visi investuoti pinigai yra grąžinami naudotojams	2, 23, 46, 62, 85, 94, 96, 120
Žetonų atsiėmimas tik įvykus ICO	Įvykus ICO naudotojai gauna nusipirktus žetonus	2, 126
Kainos už žetoną pakeitimas	Žetono kaina gali kisti neatsižvelgiant į pasikeičiančias pakopas	2, 40
Kontrakto savininko nustatymas bei kai kurio funkcionalumo priskyrimas tik jam	Išmanusis kontraktas turi priviligijuotą adresą - kontrakto savininką, kuris turi daugiau funkcionalumo nei įprastas naudotojas	2, 12, 13, 16, 23, 31, 43, 46, 62, 72, 85, 86, 94, 106, 114, 116, 120, 126
Galimybė sustabdyti ICO ir vėl paleisti iš naujo	ICO sustabdymo metu nebūtų galima nusipirkti žetonų	12, 31
Galimybė pirkimo metu nustatyti kitą adresą kaip pirkėją	Naudotojas pirkdamas žetonus nustato adresą (pirkėją), kurį atstovauja	12, 31, 62
Pirkimas tik indentifikuotiems naudotojams	Pirkti žetonus gali tik anksčiau į išmaniajame kontrakte pažymėti naudotojai	12, 16, 31
ICO sustabdymas ar galimybė keisti datą	ICO galima sustabdyti ar keisti jo pabaigos datą	16, 43, 46, 86, 106
Pelno pasiskirstymas tarp komandos ir įmonės	Surinkta pinigų suma yra paskirstyta ne tik įmonei, bet ir komandai	23

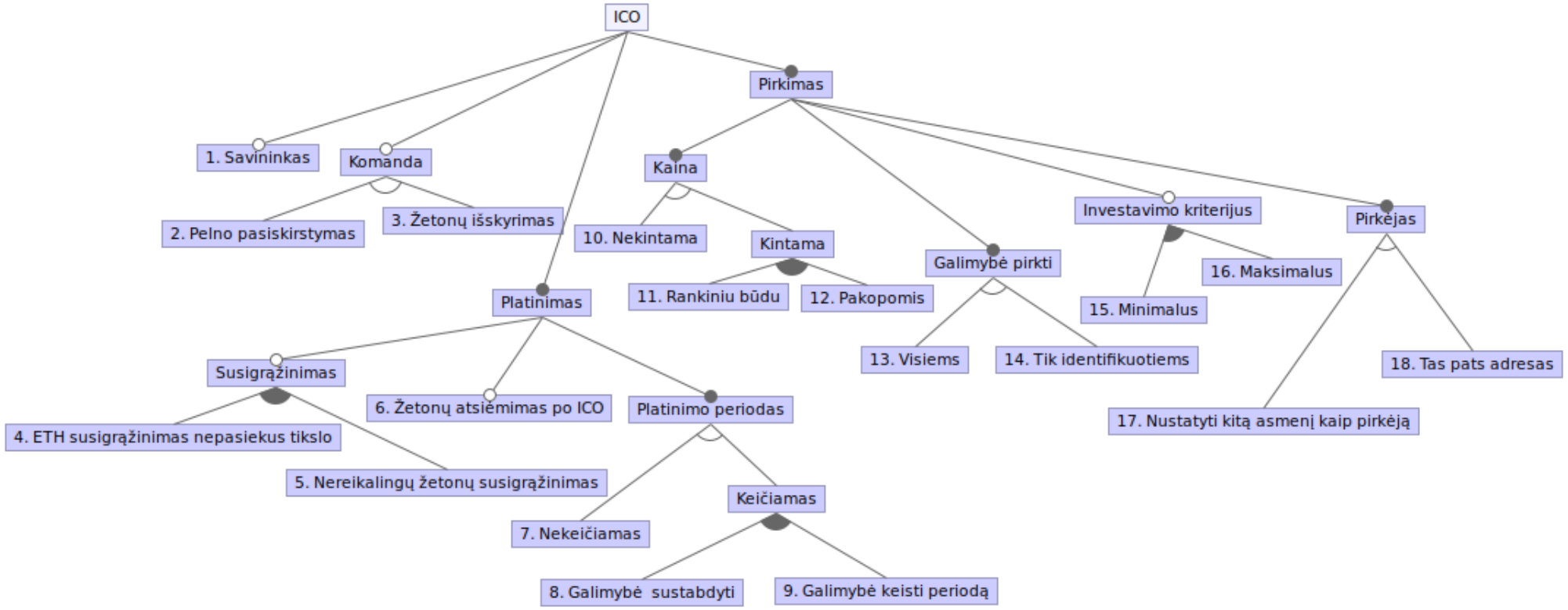
Savybės pavadinimas	Apibūdinimas	ICO išmaniųjų kontraktų numeriai
ICO pradžios nustatymas	ICO pradžios data gali būti keičiama	40, 86, 106, 126
Maksimalaus galimo surinkti ETH keitimas	Galimybė papidinti ar sumažinti investicijų sumos skaičių	43, 86
Nustatytas maksimalus galimas žetonų kiekis nupirktas per kartą	Naudotojas gali nusipirkti ne daugiau žetonų nei yra nustatyta	43
Nereikalingų žetonų išsitraukimas iš kontrakto	Likusių žetonų nuo ICO susigrąžinimas	13, 62, 94, 114, 120, 126
Žetonų išskyrimas komandai	Žetonų paskirstymas komandos nariams	62, 72, 94, 114, 120, 126

