# Team Assignment Project

## SCMA 645 Management Science

## Virginia Commonwealth University

## Spring 2021

By

**Allen Philip Agnelo**

**Bhumika Modi**

## Executive Summary-

Federal Reserve bank of Richmond reorganizes its Informational Technology functional group into small multi-functional teams based on various criteria. These multi-functional agile teams would be able to take products from start to finish without the time lost due to transitioning projects between the current functional groups. As students at Virginia Commonwealth University School of Business we had the opportunity to build this model under the guidance of Prof. Paul Brooks, we got the opportunity to build this model as a class project which involved asking questions to the representative from the bank and developing a prototype model which was further developed into a model which accommodates the requirements and information provided by the client.

The bank has about 143 employees that need to be assigned to 14 teams. The ideal size of each team would be between 8 to 12. Agile project management philosophy is adopted. Each team will have a scrum master who will support two teams. A part-time or full-time agile coach will be hired if the team lacks experience working in an agile environment. Time zones will have to be considered when allocating employees to their teams as they are in different location time zones and need to facilitate scheduling while operating virtually.

Based on their product area expertise, employees are classified as “developers”, “quality assurance testers”, “user acceptance testers”, “networking professionals”, and “security experts” and each new team will have a worker from every product area.

We used Assignment approach to try and develop a model for employees who are assigned to different teams based on the criteria stated by the clients. Pyomo and Gurobi solver was used to carry out the optimization process.

The model was developed for 143 and was distributed to over 13 teams. The results involved proper distribution of the workers to different teams based on size and other constraints mentioned and resulted in a total minimum cost of $429 for the assignment made.

The bank can use this same model for the next six months as well as to reduce the total cost incurred. This model also assigns employees from different time zones location to be in teams which have similar time zones so that they can meet and operate virtually.

**Problem Description:**

The Federal Reserve Bank of Richmond is looking to organize its information technology functional groups into smaller multi-function teams based on various criteria. By creating these multi-functions agile teams, we will be able to take products from start to finish without the time lost due to transitioning projects between the current functional groups.

## Introduction

The Federal Reserve System (the Fed) is the central banking system of the United States of America. The Federal Reserve performs five general functions which are conducting the nation's monetary policy, regulating banking institutions, monitoring and protecting the credit rights of consumers, maintaining the stability of the financial system, and providing financial services to the U.S. government.

The Federal Reserve Bank of Richmond is the headquarters of the Fifth District of the Federal Reserve located in Richmond, Virginia. It covers the District of Columbia, Maryland, North Carolina, South Carolina, Virginia, and most of West Virginia.

Federal Reserve Information Technology (FRIT) is a national IT service provider within the Bank that provides operational and project services, enterprise IT architecture and standards services, and enterprise information security policy and assurance services throughout the Federal Reserve System.

**Objectives in words -**

To assign workers into small multi-functional agile teams, considering their functional expertise area and other constraints in order to minimize the total cost.

**Subject to constraints:**

- Allocating workers to a team size of 8 to 12 members.
- Each team must have a worker from different expertise area.
- Each team must have a product owner from each function
- At least one member should be added in a team from different work functions (plan, build, run).
- Each team can have a member who is highly certified and experienced.

- Each team should have at least 30% of highly certified and experienced, training/experience members with them.
- Scrum master can support two teams.
- Workers from similar time zones are to be added to a team in a way they can facilitate scheduling.

**Data**

Let W = [1,2,3,4……142,143] Person ID

T = [1, 2, 3……20] be the set of teams

S = [ Developer/Engineer, Scrum Master, Tester, Dev/QA Manager (DevOps), Application Architect, Solution Architect, Release Management, Analyst, Product Owner, Architect, Communication Specialists, Governance, Operations Specialist] be the skillset of employees at the FED.

E = [advanced, associate, intermediate, senior] be set of experience levels.

F = [Plan, build, run, scrum] be set of work functions

Y = [0-5, 5-10, 10-20, 20+] be the set of years of experience

Z = [ET, CT, PT]

A = [basic, highly experienced, and certified, training/experience] be set of agile experience

