Resource Planning ve Organizational Structure Meselesi

- Öncelikle organizational structure konusunda generic bir program mı yazıcaz yoksa her organizational structure için farklı bir template şeklinde mi olucak projemiz ona karar vermemiz lazım. En çok kullanılan 3 organizational structure:

    - Project based Org. Structure(projectized)
        - http://pmstudycircle.com/2012/08/what-is-a-projectized-organization-structure/
    - Functional Org. Structure
        - http://pmstudycircle.com/2012/08/what-is-a-functional-organization-structure/
    - Matrix Org. Structure(**kazananımız bu**)

    - **project based** olan her proje için farklı insanların atanması ile oluşuyor. Mesela Proje 1 için Ahmet Mehmet çalışıyor ve proje müdürleri de Kazım diyelim. Proje 2 için de Ayşe Fatma çalışıyor ve proje müdürü de Kemal. Bu bağlamda bu insanların İK Departmanı benzeri özelleşmiş bir departmanları yok, sadece proje bazlı yeteneklerine göre bir araya geliyorlar.

    - Functional organizational structure ise İK, IT gibi fonksiyonel olarak insanları gruplara ayırarak fonksiyonel görevlerne odaklanmalarınıamaçlıyor.

    - Matrix de bu ikisinin karışımı, çalışanlar hem functional görevlerine hem de atandıkları projelere birlikte çalışıyorlar. Bu biraz daha ideal çünkü diğer ikisini de barındırıyo aslında.

- Benim önerim Matrixe uygun olucak şekilde programı yazıp sonradan gerekirse diğerlerine de uyarlamak. Önemsiz gibi dursa da şirketten şirkete çok fark ediyor eğer ürün olarak çıkartıp satmak da aklımızda varsa.

Resource Planning ve Self-Improvement/Talent Management Meselesi

- Bu başlık altında da çalışanların proje boyunca hangi parametrelerin nasıl artacağını ve nasıl azalacağına dair bulduğum kaynakları özetliycem. (Ama önce parametreler seçmek lazım tabi :DD)

Employee Parametre Seçimi Linkler

http://www.citehr.com/28046-performance-parameters-company.html

http://smallbusiness.chron.com/list-attributes-used-employee-performance-reviews-33245.html

http://www.eremedia.com/fordyce/7-easy-ways-to-measure-employee-performance/

Linklerdeki Parametreler:

Workaholism
Punctuality
Regularity
Discipline
Initiative
Behaviour With Collegues(Communication Skills/Being a Team Player**)**

Bu 5'i var sanırım olası parametrelerde

Bence buraya bir de workaholism ekleyelim. Eğer bir task azcık insan üstü bir çalışma isterse, fazla mesai yapıcak biri gerekirse workaholism fazla olanı atarız?
ok mantıklı
regularity ne abi ne alaka
taam
ya da direk total bir önem formülü mü çıkarsak burdan

anladığım kadarıyla istikrar gibi, consistency yapabiliriz.

evet benim fikrim de şöyle: parametre sayımız çok olucak. ve buna göre büyük bir formül çıkartıcaz, her parametreyi formule katarak. bu formülde hem employee parametreleri hem de task difficultyleri olcak.
mesela a = formül olcak taskları a'ya göre mi dağıtcaz
yoksa a= formül1
b=formül2

tasklar a ve b ye göre mi dağıtılcak
Formül : a ise anın sonucuna göre bir treshold koyabiliriz, veya bunu bir distribution haline getirip, belirli değerler arasını sadece employer parametreleri ile hazırlayacağımız bir b formülünün sonucuna göre atarız.

Mesela formül a'ya göre 3. derece zor bir taska, formül b'ye göre 2.derece çalışkan bir adam atanabilir gibi. Bu mantıklı bence hem birden çok formül daha çok intelligence :))))))))))

holeeeeeeeey hadi bundan yapalıım

o zaman şimdi parametrelerimize daha çok ekleyelim mi. eklemeyeceksek formülü düşünelim.

bu kadar parametre yetersiz bence, çok parametremiz olsun birkaçını birleştiririz zamanı gelince olmadı, ya da sileriz gereksizleri, formül işi biraz daha karışık, onun için kağıt üzerinde çalışmak lazım, parametre ekleyip ona bakalım

tamam o zaman bundan sonraki süreçte parametre düşünüyoruz
Tamamdır

Parametreler ve Açıklamaları:

- **//Regularity: Bu iş performansındaki iniş çıkışların azlığı gibi yorumlanabilir. Bunu da due date'i yakın olan tasklar ataması için kullanabiliriz.(sabit)**
- **//Discipline: bu regularityle aynı şey sanki bu konsepte. Ya da düzenli çalışma olarak düşünebilir ama o da performans oluyor. Buna emin olamadım.(sabit)**
- **//Creativity: bunu yaratıcılık isteyen taskların ataması yapılırken kullanabiliriz.(sabit)**
- **Ability with types of tasks: Bu direk tasklar hakkındaki bilgisi olur. Tasklar türlere göre bu kritere bakılarak atanır.(değişebilir)**
- **Depreciation: yıpranma payı. Bu çalışma ve dinlenme saatlerine göre belirli oranlarda hesaplanacak.**
- **Workhaolism difficulty.(saat olarak fazladan çalışma eğilimi)(sabit)**

