CollaborateR:
extract a collaboration graph from a version control system log

eRum 2020 – Parallel session 3 - Applications

3:00 PM - 3:15 PM

Wed June 17th, 2020

Leen Jooken

Joint work with Mathijs Creemers, Mieke Jans, Benoît Depaire and Gert Janssenswillen

UHASSELT

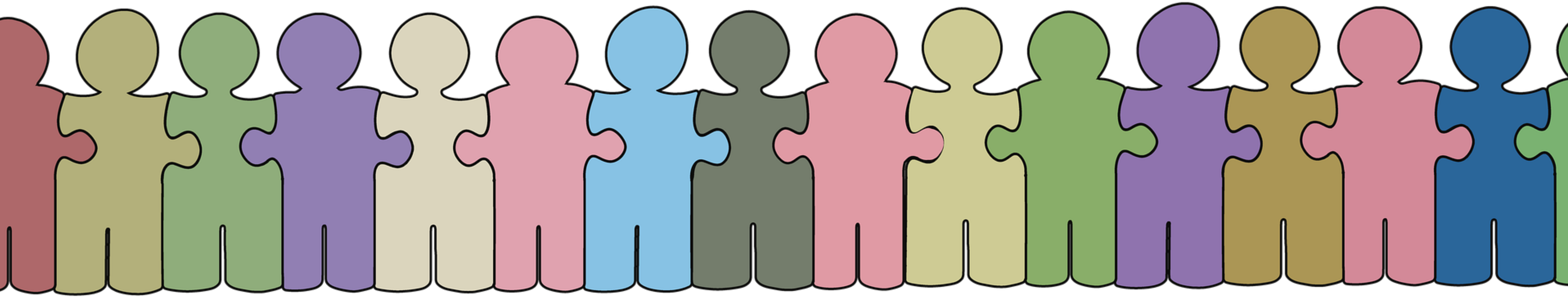
CollaborateR

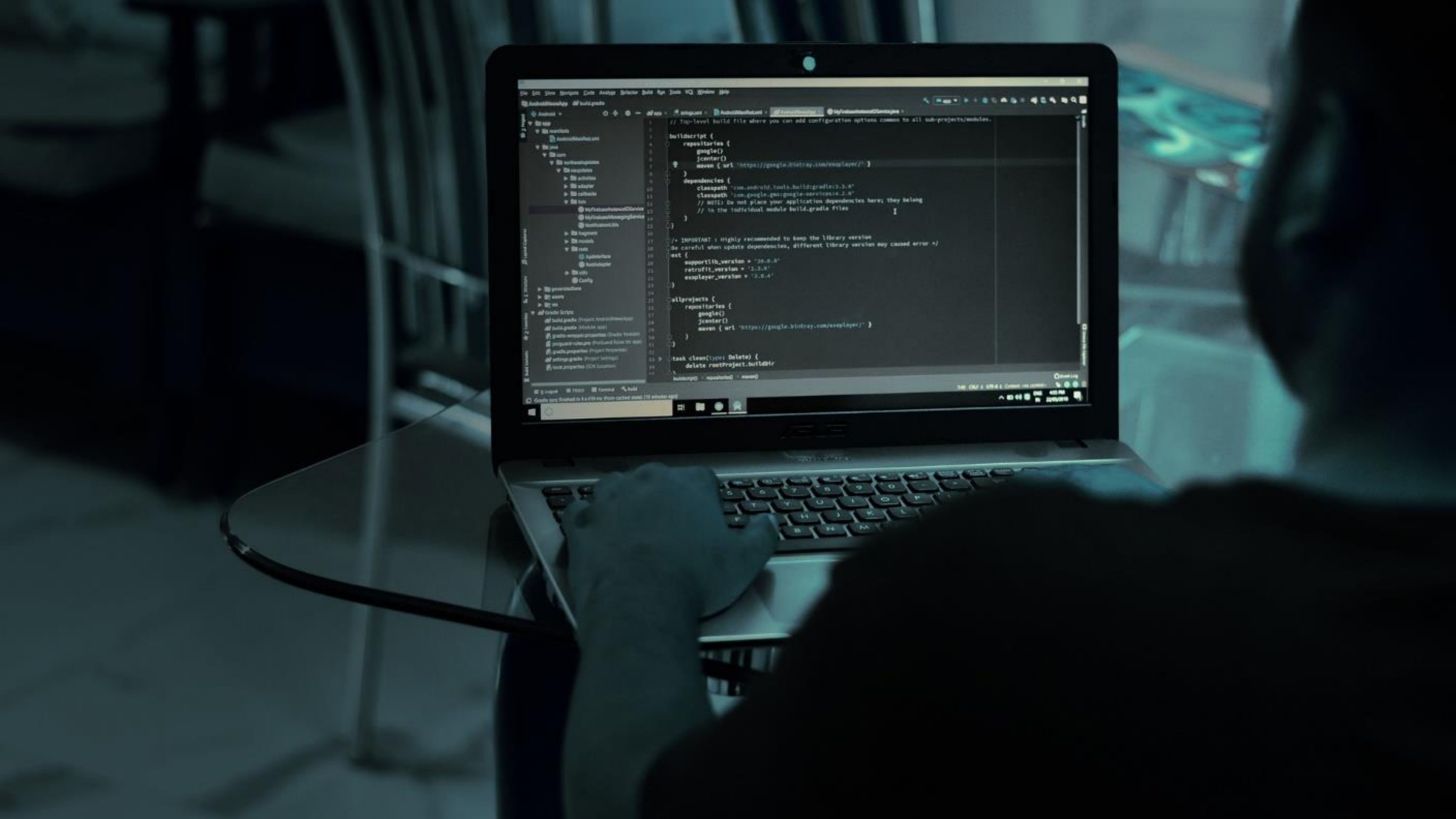
- Support knowledge management for software engineering environments
- Constructs collaboration graph

The Idea

The Idea

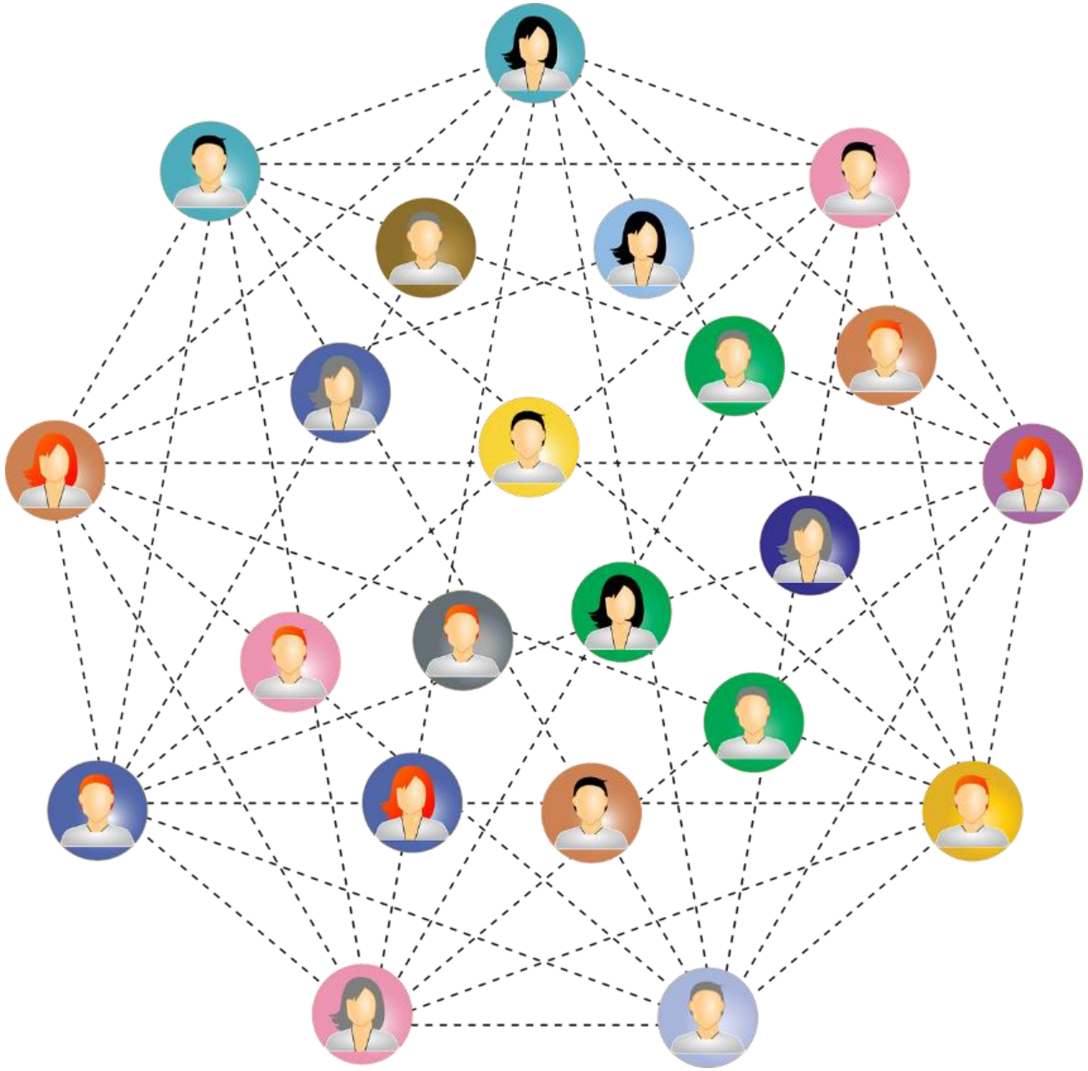
KNOWLEDGE MANAGEMENT
in software engineering environments




// Top-level build file where you can add configuration options common to all sub-projects/modules.
buildscript {
repositories {
google()
jcenter()
maven { url 'https://google.bintray.com/exoplayer/' }
dependencies {
// NOTE: Do not place your application dependencies here; they belong
// in the individual module build.gradle files
/* IMPORTANT : Highly recommended to keep the library version
Be careful when update dependencies, different library version may caused error */
allprojects {
task clean(type: Delete) {
delete rootProject.buildDir


buildscript {
repositories {
google()
jcenter()
dependencies {
allprojects {
repositories {
google()
jcenter()
task clean(type: Delete) {
delete rootProject.buildDir

The Idea



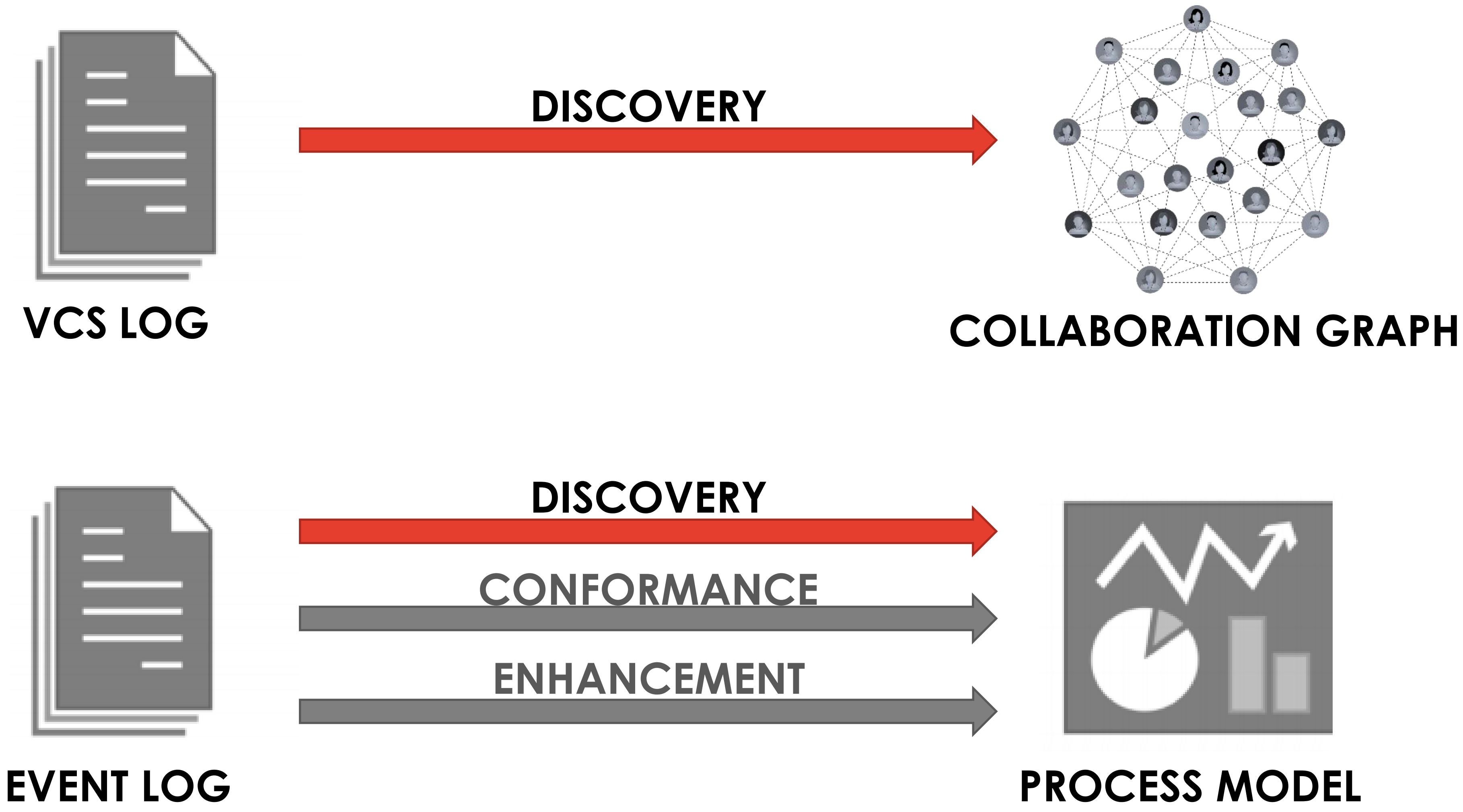
- Knowledge preservation
- General structure of collaboration
- Crucial Resources

