**RĪGAS TEHNISKĀ UNIVERSITĀTE**

Datorzinātnes, informācijas tehnoloģijas un enerģētikas fakultāte

**Gļebs Bondarevs**

bakalaura akadēmisko studiju programmas “Datorsistēmas”

students, stud. apl. nr. 211RDB375

**Automatizētas dokumentācijas ģenerēšanas ietekme izpēte uz izstrādes procesu**

**BAKALAURA DARBS:** **rudens semestra atskaite**

Zinātniskais vadītājs: Dr.sc.ing , Rīgas Tehniskās universitātes as.prof

Aleksejs Jurenoks

RĪGA 2024

Anotācija

Šis dokuments ir jāizmanto, kā ieteicamā veidne bakalaura darba rudens semestra atskaites noformēšanā. Visi dokumenta elementi, kas ir norādīti starp <> simboliem, ir jāaizstāj ar bakalaura darba autora datiem, piemēram, autora vārdu, tēmu vai konkrētas darba nodaļas nosaukumu. Bakalaura darba atskaites autors var brīvi papildināt atskaites struktūru ar sev nepieciešamajām nodaļām. *Slīprakstā* esošais teksts ir informatīvs un tas ir jādzēš ārā.

Satura rādītājs

Saturs

[Anotācija 2](#_Toc155539177)

[Satura rādītājs 3](#_Toc155539178)

[1. Darba būtība un aktualitāte 4](#_Toc155539179)

[2. informācijas avotu apstrādes rezultāti 8](#_Toc155539180)

[2.1. Zinātnisko rakstu meklēšana un atlasīšana 8](#_Toc155539181)

[2.2. Atslēgvārdu izvēle 8](#_Toc155539182)

[2.3. Zinātnisko rakstu izvēle 8](#_Toc155539183)

[2.4. Tēmas un pētījumu aktualitāte 9](#_Toc155539184)

[2.5. Hipotēze 9](#_Toc155539185)

[2.6. Ievads avotu apstrāde 9](#_Toc155539186)

[2.7. Pirmkoda dokumentācija 11](#_Toc155539187)

[2.8. Programmētāji 12](#_Toc155539188)

[2.9. Atslēgvārdi 13](#_Toc155539189)

[2.10. Dokumentācijas rīki 13](#_Toc155539190)

[2.11. Secinājumi 15](#_Toc155539191)

[Izmantotie informācijas avoti 16](#_Toc155539192)

1. Darba būtība un aktualitāte

**Bakalaura darba mērķis:**

Izpētīt un izanalizēt automatizētas dokumentācijas ģenerēšanas ietekmi uz programmatūras izstrādes procesu, ar nolūku identificēt priekšrocības un trūkumus, kas saistīti ar šo tehnoloģiju pielietojumu.

**Bakalaura darba uzdevumi:**

**1.** Izpētīt esošos risinājumus un tehnoloģijas, kas paredzētas dokumentācijas ģenerēšanai, un identificēt to funkcijas, pielietojuma jomas un priekšrocības izstrādes procesā.

**2**. Veikt literatūras apskatu par dokumentācijas nozīmi izstrādes procesā un to, kā automatizētā dokumentācija varētu uzlabot efektivitāti un kvalitāti.

**3.** Definēt programmatūras dokumentācijas ietekmes novērtēšanas kritērijus, kas nosaka, kā izmaiņas dokumentācijā ietekmē izstrādes procesa efektivitāti un kvalitāti.

**4.** Izstrādāt automātiskās dokumentācijas ģenerācijas izstrādes metodoloģiju un pielāgot to konkrētai izstrādes vides vajadzībām.

**5.** Veikt praktisko validāciju, pielietojot izvēlēto dokumentācijas ģenerēšanas risinājumu reālajā izstrādes projektā.

**6.** Salīdzināt rezultātus pirms un pēc automatizētas dokumentācijas ģenerēšanas risinājuma pielietošanas, izmantojot kvantitatīvus rādītājus, lai noteiktu ietekmi uz izstrādes procesu.

**7.** Balstoties uz eksperimenta rezultātiem sniegt secinājumus par automatizētas dokumentācijas ģenerēšanas ietekmi uz izstrādes procesu.

