Agent Based models of Developer Interaction in Large Scale Software Ecosystems

A PROJECT REPORT

Submitted by:
Soumee Mukherjee, Roll No: 1951024

Under the Supervision of: Prof. Subhajit Datta
Associate Professor, Department of Computer Science and Technology

In Partial Fulfillment of the Requirements for the
Degree of
BACHELOR OF TECHNOLOGY

In
COMPUTER SCIENCE AND ENGINEERING

HERITAGE INSTITUTE OF TECHNOLOGY, MAULAN ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY
Kolkata, India

May, 2023

BONAFIDE CERTIFICATE

HERITAGE INSTITUTE OF TECHNOLOGY, KOLKATA
MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

Certified that this project report “Agent Based models of Developer Interaction in Large Scale Software Ecosystems” is a bonafide work of “ Miss Soumee Mukherjee” who carried out the project under my supervision.

_____	_____-
Prof. (Dr.)Suhashis Majumder	Prof. (Dr.)Subhajit Datta
Head of the Department	Associate Professor
Computer Science & Engineering	Computer Science & Engineering

_____

Examiner

MAY, 2023

ACKNOWLEDGEMENTS

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

We would take this opportunity to thank our principal, Prof. Dr. Basab Chaudhuri, for allowing us to form a group of four people and for providing us with all the necessary facilities to make our project work and of worth.

We will be thankful to our Head of Department Prof. Dr. Suhashis Majumder, for constantly supporting and giving us invaluable insights. His words of encouragement motivated us to achieve our goal and impetus to excel.

We are grateful to our project guide Prof Subhajit Datta for the guidance, inspiration and constructive suggestion that helped us in the partial fulfillment for the award of the degree.

ABSTRACT

The software development industry is now in full bloom. With the influx of multi-million dollar enterprise-level projects, like ChatGPT, or Autonumous Driving Cars, huge networks of developers are becoming more and more common behind such projects to drive them to completion. Along with human and organizational capital, social capital and networking are also necessary for participation in large-scale software development, both for novice teams and for mature teams working on complex, unfamiliar, or interdependent tasks. The thesis proposes a model with the help of netlogo, which is an agent based modelling platform. The model will be key to analysing significant network characteristics like Connection.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Chapter 1

INTRODUCTION

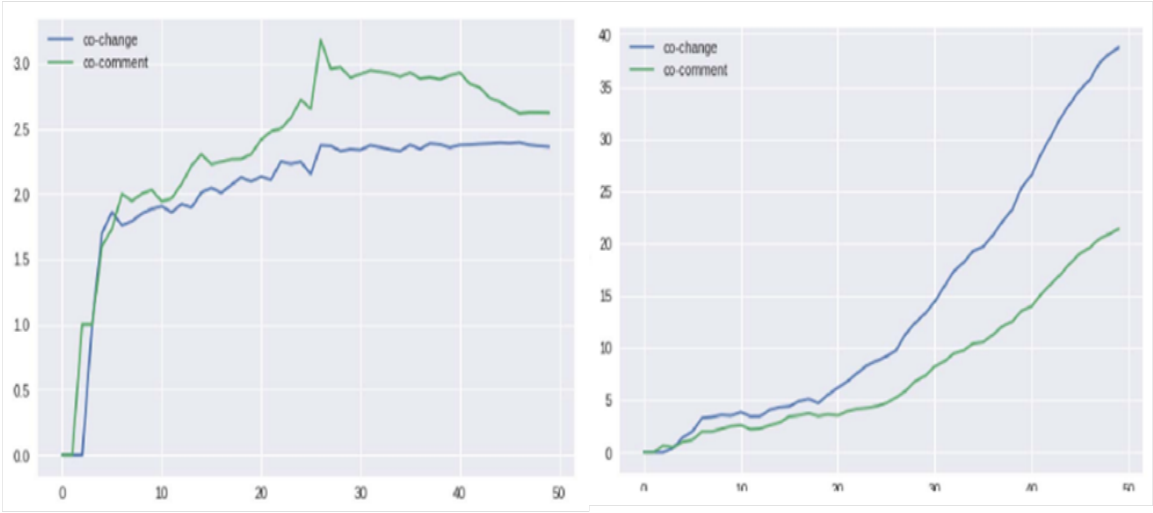
1.1 Context

The software development industry is now in full bloom. With the influx of multi-million dollar enterprise-level projects, like ChatGPT, or Autonumous Driving Cars, huge networks of developers are becoming more and more common behind such projects to drive them to completion. Along with human and organizational capital, social capital and networking are also necessary for participation in large-scale software development, both for novice teams and for mature teams working on complex, unfamiliar, or interdependent tasks. In fact, empirical findings from ten software teams from two large-scale software development projects in Ericsson and ABB demonstrated that teams receive and share their knowledge with a large number of contacts, including other team members, experts, administrative roles, and support roles[16]. The inherent invisibility and unvisualizability of software along with software essentially being "people-ware", however, [7, 8] leads to the typical complexities of large and distributed software development. Thus developer interaction is multi-faceted for any non-trivial system.[7] Analysing and understanding how these networks evolve with time can provide crucial insights into the development life cycle of many such projects. With this ideal in mind, researchers have tried to come up with various ways to understand these network characteristics, using some established metrics of Network Science like the Degree of Connection.

