­

Eksperimentinės plėtros veiklos ataskaita

Projekto pavadinimas: Nurodyti

Projekto numeris: J05-LVPA-K-01-XXXX

Projekto vykdytojas: *UAB „Omnisend“*

Projekto partneriai: UAB „Tokenmill“

Veiklos numeris: Nurodyti

Veiklos pavadinimas: *Nurodyti*

Ataskaitos data: 2019-09-03

Lapų skaičius: Nurodyti

**Turinys**

[1. Veiklos tikslas 3](#__RefHeading___Toc879_3920256043)

[2. Veiklos užduotys 3](#__RefHeading___Toc881_3920256043)

[3. Bandomoji el-parduotuvė 3](#__RefHeading___Toc883_3920256043)

[3.1. Veiklos aprašas 3](#__RefHeading___Toc885_3920256043)

[3.2. Veiklos vykdymo eiga 4](#__RefHeading___Toc887_3920256043)

[3.2.1. Duomenų struktūros apibrėžimas 4](#__RefHeading___Toc889_3920256043)

[3.2.2. Shopify integracija 5](#__RefHeading___Toc891_3920256043)

[3.2.3. Veiklos Rezultatai 6](#__RefHeading___Toc893_3920256043)

[4. Abstrakčios prasmės reprezentavimo modelio plėtimas 7](#__RefHeading___Toc895_3920256043)

[4.1. Veiklos aprašas 7](#__RefHeading___Toc897_3920256043)

[4.2. Veiklos vykdymo eiga 7](#__RefHeading___Toc899_3920256043)

[4.3. Rezultatai 8](#__RefHeading___Toc901_3920256043)

[5. Teksto generavimo komponento vystymas 9](#__RefHeading___Toc903_3920256043)

[5.1. Veiklos aprašas 9](#__RefHeading___Toc905_3920256043)

[5.2. Veiklos vykdymo eiga 9](#__RefHeading___Toc907_3920256043)

[5.3. Rezultatai 10](#__RefHeading___Toc909_3920256043)

[6. Vartotojo sąsajos tobulinimas 10](#__RefHeading___Toc911_3920256043)

[6.1. Veiklos aprašas 10](#__RefHeading___Toc913_3920256043)

[6.2. Veiklos vykdymo eiga 10](#__RefHeading___Toc915_3920256043)

[6.3. Rezultatai 11](#__RefHeading___Toc917_3920256043)

[7. Įvykdyti paslaugų pirkimai 12](#__RefHeading___Toc919_3920256043)

[8. Užduočių atlikimas 12](#__RefHeading___Toc921_3920256043)

[9. Pasiekti rezultatai 12](#__RefHeading___Toc923_3920256043)

[10. Nepasiekti rezultatai 13](#__RefHeading___Toc925_3920256043)

[11. Veiklos vykdymo metu atliktų tyrimų eigos pakeitimai 13](#__RefHeading___Toc927_3920256043)

[12. Rezultatų naujumas ir nauda projektui 13](#__RefHeading___Toc929_3920256043)

[Bibliografija 14](#__RefHeading___Toc931_3920256043)

# Veiklos tikslas

Sukurti metodiką ir prototipą prekių, elektroninės prekybos sistemose, aprašymų generavimui, panaudojant natūralios kalbos generavimo (NLG) ir dirbtinio intelekto (AI) metodus.

# Veiklos užduotys

Šiame etape buvo apibrėžtos tokios veiklos užduotys:

* 1. Bandomasis diegimas testinėje el-komercijos platformoje. Buvo sukurta bandomoji el-parduotuvė Shopify.com platformoje, kurioje prekių aprašymas yra atliekamas naudojant mūsų teksto generavimo variklį.
  2. Abstrakčios prasmės reprezentavimo modelio plėtimas. Tęsiamas praeitų etapų darbas (2019-06-14 dienos ataskaita 5 skyrius [1]), aprašant galimas prasmės reprezentavimo struktūras.
  3. Teksto generavimo komponento tolimesnis vystymas ir taikymas.
  4. NLG vartotojo sąsajos tobulinimas.