2.3. Savybių abstrakcija, klasifikacija, modeliavimas

Savybių modelis ICO išmaniesiems kontraktams (3 pav.) buvo sudarytas naudojantis savybėmis, kurios yra išvardintos 2.2. skyriuje, 1 lentelėje. Savybių pavadinimai esantys lentelėje ir modelyje gali nesutapti, kadangi savybės modelyje yra padaryta savybių abstrakcija ir kai kurios savybės ar jos dalys yra sujungtos. Jei savybė lentelėje neturėjo sau priešingos, tai modelyje priešinga savybė jau yra įvardinama ir tai priimtina kaip savaime suprantamas reiškinys. Visos savybės buvo klasifikuotos ir struktūrizuotos naudojant „susideda iš" santykį, taip pat savybės modelyje identifikuotos naudojantis santykių kategorijomis tarp savybių (1 pav.). Modelio kompozicijos taisyklės:

1. Nereikalingų žetonų susigrąžinimui reikalingas savininkas;
2. Platinimo periodo keitimui reikalingas savininkas;
3. Kainos pakeitimui rankiniu būdu reikalingas savininkas;
4. Žetonus atsiimti po ICO gali tik pirkėjas;
5. ETH susigrąžinti nepasiekus tikslo gali tik pirkėjas.

Savybės, kurios neturi vaikinių savybių, turi identifikacijos numerį, tam, kad validuojant modelį būtų galima lengviau identifikuoti bendras savybes.



3 pav. Savybių modelis ICO išmaniesiems kontraktams

2.4. 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 2.2. skyriuje). Iš dar nenaudotų dokumentų buvo atsitiktinai atrinkta 10 išmaniųjų kontraktų ir patikrinta ar savybių modelis (3 pav.) yra validus. Validacijos rezultatas pateiktas lentelėje:

2 lentelė. ICO savybių modelio validacija

Dokumentas	**Savybės numeris**	**Validumas**
contract0	1, 5, 7, 10, 13, 15, 16, 18	validus
contract6	1, 7, 10, 13, 15, 18	validus
contract22	1, 5, 9, 10, 13, 15, 18	validus
contract33	1, 8, 9, 11, 14, 18	validus
contract39	1, 4, 8, 9, 10, 13, 15, 18	validus
contract51	1, 3, 5, 7, 10, 13, 18	validus
contract71	7, 10, 13, 18	validus
contract91	7, 10, 13, 18	validus
contract103	1, 8, 9, 10, 14, 15, 16, 18	validus
contract122	1, 7, 10, 13, 15, 16, 18	validus

Rezultatai ir išvados

Šiame darbe pasiekti rezultatai:

1. Apžvelgtos kodo pernaudojimo galimybės naudojantis savybių modeliavimu;
2. Surinkti ICO išmanieji kontraktai bei ištirtos jų savybės;
3. Nustatyta, kurios savybės yra pastovios ir kintamos;
4. Kodo pernaudojamumui didinti sudarytas savybių modelis.

Darbo tikslas - ištirti pirminio finansavimo kriptovaliutomis išmaniuosius kontraktus, nustatyti, kokios savybės yra pastavios, o kokios - kintamos bei pasiūlyti būdus kodo pernaudojamumui didinti - pasiektas.