One of the several routes to follow, for scientific discovery can be to build a simulation of a phenomenon, in order to understand it. In order to build a simulation for our particular problem setting of Large -scale software development, we will need to take into account behaviour embedded in complex networks of personal relationships, communities and markets. The reason is that the timeline and outcome of software development projects are not just dependent on individualistic action by the collective decisions made by a team or the groups of teams involved with the project. To understand these complex social processes of our time, it is essential that research draws attention both to human behaviour and to the structure of social networks and their dynamics. A promising approach to address these two aspects is the combination of social network analysis (SNA) and agent-based modelling (ABM)[15].

1.2 Motivation

Empirical studies by Social Network Analysts like Subhajit Datta et al.[7], have revealed some interesting insights into developer interactions as functions of time. They have made use of the OpenStack Data as published by Gonzalez-Barahona et al.[9], which is a popular open source cloud computing platform that is available for deployment as I-a-a-S(Infrastructure as a Service), processed and filtered it to produce the following set of graphs as in Fig 1.1.



(a) (b)

Figure 1.1: Network Characteristics: a) Separation b) Connection

However, these empirical findings with a clear dependency on time, as deduced from the graphs have yet to be explained theoretically which brings us to our research question:

Can we develop generative models using the agent based modelling(ABM) paradigm to explain developer interaction characteristics in large scale software ecosystems?

We sought to build a simplistic Agent-Based Model that will provide us with a theoretical understanding of what happened in these graphs. Our metric of interest here i.e., the Degree of Connection Metric[2, 7] as of Fig 1.1 b) is the one we sought to replicate with our models and this metric was chosen due to the inherent simplicity in being able to monitor the same.

1.3 Related Work

Conway conjectured that any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.[5]. The association between the structure of communication in a team, and structure of the product built by that team has thus long been an intriguing question. Empirical validation of the relationship between communication structure and product structure has attracted significant research interest in recent years.[14]. Datta. [6] analysed product development data to observe developer interactions and its impact on the quality of software and came to the conclusion that higher connection between team members leads to higher defects in the product. Wagstrom and Datta.[14] studied the effects of temporal separation and geographical separation between developers and found little impact of geographical separation on a developer's productivity.

Chapter 2

METHODOLOGY

2.1 Model Summary and Parameters

The main model was built using Netlogo. NetLogo is a multiagent programming language and modeling environment for simulating complex phenomena[3, 4]. The model records the temporal variation of a well studied network metric called the **Degree of Connection and Degree of Separation.**[7] A variety of models were considered before we came upon our present model with the 5 parameters that we input as users and they are mentioned subsequently. The parameters are: **rate-of-Connection**, the rate at which the developer get connected, **num-people**, initial Number of people in the team, **num-teams**, number of teams that exist, **add-people**, number of people to be added to be the team after each time step and **rate of Inter Team Connection**, number of connections between teams that can crop up. The model presently provides us with a graph for the degree of connection and separtion that has a striking resemblance to the empirical curve of the Degree of Connection and Separation metrics respectively, thereby lending more support to the model to be an approximate model of the real world scenario for developer interactions.

2.2 Model Inspiration

The model is inspired by the caveman graph[11, 13] as depicted in Fig 2.1. In this model, the software development world in divided into teams and each team has a manager. The constraints are that only the managers can of each team can be communicate to the other managers in other teams. But the developers in a team can only talk to their teammates but not their fellow developers from other teams.

These underlying assumptions are also very consistent with what we observe in real life software development scenarios. An enterprise level application is often broken down into several modules where each of the modules are developed by separate teams. There is often little need for individual developers developing separate modules to interact. Heads of teams often bear the responsibility for co-ordination between different modules.

Two parameters, (named num-people and num-team), set up the world with the initial values of the number of developers in each team and the number of teams/modules that exist respectively for this project. After each iteration of the model, connections are generated between team members with a constant probability (named rate-of-connection) from the number of people a developer has in his team. Also one random team out of the number of teams that exist gets a constant number of people added to their team.