The Data

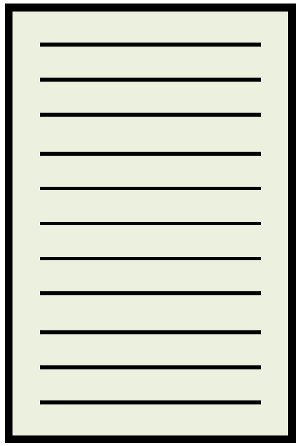
The Data

Inspiration

Inspiration

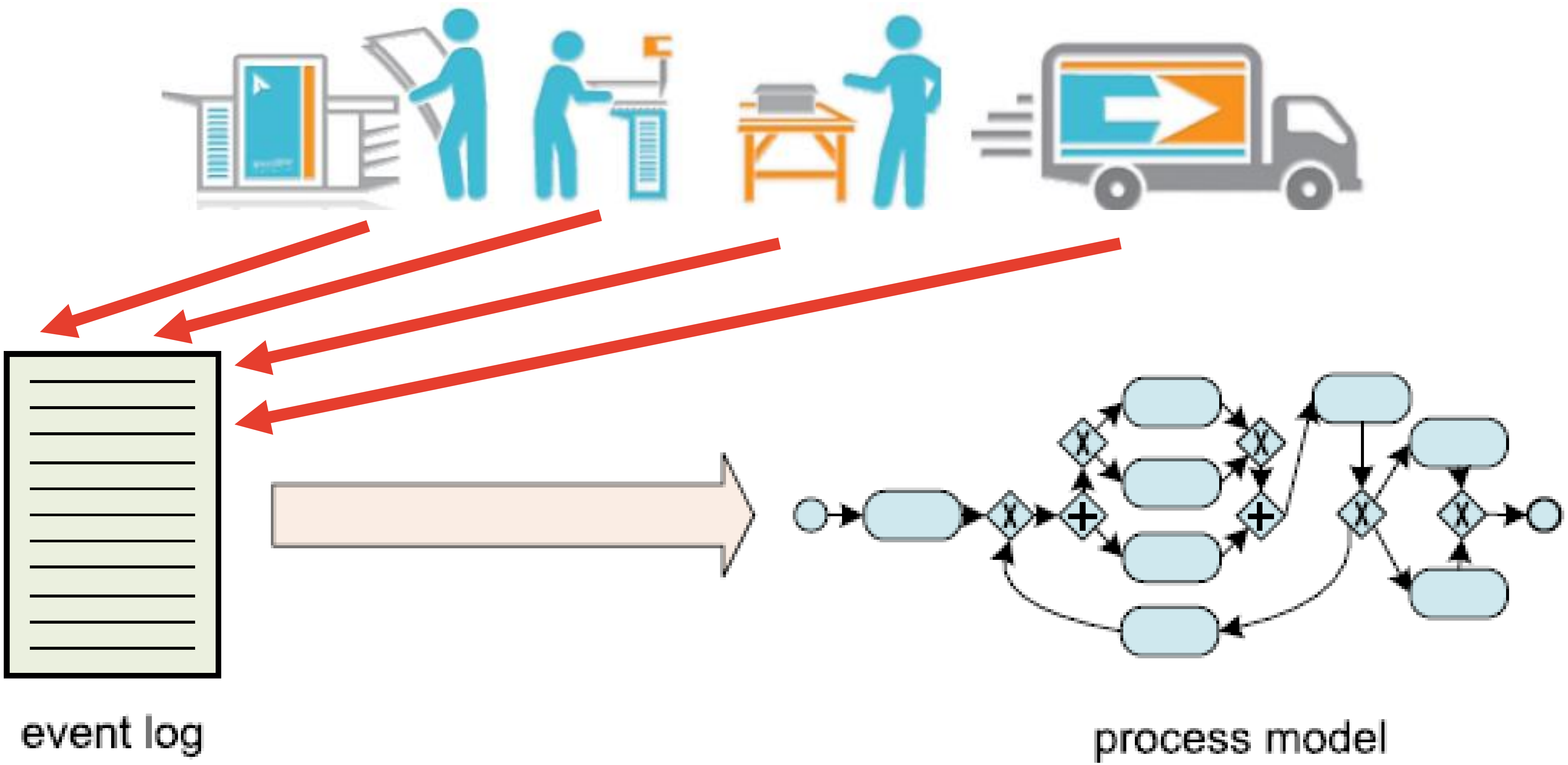


Inspiration

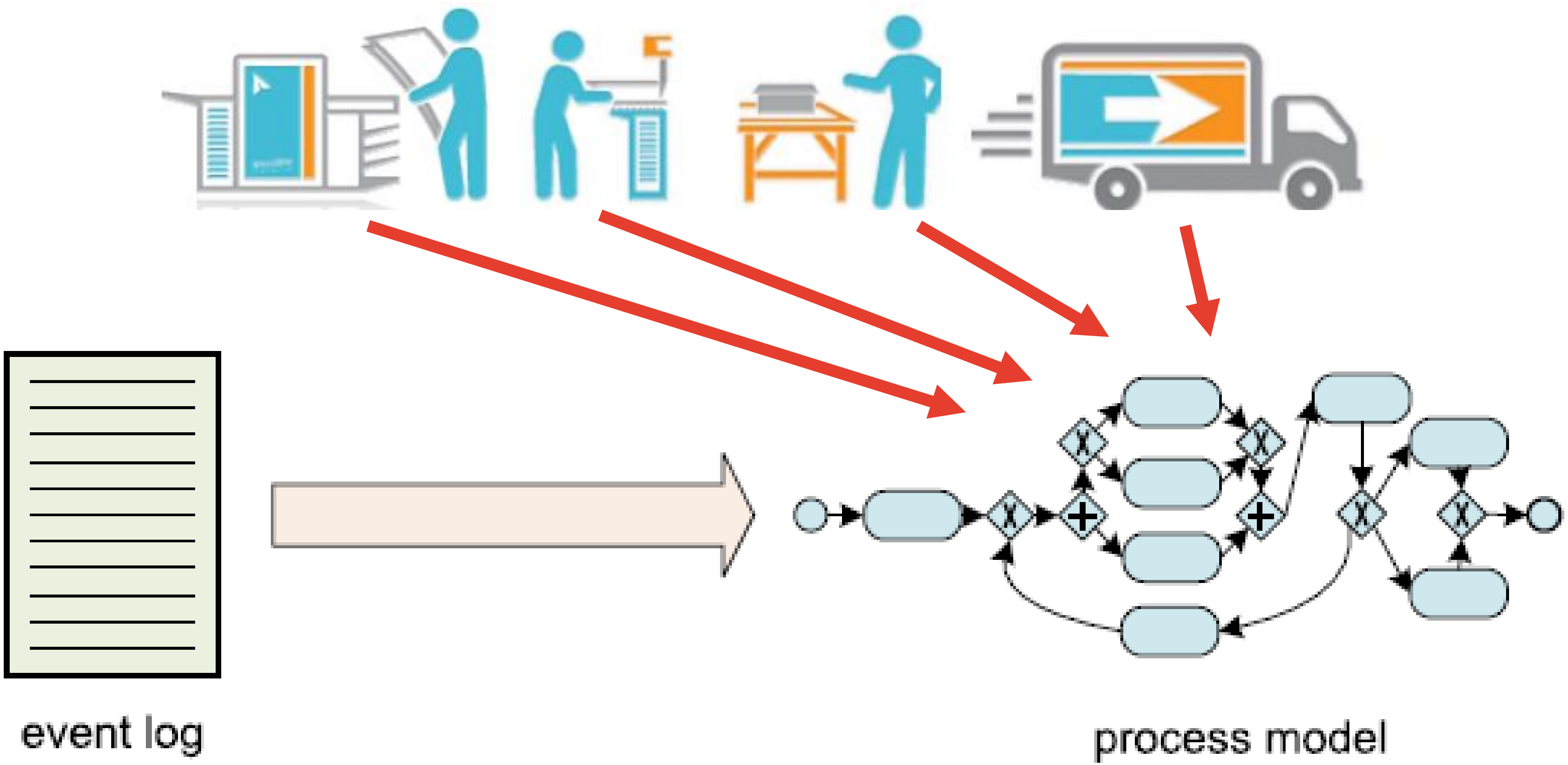


Inspiration

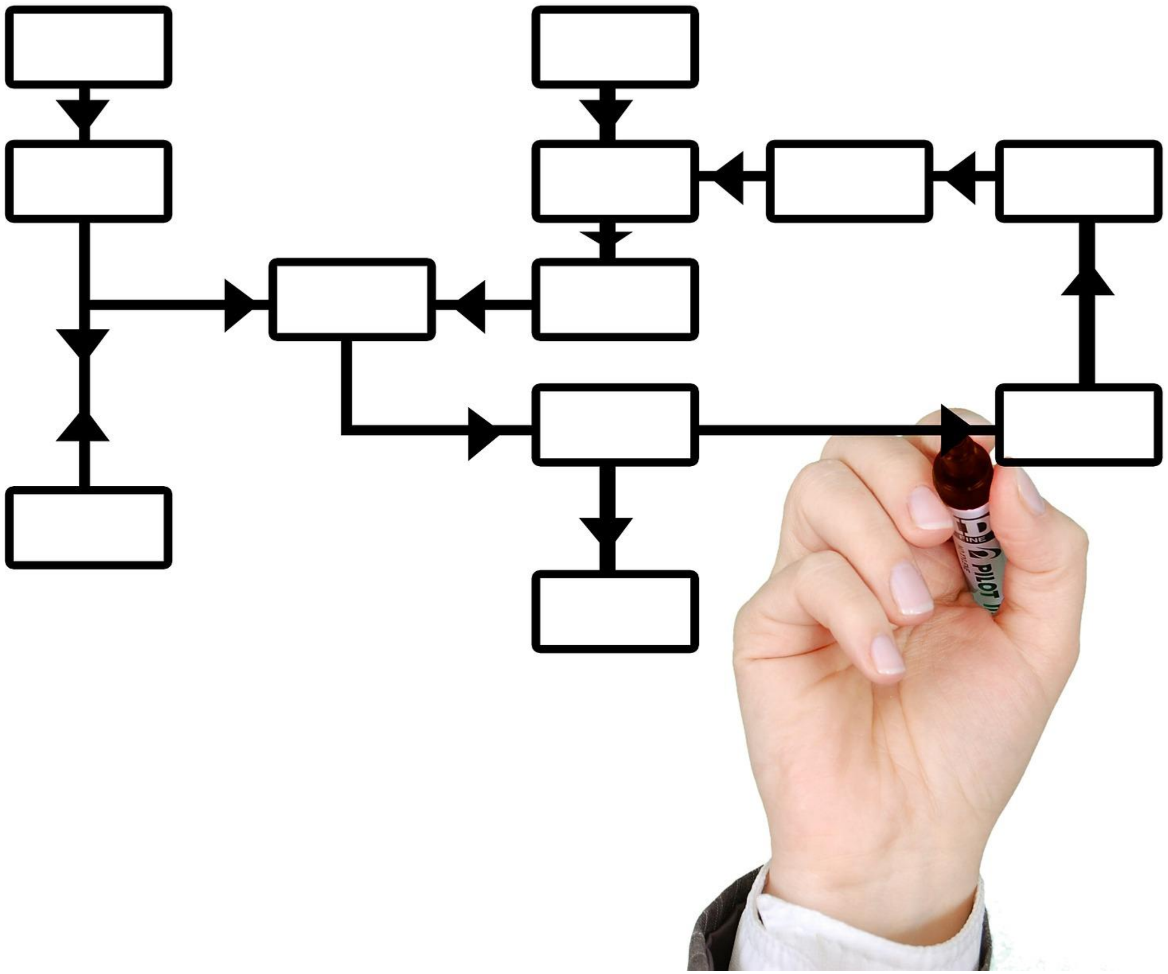
Inspiration



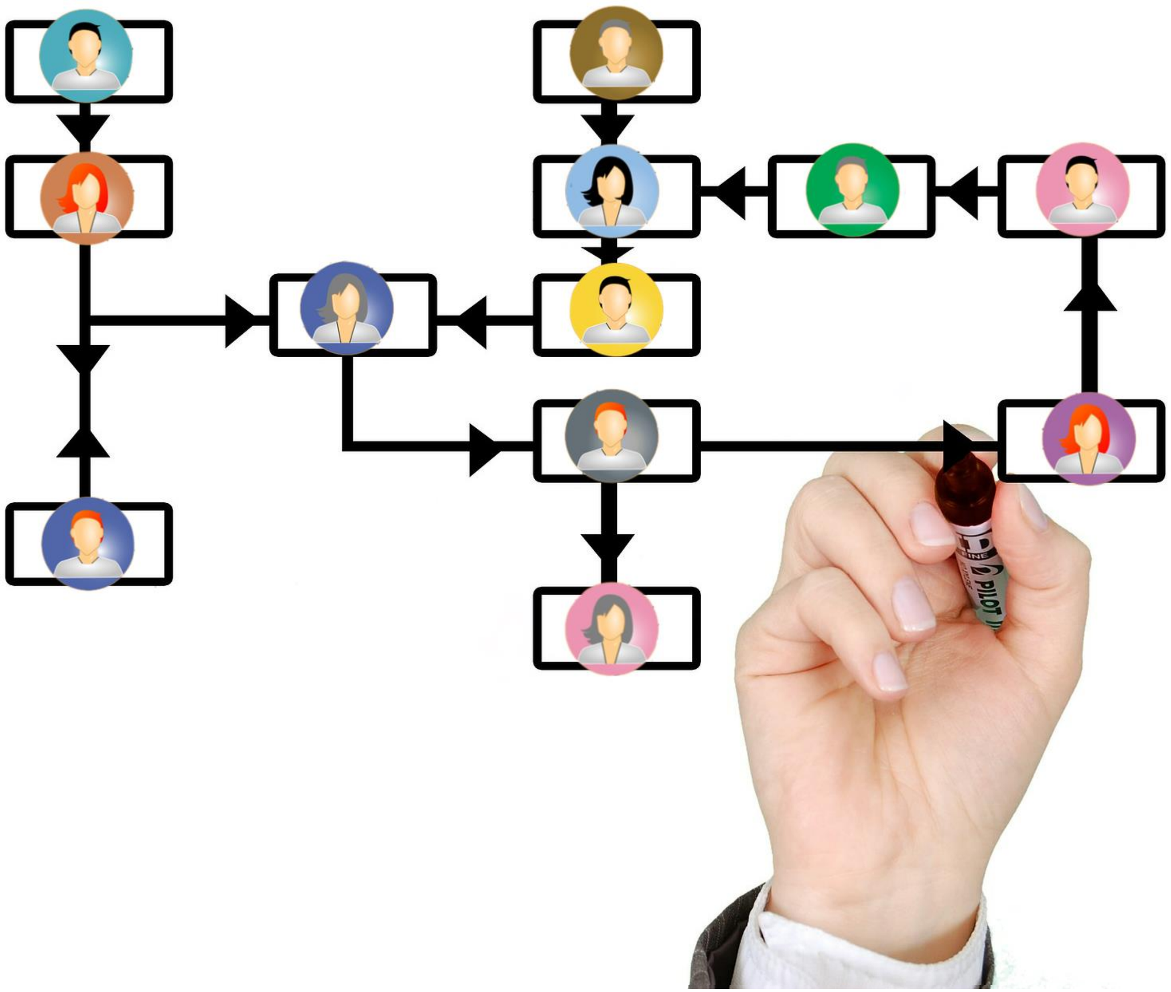
Inspiration



Inspiration



Inspiration



How To Use

How To Use

The input

How To Use

The input

FILE ID	ACTIVITY TYPE	TIMESTAMP	RESOURCE	REVISION	MODIFIER STATUS
1	Commit	25/03/2020 12:03:03	Jack Smidth	233	Modified
4	Commit	03/04/2020 16:37:12	Andy Joseph	234	Deleted
…	…	…	…	…	…