Pasiekti rezultatai leidžia daryti išvadas:

1. Savybių modeliavimas yra geras būdas ICO išmaniųjų kontraktų kodo pernaudojamumui didinti;
2. Yra sukurta pakankamai daug ICO išmaniųjų kontraktų turinčių įvairių savybių, todėl žinant, koks savybių rinkinys yra reikalingas, būtų galima pernaudoti esamą kontraktą ar jo dalį.

Darbą galima toliau tęsti šiomis kryptimis:

1. Būtų galima tobulinti savybių modelį paimant didesnę aibę ICO išmaniųjų kontraktų;
2. Būtų galima ICO savybių modelį panaudoti kuriant ICO išmaniųjų kontraktų produktų liniją;
3. Būtų galima panaudoti savybių modelį jau sukurtoms išmaniųjų kontraktų bibliotekoms tobulinti;
4. Būtų galima sukurti ICO išmanųjį kontraktą, kuris remtusi savybėmis iš modelio bei jau sukurtu kodu ir leistų jais lengvai varijuoti.

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.

[But14] Vitalik Buterin. A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM. 2014. URL: https://www.weusecoins.com/assets/pdf/library/Ethereum%7B%5C_%7Dwhite%7B%5C_%7Dpaper-a%7B%5C_%7Dnext%7B%5C_%7Dgeneration%7B%5C_%7Dsmart%7B%5C_%7Dcontract%7B%5C_%7Dand%7B%5C_%7Ddecentralized%7B%5C_%7Dapplication%7B%5C_%7Dplatform-vitalik-buterin.pdf.

[CG18] Christian Catalini ir Joshua S Gans. Initial Coin Offerings and the Value of Crypto Tokens, 2018. URL: http://creativecommons.org/licenses/by-nc/4.0/%20https://ssrn.com/abstract=3137213.

[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%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf.

[Dan17] Chris Dannen. Introducing Ethereum and Solidity Foundations of Cryptocurrency and Blockchain Programming for Beginners Introducing Ethereum and Solidity Foundations of Cryptocurrency and Blockchain Programming for Beginners Introducing Ethereum and Solidity: Foundation. *Library of Congress Control Number*, 2017. DOI: 10.1007/978-1-4842-2535-6. URL: http://smartcontracts.engineer/wp-content/uploads/2017/09/Etherium.pdf.

[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.

[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.

[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://www.comp.nus.edu.sg/%7E~%7Dloiluu/papers/oyente.pdf.

[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%7B%5C_%7DLee/publication/221553200%7B%5C_%7DConcepts%7B%5C_%7Dand%7B%5C_%7DGuidelines%7B%5C_%7Dof%7B%5C_%7DFeature%7B%5C_%7DModeling%7B%5C_%7Dfor%7B%5C_%7DProduct%7B%5C_%7DLine%7B%5C_%7DSoftware%7B%5C_%7DEngineering/links/558bdbde08ae591c19d8d3ce.pdf.

[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%7B%5C_%7DRavichandran/publication/220423696%7B%5C_%7DSoftware%7B%5C_%7Dreuse%7B%5C_%7Dstrategies%7B%5C_%7Dand%7B%5C_%7Dcomponent%7B%5C_%7Dmarkets/links/54201aa60cf2218008d43cdb.pdf.

[SVB01] Mikael Svahnberg, Jilles Van Gurp ir Jan Bosch. On the Notion of Variability in Software Product Lines, 2001. URL: www:%20http://www.ipd.hk-r.se/[msv%7B%5C%%7D7Cjvg%7B%5C%%7D7Cbosch].

[Swa15] Melanie Swan. *Blockchain. Blueprint for a new economy*. 2015, p. 149. ISBN: 978-1-491-92049-7. URL: http://w2.blockchain-tec.net/blockchain/blockchain-by-melanie-swan.pdf.

[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.

[The01] Mike Thelwall. A web crawler design for data mining. *Journal of Information Science*, 27(319), 2001. DOI: 10.1177/016555150102700503. URL: http://jis.sagepub.com/cgi/content/abstract/27/5/319.

Sąvokų apibrėžimai

Turing complete - bet kuri sistema, kuri yra pakankamai galinga atpažinti visus galimus algoritmus [Tel94].

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

Santrumpos

PLSE - produktų linijos programinė įranga (angl. product line software engineering)
ICO - pirminis kriptovaliutų platinimas (angl. initial coin offering)
FODA - Feature-Oriented Domain Analysis [KCH+90]