Attached below in Fig:2.2 is the Netlogo Interface that was built. The sliders represent the variable

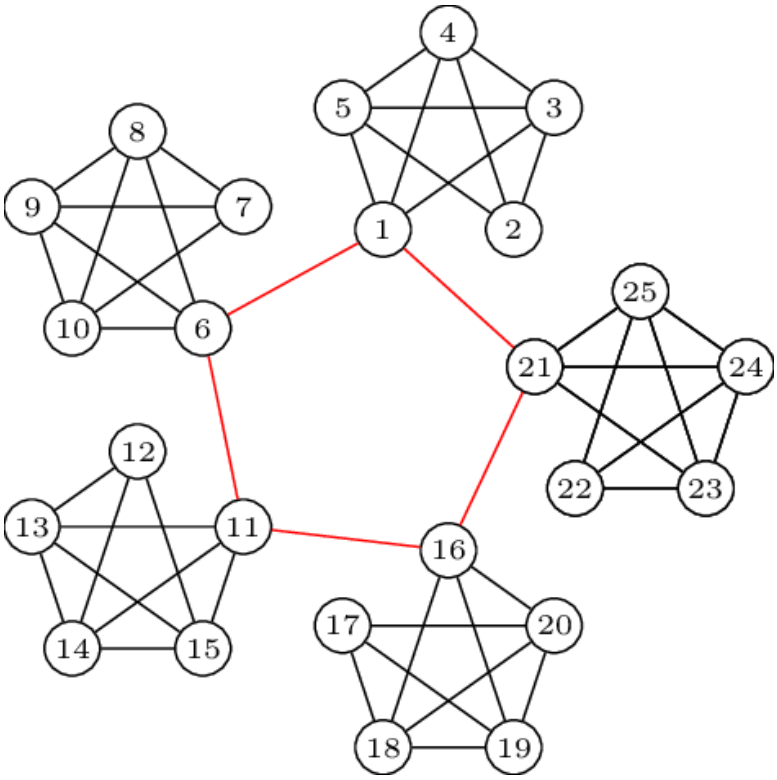


Figure 2.1: Caveman Graph

parameters in our model. The setup calls to setup the features of the world with the aforementioned parameters. The Go-button is used to run the model given on these parameters.

2.3 Key Points for Social Nework Analysis

Social Networks as Graphs In order for a network to be represented on computer, a graph structure can be a fair representation. There is extensive literature where social networks have modelled as graphs and their characteristics dealt with using graph mining techniques and algorithms.[1, 12] Development networks can similarly be thought of graphs with the developers being the vertices of the graph and the connections between the developers being the edges in the graph. The network parameter of interest here is Connection and Separation[2].

Connection This metric has a numeric formula, derived by empirical models,

$$DegreeOfConnection = \frac{2*NumberOfConnections}{NumberOfDevelopers}$$

This same equation can be remodelled into counting the edges and vertices of the graph, to calculate our desired metric and members of network can be programatically tweaked using traditional graph algorithms for graph traversals like Breadth First Search and Depth First Search. The model too was similarly set up on Netlogo as a graph and the computations on the network can be performed using similar recursion based algorithms.

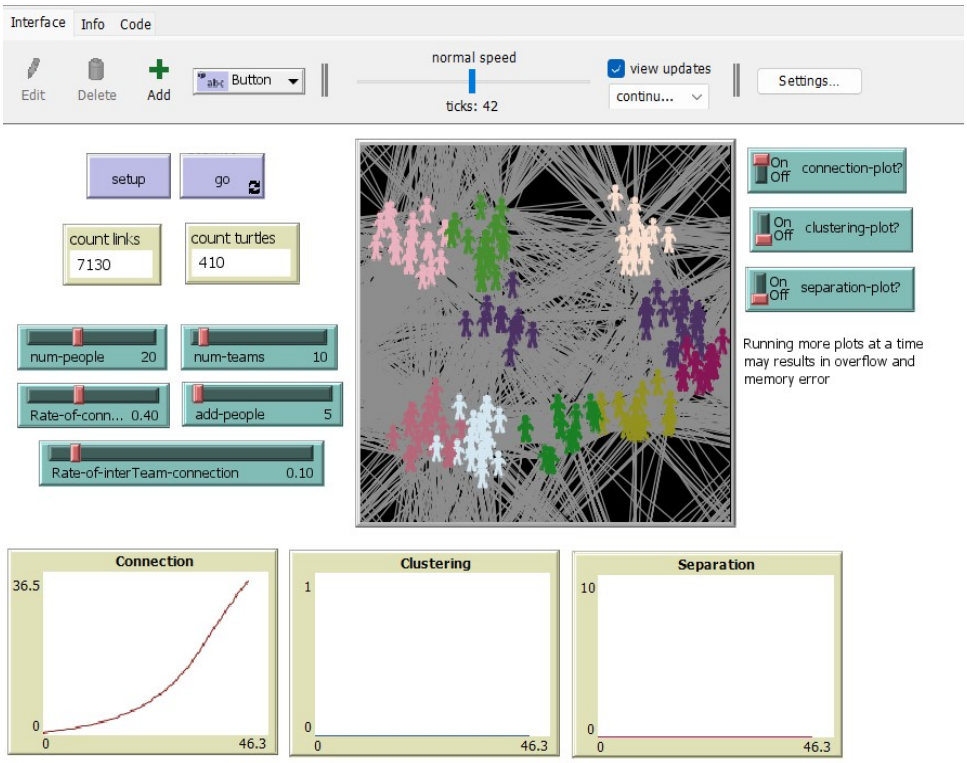


Figure 2.2: Input Interface

Separation The ease with which two vertices i and j in a network can communicate with one another is influenced by the length of the shortest path lij between them. The average of lij taken over all pairs of vertices is the average separation of the network. We take this to denote the degree of separation between developers in this study