How To Use

The input

Business Process Analysis

bupaR

https://www.bupar.net

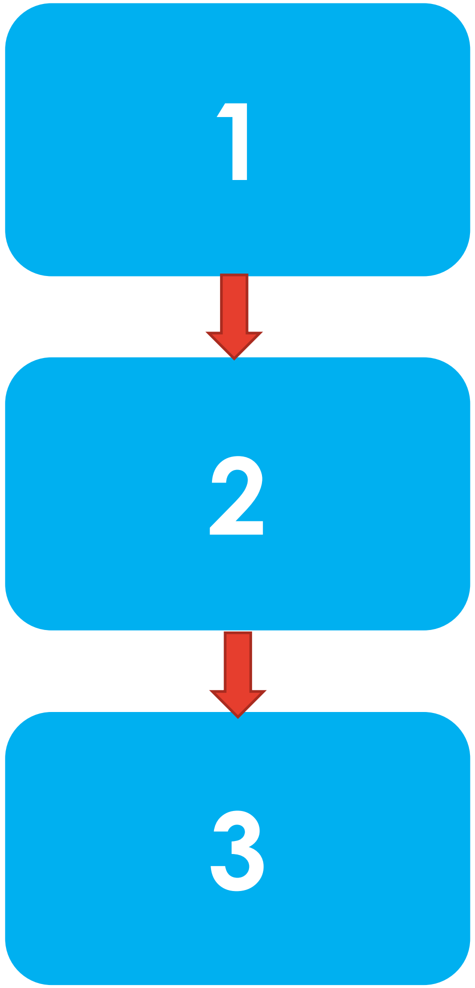
How To Use

The output:

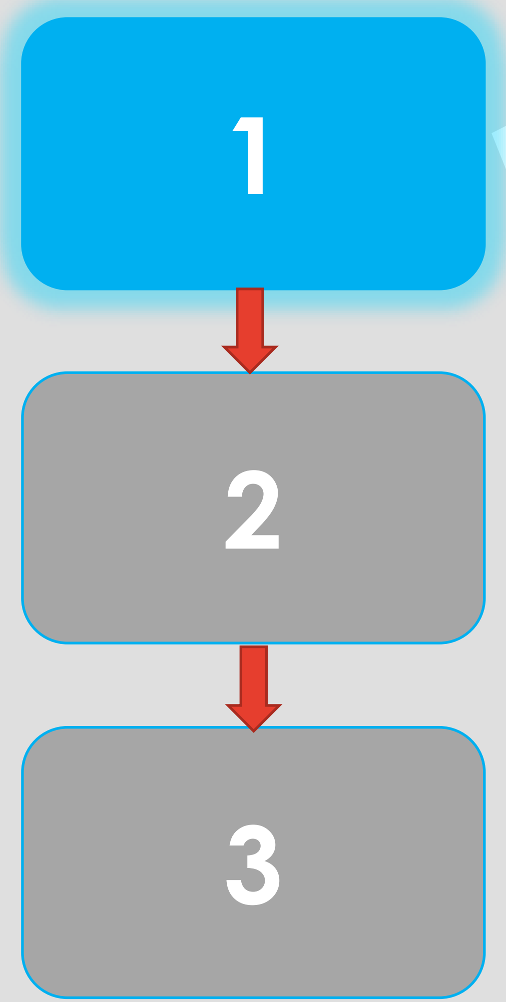
```
#read the log
log <- read_vcs_eventlog(filePath)

#build the graph
g <- build_graph(log, ... optional_parameters)
g$nodes
g$edges


#write graph to csv
writeGraphToCSV(g)

#visualize graph
visualizeGraph(g)
```

The Algorithm

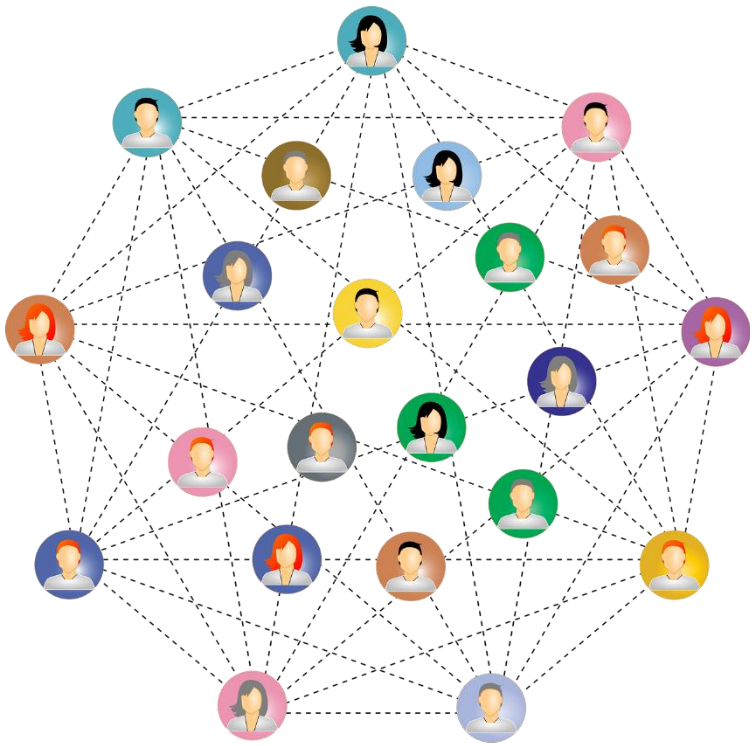
The Algorithm



Step 1: Building the Base Graph



- Include every programmer
- Add edge if file in common



Step 2: Calculating the Weights

2

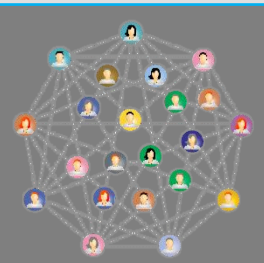
3

Purpose

1. Emphasis in the visualization

2. Simplification of the graph

Step 2: Calculating the Weights



2

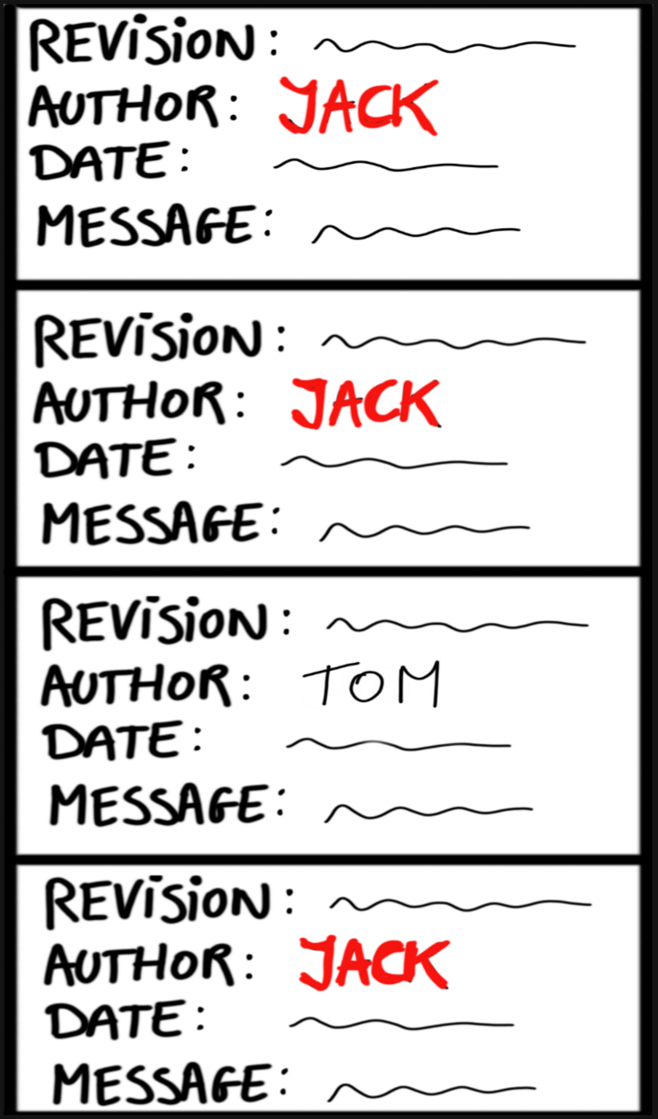
3

Importance of Programmer

$$\sum_{j=1}^{n} w_j \begin{cases} \text{Unary Frequency Significance} \\ \text{Betweenness Centrality} \\ \text{Eigenvector Centrality} \\ \text{Degree Centrality} \end{cases}$$

- Unary Frequency Significance
- Betweenness Centrality
- Eigenvector Centrality
- Degree Centrality

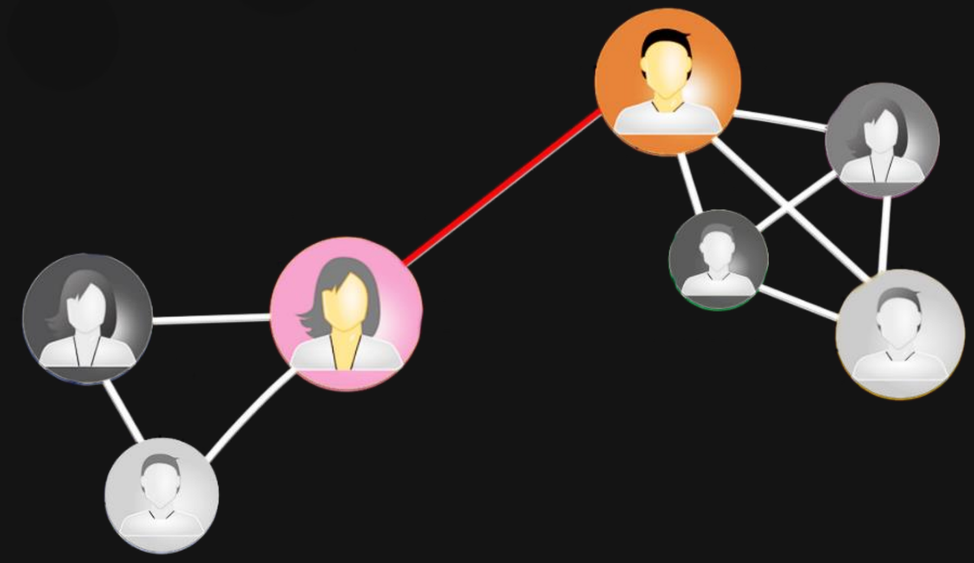
Step 2: Calculating the Weights



Unary Frequency Significance

The more often a programmer appears in the log, the more significant he is.

Step 2: Calculating the Weights



Betweenness Centrality

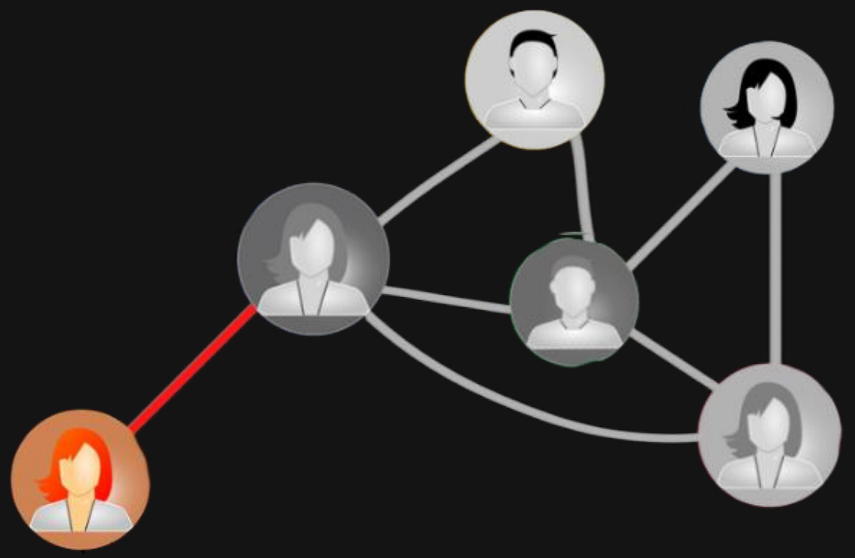
Handle programmers that are a part of several different teams.

Step 2: Calculating the Weights

Eigenvector Centrality

A node is highly important
if many other
highly important nodes link to it.