# Bandomoji el-parduotuvė

## Veiklos aprašas

Kuriama teksto generavimo sistema turi integruotis su įvairiomis, turinį publikuojančiomis, platformomis. Dabartinis tikslas yra turėti sistemą veikiančią, bet neapsiribojančią, su pagrindinėmis el-komercijos platformomis. Bandomajam diegimui buvo pasirinkta *Shopify* platforma.

Teksto publikavimo integracijos įgyvendinimui reikalingi žingsniai:

1. *Duomenų sąsaja*. NLG sistema turi pasiimti pradinius, prekę aprašančius, duomenis.
2. *Teksto generavimas*. Prekę aprašančio teksto paruošimas, pagal pirmame žingsnyje gautus duomenis
3. *Teksto perdavimo sąsaja*. Sugeneruotas tekstas turi būti grąžinamas į el-parduotuvę.

Pirmojo ir trečiojo žingsnio vykdymo eiga aprašyta sekančiame skyriuje. Antrasis žingsnis čia neaprašomas, nes yra detaliai aptartas ankstesnėse ataskaitose.

## Veiklos vykdymo eiga

Tiek pradinių duomenų gavimas tiek jų grąžinimas į el-prekybos platformą yra atliekamas per GraphQL interfeisą aptartą 2019-06-14 ataskaitos *3.2 Technologiniai komponentai* skyriuje. Shopify platformos API taip pat yra įgyvendintas GraphQL protokolu. Tai palengvina integraciją su mūsų produktu bei patvirtina mūsų apsiriktos architektūros tinkamumą el-prekybos kontekste. Toliau GraphQL naudojimo specifika nebus aptariama, fokusuojantis į labiau funkcionalius tokios integracijos vykdymo eigos žingsnius.

### **Duomenų struktūros apibrėžimas**

Duomenų struktūra, su kuria yra dirbama yra labai paprasta. Ją galima atspindėti paprasta lentele, kurioje yra išvardinami visi prekės atributai. Prekę aprašantis tekstas yra toks pat atributas kaip ir kiti duomenys.

|  |  |  |  |
| --- | --- | --- | --- |
| **Title** | **Authors** | **Kiti atributai** | **Teksto variantai** |
| Building Search Applications | Manu Konchady | ... | Manu Konchady is the author of Building Search Applications with Lucene, LingPipe, and Gate… |
| Moving to the Cloud | Dinkar Sitaram, Geetha Manjunath | Moving to the Cloud watch with Developing Apps in the New World of Cloud Computing was written by Dinkar Sitaram, Geetha Manjunath… |
| Text Processing with GATE | Hamish Cunningham, Kalina Bontcheva, Diana Maynard | Text Processing with GATE watch with Version 6. Authors are Hamish Cunningham, Kalina Bontcheva, Diana Maynard... |

Lentelė 1 Knygos tipo prekių duomenų struktūros iliustracija

Duomenų apsikeitimo veiklos tikslas yra rast sprendimą, kuris leidžia maksimaliai lanksčiai keistis tokio tipo duomenimis. Todėl, NLG sistemoje tokios, prekių aprašymo lentelės yra konvertuojamos į įrašų masyvus, kurie saugo duomenis tokiu formatu:

**Eilutė**, turi daug **Laukų**, kiekvienas laukas sudarytas iš **Vardo** ir **Reikšmės** elementų. Formaliau (JSON formatu):

[

[

{ "field-name" : "title", "field-value" : "Building Search Applications },

{ "field-name" : "authors", "field-value" : "Manu Konchady" }, …

]

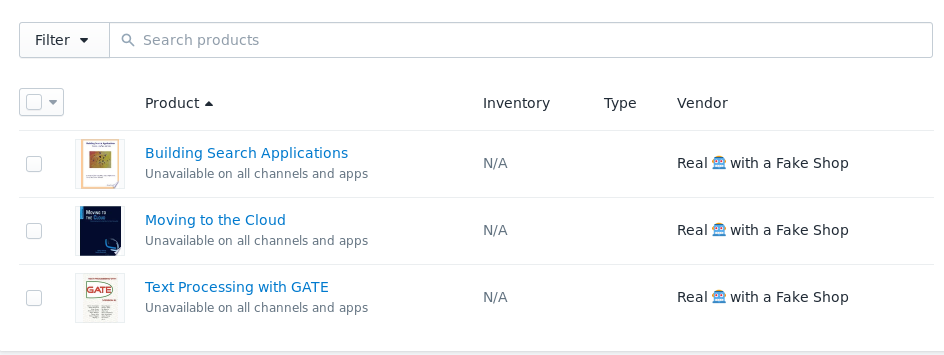
…

]

Tokiu būdu mes galime lanksčiai integruotis su el-komercijos sistemomis, kurios skirtingai reprezentuoja prekių duomenis.