2.4 Utility Tool

Given the above modifiable parameters, Netlogo offers a tool called the BehaviourSpace, which lets the users run experiments on the model with a range of parameters at the same time. The results get recorded in a CSV file with the the corresponding values after each run for each combination of parameters mentioned. However, the data cannot always be visualised as most programs like Excel do not allow exploring such huge datasets and manually parsing them can be extremely tiresome as well.

To automate the process and to generate visually appealing dynamic plots, we attempted to build a User Interface, built and deployed on the streamlit cloud that can appropriately parse the csv files. Snapshots of the utility tool are attached below through Fig 2.3 to Fig 2.4. The code is available in a Github repository and can be downloaded from: https://github.com/SOUMEE2000/Netlogo-Models-of-Developer-Interaction

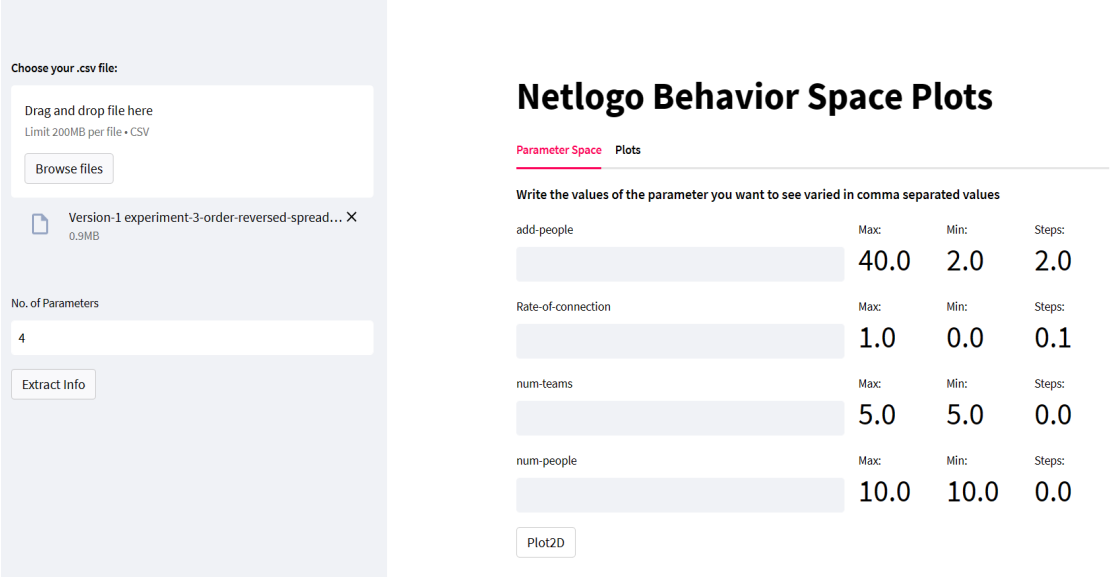


Figure 2.3: Input Interface

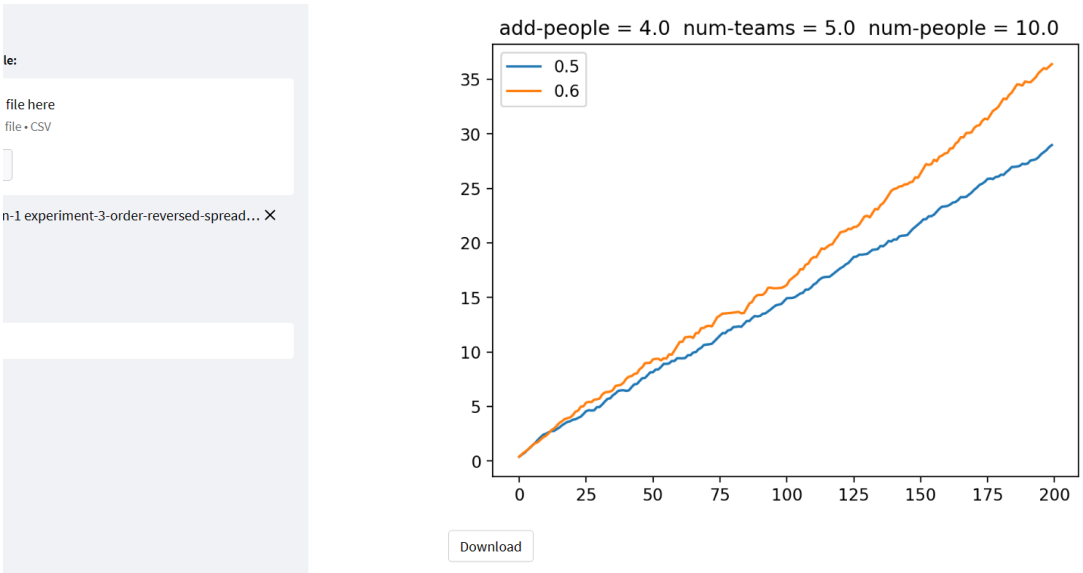


Figure 2.4: Output Interface

RESULTS

Mentioned below are some of the results that the Utility tool provided after reading the CSV file generated by our Netlogo Model after a few combinations of input parameters to the model.

