Thank You!

Your order number is: 3I39126393

Order Purpose
Personal Consumption

Order Information

Order Date
28-03-2023

Order Status
Submitted

Order Purpose
Personal Consumption

Shipping Details

Order(s) will be shipped from the warehouse from Monday to Saturday. The estimated shipment time is within 5-7 working days from the date of order confirmation.

Please be informed that there would be a delay in Shipment Dispatches by 4-5 additional business days due to unavoidable circumstances. We are trying our best to ship all the pending orders. There will be a delay in updating the shipment status on Click post.

Thank you for your patience.

Ship To
Shakeena Bano
Laxmi Market Janta Sweet Shop
Lucknow, UTTAR PRADESH, 226010
(873) 694-7706
Personal

SMS & Email Notification
shakeenahrl@gmail.com

Associate Number
8736947706

Invoice Option
Ship Invoice with Package

Volume Details

Order Month:	**Mar 2023**
Order Month Volume:	**83.10**
Total Volume for This Order:	**142.95**
Product Earn Base:	**₹11,052.00**

Order Details

Cart Retail Total:	**₹13,268.00**
Discount 35%:	**- ₹3,868.20**
Delivery Charges:	**₹0.00**
CGST	**₹845.96**
SGST	**₹846.0**
Grand Total:	**₹11,091.76**

Payment Method

Amount

₹ 11,091.76

ABOUT UPI PAYMENTS:

- As per Herbalife policy, order payment must be done by the purchaser only.
- QR code payment limit is maximum Rs.100000/- per day.
- Please choose your QR app from your mobile, scan the QR code and make the payment.
- Order will be applied and released immediately, once the payment is successful.

Cart Items

ITEM

Cell-U-Loss 90 Tablets
SKU 0111
Volume: 15.75
Earn Base: ₹ 1,305.00

QTY	Unit Price	Retail Price	Your Price
1	₹ 1,508.00	₹ 1,508.00	₹ 1,051.25

Herbalife Calcium Tablets
SKU 0020
Volume: 10.25
Earn Base: ₹ 921.00

QTY	Unit Price	Retail Price	Your Price
1	₹ 1,064.00	₹ 1,064.00	₹ 741.65

Cell Activator New 60 Tablets
SKU 3123
Volume: 21.95
Earn Base: ₹ 1,696.00

QTY	Unit Price	Retail Price	Your Price
1	₹ 1,960.00	₹ 1,960.00	₹ 1,366.40

Formula 1 Nutritional Shake Mix kulfi 500 g
SKU 4114
Volume: 43.50
Earn Base: ₹ 1,668.00

QTY	Unit Price	Retail Price	Your Price
2	₹ 1,928.00	₹ 3,856.00	₹ 2,688.40

Personalized Protein Powder 200 g
SKU 1233
Volume: 23.00
Earn Base: ₹ 991.00

QTY	Unit Price	Retail Price	Your Price
2	₹ 1,145.00	₹ 2,290.00	₹ 1,596.30

ShakeMate
SKU 183K
Volume: 12.90
Earn Base: ₹ 284.00

QTY	Unit Price	Retail Price	Your Price
2	₹ 577.00	₹ 1,154.00	₹ 955.20

Afresh Energy Drink Mix Ginger 50 g
SKU 1291
Volume: 15.60
Earn Base: ₹ 622.00

QTY	Unit Price	Retail Price	Your Price
2	₹ 718.00	₹ 1,436.00	₹ 1,000.60

Subtotal (Excluding tax and Delivery Charges) ₹9,399.80
Total Volume: 142.95
Product Earn Base: ₹ 11,052.00

FSSAI License number : 10013043000639

en-IN | 27-03-2023 22:51:45 | zuswpwssb000001 | MyHL | 28-03-2023 11:21:45 | 1.23.0323.06

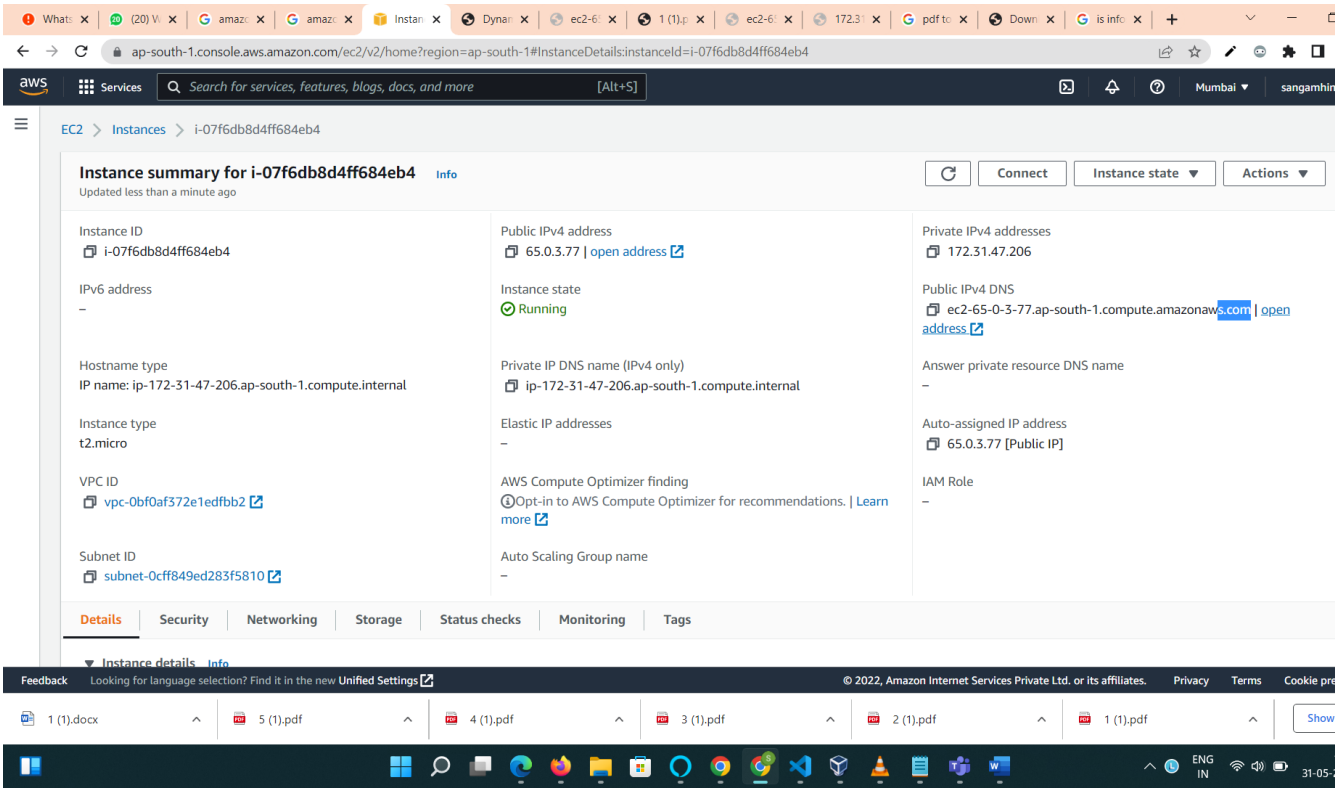
DevOps Lab Assignment -1

Name: SANGAMMISHRA

Sec: S

Roll no.: 26

Objective: Develop a Dynamic website using PHP and upload it on cloud platform.

1. Login to your AWS Account.
2. Launch an EC2 Instance:
   a. **Choose Amazon Linux 2 AMI (HVM), SSD Volume Type (Free Tier only). Click on Select.**
   b. **Select t2. Click on Next.**
   c. **Set all the default values. Click on Next.**
   d. **Set all default values. Click on Next.**
   e. **Click on Add Tags. Give "Name" as Key and "Any Name" as Value. Click on Configure Security Group.**
   f. **Click on Add Rule. Select HTTP, TCP, 80, Custom, 0.0.0.0/0, ::/0. Click on Review and Launch.**
   g. **Click on Launch.**
   h. **Create a new Key-Pair with a new name. Download Key-Pair. Click on Launch Instance.**
   i. **Your instance is now getting initialized and will be in running state after few seconds.**
   j. **Wait until Instance Status for your instance reads as Running before continuing.**
3. Go to EC2-Dashboard→Instances. Click on Newly created instance.
4. 

5. Click on Connect on the upper right corner. Click Connect button on the bottom right on the new page. You can see your console on the browser.
6. Update the Linux by giving the command: sudo yum update –y
7. If it shows an error of "cannot find a baseurl for repo", then do the following:
   (a) Give permission to write in the file: sudo chmod 777 /etc/resolv.conf
   (b) Open the file using vi editor: vi /etc/resolv.conf
   (c) Add these entries to /etc/resolv.conf:
   nameserver 8.8.8.8

nameserver 8.8.4.4

(d) Save and exit using pressing Esc, followed by typing :wq, and press Enter Key.

(e) Go to Step 5

8. Install PHP on your Linux console: sudo amazon-linux-extras install -y php7.2

9. Install Apache on the EC2-Instance: sudo yum install -y httpd
10. Start the web server with the command shown following: sudo systemctl start httpd
11. Enable the service: sudo systemctl enable httpd
12. Check whether the httpd service is started or not: systemctl status httpd (An active running value of Active field indicates running)
13. Open the Apache Home page by entering the Public IPv4 DNS of your instance. Make sure it is using http protocol and not https protocol.
14. Add the www group to your EC2 instance with the following command: sudo groupadd www
15. Add the ec2-user user to the www group: sudo usermod -a -G www ec2-user
16. Log out to refresh your permissions and include the new www group: exit
17. Log back in again and verify that the www group exists with the groups command.
    groups
18. Your output looks similar to the following:
    ec2-user adm wheel systemd-journal www
19. Change the group ownership of the /var/www directory and its contents to the www group.
    sudo chgrp -R www /var/www
20. Change the directory permissions of /var/www and its subdirectories to add group write permissions and set the group ID on subdirectories created in the future.
    sudo chmod 2775 /var/www
    find /var/www -type d -exec sudo chmod 2775 {} +
21. Recursively change the permissions for files in the /var/www directory and its subdirectories to add group write permissions.
    find /var/www -type f -exec sudo chmod 0664 {} +
22. Change the directory to /var/www/html.
    cd /var/www/html
23. Create a new file in the html directory named index.html as follows:

```
<html>
<head>
<title>Dynamic Website</title>
<head>
<body>
<form method=get  action=result.php>
<table border=2>
<tr>
<td>Name</td>
<td><input type=text name=t1></td>
</tr>
<tr>
<td>Gender</td>
<td><input type=radio name=gen value=male>Male<br>
<input type=radio name=gen value=female>Female </td>
</tr>
<tr>
<td colspan=2><input type=submit value=OK> </td>
</tr>
</table>
</form>
```

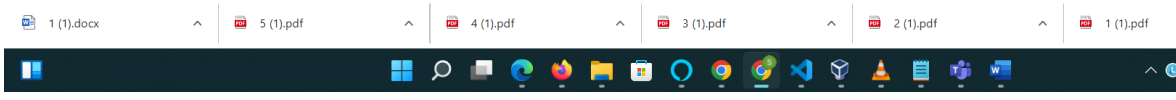
```
<footer>&copy;Copyright 2022 MAYANK SRIVASTAVA</footer>
    </body>
    </html>
```

24. Note that the above file is a form that will send the data to result.php. So, another file needs to be created with the name result.php as follows:

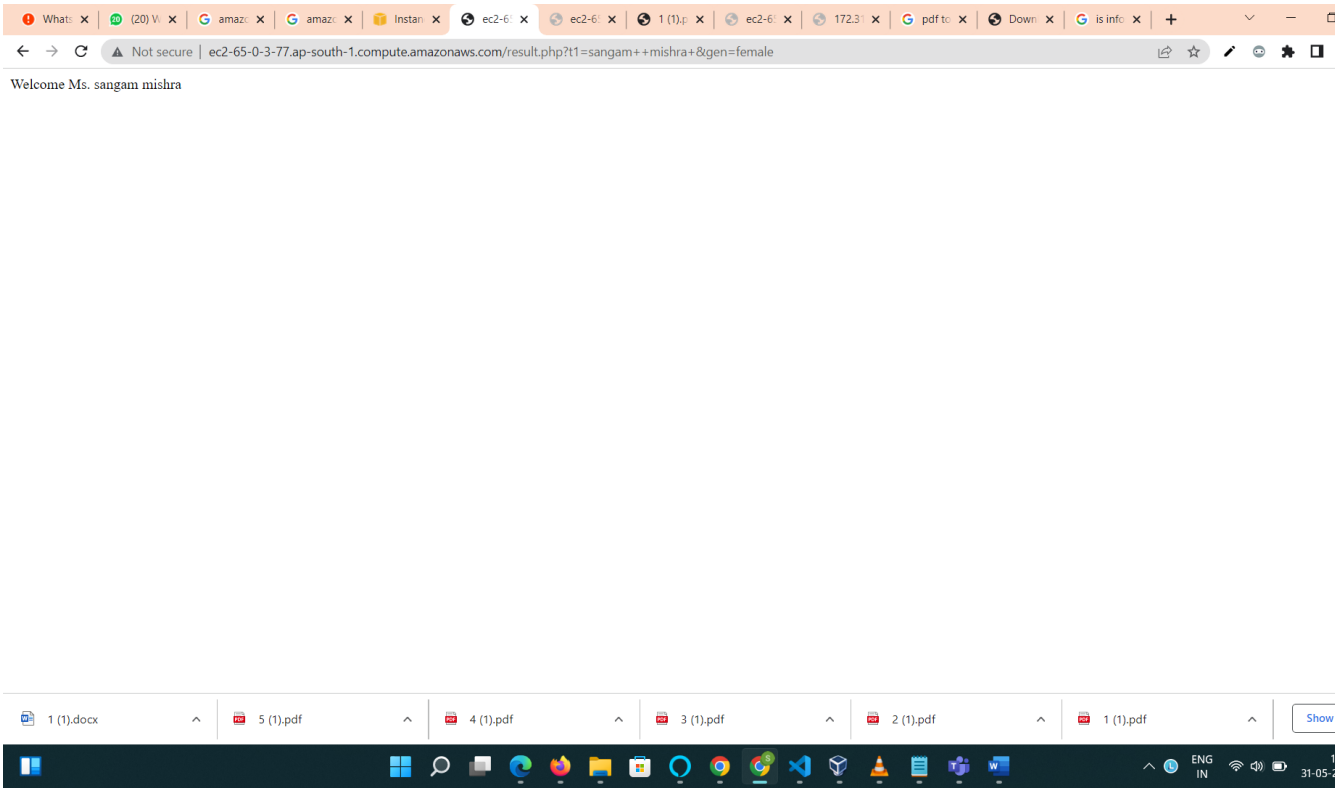
```
<?php
$name=$_GET['t1'];
$gen=$_GET['gen'];
if ($gen=="male")
{
echo "Welcome Mr. $name";
}
else
{
echo "Welcome Ms. $name";
}
?>
```

25. Now open **Public IPv4 DNS** of your EC2 Instance (For example: ***http://ec2-54-191-252-237.us-west-2.compute.amazonaws.com)*** and you will find your index.html page on the browser. After filling the form, press the OK button and you will have your output of the result.php page in html format.
26. Finally, your dynamic website has been hosted.

HOMEPAGE



FINAL PAGE


Not secure | ec2-65-0-3-77.ap-south-1.compute.amazonaws.com/result.php?t1=sangam++mishra+&gen=female
Welcome Ms. sangam mishra
1 (1).docx
5 (1).pdf
4 (1).pdf
3 (1).pdf
2 (1).pdf
1 (1).pdf
Show
ENG
IN

SNEHA RAI

Elite ● Jhansi, 281002.
●sneha.rai_cs.da20@gla.ac.in ● +91 6388017902
https://www.linkedin.com/in/sneharai48/

EDUCATION

BTech in Computer Science, *GLA University, Mathura* 2020-2024

Intermediate, RNS WORLD School, Jhansi 2018-2020

High School, Saint Francis convent inter college, Jhansi 2016-2018

PROJECTS

Weather Forecasting
Machine Learning Oct 2022 - Nov 2022

- *Designed a ML model to predict weather report.*
- *Automate the model through api call.*

Sentiment analysis of twitter
Machine Learning Jun 2022 - July 2022

- *Analyzed the sentiment of tweets using NLP.*
- *It helped to understand view of a person by its tweets.*

SKILLS

Technical

- Programming languages: C, C++, Java, Python
- Program solving Skill
- Data Science.
- Data Structure and Algorithms.
- **Core:** OS and DBMS.

Professional

- Teamwork
- Ability to identify and solve problem
- Hardworking
- Leadership

EXTRA/CO-CURRICULAR ACTIVITIES

- Participated in "**National IBM Hackathon – CodeWars'23"** at **UPES, Dehradun**.
- Coordinated **IBM** Workshop.
- Attended workshop of "**Use of Git & GitHub in daily life**".
- Attended IBM workshop on fundamental of **data mining and predictive model**.
- Attended workshop on **enterprices**.
- Member of **Androkit** Club.

DECLARATION

I hereby declare that all the above-mentioned information is true and correct to the best of my knowledge.

Shubham Saxena
(191510028)

DevOps Lab

Experiment #1

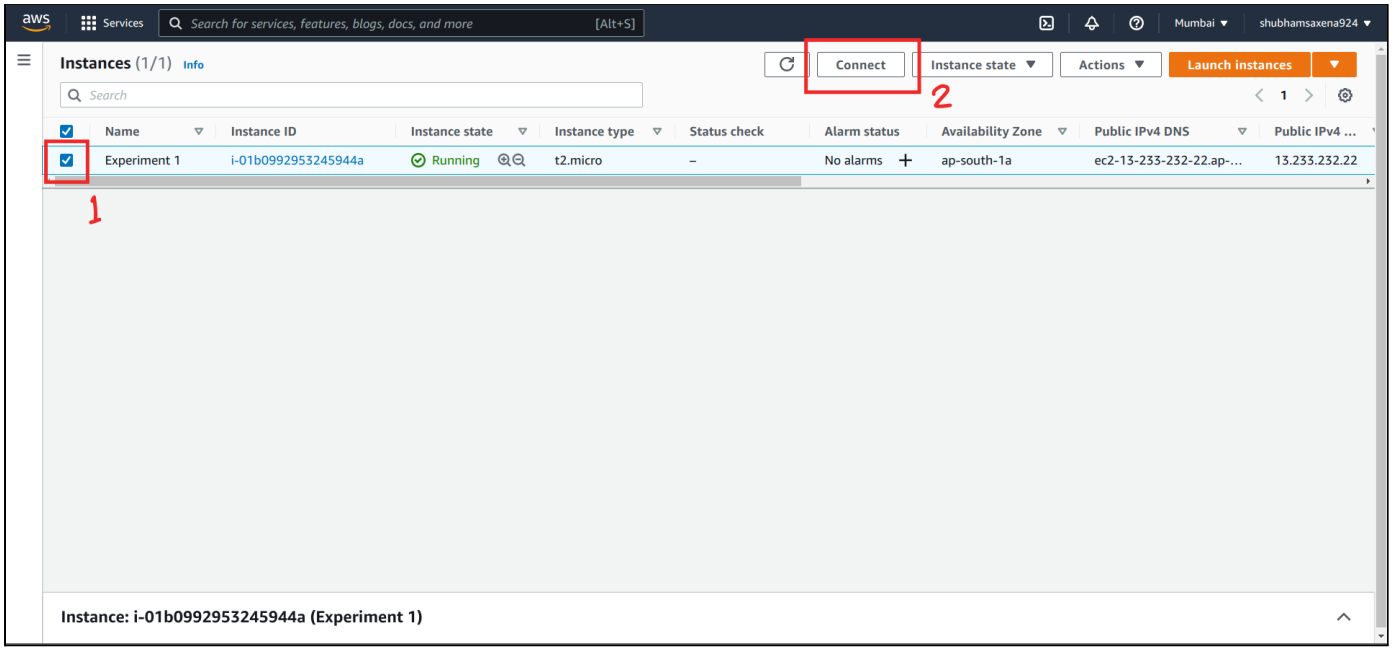
Dynamic Website on EC2

Task: Develop a dynamic website using PHP & host it on cloud (AWS)

Tools Used: VS Code, AWS EC2

Procedure:

1. Login to your Amazon Web Services account.
2. Launch an EC2 Instance:
    a. Start by giving your instance a name. You can also add additional tags to your instance.
    b. Choose Amazon Linux 2 AMI (HVM), SSD Volume Type (Free Tier).
    c. Select t2.micro in Instance Type.
    d. Create a new key-pair or you can use the existing one to log into your EC2.
    e. Configure security group rule: Click on add rule. Select HTTP, TCP, 80, Custom, 0.0.0.0/0, ::0. Enable SSH from anywhere.
    f. Click on 'Launch'.
    g. Your instance is now getting initialized and will be in running state after a few seconds.
    h. Wait until Instance Status for your instance reads as Running before continuing.
3. Go to EC2-Dashboard -> Instances.
4. Select the instance you just created and click on 'Connect' on the upper right corner.



