

ДОДАТОК А
Програмний код

Програмне забезпечення для отримання метрик проектів

Файл Project.py

```

import sys
from datetime import datetime, timedelta
from LinksManager import LinksManager as LM
from Storage import Storage
from DBClient import MongoClient as DB

from XMLParser import RequestStatusParser as RSP
from XMLParser import ProjectParser
from XMLParser import SizeParser
from XMLParser import ActivityParser
from XMLParser import ContributorParser
from XMLParser import FacotidsParser
from XMLParser import AnalysisParser

import RequestRawData as Request

class ProjectCore(object):

    __savePath = "savePath\"
    __format = ".csv"

    def __init__(self):
        self.request = Request.RequestRawData()
        self.requestStatus = RSP.RequestStatusParser()

        self.parserDictionary = {
            "size" :
SizeParser.SizeParser(),
            "project" :
ProjectParser.ProjectParser(),
            "activity" :
ActivityParser.ActivityParser(),
            "factoid" :
FacotidsParser.FacotidsParser(),
            "analysis" :
AnalysisParser.AnalysisParser(),
            "contributors" :
ContributorParser.ContributorParser()
        }

        self.linksManager = LM.LinksManager()
        self.db = DB.MongoDBClient()

    def GetCurrentData(self, link, parserKey):
        response = self.request.GetRawXML(link)
        isSuccess =
self.requestStatus.GetRequestStatus(response)

```

```

    if isSuccess:
        parser = self.parserDictionary[parserKey]
        data = parser.Parse(response)
        return data
    else:
        return []

    def GetHistoricalData(self, id):
        allLinks = self.linksManager.GetProjectLinks(id)
        project = self.GetCurrentData(allLinks["project"],
"project")
        if project:
            self.db.SaveNewProject(project)

    def UpdateExistingProjectData(self, projectId):
        allLinks = self.linksManager.GetProjectLinks(projectId)

        activity = self.GetCurrentData(allLinks["activity"],
"activity")
        self.db.UpdateProjectData(projectId, activity,
"activity")

        size = self.GetCurrentData(allLinks["size"], "size")
        self.db.UpdateProjectData(projectId, size, "size")

        analysis = self.GetCurrentData(allLinks["analysis"],
"analysis")
        self.db.UpdateProjectData(projectId, analysis,
"analysis")

        contributors =
self.GetCurrentData(allLinks["contributors"], "contributors")
        self.db.UpdateProjectData(projectId, contributors,
"contributors")

        factoids = self.GetCurrentData(allLinks["factoid"],
"factoid")
        self.db.UpdateProjectData(projectId, factoids,
"factoids")

    def GetProjectData(self):
        return self.db.Load("project")

    def ProcessAllProjects(self):
        allProjects = self.GetProjectData()
        for p in allProjects:
            print p["name"]
            self.SaveProjectDataToCsv(p)

    def SaveProjectDataToCsv(self, data):
        try:
            topLine = "Date,CodeLines Added,CodeLines
Removed,CommentsLines Added, CommentsLines Removed,Blanks

```

```

Added,Blanks
Removed,ManMonths,CodeLines,Comments,Blanks,TotalLines,ManMonthD
elta,Commits,Contributors \n"
    lineTemplate =
"{0},{1},{2},{3},{4},{5},{6},{7},{8},{9},{10},{11},{12},{13},{14
} \n"
    name = data["name"]
    userCount = data["userCount"]
#title = "Name,{0},User Count,{1} \n".format(name,
userCount)
    prevManMonth = 0
    prevCommits = 0
    prevContributors = 0
    activity = data["activity"]
    sizeArray = data["size"]
    for a in activity:
        retrieveDate = a["date"]
        size =
self.FindSizeByDate(retrieveDate,sizeArray)
        totlaLines = int(size["codeLines"]) +
int(size["comments"]) + int(size["blanks"])

        manMonth = size["manMonths"]
        manMonthDelta = manMonth - prevManMonth
        prevManMonth = manMonth

        commits = a["commits"]
        contributors = a["contributors"]

        t = lineTemplate.format(retrieveDate.date(),
a["codeAdded"], a["codeRemoved"],a["commentsAdded"],
a["commentsRemoved"],a["blanksAdded"],a["blanksRemoved"],manMont
h,size["codeLines"],size["comments"],size["blanks"],totlaLines,m
anMonthDelta,commits,contributors)
        topLine += t
        f = open(self.__savePath + name + self.__format,
'w')
        f.write(topLine)
    except IOError:
        print "IO error: " + name

def FindSizeByDate(self,activityDate,sizeArray):
    for size in sizeArray:
        sizeDate = size["date"]
        if(sizeDate == activityDate):
            return size
        else:
            continue
    return self.findNearestSize(activityDate,sizeArray)

def findNearestSize(self,activityDate,sizeArray):
    for idx, size in enumerate(sizeArray):
        sizeDate = size["date"]