Step 2: Calculating the Weights

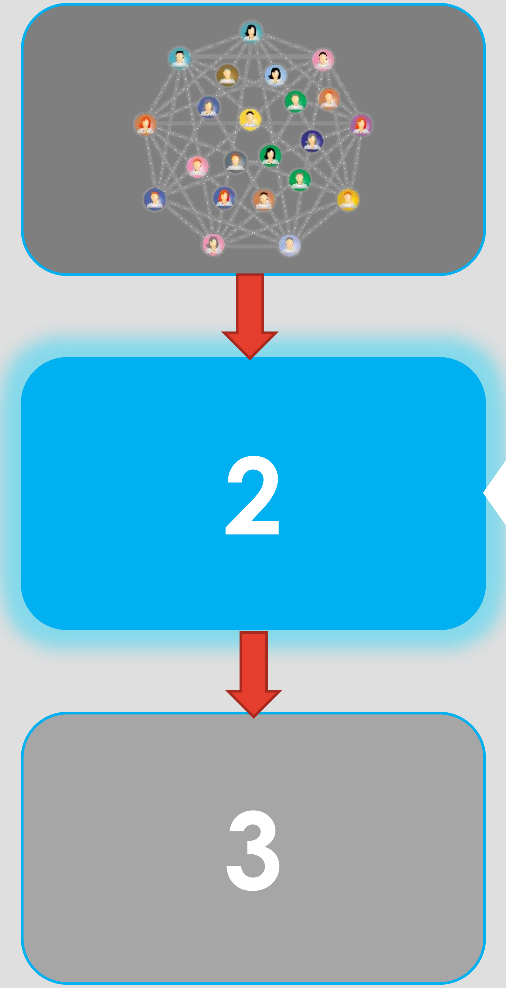


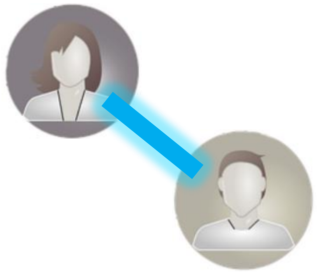
Degree Centrality

Number of edges incident upon the node.

→ *Identify isolated nodes*

Step 2: Calculating the Weights

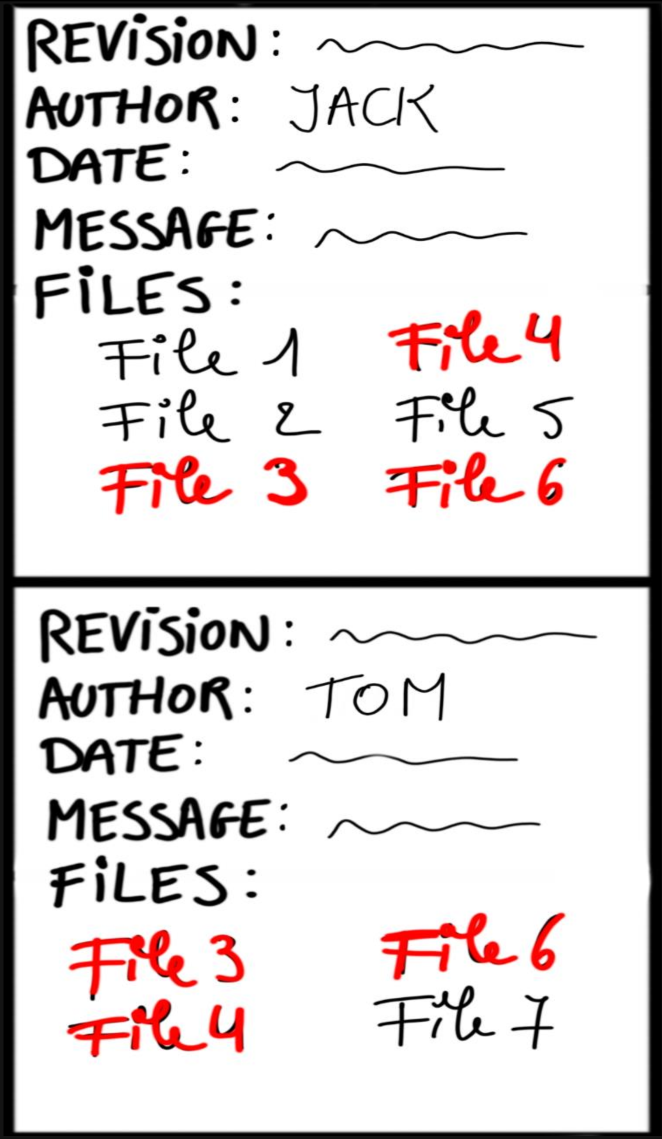




Importance of Collaboration

$$\sum_{j=1}^{n} w_j \begin{cases} \bullet \quad \text{Binary Frequency Significance} \\ \bullet \quad \text{Proximity Correlation} \end{cases}$$

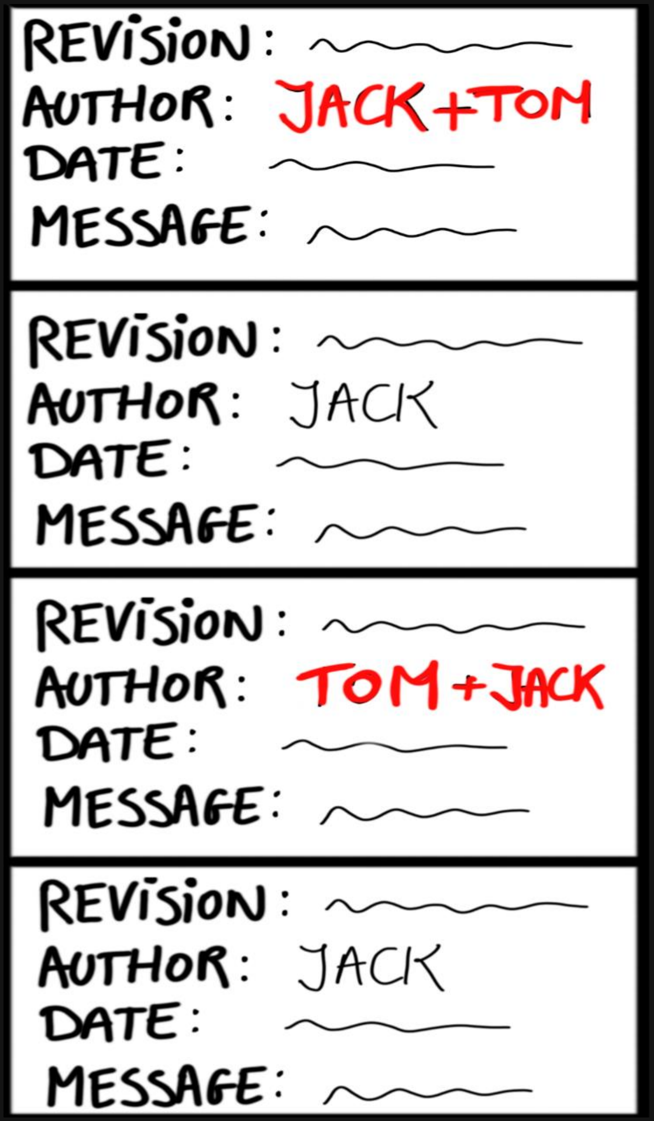
Step 2: Calculating the Weights



Binary Frequency Significance

The more files are worked on together, the stronger the relationship.

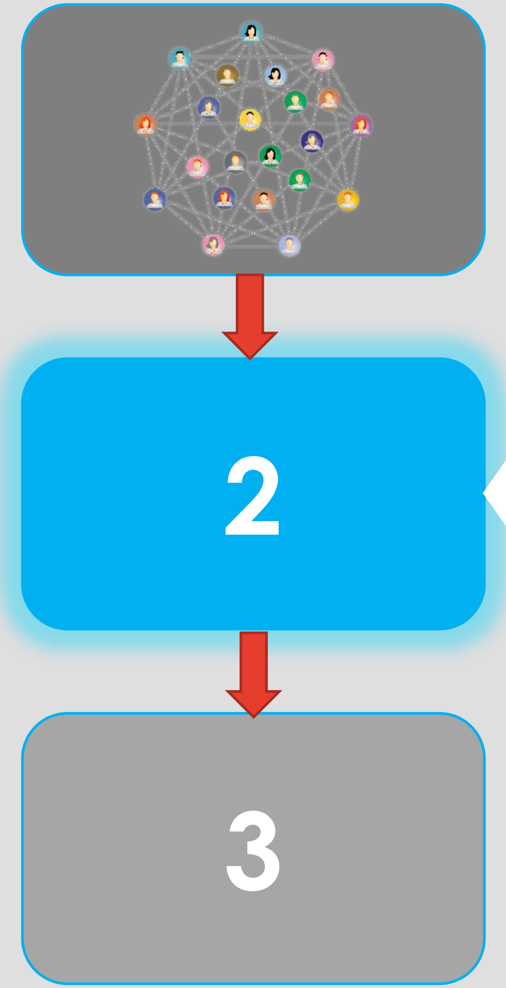
Step 2: Calculating the Weights

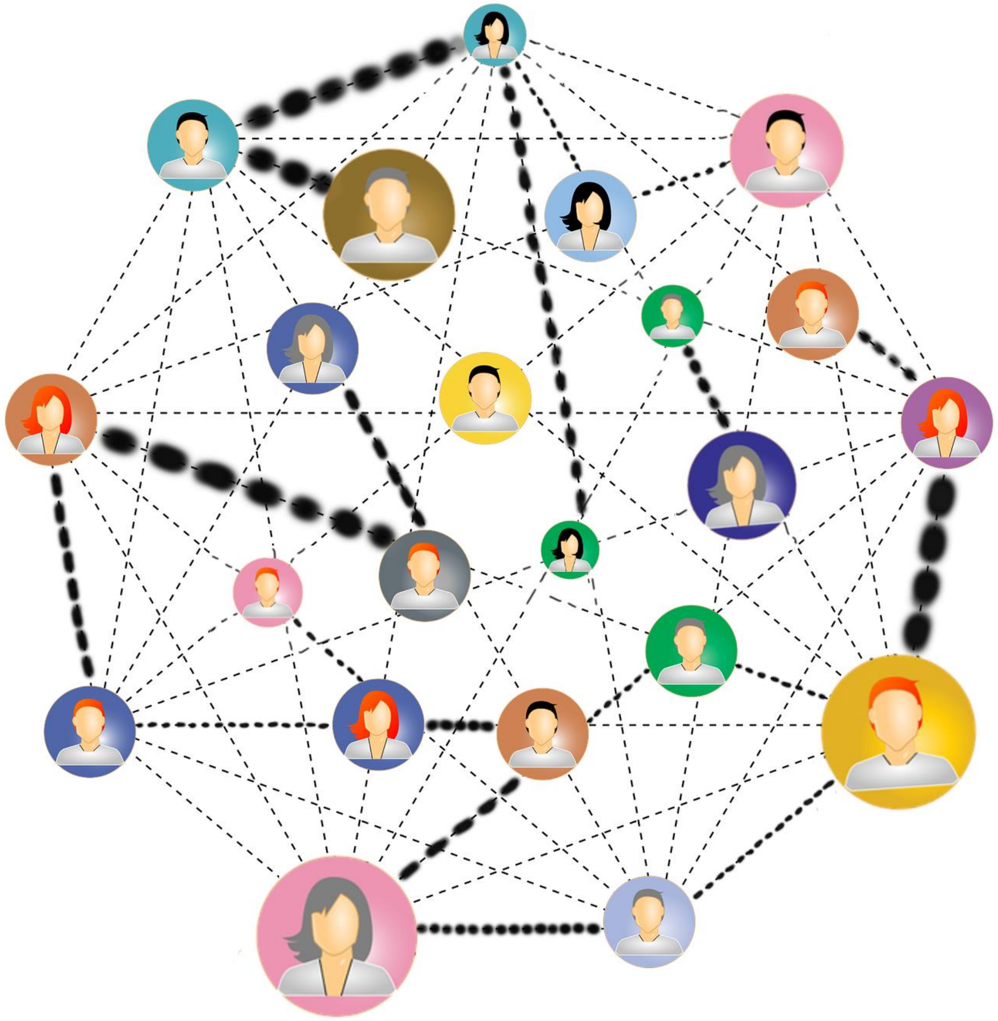


Proximity Correlation

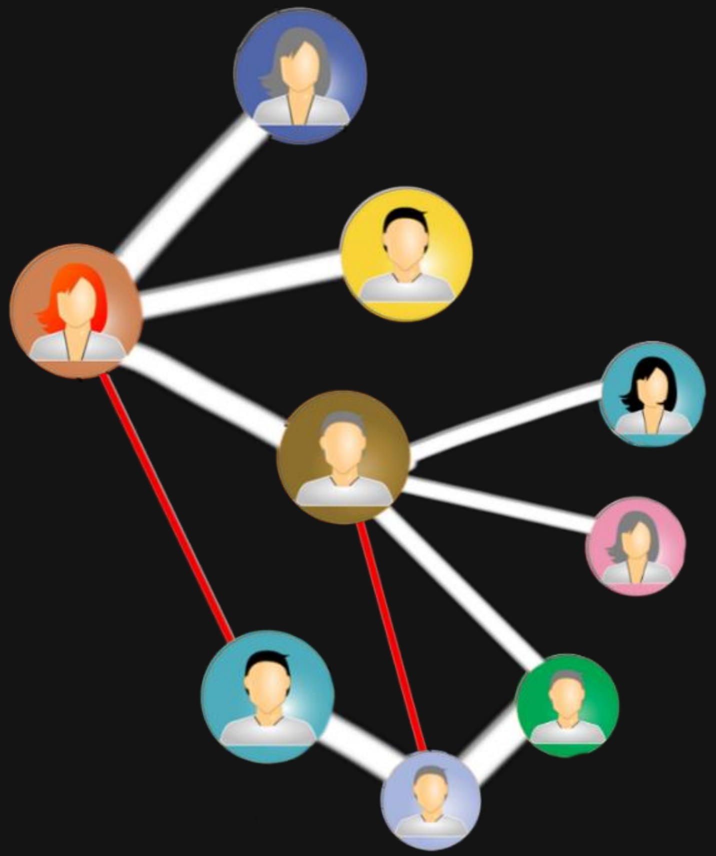
Pair programming
as a closer collaboration