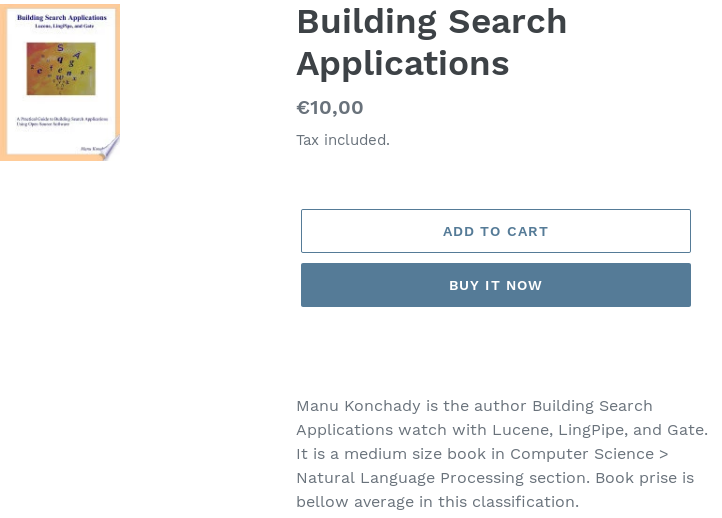
### **Shopify integracija**

Shopify, kaip ir kitų panašių el-prekybos platformų, darbo procesas susijęs su prekių įkėlimu vyksta per prekių inventoriaus valdymo darbą. Todėl aprašančio teksto generavimo sistema neturėtų, keisti tokio darbo įpročių ir nusistovėjusių procesų.



*Pav 1. Prekių inventorius Shopify parduotuvėje. Inventoriaus sąrašas yra integruotas su NLG sistema.*

Mūsų diegiama NLG sistema gali gauti pranešimus apie prekių pakeitimus inventoriaus lentelėje ir papildyti prekės duomenis, prekės automatinio aprašo lauku. Naudojant mūsų sistemos API susiejimą su Shopify API [2], mes galime visiškai automatizuoti tokį procesą ir atsidarius prekės langą rodyti mūsų sugeneruotą aprašą.



Pav 2. Prekės pardavimo langas Shopify el-parduotuvėje, su automatiškai sugeneruotu aprašo tekstu.

### **Veiklos Rezultatai**

|  |  |  |
| --- | --- | --- |
| **Tema** | **Rezultatai** | **Iššūkiai** |
| Integracijos duomenų struktūros | a) GraphQL leidžia lankstų integracinio API apibrėžimą. Tai yra populiarus sąsajų apibrėžimo formatas, kas leidžia tikėtis, kad daugiau duomenų platformų bus įgyvendinta šiuo būdu. Tai įgalina ir platesnį mūsų NLG sistemos diegimą.  b) Pačių duomenų dekomponavimas į atskirų laikų masyvus, leidžia neapibrėžti fiksuotos duomenų schemos, kas savo ruožtu nereikalauja, kad NLG naudojanti sistema būtų keičiama. | a) Integracija visada bus tik dalina, negalime turėti sistemos, kuri be pakeitimų integruojasi į bet kokią išorinę platformą. |
| Shopify integracija | a) Pilna integracija per prekių inventoriaus pasikeitimų stebėjimą  b) Generuojami tekstai automatiškai įkeliami į el-parduotuvę. | Iššūkių nėra, iš dalies todėl, kad mes patys pasirinkome Schopify integraciją. Pats Shopify turi itin išvystytą integracijos platformą. |

# Abstrakčios prasmės reprezentavimo modelio plėtimas

## Veiklos aprašas

Abstrakčios prasmės reprezentavimo (AMR, angl: *abstract meaning representation*) modelio kūrimas ir jo integracija su gramatikos generatoriumi buvo baigti praeitų etapų metu. Šiame etape, naudojantis AMR struktūromis aprašytomis VerbNet projekte [3], buvo analizuojamos pasikartojančios AMR sintaksės struktūros. Pasikartojančios struktūros, turinčios skirtingas abstrakčias reikšmes, turi vieną implementaciją AMR modelyje. Tokiu būtu mes galime pasiekti platų, skirtingų prasmių realizavimą, naudojantis siauru AMR modeliu.