```

```

        if(sizeDate > activityDate):
            return size
    return sizeArray[len(sizeArray) - 1]

```

Файл Account.py

```

import sys
from LinksManager import LinksManager as LM
from Storage import Storage
from DBClient import MongoClient as DB
import RequestRawData as Request

from XMLParser import AccountParser
from XMLParser import RequestStatusParser

class AccountCore():

    def __init__(self):
        self.linksManager = LM.LinksManager()
        self.db = DB.MongoDBClient()
        self.parser = AccountParser.AccountParser()
        self.request = Request.RequestRawData()
        self.statusCheck = RequestStatusParser.RequestStatusParser()

    def GetAccountData(self, accountId):
        link = self.linksManager.GetAccountLink(accountId)
        response = self.request.GetRawXML(link)
        isSuccess = self.statusCheck.GetRequestStatus(response)
        if isSuccess:
            account = self.parser.ParseAccountData(response)
            self.db.SaveAccount(account)

```

Файл Language.py

```

import sys
from LinksManager import LinksManager as LM
from Storage import Storage
from DBClient import MongoClient as DB
from XMLParser import RequestStatusParser as RequestStatus
from XMLParser import LanguageParser as LP
import RequestRawData as Request

class LanguageCore(object):

    def SaveLanguageData(self):
        linkManager = LM.LinksManager()
        statusChecker = RequestStatus.RequestStatusParser()
        languageParser = LP.LanguageParser()
        request = Request.RequestRawData()
        db = DB.MongoDBClient()
        linkTemplate = linkManager.GetLanguageLinkTemplate()
        index = 1;

```

```

while True:
    link = linkTemplate.format(index)
    response = request.GetRawXML(link)
    status = statusChecker.GetRequestStatus(response)
    if status == True:
        languageData =
languageParser.ParseLanguageData(response)
        if languageData:
            db.SaveLanguages(languageData)
            index += 1
        else:
            break
    else:
        break

```

Файл MongoDBClient.py

```

import sys
from pymongo import MongoClient

class MongoDBClient(object):
    """Class for savig/loading data to/from MongoDB instance"""

    def __init__(self):
        connection = MongoClient("localhost", 27017)
        self.db = connection.OSStatsDB

    def SaveNewProject(self, value):
        id = -1
        project = self.db.project
        id = project.insert(value)
        return id

    def UpdateProjectData(self, projectId, values, key):
        self.db.project.update( { "id" : projectId}, { "$addToSet":
{ key: {"$each" : values } } } )

    def SaveLanguages(self, value):
        self.db.languages.insert(value)

    def SaveAccount(self, value):
        count = self.db.account.find(value).count()
        if count == 0:
            self.db.account.insert(value)

    def Load(self, key):
        result = []
        if key == "project":
            for p in self.db.project.find():
                result.append(p)
        return result