**Problēmas nostādne:**

Informācijas tehnoloģijas jomā pēdējos gados notikušas dažādas izmaiņas, viens no tiem ir dokumentācijas izveide. Dokumentācijas izveide no cilvēkā prasa daudz pūles un laika lai izveidotu to, bet izmantojot automātisku dokumentāciju tas samazinā projekta, uzturēšanas un atjaunināšanas procesu izmaksas.[10]. Izmantojot pārbaudes un tēstus reālus laika apstākļos var secināt kā dokumentācija kura ir uzrakstīta manuālī ir nepilnīga, prasa ilgu laiku un tas regulāri prasa atjaunināšanu.[9]. Automātiskā dokumenta ģenerēšanā mūsdienu ir priekšroka, jo cilvēkam piemīt īpašība kļūdieties, līdz ar to dokumentācijas laika programmētājs var pārkāpt dažos dokumentācijas standartus.[8].

Neskatoties uz to, ka automātiskā dokumentācijas ģenerēšanā jau gadu gadiem ir aktuāla, joprojām neeksistē sistēma kura strādā tieši bez cilvēka palīdzības. Tas ir galvenā problēma šādai tehnoloģijai. Kā piemēru, var minēt: DoxyGen un JavaDoc. [9]. Darba gaitā mēs plānojam identificēt kādas ir priekšrocības un trūkumi, kas saistītas ar šo tehnoloģijas izmantošanu, kā ari sagaidam, ka šādai tehnoloģijai būs vairāk priekšrocības nekā trūkumus.

**Tēmas aktualitātes pamatojums:**

Nav noslēpums kā uz šodienas dienu visas tehnoloģijas strauji attīstās, kā ari pieaug vajadzība pēc dažādu rīku vai tehnoloģijas automatizāciju, lai paātrinātu darbošanās vai izstrādes laiku. Automatizētas dokumentācijas ģenerēšanas ietekme izpēte uz izstrādes procesu šodienas ir ļoti būtiska un aktuālā tēma. Turpmākajā teksta ir tieši aprakstīts kāpēc šī tēma ir aktuāla un kāpēc tas ir interesanti izpētīt ietekmi uz to, dotajā momenta.

Pirmkārt, dokumenta automatizācijas ģenerēšana ir ļoti aktuāla priekš laika ekonomijas un izstrādes efektivitāte. Izmantojot automatizēta dokumenta ģenerēšanu programmētājs vai produkta izstrādātājs var ietaupīt daudz laika uz produkta izstrādājumu, nerakstot dokumentu, rediģējot to un nelabojot gramatiskas kļūdas kas bija veidotas rakstīšanas laika. Izstrādātajam kopa ar automatizēta dokumenta ģenerēšana palīdzību ari ir iespēja viegli un ļoti ātri veikt rediģēšanas darbības, kurus pilnīgi atspoguļos visas izmaiņas produktā, kodā un projektā kopumā.

Otrkārt, kā bija minēts iepriekš, izmantojot dokumenta automatizācijas ģenerēšanu var bez jebkādiem kļūdām vai problēmām tieši sekot dokumenta standartam dokumenta izstrādes procesā. Noteikts standarts var būt izmantots visam produktiem izstrādātam uzņēmuma ietvaros, kā ari šāds uzģenerēts dokuments būs viegli saprotams visiem programmētajam. Standartizēts dokuments gan paātrinās produkta izstrādi, gan atbrīvos programmētājus no liekam darbībām un atvieglos dzīvi izstrādātajam.

Treškārt, izmantojot dokumenta ģenerēšanu var nodrošināt dokumentāciju izveidošanu dažādas valodas, lai visi projekta dalībnieki kas piedalās produkta izstrāde varētu saprast visu kas ir minēts dokumentā. Tas ne tikai paātrinās produkta izstrādi bet ari atbrīvos izstrādātajā no dokumenta tulkošanai.

Beigas var secināt, kā tēma par automatizētas dokumentācijas ģenerēšanu ietekmi uz izstrādes procesu ir aktuāla mūsdienas. Tas var uzlabot izstrādes procesu, uzlabot produkta kvalitāti, kas iekļaus sevi pašu produktu un dokumentāciju, un paātrinās tas izstrādes procesu.