Imkime du, VerbNete aprašytus, veiksmažodžius: *cut* ir *bend*. Šie žodžiai yra dviejų, visiškai skirtingų prasmės pranešimų, pagrindas. Tačiau abu turi tą pačią sintaksės struktūrą:

|  |  |  |
| --- | --- | --- |
| **NOUNT** | **VERB** | **NOUN** |
| Carol | **cut** | the bread |
| Tony | **bent** | the rod |

Tokiu būdu mūsų AMR modelyje mes galime turėti vieną NOUN-VERB-NOUN abstrakčią struktūrą, kuri gali realizuoti daugybę skirtingų konkrečių AMR implementacijų.

Antra, panaši, šio komponento veiklos užduotis yra panaudoti VerbNete naudojamus AMR sudarančių elementų ribojimo dėsningumų aptikimas ir aprašymas. Aukščiau pateiktame *cut* ir *bent* pavyzdyje, AMR aprašymai gali papildomai nurodyti daiktavardžių[[1]](#footnote-2) panaudojimo ribojimus. Tarkime *cut* AMR gali reikalauti kad antrasis daiktavardis yra būtinai maisto produktas. Abu *cut* ir *bent* AMRai reikalauti, kad pirmasis daiktavardis yra asmenvardis.

## Veiklos vykdymo eiga

Veikla vykdyta dviem žingsniais. Pirmasis yra identifikuoti AMR sintaksės ir AMR apribojimų dėsningumus. Viso identifikuota 37 AMR sintaksės dėsningumai. Keli pavyzdžiai:

LEX LEX VERB NP

LEX VERB NP

NP VERB

NP VERB ADJ

NP VERB ADJ PREP NP

NP VERB NP NP NP

PREP NP LEX VERB NP

PREP NP VERB NP

Šios sekos atspindi kalbos dalių vartojimą, išreiškiant pasirinktą prasmės reprezentaciją

AMR ribojimų dėsningumų yra daugiau – 113. Keletas pavyzdžių:

ADJ None

ADV Trajectory

ADV None

NP Beneficiary

NP Causer

NP Co-Agent

NP Destination

NP Duration

NP Experiencer

NP Extent

NP Goal

NP Precondition

PREP against on

PREP against

PREP as

PREP at for

Skirtingai nuo sintaksės dėsningumų, apribojimų dėsningumai kartais turi ir fiksuotus naudotinus žodžius, kas daro jų platesnį panaudojimą mažiau efektyvų.

AMR ribojimų įgyvendinimo metu identifikuotas poreikis išplėsti šiuo metu turimą vartotojo sąsają. Šalia plano, žodyno ir duomenų redaktoriaus reikia įvesti ir ribojimų ontologijos redaktorių. T.y. vietą kur vartotojas galėtų apibrėžti pagrindinius veiklos srities esinius ir jų ryšius. *Cut* ir *Bend* pavyzdyje matėme, kad reikia operuoti tokiais konceptais kaip: asmuo, maisto produktas ir panašiai. Kol kas šis pakeitimas nebus įgyvendinamas, kuriamam prototipui tokia ontologija, minimaliame variante, gali būti išvesta iš el-parduotuvės inventoriaus duomenų bazės.

Žodynų valdymo - jau praeituose etapuose įgyvendintas komponentas – taip pat turėjo būti patobulintas, tam, kad įgalintų AMR veikimą. Šiame etape mes integravome WordNet [7] žodyną iš kurio galime gauti įvarių žodžių kalbos dalis bei sinonimų rinkinius. WordNet yra bendrinis žodynas ir nėra visiškai tinkamas specifinėms veiklos sritims, kaip el-komercijos tekstų generavimas. Tačiau šiame projekto vystymo etape, mums nebūtina optimizuoti sistemos siaurose veiklos srityse.