**Decision Variables**

Let,

$$x_{ij} = \begin{cases} 1, if\ worker\ i\ is\ assigned\ to\ team\ j, i \in W, j \in T \\ 0, otherwise \end{cases}$$

$$y_{jk} = \begin{cases} 1, when\ team\ j\ is\ assigned\ to\ time\ zone\ k\ , j \in T\ and\ k \in Z \\ 0, otherwise \end{cases}$$

$$z_j = \begin{cases} 1, when\ team\ j\ is\ assigned, j \in T \\ 0, otherwise \end{cases}$$

$$e_{jk} = \begin{cases} 1, when\ team\ j\ is\ assigned\ to\ different\ expertise\ level\ k\ , j \in T\ and\ k \in E \\ 0, otherwise \end{cases}$$

$$f_{jk} = \begin{cases} 1, when\ team\ j\ is\ assigned\ to\ plan\backslash build\backslash run\ k\ , j \in T\ and\ k \in P \\ 0, otherwise \end{cases}$$

$$he_{j} = \begin{cases} 1, when\ team\ j\ is\ assigned\ to\ highly\ experienced\ and\ certified\ , j \in T \\ 0, otherwise \end{cases}$$

$$te_{j} = \begin{cases} 1, when\ team\ j\ is\ assigned\ to\ highly\ experienced\ and\ certified\ and\ trained\ or\ experienced\ \ , j \in T \\ 0, otherwise \end{cases}$$

**Algebraic Formulation –**

$$\begin{aligned} min\ agile\_weekly\_cost* \sum_{j \in T} Z_j + time\ \_zone\_\ cost \\ * \sum_{j \in T} \sum_{k \in Z} y_{jk} + team\_size\_cost * \sum_{j \in T} (tsplus_j + tsminus_j) \\ + functional\_\ exp\_cost* \sum_{j \in T} \sum_{k \in E} \left(feplus_{jk} + feminus_{jk}\right) \\ + product\_owner\_cost \sum_{i \in W} \sum_{j \in T} x_{ij} \\ + area\_cost \sum_{j \in T} \sum_{k \in P : K \neq "Scrum"} \left(fpplus_{jk} + fpminus_{jk}\right) \end{aligned}$$

$$(Total\ Cost\ objective)$$

**Subject to Constraint:**

$$\sum_{i \in W : S_i = "Scrum\ Master"} \sum_{j \in T} x_{ij} == 1 \qquad (Team\ Allocation)$$

$$\sum_{i \in W} \sum_{j \in T} x_{ij} \leq 10 + tsplus - tsminus \qquad (Team\ Size)$$

$$\sum_{j \in T} \sum_{K \in E} e_{jk} \leq 2.5 + feplus - feminus \qquad (Functional\ Expertise\ diversity)$$

$$\sum_{i \in W: P_i = "Y"} \sum_{j \in T} x_{ij} \geq 1 \qquad (Product\ owner)$$

$$\sum_{i \in W} \sum_{j \in T} x_{ij} == functiona\_target + fpplus_{jk} - fpminus_{jk} \ , k \in P: k \neq \text{Scrum"}$$

$$(Functional\ constraint)$$

$$\sum_{i \in W: A_i = \text{"highly experienced and certified"}} x_{ij} \geq he_j, j \in T$$

$$(Agile\ for\ highly\ experienced\ and\ certified)$$

$$\sum_{i \in W: A_i \in \{\text{"highly experienced and certified","training/experience"}\}} x_{ij} - 0.3 \sum_{i \in W} x_{ij} \geq -12\big(1 - te_j\big), j \in T.$$

$$(30\%\ of\ the\ team\ are\ at\ least\ \text{“}training/experience\text{”}\ in\ agile)$$

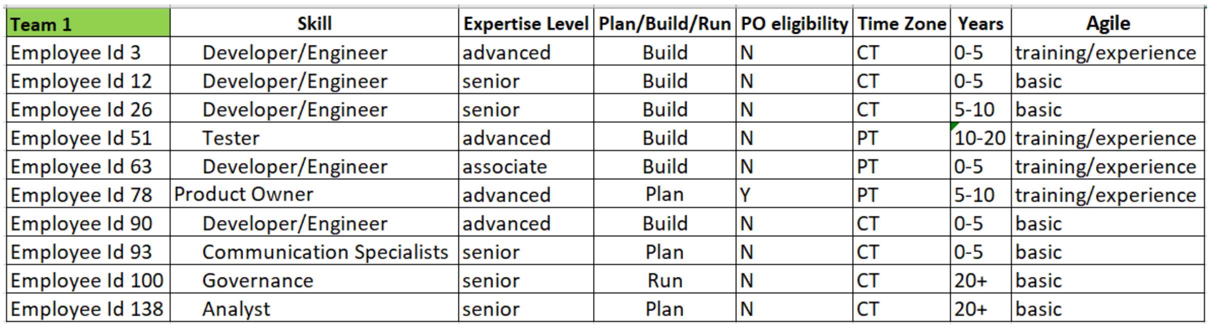
$$\sum_{i \in W: S_i = "Scrum\ Master"} \sum_{j \in T} x_{ij} \geq 2\ and\ x_{ij} \geq 1 \qquad (Scrum\ Master)$$