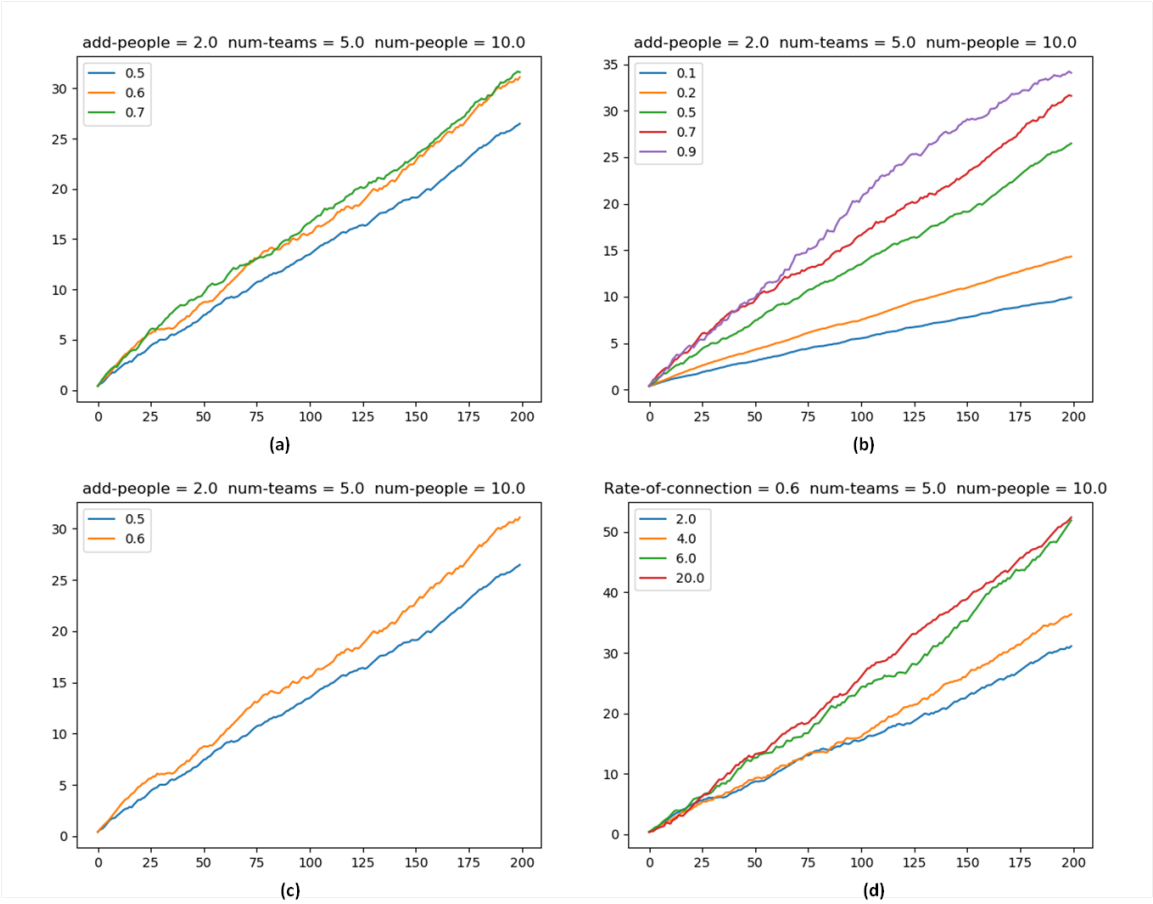


Figure 3.1: Results: (a), (b) and (c). Rate of Connection varying over time (d) Number of people added varying over time

3.1 Empirical Curve Analysis

Connection It is to be noted that the empirical curve of degree of connection, Fig 3.2 derived from real world network data[7] has 3 distinct stages. In the first stage, the degree of connection grows at a steady rate, which is depicted in the initial stages of the graph. The graph then somewhat stagnates in the second stage and is then followed by an exponential growth.

Separation As for the empirical graph of the separation metric, Fig 3.3, the curve shoots up in the first few instances after it stagnates a bit and keeps on rising steadily.