5. Click the Connect button on the bottom right on the new page (EC2 Instance Connect). You can see your console on the browser. (You may also use other third party SSH Clients to connect to your instance.
6. Run the following commands in the console window of your instance.
    a. Update the Linux by giving the command: ***sudo yum update –y***
    b. If it shows an error of "cannot find a baseurl for repo", then do the following:
        i. Give permission to write in the file: ***sudo chmod 777 /etc/resolv.conf***
        ii. Open the file using vi editor: ***vi /etc/resolv.conf***
        iii. Add these entries to /etc/resolv.conf:

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

        iv. Save and exit using pressing Esc, followed by typing :wq, and press Enter Key.
        v. Go to Step 6.a
    c. Install PHP on your Linux console: ***sudo amazon-linux-extras install -y php7.2***
    d. Install Apache on the EC2-Instance: ***sudo yum install -y httpd***
    e. Start the web server with the command shown following: ***sudo systemctl start httpd***
    f. Enable the service: ***sudo systemctl enable httpd***
    g. Check whether the httpd service is started or not: ***systemctl status httpd*** (An active running value of Active field indicates running)
    h. Open the Apache Home page by entering the Public IPv4 DNS of your instance. Make sure it is using http protocol and not https protocol.
    i. Add the www group to your EC2 instance with the following command: ***sudo groupadd www***
    j. Add the ec2-user user to the www group: ***sudo usermod -a -G www ec2-user***
    k. Log out to refresh your permissions and include the new www group: ***exit***
    l. Log back in again and verify that the www group exists with the groups command: ***groups***
    m. Your output looks similar to the following: *ec2-user adm wheel systemd-journal www*
    n. Change the group ownership of the /var/www directory and its contents to the www group: ***sudo chgrp -R www /var/www***
    o. Change the directory permissions of /var/www and its subdirectories to add group write permissions and set the group ID on subdirectories created in the future: ***sudo chmod 2775 /var/www*** & ***find /var/www -type d -exec sudo chmod 2775 {} +***
    p. Recursively change the permissions for files in the /var/www directory and its subdirectories to add group write permissions: ***find /var/www -type f -exec sudo chmod 0664 {} +***
    q. Change the directory to /var/www/html: ***cd /var/www/html***
    r. Create a file named index.html in the current directory and write the following code in it.

```
<!DOCTYPE html>
<html>
 <head>
   <title>Exp1 Dynamic Website</title>
 </head>
 <body>
   <h3>Please fill out the form to continue!</h3>
   <form
     method="get"
     action="result.php"
     style="display: flex; flex-direction: column"
   >
     <div style="margin-top: 1rem">
       <label for="name">Name</label>
       <input id="name" type="text" name="t1" />
     </div>
     <div style="margin-top: 1rem">
       <label for="gen">Gender</label>
       <input type="radio" name="gen" value="male" />Male
       <input type="radio" name="gen" value="female" />Female
     </div>
     <input
       type="submit"
       value="Continue"
       style="margin-top: 1rem; width: max-content"
     />
   </form>
   <footer style="margin-top: 2rem">©Copyright 2022 Shubham Saxena
:)</footer>
 </body>
</html>
```

s. Write PHP Code (result.php)

```
<?php
$name=$_GET['t1'];
$gen=$_GET['gen'];
if($gen=='male') {
   echo "Welcome Mr. $name.";
} else {
   echo "Welcome Ms. $name."
}
?>
```

t. Now open the Public IPv4 DNS of your EC2 Instance (For example: http://ec2-54-191-252-237.us-west-2.compute.amazonaws.com) and you will find your index.html page on the browser. After filling the form, press the OK button and you will have your output of the result.php page in html format.
u. Finally, your dynamic website has been hosted.

HOME PAGE 👇

CLICKING ON “Continue”, THIS PAGE APPEARS 👇

Thank You :)

Shubham Saxena
(191510028)

B. Tech 3rd Year CS (CCV)

Experiment #2

GitHub

Task: Consider Yourself as a part of a project team. All the team members(at least 2) are working for the project from different geographical locations. You are advised to use a configuration management DevOps tool to create a central repository (on the cloud platform) of the code developed by the team members. Choose an appropriate DevOps tool and show the uploading of the code content in the central repository.

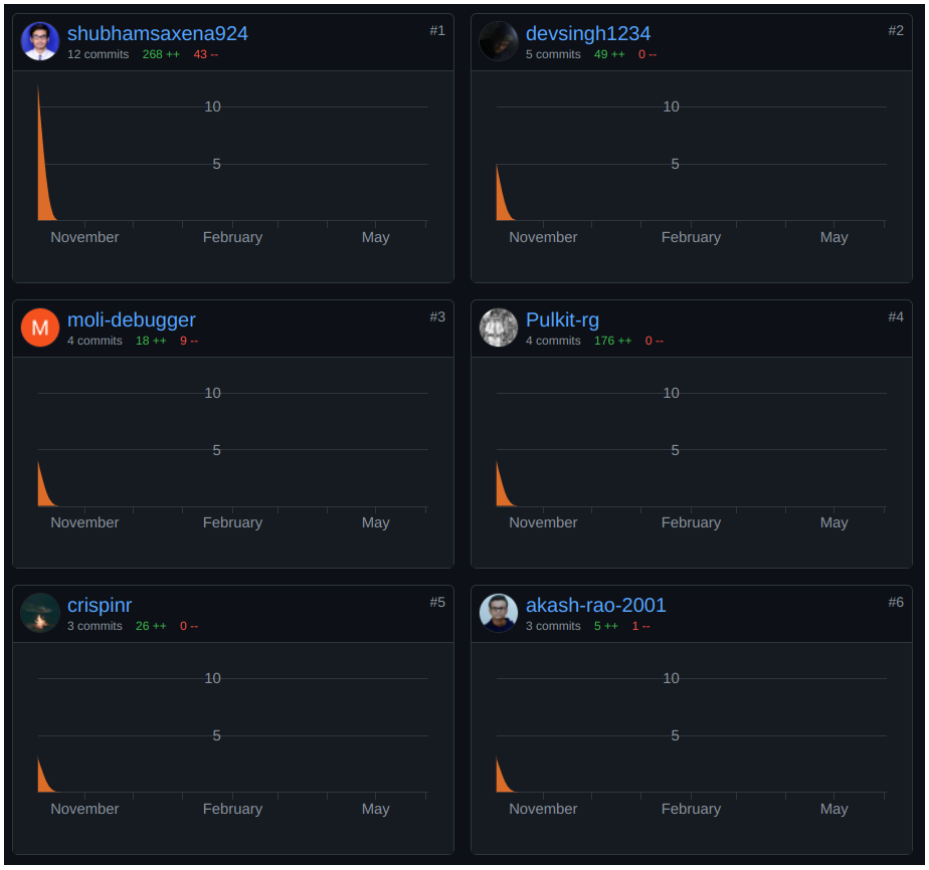
Tools Used: VS Code, Git, GitHub

Solution:

Here, the DevOps tool we are using is GitHub.

***GitHub** is a code hosting platform for version control and collaboration. It lets you and others work together on the project from anywhere. It allows for managing, maintaining, and collaborating on open-source programming projects.*

In my case, the collaborators are-

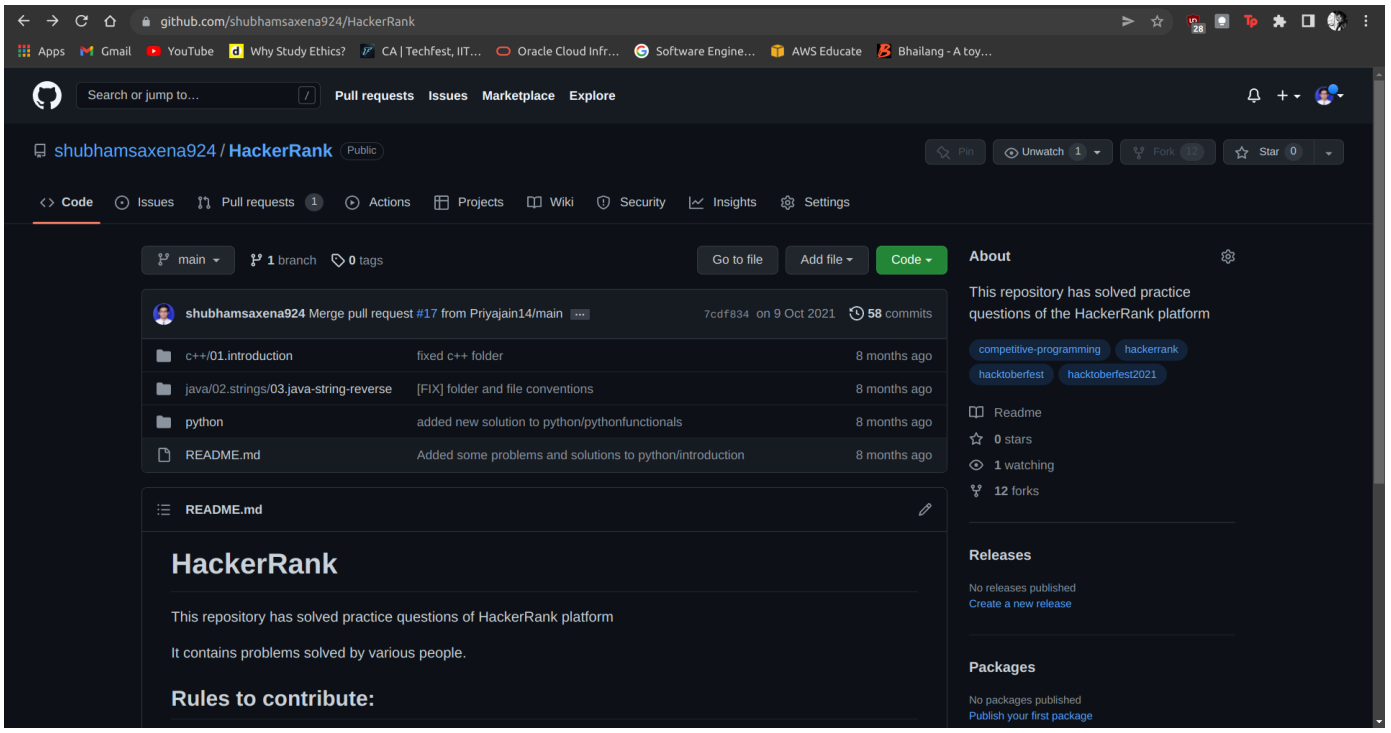


Shubham Saxena
(191510028)

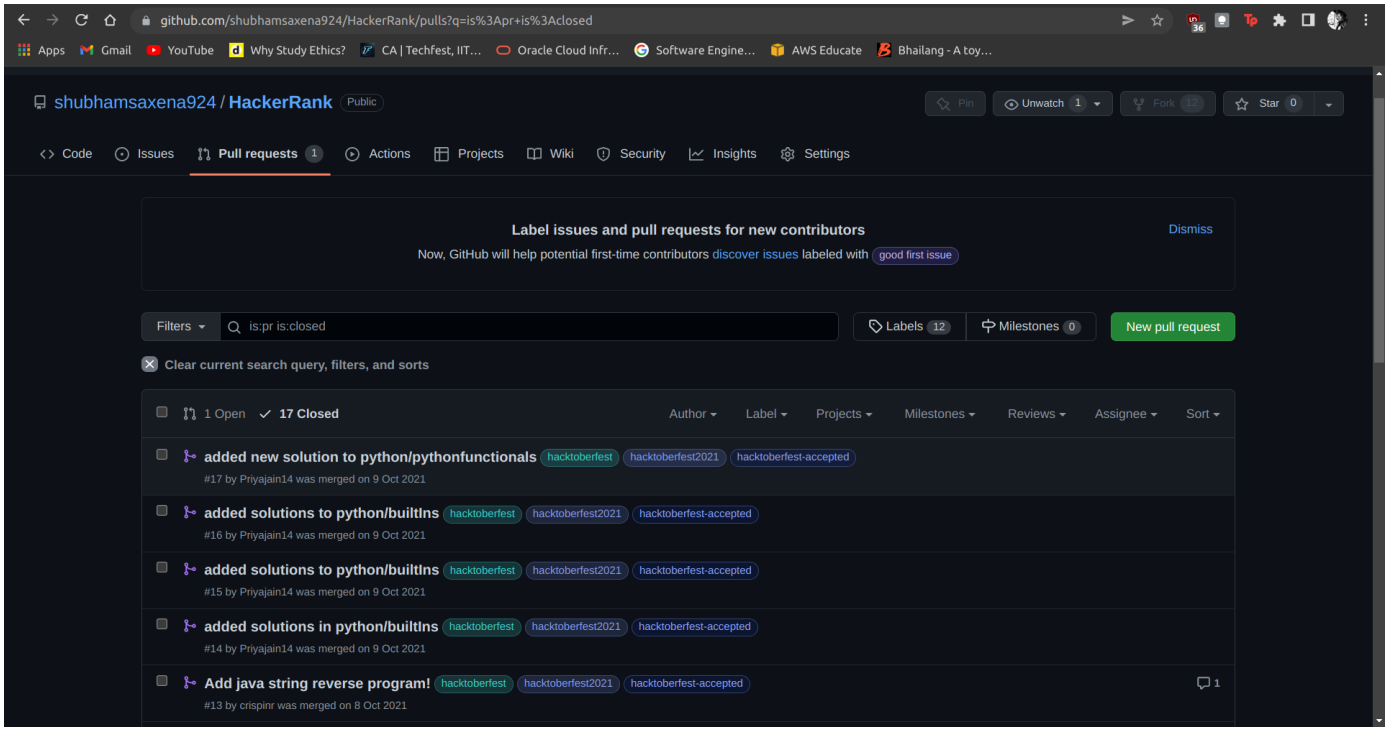
B. Tech 3rd Year CS (CCV)

The repository used is: https://github.com/shubhamsaxena924/HackerRank

This repository was used in Hacktoberfest as a open source contribution



Pull requests on the repository:



Thank You :)

Shubham Saxena
(191510028)

Experiment #3

Jenkins (Installation)

Task: Install Integration tool(Jenkins) in your machine to create a Jenkins Server. Configure Jenkins to create an environment for your upcoming project at the server using Windows Batch Commands.

Tools Used: JDK, Jenkins,

Prerequisite:

Continuous integration is a major component in the DevOps lifecycle. It makes the development, testing, and deployment of applications easier and faster.

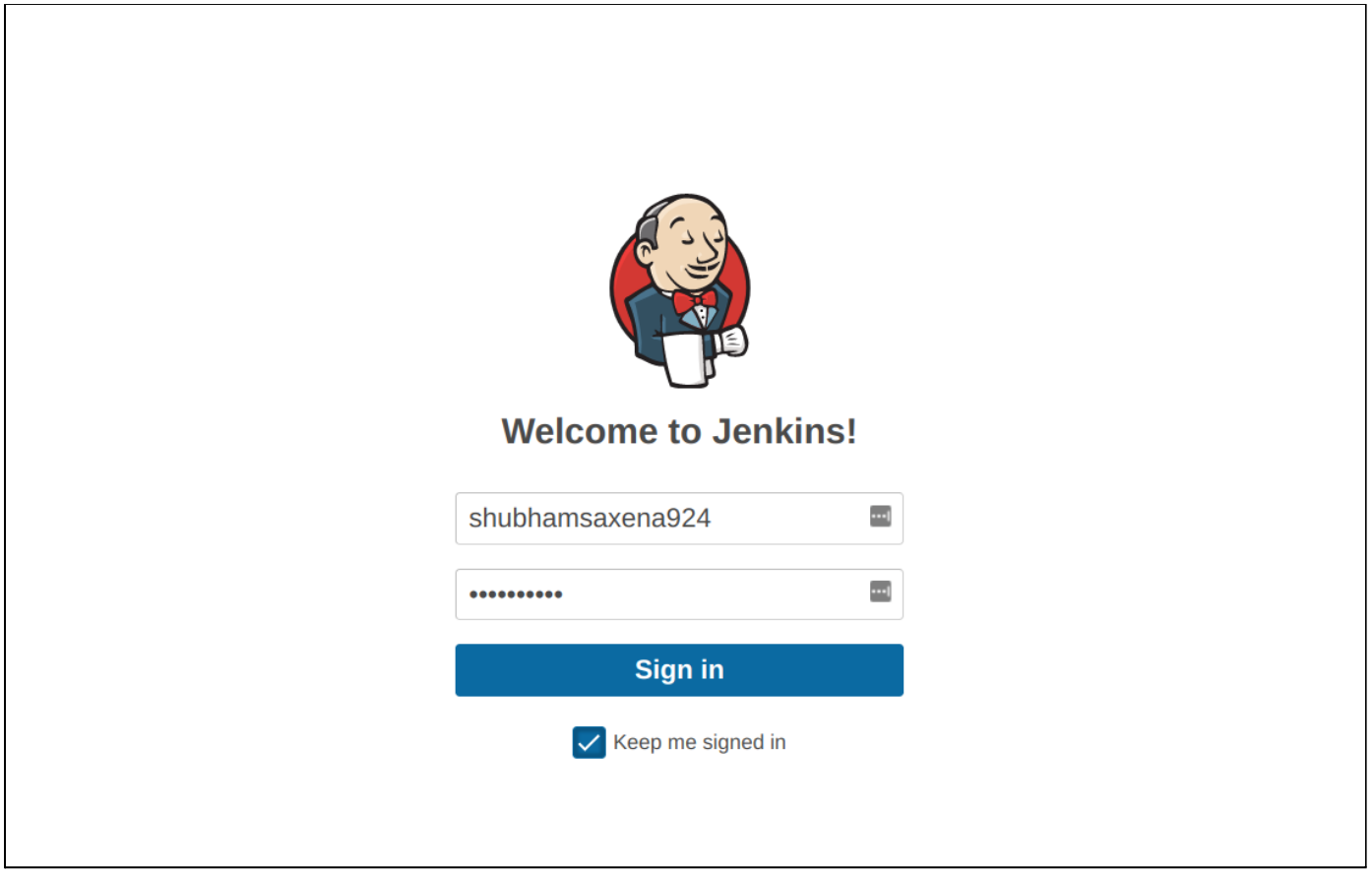
There are several continuous integration tools like Bamboo, Apache Gump, Jenkins, Buildbot, and Travis CI. Among them, Jenkins is the most popular continuous integration tool. Jenkins enables continuous development, testing, and deployment of newly created codes.

Procedure:

1. Install Java Development Kit (JDK)

   Download JDK 8 and choose windows 32-bit or 64-bit according to your system configuration.

2. Set the Path for the Environmental Variable for JDK
   a. Go to System Properties. Under the "Advanced" tab, select "Environment Variables."
   b. Under system variables, select "new." Then copy the path of the JDK folder and paste it in the corresponding value field.
3. Download and Install Jenkins
   a. Download Jenkins for windows.
   b. After the file is downloaded, unzip it. Click on the folder and install it. Select "finish" once done.
4. Run Jenkins on localhost:8080
   a. Once Jenkins is installed, explore it. Open the web browser and type "localhost:8080".
   b. Enter the credentials and log in.
   c. If you install Jenkins for the first time, the dashboard will ask you to install the recommended plugins. Choose None.