```

LinksManager.py

```

import sys

class LinksManager(object):

    __apiKey = "API_KEY"

    __dict = {
        'project':
'http://www.ohloh.net/projects/{0}.xml?api_key={1}',
        'analysis':
'http://www.ohloh.net/projects/{0}/analyses/latest.xml?api_key={1}',
        'size':
'http://www.ohloh.net/projects/{0}/analyses/latest/size_facts.xml?ap
i_key={1}',
        'contributors':
'http://www.ohloh.net/projects/{0}/contributors.xml?api_key={1}',
        'activity':
'http://www.ohloh.net/projects/{0}/analyses/latest/activity_facts.xml
?api_key={1}',
        'factoid':
'http://www.ohloh.net/projects/{0}/factoids.xml?api_key={1}'
    }

    def GetProjectLinks(self, projectId):
        links = {}
        for key in self.__dict.keys():
            links[key] = self.__dict[key].format(projectId,
self.__apiKey)
        return links

    def GetLanguageLinkTemplate(self):
        rawLink = "http://www.ohloh.net/languages.xml?api_key={0}"
        link = rawLink.format(self.__apiKey) + "&page={0}"
        return link

    def GetAccountLink(self, accountId):
        rawLink =
"http://www.ohloh.net/accounts/{0}.xml?api_key={1}"
        link = rawLink.format(accountId, self.__apiKey)
        return link

```

ProjectParser.py

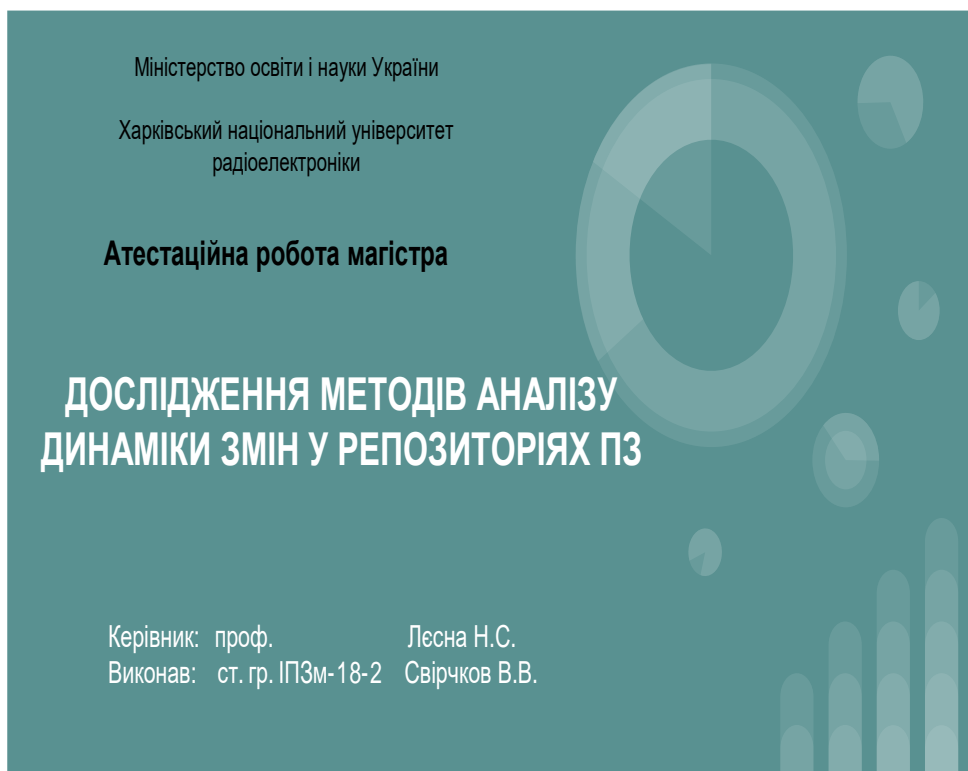
```

import sys
import xml.etree.ElementTree as ET
import datetime

class ProjectParser(object):

```

ДОДАТОК Б
Слайди презентації



Об'єкт дослідження

процеси розробки проектів з відкритим кодом.

- Предмет дослідження: методи збору інформації про історію розвитку Open Source проектів, методи оцінки стану проектів.

Аналіз метрик історії роботи над ІТ-проектом може стати ефективним засобом оцінки поточного стану та перспектив розвитку Open Source проектів.

Мета роботи

- Розробка алгоритму прогнозу розвитку Open Source проектів, що дозволяє більш точно оцінювати поточний стан проекту, а також перспективи його розвитку.
- Розроблені методи дозволять керівникам розробки проектів більш ефективно планувати подальшу розробку, а кінцевим користувачам отримувати додаткову інформацію при виборі проекту.
- Методи дослідження – для аналізу метрик роботи над проектами будуть використані моделі і методи розрахунку метрик складності програмного забезпечення, статистичний аналіз.