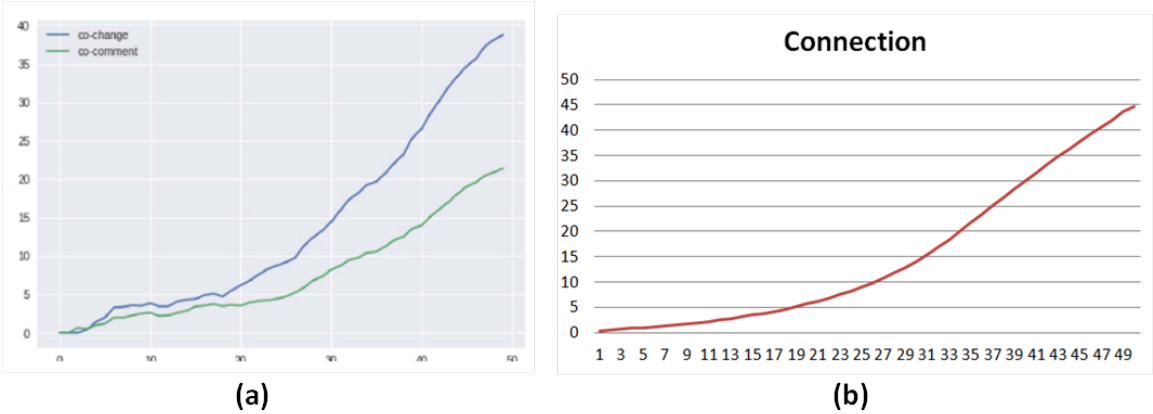


Figure 3.2: Results: Degree-of-Connection (a): Empirical Results (b): Our Results

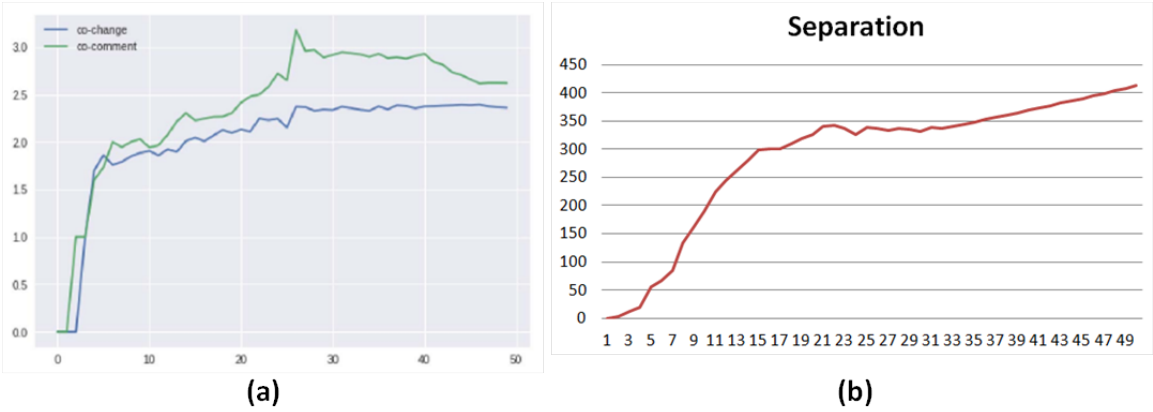


Figure 3.3: Results: Degree-of-Separation (a): Empirical Results (b): Our Results

3.2 Our Results

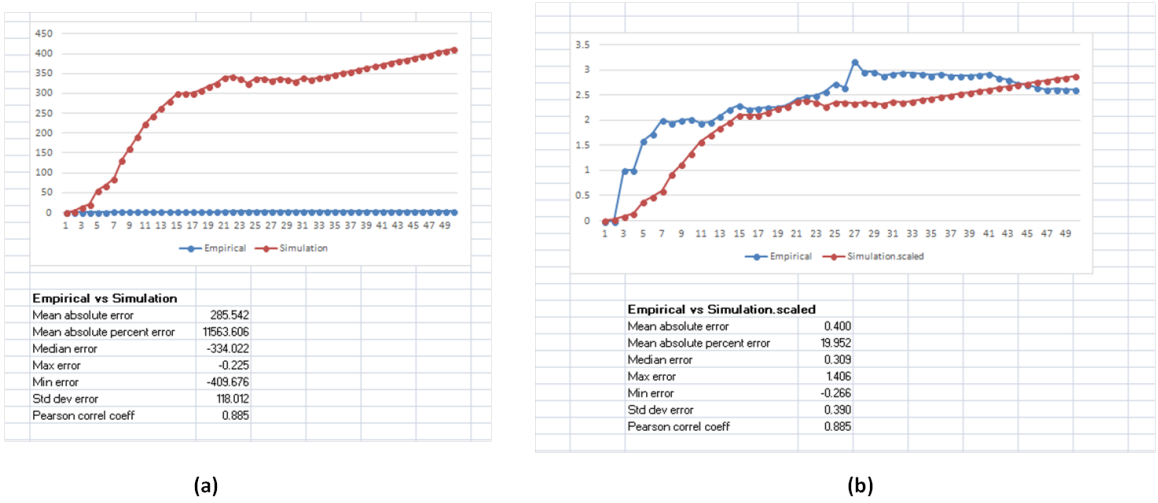
Upon a further inspection of the plots from the Utility tool, we came to some interesting results. With value of the parameters, **rate-of-Connection = 0.4**, **num-people = 20**, **num-teams = 10**, **add-people = 5**, **and rate-of-interTeam Connection = 0.1** we got really interesting results, as depicted in Fig:3.2 (b). The graph from our minimalist model also has those 3 distinct stages and the 2 distinct stages as portrayed displayed in Fig:3.3 (b)