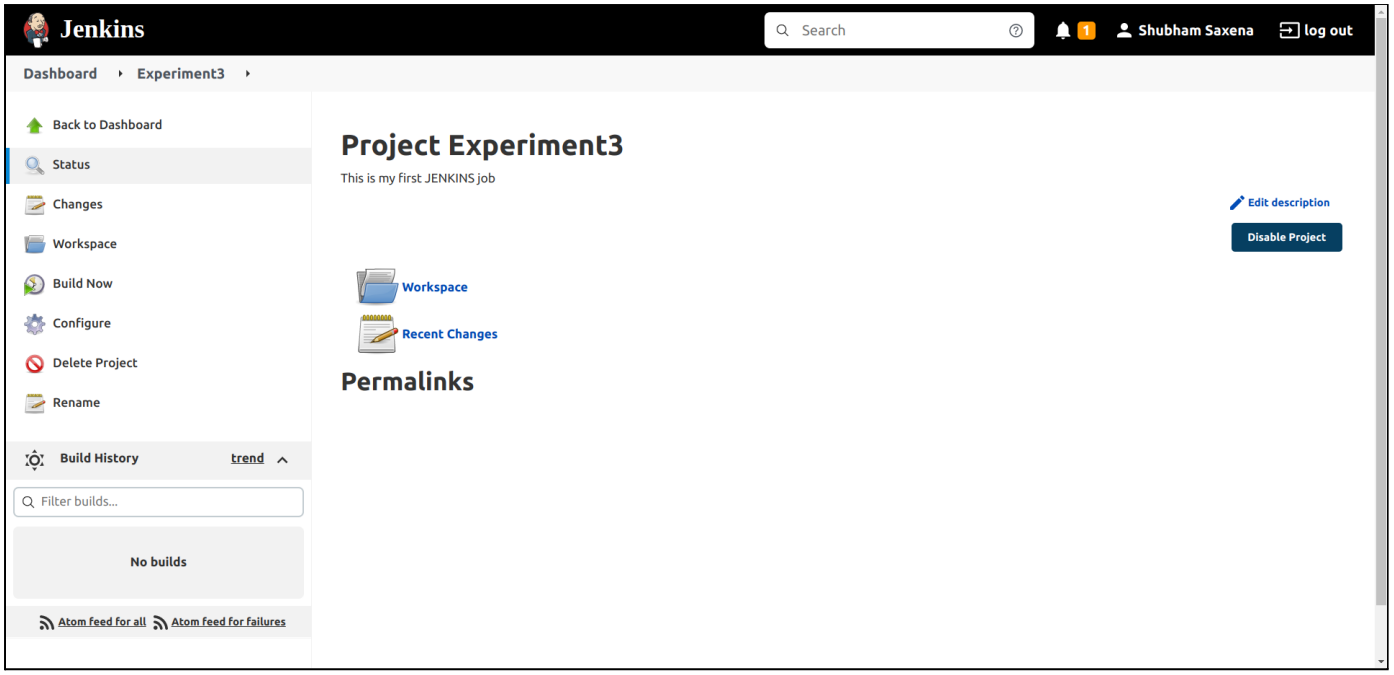
5. Jenkins Server Interface:
    a. New Item allows you to create a new project.
    b. Build History shows the status of your builds.
    c. Manage System deals with the various configurations of the system.
6. Build and Run a Job on Jenkins
    a. Select a new item (Name – Practical on Jenkins).
    b. Choose a freestyle project and click Ok.
    c. Under the General tab, give a description like "This is my first Jenkins job."
    d. Under the "Build Triggers" tab, select add build step and then click on the "Execute Windows" batch command.
    e. In the command box, type the following:

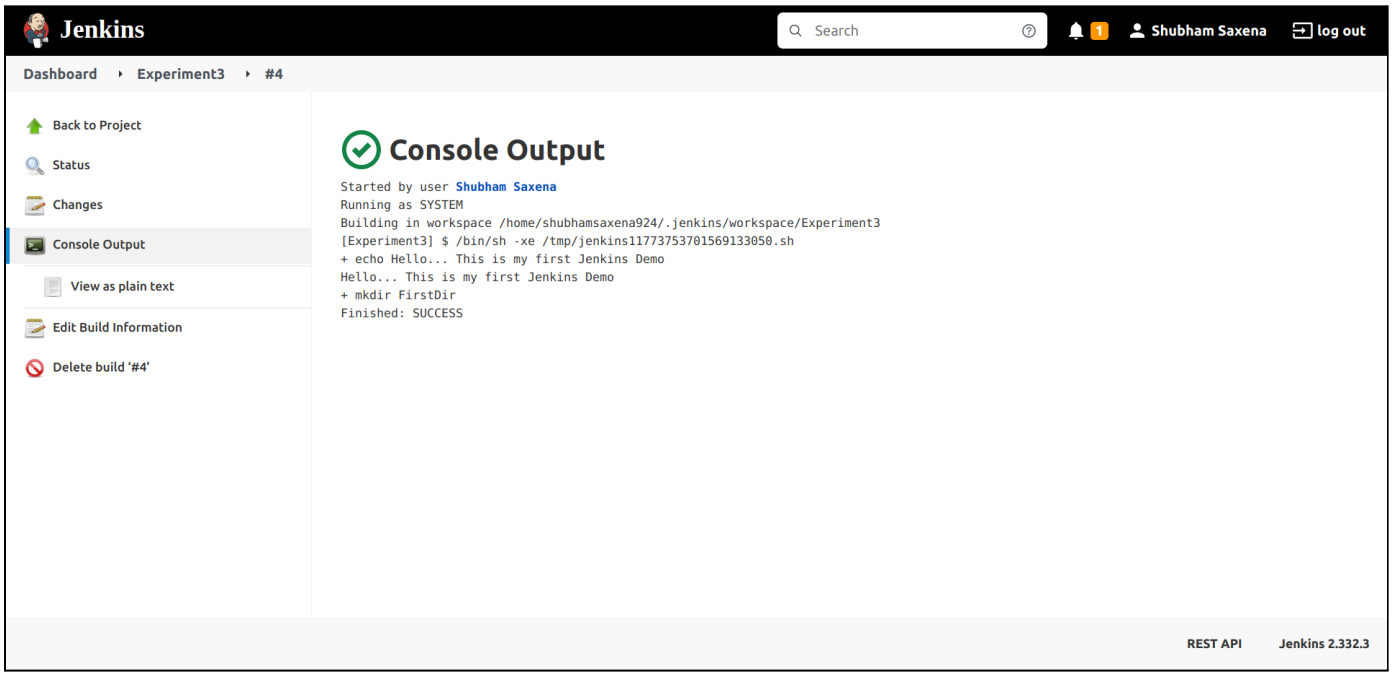
        echo "Hello... This is my first Jenkins Demo: %date%:%time% "

    f. Click on apply and then save.
    g. Select build now. You can see a building history has been created. Click on that. In the console output, you can see the output of the first Jenkins job with time and date.

Shubham Saxena
(191510028)

B. Tech 3rd Year CS (CCV)





Thank You :)

Shubham Saxena
(191510028)

Experiment #4

Jenkins (CI/CD Pipeline)

Task: Create an automated Continuous Integration-Continuous deployment (CI-CD) pipeline using the following tools:

(a) GitHub as Source Code Management tool (b) Jenkins as Continuous Integration tool (c) Amazon Web Services as Deployment Server

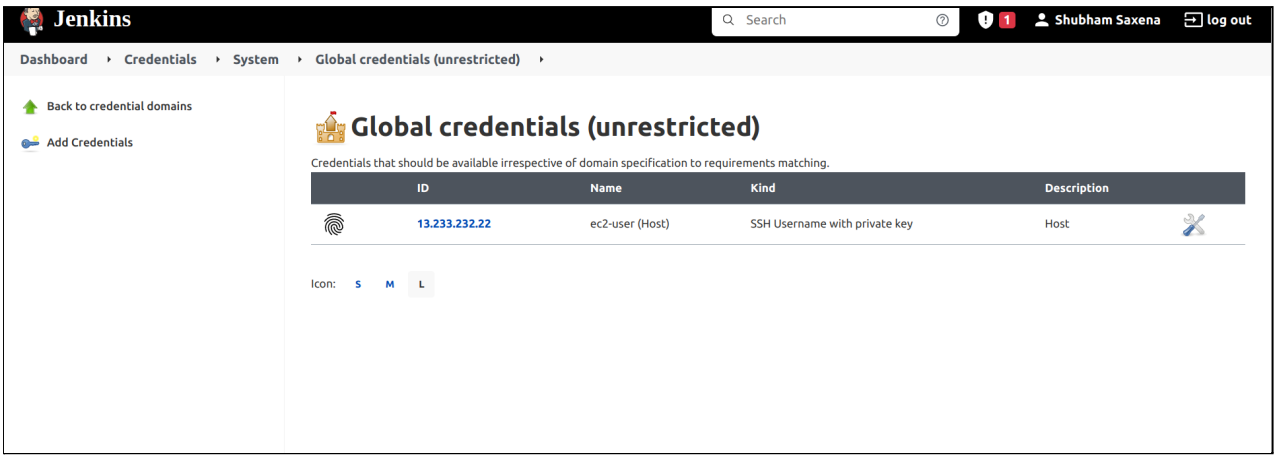
Tools Used: VS Code, Jenkins, Git, GitHub, AWS EC2

Prerequisite:

First go to "manage plugins". Then install the plugins: *publish over ssh, ssh credential plugin, ssh plugin, ssh server, git plugin*

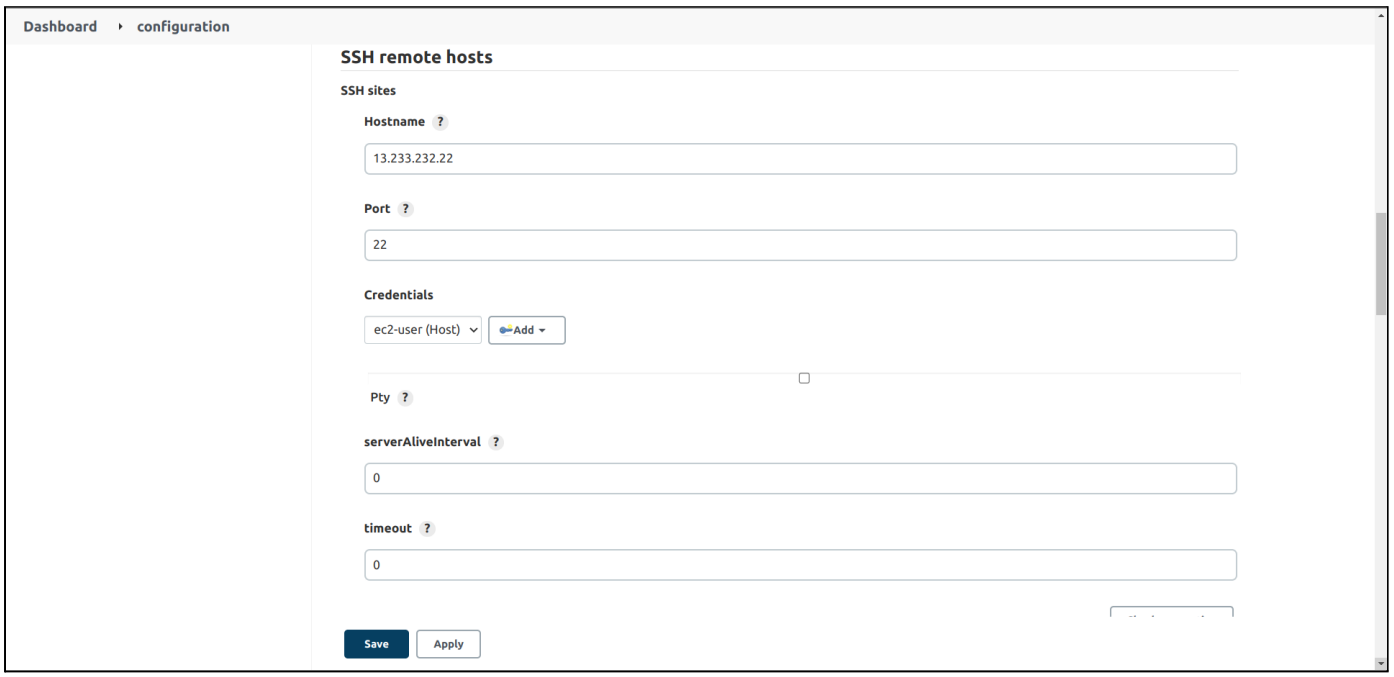
Procedure:

1. Go to manage credentials
2. Click global
3. Click "Add credentials" option present just below the Dashboard
4. In the option kind click ssh username with private key. Select ok
5. In the option id mention the public IP of the ec2-instance
6. In the option description write anything. For example: host or anything
7. In the option username: *ec2-user.* In private key: selected "enter directly"
8. In the box key paste the content of the PEM file which is the Private key
9. Leave the passphrase blank
10. Click ok



11. Again go to the option Manage Jenkins

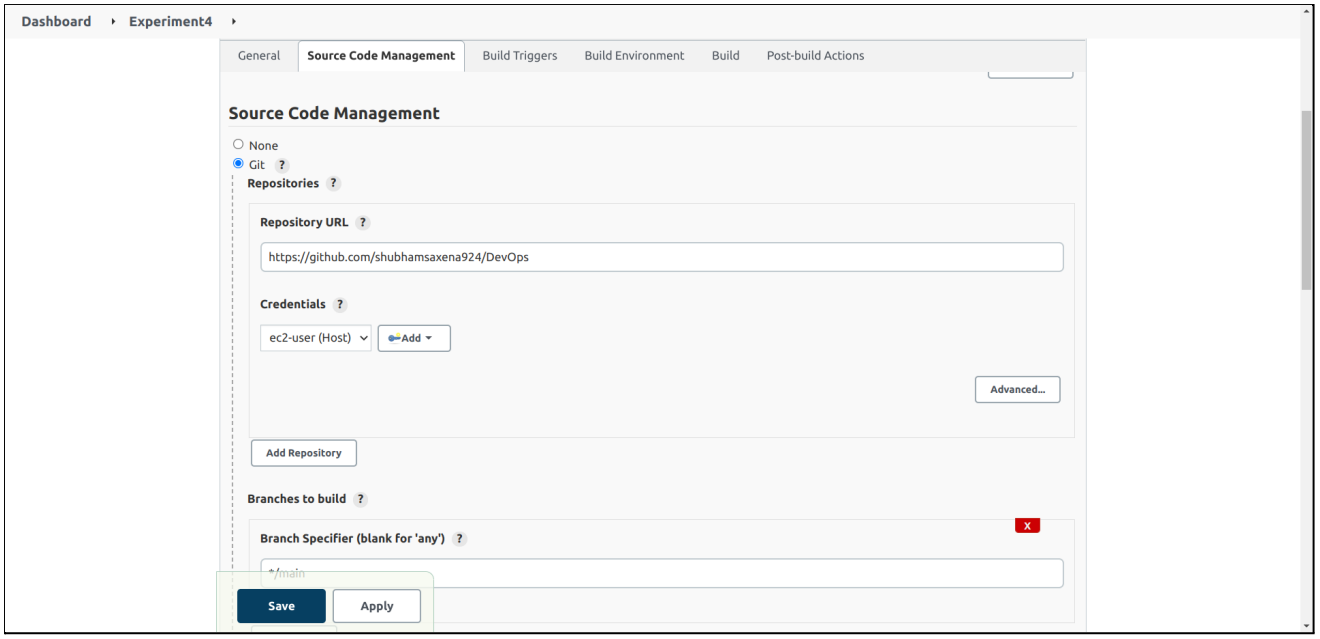
12. Click configure system
13. Find the option SSH remote hosts
14. SSH sites - click Add
15. Hostname - add public IP, port 22, in credentials you will find the option ec2-user(host)
16. leave blank last two option
17. Check the connection
18. Click save in usage statistics below

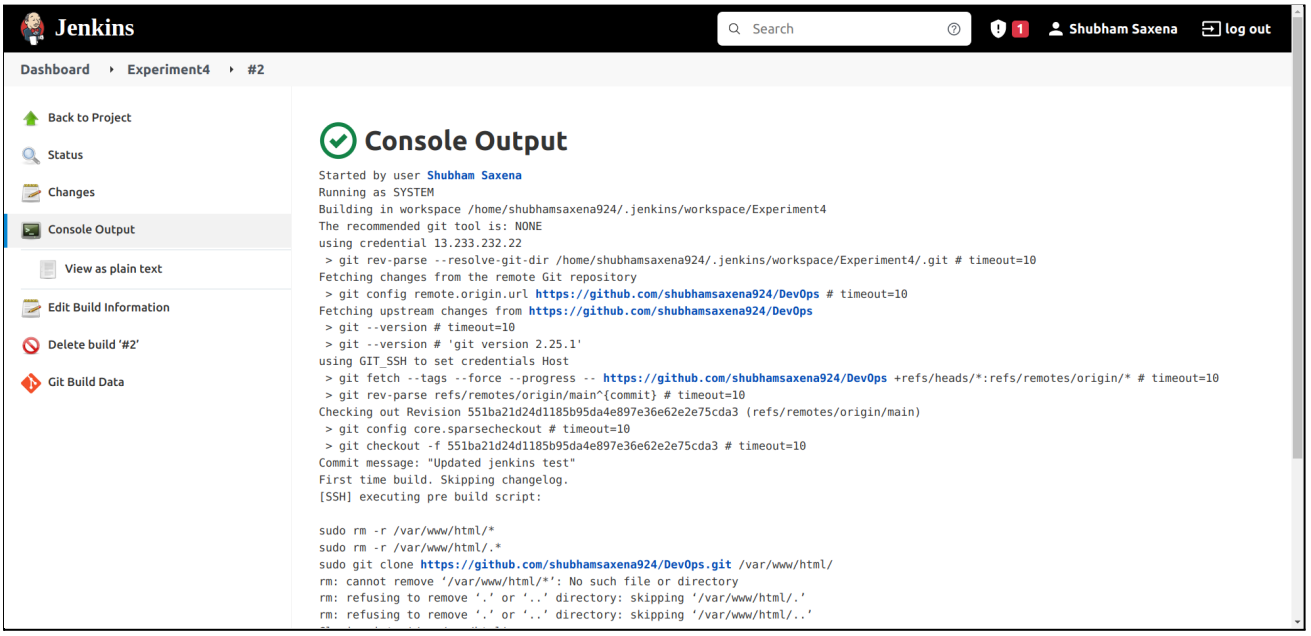


19. Create the freestyle project
20. Source code management - select git (You may need to install git plugin first)
21. Repository URL – paste the repo of your GitHub repository
22. In the option Branches to build - select the same branch as on GitHub
23. In the build option select - execute shell scripts on remote hosts using ssh
24. In the ssh site you will find ec2-user…
25. In the command box write the scripts

```
sudo rm -r /var/www/html/*
sudo rm -r /var/www/html/.*
sudo git clone (repo link) /var/www/html/
```

Shubham Saxena
(191510028)





If git is not already installed on your linux machine, then install it before building.

sudo yum install git -y for Amazon Linux

Thank You :)

Shubham Saxena
(191510028)

Experiment #5

Docker (Installation)

Task: Do as Directed:

1. Install Docker Desktop
2. Pull Ubuntu Container
3. Create a directory and some files using Ubuntu commands
4. Create your own Container
5. Push the created Container in Docker hub.
6. Ask your colleague to pull your created container and work on it.