3

Основні властивості ВПЗ

- Можливість виправлення помилок і покращення програм – найважливіша особливість вільного і відкритого програмного забезпечення
- Поняттю вільного програмного забезпечення (freeware) близько поняття відкритого програмного забезпечення (opensource software) - програмне забезпечення з відкритим вихідним кодом.
- Вихідний код таких програм доступний для перегляду, вивчення та зміни, що дозволяє користувачеві взяти участь в доопрацюванні найбільш відкритою програми, використовувати код для створення нових програм і виправлення в них помилок – через запозичення вихідного коду, якщо це дозволяє ліцензія, або через вивчення використаних алгоритмів, структур даних, технологій, методик і інтерфейсів.

4

Моніторинг змін в репозиторіях проектів

- Зростання кількості проектів з відкритим вихідним кодом і як наслідок зростання числа репозиторіїв призводить до складності відстеження та порівняльного аналізу різноманітних проектів.
- Цим обумовлена необхідність розробки методу і ПЗ для моніторингу змін в репозиторіях проектів з відкритим вихідним кодом.
- В рамках магістерської роботи буде досліджено метрики, що характеризують історію роботи над проектами.
- Як джерело вихідного коду для вивчення буде використано відкрите ПЗ.

5

Метрики програмного забезпечення

- **software metric** – міра, що дозволяє одержати чисельне значення деякої властивості програмного забезпечення або його специфікацій.
- Метрики необхідні для того, щоб:
 - показати, яким чином діяльність у деякій конкретній сфері функціонування підприємства вносить вклад у досягнення заздалегідь певних цілей;
 - виявити зміни й істотні аномалії в процесах і прийняти обґрунтовані рішення по виправленню або поліпшенню процесів

6

Метрики виміру якості програм



використовуються для виміру характеристик і критеріїв якості.

Як правило, метрики якості подають інформацію рекомендаційного характеру й використовуються для вивчення складності розроблювального ПЗ, оцінки затрат праці й обсягу робіт, необхідних для реалізації проекту, і дають оцінювання про можливості поліпшення програмного коду або оптимізації зусиль, затрачуваних розроблювачем на створення програми.

Також, метрики якості можуть слугувати критерієм вибору варіантів розпаралелювання для програм, що функціонують у розподіленому режимі.

7

Метрики оцінки якості програмного забезпечення



надають уявлення про надійність програм, їхньої продуктивності, складності супроводу та зміни і т.ін.

- У атстаційній роботі досліджуються метрики оцінки складності, які поділяються на 3 основні групи:
 - метрики розміру програм, або кількісні метрики;
 - метрики складності потоку керування програми;
 - метрики складності потоку даних програм.

8

Базис для оцінки складності програм

метрика цикломатичної складності ПЗ при високоточних обчисленнях в паралельному режимі.

```
A = function(n1, m1); S = 0;
for (i = 0; i <100; i++)
{
  if (A)
  {
  }
  else
  {
  }
  AfxMessageBox("Starting calculation");
  AfxMessageBox("Error!");
  for (j = 0; j < m1; j++)
  {
  S += S * i + n1 * j; if (S > A)
  {
  AfxMessageBox("Value found!");
  }
  }
}
```

Переваги оцінки цикломатичної складності:

- направляє процес тестування обмежуючи складність програмної логіки на етапі розробки;
- зручна в застосуванні.

Недоліки:

- це метрика виміру складності керування програми, а не її даних;
- при оцінці складності не враховується параметризація графів, тому програми зі схожими графами будуть оцінені однаково;
- при оцінці не відслідковується потік даних, що може також дати перекручене подання про складність програми.

9

Аналіз факторів і висновків програмного експерименту

- В термінології планування експериментів вхідні змінні і структурні допущення, що становлять модель, називаються факторами, а вихідні показники роботи – висновками.
- У багатьох випадках фактори можуть носити не тільки кількісний, але і якісний характер.
- Якщо при проведенні експерименту можна змінювати рівень фактора, то експеримент називається активним, в іншому випадку – пасивним.
- Факторами буде інформація про історію змін вихідного коду різних Open Source проектів за деякий період часу.
- Джерелом даних є API сайту Ohloh.net.
- В ході експерименту ніяких змін в вихідні коди досліджуваних проектів вноситься не буде. Отже, експеримент носить пасивний характер.