1.1. tabula

Bakalaura darba izstrādes plāns

|  |  |
| --- | --- |
| Aktivitāte | Termiņš |
| Darba būtība un aktualitātes uzrakstīšana | 24.12.2023 |
| Savākt informācijas avotus | 24.12.2023 |
| Analizēt informācijas avotus | 28.12.2023 |
| Uzrakstīt tēmas aktualitātes pamatojumu | 29.12.2023 |
| Uzrakstīt informācijas avotu apstrādes rezultātu | 07.01.2024 |

2. informācijas avotu apstrādes rezultāti

Priekš bakalaura darba bija izvelēta tēma ar nosaukumu “Automatizētas dokumentācijas ģenerēšanas ietekme izpēte uz izstrādes procesu”. Lai iepazīties un saprast par ko tieši ir saistīta izvelēta tēma ir jāizanalizē informācijas avotus, kā ari turpmāk jāsagatavo rudens atkaiti, kas ietver sevi informācijas avotu apstrādes rezultātu izvelētai tēmai.

2.1. Zinātnisko rakstu meklēšana un atlasīšana

Lai iepazīties un uzrakstīt informācijas avotu apstrādes rezultātu bakalaura darba tēmai bija izvelēti un izlasīti 5 dažādi elektroniski informācijas avoti. Zinātnisko rakstu meklēšana un piekļuve pie tā bija nodrošinātā izmantojot Rīgas Tehniskās universitātes bibliotēkas e-resursus. Rakstu meklēšanā bija veidotā izmantojot elektroniskos resursus, tādus kā: “IEEE Xplore Digital Library”, “SpringerLink”, “Web of Science”, “ScienceDirect”, “ACM Digital Library” un “WILEY Online Library”.

2.2. Atslēgvārdu izvēle

Resursu meklēšanai bija izmantoti atslēgvārdi lai izvairīties no liekiem zinātniskiem rakstiem, kas nav saistīti ar izvelētu bakalaura darba tēmu, kā ari atvieglot meklēšanas procedūru. Tā kā bakalaura tēma ir saistīta ar automatizētas dokumentācijas ģenerēšanu un tas ietekme uz izstrādes procesu bija izvelēti pamata atslēgvārdi, tajos ietilpst tādi vārdi kā: automatic, automated, document, documentation, generation, process, development. Atslēgvārdi bija izvelēti tikai tie kas tieši ir saistīti ar bakalaura tēmu, jo automatizētas dokumentācijas ģenerēšanai ir dažādi virzienu līdz ar to daži zinātniskie raksti nav saistīti ar izstrādes procesu.

2.3. Zinātnisko rakstu izvēle

Izmantojot elektroniskus resursus un izmantojot atslēgvārdu zinātnisko rakstu meklēšanai bija izvelēti 5 dažādie zinātniskie raksti, kas ir saistīti ar dažādiem tēmām. Rakstu izvēle bija saistīta ar citātu skaitu, žurnāla indeksu un autora pieejamību, kā arī autora vai publicista reputāciju. Dažiem publicētiem rakstām bija kā daļā no liela zinātniskā raksta, viena no tām ir izlasīts zinātnisks raksts ar nosakāmu “Automatic Documentation Generation via Source Code Summarization” [5], kas bija paņemts no “IEEE International Conference on Software Engineering”. Izvēle bija atkarīga ne tikai no resursa indeksiem, citātu skaitu un autora pieejamību bet arī no raksta konteksta, kas ir svarīgs punkts pareizai zinātnisko rakstu izvēlei.

2.4. Tēmas un pētījumu aktualitāte

Pilno tēmas un pētījumu aktualitāte var atrast pirmā sadaļa “Tēmas aktualitātes pamatojums”. Bet runājot īsi par to, var izcelt 3 galvenos iemeslus kāpēc pētījums ir aktuāls un pirmkārt tāpēc kā, tas ekonome izstrādātajā laiku un palielina programmatūras izstrādes efektivitāte. Otrkārt, izmantojot dokumenta automatizācijas ģenerēšanu var bez jebkādiem kļūdām vai problēmām tieši sekot dokumenta standartam dokumenta izstrādes procesā. Treškārt, izmantojot dokumenta ģenerēšanu var nodrošināt dokumentāciju izveidošanu dažādas valodas. Tēma paliek aktuāla, jo pagaidām nav standartizēta un viena risinājuma dokumenta automatizācijas ģenerēšanai, kā ari nav īsti izpētīts tas ietekme uz izstrādes procesu.

2.5. Hipotēze

Pirms lasot un analizējot zinātnisko rakstu avotus ir vispirms jānotiek kāda ir hipotēzē par izvelētu bakalauru tēmu. Hipotēze par automatizētas dokumentācijas ģenerēšanu ir saistīta ar to, kā ietekme uz izstrādes procesu ir pozitīva un programmētāji, lietotāji un uzņēmumi gūs tikai priekšrocības izmantojot to. Tas uzlabos dokumenta efektivitāti un kvalitāti. Protams analizējot rakstus var būt atradīti dažādi trūkumi izmantojot to.