Tools Used: VMWare, Ubuntu, Docker, DockerHub

Procedure:

1. Starting a container

a.	docker pull ubuntu: Pull the ubuntu image from docker hub.
b.	docker images -a: *To show all the images pulled on the system*
c.	docker run -it *image-name*: To run container of a given image

2. Task Inside the Ubuntu Container
   a. Create a directory named 'CCV'
   b. Create 10 different txt files in the directory and add some content
3. Creating custom image from container

a.	docker commit *container-name new-image-name*: Create a custom image of the container with all the files inside it.

```
shubham@shubham-VirtualBox:~$ sudo su
[sudo] password for shubham:
root@shubham-VirtualBox:/home/shubham# docker container ls
CONTAINER ID   IMAGE       COMMAND   CREATED          STATUS          PORTS     NAMES
41d14bd57669   lab_image   "bash"    28 minutes ago   Up 28 minutes             ecstatic_bhaskara
```

```
root@shubham-VirtualBox:/home/shubham# docker commit ecstatic_bhaskara lab-image-with-files
sha256:cf89aa8a455db0377f98934c8af079945cfbc2a777c5a9c2d354894179d6fd60
```

4. Tagging the image

Name your local images using one of these methods:

- When you build them, using docker build -t <hub-user>/<repo-name>[:<tag>]

- By re-tagging an existing local image docker tag <existing-image> <hub-user>/<repo-name>[:<tag>]

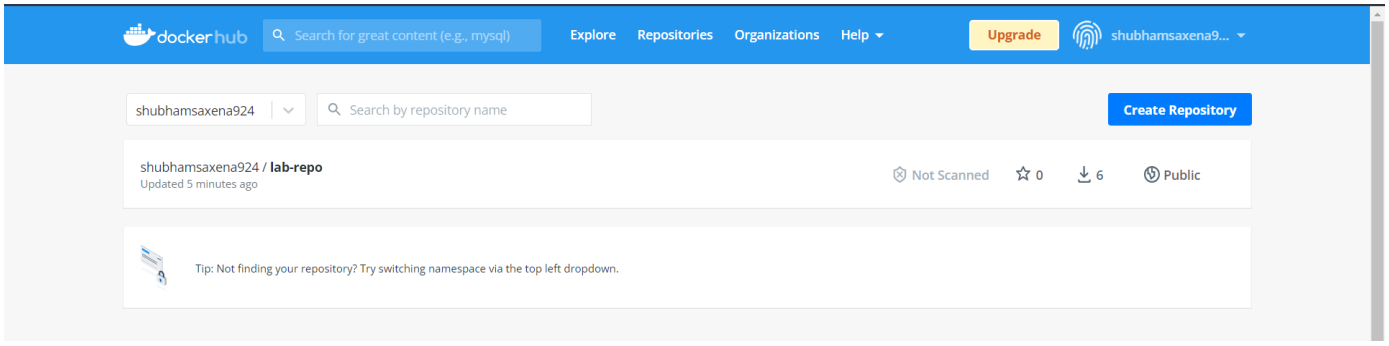
- By using docker commit <existing-container> <hub-user>/<repo-name>[:<tag>] to commit changes

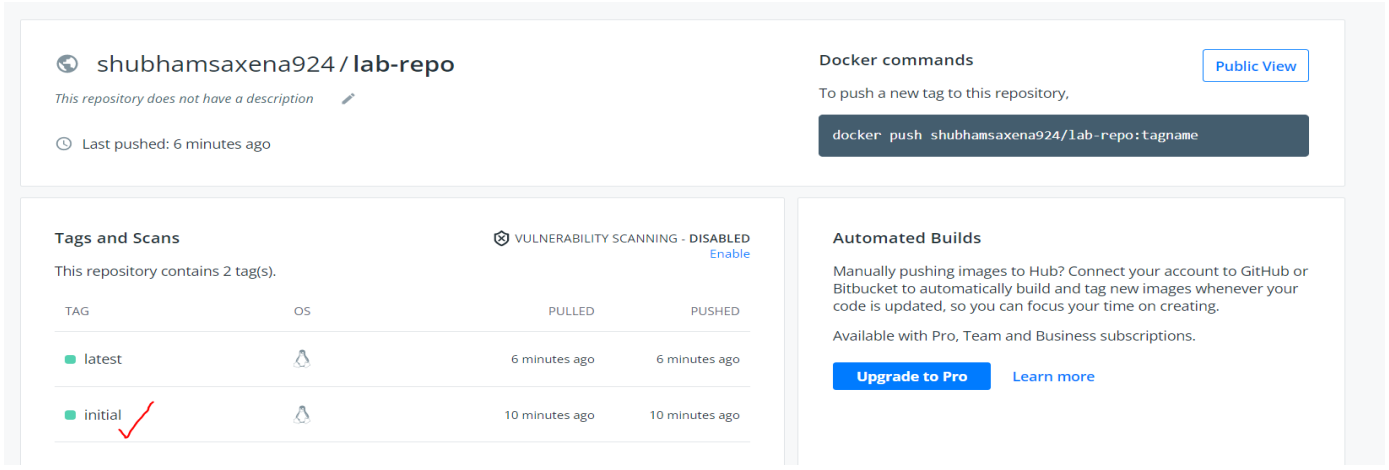
```
root@shubham-VirtualBox:/home/shubham# docker tag lab-image-with-files shubhamsaxena924/lab-repo:initial
root@shubham-VirtualBox:/home/shubham# docker images -a
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
lab-image-with-files        latest    cf89aa8a455d   7 minutes ago    72.8MB
shubhamsaxena924/lab-repo   initial   cf89aa8a455d   7 minutes ago    72.8MB
lab_image                   latest    336df82e152a   35 hours ago     72.8MB
<none>                      <none>    98dfe8f6e9bb   7 days ago       305MB
input_ubuntu_image          latest    da3701e3e764   7 days ago       163MB
<none>                      <none>    009317a7c5ce   8 days ago       163MB
ubuntu                      latest    ba6acccedd29   5 weeks ago      72.8MB
centos                      latest    5d0da3dc9764   2 months ago     231MB
```

5. Pushing the image

a. docker login: Login to your docker hub account.
b. docker push *username/reponame:tagname* : To push your image to the docker hub

```
root@shubham-VirtualBox:/home/shubham# docker push shubhamsaxena924/lab-repo:initial
\The push refers to repository [docker.io/shubhamsaxena924/lab-repo]
e5ee8c89361f: Pushed
9f54eef41275: Mounted from library/ubuntu
initial: digest: sha256:efbe236723ad99162b6eb6898a52bf76c2d832f2919b587b05ab1e18999ec92e size: 736
```





Thank You :)

Shubham Saxena
(191510028)

Experiment #6

nginx & Mario

Task: Configure two application servers namely nginx and Mario using YAML file on port numbers 81 and 8600 respectively using Docker-compose. Show the execution of both of them in your browser.

Tools Used: AWS EC2, Docker

Prerequisite:

1. Create AWS EC2 Instance
2. Install docker on the same. Follow steps given below for Amazon Linux 2:
    a. Update the installed packages and package cache on your instance: ***sudo yum update -y***
    b. Install the most recent Docker Engine package: ***sudo amazon-linux-extras install docker***
    c. Start the Docker service: ***sudo service docker start***
    d. To ensure that the Docker daemon starts after each system reboot, run the following command: ***sudo systemctl enable docker***
    e. Add the ec2-user to the docker group so you can execute Docker commands without using sudo: ***sudo usermod -a -G docker ec2-user***
    f. Log out and log back in again to pick up the new docker group permissions. Verify by command: ***groups***
    g. Test by running any docker command without sudo: ***docker info***
3. Also install docker-compose:
    a. ***sudo curl -L https://github.com/docker/compose/releases/download/1.20.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose***
    b. ***sudo chmod +x /usr/local/bin/docker-compose***

Procedure:

1. Change directory to where you want to create your docker compose file.
2. Create a file docker-compose.yml
3. Type the following configuration into the file

```
version: "3"
services:
 nginx:
   image: "jc21/nginx-proxy-manager:latest"
   container_name: nginx
   restart: unless-stopped
```

```
  ports:
    # These ports are in format <host-port>:<container-port>
    - "80:80" # Public HTTP Port
    - "443:443" # Public HTTPS Port
    - "81:81" # Admin Web Port
  volumes:
    - ./data:/data
    - ./letsencrypt:/etc/letsencrypt
mario:
  image: pengbai/docker-supermario
  container_name: mario
  ports:
    - "8600:8080"
```

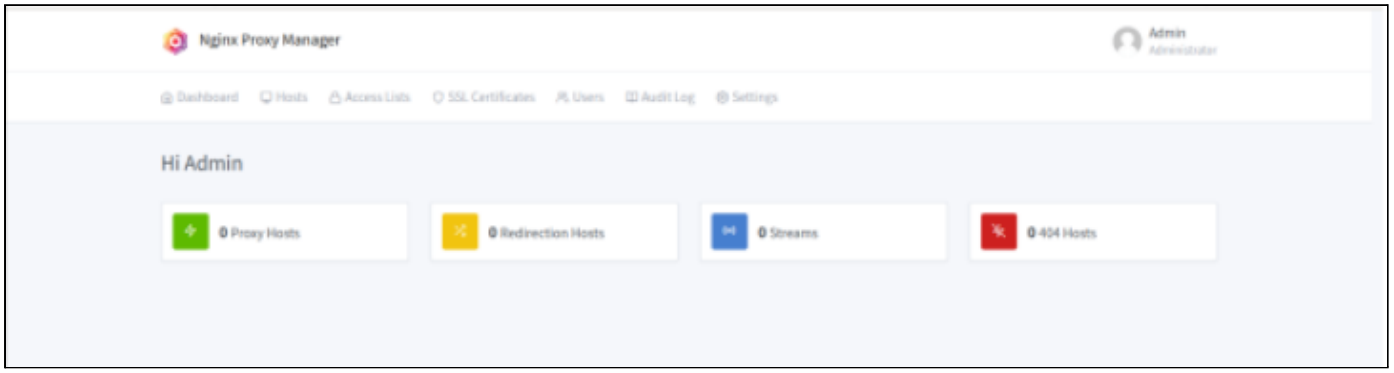
4. To deploy the stack use the following command: ***docker-compose up -d***

nginx:

This container contains NGINX Proxy Manager (N.P.M).

NPM will allow us to easily expose our container to the outer network using SSL certificate from Letsencrypt.

The container is available at port number 81.



Default username and password are:

Email: admin@example.com

Password: changeme

Mario:

This is our main container which runs the game

The game is available at port number 8600.

Shubham Saxena
(191510028)

B. Tech 3rd Year CS (CCV)

Above port can be change by changing the ports in docker-compose.yml

Thank You :)

Shubham Saxena
(191510028)

B. Tech 3rd Year CS (CCV)

Experiment #7

Flask Application

Task: Develop a web application using Docker-Compose, Flask and python that displays the table of numbers from 2 to 5, one line at a time, on each refresh. After displaying the table up to 5, it shall restart on refresh.

Tools Used: AWS EC2, Docker, Flask, VS Code

Prerequisite: Make sure you have already installed both Docker Engine and Docker Compose. You don't need to install Python or Redis, as both are provided by Docker images.

Procedure:

Section 1- Setup *(Define the application dependencies)*

1. Create a directory for the project.
2. Create a file called app.py in your project directory and paste the following code:

```
import time
from flask import Flask
app = Flask(__name__)
x = 2
y = 1
str = ""


def get_result():
   return x*y


@app.route('/')
def hello():
   global str
   global x
   global y
   if x == 2 and y == 1:
       str = ""
   if y > 10:
       x = x+1
       y = 1
       str += "<br>"
   if x > 5:
```

```
        str += "<br>Hope you have enjoyed learning the tables! Refresh to restart."
        x = 2
        y = 1
        return str
    count = get_result()
    str += "<br>{} * {} = {}".format(x, y, count)
    y = y+1
    return str
```

3. Create another file called requirements.txt in your project directory and paste this:

```
flask
```

Section 2- Create a dockerfile *(In this step, you write a Dockerfile that builds a Docker image. The image contains all the dependencies the Python application requires, including Python itself.)*

1. In your project directory, create a file named Dockerfile and paste the following:

```
# syntax=docker/dockerfile:1
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
EXPOSE 5000
COPY . .
CMD ["flask", "run"]
```

This tells Docker to:

- Build an image starting with the Python 3.7 image.
- Set the working directory to /code.
- Set environment variables used by the flask command.
- Install gcc and other dependencies
- Copy requirements.txt and install the Python dependencies.
- Add metadata to the image to describe that the container is listening on port 5000
- Copy the current directory in the project to the workdir in the image.
- Set the default command for the container to flask run.

Section 3- Define services in a compose file

1. Create a file called docker-compose.yml in your project directory and paste the following. This Compose file defines one service: web

   The web service uses an image that's built from the Dockerfile in the current directory. It then binds the container and the host machine to the exposed port, 8000.

   This example service uses the default port for the Flask web server, 5000.

```
version: "3"
services:
 web:
   build: .
   ports:
     - "8000:5000"
   volumes:
     - .:/code
   environment:
     FLASK_ENV: development
```

Section 4- Build and run your app with Compose

1. From your project directory, start up your application by running ***docker-compose up -d***

```
D:\composetest>docker-compose up
Creating network "composetest_default" with the default driver
Creating composetest_web_1 ... done
Attaching to composetest_web_1
web_1  |  * Serving Flask app 'app.py' (lazy loading)
web_1  |  * Environment: development
web_1  |  * Debug mode: on
web_1  |  * Running on all addresses (0.0.0.0)
web_1  |    WARNING: This is a development server. Do not use it in a production deployment.
web_1  |  * Running on http://127.0.0.1:5000
web_1  |  * Running on http://172.21.0.2:5000 (Press CTRL+C to quit)
web_1  |  * Restarting with stat
web_1  |  * Debugger is active!
web_1  |  * Debugger PIN: 138-065-954
```

2. Enter http://localhost:8000/ in a browser to see the application running.

   If you're using Docker natively on Linux, Docker Desktop for Mac, or Docker Desktop for Windows, then the web app should now be listening on port 8000 on your Docker daemon host.

3. Refresh the page.
4. Stop the application,by running docker-compose down from within your project directory.

Shubham Saxena
(191510028)

