Kanban Notlarım v1 - 2016 - 02

Taditional sword maker builds quality in. (Image via Jon Archers Bloghttp://castleinskidrow.blogspot.c...)

Altuğ Bilgin Altıntaş - 2016 - altug@kodcu.com

2012 yılından beri Yalın (Lean), Çevik süreçler (Agile process), Scrum ve Kanban hakkındaki düşüncelerimi kodcu.com adresinde yazıyorum. Bu yazılarımı derleyip ufak bir kitap haline getirerek yazılım projelerinde Kanban mekaniklerini daha rahat giriş yapmanızı amaçladım.

Yalın kültür ile 2010 yılında danışmanlık yaptığım Toyota Adapazarı fabrikasında tanıştım. Fabrika içerisinde her şeyin düzenli ve sistematik ilerlemesi beni çok etkilemişti. Kariyerim boyunca Türkiye'de ki hemen hemen en büyük firmalarında çalışmama rağmen hiç birinde böyle bir kültür ve düzen görmemiştim. Bu işleyişin arkasındaki kültürün son derece önemli olduğunu gördüm. Bu kültürün oluşmasında Yalın (Lean) yaklaşımın önemi anladıktan sonra Yalın yaklaşıma olan ilgim daha da arttı.

Toyota fabrikasından gördüğüm kültür ve mekanikleri acaba yazılım dünyasına uygulayabilir miyim diye düşünürken, 2011 yılında David Anderson ile tanıştım. David Anderson 2006 yılında Kanban Microsoft şirketinde Kanban yaklaşımını başarı bir şekilde uygulayarak Kanban yaklaşımını markalaşmıştı bile.

Kanban, Scrum veya başka mekanik sistemler güzeldir. Size odanın ışıklarını açmanızı ve kaplanları (tehlikeleri) gösterirler ama bu tehlikeler ve engeller karşısında çözüm üretecek olan sizsiniz.

İçerik

Bölüm 1

Kanban Sisteminde Tahmin ve Ölçüm Yaklaşımları

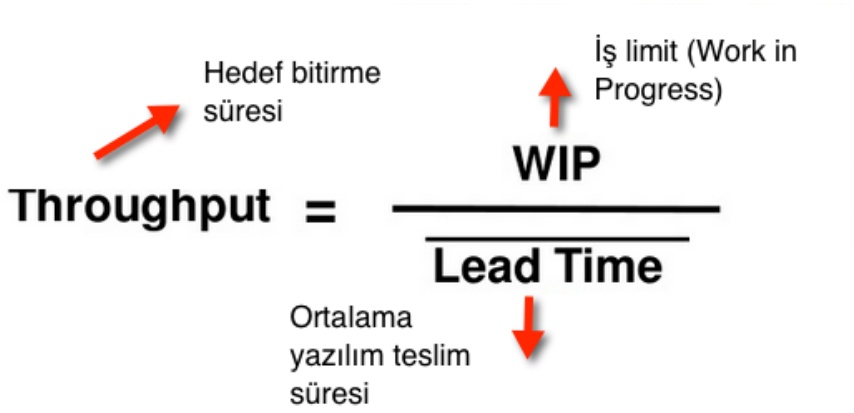
Çevik (agile) yöntemlerin en büyük hedefi müşteriye istenilen yazılımı en kısa sürede ve doğru bir biçimde teslim edilmesini sağlamaktır. Çevik yöntemler kalite noktasında çok fazla birşey söylemezler, işte bu noktada yalın (lean) yaklaşımın öğretilerini Kanban sistemiyle devreye alabilirsiniz.

Kanban sistemi aşağıdaki noktalarda projenize katkı sunabilir.

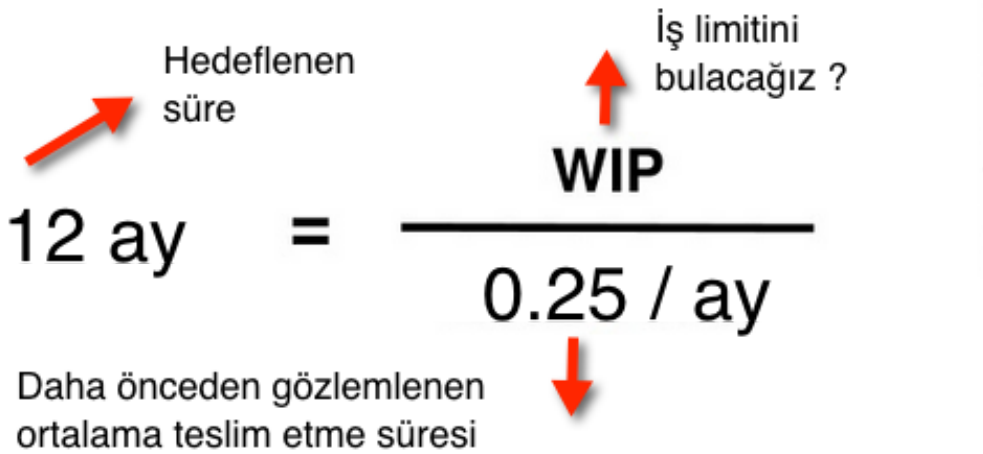
- Proje akışının görünürlüğü arttırır (kim neyi yapıyor, yolu tıkayan bir problem var mı ?)
- Aynı anda yapılan işe limit getirerek yazılımın daha hızlı ve kaliteli bir şekilde müşteriye teslim edilmesini sağlar
- Proje akışında bir tıkanıklık varsa bunu hemen fark edilmesine sağlar.
- Tıkanaklıklar çözülmesi için ateşleyici etkiye sahiptir.

WIP (Work in Progress) limitini ve Little's Law

Kanban sistemini bir kuyruk yapısına benzetebiliriz. İşler bittikçe arkadan gelen yeni işler üzerinde çalışmaya başlanır. Bu noktada geleceği planlamak için Little's Law formülüyle WIP limitinin ne olması gerektiğini hesaplayabiliriz.
WIP limiti projeye ve ekibe göre farklılıklar gösterebilir. Bu yüzden Kanban sistemi bütün projelerde rahatlıkla uygulanabilir. WIP limitinin ne olacağı, Little's Law kolayca hesaplayabilirsiniz.



Projenin bitiş zamanı belli ise ve bir işin ortalamada ne kadar da bitirdiğinizi biliyorsanız (Örneğin 1 haftada bir özellik) , Kanban sisteminde WIP limitinizi kolayca hesaplayabilirsiniz. Örnek :



12 x 0.25 = 3
WIP limiti 3 olmasında fayda var.
Eğer bir kişiye 1 birimlik iş düşüyorsa projede 3 kişi eş zamanlı olarak çalışabilir.

Eğer projeniz 1 sene sürecekse ve ortalamada yaklaşık 1 haftada 1 işi teslim etme oranınız varsa, WIP limitinizi 3 olarak ayarlayabilirsiniz. Bu durumda 3 kişi ile bu proje başlatılabilir.

Limit şart mı ?

Bir işi hakkıyla bitirebilmek için odaklanmak gerekir. İşe limit getirmek odaklanmayı sağlayacağı için gereksinimlerin teslim süresini (lead time) hızlanacaktır.
Hepimizin başına gelmiştir, aynı anda 10 iş yapayım derken ortaya hiç bir şey çıkmaz. Bunun başlıca sebebi işten işe geçerken arada kaybolan zamandır (switching cost). İşe limit getirmek bir kişinin bir işten başka bir işe atlamasını engelleyeceği için otomatik olarak zaman kazanılır.
Ayrıca daha az hatalı (bug) ürünler ortaya koymanızı sağlar. Yalın sisteme göre hatalar (bug) en büyük israftır. Muda (israf) projelerde kaçınılması gereken en önemli noktadır.

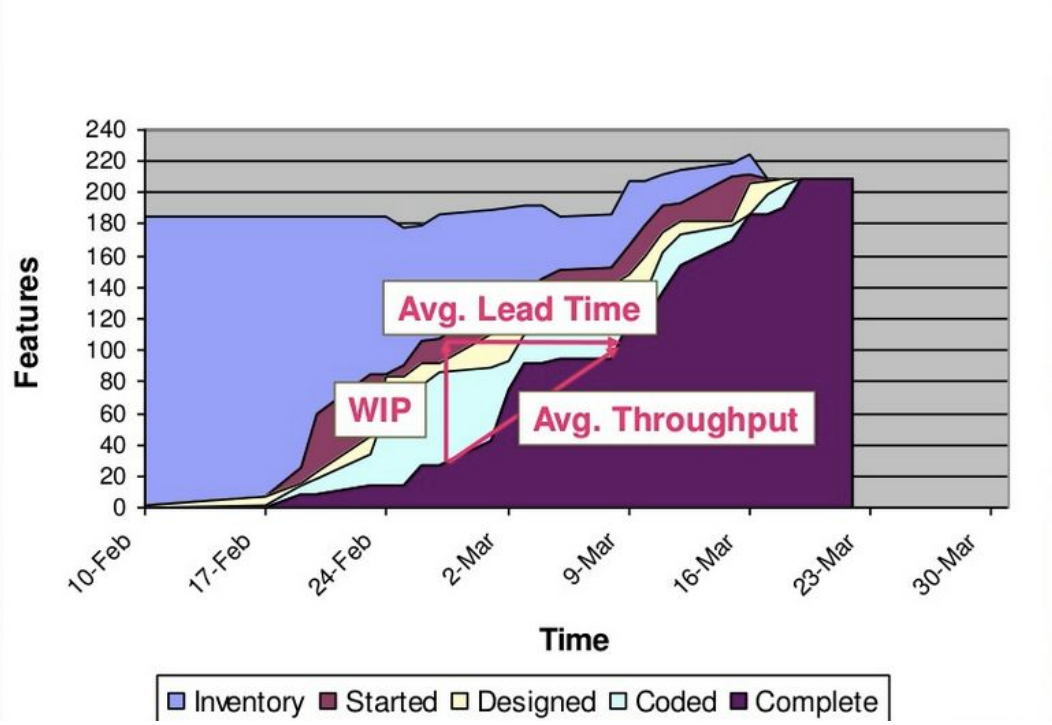
Kanban sistemine göre projenin takibi ?

Benim kullandığım iki popüler grafik türü var;

1. CFD (Cumulative Flow Diagram)
2. Haftalık Hız (Velocity) Analizi

David Anderson'ın kitabında da gösterdiği CFD grafiği, standart "burn down" grafiklerine göre daha net bir görüş sağlamaktadır.

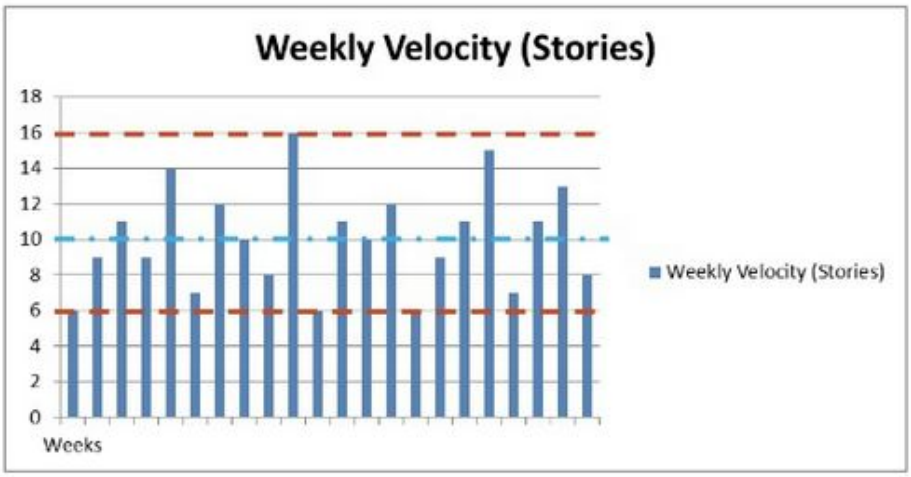
CFD (Cumulative Flow Diagram)



Görüldüğü üzere açık mavi ile gösterilen gereksinim miktarının zaman içinde azaldığını ve mor ile gösterilen tamamlanma miktarının git gide çoğaldığını görebilirsiniz. Problemli bir projede gereksinim tamamlanma miktarı düz bir seyir izler. CFD grafiği projenin hangi durumda olduğunu anlamak için iyi bir göstergedir.

Haftalık Hız (Velocity) Analizi

Bir haftada kaç gereksinim bitti ? Bu göstergede projelerin sağlığını ölçmek için son derece önemlidir.



Investment Bank, London, Extreme Programming
Weekly Mean 10, Max = 16, Min = 6
Spread (+/- 1.6x)

Kanban sistemi baltanızı bilemek için size zaman verir

Sürekli iş üzerinde çalışmak iyi değildir. Devamlı %100 CPU devrinde çalışmak bir süre sonra verim yerine verimsizlik getirecektir. Proje ekibine boş zamanlar oluşturmak onların baltalarını bilemeleri için zaman verecektir. Bu da işlerin daha hızlı bir şekilde bitmesine sağlayacaktır.