3.3 Scaling and Model Validation

It is clear that reality can not be modelled by such a simplistic example. [6]. The addition of a linear scaling factor can aid in being able to attain more realistic values, in congruence with the real-world values available. The data obtained from the OpenStack dataset [9] as discussed above have been used to get the empirical findings in blue while our simulations are represented by the graph in red after having been scaled appropriately with the scalar numbers of 0.5 and 0.007 for Connection and Separation respectively.

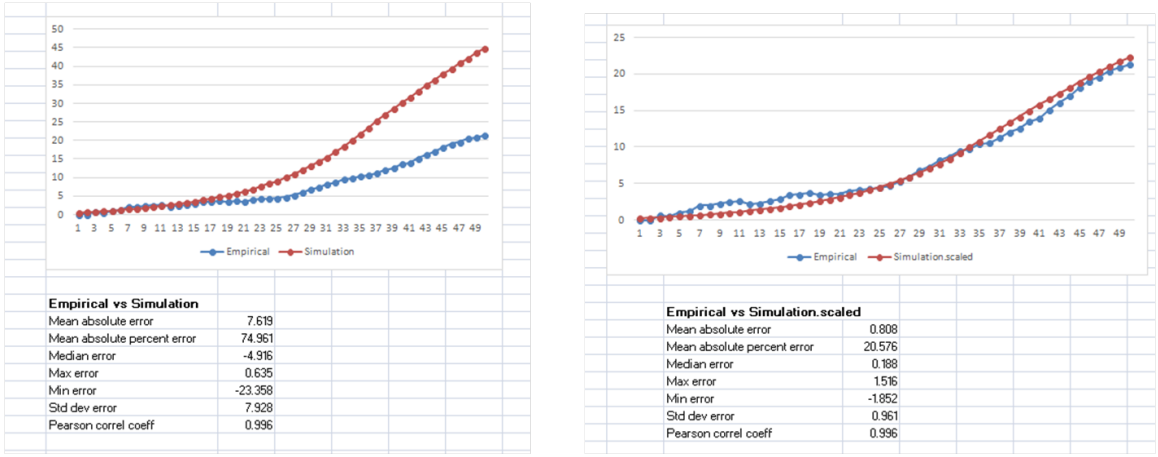
The figures 3.4 and 3.5 provides a depiction of how each of the metrics as proposed by our simulated

model looks like with respect to the graphs of the empirical data both scaled and unscaled. Having scaled the results, the *Mean Absolute Error*(MAE) between the empirical and scaled simulated graph of Connection is 0.808 and *Pearson correlation* coefficient of 0.996 . For the separation graphs (empirical vs simulation scaled, we get a mean square error of 0.400 and Pearson Correlation coefficient of 0.885.



(a) (b)

Figure 3.4: Results: Degree-of-Separation, scaling factor = 0.007 (a): Empirical vs Simulation Results (b): Empirical vs scaled Results



(a) (b)

Figure 3.5: Results: Degree-of-Connection, scaling factor = 0.5 (a): Empirical vs Simulation Results (b): Empirical vs scaled Results

Chapter 4

SUMMARY AND CONCLUSION

In conclusion, we have presented a model which has some capacity to be able to explain a few of these characteristics of a social network and thus be a simpler explanation of the growth that can happen in the context of large scale software development. Our simplistic has some underlying ideal assumptions which does leave room for debate in its generalizability in the real world.

However, we have extensively validated our model on the Openstack dataset on a number of metrics like Mean Absolute Error and Pearson Correlation coefficient to signify the likeness in the empirical graph and our simulated graphs. The dataset in question is vast in its expanse and has been used for research in the Software development ecosystem[7, 10] previously. This work will definitely be useful in providing some theoretical understanding how teams and development modules grow and evolve with time and thus provide a blue-print on

4.1 Plans for Future Work

In our future work, we plan to validate our model extensively on other datasets in order to improve on its generalizability. We are also looking to lessen our underlying assumptions by looking at our current problem in different ways and make the model be able to explain a higher number of characteristics than we are currently looking at.