B. Tech 3rd Year CS (CCV)

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30

4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
```

```
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30

4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40

5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

Hope you have enjoyed learning the tables! Refresh to restart.
```

Thank You :)

Dr. UJJWAL PATNI

businessJeeto.com

FREE FORMATS
FOR ALL VIEWERS OF

Mr SANDEEP MAHESHWARI &
Dr UJJWAL PATNI's YouTube Channel

LED 3 GUINNESS WORLD RECORDS

For any support : 88787-59999

UJJWAL PATNI
INTERNATIONAL
MENTORING SUCCESS

ONE GOAL ONE DAY

NAME _____ mm / yyyy

GOAL OF THE DAY

WEEKLY REWARD / PENALTY

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____
10. _____
11. _____
12. _____
13. _____
14. _____
15. _____
16. _____
17. _____
18. _____
19. _____
20. _____
21. _____
22. _____
23. _____
24. _____
25. _____
26. _____
27. _____
28. _____
29. _____
30. _____
31. _____

UJJWAL PATNI
INTERNATIONAL
CREATING *SUCCESS* STORIES

DAILY SELF AUDIT

Start Date – ..... /..... / .....

What did I do today that I should not have done? आज मैंने ऐसा क्या किया जो मुझे नहीं करना चाहिए था ?		What did I do today of which I am proud? आज मैंने ऐसा क्या किया जिस पर मुझे गर्व है ?
	----- Day - 1 -----	
	----- Day - 2 -----	
	----- Day - 3 -----	
	----- Day - 4 -----	
	----- Day - 5 -----	
	----- Day - 6 -----	

BCSC1002: Object-Oriented Programming (New)

OBJECTIVE

This course introduces the Object-Oriented programming paradigm to students. It also teaches a student how to think objectively and model a Java program for solving real-world problems.

CREDITS: 3 **L-T-P:3-0-0**

Module No.	Content	Teaching Hours
I	**Object-Oriented Programming:** Features of Object-Oriented Programming, Introduction to Object-Oriented Java Programming. **Understanding Java Technology & Environment:** Understanding the compilation process of the JVM, JVM vs JDK vs JRE, Key Features of Java, Structure of a simple Java program. **Working with Java Primitive Data Types:** Strongly Typed nature of Java, Primitive Data Types in Java, The new 'var' keyword, Scope of a variable. **Accepting User Input in Java Programs:** using the Scanner class, using command line arguments. **Programming Constructs:** Sequence, Selection, Iteration & Transfer Statements, For-Each Loop. **Working with Java Arrays**: Declaring and Initializing One-Dimensional and Two-Dimensional Arrays in Java, Introduction to java.util.Arrays class. **The String API:** String Data Type, commonly used methods from the String API, StringTokenizer, StringBuilder & StringBuffer. **Creating and Using Methods:** Signature of a method, Types of Methods, Overloading methods in a class, Static and Non-Static Methods.	14 hours
II	**Describing and Using Objects & Classes:** Declare the structure of a Java class, declaring members of a class (fields and methods), declaring and using Java Objects, lifecycle of an Object (creation, assignment, dereferencing and garbage collection), Constructors of a class, Overloading Constructors, Constructor chaining using 'this' and 'super' keyword. **Using Java Packages:** create and import Java packages and static imports, abstracting program logic to packages, creating executable main class, running the executable class inside a package. **Applying Encapsulation**: Using access modifiers with/in a class, principles of encapsulation. **Programming Abstractly Through Interfaces:** create and implement Interfaces for programs, private and default methods in Interfaces, declaring Abstract Classes, Constructors in Abstract Classes. Marker Interface, Functional Interfaces, Lambda Expressions in Java. **Reusing Implementations using Inheritance:** Declaring Subclasses and Superclasses, extend Abstract Classes, implementing Interfaces, exploring polymorphic behaviour by overriding methods, Object Types vs Reference Types, differentiate overloading, overriding and hiding. **Exception Handling:** Exception Hierarchy, Need of Exception Handling, Checked Exceptions, Unchecked Exceptions and Errors, Try-Catch Blocks, Finally, Throw & Throws Keywords, creating and handling Custom Exceptions.	14 hours
III	**Threads in Java:** Life Cycle of a Thread, Creating threads using Runnable and Thread, 'sleep()', Thread Priorities. **Using Wrapper Classes:** Wrapper Classes in Java, Boxing-Unboxing-Autoboxing-AutoUnboxing. **Generics & Collections:** Creating Generic classes, Generic Methods, Diamond Notation, Wildcards, Type Erasure, Collection Hierarchy, Base Interfaces, Lists, Sets and Maps. **The Stream API:** Introduction to the Stream API, using lambda expressions in Streams. **Regular Expressions:** Pattern and Matcher Class. **JDBC:** JDBC Drivers, Connecting to a MySQL Database, DriverManager, Connection Interface, Statement Interface, ResultSet Interface, PreparedStatements.	14 hours

Text Books:

- Herbert Schildt (2019), "The Complete Reference, Java Eleventh Edition", Oracle Press.

Reference Books:

- Cay S Hosrtmann (2018), "Core Java Volume I—Fundamentals, Eleventh Edition", Pearson
- Rogers Cadenhead (2020), "Sams Teach Yourself Java in 21 Days (Covers Java 11/12), 8th Edition", Pearson

2IS55 Software Evolution

Software metrics (2)

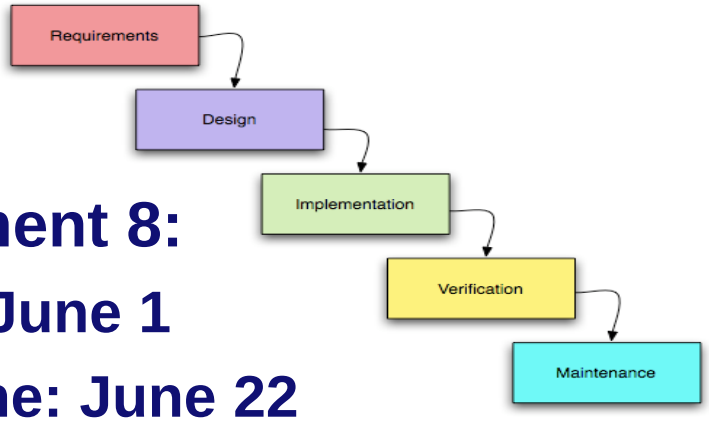
Alexander Serebrenik

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

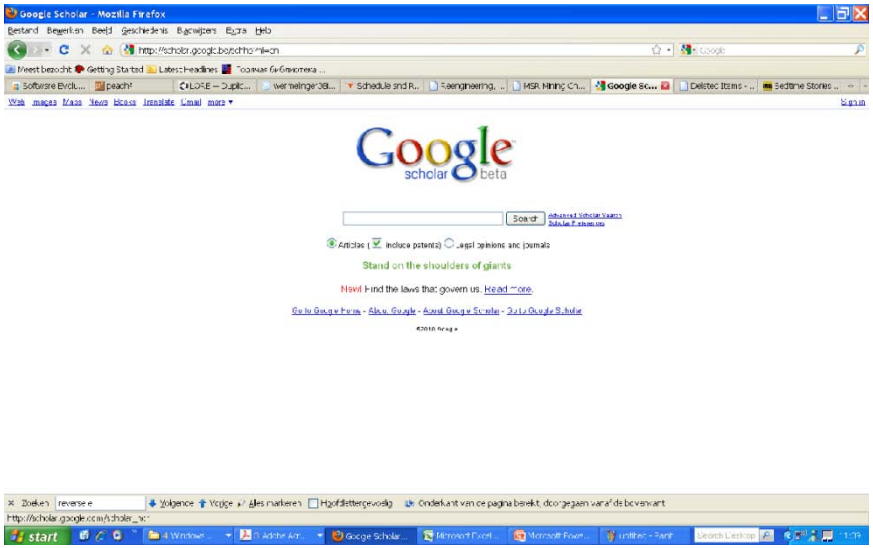
Assignment 6

- **Assignment 6:**
  - **Deadline: May 11**
  - **1-2 students**

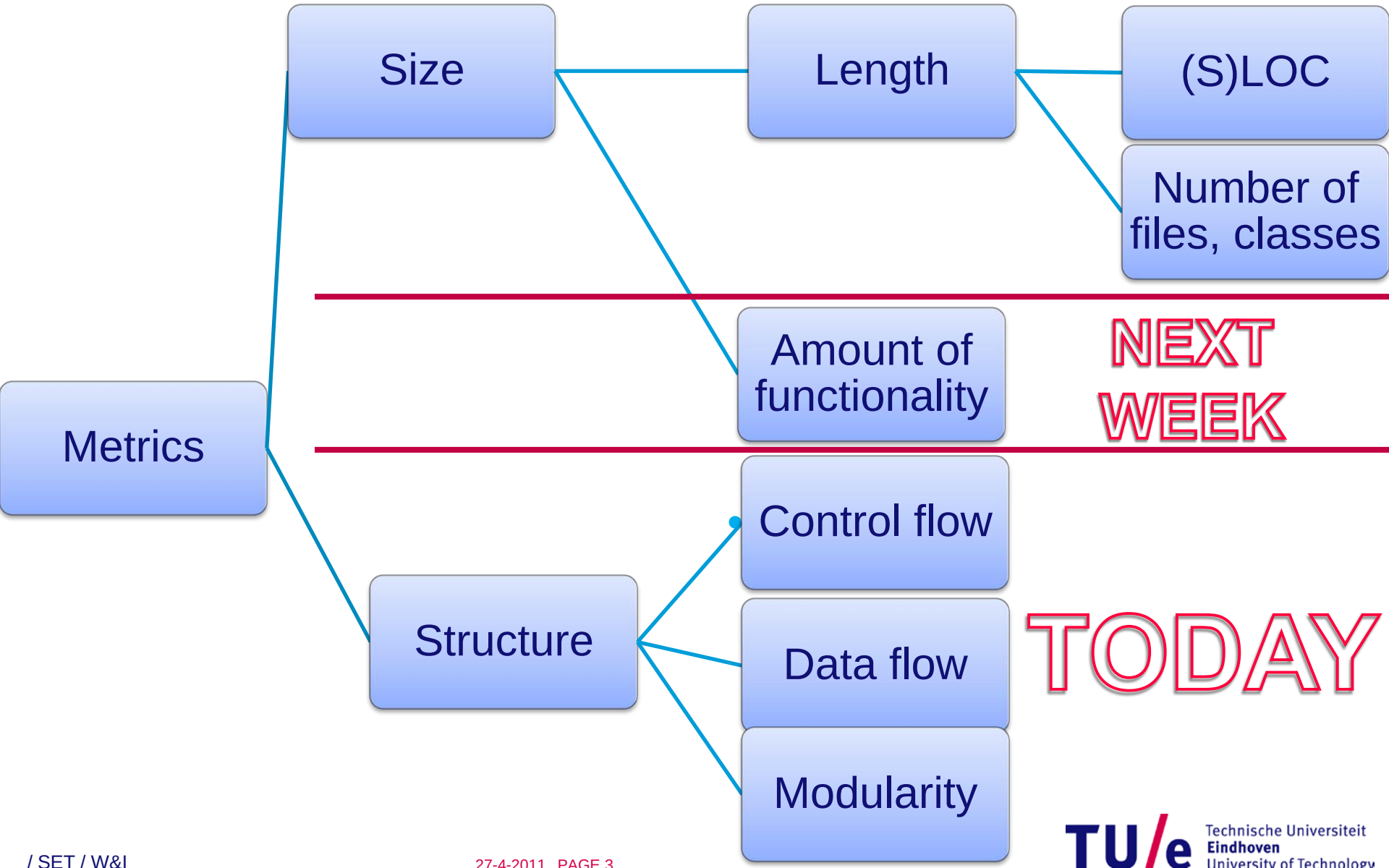


- **Assignment 8:**
  - **Open: June 1**
  - **Deadline: June 22**
  - **1-2 students**
  - **ReqVis** http://www.student.tue.nl/Q/w.j.p.v.ravensteijn/index.html
    - **Try it!**
    - **Give us feedback before June 1!**
    - **Mac-fans: Talk to Wiljan!**

Sources


Google Scholar - Mozilla Firefox
Google
scholar beta
Search
Stand on the shoulders of giants
Read more.

So far


Size
Length
(S)LOC
Number of files, classes
Amount of functionality
NEXT WEEK
Metrics
Control flow
Structure
Data flow
TODAY
Modularity

Complexity metrics: Halstead (1977)

- Sometimes is classified as size rather than complexity
- Unit of measurement

Parts of a statement: operators and operands

Line: LOC, SLOC, LLOC

Units, files, classes

Packages, directories

- Operators:
  - traditional (+,++, >), keywords (return, if, continue)
- Operands
  - identifiers, constants

Halstead metrics

- Four basic metrics of Halstead

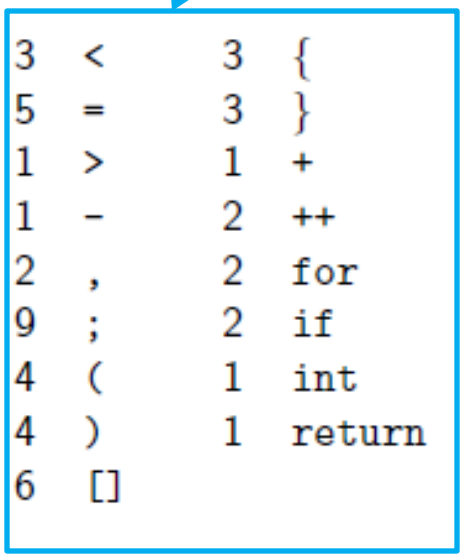
	Total	Unique
Operators	N1	n1
Operands	N2	n2

- Length: N = N1 + N2
- Vocabulary: n = n1 + n2
- Volume: $V = N \log_2 n$
  - Insensitive to lay-out
  - VerifySoft:
    - $20 \leq$ Volume(function) ≤ 1000
    - $100 \leq$ Volume(file) ≤ 8000

Halstead metrics: Example

```
void sort ( int *a, int n ) {
int i, j, t;

   if ( n < 2 ) return;
   for ( i=0 ; i < n-1; i++ )  {
         for ( j=i+1 ; j < n ; j++ ) {
               if ( a[i] > a[j] ) {
                     t = a[i];
                     a[i] = a[j];
                     a[j] = t;
               }
         }
   }
}
```

- Ignore the function definition
- Count operators and operands



3	<	3	{	1	0
5	=	3	}	2	1
1	>	1	+	1	2
1	-	2	++	6	a
2	,	2	for	8	i
9	;	2	if	7	j
4	(	1	int	3	n
4	)	1	return	3	t
6	[]				

	Total	Unique
Operators	N1 = 50	n1 = 17
Operands	N2 = 30	n2 = 7

$V = 80 \log_2(24) \approx 392$

Inside the boundaries [20;1000]

Further Halstead metrics

	Total	Unique
Operators	N1	n1
Operands	N2	n2

- Volume: $V = N \log_2 n$
- Difficulty: D = (n1 / 2) * (N2 / n2)
  - Sources of difficulty: new operators and repeated operands
  - Example: 17/2 * 30/7 ≈ 36
- Effort: E = V * D
- Time to understand/implement (sec): T = E/18
  - Running example: 793 sec ≈ 13 min
  - Does this correspond to your experience?
- Bugs delivered: $E^{2/3}/3000$
  - For C/C++: known to underapproximate
  - Running example: 0.19

Halstead metrics are sensitive to…

- What would be your answer?

- Syntactic sugar:

i = i+1	Total	Unique
Operators	N1 = 2	n1 = 2
Operands	N2 = 3	n2 = 2

i++	Total	Unique
Operators	N1 = 1	n1 = 1
Operands	N2 = 1	n2 = 1

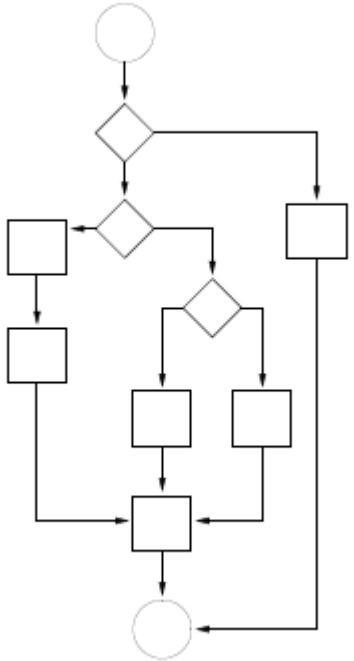
- Solution: normalization (see the code duplication slides)

Structural complexity

- **Structural complexity:**
  - **Control flow**
  - **Data flow**

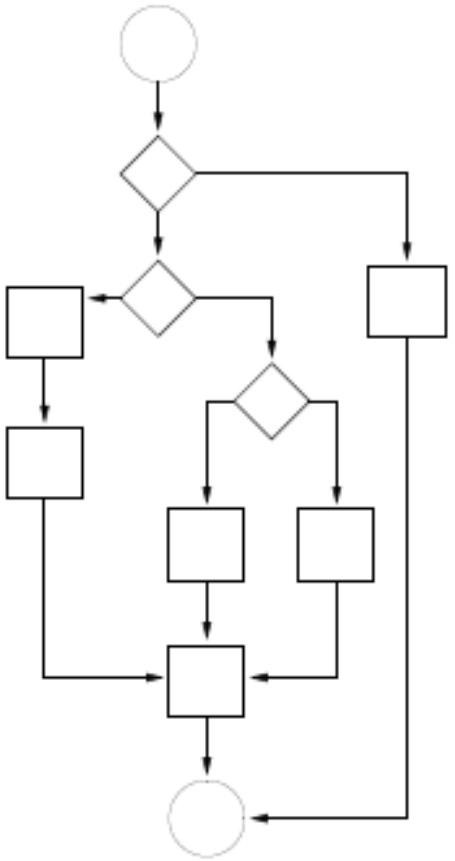
  } **Commonly represented as graphs** → **Graph-based metrics**

  - **Modularity**



- **Number of vertices**
- **Number of edges**
- **Maximal length (depth)**

McCabe complexity (1976)



In general

- **v(G) = #edges - #vertices + 2**

For control flow graphs

- **v(G) = #binaryDecisions + 1, or**
- **v(G) = #IFs + #LOOPs + 1**

Number of paths in the control flow graph.

A.k.a. "cyclomatic complexity"

Each path should be tested!

v(G) – a testability metrics

Boundaries
- **v(function) ≤ 15**
- **v(file) ≤ 100**

McCabe complexity: Example

```
void sort ( int *a, int n ) {
int i, j, t;

   if ( n < 2 ) return;
   for ( i=0 ; i < n-1; i++ )  {
         for ( j=i+1 ; j < n ; j++ ) {
               if ( a[i] > a[j] ) {
                     t = a[i];
                     a[i] = a[j];
                     a[j] = t;
               }
         }
   }
}
```

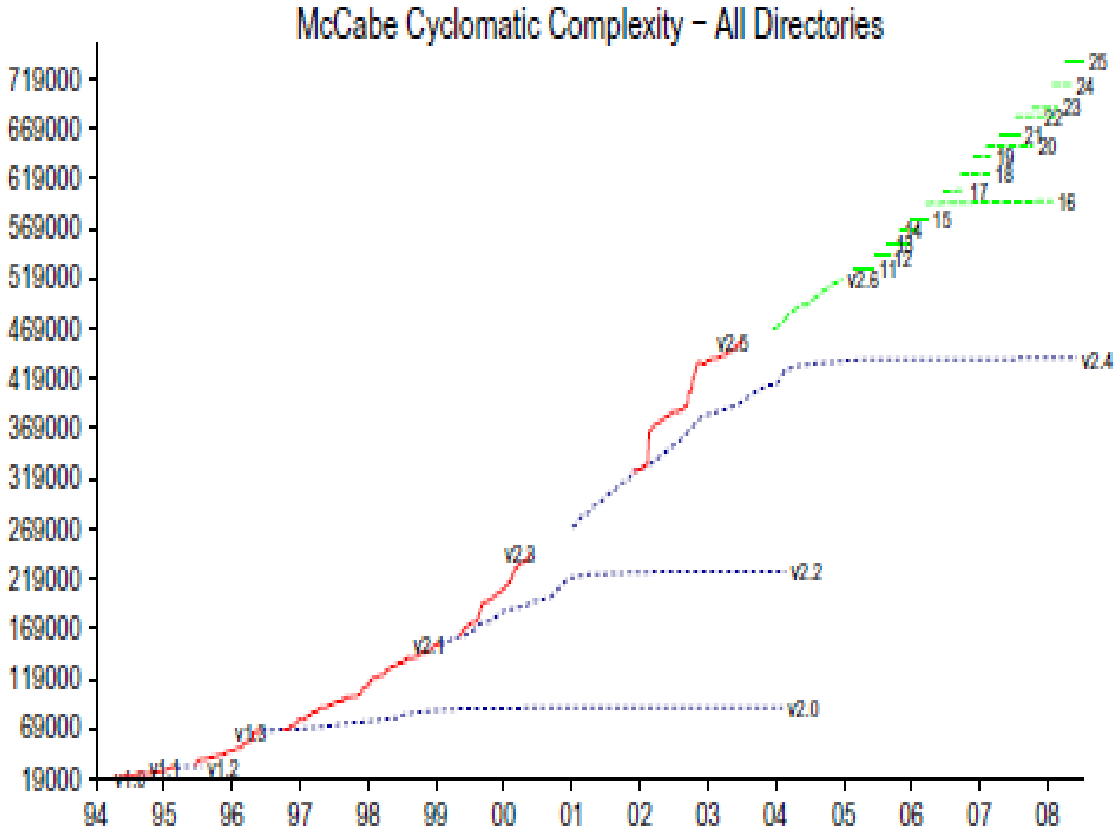
- Count IFs and LOOPs
- IF: 2, LOOP: 2
- v(G) = 5
- Structural complexity

Question to you

- **Is it possible that the McCabe's complexity is higher than the number of possible execution paths in the program?**

- **Lower than this number?**

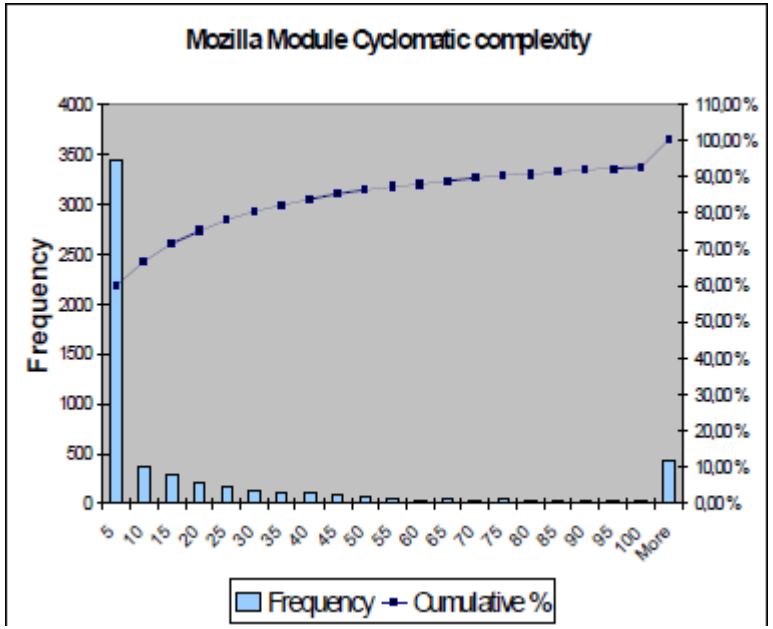
McCabe's complexity in Linux kernel

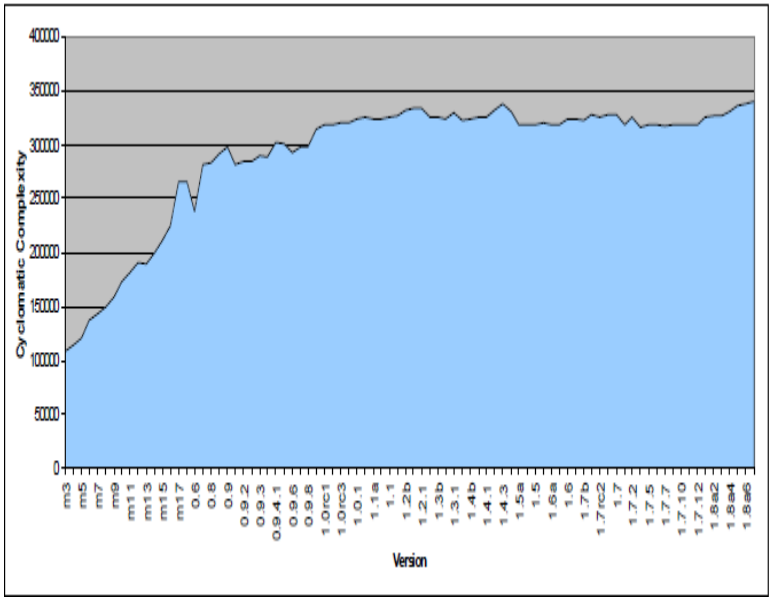


A. Israeli, D.G. Feitelson 2010

- Linux kernel
- Multiple versions and variants
  - Production (blue dashed)
  - Development (red)
  - Current 2.6 (green)

McCabe's complexity in Mozilla [Røsdal 2005]





- **Most of the modules have low cyclomatic complexity**
- **Complexity of the system seems to stabilize**

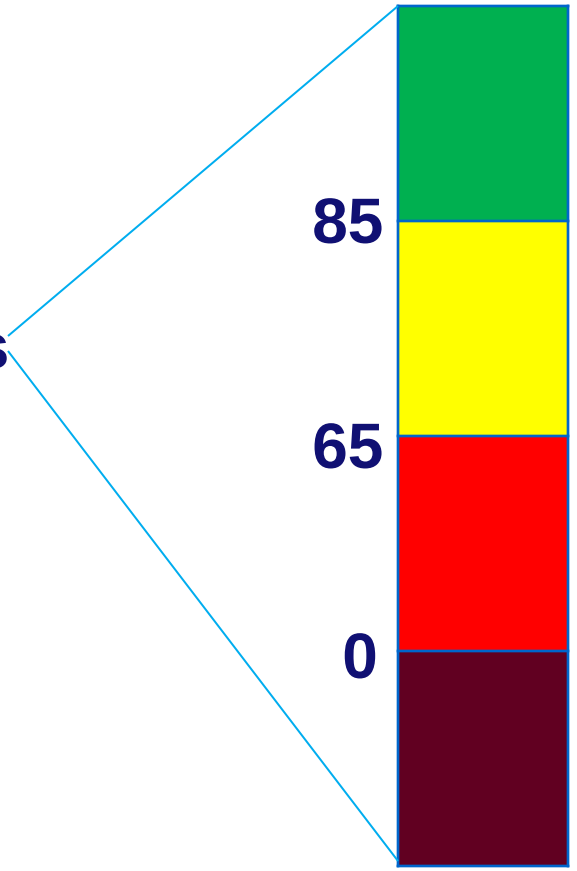
Summarizing: Maintainability index (MI) [Coleman, Oman 1994]

$$MI_1 = 171 - 5.2\ln(V) - 0.23V(g) - 16.2\ln(LOC)$$

Halstead McCabe LOC

$$MI_2 = MI_1 + 50\sin\sqrt{2.46\,perCM}$$

% comments



- MI_2 can be used only if comments are meaningful
- If more than one module is considered – use average values for each one of the parameters
- Parameters were estimated by fitting to expert evaluation
  - BUT: few not big systems!

McCabe complexity: Example

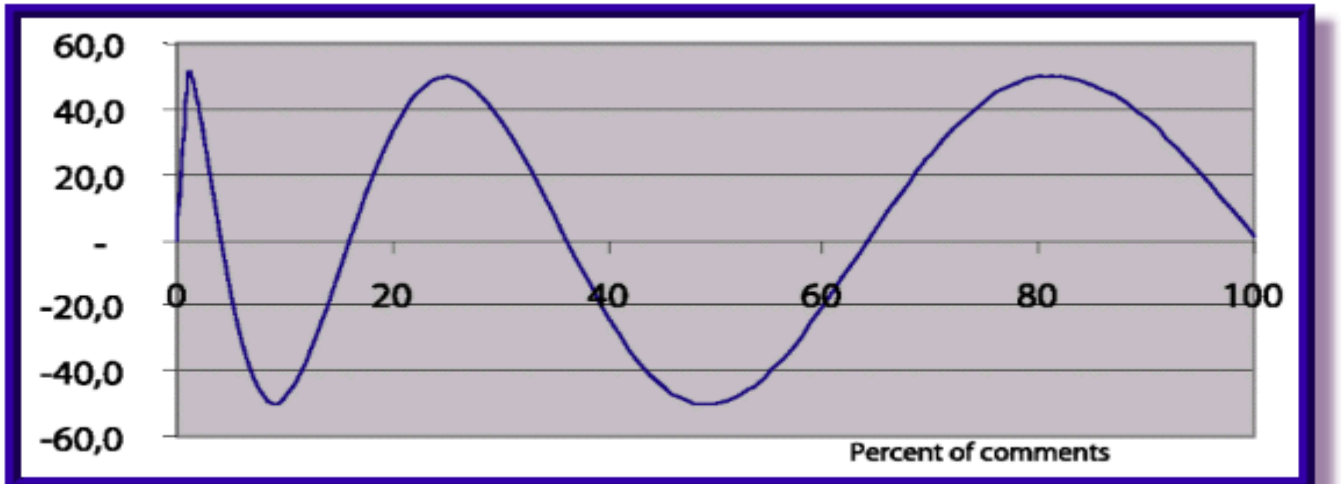
```
void sort ( int *a, int n ) {
int i, j, t;