Kaizen için öncesinde Kanban sistemi

İşlerinize limit getirmek problemlerin daha çabuk bir şekilde fark edilmesini sağlayacaktır. Fark edilen her problem gelişim (Kaizen) için en iyi fırsattır. Kaizen ruhunu yakalayabilmeniz için Kanban sistemi kesinlikle incelemenizi ve uygulamanızı öneriyorum.

Bölüm 2

Daha iyi proje yönetimi için 3 öneri

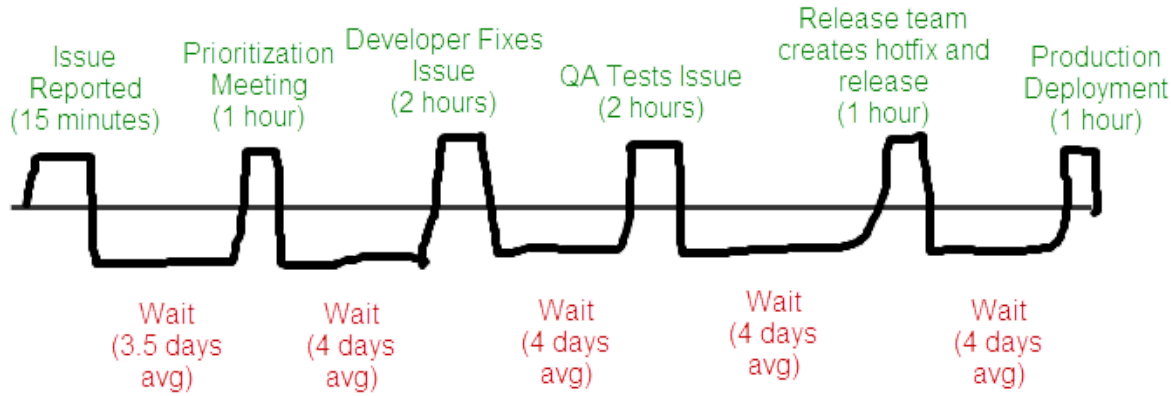
Başlığın biraz iddialı olduğunun farkındayım. Her proje ve her ekip birbirinden çok farklıdır. Bu yüzden herkes için sihirli öneriler sunacak değilim. Bu yazımda sizlere aktarmak istediğim çekirdek öneriler; yani projelerinizde uygulayabileceğiniz 3 adet basit ve esnek yaklaşımdır. Özellikle esnek dememin sebebi, bu yöntemlerin her projede uygulayabilme ihtimalinin fazla olmasını vurgulamak istememdir. Bu üç yöntem sırasıyla

1. Görselleştirin
2. Limit koyun
3. Ölçün

Öneri 1 : Görselleştirin; özellikle iş akışınızı

Şu an projelerinizde her ne yapıyorsunuz görselleştirin. Görselleştirmek süreçlerinizi daha net bir şekilde görmenizi sağlayacaktır böylece potansiyel *muda* (israf) noktalarını bulma şansınız artar. Elbette öncelikle yaptığınız işleri süreçlere bölmeniz gerekir. W. Edwards Deming dediği gibi “Eğer yapılan işi süreç haline getiremiyorsanız, ne yaptığınızı bilmiyorsunuz demektir.”

Elbette yukarıdaki gibi bir süreç gösterimi harika olurdu ama basitçe aşağıdaki gibi yapılan süreç tarifi de kafi olur. Yeter ki görsellik olsun. Görsellik 1. numaralı kural diyebiliriz.



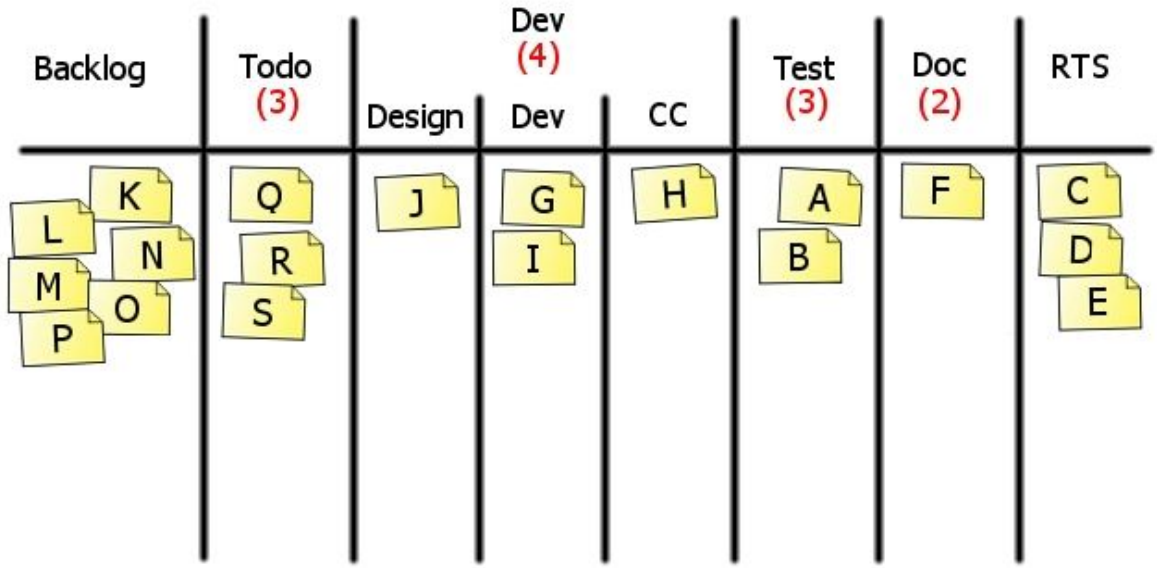
Kısacası gün içerisinde ne yapıyorsunuz ? Tekrar tekrar nerelerde takılıyorsunuz bunu ortaya çıkartmak. Ortaya çıkartmanın en iyi yolu bunu görselleştirmek. Süreçlerin herkes tarafından net anlaşılması ve darboğazların ortaya çıkması için görselleştirme kesinlikle en önemli maddelerden bir tanesidir.

Öneri 2 : Süpermen değilsiniz

Fiziksel limitlerimizin genelde farkındayızdır, örneğin apartmanın 2. katına çıkmak için zıplayaMAyız çünkü bu şekilde tasarlanmamışızdır. Onun yerine fiziksel limitlerimizi bilir ve merdiven veya asansör kullanırız. Yada vücut geliştirmeye giderken daha ilk günden 100 kg ile çalışmaya başlamayız (istisnalar olabilir).

Fiziksel limitlerimizi iyi bilmemize rağmen iş zihinsel aktivitelere gelince limit koymuyoruz ve bunun doğal sonucu olarak işler birbirine giriyor. Devamlı üzerinde çalışılan işler bölünüyor. Aynı anda 10 farklı iş yapılmaya çalışılıyor vb… daha niceleri. Sonuç doğal olarak stres ve verimsizlik.
Strese girmemek ve rahat çalışabilmek için yapılacak işlere limit getirmenizi kesinlikle öneriyorum (öncesinde görselleştirme lütfe). Yani işi körü körüne iteklemek yerine, iş bittikçe havuzdan bir sonraki işi çekmek sizi çok verimli kılacaktır.
Bahsettiğim sistemin adı kısaca Kanban. Kanban yaklaşımını projeler uygulayabileceğiniz gibi kişisel yaşantınızda da uygulayabilirsiniz, bu konuda yazılmış Personel Kanban kitabını okumanızı tavsiye ederim.

Aşağıdaki Kanban tahtasında kırmızı ile gösteren sayılar, işin ilgili süreçlerinin limitleridir. Örneğin geliştirme (*development*) aşamasında aynı anda 4 ‘den (projeye göre daha az veya fazla limit koyabilirsiniz) fazla iş olamaz (limitlerimiz olduğunu tekrardan hatırlatırım). Bu hem uygulama geliştiricileri koruyan hemde kalitenin devamlılığı için bir sigorta niteliğindedir.



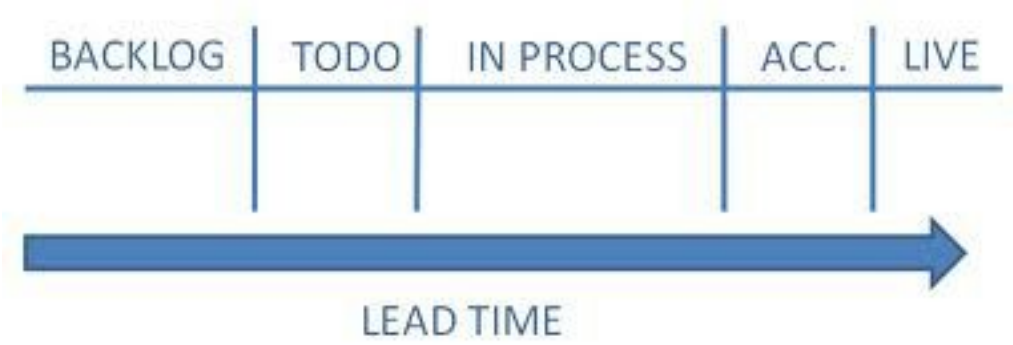
Öneri 3 : Ölçemiyorsan yönetemezsin…

Gerçekleşen performansı kayıt altında almak ve bu bilgi ile gelecek hakkında yorumlar yapabilmek çok kıymetlidir. Zaten insanoğlu olarak derdimiz hep geleceği bilmek ve kontrol altında alabilmek değil mi ? Kanban sistemi bu noktada iki önemli bilgiyi toplamanıza yardım olur. Bunlar :

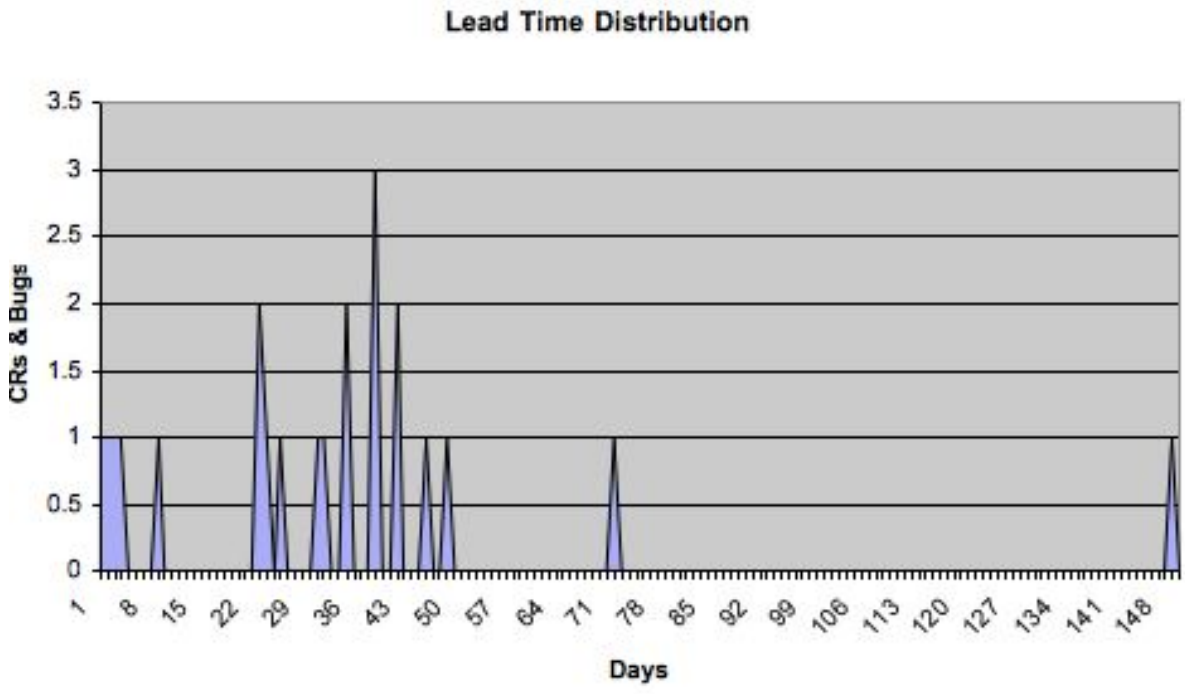
1. Kontrol çizelgesi (bunun detaylarından şimdi bu yazıda bahsetmeyeğim)
2. İş tamamlama (Lead Time) dağılımı çizelgesi

Proje yönetiminde ana performans göstergelerinden biri olan tamamlanma süresi (Lead Time) dağılımı , proje ekibinin performansı ve gelecek performansları hakkında ihtimale dayalı bilgi verebilir. Nasıl mı ?

Öncelikle Lead Time (İş tamamlama) nedir ?