2.6. Ievads avotu apstrāde

Tas nav noslēpums ka mūsdienas dokumentācija spēlē svarīgu lomu dažādas jomas, kā ari tās palīdz darbiniekiem darbošanās laika. Pie šādiem jomam ari attiecas IT jomā. IT izmanto dokumentāciju katrā izstrādes posma, lai nodrošinātu izstrādes procesu drošu, efektīvu un ātru. Kā piemēru kuru izmanto sistēmas vai programmatūras izstrādes procesa kuru izmanto sākotnējā fāze var minēt PPS – Programmas prasību specifikācija.

Dokumentācijas uzrakstīšana, izgatavošana, un uzturēšana ir dārga, un tas bieži vien laika gaitā dēļ noveco. [3]. Dokumentācijas laika gaita kļūst novecojuši, tas notiek tāpēc kā programmatūra patstāvīgi mainās izstrādes vai uzturēšana laika.[4][5]. Dokumentācijā ir ideāls sabiedrotais jebkura programmatūras sistēmai, lai ieinteresētajam personām būtu piekļuvē pie noderīgai informācijai par sistēmu un tas saistītam funkcijām [6]. Kā ari dokumentācijas rakstīšana ir sarežģīta, jo projekta programmētājiem ir jāsniedz zināšanas citiem programmētajiem kas piedalās projekta ietvaros, piedalās sistēmas izstrādāšanai vai tā uzturēšanai.[3]. Līdz ar to pēdējā laika ir palielinājušies mēģinājumi izveidot standartizētu automātisko dokumentācijas ģeneratoru.[4]. Neizmantojot automātisko dokumentācijas ģeneratoru lietotājiem vai izstrādātajam ir jāsamontē dažādas utilītos lai iegūtu nepieciešamo inženiertehnisko informāciju un izveidotu labu dokumentāciju.[2]. Izmantojot šādas utilītos vai citas sistēmas var radīties kā dažām dokumentu publicēšanas sistēmām izstrādātājiem ir jāsamontē dokumentu, izmantojot izvēlnes, lai no repozitorijos bija iespējams izvilkt nelielus inženiertehniskās informācijas avotus vai fragmentus.[2]. Lai saprast kas ir dokumentu automatizācijas ģenerēšana vispirms vajag uzzināt kas ir slikta un labā dokumentācijas. Laba dokumentācija kura ir uzrakstīta pareizi, sekojot visiem standartiem, protokoliem un metodēm, ir ļoti svarīga efektīvai un ātrai programmatūras izrādei.[5]. Savukārt slikta programmatūras dokumentācija var izraisīt programmatūras sistēmas strauju degradāciju kvalitāte un tas lietojamības.[3]

Kā bija minēts iepriekš, programmatūras dokumentācijā nodrošina izstrādes procesu drošu, efektīvu un ātru, kā ari dokumentācijā pastāv, lai palīdzētu programmatūras projektētajiem, izstrādātajiem un uzturētājiem izprast sistēmu un tās gan galvenos, gan sekundāros procesus. [3]. Bet runājot par procesa automatizāciju, lai nodrošināt dokumentācijas ģenerēšanu automatizēšanu ir jāizmanto noteiktas veida izstrādes metodes, rīkus kas ievieš šo metodi un tas dokumentācijas standartus[2], kurus dažreiz cilvēks var neizmantot dokumentācijas uzrakstīšanas laika neuzmanības vai aizmiršanas dēļ. Tas var minēt kā pirmo noderīgo trūkumu automātiskas ģeneratora izmantošanai dokumentācijai. Izstrādājot dokumentāciju priekš sistēmai tas vajadzētu būt drošai, labai un efektīvai, lai katram izstrādās dalībniekam būtu pilnīga sapratnē par izstrādājamo sistēmu un tas funkcionalitāte. Lai iegūt efektīvu dokumentu izgatavošanas automatizāciju tas vajadzētu sekot stingri noteiktai un detalizētai izstrādes procesa ievērošanai. [2].