   if ( n < 2 ) return;
   for ( i=0 ; i < n-1; i++ )  {
         for ( j=i+1 ; j < n ; j++ ) {
               if ( a[i] > a[j] ) {
                     t = a[i];
                     a[i] = a[j];
                     a[j] = t;
               }
         }
   }
}
```

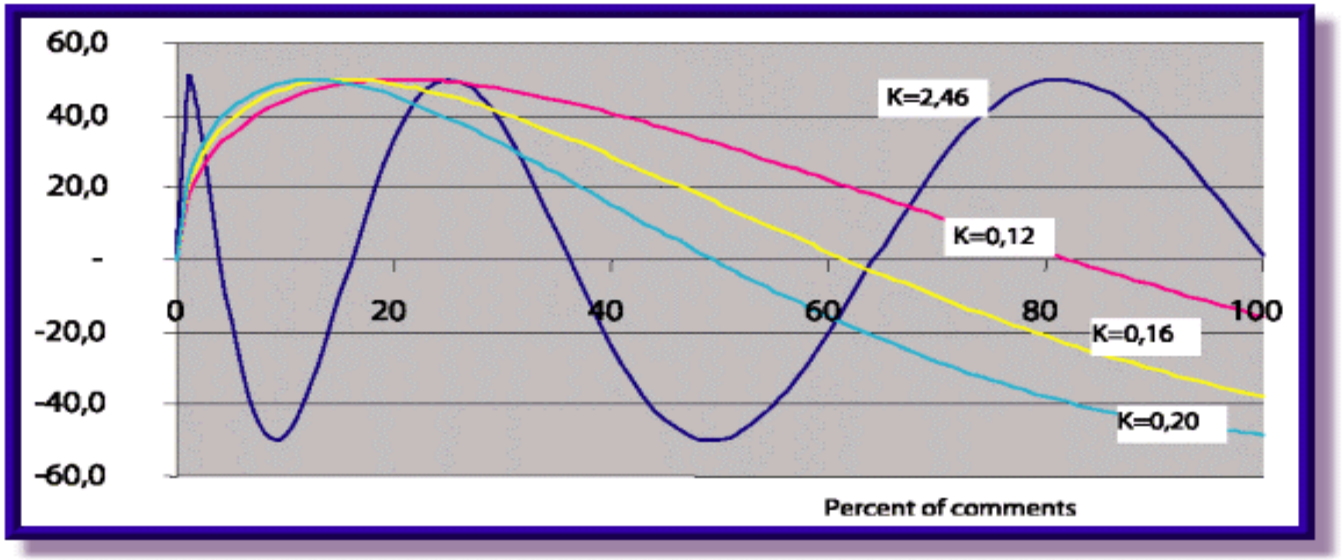
- Halstead's V ≈ 392
- McCabe's v(G) = 5
- LOC = 14
- MI_1 ≈ 96
- Easy to maintain!

Comments?

$$50 \sin \sqrt{2.46\, perCM}$$

[Liso 2001]





Peaks:

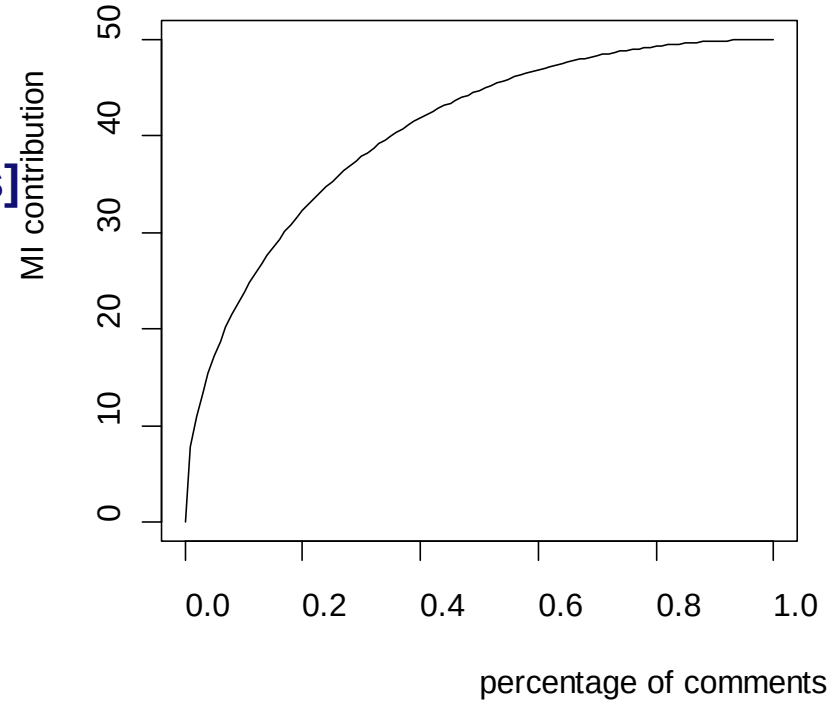
- 25% (OK),
- 1% and 81% - ???

Better:

- $0.12 \leq K \leq 0.2$

Another alternative:

- **Percentage as a fraction [0;1] – [Thomas 2008, Ph.D. thesis]**
- **The more comments – the better?**



Evolution of the maintainability index in Linux

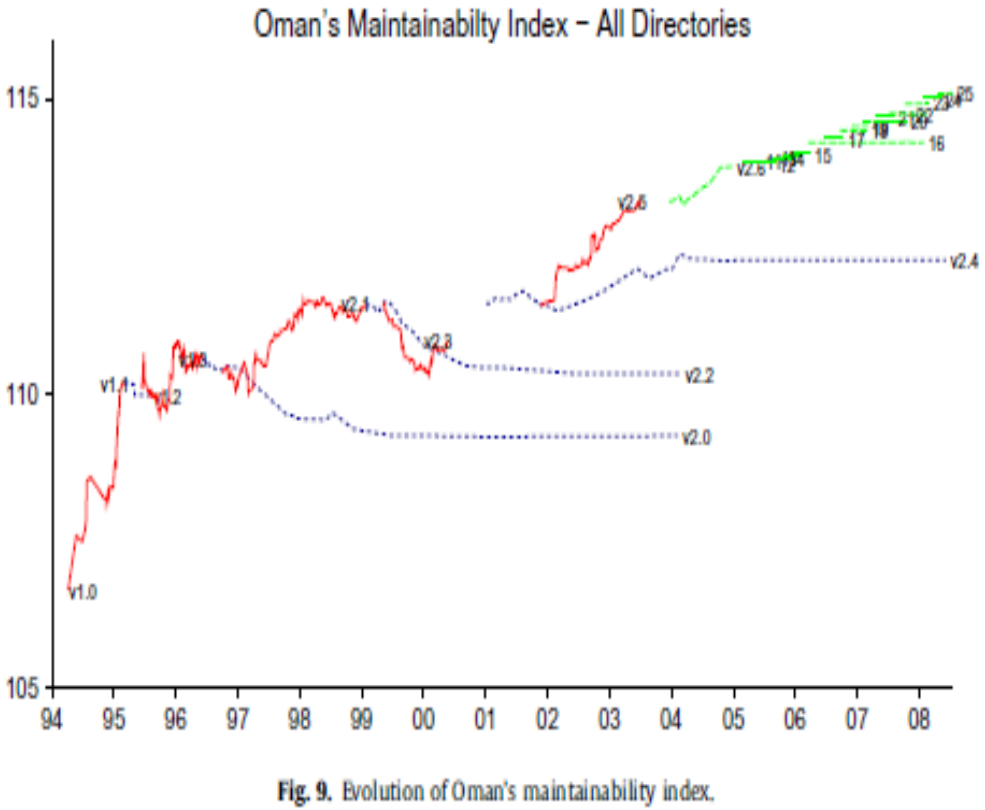


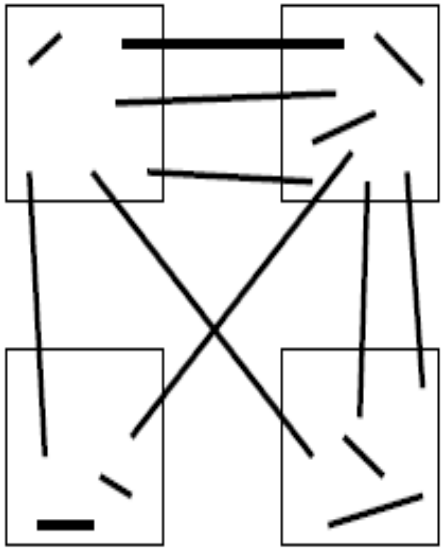
Fig. 9. Evolution of Oman's maintainability index.

- **Size, Halstead volume and McCabe complexity decrease**
- **% comments decreases as well**
  - **BUT they use the [0;1] definition, so the impact is limited**

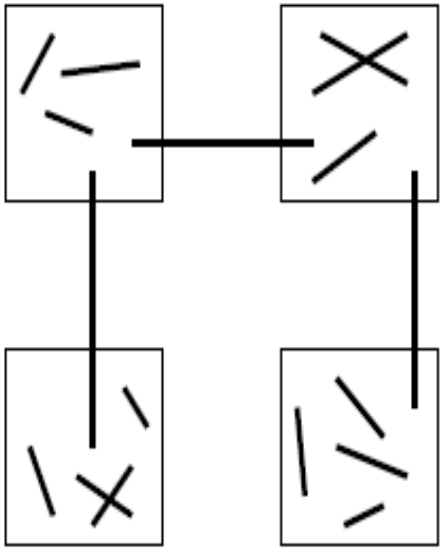
A. Israeli, D.G. Feitelson 2010

What about modularity?

Design A



Design B

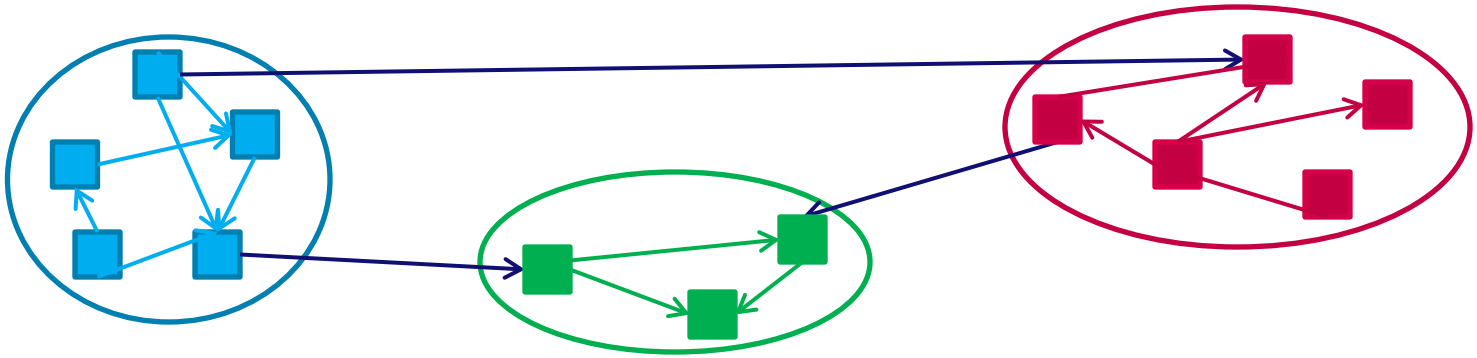


- **Cohesion: calls inside the module**
- **Coupling: calls between the modules**

	A	B
Cohesion	Lo	Hi
Coupling	Hi	Lo

- **Squares are modules, lines are calls, ends of the lines are functions.**
- **Which design is better?**

Do you still remember?



- **Many intra-package dependencies: high cohesion**

$$A_i = \frac{\mu_i}{N_i^2} \quad \textbf{or} \quad A_i = \frac{\mu_i}{N_i(N_i - 1)}$$

- **Few inter-package dependencies: low coupling**

$$E_{i,j} = \frac{\varepsilon_{i,j}}{2N_iN_j}$$

- **Joint measure**

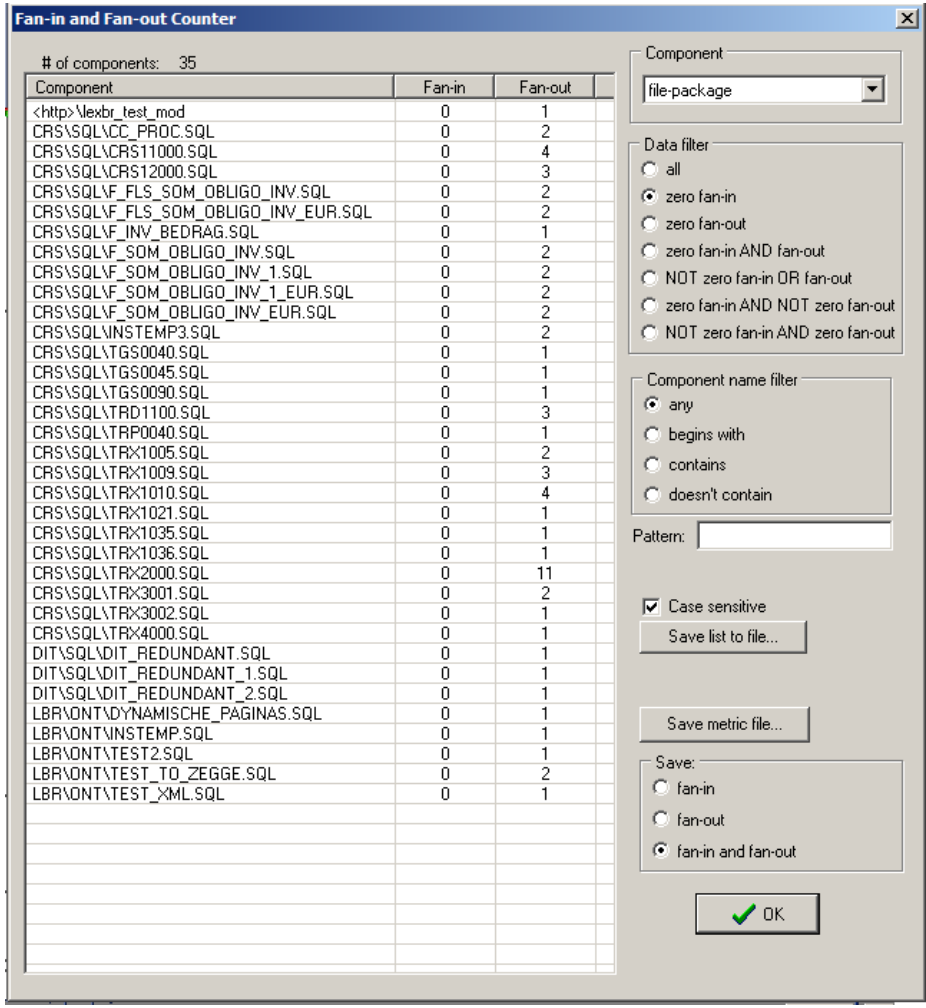
$$MQ = \frac{1}{k}\sum_{i=1}^{k} A_i - \frac{2}{k(k-1)}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} E_{i,j}$$

k **- Number of packages**

Modularity metrics: Fan-in and Fan-out

- **Fan-in of M: number of modules calling functions in M**
- **Fan-out of M: number of modules called by M**

- **Modules with fan-in = 0**
- **What are these modules?**
  - **Dead-code**
  - **Outside of the system boundaries**
  - **Approximation of the “call” relation is imprecise**



Henry and Kafura's information flow complexity [HK 1981]

- **Fan-in and fan-out can be defined for procedures**
  - **HK: take global data structures into account:**
    - **read for fan-in,**
    - **write for fan-out**

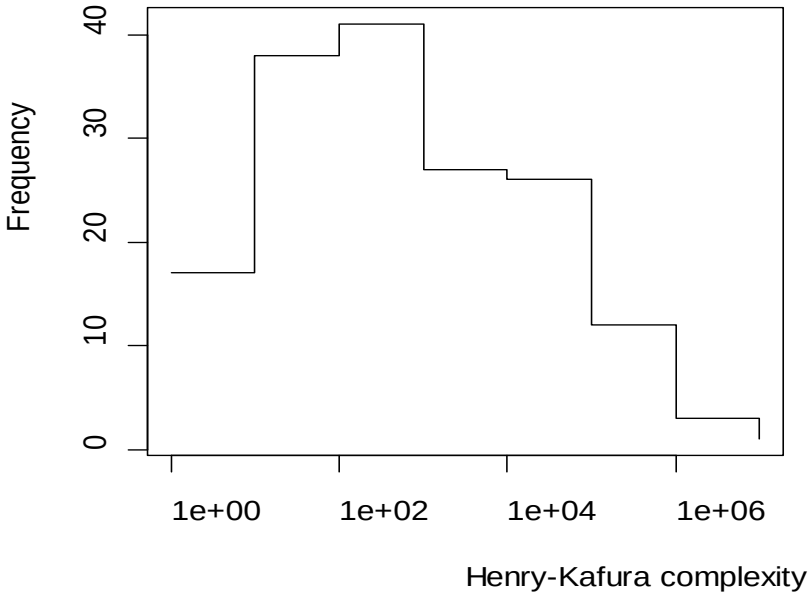
- **Henry and Kafura: procedure as HW component connecting inputs to outputs**

$$hk = sloc * (\ fanin * \ fanout)^2$$

- **Shepperd**

$$s = (\ fanin * \ fanout)^2$$

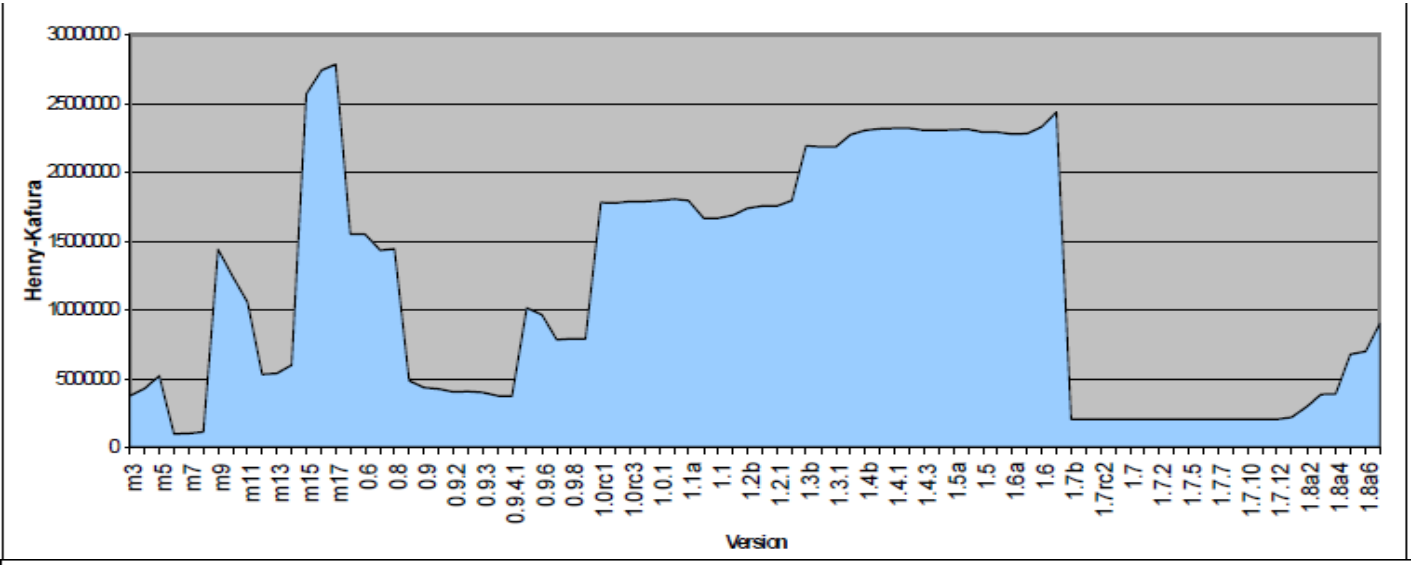
Information flow complexity of Unix procedures

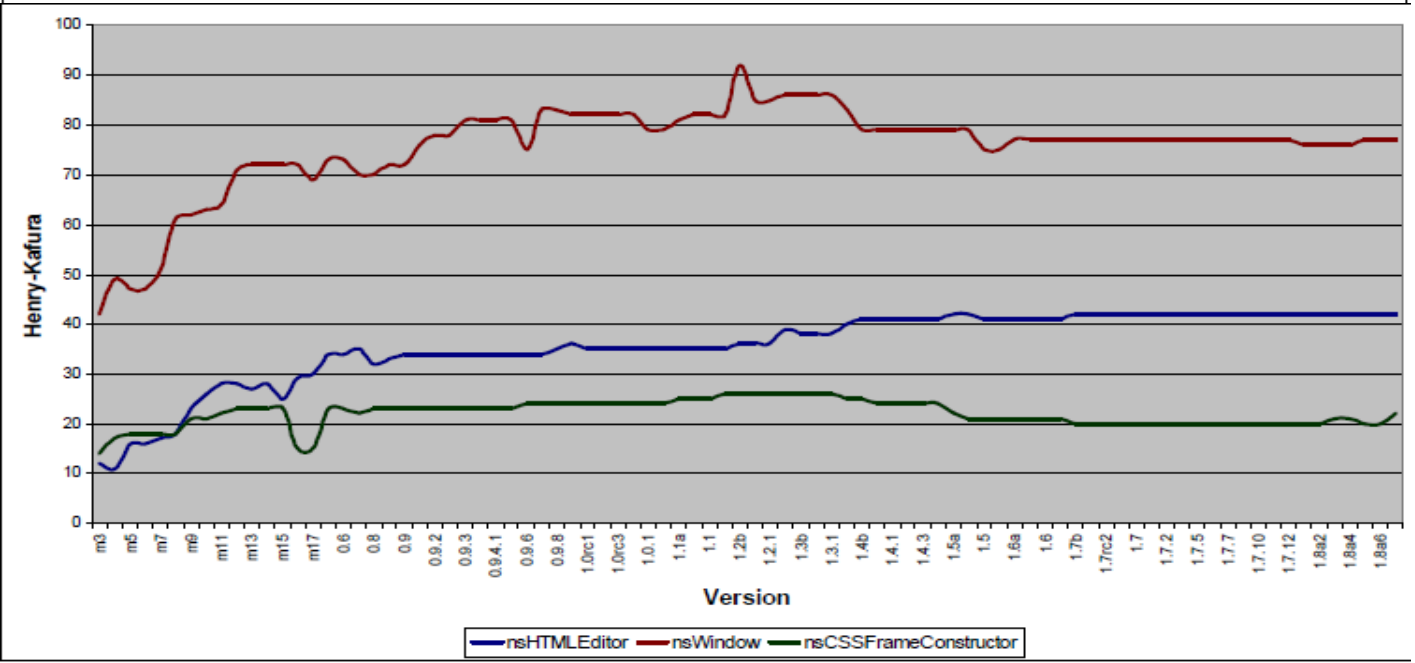


- Solid – #procedures within the complexity range
- Dashed - #changed procedures within the complexity range

- Highly complex procedures are difficult to change but they are changed often!
- Complexity comes the “most complex” procedures

Evolution of the information flow complexity



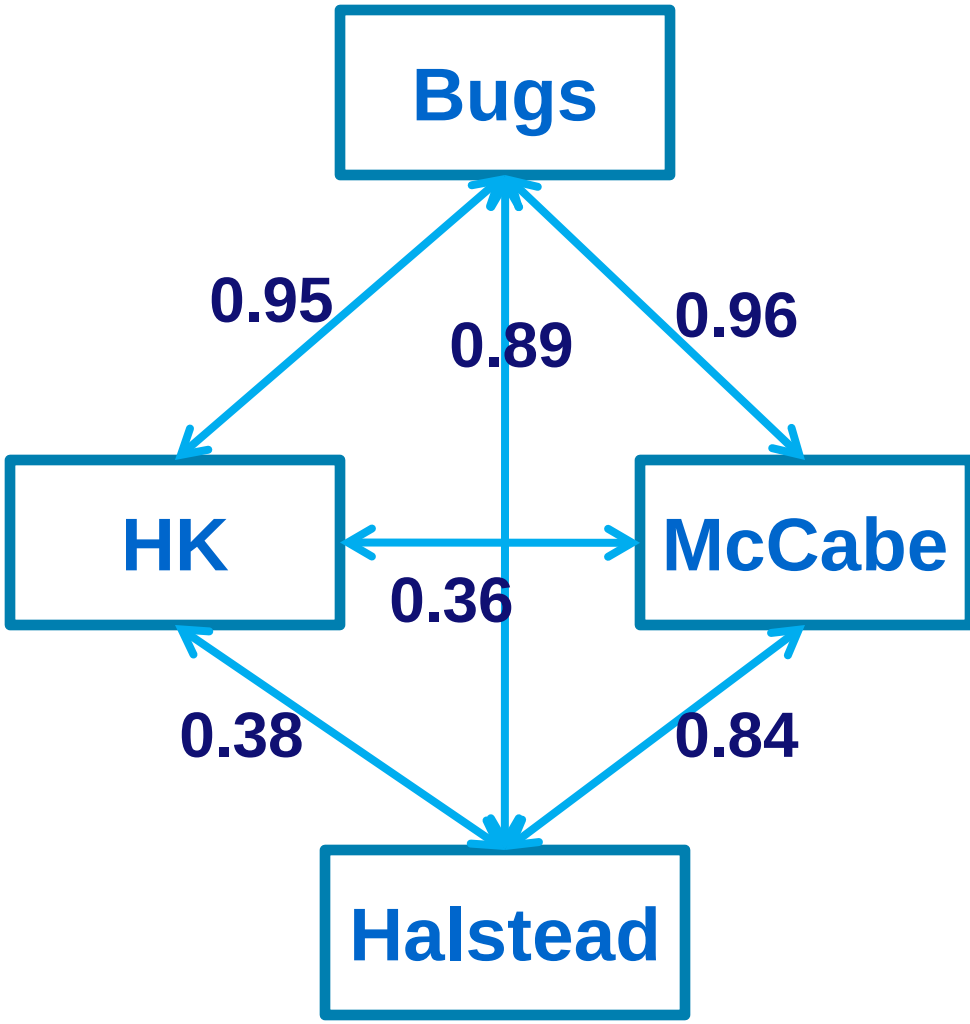


- Mozilla
- Shepperd version
- Above: Σ the metrics over all modules
- Below: 3 largest modules
- What does this tell?

TU/e Technische Universiteit Eindhoven University of Technology

Summary so far…

- Complexity metrics
  - Halstead's effort
  - McCabe (cyclomatic)
  - Henry Kafura/Shepperd (information flow)
- Are these related?
- And what about bugs?
- Harry,Kafura,Harris 1981
  - 165 Unix procedures
- What does this tell us?



From imperative to OO

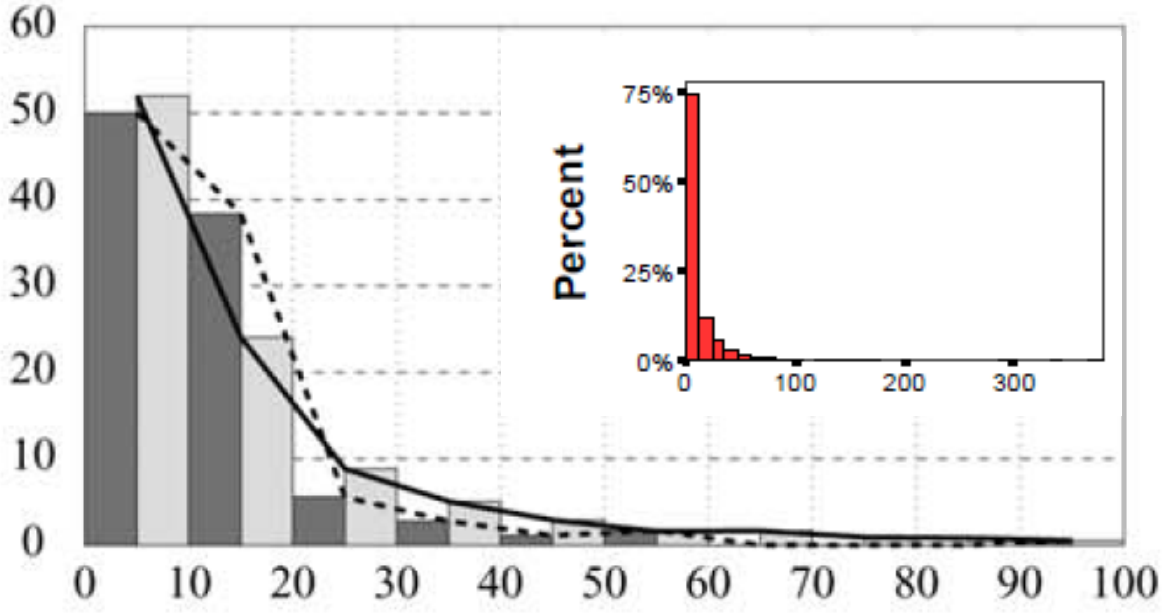
- All metrics so far were designed for imperative languages
  - Applicable for OO
    - On the method level
    - Also