## Rezultatai

|  |  |  |
| --- | --- | --- |
| **Tema** | **Rezultatai** | **Iššūkiai** |
| AMR sintaksės dėsningumai | a) Identifikuoti 37 dėsningumai  b) 12 dėsningumų jau yra palaikomi teksto generavimo sistemoje | a) Dalis VerbNet tyrėjų aprašytų AMRų neturi gramatinės sintaksės, naudojamos nelanksčios tekstinės citatos. Šios struktūros turės būti konvertuojamos į taisyklingas gramatikas. |
| AMR ribojimų dėsningumai | a) Identifikuoti 113 dėsningumai.  b) Paruošta sistemos architektūra jų palaikymi | a) VerbNet aprašo per daug ir per bendrus ribojimus, mums reikia aprašyti su el-parduotuvės veikla susijusius ribojimus.  b) AMR ribojimų valdymas turi būti įgyvendintas vartotojui suprantamu ir lengvai operuojamu būdu. |
| WordNet žodyno integravimas | a) Integruotas AMR funkcinis ir WordNet žodynas | a) WordNet nėra veiklos srities žodynas, jo siaurinimaui iki veiklos specifikos reikia įgyvendinti, glaimai, žodžių vektoriais [8] paremtą sinonimų generatorių. |

# Teksto generavimo komponento vystymas

## Veiklos aprašas

Šiame etape teksto generavimo komponentas nebuvo plačiamas naujomis funkcijomis. Visi darbai buvo susiję su pakeitimais, kurių reikalavo darbai susiję su AMR įgyvendinimu.

Buvo suplanuota ir įgyvendinta darbą su CCG gramatika [4] lengvinanti specifinė veiklos srities kalba (angl: *Domain Specific Language*) [5]

## Veiklos vykdymo eiga

AMR įgyvendinimas, CCG gramatika paremtu teksto generavimu, reikalauja vis sudėtingesnių gramatinių struktūrų įgyvendinimo. Tam sukurta ir projekto reikmėms naudojama DSL kalba.

Gramatikos aprašymo DSL pavyzdys:

Tarkime reikia aprašyti šią CCG struktūrą (minimali veiksmažodinė frazė):

S\NP/NP

Mūsų naudojamas OpenCCG [4] šią gramatiką aprašo arba per sudėtingą XML struktūrą, arba per taip pat sudėtingą Java objektų konstravimo programą. Mūsų DSL leidžia tai aprašyti pakankamai lengvai, naudojant mums įprasta Lisp S-ekspresijas [6]:

(>F (<B NP NP))

kur (naudojant Lamda calculus ir CCG terminologiją)

*>F* – Forward composition

*<B* – Backward composition

*NP* - daiktavardis

## Rezultatai

Ši veikla nebuvo orientuota į naujų rezultatų pasiekimą. Buvo siekiama toliau vystyti esamą komponentą ir taikyti jį visos sistemoms reikmėms.

Aukščiau trumpai aptartas DSL, ateityje galėtų tapti nepriklausoma nuo mūsų produkto CCG gramatikas aprašančia kalba.

# Vartotojo sąsajos tobulinimas

## Veiklos aprašas

Kaip buvo planuota ir įgyvendinta prieš tai buvusiuose etapuose, dokumento planavimo įrankis veikia, kaip programavimo aplinka (angl: *Integrated Development Environment*). Kur vietoje įprastinių programavimo kalbų, yra kuriamas teksto realizavimo planas. Tam, kad toliau vystyti šią *porgramavimo* paradigmą, mes įgyvendiname eilę patobulinimų skirtų darbo optimizavimui.

Pati aplinka pradėta testuoti kuriant konkrečius el-komercijos srities tekstus. Šie planai yra naudojami aukščiau aprašytoje Spotify parduotuvėje (3 Bandomoji el-parduotuvė skyrius).