Task parametreleri:

- **Due time: Taskın ait olduğu projenin due date'i.**
- **Priority: Taskın ait olduğu projenin önemi.(project)**
- **Min required knowledge about subject matter.(C++ 5 birim mesela en az)**
- **Prereq: task sırasını belirliycek.(birden çok taskın olabilir)**

/////Olası algoritma(ilter):

Farklı task queuelar olur. Her queue taskları bir özelliğe göre sıralar. Ayrıca insanları da farklı quelarda sıralayabiliriz.

İnsan queueları:

- **Workaholism queue:** workhaolism, discipline ve initiative kullanılır.
- **Speed queue:** punctuality, regularity, discipline kullanılır.
- **Collaboration queue:** behaviour with collegues kullanılır.
- **Task queues:** bu knowledge with types of tasks kullanılır. Burada birden çok queue olcak her çeşit task için.

Task queuesu:

- Taskları formüle göre buraya sırala.

Scheduler

Task queueuesundaki taskları insan quelarına göre dağıt.

Scheduler sorts tasks of projects with respect to project's priorities and due left values.

(**There will be a constant a such that "importance=a*(priority)+(1-a)*(48/due left)" and if a task is pre of another task it must be before that task**)

They are stored in a queue. If same tasks appears the higher importance will be accepted in the queue(no repetition of tasks in the queue[think this as a dynamic programming]).

TheoricTaskTime(hours): (minimum 1 hour maximum 8 hours)
(Tasks smaller than 1 hour is ignored as they can be considered as everyday jobs.)

Real task time = (10/Ability)*TheoricTaskTime*(Current Depreciation Level/10))/C

Depreciation Level: (Decreases as the resuource used and increased as the resource set to a rest state(holiday)) Max:10 Min:0

*Depreciation Decrease Coefficient: 0.2(bunu dinamik hesaplıycaz mesela adamın bilgisi 10 task 5 gerektiriyosa azalmasın gibi)*TaskWorkhaolismLevel/hour(bura düşünülebilir Melih demişti)*

Depreciation Increase Coefficient: 0.5(bu coefficienti belirleyelim)/hours(Bura düşünülebilir Melih demişti)

Algorithm:

```
constant passTreshold
now = 0
while now <= 60//As there are 12x5 = 60 work hours in a week(assumed)
    If pass<=passTreshold
        Pop the first task from the queue
        Check task type
        Find best match for the task type by calling (*) algorithm//(*)algorithm
            is called with parameters C and taskType
        If the queue is empty
            Push task into the queue
```

```
                    If now mod 12 = 8
                        Increase now by 4
                    Else
                        Increase now by 1
                    Continue while loop's next iteration
                End if
                With the best match calculate the real task time with C
                Assign the task to best match in normal working hours
                Increase the best match's knowledge about the task by 1 at the
                    completion time of the task
                Decrease the depreciation level of the best match by
                    (Depreciation Decrease Coefficient)*(Real Task Time)
                Decrease each type of knowledge(except the task it completes) of all
                    people in the group by 0.01*(real task time) at the
                    completion time of the task // 0.01 too high
            End of if
            Else
                Pop the first task from the queue
                Check task type
                If now mod 12 >= 8
                    Find best match for the task type by calling (**)
                        algorithm//(**)algorithm is called with parameters C and
                        taskType
                Else
                    Find best match for the task type by calling (*)
                        algorithm//(*)algorithm is called with parameters C and
                        taskType
                End else-if
                If the queue is empty
                    Increase now by 1
                    Continue while loop's next iteration
                End if

                With the best match calculate the real task time with C
                Assing the task to best match such that
                    If group consists of people without workhaolism assign in
                        normal work hours
                    Else if group consists of people with workhaolism assign in
                        every hour
                Increase the best match's people's knowledge about the task by 1 at
                    the completion time of the task
                Decrease the depreciation level of the best match by
                    (Depreciation Decrease Coefficient)*(Real Task Time)
                Decrease each type of knowledge(except the task it completes) of all
                    people in the group by 0.01*(real task time) at the
                    completion time of the task
            End else
        end of while
```

If any project fails to be completed before its due time.
Incerase pass by one
Increase C by one
Restart the process
End of if

(*)algorithm (input C, input taskType, input time)
Create a queue of groups of C people available at time with respect to realtask time of the taskType.
Sort the list
Return the first group in the queue

(**)algorithm (input C, input taskType, input time)
Create a queue of groups of C people available at time, who possesses workhaolism, with respect to realtask time of the taskType.
Sort the list
Return the first group in the queue

/////Olası algoritma sonu(ilter):

ai kapalı olan scheduler -> ilter
ai'lı scheduler => algorithm
Xml parser -> çağatay
örnek input proje
algorithm(*) ve algorithm (**)