      - Number of files → number of classes/packages
      - Fan-in → afferent coupling (C_a)
      - Fan-out → efferent coupling (C_e)
  - But do not reflect OO-specific complexity
    - Inheritance, class fields, abstractness, …
- Popular metric sets
  - Chidamber and Kemerer, Li and Henry, Lorenz and Kidd, Abreu, Martin

Chidamber and Kemerer

- WMC – weighted methods per class
  - Sum of metrics(m) for all methods m in class C
- DIT – depth of inheritance tree
  - java.lang.Object? Libraries?
- NOC – number of children
  - Direct descendents
- CBO – coupling between object classes
  - A is coupled to B if A uses methods/fields of B
  - CBO(A) = | {B|A is coupled to B} |
- RFC - #methods that can be executed in response to a message being received by an object of that class.

Chidamber and Kemerer

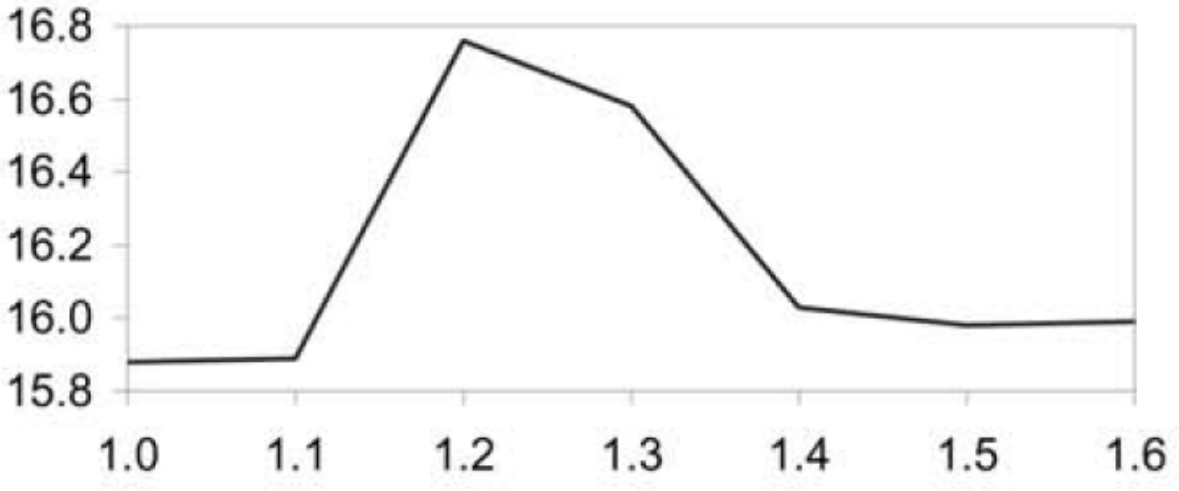
- **WMC – weighted methods per class**
  - **Sum of metrics(m) for all methods m in class C**
  - **Popular metrics: McCabe's complexity and unity**
  - **WMC/unity = number of methods**
  - **Statistically significant correlation with the number of defects**



- **WMC/unity**
- **Dark: Basili et al.**
- **Light: Gyimothy et al. [Mozilla 1.6]**
- **Red: High-quality NASA system**

Chidamber and Kemerer

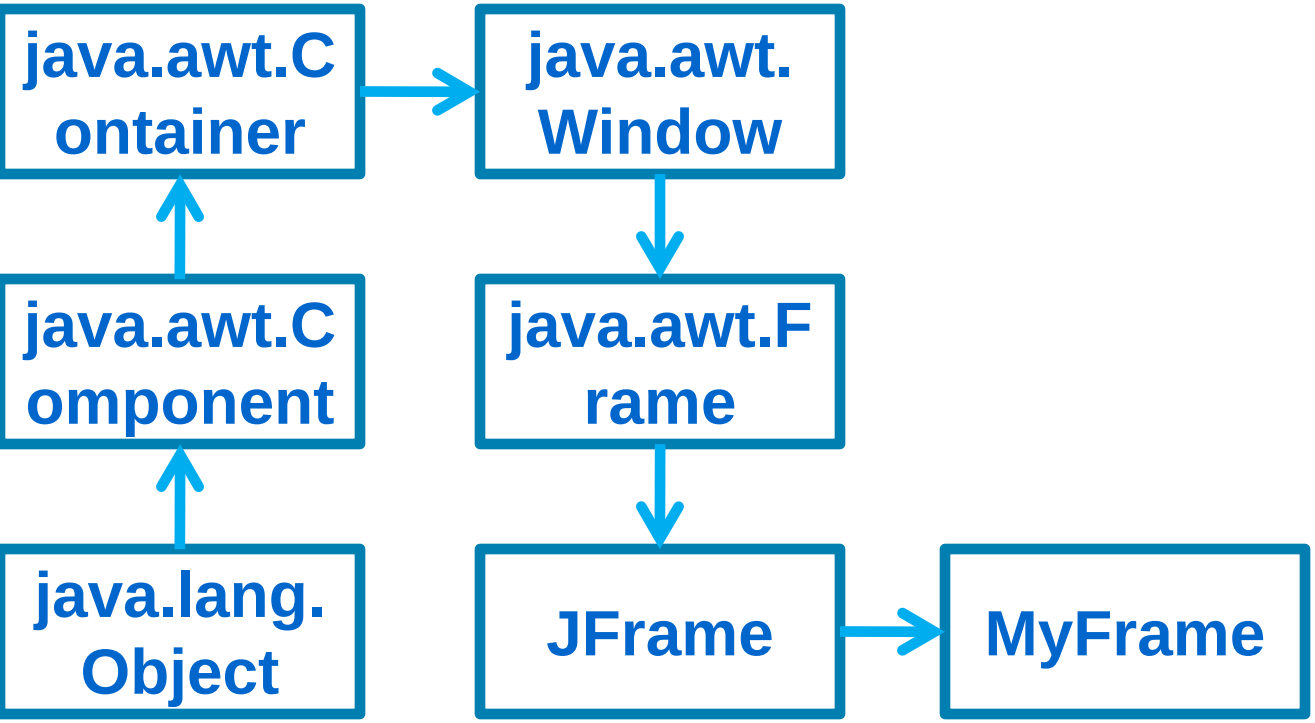
- **WMC – weighted methods per class**
  - **Sum of metrics(m) for all methods m in class C**
  - **Popular metrics: McCabe's complexity and unity**
  - **WMC/unity = number of methods**
  - **Statistically significant correlation with the number of defects**



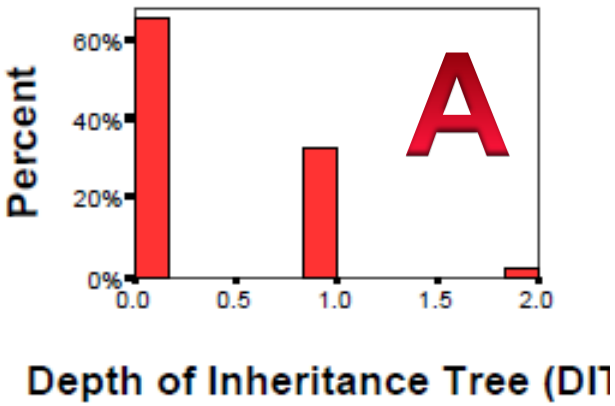
- **WMC/unity**
- **Gyimothy et al.**
- **Average**

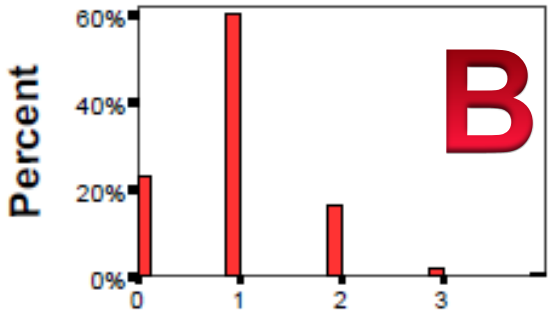
Depth of inheritance - DIT

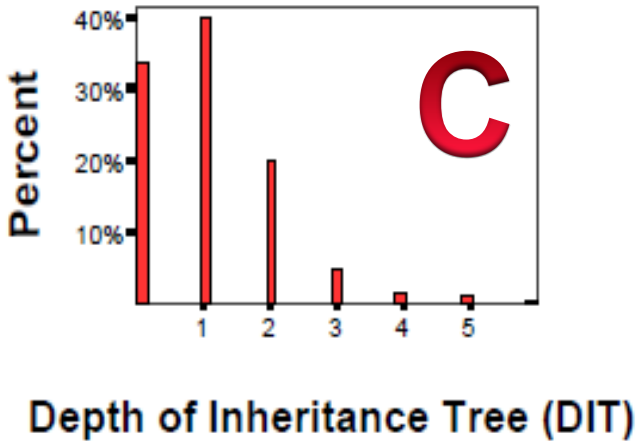
- **Variants: Were to start and what classes to include?**
  - **1, JFrame is a library class, excluded**
  - **2, JFrame is a library class, included**
  - **7**



DIT – what is good and what is bad?

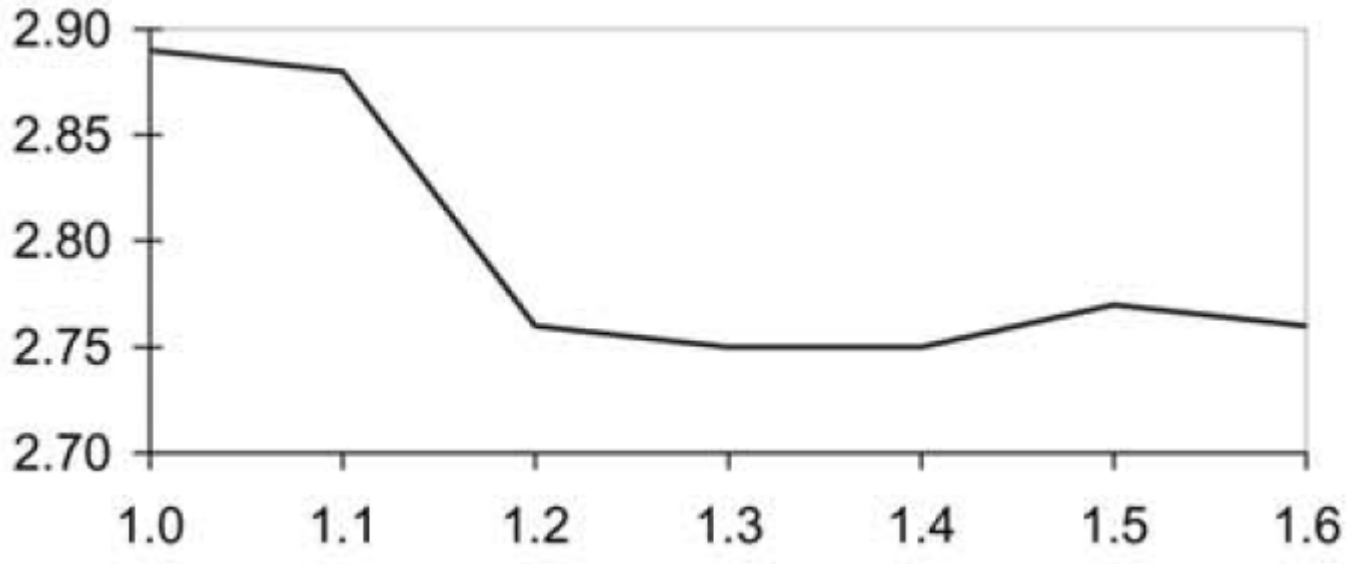






- Three NASA systems
- What can you say about the use of inheritance in systems A, B and C?
- Observation: quality assessment depends not just on one class but on the entire distribution

Average DIT in Mozilla



- **How can you explain the decreasing trend?**

Other CK metrics

- **NOC – number of children**
- **CBO – coupling between object classes**
- **RFC - #methods that can be executed in response to a message being received by an object of that class.**
- **More or less "exponentially" distributed**

Metric	Our results	[1]	[22]	[21]
WMC	++	+	++	++
DIT	+	++	0	−
RFC	++	++	+	
NOC	0	++	−−	
CBO	++	+	+	+

Significance of CK metrics to predict the number of faults

Modularity metrics: LCOM

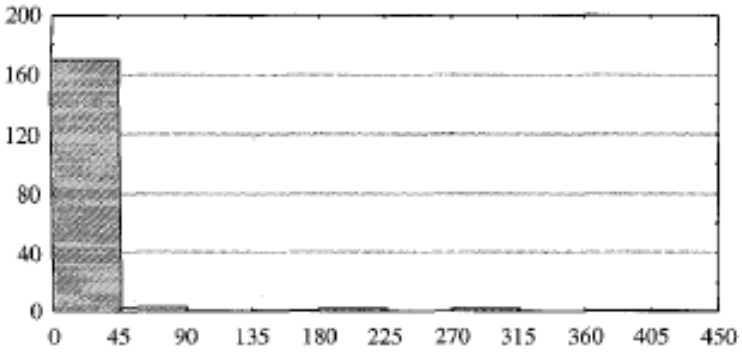
- **LCOM – lack of cohesion of methods**

- **Chidamber Kemerer:**

$$LCOM(C) = \begin{cases} P - Q & \text{if } P > Q \\ 0 & \text{otherwise} \end{cases}$$

where

- **P = #pairs of distinct methods in C that do not share variables**
- **Q = #pairs of distinct methods in C that share variables**



[BBM] 180 classes

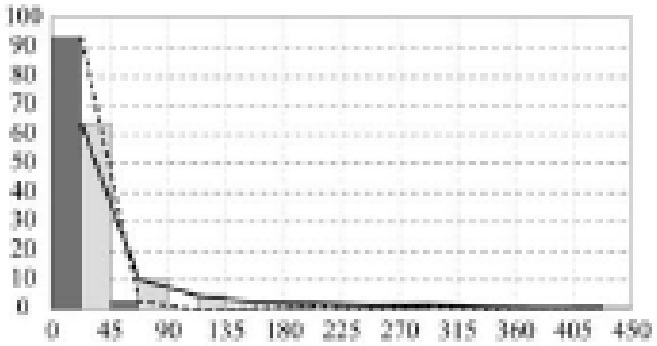
Discriminative ability is insufficient

What about get/set?

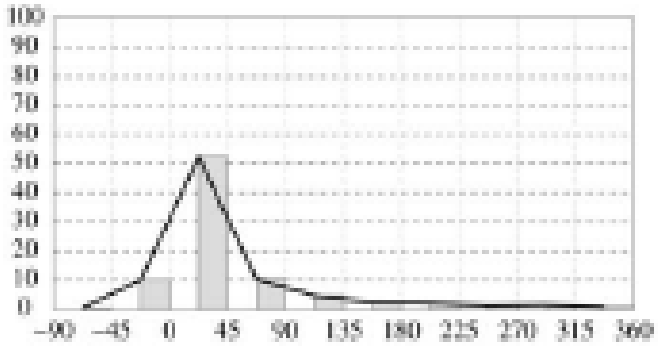
First solution: LCOMN

- **Defined similarly to LCOM but allows negative values**

$$LCOMN(C) = P - Q$$

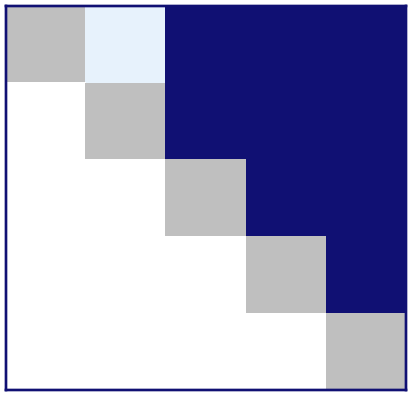


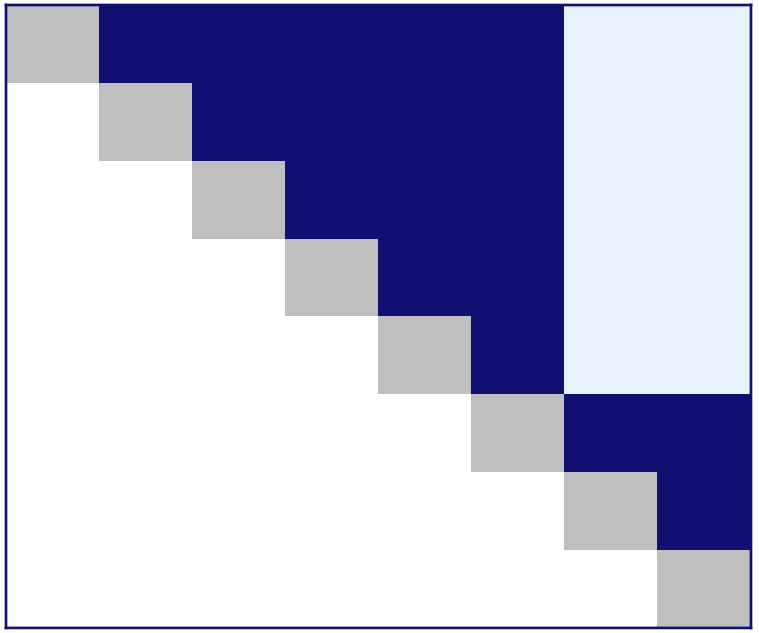
LCOM



LCOMN

Still…





- **Method * method tables**
  - **Light blue: Q, dark blue: P**
- **Calculate the LCOMs**
- **Does this correspond to your intuition?**

Henderson-Sellers, Constantine and Graham 1996

- **m – number of methods**
- **v – number of variables (attrs)**
- **m(V_i) - #methods that access V_i**

$$\frac{\left(\frac{1}{v}\sum_{i=1}^{v} m(V_i)\right) - m}{1 - m}$$