**Implementation:**

Refer to the codes attached in appendix at the end of the report for implementation using Pyomo and gurobi solver. Also, see attached Jupyter notebook file named Project Final.ipynb

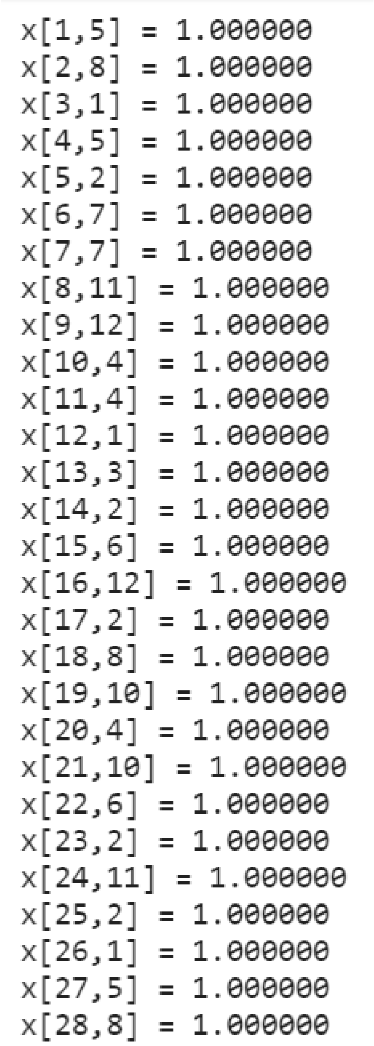
**Results –**

The assignment of employees to workspaces is shown in table below –



| Team 1 | Skill | Expertise Level | Plan/Build/Run | PO eligibility | Time Zone | Years | Agile |
|---|---|---|---|---|---|---|---|
| Employee Id 3 | Developer/Engineer | advanced | Build | N | CT | 0-5 | training/experience |
| Employee Id 12 | Developer/Engineer | senior | Build | N | CT | 0-5 | basic |
| Employee Id 26 | Developer/Engineer | senior | Build | N | CT | 5-10 | basic |
| Employee Id 51 | Tester | advanced | Build | N | PT | 10-20 | training/experience |
| Employee Id 63 | Developer/Engineer | associate | Build | N | PT | 0-5 | training/experience |
| Employee Id 78 | Product Owner | advanced | Plan | Y | PT | 5-10 | training/experience |
| Employee Id 90 | Developer/Engineer | advanced | Build | N | CT | 0-5 | basic |
| Employee Id 93 | Communication Specialists | senior | Plan | N | CT | 0-5 | basic |
| Employee Id 100 | Governance | senior | Run | N | CT | 20+ | basic |
| Employee Id 138 | Analyst | senior | Plan | N | CT | 20+ | basic |

Similarly, the rest of the team members were assigned to teams based on the constraints mentioned.



```
x[1,5] = 1.000000
x[2,8] = 1.000000
x[3,1] = 1.000000
x[4,5] = 1.000000
x[5,2] = 1.000000
x[6,7] = 1.000000
x[7,7] = 1.000000
x[8,11] = 1.000000
x[9,12] = 1.000000
x[10,4] = 1.000000
x[11,4] = 1.000000
x[12,1] = 1.000000
x[13,3] = 1.000000
x[14,2] = 1.000000
x[15,6] = 1.000000
x[16,12] = 1.000000
x[17,2] = 1.000000
x[18,8] = 1.000000
x[19,10] = 1.000000
x[20,4] = 1.000000
x[21,10] = 1.000000
x[22,6] = 1.000000
x[23,2] = 1.000000
x[24,11] = 1.000000
x[25,2] = 1.000000
x[26,1] = 1.000000
x[27,5] = 1.000000
x[28,8] = 1.000000
```

**The resulting minimum cost of this assignment is $429.**

## Conclusion-

Using the logic of assignment method to allocate employees to different teams has proven to be effective. Using assignment method approach helped us in developing a model for our client i.e., Federal Bank of Richmond. The model can take the exact data as input such as how many employees are to be assigned to which team, which time zone to be met, which worker function to be assigned to which team, etc. Once, this data is available, the model can be run to address the assignment issue.

## Recommendations –

Since they keep reorganizing the teams, the bank can use this same model for the next six months, as to reduce the total cost incurred. This model also assigns employees from different time zones location to be in teams which have similar time zones so that they can meet and operate virtually.