## Veiklos vykdymo eiga

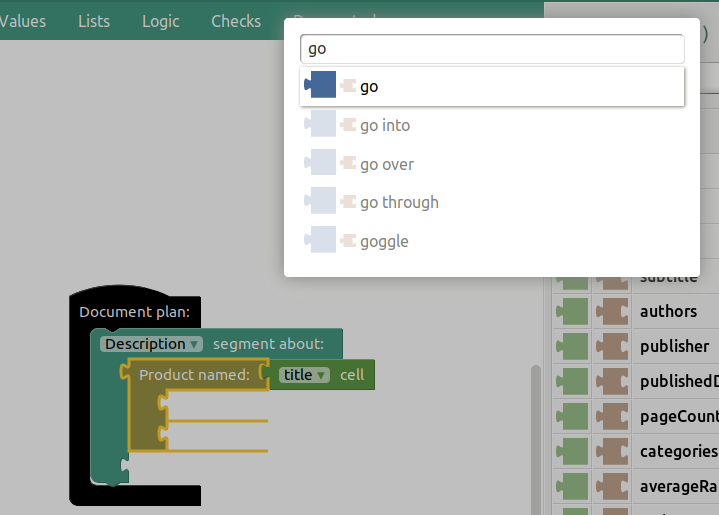
Dokumento plano konstravimo strategija remiasi AMR pricipu, kur pranešimo prasmė dažniausiai konstruojama per veiksmažodį. Tai nėra intuityvu sistemos naudotojams. Žmonės konstruoja sakinius pradėdami ir galvodami apie daiktavardį – t. y. tai kas atlieka veiksmą:

*Atėnai kariavo su Sparta*

O ne:

*Kariavo: Atėnai, Sprata.*

Tačiau būtent antrasis – funkcinis – būdas yra naudojamas AMR’e (taip pat ir CCG). Tam, kad vartotojui palengvinti darba su tokiomis AMR struktūromis, mes įgyvendinome, greitos kontekstinės paieškos sistemą, kuri liedžia greitai rasti norimus AMR pranešimo elementus.

  
Pav 1: Kontekstinė paieška

Paveiksle parodyta kontekstinė paieška geba parinkti reikiamos kalbos dalies žodžius ir kaip konceptus automatiškai įterpti į reikiamą dokumento plano vietą.

## Rezultatai

|  |  |  |
| --- | --- | --- |
| **Tema** | **Rezultatai** | **Iššūkiai** |
| Kontekstinė lingvistinių pranešimų paieška | a) Įgyvendinta kontekstinė paieška leidžianti leigviausurasti norimas kalbines struktūras. | a) Vartotojo sąsajoje mes stengiamės visiškai išvengti gramatinių konceptų (kalbos dalis, artikeliai anglų kalbai, skiryba ir panašiai). Tačiau kontekstinė paieška turi remtis bent jau kalbos dalimi. Sekančiame etape bus ieškoma, kaip išvengti šios priklausomybės. |
| Kiti smulkūs patobulinimai | a) Klaviatūros komandų įvedimas  b) Projekto dokumentavimas ir paruošimas galimam publikavimui atviro kodo repositorijoje. |  |

# Įvykdyti paslaugų pirkimai

Amazon AWS serveriai ir paslaugos (dalis paslaugų šiuo metu neviršija nemokamo naudojimo limito):

1. *Amazon Lambda* – paslaugos ir NLG operacijos
2. *Amazon ES* – tekstynų saugojimas
3. *Amazon VCP* – saugumo infrastruktūra
4. *Amazon DinamoDB* – darbinių duomenų saugykla

# Užduočių atlikimas

Pateikti informaciją, kokios užduotys buvo atliktos projektą vykdančių asmenų, kas jas atliko, kiek laiko tam skyrė, naudoti lentelės formą.

| **Nr.** | **Užduotis** | **Darbuotojai/darbovietė** | **Laiko sąnaudos valandomis** |
| --- | --- | --- | --- |
| 1. | CCG analizė. Pradinių modelių įgyvendinimas | Žygimantas Medelis, UAB “TokenMill” | 168 |
| 2. | NLG sistemos architektūra ir implementacija. | Dainius Jocas UAB “TokenMill” | 431 |
| 3. | CCG gramatikos ir dokumento plano susiejimas ir konvertavimas | Šarūnas Navickas UAB “TokenMill” | 446 |
| 4. | Dokumento plano redaktorius | Tomas Čerkaukas UAB “TokenMill” | 447 |
| 5. | Kiti NLG tyrimai | Žygimantas Medelis, UAB “TokenMill” | 80 |

# Pasiekti rezultatai

Kuriamą produktą galima suprasti kaip priemonę kuri atlieka dvi funkcijas:

1. Suteikia redaktoriui priemones apibrėžti komunikacijos tikslus ir planą. Šie turi būti apibrėžti taip, kad nekiltų pranešimo prasmės dviprasmybių ar kitų netikslumų.
2. Remiantis sukurta pranešimo prasmės struktūra, generuoti rišlius teksto variantus.