- **Cohesion is maximal: all methods access all variables**

$$m(V_i) = m \text{ and } LCOM = 0$$

- **No cohesion: every method accesses a unique variable**

$$m(V_i) = 1 \text{ and } LCOM = 1$$

- **Can LCOM exceed 1?**

LCOM > 1?

- **If some variables are not accessed at all, then**

$$m(V_i) = 0$$

and

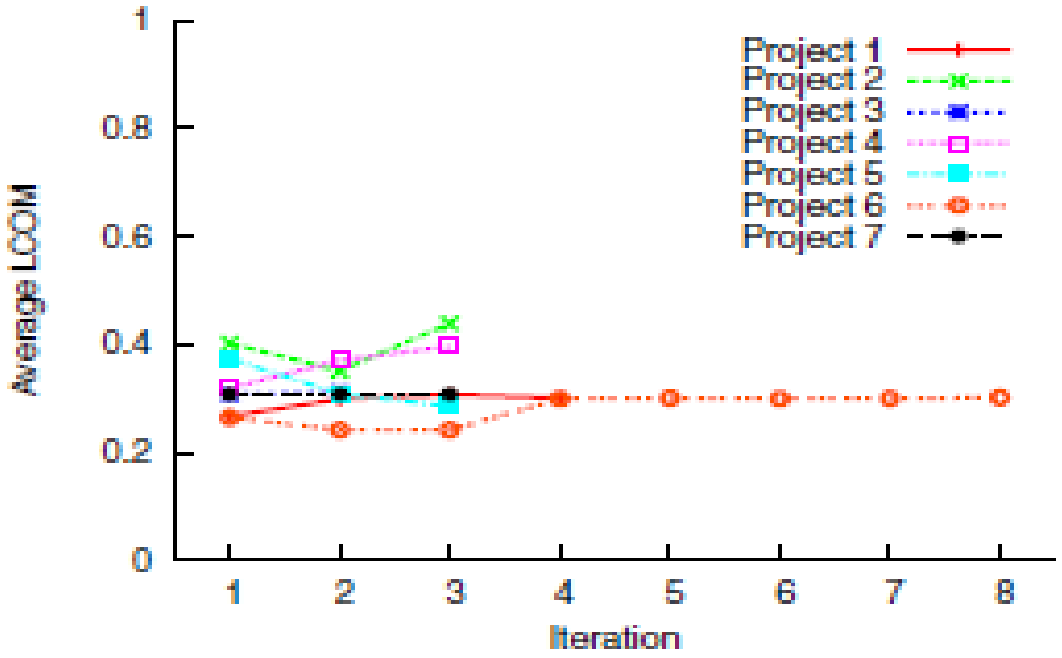
$$\frac{\left(\frac{1}{v}\sum_{i=1}^{v} m(V_i)\right) - m}{1-m} = \frac{-m}{1-m} = 1 + \frac{1}{m-1}$$

Hence

LCOM is undefined for m = 1

LCOM ≤ 2

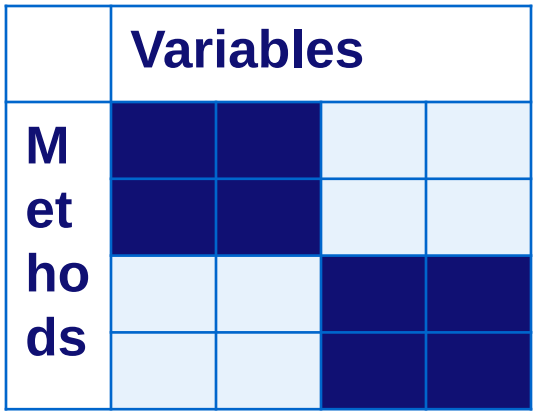
Evolution of LCOM [Henderson-Sellers et al.]

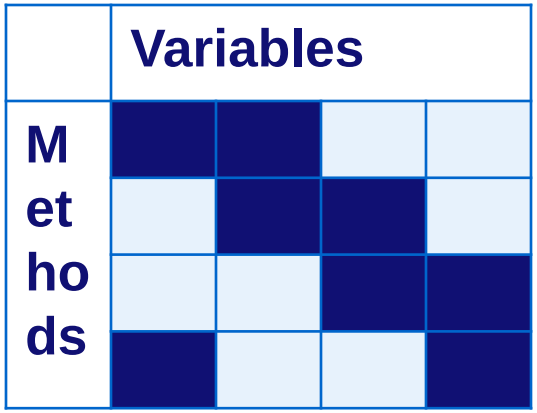


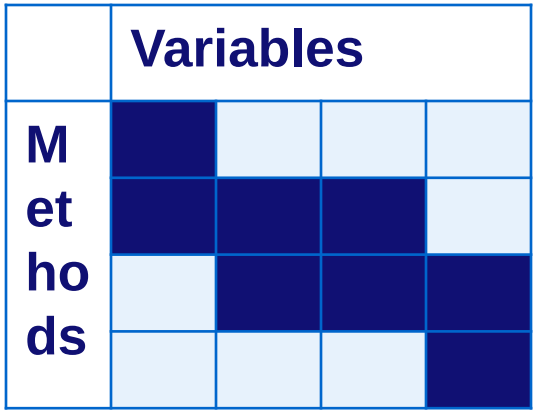
Sato, Goldman, Kon 2007

- **Project 6 (commercial human resource system) suggests stabilization, but no similar conclusion can be made for other projects**

Shortcomings of LCOM [Henderson-Sellers]







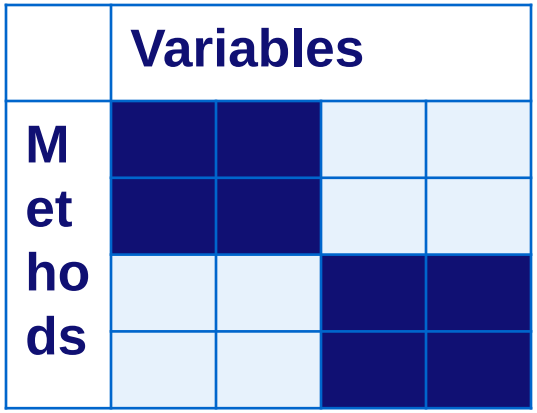
- Due to [Fernández, Peña 2006]
- Method-variable diagrams: dark spot = access
- LCOM( ) = LCOM( ) = LCOM( ) = 0.67

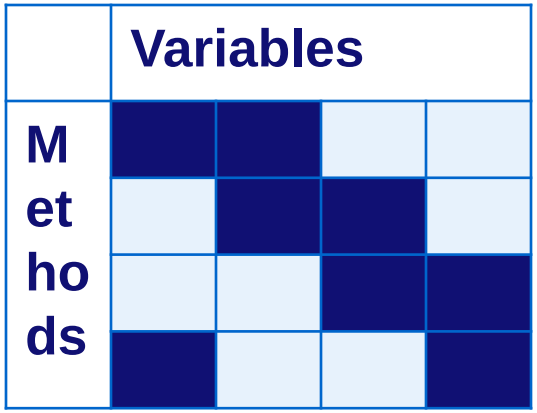
  seems to be less cohesive than and !

Alternative [Hitz, Montazeri 1995]

- LCOM as the number of strongly connected components in the following graph
  - Vertices: methods
  - Edge between a and b, if
    - a calls b
    - b calls a
    - a and b access the same variable
- LCOM values
  - 0, no methods
  - 1, cohesive component
  - 2 or more, lack of cohesion

Question: LCOM?





Experimental evaluation of LCOM variants

Cox, Etzkorn and Hughes 2006	Correlation with expert assessment	
	Group 1	Group 2
Chidamber Kemerer	-0.43 (p = 0.12)	-0.57 (p = 0.08)
Henderson-Sellers	-0.44 (p = 0.12)	-0.46 (p = 0.18)
Hitz, Montazeri	-0.47 (p = 0.06)	-0.53 (p = 0.08)

Etzkorn, Gholston, Fortune, Stein, Utley, Farrington, Cox	Correlation with expert assessment	
	Group 1	Group 2
Chidamber Kemerer	-0.46 (rating 5/8)	-0.73 (rating 1.5/8)
Henderson-Sellers	-0.44 (rating 7/8)	-0.45 (rating 7/8)
Hitz, Montazeri	-0.51 (rating 2/8)	-0.54 (rating 5/8)

LCC and TCC [Bieman, Kang 1994]

- Recall: LCOM HM "a and b access the same variable"
- What if a calls a', b calls b', and a' and b' access the same variable?
- Metrics
  - NDP – number of pairs of methods directly accessing the same variable
  - NIP – number of pairs of methods directly or indirectly accessing the same variable
  - NP – number of pairs of methods: n(n-1)/2
- Tight class cohesion TCC = NDP/NP
- Loose class cohesion LCC = NIP/NP
- NB: Constructors and destructors are excluded

Experimental evaluation of LCC/TCC

Etzkorn, Gholston, Fortune, Stein, Utley, Farrington, Cox	Correlation with expert assessment	
	Group 1	Group 2
Chidamber Kemerer	-0.46 (rating 5/8)	-0.73 (rating 1.5/8)
Henderson-Sellers	-0.44 (rating 7/8)	-0.45 (rating 7/8)
Hitz, Montazeri	-0.51 (rating 2/8)	-0.54 (rating 5/8)
TCC	-0.22 (rating 8/8)	-0.057 (rating 8/8)
LCC	-0.54 (rating 1/8)	-0.73 (rating 1.5/8)

Conclusions: Metrics so far…

Level	Matrics
Method	LOC, McCabe, Henry Kafura
Class	WMC, NOC, DIT, LCOM (and variants), LCC/TCC
Packages	???

Next time:

- **Package-level metrics (Martin)**
- **Metrics of change**

GLA UNIVERSITY
MATHURA

Accredited with A Grade by NAAC

Job Oriented Value-Added Course

on

Hybrid Mobile App Development With Flutter (Google)

Certificate of Participation

This certificate is awarded to **SHUBHANSHU NISHAD**, Roll No. **201510021** from **GLA University, Mathura**, for successfully completing the *Job Oriented Value-Added Course* on **Hybrid Mobile App Development With Flutter (Google)** as **Gold Medalist**, organized by Department of Computer Engineering & Applications (CEA) GLA University, Mathura (U.P.), India from **01st June, 2022** to **15th July, 2022**.

Dr. Rohit Agarwal
Head of Department

Mr. Rahul Pradhan
Course Coordinator