# Appendix

May 13, 2021

[1]:

```
from pyomo.environ import *
from brooks import *
import pandas as pd
```

[2]:

```
team_data = pd.read_excel("project_data_model.xlsx",
                          sheet_name="Sheet2")
```

Define data

[3]:

```
W = list(team_data["Person ID"])
S = ["Analyst","Application Architect","Architect","Communication␣
 ↪Specialists","Dev/QA Manager (DevOps)","Developer/
 ↪Engineer","Governance","Operations Specialist","Product Owner","Release␣
 ↪Management","Scrum Master","Solution Architect","Tester"]
E = ["advanced","associate","intermediate","senior"]
P = ["Build","Plan","Run","Scrum"]
Z = ["ET", "CT", "PT"]
Y = ["0-5","10-20","20+","5-10"]
A = ["basic","highly experienced and certified","training/experience"]
T = [1,2,3,4,5,6,7,8,9,10,11,12,13,14]
#timezone = dict(zip(W,list(team_data["Time Zone"])))
teams_max = 14
tsplus = 2
tsminus = 2
feplus = 1
feminus = 1
fpplus = 1
fpminus = 1
function_traget = 3
agile_weekly_cost = 2066.67
time_zone_cost = 4
team_size_cost = 14
function_exp_cost = 6
product_owner_cost = 3
area_cost = 5
```

Declare a model object

[4]:

```
model = Model()
```

Declare non negative for variables

[5]:

```
if hasattr(model,'x'):
    model.delete(model.x)
model.x = Var(W,T, domain = Binary)
```

[6]:

```
if hasattr(model,'y'):
    model.delete(model.y)
model.y = Var(T,Z, domain = Binary)
```

[7]:

```
if hasattr(model,'z'):
    model.delete(model.z)
model.z = Var(T, domain = Binary )
```

[8]:

```
if hasattr(model,'e'):
    model.delete(model.e)
model.e = Var(T,E, domain = NonNegativeIntegers)
```

[9]:

```
if hasattr(model,'f'):
    model.delete(model.f)
model.f = Var(T,P, domain = NonNegativeIntegers)
```

[10]:

```
if hasattr(model,'g'):
    model.delete(model.g)
model.g = Var(Z,T, domain = Binary )
```

[11]:

```
if hasattr(model,'he'):
    model.delete(model.he)
model.he = Var(T, domain = Binary )
```

[12]:

```
if hasattr(model,'te'):
    model.delete(model.te)
model.te = Var(T, domain = NonNegativeIntegers )
```

[13]:

```
if hasattr(model,'tsplus'):
    model.delete(model.tsplus)
model.tsplus = Var(T, domain = NonNegativeReals, bounds = (0,2))
```

[14]:

```
if hasattr(model,'tsminus'):
    model.delete(model.tsminus)
model.tsminus = Var(T, domain = NonNegativeReals, bounds = (0,2))
```

[15]:

```
if hasattr(model,'feplus'):
    model.delete(model.feplus)
model.feplus = Var(T,E,domain = NonNegativeIntegers, bounds = (0,1))
```

[16]:

```
if hasattr(model,'feminus'):
    model.delete(model.feminus)
model.feminus = Var(T,E,domain = NonNegativeIntegers, bounds = (0,1))
```

[17]:

```
if hasattr(model,'fpplus'):
    model.delete(model.fpplus)
model.fpplus = Var(T,P,domain = NonNegativeIntegers, bounds = (0,1))
```

[18]:

```
if hasattr(model,'fpminus'):
    model.delete(model.fpminus)
model.fpminus = Var(T,P,domain = NonNegativeIntegers, bounds = (0,1))
```

Define the objective function

[19]:

```
if hasattr(model, 'cost_objective'):
    model.delete(model.cost_objective)

model.cost_objective = Objective(expr = (agile_weekly_cost*sum(model.z[j]  for j␣
 ↪in T))+ (time_zone_cost*sum(model.y[j,k] for j in T for k in Z))+␣
 ↪(team_size_cost*sum(model.tsplus[j] + model.tsminus[j] for j in T)) +␣
 ↪(function_exp_cost*sum(model.feplus[j,k] + model.feminus[j,k] for j in T for k␣
 ↪in E))+ (product_owner_cost*sum(model.x[i,j] for i in W for j in T )) +␣
 ↪(area_cost* sum(model.fpplus[j,k]+model.fpminus[j,k] for j in T for k in P if␣
 ↪k!="Scrum"))
 ,sense=minimize)
```

Specifying Allocation constraint

[20]:

```
# All workers must be allocated constraint
if hasattr(model, 'allocation_constraint'):
    model.delete(model.allocation_constraint)
model.allocation_constraint = ConstraintList()
for i in W:
    if S != "Scrum Master":
        model.allocation_constraint.add(sum(model.x[i,j] for j in T) == 1)
```