CODE SNIPPETS

```
extensions [ nw ]
globals [colour counter node2-list existing-connections]
turtles-own [ team-number manager? visited?]

to setup
  clear-all
  setup-nodes
  setup-manager-links
  ;print existing-connections
  reset-ticks
end



to go
  make-team-connections
  make-inter-team-connections
  add-new-people
  ;print existing-connections
  tick
end
```

```
to-report Degree-of-Connection
  if connection-plot?
  [
    let count-links (count links)
    let count-people (count turtles)
    let connection (2 * count-links) / count-people
    report connection
  ]
  report 0
end

to-report Degree-of-Clustering
  ifelse clustering-plot?
  [
      let sum-clustering 0

      ask turtles [ set visited? false ]
      ask turtles with [count link-neighbors > 1]
      [
        let myNeighbors  link-neighbors

        let Kv ( count myNeighbors )
        let possibleEdges ( Kv * ( Kv - 1) / 2 )

        let actualEdges 0
        ask myNeighbors [ set visited? true ]

        ask myNeighbors
        [
          set actualEdges ( actualEdges + ( count link-neighbors with [ visited? = true] ) )
        ]

        ask myNeighbors [ set visited? false ]

        set actualEdges ( actualEdges / 2 )
```

Figure 5.1: Code Snippets: Netlogo

application.py

```
import streamlit as st
from backend import Backend_functions as bf
import io


if "extract_button" not in st.session_state:
    st.session_state["extract_button"] = 0
if "num_params" not in st.session_state:
    st.session_state["num_params"] = 0
if "plotted" not in st.session_state:
    st.session_state["plotted"] = 0
if "uploaded_files" not in st.session_state:
    st.session_state["uploaded_files"] = 0

with st.sidebar:
    uploaded_files = st.file_uploader('**Choose your .csv file:** ', type="csv", accept_multipl
    if uploaded_files is not None:
        st.session_state["uploaded_files"] = 1
    else:
        st.session_state["uploaded_files"] = 0
    st.write(" ")
    st.write(" ")
    num_params = st.text_input(label = "No. of Parameters")
    extract_button = st.button("Extract Info", disabled = not st.session_state["uploaded_files"
    if num_params and extract_button == 1:
        st.session_state["extract_button"] = 1
```

Figure 5.2: Code Snippets: GUI

REFERENCES

Bibliography

[1] N. K. Ahmed, F. Berchmans, J. Neville, and R. Kompella. Time-based sampling of social network activity graphs. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, MLG '10, page 1–9, New York, NY, USA, 2010. Association for Computing Machinery.

[2] A. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3):590–614, 2002.

[3] Q. A. Chaudhry. An introduction to agent-based modeling modeling natural, social, and engineered complex systems with netlogo: a review. *Complex Adaptive Systems Modeling*, 4(1):11, Jul 2016.

[4] F. Chiacchio, M. Pennisi, G. Russo, S. Motta, and F. Pappalardo. Agent-based modeling of the immune system: Netlogo, a promising framework. *BioMed Research International*, 2014:907171, Apr 2014.

[5] M. E. Conway. How do committees invent? *Datamation*, April 1968.

[6] S. Datta. How does developer interaction relate to software quality? an examination of product development data. *Empirical Software Engineering*, 23, 06 2018.

[7] S. Datta, A. Mysore, H. Wira, and S. Sarkar. Clustering, separation, and connection: A tale of three characteristics. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 669–673, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society.

[8] T. DeMarco and T. Lister. *Peopleware: Productive Projects and Teams*. Dorset House Publishing Co., Inc., USA, 1987.

[9] J. Gonzalez-Barahona, G. Robles, and D. Izquierdo-Cortazar. The metricsgrimoire database collection. pages 478–481, 05 2015.

[10] W. Huang, W. Zhang, D. Zhang, and L. Meng. Elastic spatial query processing in openstack cloud computing environment for time-constraint data analysis. *ISPRS International Journal of Geo-Information*, 6(3), 2017.

[11] Y. Lim, U. Kang, and C. Faloutsos. Slashburn: Graph compression and mining beyond caveman communities. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3077–3089, 2014.

[12] B. Rao and A. Mitra. A new approach for detection of common communities in a social network using graph mining techniques. In *2014 International Conference on High Performance Computing and Applications (ICHPCA)*, pages 1–6, 2014.

[13] Y. Tsai, P.-N. Hsiao, and C.-C. Lin. A social network model based on caveman network. In *2006 First International Conference on Communications and Networking in China*, pages 1–6, 2006.

[14] P. Wagstrom and S. Datta. Does latitude hurt while longitude kills? geographical and temporal separation in a large scale software development project. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, page 199–210, New York, NY, USA, 2014. Association for Computing Machinery.

[15] M. Will, J. Groeneveld, K. Frank, and B. Muller. Combining social network analysis and agent-based modelling to explore dynamics of human interaction: A review. *Socio-Environmental Systems Modelling*, 2:16325, Feb. 2020.

[16] D. Šmite, N. B. Moe, A. Šāblis, and C. Wohlin. Software teams and their knowledge networks in large-scale software development. *Information and Software Technology*, 86:71–86, 2017.