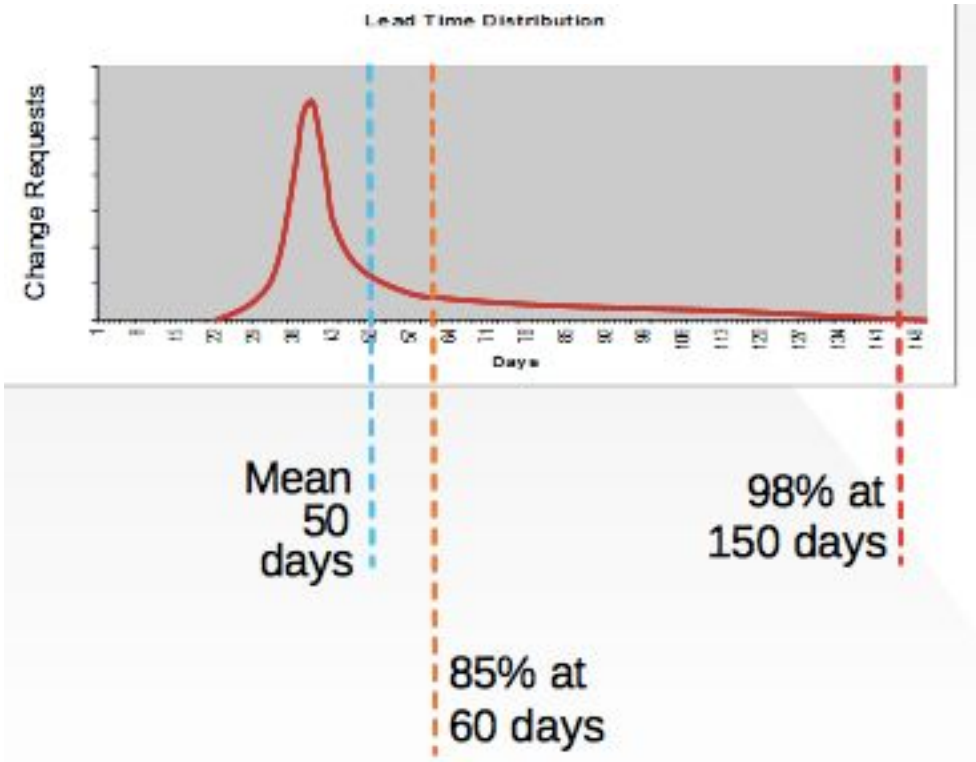
Müşterinin bir işi onaylanıp listeye eklenmesinden, o işin tamamlanmasına kadar geçen süredir. Yani örneğin müşterileri şöyle bir istekte bulunuyor diyelim : "Siparişleri geç teslim eden firmaları görmek istiyorum". Bu istek tahtaya eklendikten sonra kaç gün içerisinde bitti (canlı sistemde kullanıldı) ? Diyelim ki 16 gün sonra. Bu 16 gün o işin tamamlanma süresi (Lead time) olarak kaydediyoruz. Böyle böyle tüm işlerin tamamlanma sürelerini not ettiğimizde aşağıdaki gibi bir dağılım elde edebiliriz.



Örnek bir tamamlanma süresi (lead time) dağılımı yukarıda mevcuttur. Bu dağılım grafiğinin takip ettiği şey, dağılım grafiğinin projenin hangi gününde kaç adet istek tamamlanmıştır.

Peki sonra ?

Bu dağılım grafiği size çok önemli bir şey söyler. Artık müşteri size gelip “Yeni bir istek var, acaba kaç günde biter” dediği zaman. Kafadan atmak yerine daha bilimsel bir şekilde tahminde bulabilirsiniz.



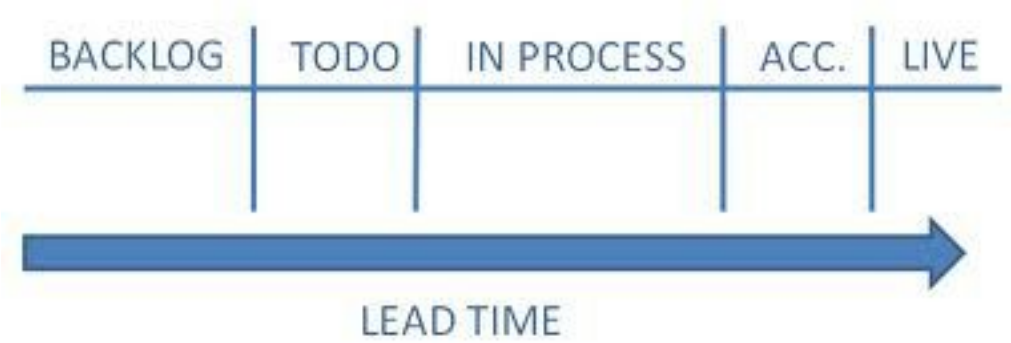
Tamamlanma süresi (Lead time) dağılım grafiğini elde ettikten sonra artık müşterinin bir sonraki istek ne zaman biter sorusuna aşağıdaki gibi cevap verilebilir. Yeni istek ortalama 50 günde. %85 ihtimalle 60 günde ve %98 ihtimalle 150 günde bitecektir diyebilirsiniz. Bu tür bir yaklaşım müşterinin psikolojik olarak ta proje ekibine güven duyma ihtimalini arttırabilir.

Proje ilk başladığında elinizde bu kadar veri olmayabilir. O zaman eski ve hemen hemen benzer bir başka projedeki istatistiksel veriyi kullanabilirsiniz, eğer bu da mevcut değilse yapacak bir şey yok. Son seçenek ihtimal hesabına dayalı tahminleri veri toplanana kadar ertelemek. Buradaki amaç hedef koymaktır, veri olmaması şevk kırıcı olmamalı aksine veri toplamak ve değerlendirme yapmak için bir fırsat olarak görülmelidir.

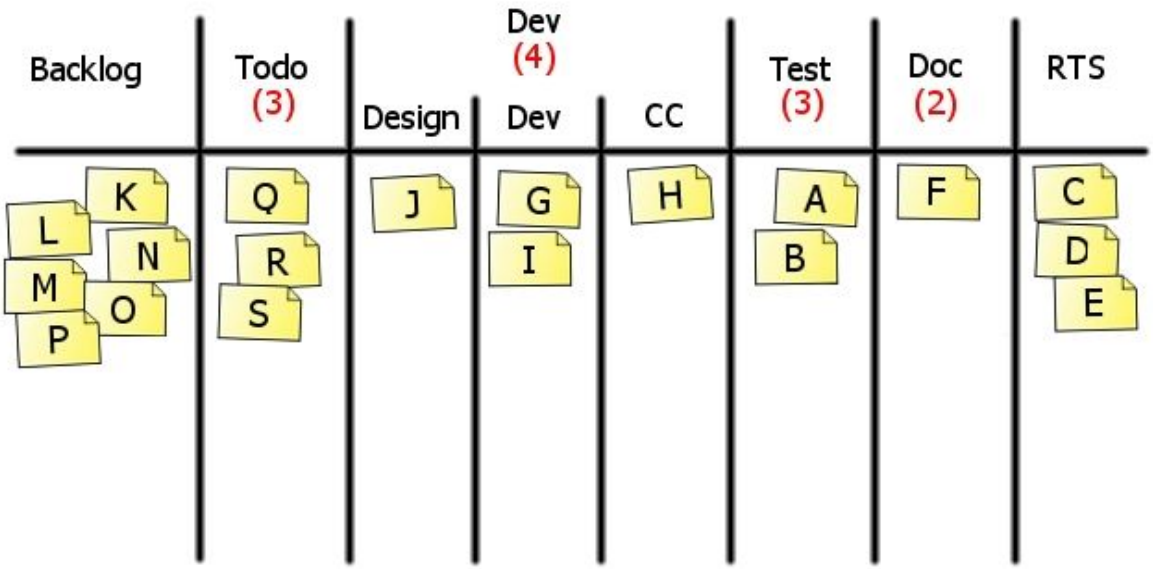
Bölüm 3

Kanban ile kontrol çizelgesinin sağladığı 3 değerli bilgi

Bir işin başlangıcı ile bitişi arasındaki geçen süreye işin teslim süresi (*Lead Time*) denir. *Lead Time* süresinin takip edilmesi, projenin gidişatı için son derece önemlidir.



Aşağıdaki örnekte; ***Kanban*** tahtasında *todo* (yapılacaklar) listesine **3** limiti getirilmiştir. Yani *backlog* (rezerv) alanından en fazla **3** tane iş yapılacaklar (*todo*) listesine geçebilir. İşe limit koymak bizi otomatik olarak en öncelikli işlere odaklanmaya iter – ki bu Kanban sisteminin en büyük avantajlarından bir tanesidir.



***İpucu* : Eğer işin teslim süresini hızlandırmak istiyorsanız, WIP (Work In Progress) limitini aşağıya çekin.**

Yine aynı yukarıda belirtilen Kanban tahtasında *development* (uygulama geliştirme) alanının limiti **4** olarak belirtilmiştir. Elbette bu limitler proje ekibinin sayısı, kapasitesi ve verimliliği ile doğrudan ilişkilidir. Bu limitlerin belirlendiği esnada kesinlikle müşteri tarafıda (ürünü kullanacak kişiler) dahil edilmelidir. Çoğu yerde şahit olduğumuz diğer bir konuda, Kanban sisteminin şeffaflığını gören müşterinin , projeye daha fazla katkı vermeye başlamasıdır.

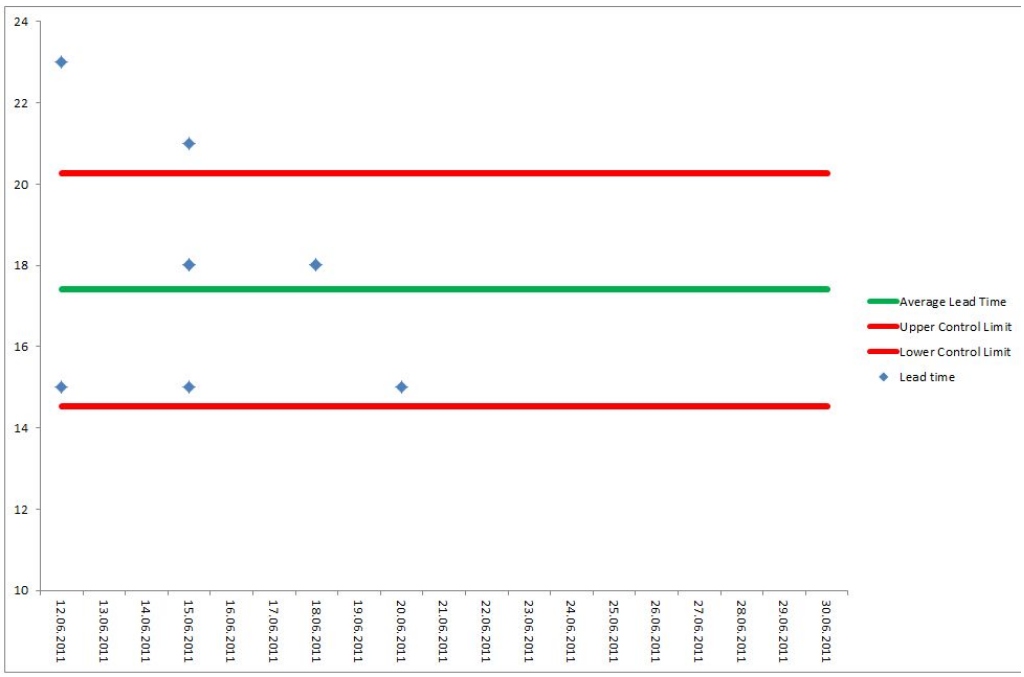
Projenin yolunda gidip gitmediği ölçmek için Kanban sisteminde kullanılan bir çok erken uyarı sistemi mevcuttur. En popüler olanlar :

1- Kontrol çizelgesi (Control Chart)
2- İşin teslim süresi (Lead Time) dağılım grafiği
3- CFD (Cumulative Flow Diagram)

Bugün sizlere kontrol çizelgesinden bahsetmek istiyorum.

Kontrol Çizelgesi

Kontrol çizelgesinin en büyük faydası, teslim edilen işlerin (Lead Time) normalden ne kadar saptıklarını tespit etmektir. Yani bir iş 3 günde, diğer iş 5 günde ama üçüncü iş 28 günde teslim ediliyorsa, bunun görsel şekilde ortaya koymak önemlidir.