10

Висновок програмного експерименту

є набір числових характеристик, що описують історію роботи над проектом.

Для кожного проекту будуть отримані такі метрики:

- кількість доданих рядків коду;
- кількість вилучених рядків коду;
- кількість доданих рядків коментарів;
- кількість вилучених рядків коментарів;
- кількість доданих порожніх рядків;
- кількість вилучених порожніх рядків;
- число розробників;
- число коммітов;
- число рядків коду;
- число коментарів;
- число порожніх рядків;
- число людино-місяців;
- число користувачів;
- загальна кількість рядків коду;
- дата збору інформації.

11

В результаті виконання експерименту

будуть отримані чисельні значення набору метрик для кожного досліджуваного проекту.

Зважаючи на велику кількість отриманих даних за доцільне виглядає застосувати методи статистики для їх обробки.

Для кожного набору значень по одній метриці отримано наступні статистичні характеристики:

- мінімальне значення метрики;
- максимальне значення метрики;
- математичне очікування – середнє значення випадкової величини

12

В результаті для кожного з досліджуваних проектів має бути отримана таблиця

Назва	Тип даних	Коментар
Code added	Integer	Кількість доданих рядків коду
Code removed	Integer	Кількість вилучених рядків коду
Comments added	Integer	Кількість доданих рядків коментарів
Comments removed	Integer	Кількість вилучених рядків коментарів

13

Коефіцієнт лінійної кореляції Пірсона

– найбільш часто використовуваний коефіцієнт кореляції.

Призначений для розрахунку сили і напрямку лінійної залежності між змінними дослідження. Передбачається, що змінні виміряні в інтервальній шкалі або в шкалі відносин.

- Загальна формула для обчислення коефіцієнта лінійної кореляції Пірсона:

$$r_{xy} = \frac{\sum(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \cdot \sum(y_i - \bar{y})^2}}$$

де x_i – значення, що приймаються змінною X,

y_i – значення, що приймаються змінною Y,

\bar{x} – середня по X,

Розрахунок коефіцієнта кореляції Пірсона припускає, що змінні X і Y розподілені нормально.

14

Розроблений алгоритм

Для виконання даного дослідження необхідно розробити ПЗ, що має наступні функції:

- збір вихідних даних про проекти за допомогою періодичного опитування API;
- перетворення отриманих даних до придатного для подальшої обробки вигляду;
- виконання необхідних розрахунків (видалення надлишкових параметрів, перевірка гіпотез).

15

Розрахунок метрик

Розрахунок продуктивності розробника призначений для визначення середньої кількості рядків коду, коментарів, а також порожніх рядків, доданих або видалених одним розробником за один місяць.

$$\text{codeLines} = \frac{\text{CodeAdd} + \text{CodeRem} + \text{ComAdd} + \text{ComRem} + \text{BlankAdd} + \text{BlankRem}}{\text{ManMonthDelta}}$$

Розрахунок відсотка рядків коментарів – дана метрика вказує ступінь покриття вихідного коду коментарями і є важливим показником самодокументованості коду. Дана метрика розраховується за такою формулою:

$$\text{Comments} = \frac{\text{CommentLines}}{\text{CodeLines} + \text{CommentLines} + \text{BlankLines}}$$

16

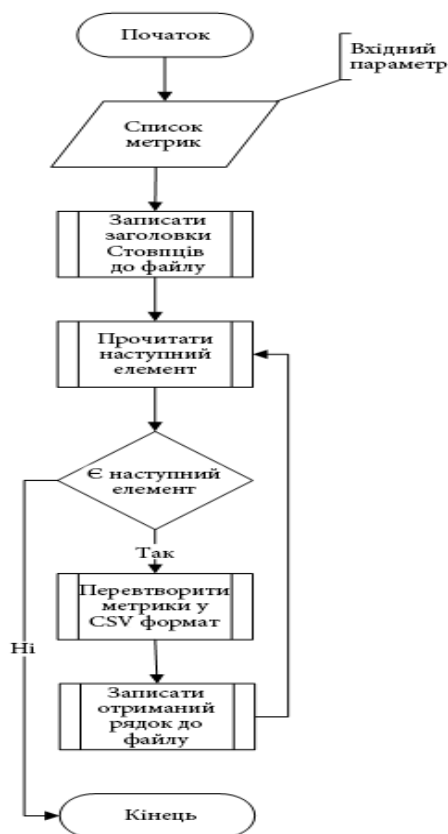
Опис розробленого алгоритму збору даних про історію роботи над проектом

Вхідним параметром даного алгоритму є ID проекту.