Step 2: Calculating the Weights





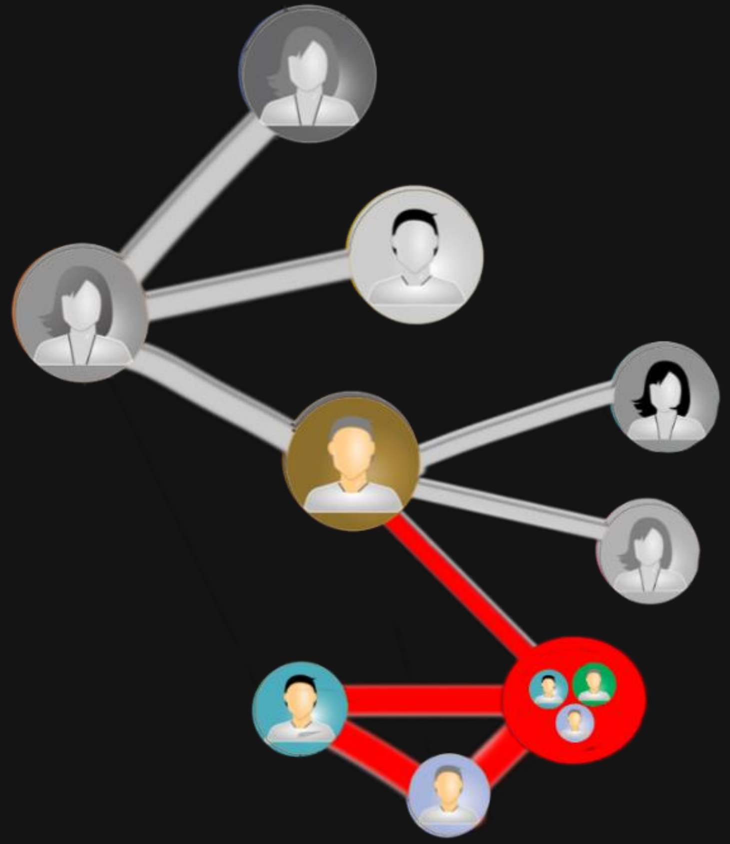
Step 3: Simplifying the Graph



Edge Filtering

Only the edges with the highest utility values are preserved

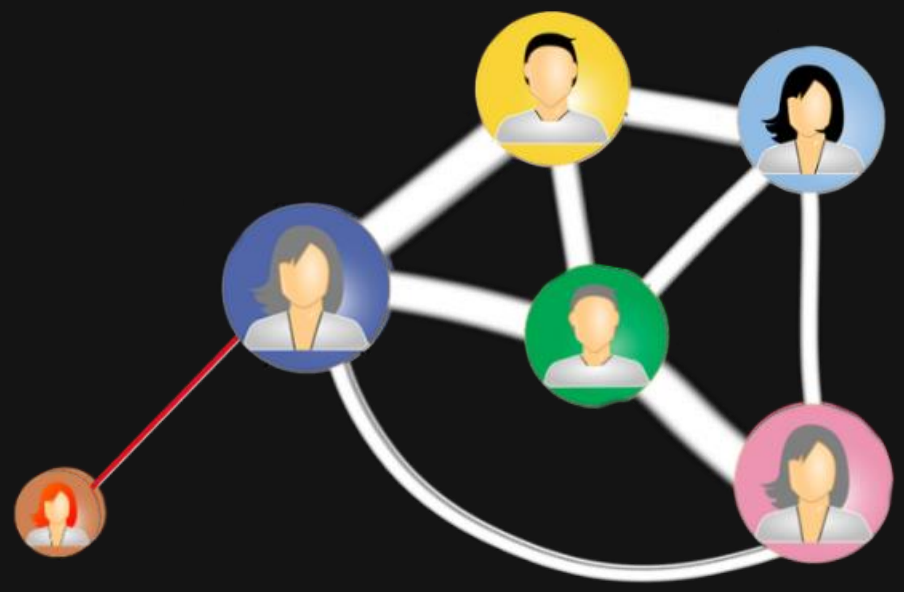
Step 3: Simplifying the Graph



Aggregation

Cluster less important but strongly connected programmers

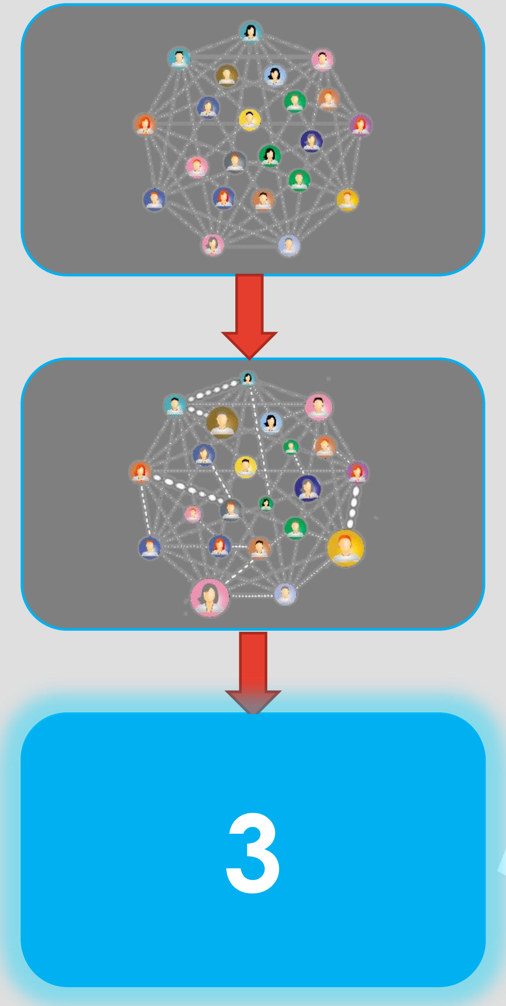
Step 3: Simplifying the Graph

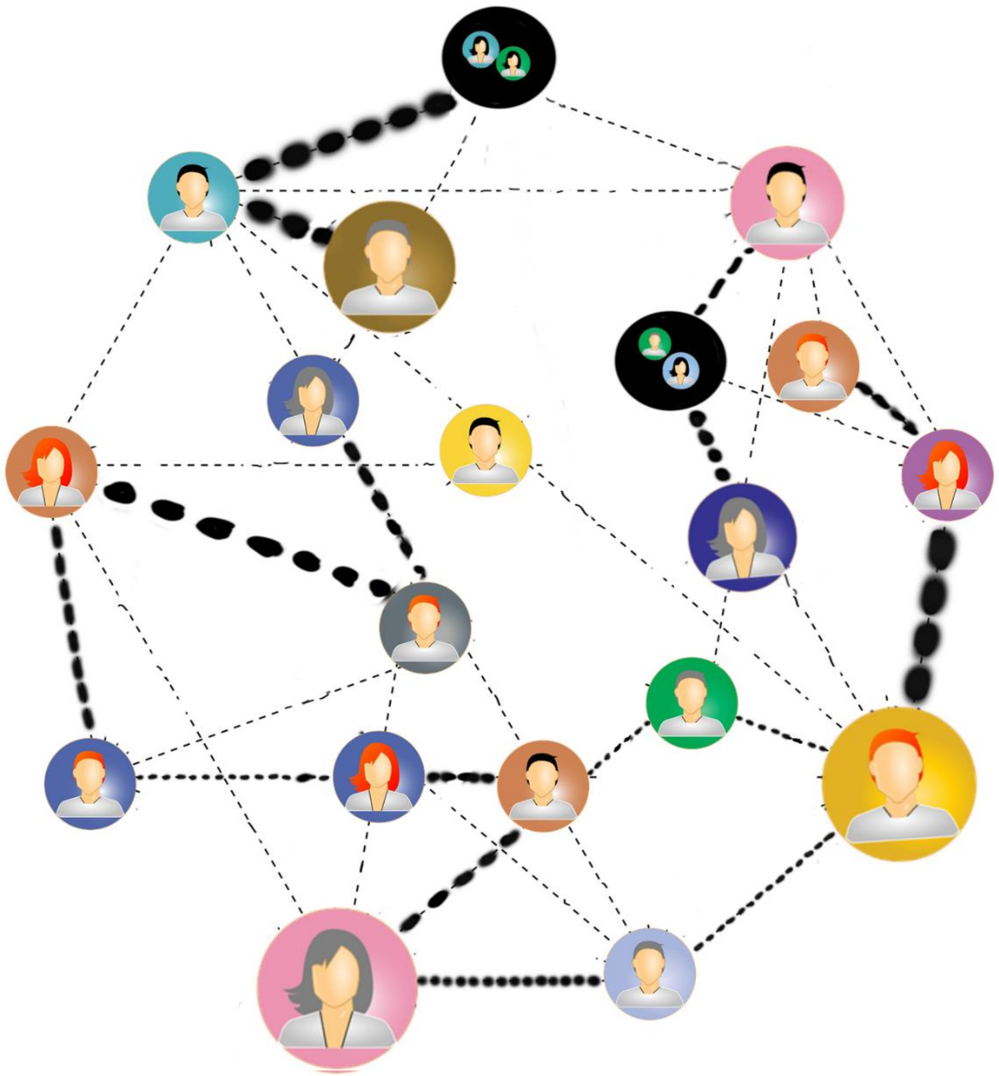


Abstraction

Abstract insignificant programmers that are weakly connected to the graph

Step 3: Simplifying the Graph

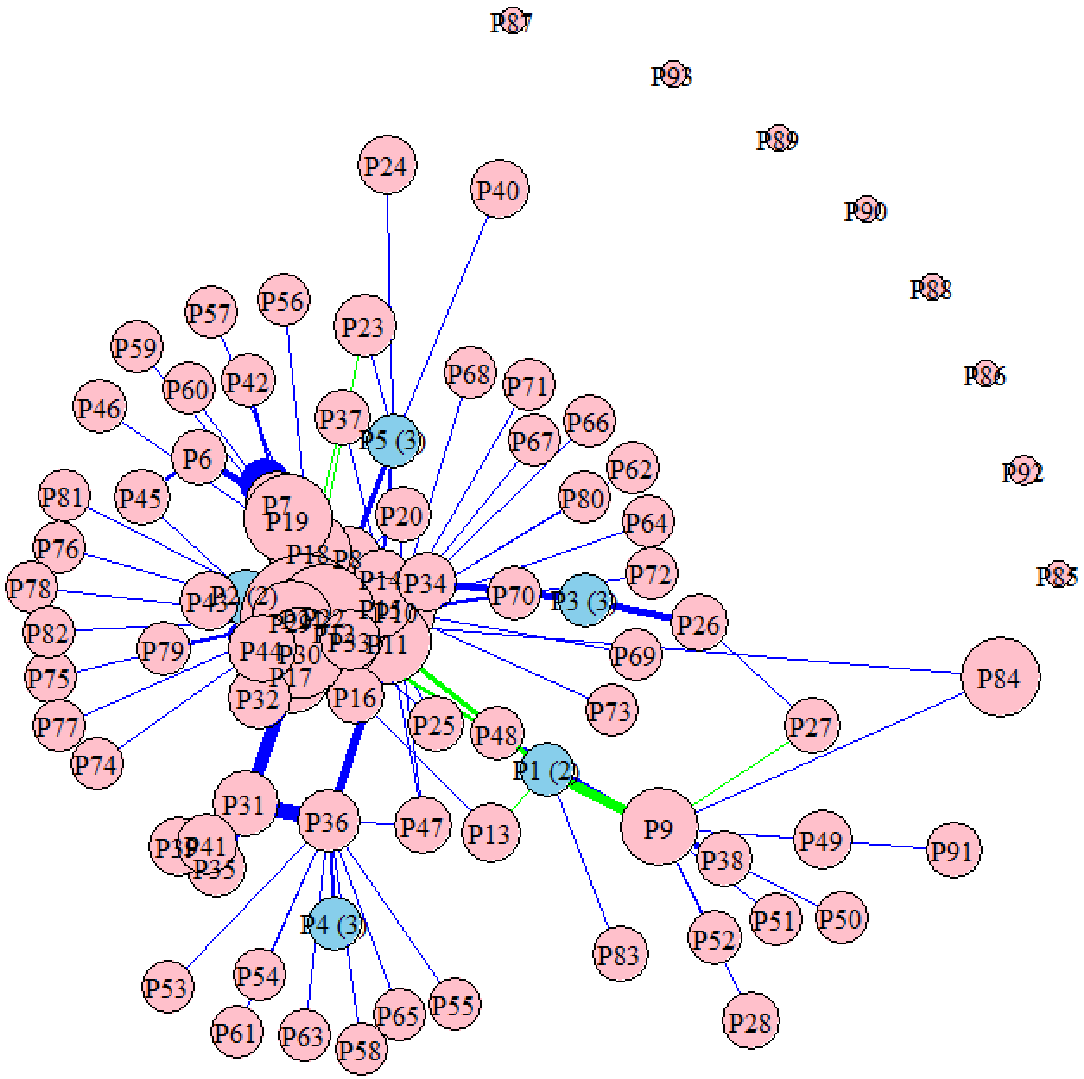




Results

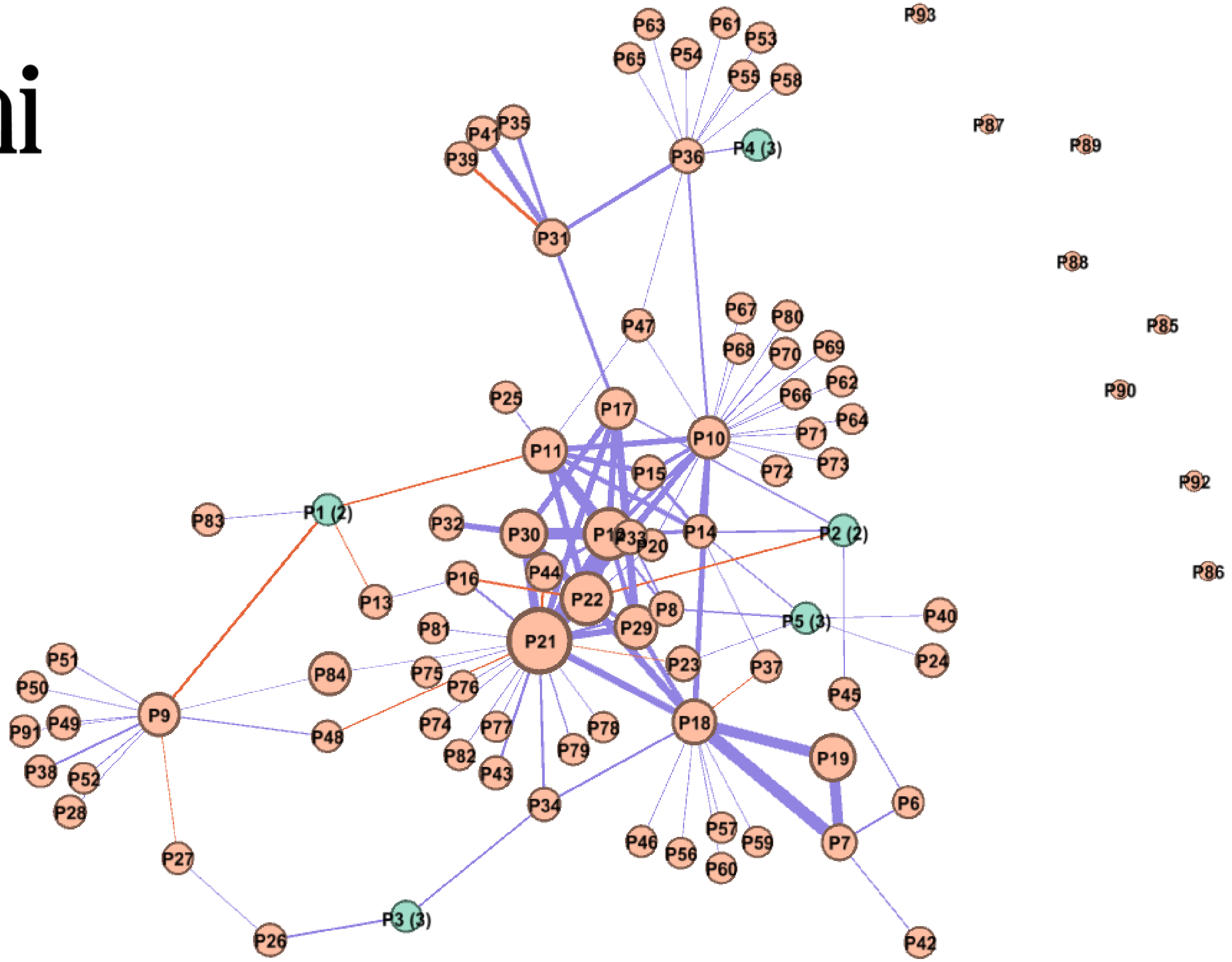
Results

```
g <- build_graph(log)
visualizeGraph(g,anonymize = TRUE)



#direct access
g$nodes
g$edges

#create an igraph object
igraph <- createIGraphObject(g)

#write graph to csv
writeGraphToCSV(g)
```