Protams, runājot par automatizācijas procesu nevar aizmirst kā bez noteiktai informācijai nevar izveidot labu un efektīvu dokumentu automatizāciju. Pie šādai informācijai attiecas 4 galvenie punkti: datu veids, veidu kā pie tā var piekļūt, atrašanas vieta un vai ir pieejamās formatēšanas iespējas[2]. Lai izvārīties no viena no galvenēm problēmām manuālai dokumentācijas rakstīšanai, kuras bija aprakstītas iepriekš - 2.4 sadaļā sakumā, vajag nodrošināt elastīgu darbību. Automatizētajam dokumentu ģeneratoram ir jānodrošina bagātīgs un elastīgs utilītu komplekts, ko lietotāji var izmantot, lai no katra izstrādes rīka iegūtu vislabāko pielāgošanas informāciju.[2]. Tas ir vajadzīgs lai vienlaikus iegūtu informāciju no vairākiem dažādiem rīku krātuvēm, ģenerētu dokumentus piemēram populāram dokumentu publicēšanas sistēmām un arī sagatavo dažādus dokumentus vien laikā.[2]. Turpinot runāt par automatizētajam dokumentu ģeneratoram, tam ari ir jānodrošina pazīstams, konsekvents lietotāja interfeiss. [2]. Tas ir vajadzīgs lai ar to būtu viegli un saprotami strādāt, neveidojot liekas darbības veidotās neuzmanības dēļ, kuras var izraisīt problēmas un liekas darbības vai pienākumus sistēmas izstrādātajiem. Kā ari, dokumentu ģeneratoram ir jānodrošina dokumentu veidnes daudziem valdības, komerciālajiem un starptautiskajiem standartiem.[2]. Veidojot un sekojot dažādiem līmeņu standartiem rakstot manuālu dokumentāciju, izstrādātajiem tas var aizņemt divas vai trīs reizes vairāk laika un spēka salīdzinot ar automātisko dokumentācijas ģenerēšanu izmantošanu, kur veidnes šiem standartiem ir jau pieejamās dokumentācija ģeneratora.

Nodrošinājāt dokumentu ģenerēšanu drošu, elastīgo un bagātīgu var gan izvairīties no dažādam darba pienākumiem programmatūras izstrādātajiem, gan atvieglot izstrādes procesu, gan paātrināt to.

2.7. Pirmkoda dokumentācija

Programmatūras izstrādes procesa dokumentācija tiek izmantotā dažādos izstrādes fāzes, un viena no tām ir pirmkoda dokumentācija. Kā bija pateikts iepriekš programmatūras dokumentācijā eksistē, lai palīdzētu izstrādātājiem un lietotājiem izprast sistēmu un tās procesus [3],[6]. Tomēr laika prakse parāda izstrādātajiem un uzņēmumiem kā manuāla pirmkoda dokumentācijas izveidošana ir dārga, kā rezultātā dokumentācijā bieži vien ir nepilnīga un tas ir zemes kvalitātes dokuments [3]. Bet tas avotu koda dokumentācija ir svarīgs artefakts efektīvai darbībai programmatūras izstrādei[5]. Protams ar mūsdienas automatizācijas tendence pirmkoda dokumentācijā vajag arī iegūt atjauninājumus. Kodu dokumentācijā varētu gūt lielu labumu no automatizācijas, jo ar rokas uzrakstīta vai manuāli datorā uzrakstīta dokumentācija bieži ir darbietilpīgā, resursu un laikietilpīgā darbība [5]. Runājot tieši kas ir avotu dokumentāciju, tas ir pirmkoda dokumentācijā kurā ir aprakstīts kā dažādos veidos darbojas pirmkods, vienkāršā tekstā dažādos valodas [4]. Tas koda kopsavilkums ir neliels teksta daudzums, apmēram no 1 līdz 3 teikumiem, kas izskaidro avota koda darbību vai teksts kā to var izmantot [4].

Daži pētījumi ir izvērsuši ka ir problēma ar automātisko pirmkoda kopsavilkumu izveidošanu[4]. Tagad izmantoti rīki priekš pirmkoda kopsavilkuma ģenerēšanai, kura mērķis ir uzdevuma veikšana, kļuvusi efektīva izmantošana īpašos apstākļos, bet tas tāpat nespēj sasniegt kvalitāti ko cilvēks var sniegt veicot vai izpildīt šos uzdevumus[4].

Secinājumus par pirmkoda dokumentāciju var pateikt kā uz dotu mirklī ir grūti izveidot sistēmu kas pilnīgi automatizē pirmkoda dokumentāciju, jo pietrūkts informācija kas tieši ir nepieciešams programmētajiem. Ar uzlabotu izpratni par programmētāju rakstīšanu dokumentāciju, būtu iespējams izveidot sistēmu kas varēs automātiski ģenerēt pirmkoda kopsavilkumu un dokumentāciju.