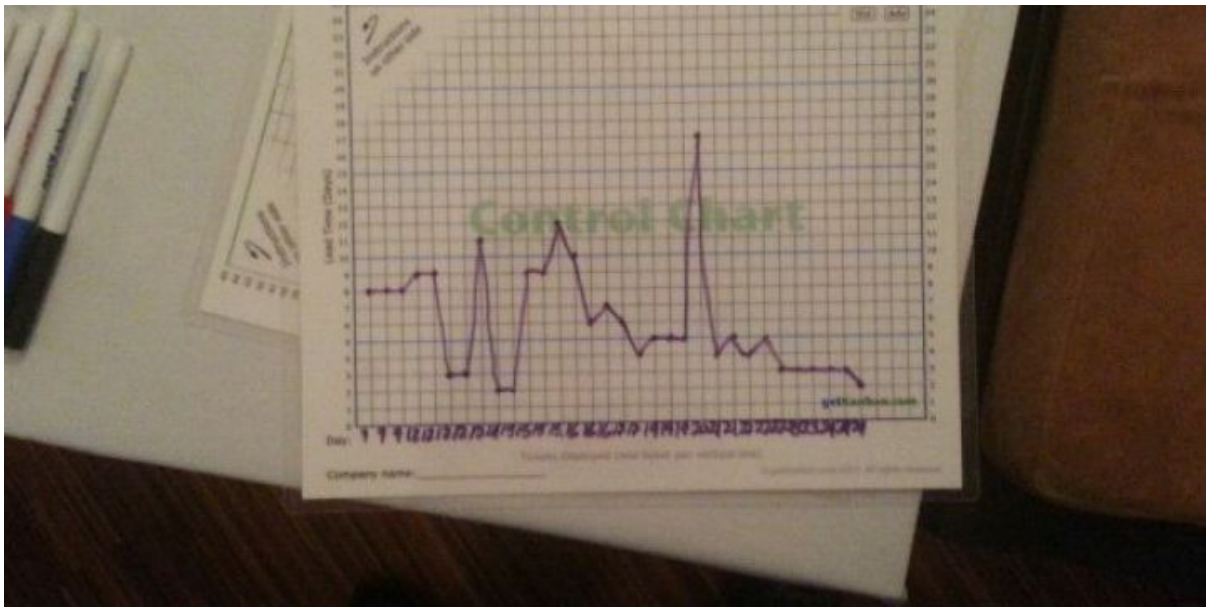
Yukarıda ki örnek kontrol grafik çizelgesine baktığımızda, mavi ufak kristaller ile ifade edilen noktaların işlerin teslim sürelerini (lead time) ifade ettiğini görürüz. Aynı şekilde bu işlerin bitiş süresinin, standartlardan ne kadar saptıklarını da görebiliriz. Alttaki kırmızı çizgi minimum bitiş süresini (ortalama – standart sapmayı) ifade ediyor. Yani minimum sürede iş ne zaman bitirilmesi gerektiğinin cevabını burada bulabiliriz. Aynı şekilde yukarıdaki kırmızı çizgi ise bir işin maksimum bitiş süresini (ortalama + standart sapmayı) ifade ediyor.

Bu iki kırmızı çizginin arasında kalan **yeşil** çizgi ise ortalama işin teslim süresi ifade etmektedir. Bu yeşil çizgiye yakın olarak tamamlanmış işlere, makul ölçülerde tamamlanmıştır denebilir.

Özetlersek kontrol çizelgeleri 3 önemli soruya cevap veriyor.

- İşler minimum ne kadarlık bir sürede tamamlanıyor ?
- İşler maksimum ne kadarlık bir sürede tamamlanıyor ?
- Minimum ve maksimum aradığından sapan gecikmeli işler hangileri ?

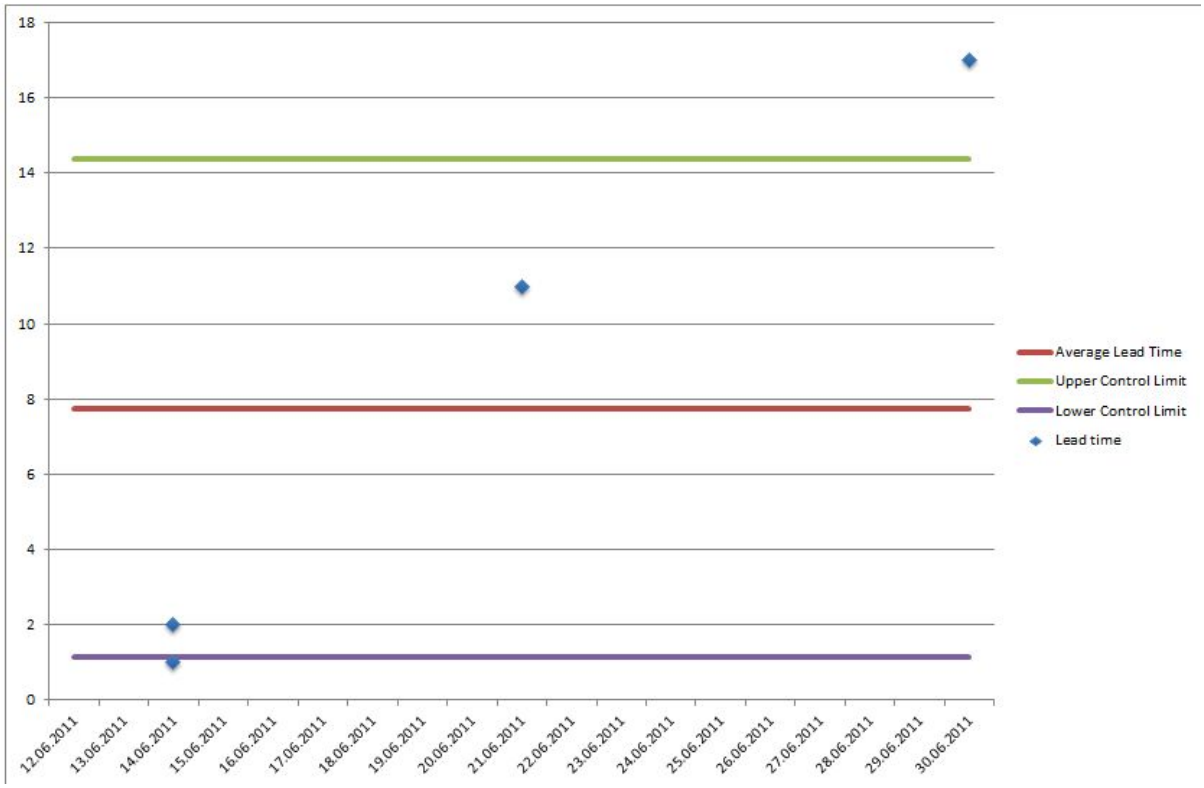
Aynı Kanban tahtasının tıkanmaları anında gösterdiği gibi, kontrol çizelgesinde anormal sürelerde biten işleri anında görebilmemizi sağlar.



Kontrol çizelgesini oluşturmak ve takip etmek için elle basit bir kağıt kullanabileceğiniz gibi benim beğendiğim ikinci opsiyon ise Hakan Forss'un sayfasında gösterdiği excel şablonunu kullanarak elektronik ortamda da takip edebilirsiniz.

	A	B	C	D	E	F	G
1	Id	Title	StartDate	EndDate	LeadTime	Type	Comments
2	1010	Feature 1	20.05.2011	12.06.2011	23	Feature	
3	1011	Feature 2	25.05.2011	15.06.2011	21	Feature	
4	1012	Feature 3	31.05.2011	15.06.2011	15	Feature	
5	1013	Feature 4	31.05.2011	18.06.2011	18	Feature	
6	1014	Feature 5	05.06.2011	20.06.2011	15	Feature	
7	2020	Bug 1	10.06.2011	21.06.2011	11	Bug	
8	2021	Bug 2	12.06.2011	14.06.2011	2	Bug	
9	2022	Bug3	13.06.2011	14.06.2011	1	Bug	
10	2081	Bug4	13.06.2011	30.06.2011	17	Bug	
11	1015	Feature 6	28.05.2011	12.06.2011	15	Feature	
12	1016	Feature 7	28.05.2011	15.06.2011	18	Feature	
13	1017	Feature 8	28.05.2011	15.06.2011	18	Feature	
14	1018	Feature 9	31.05.2011	18.06.2011	18	Feature	
15	1019	Feature 10	05.06.2011	18.06.2011	13	Feature	
16					0		
17					0		
18					0		
19					0		
20					0		
21							
22							
23							

Hakan Forss'un oluşturduğu bu excel şablonu sayesinde girilen işin teslim süreleri otomatik olarak hesaplanarak kontrol grafiğine dönüştürülmektedir.



Elle tek tek uğraşmak yerine, bu işleri otomatik yapan birçok Kanban araçları da mevcut. İlgilenenler aşağıdaki araçları inceleyebilirler.

- http://leankit.com/
- https://kanbanflow.com/
- http://kanbanize.com/

Bölüm 4

Nasıl daha gerçekçi tahminlerde bulunabiliriz ?

Mevcut sıkıntılı durumla daha iyi baş edebilmemiz için elimizde o an anlar için iyi formüllerin (algoritmalar) olması gerektiğine inanıyorum. Bir başka tabirle yumurta kapıya geldiğinde hissi davranışlar göstermek daha kötü sonuçlara sebebiyet verebilir. Yazılım projelerinde daha iyi tahminler nasıl yapılabilir konusunda geçmeden önce sistematik davranmanın nasıl hayat kurtardığını sağlık alanından bir hikaye ile sizlere aktarmak istiyorum.

Apgar formülü ve bebek ölümlerinin azaltması arasındaki ilişki

1953 yılına kadar yeni doğan bebeklerin ilk doğduklarında sağlıklı olup olmadığını ve ilk yardıma ihtiyaç duyup duymadığını ölçmek için bir standart yoktu. Standart olmadığı için bebek yeni doğduğunda, bazı doktorlar sadece kalp atışına, bazıları sadece ağlayıp ağlamadığına gibi noktalara odaklanarak bebeğin önündeki kritik kaç dakika içerisinde tıbbı müdahaleye gerek olup olmadığını tahmin etmeye çalışıyorlardı. Elbette her doktor çoğu zaman o anın karmaşası ile önemli noktaları kaçırabiliyor ve sonuç olarak bebek kayıpları yaşanabiliyordu.

Tahmin için belli standartlar

Doktor Apgard'ın geliştirdiği basit bir algoritma ile yeni doğan bebeğin mevcut durumunu tespit edip , oluşabilecek problemleri erkenden **tahmin** edilmesine olanak verdi. Bu algoritmaya göre yeni doğan bebeğin 5 farklı noktasına (kalp atışı, nefes alımı, refleks, kas gerginliği ve deri rengi) bakıp 0, 1 veya 2 şeklinde skor vererek çıkan sonuca göre tahminde bulunup müdahale edilmeye başlandı.

Eğer bebeğin skoru beş noktaya göre 8 ise yani bebeğin kalp atışları iyi, ağlıyorsa, refleksleri iyi ise, hareketiyse ve rengi pembe ise o zaman bu bebek Apgard algoritmasına göre sağlıklı olarak kabul edilir ve müdahale öngörülmezdi edilmez. Fakat bebeğin skoru 4 veya altında ise bebeğe müdahale edilmelidir sonucu çıkartılabilir.

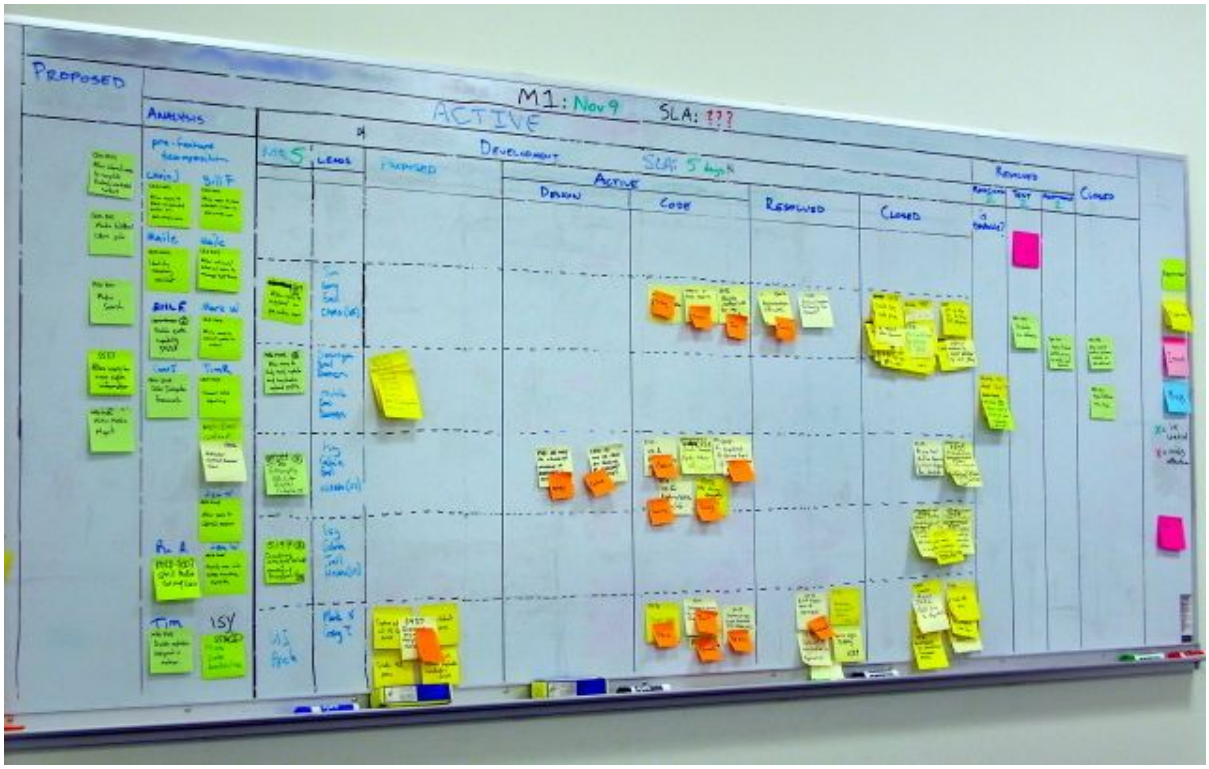
Basit algoritma ile kurtulan hayatlar

Apgard skorlaması bugün bile kullanılmaktadır. Bu basit ve etkili kurallar bütünü (algoritması) bebekleri hayatını halen kurtarmaya devam ediyor. Acil ve zor durumlarda doğru kararlar verebilmek ve isabetli tahminlerde bulunmak için elimizde formüllerin (algoritmaların) olması son derece önemlidir.