Results

Gephi



Results

```
g <- build_graph(log, AgVP = 0.25, AgCP = 0.15, AbVP = 0.7, AbCP = 0.3)
```

Results

```
g <- build_graph(log, AgVP = 0.25, AgCP = 0.15, AbVP = 0.7, AbCP = 0.3)
```

Clustering

- Control number of candidates
- Control strictness cluster condition

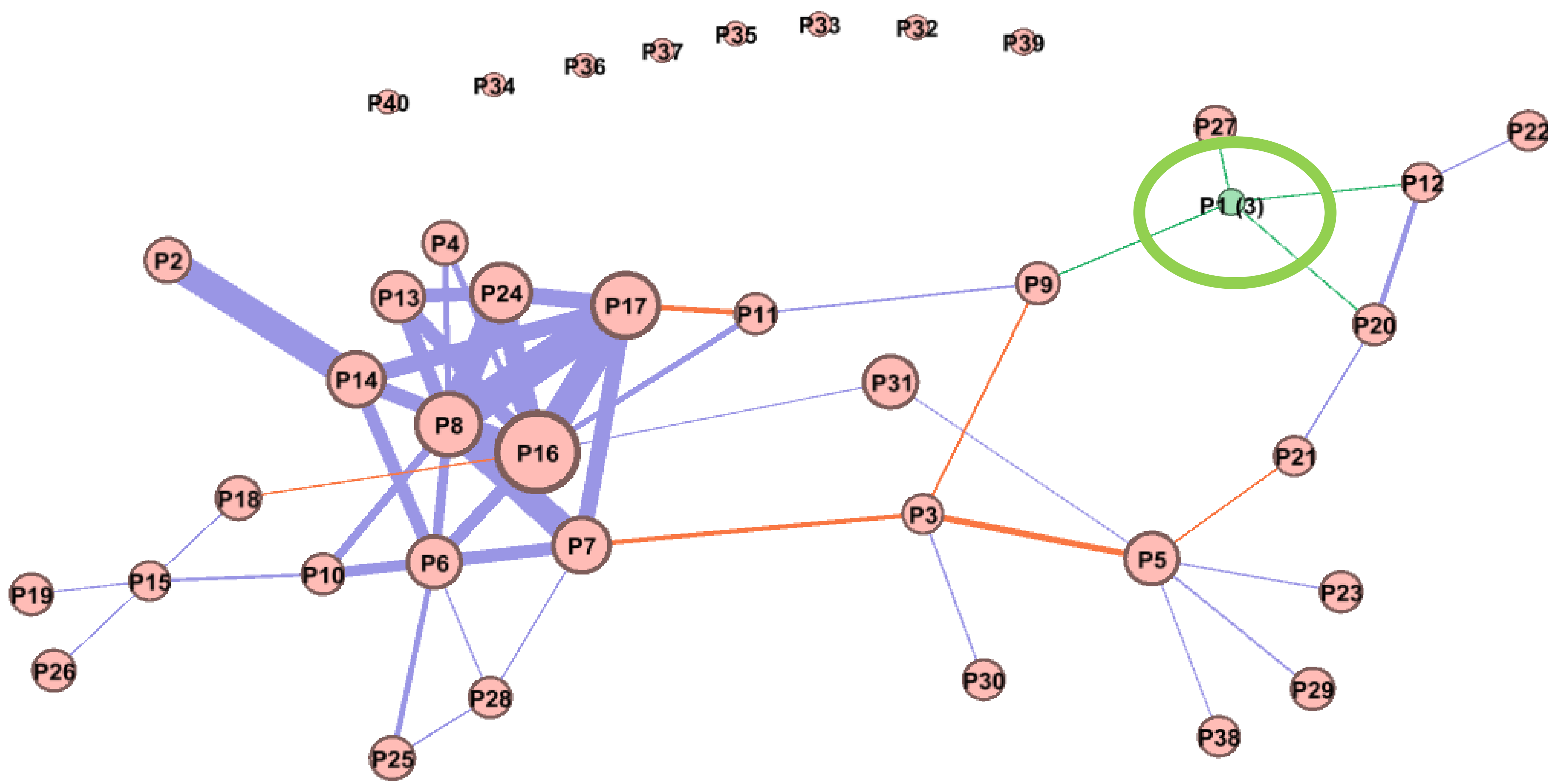
Results

```
g <- build_graph(log, AgVP = 0.25, AgCP = 0.15, AbVP = 0.7, AbCP = 0.3)
```

Abstraction

- Control number of candidates
- Control strictness abstraction condition

Results



P40
P34
P36
P37
P35
P33
P32
P39
P27
P22
P12
P1 (3)
P2
P4
P13
P24
P17
P11
P9
P20
P14
P31
P8
P16
P21
P18
P3
P7
P5
P19
P15
P10
P6
P23
P26
P28
P30
P29
P38
P25

Results

```
g <- build_graph(log, AgVP = 0.45, AgCP = 0.25, AbVP = 0.1, AbCP = 0.4)
```

Results

```
g <- build_graph(log, AgVP = 0.45, AgCP = 0.25, AbVP = 0.1, AbCP = 0.4)
```

Clustering

- More candidates
- More strict cluster condition

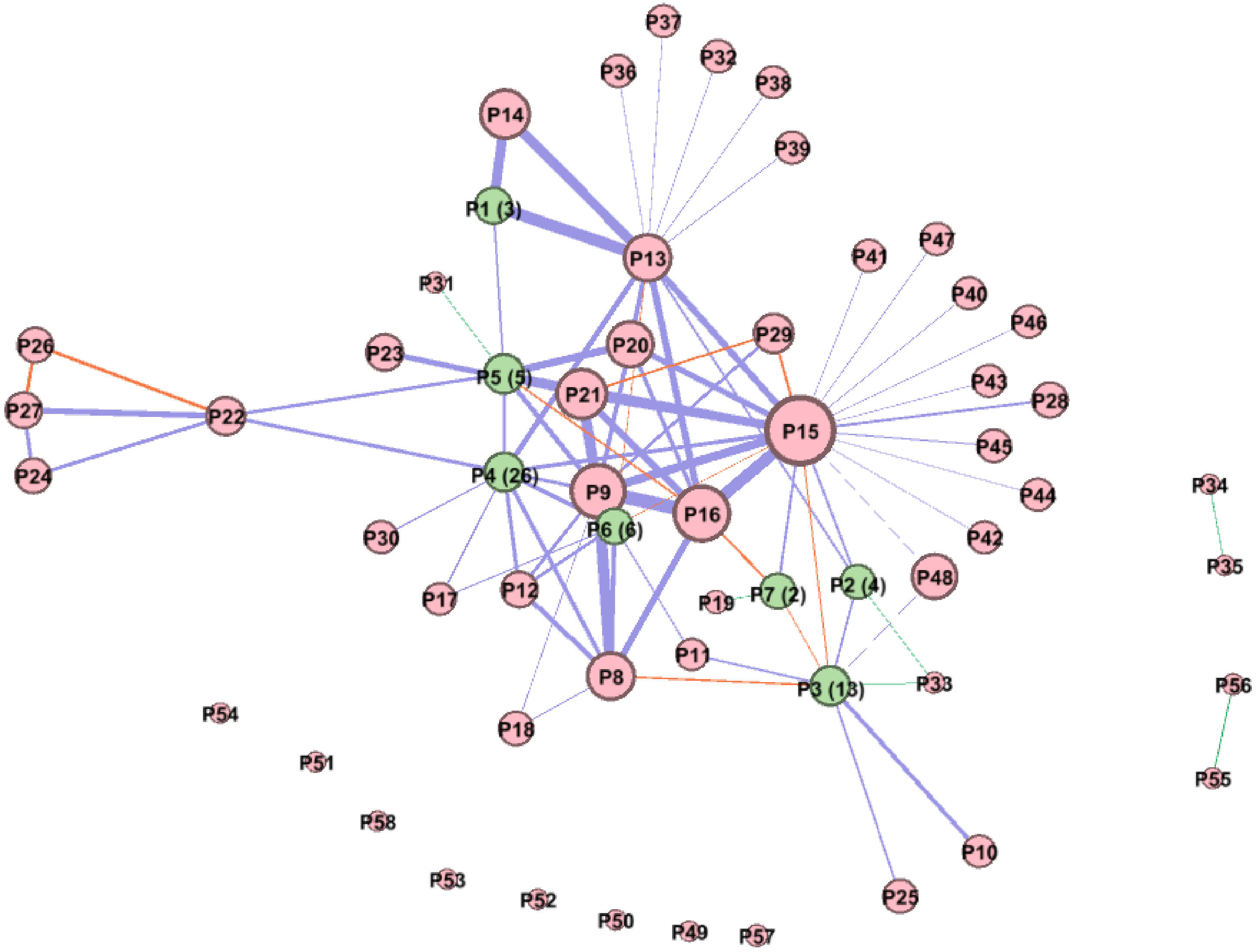
Results

```
g <- build_graph(log, AgVP = 0.45, AgCP = 0.25, AbVP = 0.1, AbCP = 0.4)
```

Abstraction

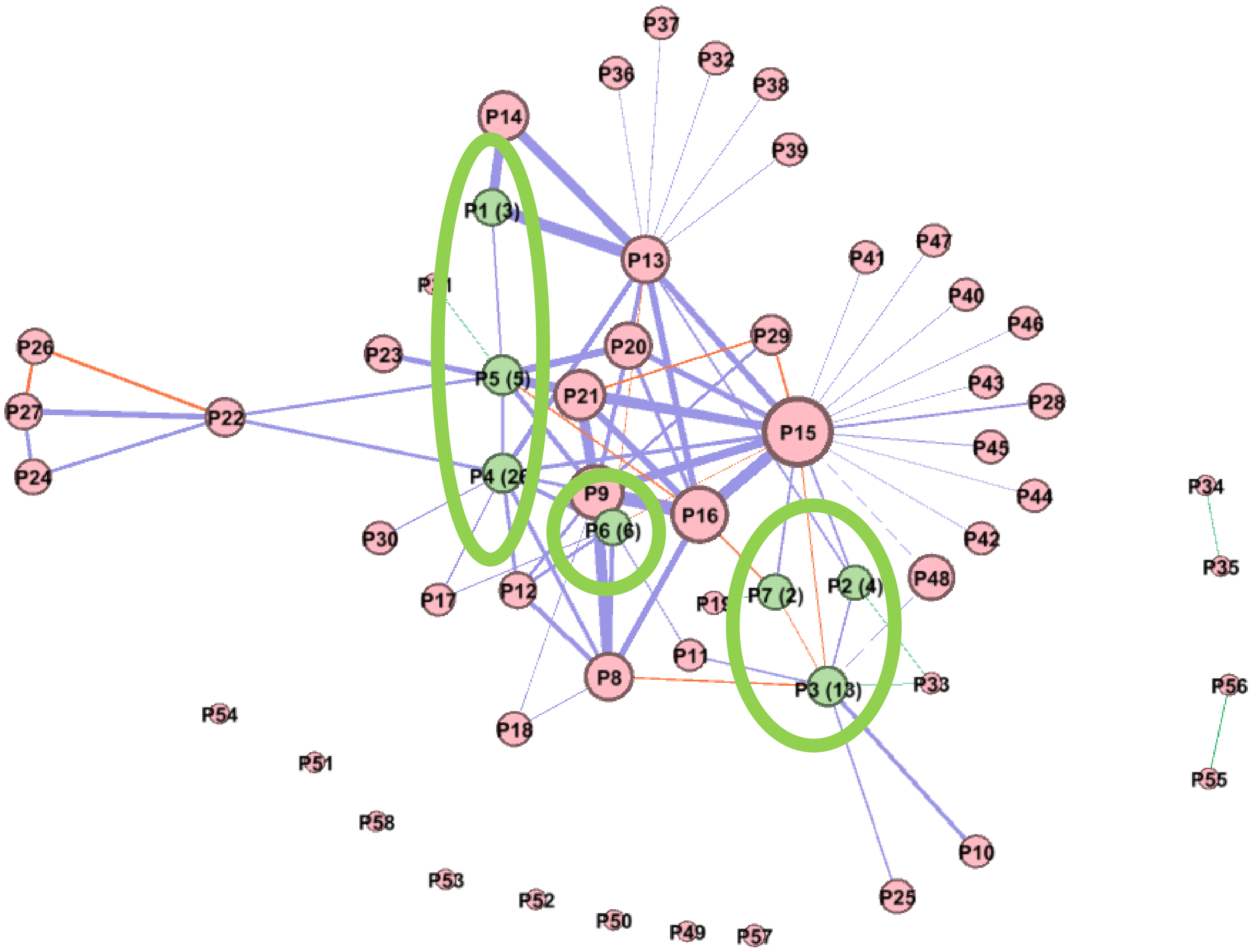
- Less candidates
- Less strict abstraction condition