2.8. Programmētāji

Programmētāji ir neatņemta daļa gan no izstrādes procesā, gan neatņemtā daļa no dokumentācijas veidošana un to lasīšana. Kopuma, dokumentācijā izstrādes procesa ir tieši vajadzīga priekš programmētājiem [6]. Kā bija apkopots iepriekš, sistēmas izstrādātāji izmanto dokumentāciju lai izprast kodu, tas darbību veikt sistēmas uzlabojumus izmantojot dokumentāciju. [6]. Viss sastopama informācijā kas tiek ietverta tehniskā dokumentācijā ir kods, koda apraksts, algoritmu apraksts, informācija par dizainu, saskarēm un programmatūras funkcionalitāte.[6],[4]. Diemžēl daudziem programmētājiem nav laika un vēlmes izveidot un uzstūrēt dokumentāciju[4]. Kā bija pateikts vienā no zinātniskiem rakstiem: “Programmētāji paļaujas uz pirmkoda dokumentāciju lai ātri saprast, ko un kā dara izmantots kods” [4].

Programmētāji ir atbildīgi ne tikai par tehnisko dokumentāciju izstrādi kuru viņi izmanto izstrādes laika, bet arī ir atbildīgi par lietotāja rokasgrāmatu izveidošanu. Rokasgrāmatas dokuments ir vajadzīgs priekš sistēmas vai programmatūras lietotājiem lai paskaidrot, kā lietotājiem ir jāsadarbojas ar sistēmu. [6]. Dokumentā tiek pievienotā informācija par noteiktas funkcijas darbību, darbības secība kas tieši vajag izdarīt lai dabūt vēlamo rezultātu, kā ari ir tehniska informācija par sistēmu kura var būt noderīga priekš programmatūras izmantošanai.

Neskatoties uz to, kā dokumentācijā ir ļoti vajadzīga priekš izstrādes procesa un tā izstrādātājiem izveidot to aizņem daudz laika un spēka. Lai risinātu šo problēmu vismaz daļēji, ir ierosinātas dažādas pieejas un instrumenti, par kuriem būs vairāk aprakstīts nākošā sadaļa, lai palīdzētu izstrādātājiem programmatūras dokumentācijas laika. Tomēr, lai panāktu pilnīgas automātiskas dokumentācijas ģenerēšanai bez cilvēka palīdzības ir vispirms nepieciešama dziļa programmatūras izpratne kuru var dabūt tieši no programmētajiem un testa laika [6].

Secinājumos var pateikt kā dokumentācijā nav tikai vajadzīga un svarīga priekš programmētajiem, bet arī tas ir izstrādātā priekš cilvēkiem kas izmantots sistēmu katru dienu. Kā ari priekš izstrādātajiem kas atjauninās un uzturēs šo sistēmu pēc izstrādes laika.

2.9. Atslēgvārdi

Atslēgvārdi ir ari viena no neatņemtam dalām dokumentācijā automatizācijas izveidošanai. Viena no zinātniskiem rakstiem bija izveidota un izstrādāts pētījums, kura ietvaros bija uzdevums saprast uz ko tieši programmētāji pievērs uzmanību lasot dokumentāciju. Izmantojot acu izsekošanas sensoru mērķis bija saprast uz ko tieši skatās izstrādātāji lai turpmāk tas varētu izmantot avota koda apkopojuma un turpmāk dokumentācijas izveidošanai. [4]. Pētījuma bija paradīts ka programmētāji lasītie atslēgvārdi patiesība ir tie atslēgvārdi, kurus neatkarīgs izstrādātāju kopums uzskatīja par svarīgiem.[4]. Izmantojot šo pētījumu var tieši saprast kurus atslēgvārdus vajag izmantot automatizācijas dokumenta ģenerēšanai. Bet protams, mūsdienas jau eksistē viens paņēmiens atslēgvārdu izvelēšanai. Viens vismodernākais paņēmiens atslēgvārdu atlasei no avota kodu apkopojumam ir aprakstījuši Haiduc. [3],[4]. Galvenā idejā šajā paņēmienā ir vektora telpas modeļa izmantošana, kas, kā seciņā zinātniska raksta autors, ir klasiska dabiskās valodas izpratnē tehnika.[4].