Šios abi dalys siejasi per AMR (abstrakčios prasmės reprezentaciją, *angl: abstract meaning representation*) konceptą. Jei AMR naudojimo ir jo konvertavimo į gramatiką ir iš ten į tekstą, aspektai, jau buvo ištirti ir įgyvendinti praeituose etapuose, tai šiame etape buvo išgryninti šios dvi AMR koncepto naudojimo produkte pusės: *vartotojui* – apibrėžti prasmę ir *NLG generatoriui* – sukurti tekstą.

Vartotojo lygyje AMR toliau vystomas pagal vizualios programavimo kalbos redaktoriaus paradigmą aprašytą pirmojoje 2018-12 ataskaitoje skyriųje *3. Automatizuotas dokumento plano redaktorius*  [10]. Šiame etape išplečiant šią paradigma kontekstinės paieškos komponentu, kuris įgalina programavimo principus naudoti ligvistiniame kontekste.

AMR naudojimas teksto generavimo pusėje toliau vystomas pagal numatytą architektūrą, atrandant ir įgyvendinant reikiamas, prasmės reprezentavimo konvertavimo į gramatiką, struktūras.

# Nepasiekti rezultatai

Visi numatyti rezultatai buvo pasiekti.

# Veiklos vykdymo metu atliktų tyrimų eigos pakeitimai

Remiantis praeito etapo tyrimu, buvo planuojama, kad dokumento plane mes galėsime remtis visiškai nuo gramatinės specifikos suformuluoti prasmės pranešimu. Kaip antai:

*He* ***worked*** *for two hours.*

(w / work-01

:ARG0 (h / he)

:duration (t / temporal-quantity

:quant 2

:unit (h2 / hour)))

Tačiau šio etapo metu paaiškėjo, kad gramatinis šablonas, bent kol kas, yra būtinas ir aukščiau pateiktą AMR pavyzdį reikia papildyti sintaksės šalonu, kurį galima gauti per VerbNet’ą [9].

Šis pakeitimas reikalauja papildomo, duomenų apdorojimo žingsnio, sistemos veikime, tačiau nekeičia jos konstrukcijos ir veikimo principų.

# Rezultatų naujumas ir nauda projektui

Tolimesni tyrimai nebeatliekami, todėl šio etapo rezultatai nėra nukreipti į naujų teorijos taikymų atradimą. Šiame ir kituose etapuose yra įgyvendinama ir tobulinami ankstesnių tyrimų rezultatai.

Ankstesnių rezultatų ir jų taikymo šiame etape, nauda projektui:

1. Palaikomų AMR ir gramatikos konstrukcijų plėtimas, šio etapo metu, leido mums sukurti bandomąją parduotuvę.
2. Dokumento plano redaktoriaus plėtimas leidžia paprasčiau apibrėžti dokumento planą. Dabartinis redaktorius yra pilnai fukcionalus kurti įvariausiams tekstų tipams.

# Bibliografija

1. 2019-06-14 dienos projekto ataskaita
2. *GraphQL examples for products* (Skaityta 2019-09-03) <https://help.shopify.com/en/api/graphql-admin-api/examples/product>
3. *A Class-Based Verb Lexicon* (Skaityta 2019-09-04) <https://verbs.colorado.edu/~mpalmer/projects/verbnet.html>
4. 20190313 dienos ataskaita
5. *Domain Specific Languages*, Martin Fowler <https://www.martinfowler.com/books/dsl.html>
6. *S-expression* (Skaityta 2019-09-05) <https://en.wikipedia.org/wiki/S-expression>
7. *WordNet* (Skaityta 2019-09-05) <https://wordnet.princeton.edu/>
8. *Distributed Representations of Words and Phrases and their Compositionality,* Mikolov T. et al, (2013) <https://arxiv.org/abs/1310.4546>
9. *VerbNet* (Skaityta 2019-09-05) <http://verbs.colorado.edu/~mpalmer/projects/verbnet.html>
10. 2018-12-19 ataskaita

1. Nebūtinai daiktavardžiai, bet aprašymo paprastinimui naudojam tik juos. [↑](#footnote-ref-2)