Yazılım dünyasında iş tahminleri için belli standartlar var mı ?

Şimdi sağlık sektöründen yazılım sektörüne geçebiliriz. Yazılım sektörü için benim karşılaştığım en sık soru : Bu iş ne zaman biter ? Yada proje ne zaman biter ? Bu tür sorulara cevap vermek için standart bir algoritma kullanıyor musunuz ? Yoksa rastgele bir sayı mı söylüyorsunuz ?

Kanban sisteminin kullanıldığı bir proje de, bu sorulara cevap vermek için, teslim süresi (lead time) dağılımları (distribution) kullanıyoruz. Örneğin %50 ihtimalle 5 günde, %85 ihtimalle 11 günde gibi…



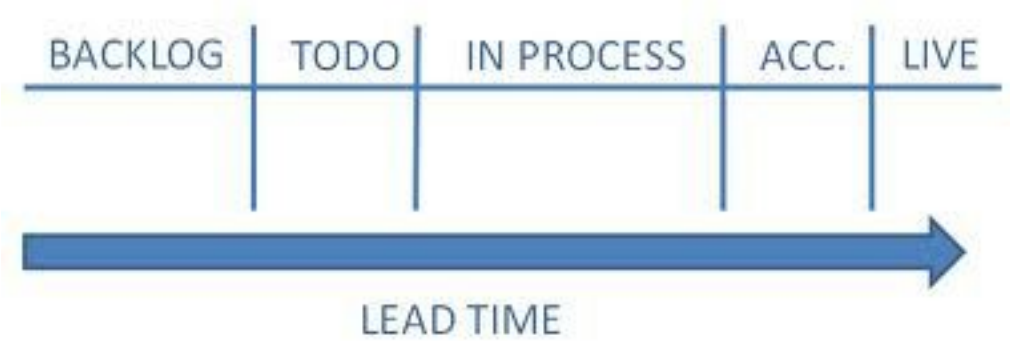
İşin Teslim süresi (Lead time) nedir ?

İşin teslim süresini belirlemek biraz muğlak bir tabir farkındayım. Yani :

İşin tamamlanması aşağıdakilerde hangisi ?

1. Müşterinin yapılan işi canlı ortamda kullanması mı ?
2. Yazılımcının yapılan işi kod seviyesinde tamamlaması mı ?
3. Yapılan işin test sunucusuna aktarılması mı ?
4. Hepsi
5. Hiçbiri

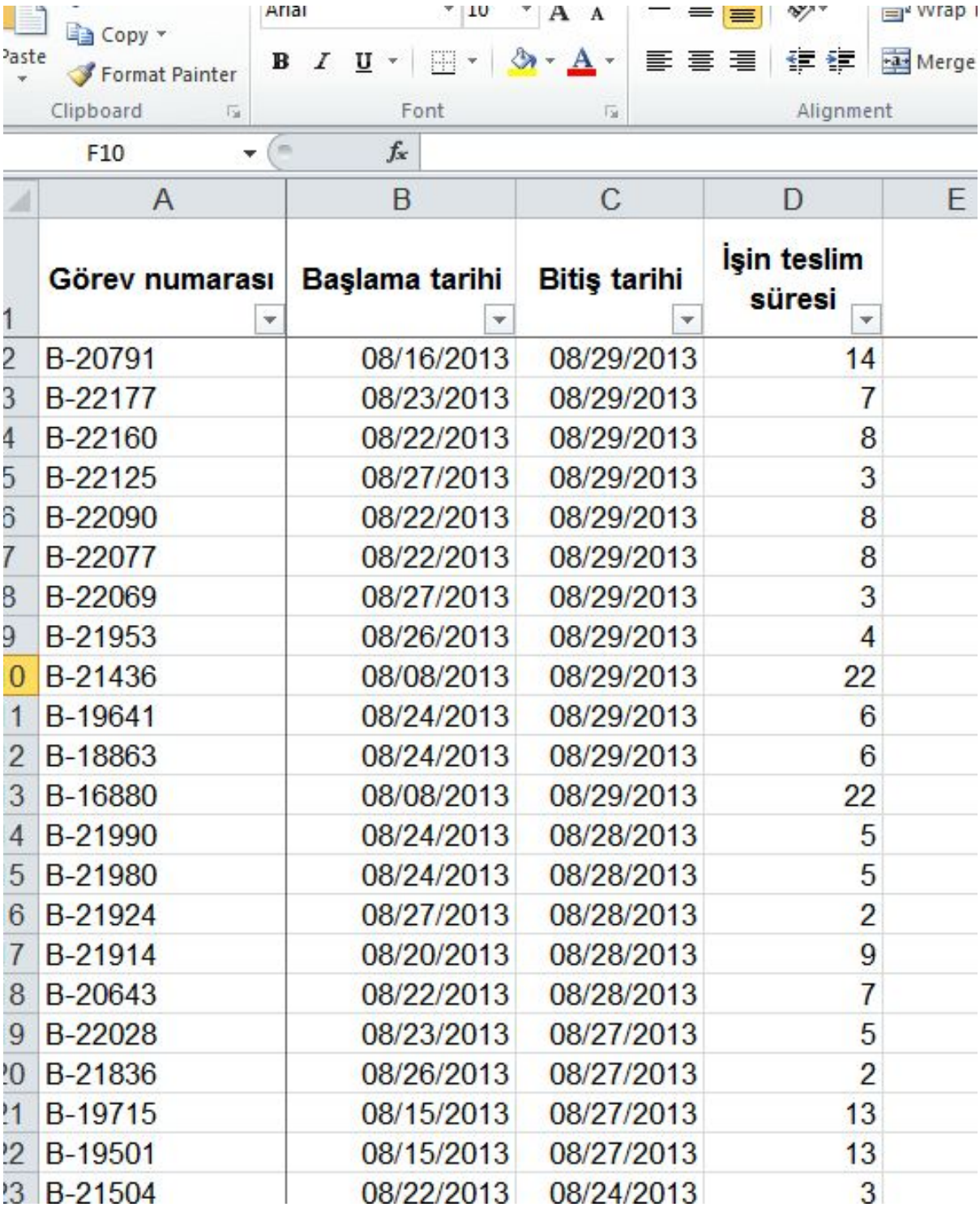
Her projenin kendi kuralları olabilir, bu yüzden burada net birşey söylemek yanlış olur.



Fakat benim kendi görüşüm ortalamada bir işin bitmesi ; test sunucu atılması ve teste hazır hale getirilmesi ile tamamlanır diyebilirim. Yapılan işin canlı ortama çıkması uzun zaman alabileceğinden dolayı, işin teslim süresini gereksiz yere artırabilir.

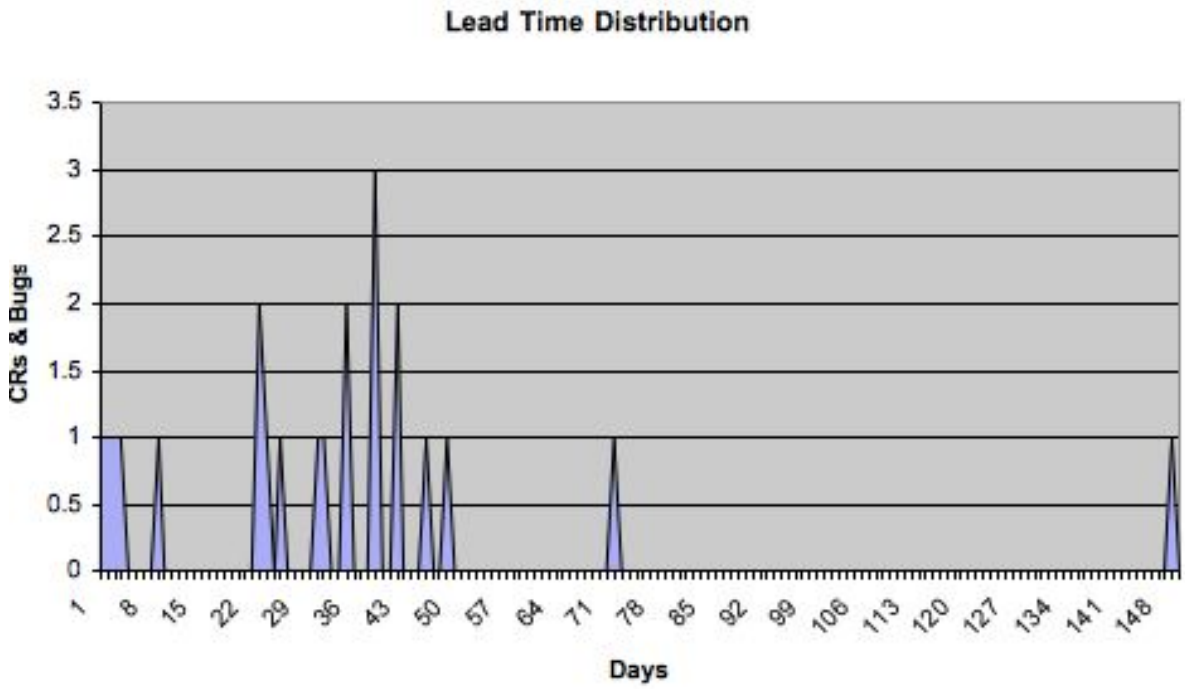
Teslim süresi dağılımı nedir ?

Projemizde 3 yazılımcı arkadaşın çalıştığını düşünelim, Yazılımcı -1 , Yazılımcı -2 ve Yazılımcı -3 ve bu kişilere verilen işlerin teslim sürelerinden bir dağlım oluşturabiliriz. Bu dağılıma işin teslim süresi dağılımı (lead time distribution) diyoruz.



	A	B	C	D	E
1	Görev numarası	Başlama tarihi	Bitiş tarihi	İşin teslim süresi	
2	B-20791	08/16/2013	08/29/2013	14	
3	B-22177	08/23/2013	08/29/2013	7	
4	B-22160	08/22/2013	08/29/2013	8	
5	B-22125	08/27/2013	08/29/2013	3	
6	B-22090	08/22/2013	08/29/2013	8	
7	B-22077	08/22/2013	08/29/2013	8	
8	B-22069	08/27/2013	08/29/2013	3	
9	B-21953	08/26/2013	08/29/2013	4	
10	B-21436	08/08/2013	08/29/2013	22	
11	B-19641	08/24/2013	08/29/2013	6	
12	B-18863	08/24/2013	08/29/2013	6	
13	B-16880	08/08/2013	08/29/2013	22	
14	B-21990	08/24/2013	08/28/2013	5	
15	B-21980	08/24/2013	08/28/2013	5	
16	B-21924	08/27/2013	08/28/2013	2	
17	B-21914	08/20/2013	08/28/2013	9	
18	B-20643	08/22/2013	08/28/2013	7	
19	B-22028	08/23/2013	08/27/2013	5	
20	B-21836	08/26/2013	08/27/2013	2	
21	B-19715	08/15/2013	08/27/2013	13	
22	B-19501	08/15/2013	08/27/2013	13	
23	B-21504	08/22/2013	08/24/2013	3	