Results



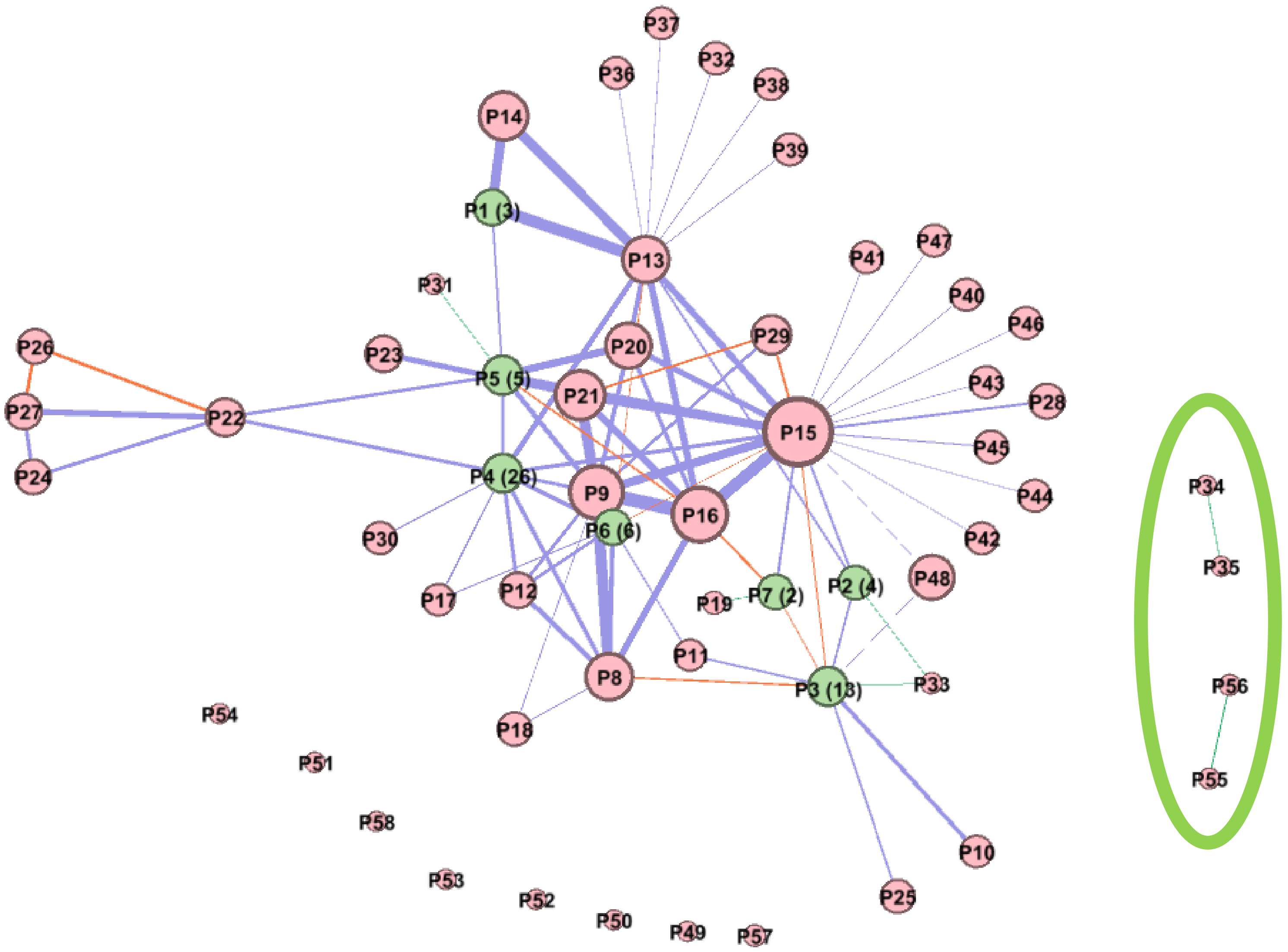
P37
P36
P32
P38
P14
P39
P1 (3)
P13
P41
P47
P31
P40
P46
P26
P23
P20
P29
P5 (5)
P21
P43
P28
P27
P22
P15
P45
P4 (26)
P24
P9
P16
P44
P34
P30
P6 (6)
P42
P48
P2 (4)
P35
P17
P12
P19
P7 (2)
P11
P8
P3 (13)
P33
P56
P54
P18
P51
P55
P58
P10
P53
P52
P25
P50
P49
P57

Results



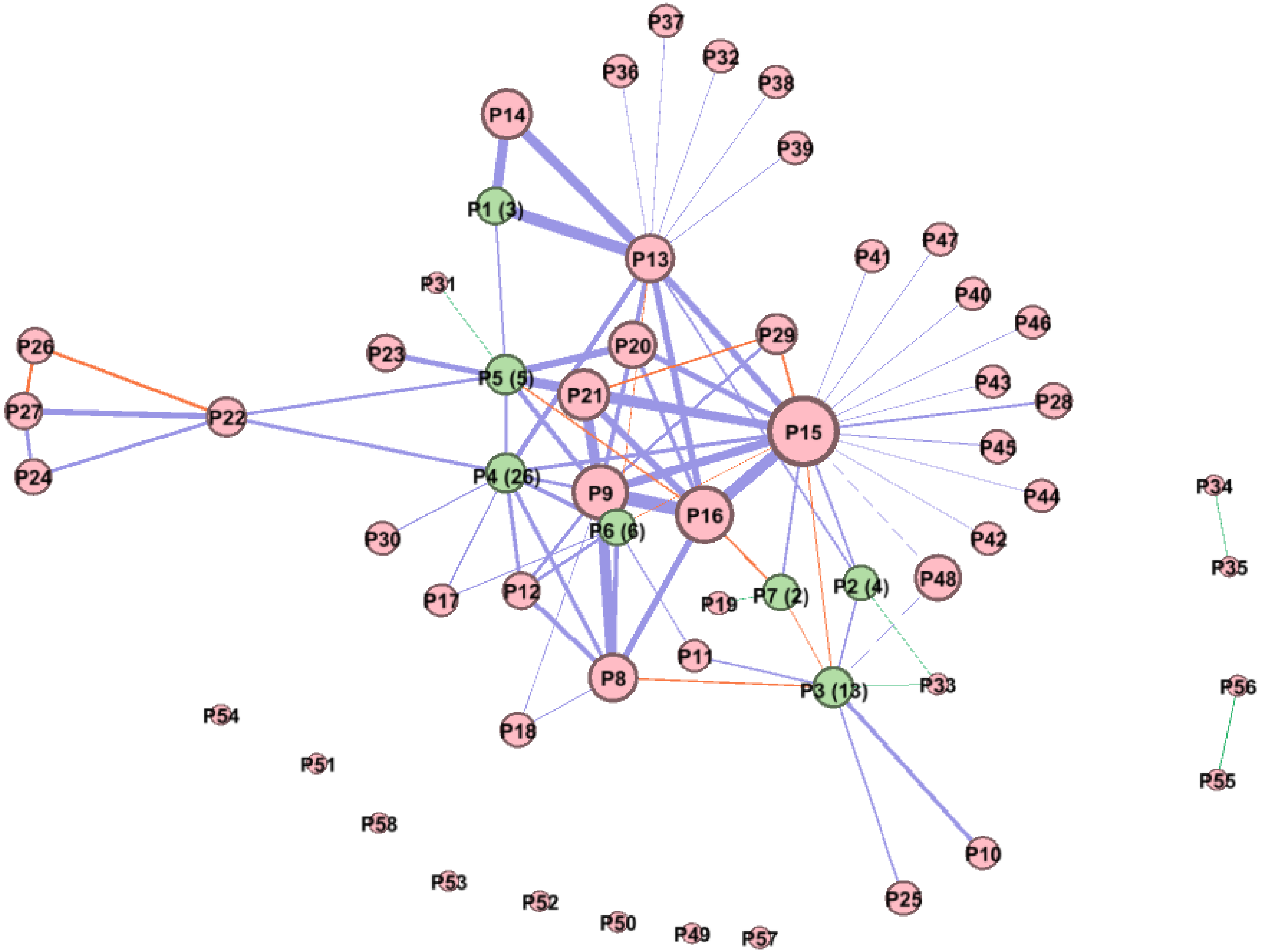
P37
P36
P32
P38
P14
P39
P1 (3)
P13
P41
P47
P40
P46
P26
P23
P20
P29
P5 (5)
P21
P43
P28
P27
P22
P15
P45
P24
P44
P34
P9
P16
P6 (6)
P30
P42
P35
P48
P2 (4)
P7 (2)
P17
P12
P11
P8
P3 (13)
P33
P56
P54
P18
P51
P55
P58
P10
P53
P25
P52
P50
P49
P57

Results

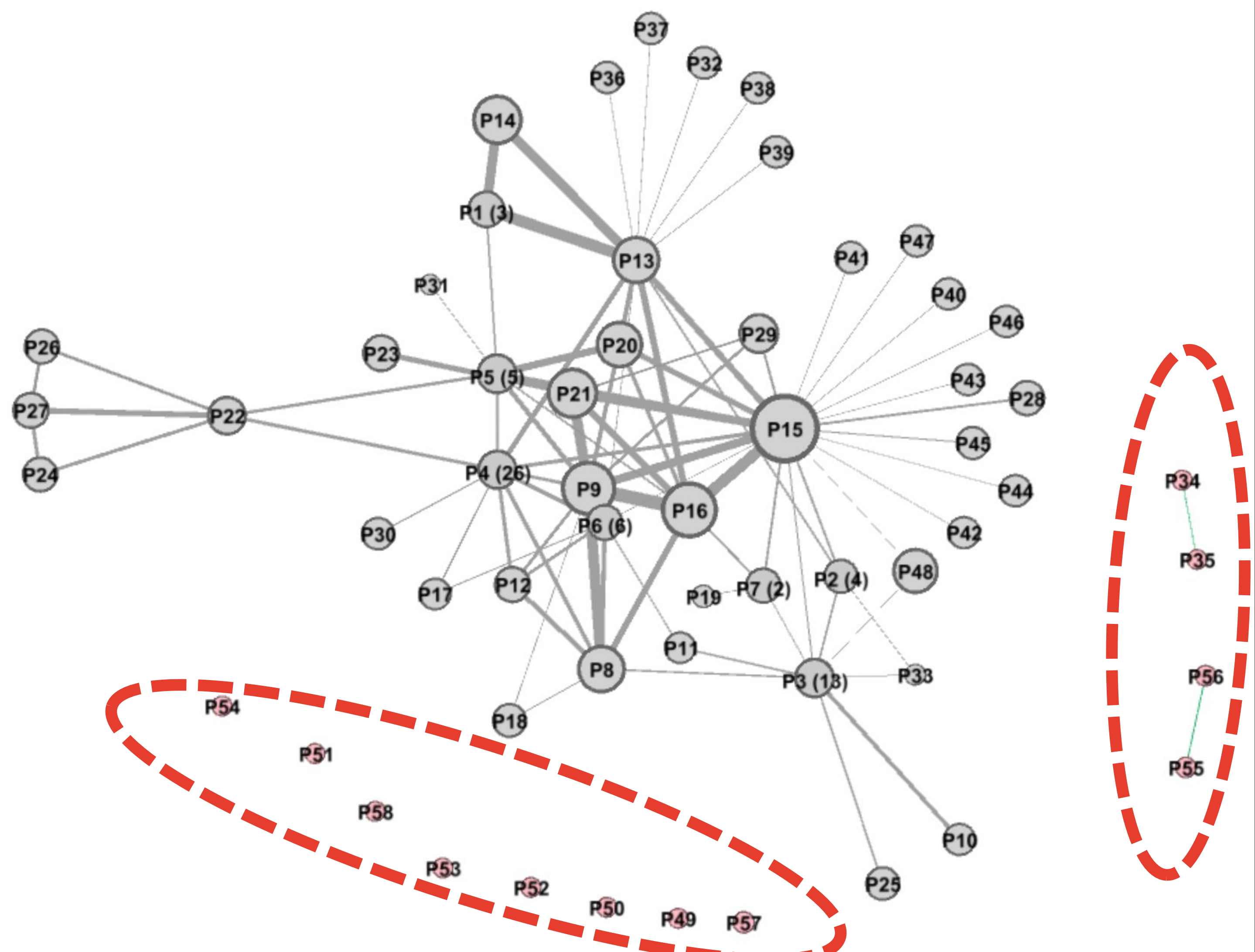

P37
P36
P32
P38
P14
P39
P1 (3)
P13
P41
P47
P31
P40
P46
P26
P23
P20
P29
P5 (5)
P21
P43
P28
P27
P22
P15
P45
P24
P4 (26)
P9
P16
P44
P34
P30
P6 (6)
P42
P35
P17
P12
P19
P7 (2)
P2 (4)
P48
P11
P8
P3 (13)
P33
P56
P54
P18
P51
P55
P58
P10
P53
P52
P25
P50
P49
P57

Insights

Insights


P37
P32
P36
P38
P14
P39
P1 (3)
P13
P41
P47
P31
P40
P46
P29
P26
P20
P23
P5 (5)
P43
P28
P21
P27
P22
P15
P45
P4 (26)
P44
P34
P24
P9
P16
P6 (6)
P42
P30
P48
P35
P17
P12
P19
P7 (2)
P2 (4)
P11
P8
P3 (13)
P33
P56
P54
P18
P51
P55
P58
P10
P53
P52
P25
P50
P49
P57