Team size constraint

[21]:

```
# 1st Constraint
if hasattr(model, 'teamsize_constraint'):
    model.delete(model.teamsize_constraint)
model.teamsize_constraint = ConstraintList()
for i in W:
    model.teamsize_constraint.add(sum(model.x[i,j] for j in T) <=␣
 ↪(10+tsplus-tsminus))
```

Functional expertise diversity constraint

[22]:

```
# 2nd DE Constraint
if hasattr(model, 'Exp_Div_constraint'):
    model.delete(model.Exp_Div_constraint)
model.Exp_Div_constraint = ConstraintList()
for k in E:
    model.Exp_Div_constraint.add(sum(model.e[j,k] for j in T) <= (2.
 ↪5+feplus-feminus))
```

Product owner constraint

[23]:

```
# 3rd Product Owner Constraint
if hasattr(model, 'Po_constraint'):
    model.delete(model.Po_constraint)
if P == "Y":
    model.Po_constraint = ConstraintList()
    for j in T:
        model.Po_constraint.add(sum(model.x[i,j] for i in W) >= 1)
```

Functional constraint

[24]:

```
# 4th funtional Constraint
if hasattr(model, 'functional_constraint'):
    model.delete(model.functional_constraint)
model.functional_constraint = ConstraintList()
for j in T:
    for k in P:
        if k != "Scrum":
            if P == k:
                model.functional_constraint.add(sum(model.x[i,j] for i in W  )␣
 ↪== function_traget+model.fpplus[j,k]-model.fpminus[j,k])
```

[25]:

```
# 5th Time zone Constraint
#if hasattr(model, 'Time_constraint'):
#    model.delete(model.Time_constraint)
#model.Time_constraint = ConstraintList()
#for j in T:
#    for i in W:
#        model.Time_constraint.add((model.x[i,j]) <= model.y[j,timezone[i]])
```

Agile constraint

[26]:

```
# 6th Agile constraint
if hasattr(model, 'he_flip_off'):
    model.delete(model.he_flip_off)
model.he_flip_off = ConstraintList()
for j in T:
    for i in W:
        if A == "highly experienced and certified":
            model.he_flip_off.add(sum(model.x[i,j])>= model.he[j])
```

Agile constraint for highly experienced

[27]:
```
if hasattr(model, 'te_flip_off'):
    model.delete(model.te_flip_off)
for j in T:
    model.te_flip_off= Constraint(expr = sum(model.x[i,j] for i in W if A in
                               ["highly experienced and certified","training/
 ↪experience"])- 0.3* sum(model.x[i,j] for i in W) >= -12*(1-model.te[j]))
```

Scrum Master constraint

[28]:
```
# 8th scrum master Constraint
if hasattr(model, 'scrum_master_constraint'):
    model.delete(model.scrum_master_constraint)
model.scrum_master_constraint = ConstraintList()
for j in T:
    if S == "Scrum Master":
        model.scrum_master_constraint.add(sum(model.x[i,j] for i in W) <= 2 and␣
 ↪(model.x[i,j]for i in W) >=1)
```

Specify solver and solve

[29]:
```
solver = SolverFactory('gurobi')
status = solver.solve(model)
```

Get status

[30]:
```
print('status = %s' % status)
```

```
status =
Problem:
- Name: x2493
  Lower bound: 429.0
  Upper bound: 429.0
  Number of objectives: 1
  Number of constraints: 292
  Number of variables: 2340
  Number of binary variables: 2058
  Number of integer variables: 2311
  Number of continuous variables: 29
  Number of nonzeros: 4205
  Sense: minimize
Solver:
- Status: ok
  Return code: 0
  Message: Model was solved to optimality (subject to tolerances), and an
```

```
optimal solution is available.
  Termination condition: optimal
  Termination message: Model was solved to optimality (subject to tolerances),
and an optimal solution is available.
  Wall time: 0.015626907348632812
  Error rc: 0
  Time: 0.18144750595092773
Solution:
- number of solutions: 0
  number of solutions displayed: 0
```

## 0.1 Result

Get solution and objective function value

[31]:

```
for i in W:
    for j in T:
        if value(model.x[i,j])>0:
            print('%s = %f' % (model.x[i,j], value(model.x[i,j])))
print("objective = %f" % value(model.cost_objective))
```

```
x[1,5] = 1.000000
x[2,8] = 1.000000
x[3,1] = 1.000000
...
...
x[142,4] = 1.000000
x[143,3] = 1.000000
objective = 429.000000
```