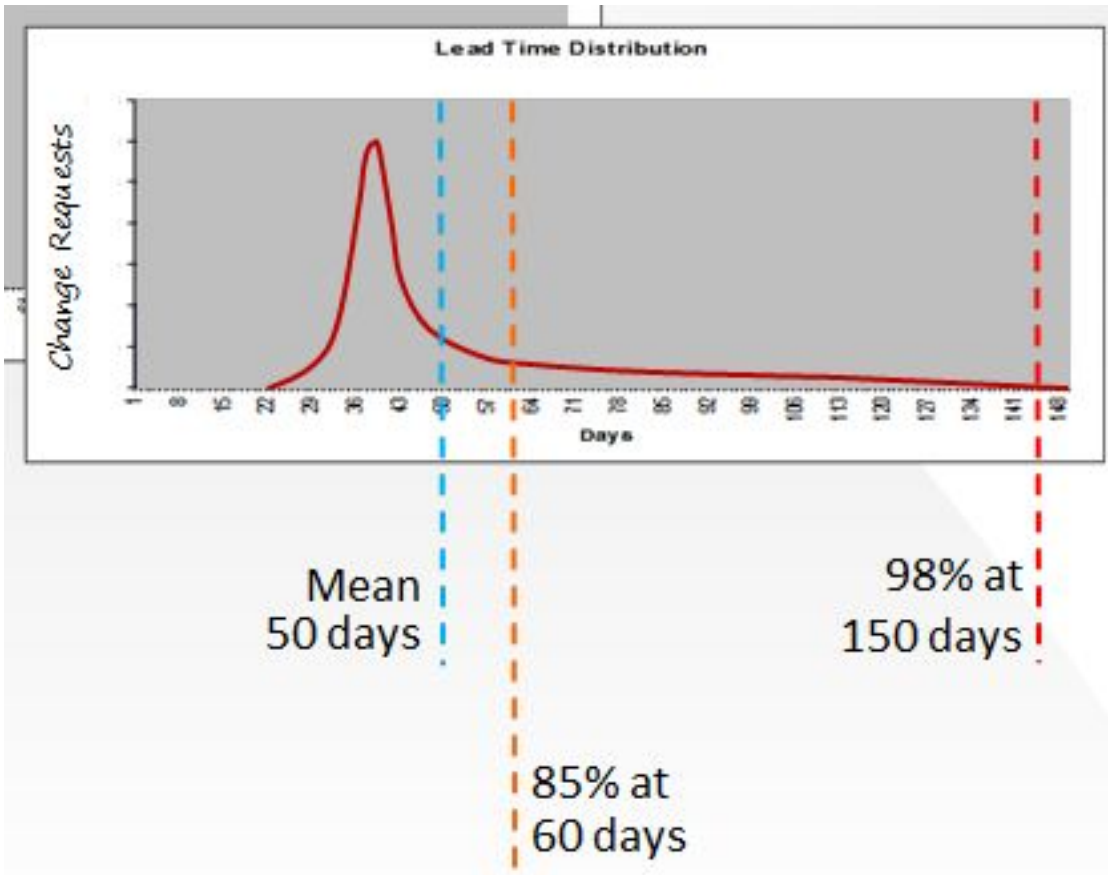
İlk kural projede yapılan işlerin takibini bir excel veya isteyeceğiniz herhangi bir yerde yapmanız. Daha sonra bu verilerle dağılımını kolayca oluşturabiliriz.



Bir sonra ki işin ne zaman biteceğini hesaplamanın yolu

Projenin belli bir aşamasında yeni bir iş geldi ve bize bu işin ne zaman biteceğini soruldu. Aynı Apgard skorlama standardı gibi bizde bir sonra işin istatistik kullanarak standart bir şekilde hesaplayarak söyleyebiliriz.

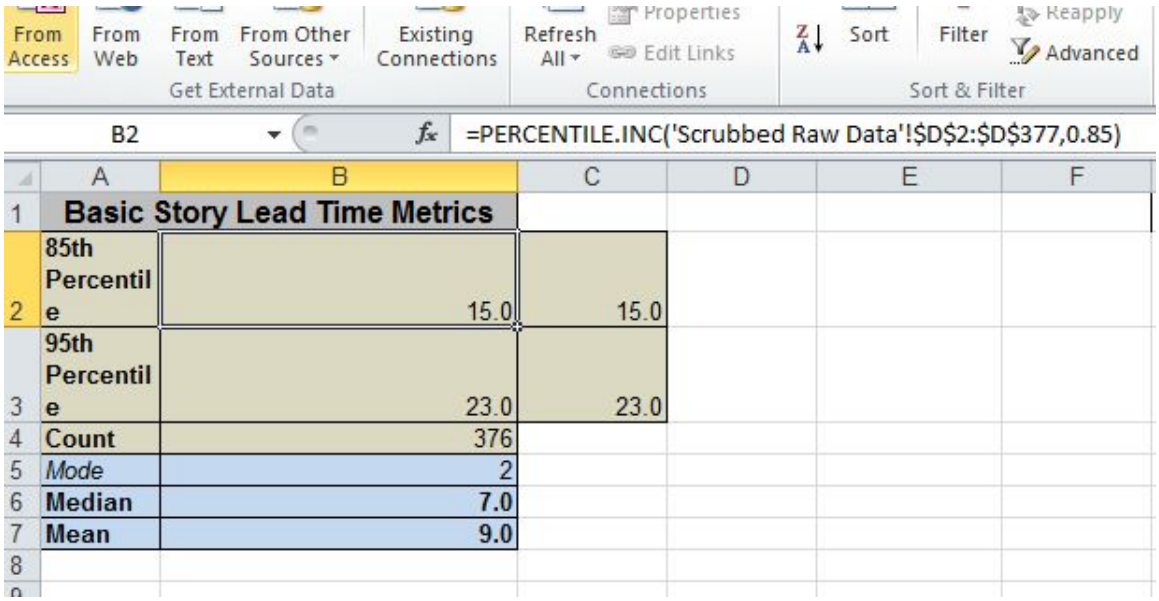
İşin teslim süresi dağılımı elde ettikten, bir sonraki işin ne zaman biteceği konusunda basit ve etkili standart elde etmiş oluruz.



Amacımız yeni işin olasılıklar dahilinde ne zaman biteceğini standart bir şekilde hesaplamak. Yani örneğin yukarıdaki büyük proje dağılımına göre konuşursak, artık yeni iş %50 ihtimalle 50 günde, %85 ihtimalle 60 günde veya % 98 ihtimalle 150 günde bitecektir diyebiliriz.

İlk yol

İlk olarak Microsoft excel içerisinde ki **Percentile.INC** fonksiyonunu kullanabilirsiniz.



Excel içerisinde yüzdesel aralığı hesaplamak için iki formül bulunuyor

1 – **Percentile.INC**

2 – **Percentile.EXC**

Veri kümenizin genişliğine ve azlığına göre istediğiniz formülü kullanabilirsiniz.

Dosyayı indirmek için : http://kodcu.com/wp/wp-content/uploads/2013/09/Metrics.xlsx

İkinci yol

Hakan Forss'un blogunda belirttiği hesaplama yöntemi. **Percentile.INC** ile karşılaştırıldığında sonuçlar çokta farklı olmadığını belirtmek isterim.

Dosyayı indirmek için : https://dl.dropbox.com/u/19243273/Histogram%20Example.xlsx

İş var iş var .

Elbette her iş aynı değil. Bu iş ne zaman biter ? sorusundan önce bu işin tipine göre ayırıp ona göre işin teslim süresi dağılımına göre tahminde bulunmak daha isabetli olacaktır.

David Anderson'a göre işleri gecikme maliyetine göre 4 kategoriye ayırabiliriz.

1 – Acil işler (expedite)

2 – Sabit tarihli işler (fixed date)

3 – Standart işler

4 – Bekletilebilir işler (Intangible)

İşte bu 4 kategoriye göre ayrı ayrı iş teslim süresi dağılımlarının (lead time distrubition) çıkartılması daha net tahminlerin yapılmasına sebebiyet verecektir.

Poker oyunu ile tahmin etmece artık bir zaman kaybı mı ?

İstatistiksel olarak bir sonra ki işin ne zaman biteceğini müşteriye söylemek mi daha gerçekçi, yoksa bir odaya kapanıp kapatıp bir sonraki işin(lerin) ne zaman biteceğini tahmin etmeye çalışmak mı ?

Bir çok projede zaman tahmini yapmak proje yöneticileri açısından artık zaman israfı (muda) olarak görülmektedir. Şu an dünyada proje yöneticileri arasında #noestimates twitter hash etiketi altında tahmin yapmanın ciddi maliyetlere sebebiyet verdiğine ait tartışmalar şiddetli biçimde devam etmektedir.

İşte bu israftan (muda) kaçınarak tahminleri istatistik zeminde cevaplamak hem daha gerçekçi hem de müşteri açısından daha güvenilir sonuçlar çıkmasına sebebiyet verecektir.

Bölüm 5

Projenizin yolunda olup olmadığını nasıl anlarsanız ?

Projenin gidişatı konusunda geçerli geri-bildirimler almak son derece önemlidir. Bu geri bildirimler sayesinde yapılan hataların farkına çabucak varabilir ve aksayan noktaları düzeltme şansı yakalayabiliriz.

Yazılım projelerinde en geçerli geri bildirim hiç şüphesiz ki müşterilerden gelmektedir ama iş müşteriye kadar uzamadan daha önce projenin dar boğazlarını görebilmek adına bu yazı da Toplu akış diyagramı (*Cumulative flow diagram*) konusundan bahsetmek istiyorum.

Geri-bildirimler bizlere sezinleme gücü verirler. Örneğin psikologlar yıllarca insanlarla çalıştıkları için zor insanla karşılaştıklarında karşı tarafın durumu hakkında hızlıca kavrayabilir ve olayı kontrol etmek için doğru adımlar atabilirler. Bu duruma Recognition-primed decision [1] adı veriliyor. Türkçesi “bilmeden bilmek” diye çevirirsem sanırım yanlış olmaz.

Aynı şekilde ilaç ve anestezi uzmanları çok iyi geri bildirimler aldıkları için verdikleri kararlar gittikçe daha hızlı ve doğru olmaktadır ve fakat bir radyolog veya patoloji uzmanı için geri bildirimler daha kısıtlı miktarlarda geldiği için sezinleme içgüdülerinin gelişimi, bir ilaç ve anestezi uzmanı kadar değildir.
Örneğin ameliyat sırasında bir anestezi uzmanı “ters giden bir şeyler hissediyorum” dediği zaman tüm ekip kötü senaryo için hemen hazırlanır çünkü anestezi uzmanının sezileri aldığı geri bildirimler sayesinde çok gelişmiştir. [2]
Peki biz projelerde ters giden noktaları sezinlemek için nasıl geri bildirim mekanizmaları kullanmalıyız ?

[1] http://en.wikipedia.org/wiki/Recognition_primed_decision

CFD (*Cumulative flow diagram*) nedir ?

Toplu akış diyagramı (*Cumulative flow diagram*) projenin akışında bir aksaklık olup olmadığı konusunda proje ekibine hemen görsel geri bildirimde bulunur. Madde madde özetlersek

- Tüm görevlerin bitip bitmediği bilgisini bize verir
- Dar boğazların nerelerde olduğunu söyler
- Ortama olarak işin tamamlama süresini ve işin limit dengesini gözler önüne serer

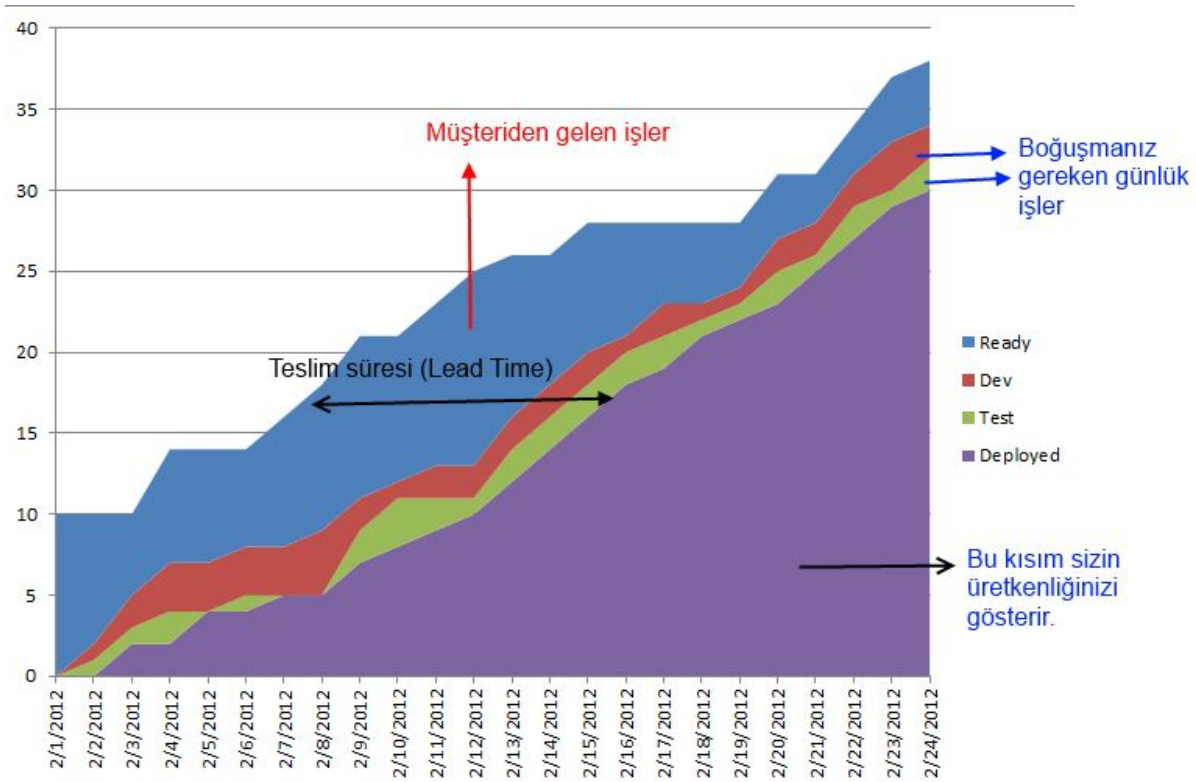
Burndown chart ve CFD arsındaki farklar ?