Insights

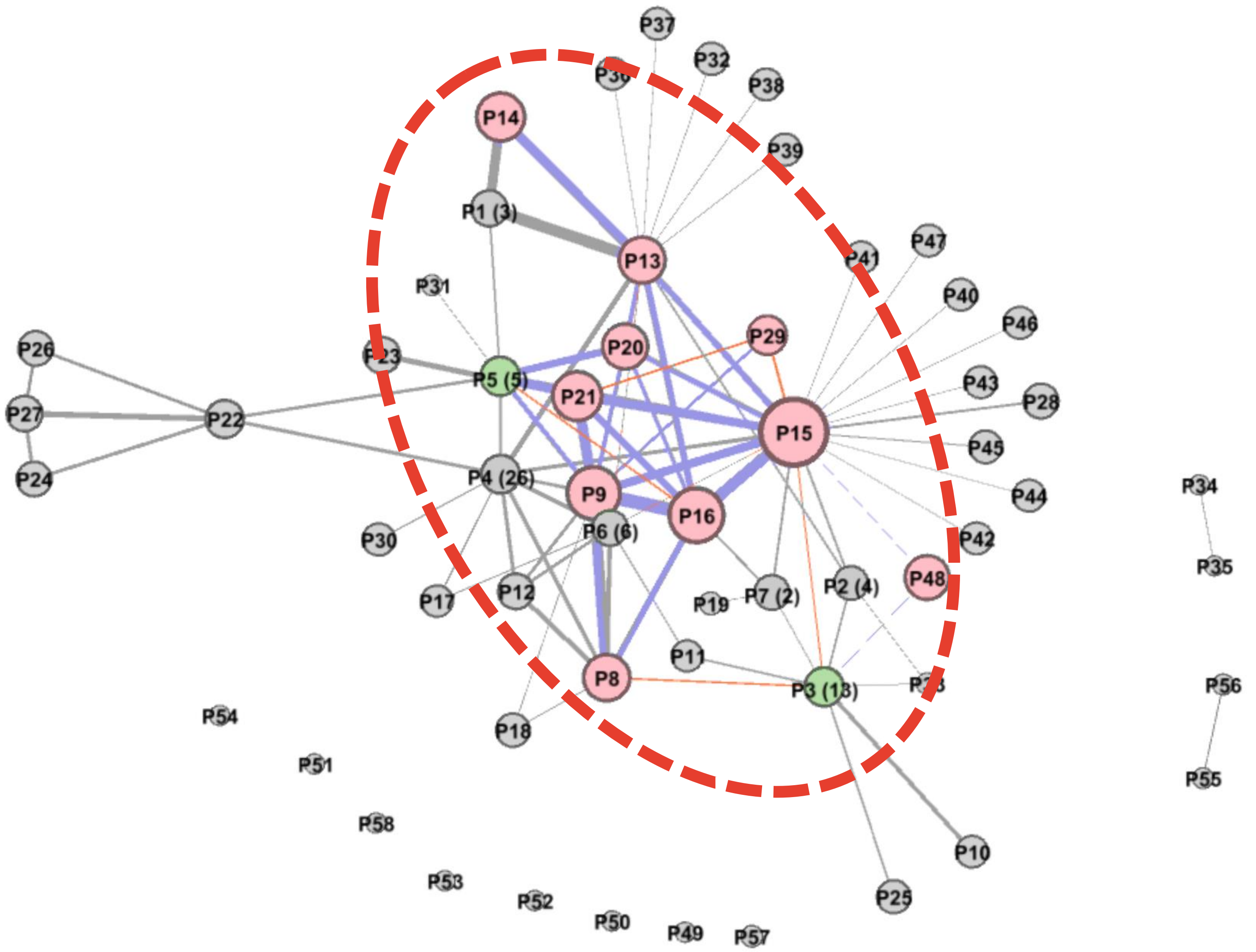


Isolated groups

Risky if:

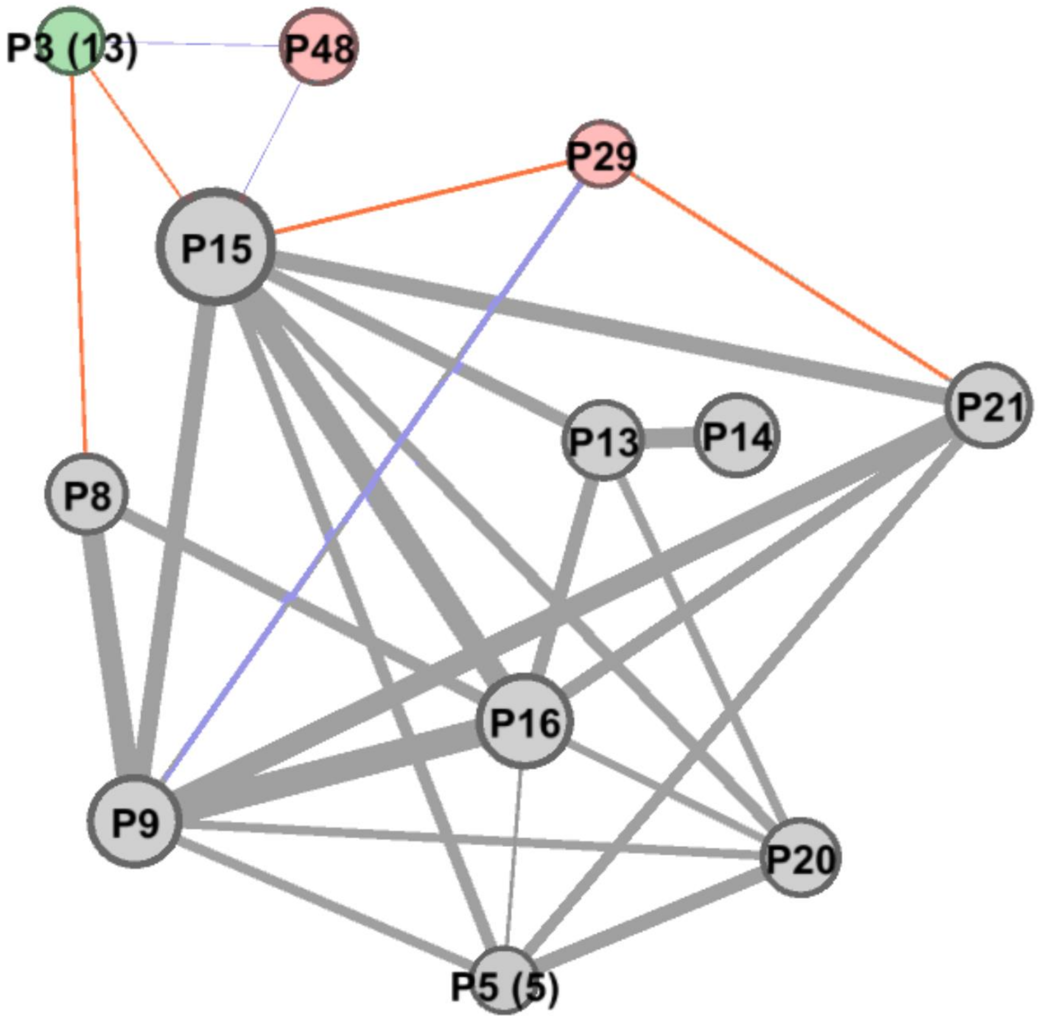
- Few members
- Members have large importance

Insights



Core developers

Insights



Core developers

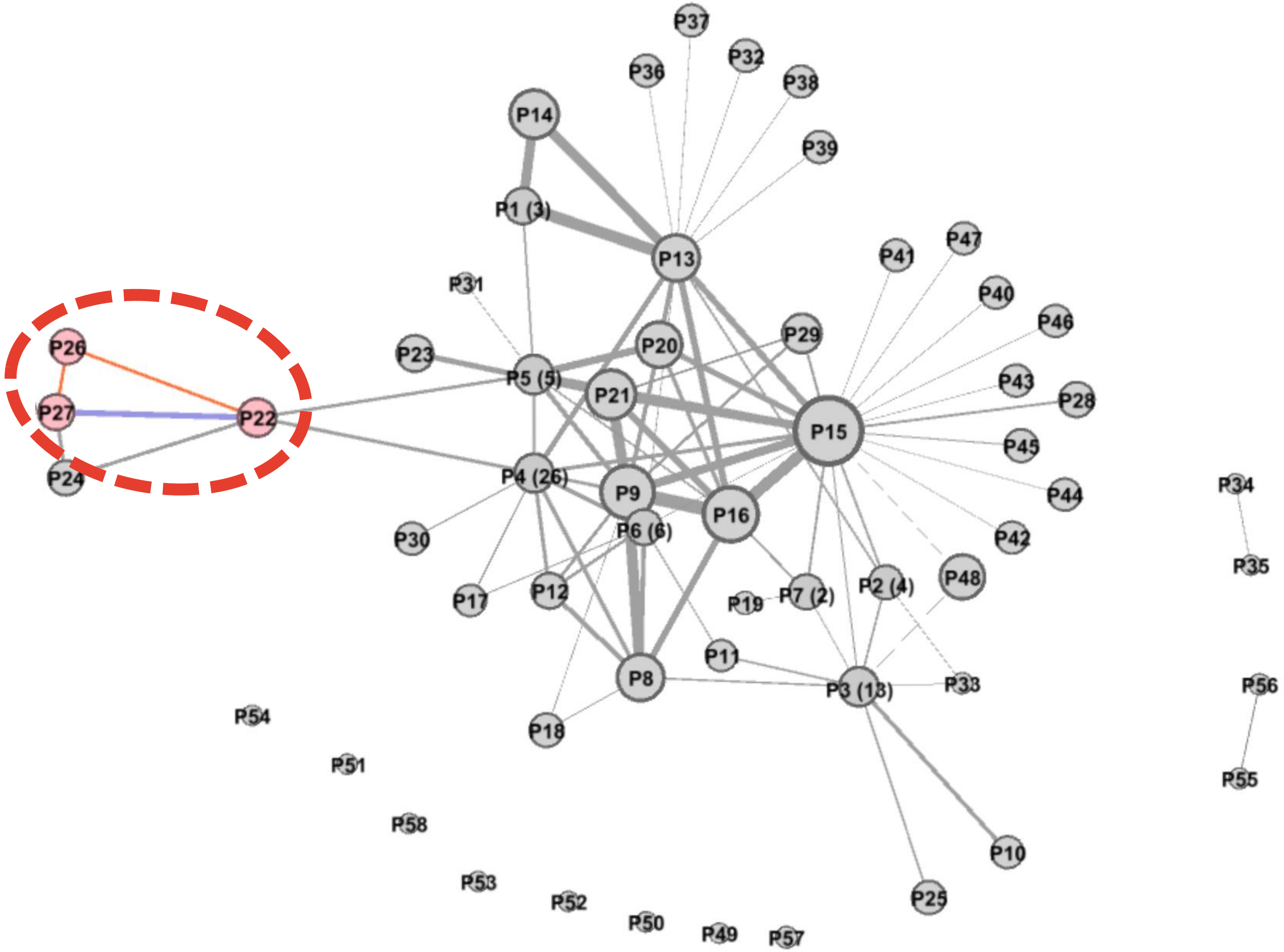
Risky because:

Important contribution

+

Weak collaboration

Insights

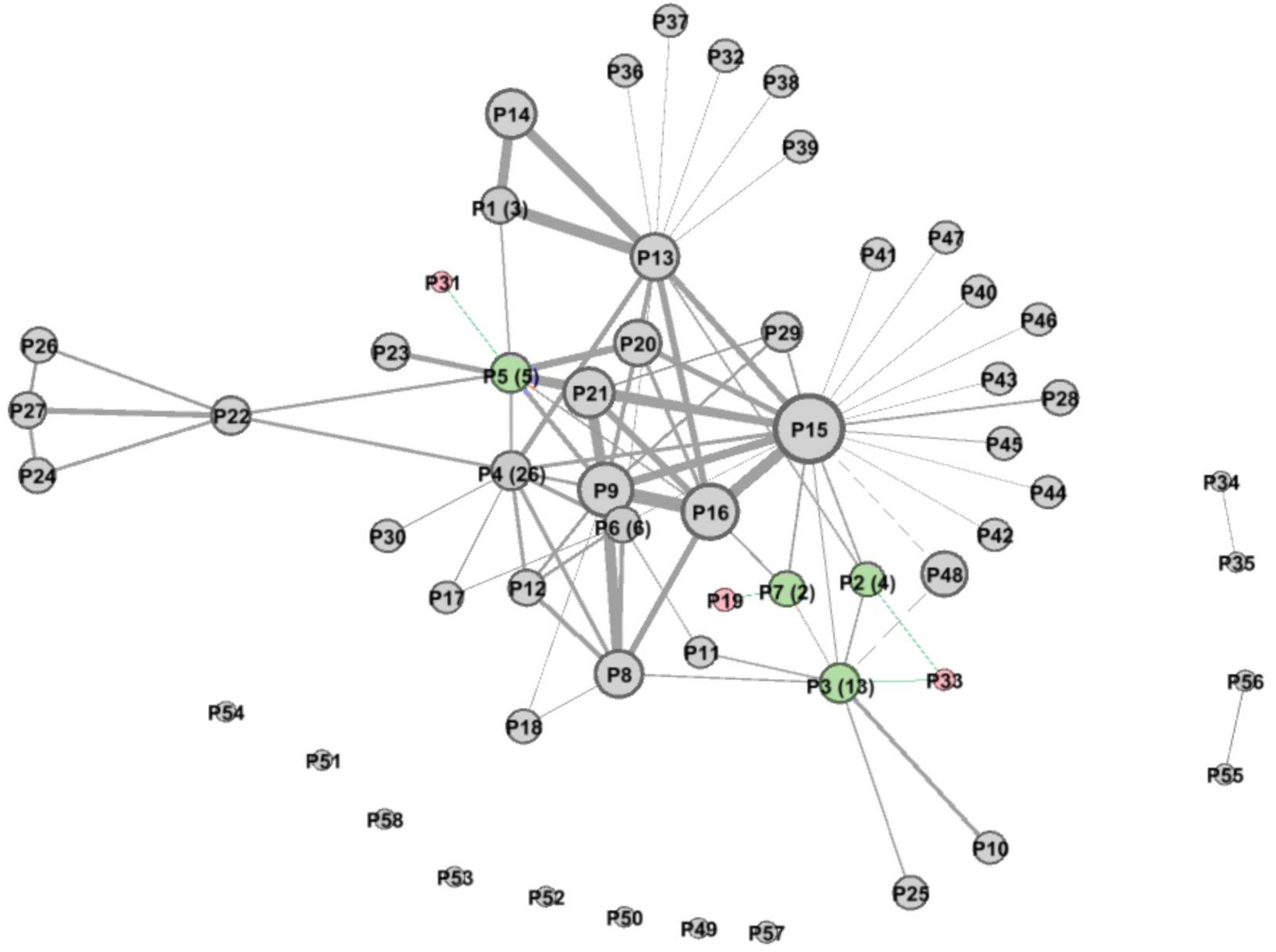


Striking relations

Strong disjunct programming relation:

The only ones working on a specific aspect of the code

Insights



Striking relations

Pair programming relation:

Teacher

+

Apprentice

Thank you

leen.jooken@uhasselt.be

https://github.com/bupaverse/collaborateR