2.10. Dokumentācijas rīki

Izstrādātāji vai cilvēki kas mēģinās izveidot automatizētu dokumentu ģeneratoru ar mērķi palīdz programmētājiem izstrādes procesa vajag zināt kā tieši strādā citi dokumentācijas rīki. Izstrādātājiem jāzina tieši kādi programmatūras izstrādes rīki un metodes tiek izmantotas, kādi dokumenti ir jāsagatavo, kādi standarti tiek izmantoti un kāda dokumenta publicēšanas sistēma tiek izmantota.[2]. Izmantojot šis iegūtas zināšanas izpētes laika izstrādātāji var izmantot izstrādājot automatizētas dokumentu ģeneratoru. Pie šādam zināšanām attiecas kopas katrai piemērojamai izstrādes rīka kombinācijai, metodei, standartiem un dokumentu publicēšanas sistēmai.

Kā bija pateikts iepriekš dokumentāciju autori bieži izmanto ģenerēšanas rīkus, lai standartizētā veidā komunicēt ar lasītājiem. [3]. Kā piemēru var minēt rīkus tādus kā DoxyGen1 un JavaDoc2. Viņi abi ir palīdzējusi avota kodam dokumentācija izveidošanu, standartizējot formātu un noformējumu pirmkoda dokumentācijai.[3]. Šie divi, tika aprakstīti rīki, prasa no programmētājiem kā ir jāraksta kopsavilkumu kā īpaši formatēti metadati, un visi rīki, ko var darīt, ir ievietot informāciju HTML dokumentos. [3], [4].

JavaDoc rīks ir tagad izmantots kā standarta prakse Java pasaulē. [1]. Savukārt rīks Doxygen piedāvā līdzīgu funkcionalitātē priekš C un C++ valodas lietotajiem, kuri paļaujas uz to. [1]. Abi rīki var arī apgriezt kodu, lai izveidotu klases diagrammas.[1].

Galvenā problēma ar ko saskaras automātiskā pirmkoda kopsavilkums, ir saistīta, ka nav saskaņota vai skaidri definēts ka tieši ir “laba” dokumentācija un dokumentācijas standarts. [3]. Aprakstītiem rīkiem kuru mērķis ir veikt uzdevumus, paradījuši efektivitātē īpašos aptīklos, bet šādi rīki nespēj tapāt sasniegt cilvēka kvalitātes kopsavilkuma izveidošanas līmeni.[4].

Nav noslēpums kā tagad jau eksistē daži rīki kas var automātiski veidot programmatūras dokumentāciju un palīdz izstrādātajiem saglabāt savu laiku un spēku uz citiem izstrādes procesiem. Viens no tādiem rīkiem, vai labāk pateikt sistēmām, ir aprakstīts [7] zinātniskā rakstā. Sistēmas nosaukums ir lpdoc. Lpdoc ir automātisks programmu dokumentācijas ģenerators LP sistēmā.[7]. Šāda ģeneratora galvenā funkcija ir automātiski ģenerēt atsauces rokasgrāmatā no viena vai vairākiem loģiskās programmēšanas sistēmu avota failiem.[7] Lpdoc apvieno informāciju no vairākiem lietotāju un sistēmas failiem un sagatavo rokasgrāmatas vairākos formātos, kas var ietvert citātus un attēlus.[7] Diagrammu šādai sistēmai var apskatīt ***Ilustrācija 1.***

A diagram of a computer

Description automatically generated

1.0.att. LP sistēmas operācijas

Beigas var secināt kā uz dotu mirkļi ir pieejami dažādi rīki kas palīdz programmētajiem veidot dokumentāciju, bet pagaidām nav izveidotā viena standartizēta sistēma kas būs attiecīga visiem izstrādātājiņiem programmatūrām.