- На першому кроці проводиться запит до API сховища Ohloh.net для отримання даних про історію роботи над проектом.
- Після чого перевіряється статус відповіді. Якщо запит виконаний успішно (статус 200), то з отриманих даних витягуються дані про метриках розміру, складності, а також дата обчислення метрик.
- Ці дані перетворюються в формат JSON і записуються в БД. У разі якщо запит виконаний невдало, виводиться повідомлення про помилку.
- Вхідними даними для даного алгоритму є список метрик, що характеризує роботу над проектом.
- На першому кроці в результуючий файл записуються заголовки стовпців. Після чого, в циклі з БД вичітаються метрики проекту.
- Дані з кожною прочитаною метрикою перетворюються в формат CSV, після чого отриманий рядок записується в файл результату.
- Цикл повторюється до тих пір, поки не будуть прочитані всі елементи.

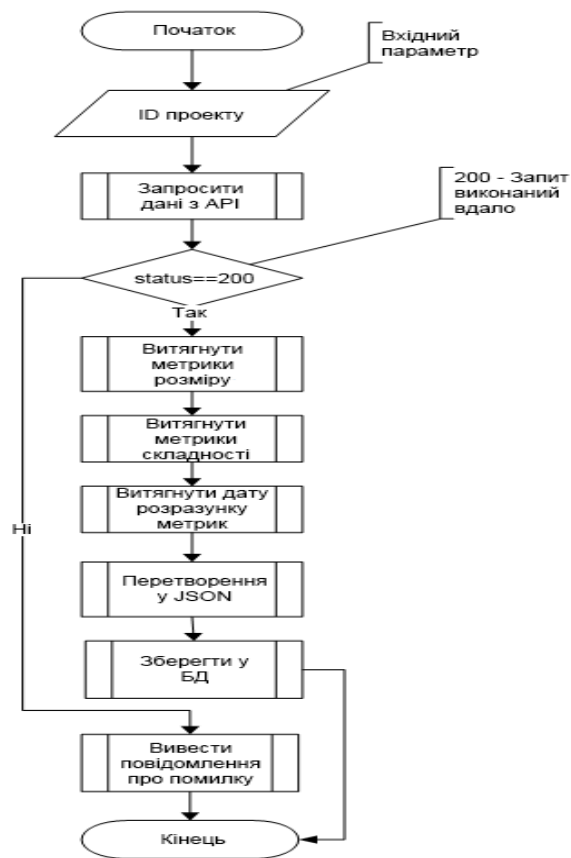
17

Схема алгоритму збору даних про історію роботи над проектом



18

Перетворення даних до плоского виду



19

Як проект для розгляду буде використаний Apache Subversion та розглянуті метрики, що характеризують додавання нових рядків у проект

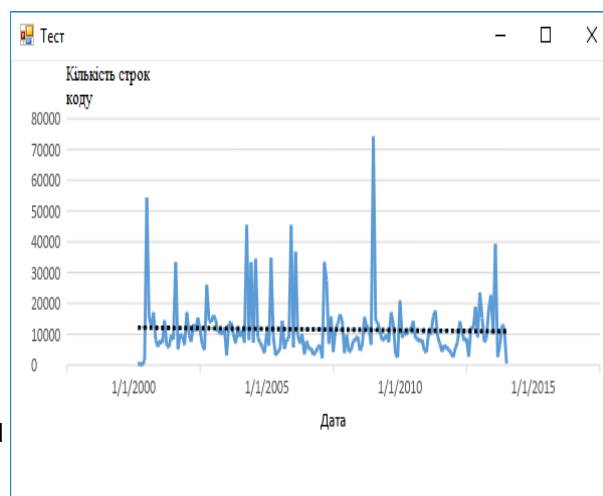
а саме:

- кількість доданих рядків коду;
- кількість доданих рядків коментарів;
- кількість доданих порожніх рядків.