Burndown chart çevik sistemlerde sıkça kullanılan bir geri-bildirim grafiğidir. *Burndown chart* geriye kaç adet iş kaldı noktasına odaklanırken, toplu akış diyagramı (*Cumulative flow diagram*) işin akışında ki (gelen istek, biten istek) dengeyi gözlememiz de bize yardımcı olur. Örneğin iş bitirmede ne kadar iyiyiz ? Bu sorunun cevabını toplu akış diyagramı (*Cumulative flow diagram*) bize verebilir.

CFD nasıl okunmalı ?

İş akışının önemi Kanban metodunun kalbini oluşturduğu için *Cumulative flow diagram* Kanban ile takip edilen projelerde ayrı bir öneme sahiptir. Toplu akış diyagramına (*Cumulative flow diagram*) bir bakışta, proje sürecininde bir problem olup olmadığını hemen anlayabilirsiniz.

Örneğin müşterinin isteklerindeki artışı ve yapılan işlerin dengesini çok rahat bir şekilde görebilirsiniz. Aşağıdaki grafik bu dengeyi çok güzel bir göstermektedir. Not : Aşağıdaki grafik sağlıklı ilerleyen bir projeye aittir.



Nasıl CFD oluşturabiliriz (excel)

Bu adrese giderek örnek şablon bir toplu akış diyagramı (*Cumulative flow diagram*) indirebilirsiniz :http://ardalis.com/excel-cumulative-flow-diagram ,
Bu excel şablonunun avantajı çok sade ve anlaşılır olması.

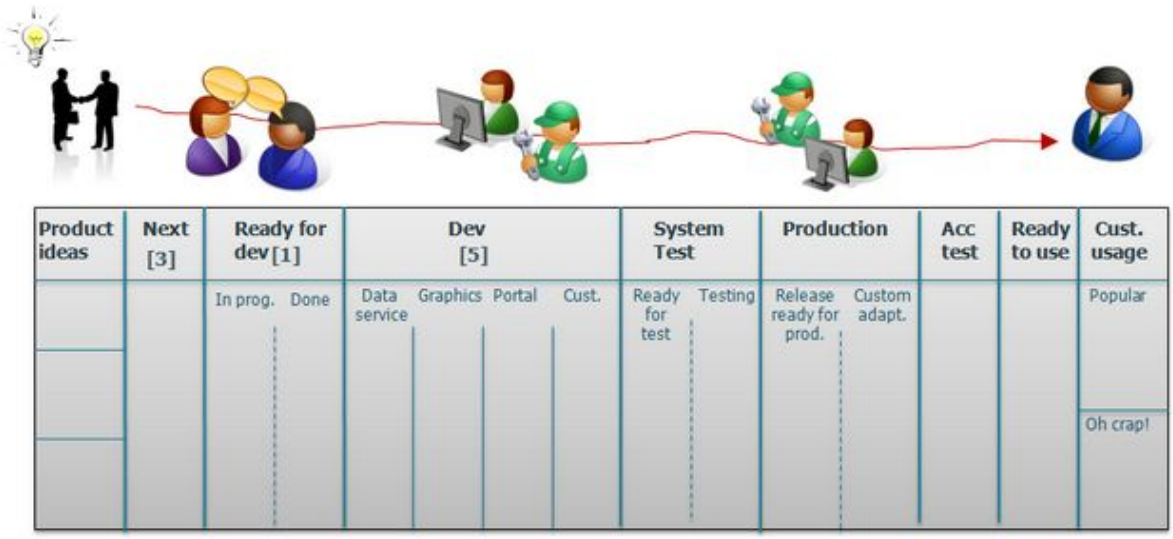
Şablon excel dosyasına kullanarak kendi proje süreçlerinize uygun Toplu akış diyagramı (*Cumulative flow diagram*) oluşturarak proje akışınızı ve dengesini rahatlıkla takip edebilirsiniz.

Elbette CFD diyagramı proje takibi için tek başına yeterli olmayabilir. Kontrol çizelgesi ve teslim süresi dağılımı (lead time distribution) diyagramları da projenin gidişatının takibi için son derece önemli olduğuna inanıyorum.

Bölüm 6

Darboğazlar nasıl aşılır ?

Seneler önce üniversiye giderken o “sözü” arkadaşımın yurt odasında ki bir takvimin üzerinde görmüştüm. Bu “sözün” çok kilit bir noktadan bahsettiğini hissetmiştim ama kavrayamamıştım. Yıllar sonra David Anderson’un Kanban eğitimde proje süreçlerinde ki darboğazlar konusunu tartışırken aklıma tekrardan bu söz geldi ve bu sefer bu sözün tam olarak neyi kastettiğini uygulama bir şekilde anlama fırsatı buldum. Bu sözün ne olduğuna geçmeden önce darboğaz kavramını sizlere biraz daha kapsamlı anlatmak isterim.



Projenin başarılı olmasındaki anahtar nokta akışın sağlıklı bir şekilde ilerlemesidir. Özellikle yazılım projelerinde müşteri gereksinimleri önceliklendirip vermeli ki ondan sonraki gelen , yazılım ve test ekibi süreçleri iyi bir şekilde gerçekleştirebilsin. Eğer müşteri gereksinimleri zamanında veremezse bu noktada bir darboğaz oluşur ve istenilen bir durum değildir.

Aynı şekilde yazılım ekibi olabildiğince hatasız ürün çıkartmalılar ki hatalardan dolayı hem yazılımcılar hemde test yapan kişiler de darboğaz oluşmasın. Darboğazlar projenin kaderi üzerinde son derece etkilidir.

Kanban yaklaşımı sayesinde bu darboğazları ortaya çıktıkları anda görüp müdahale etme şansı yakalayabiliriz. Nasıl mı ? Elbette WIP (Work-In-Progress) limitleri koyarak. Örneğin test grubunda 3 arkadaş çalışıyorsa ve aynı anda 3 iş tavan limitini getirerek (WIP) test grubundaki arkadaşların performansını korumuş oluyoruz. Kısacası test sürecine bir darboğaz oluşursa WIP sayesinde bu tıkanmayı direk anlarız. Böyle bir tıkınma yaşanırsa test sürecine daha fazla adam aktararak (örneğin analizi yapan kişilerinde test yapması gibi) darboğazı çözme şansını yakalayabiliriz.
Darboğazlar konusunda Mustafa Kemal Atatürk"ün tespiti beni çok etkilemişti. İşte o söz :

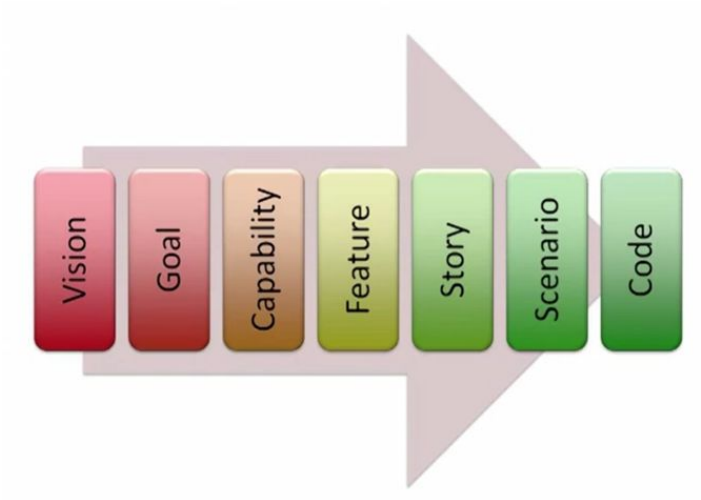
"Ben, bir işte nasıl başarılı olacağımı düşünmem; o işe neler engel olur, diye düşünürüm. Engelleri kaldırdım mı,iş kendi kendine yürür."

Proje yönetimi çerçevesinde darboğazların ne kadar ölümcül bir öneme sahip olduklarını ancak bu kadar net ortaya konabilir.

Bölüm 7

Yazılım geliştirme adımları ve Cynefin hakkında

Klasik anlamda yazılım geliştirme adımlarını aşağıdaki gibi tanımlayabilir miyiz ?



Zor gözükmüyor değil mi? Önce vizyon sonra hedefleri belirle, kapasite ve özellikleri çıkart ve sonra bunları hikayelere ve senaryolara böl ve kodla, işte bu !

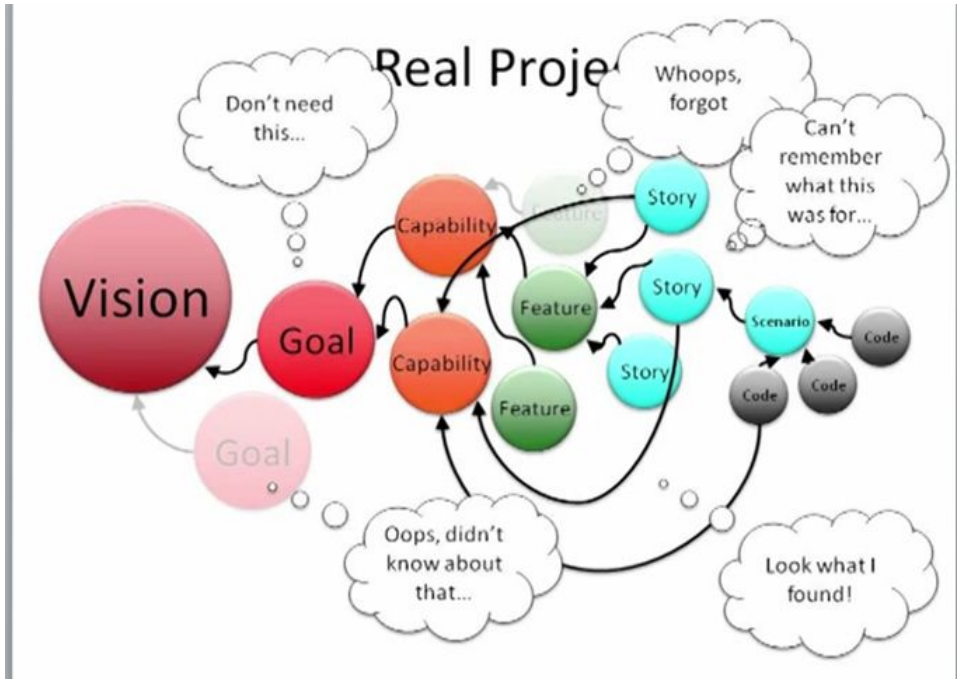
Neden evdeki hesap çarşıya uymaz ?

Sorunun cevabı karmaşıklıktır. Yazılım geliştirme süreci tek düze ve doğru bir şekilde ilerlemez. Aşağıdaki şekilde göstermeye çalıştığımız gibi birbirine içine geçmiş süreçler yumağı söz konusudur.

Peki yazılım projeleri neden karmaşık ?

Karmaşıklığın sebebi arayıştır. Yazılım projelerinde hedefler ve ona bağlı özellikler, özelliklere bağlı hikaye ve senaryolar her an değişebilir.

Birçok yazılım projesi aslında yeni bir keşiftir. Daha önce hiç denenmemiş bir şeyin gayretidir. Bu durumda doğal olarak fikir değişiklikleri olabilir. İşte bu adımlar yazılım projelerini karma karmaşık hale getirecektir.

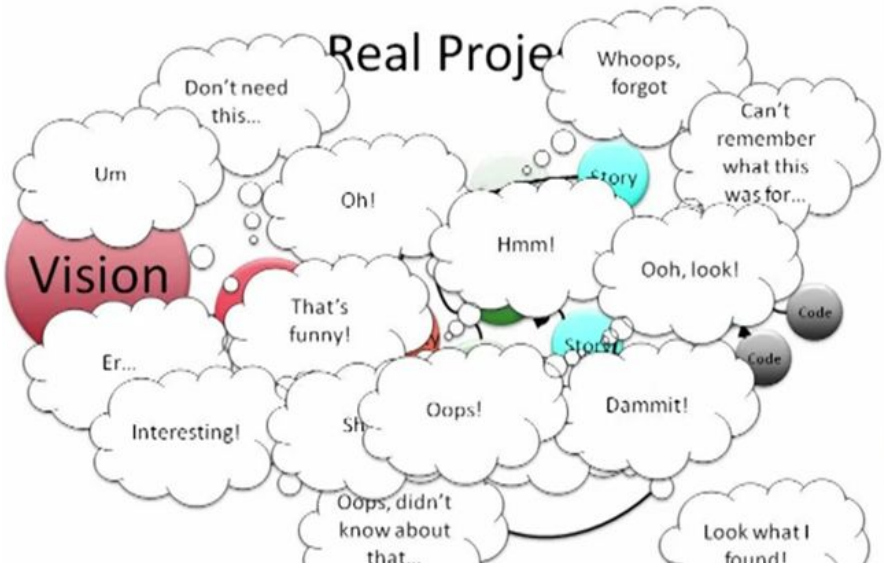


Sorumu değiştiriyorum; Yazılım projeleri neden karma karmaşık ?