2.11. Secinājumi

Izanalizējot visus izlasītos zinātniskos rakstus par automātiskā dokumentācijas ģenerēšanu vai izveidošanu, kā ari aprakstot visu iegūtu svarīgo informāciju dokumentā. Galvenais nolūks šādam dokumentam ir saprast kādu efektu automātiskā dokumentācija sniedz izstrādātajiem un izstrādes procesa kopumā, kā ari noteikt kādi rīki vai sistēmas jau eksistē tagad un kādus efektus un labojums tas sniedz programmētājiem. Izveidojot kopsavilkumu var tiešam redzēt kā automātiskā dokumentācijas ģenerēšana sniedz programmētajiem, kas izstrādā sistēmu, tikai labo efektu. Tas paātrina izstrādes procesu, atbrīvo programmētājus no liekiem darbībām, dos iespēju izveidot dokumentu viena standarta, lai citi programmētāji saprata kā tieši strādā sistēmā. Kā ari tas palīdz viņiem koncertēties uz koda rakstīšanu, bet darboties ar ne ļoti interesantu procesu. Runājot par citiem priekšrocībām priekš automātiskai dokumentācijas var definēt, kā izmantojot to: gramatisko kļūdu skaits samazināsies, dokumentu standarts pārkāpums nebūs vairs problēmai, kā ari dokumentācijas uzturēšana un atjaunināšanā var kļūt par ļoti ātru un ērtu procesu programmatūras izstrādes dzīve. Trūkumi ar kuriem var sastapties strādājot ar dokument automatizācijas rīka var aprakstot divus teikumus. Pirmā amatā problēma šādam ģeneratoram ir tāda kā pagaidām daudziem nav izpratnes kas tieši vajag iekļaut šādā dokumenta lai izstrādātāji izmantos tas procesa vai kas vajag iekļaut lietotāju rokasgrāmatā lai būtu zināšanas par izmantotu programmatūru, jo katrā programmatūrā ir unikāla. Un otrs trūkums ir saistīts ar šādas ģeneratora, rīka vai sistēmas izveidošanai priekš noteiktām darbībam, kuru prasa no cilvēka daudz tehniskas zināšanas, zināšanas par izstrādes procesu un protams laiku.

Beigas var secināt, kā izanalizējot un aprakstot zinātniskos rakstus var pateikt kā tika iegūtas daudz zināšanas par bakalaura tēmu ar nosaukumu “Automatizētas dokumentācijas ģenerēšanas ietekmes izpēte uz izstrādes procesu”, kā ari bija apstiprinātā hipotēzē kura bija aprakstīta ievada apstiprinās par šo tēmu.

Izmantotie informācijas avoti

[1] D. Spinellis, "Code Documentation," in IEEE Software, vol. 27, no.4, pp.18-19,

July-Aug. 2010.

Pieejams šeit: <https://ieeexplore-ieee-org.resursi.rtu.lv/document/5484109>

[2] L. Stein, "Trends in automating document generation," in IEEE Software, vol. 12, no. 5, pp. 116-118, Sept. 1995.

Pieejams šeit: <https://ieeexplore-ieee-org.resursi.rtu.lv/document/406781>

[3] P.W.McBurney, "Automatic Documentation Generation via Source Code Summarization," 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Florence, Italy, 2015, pp. 903-906.

Pieejams šeit: <https://ieeexplore-ieee-org.resursi.rtu.lv/document/7203110>

[4] P. Rodeghero, "Behavior-Informed Algorithms for Automatic Documentation Generation," 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 2017, pp. 660-664.

Pieejams šeit: <https://ieeexplore-ieee-org.resursi.rtu.lv/document/8094477>

[5] J.Y.Khan and G.Uddin, “Automatic Code Documentation Generation Using GPT-3” Proceedings of the 37th IEEE/ACM International Conference on automated software engeneering, ase 2023

Pieejams šeit:

<https://www.webofscience.com/wos/woscc/full-record/WOS:001062775200129>

[6] E. Aghajani et al., "Software Documentation Issues Unveiled," 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), Montreal, QC, Canada, 2019, pp. 1199-1210.

Pieejams šeit: <https://ieeexplore-ieee-org.resursi.rtu.lv/document/8811931>

[7] M.Hermenegildo, “A System for Automatically Generating Documentation for (C)LP Programs” Electronic Notes in Theoretical Computer Science, vol.30, no.4, pp. 289-307, May 2005

Pieejams šeit:

<https://www-sciencedirect-com.resursi.rtu.lv/science/article/pii/S1571066105806632>

[8] A.Andersen, “Automatic Generation of Technical Documentation” Expert Systems with Applications, vol.3, no.2, pp. 219-227, 1991.

Pieejams šeit:

<https://www.sciencedirect.com/science/article/abs/pii/095741749190150D>

[9] P.W.McBurney and C.McMilan, “Automatic Documentation Generation via Source Code Summarization of Method Context” June 2014.

Pieejams šeit: <https://www3.nd.edu/~cmc/papers/mcburney_icpc_2014.pdf>

[10] M.Pushpa Arthuer, “Automatic Source Code Documentation using Code Summarization Technique of NLP” Procedia Computer Science, vol.171, pp. 2522-2531, 2020.

Pieejams šeit:

<https://www.sciencedirect.com/science/article/pii/S1877050920312655>