Перша метрика

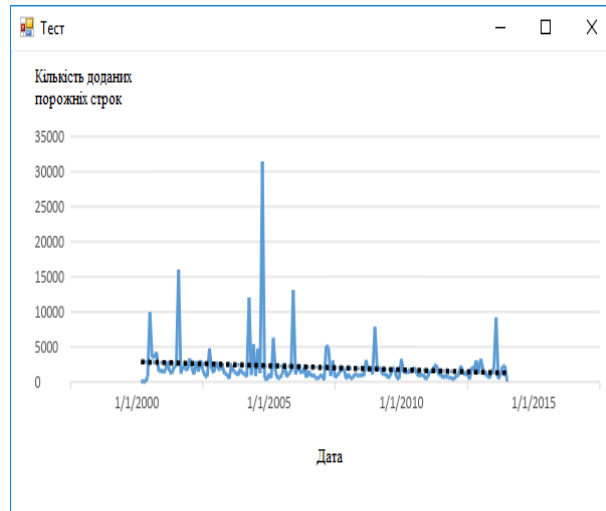
– кількість доданих рядків коду.

Графік зміни цієї метрики



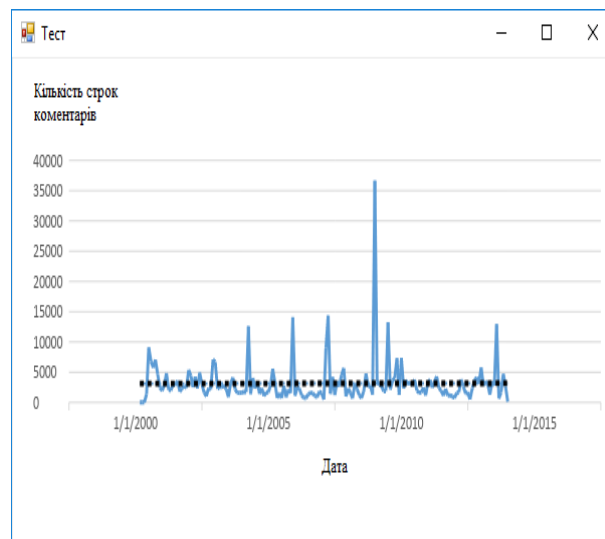
20

Графік зміни метрики «кількість доданих порожніх рядків»



21

Кількість доданих коментарів



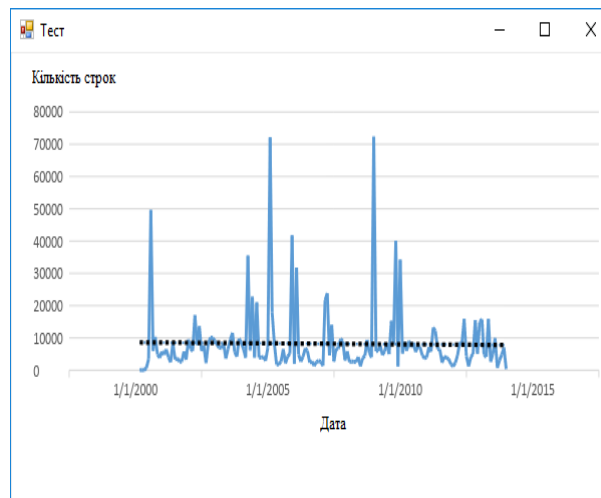
22

Метрики, що характеризують видалення рядків з проекту

це:

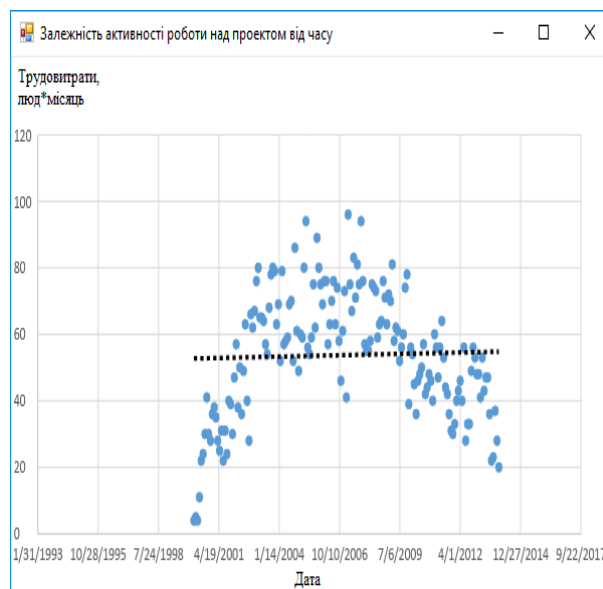
- кількість вилучених рядків коду;
- кількість вилучених рядків коментарів;
- кількість вилучених порожніх рядків.

Кількість вилучених рядків коду



23

Графік залежності активності роботи над проектом від часу для проекту Apache Subversion



24

Результат перевірки гіпотези

Результати аналізу

дозволяють зробити наступні загальні висновки про історію досліджуваних проектів.

Метрика «Кількість доданих рядків коду» сильно корелює з метриками «кількість доданих порожніх рядків» і «активність розробника»

Метрики «кількість

вилучених рядків коду» сильно корелює з метриками «кількість доданих коментарів» і «кількість порожніх доданих рядків»

Назва проекту	Тип залежності	Рівняння	R ²
Apache Subversion	поліноміальна	$y = -8E-06x^2 + 0.5895x - 11467$	0.6035
Ruby	поліноміальна	$y = -5E-07x^2 + 0.051x - 1122$	0.7310
PostgreSQL	поліноміальна	$y = -3E-07x^2 + 0.0229x - 439.21$	0.1385
Firefox	поліноміальна	$y = 3E-05x^2 - 2.2151x + 41126$	0.6016

На основі даних можна зробити висновки про те, що історія роботи над проектом повинна включати в себе фазу швидкого зростання, коли проекту активно розвивається, фазу більш стабільного зростання, коли реалізується значна частина функцій проекту і фазу спаду активності, коли активність роботи над проектом зменшується і зводиться до виправлення помилок і реалізації дрібних функцій

25

ВИСНОВКИ

Першим завданням став аналіз репозиторіїв Open Source проектів і вибір найбільш підходящого.

Було прийнято рішення використовувати API сайту Ohloh.net, так як саме він надає найбільш повну інформацію про історії роботи над проектами. Був проаналізований API даного сайту і розроблено програмне забезпечення яке дозволяє запитувати необхідну інформацію.

На основі даних, отриманих в процесі аналізу проблеми, було виконано планування експерименту.

Вхідними даними експерименту є набір метрик, що характеризує історію роботи над проектом

Обробка отриманих даних включає перетворення отриманих даних до плоского узві, застосування методів математичної статистики для аналізу отриманих результатів.

На основі плану експерименту була виконана алгоритмізація методів обробки отриманих даних.

Були розглянуті алгоритми розрахунку метрик вихідного коду, підібрані алгоритми для статистичного аналізу отриманих показників метрик. Також були обрані набір інструментів для проведення дослідження.

Вхідні дані експерименту були проаналізовані відповідно до розробленого плану. Було виконано статистичний аналіз метрик та розглянуто їх динаміка. Розрахована кореляція між метриками, знайдені пари сильно залежних метрик, обґрунтовані причини наявності залежностей.

Були висунуті гіпотези про різні аспекти розвитку проектів і виконана перевірка даних гіпотез.

На підставі цих відомостей були сформульовані висновки як про поточний стан так і перспективи розвитку проектів.

В ході даної роботи були виконані всі етапи експериментального дослідження, що дозволяє зробити висновок про повну відповідність роботи поставленому завданню.

26

ДОДАТОК В
Апробація роботи

CERTIFICATE

is awarded to

Svirchkov Volodymyr

for being an active participant in
IX International Scientific and Practical Conference

**“TOPICAL ISSUES OF THE DEVELOPMENT
OF MODERN SCIENCE”**

24 Hours of Participation

SOFIA

6-8 May 2020

sci-conf.com.ua