Özelikle başlangıç seviyesi (start-up) firmalarda proje gelişirim süreçleri işler iyice çığırından çıkabilir -ki bu normaldir çünkü firma para kazanma modelini oturtma aşamasındadır. Bu aşama şiddetli depremlerin yaşandığı bir çağdır.

Örneğin Youtube'un ilk olarak ortaya arkadaşlık sitesi olarak ortaya çıktığını söylesem sanırım bu değişim depremlerinin ne derece büyük olduğunu hayal edebilirsiniz.

Kurumsal firmaların mevcut süreçlerinin iyileştirilmesi (kaizen) aşamalarında da işler çığırından çıkabilir. Birçok yazılım projesinde mevcut durum karma karışıklığa doğru gitmeye meyillidir. Çoğu yazılım projesi yeni bir keşiftir bu ve süreci yönetmek sanıldığı kadar basit değildir.

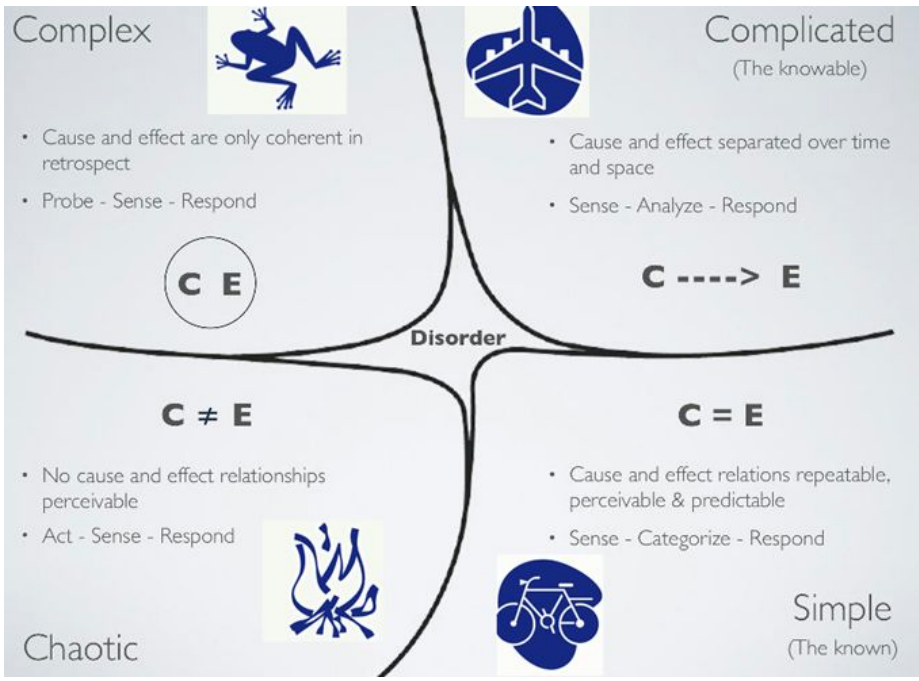


Yazılım projelerini yönetmek; vizyonunu, hedeflerini ve alt kırımlarını çıkartıp, bu süreci ilerletmek zannedilenden zordur çünkü hedefler hareketlidir ve bu da karma karışıklığı beraberinde getirir.

Projeniz hangi kategoride ?

Yazılım projelerinde hangi yönetim metodolojisini kullanacağınıza karar vermek için Cynefin sisteminden faydalanabilirsiniz.

Cynefin problem çözerken size hangi adımların atılması gerektiği noktasında yol gösterebilir.



Cynefin göre bir olayın / problemin 5 karakteristiği vardır.

1– **Basit ve bilinen konular / problemler (sağ alt)** : Örneğin bisiklete binmek gibi, aradaki sebep sonuç ilişkileri net ve herkes tarafından görünür konular. Böyle bir konu ile karşı karşı karşıya kalmışsanız, atılması gereken adımlar şöyledir : görünen ilişkiler arasında noktaları birleştirin (sense), bunları kategoriler ayırın ve yapın.

2 – **Karışık ve bilinebilir konular / problemler (sağ üst)** : Örneğin uçak kullanmak gibi, aradaki sebep sonuç ilişkisi sadece eğitimli uzmanlar tarafından görünür konular. Ya da bir saat tamircisini düşünün, bozuk bir saati parçalara ayırıp sonradan bunları rahat bir şekilde tamir edip, toplayıp size teslim edebilir. Böyle bir konu ile karşı karşı karşıya kalmışsanız, atılması gereken adımlar şöyledir : görünen ilişkiler arasında noktaları birleştirin (sense), analiz edin (uzmanlar için) ve yapın.

3 – **Karma karışık konular / problemler (sol üst)** : Bu kısma ait en meşhur örnek kurbağa örneğidir. Şöyle ki : Yukarıda ki saat tamircisinin yaptığı gibi, bir kurbağayı parçalara ayırıp, tekrardan birleştirebilir misiniz ? Kurbağa son derece dinamik ve

yaşayan alt sistemlerden oluşur, yani konu karma karışıktır. Bu yüzden tekrardan birleştirme imkansızdır.
Böyle bir konu ile karşı karşı karşıya kalmışsanız, atılması gereken adımlar şöyledir : noktaların belirmesini bekleyin ortaya çıkan ilişkiler arasında noktaları birleştirin (sense), yapın.

Yazılım projelerinin çoğu bu kategoriye girer, istekler dinamiktir ve ortam her an değişebilir.
4 – **Kaos durumda (sol alt)** ise düşünecek veya plan yapacak zaman yoktur, ev yanıyordur ve evden bir an önce çıkmanız gerekir. Böyle bir konu ile karşı karşı karşıya kalmışsanız, atılması gereken adımlar şöyledir : hemen harekete geçin, ortaya çıkan ilişkiler arasında noktaları birleştirin (sense), karşı önlem alın.

5 – Henüz karar verilememiş alan (orta alan – **disorder**)

Yazılım projeleri , Cynefin ve metodolojiler.

Projeniz için gereksinimler çok açıksa yani projeniz bilindik alanda ise (sağ alt) o zaman projenizi yönetimi için waterfall yaklaşımını tercih edilebilir. Yani tüm projeler için çevik (agile) yaklaşımlar kullanmak zorunda değilseniz.
Eğer yazılım projeniz karışık alanda ise (sağ üst) o zaman waterfall gibi geleneksel proje yönetim yaklaşımları size yardımcı olmayabilir çünkü artık karışık alanda bulunuyorsunuz. İşte bu alanda çevik (agile) yöntemler size son derece faydalı olabilir. örneğin : Extreme Programmng (XP), Scrum vb…
Fakat eğer projeniz karma karışık (complex) alanda ise -ki yazılım projelerinin çoğu bu alanda bulunur, diğer alanlarda kullandığınız proje yönetim yaklaşımları işe yaramaz. İşte bu noktada Kanban yönetimi işte bu toptan değişim kültürüne geçişin ilk adımıdır.

Kanban bu karma karmaşıklığa çözüm olabilir ?

Kanban sisteminde katı kurallar yoktur. Dışarıdan gelen bir danışmanın “Siz herşeyi yanlış yapıyorsunuz , çekilin” demesi pek gerçekçi değildir.

Problemi çözümü yine kurumun içinde ki insanların çözmesi gerektiğini savunur ve bu yüzden kurum içi liderliği ortaya çıkartmak Kanban sisteminde ana prensiplerinden biridir.
Kanban sistemini aşağıdaki adımları uygulayarak hemen başlayabilirsiniz.

Kanban sisteminin prensipleri

- Her ne yapıyorsanız onu yapmaya devam edin.
- Değişimin zaman alan evrimsel bir süreç olduğunu kabul edin.
- Şirket içinde rolleri saygı duyun ve değiştirmeye kalkmayın
- Her seviye de liderliği ön plana çıkartın

Kanban sistemin pratikleri

- Görselleştirin
- İşinizi limit koyun
- Akışı yönetin
- Politikaları açık hale getirin
- Geri bildirim döngü kanallarını açık tutun
- Bilimsel yöntemlerle iş birlikteliğini arttırın.

Bölüm 8

Kabile savaşları ve düşünce gücü

Tribal leadership kitabını okumanızı öneririm, organizasyonları ve bu organizasyonların içerisindeki grupları kabilelere benzeten kitap, özetle her kabilenin kendisine ait bir karakteri ve seviyesi olduğunu söyler. İşte bu seviyeler

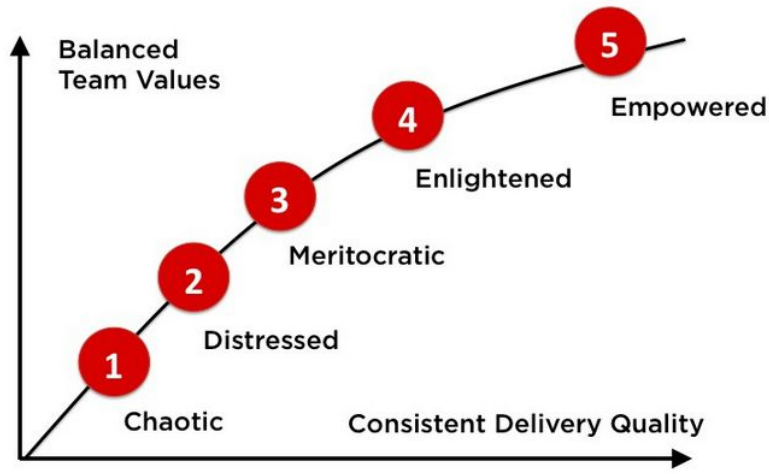
Seviye 1 – Bu dünya berbat bir yer – Kaos kültürü hakim. Kimseye güvenilmez.

Seviye 2 – Hayatım berbat – organizasyon içindeki iletişim neredeyse yok, heyecan yok

Seviye 3 – Ben iyiyim, sen değilsin – bireysellik ön planda (Meritokrasi)

Seviye 4 – Biz iyiyiz onlar kötü – takım ruhu iyi, iletişim iyi

Seviye 5 – Tam bir takım ve bütünlük ruhu – Chicago Bulls takımının 1995 deki hali gibi düşünebilirsiniz. Tam bir kenetlenme

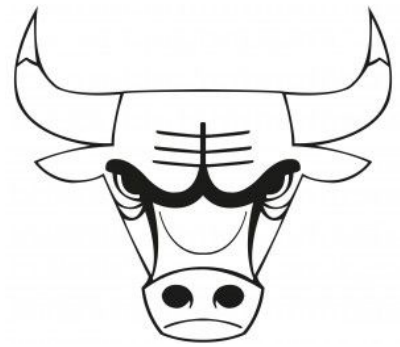


Dünyanın en iyi en modern proje metodolojilerinden birini uygulayın (bkz Lean Kanban gibi) ama eğer ortada **4 seviye** bir takım yoksa ortaya iyi ve uzun vadeli bir

başarının çıkması zordur. Elbette 4. seviye bir takım oluşturmak zaman alıcı ve emek isteyen bir süreç olduğunu belirtmeme gerek yok.
Kurulan sistemin takım oyuncuları üzerinde çok fazla kısıt koyması, örneğin sen product owner rolündesin, sen uygulama geliştirici rolündesin gibi. iyi sonuçlar getirmez.

Akışa odaklanmak yerine rollere odaklanıp kısıtlayıcılık getirmek takımın pozisyonunu **3 seviyelerinde** tutacaktır.

Yani bireyselcilik artacaktır. Ayrıca takım oyuncularının ortaya konan herhangi bir kısıtlayıcı sistemi kırmak için boşu boşuna zaman harcayacaklarını da hatırlatırım. Bu arada yanlış anlaşılmasın roller önemlidir ama her kabilenin kendi kültürü vardır. Bu içsel değerler ve kültür dikkate alınarak roller belirlenmelidir. Yoksa dışarıdan yapay olarak bazı süreçler dikte edilirse bu ancak kısa vadeli sonuç getirecektir.



Başarılı bir koçluğun sırrı, bence; takım oyuncularına oynayabilecekleri boş alanlar sunmak (örneğin yetki vermek gibi) ve bu alanlarda hata yapmalarına izin vermektir (her hatanın bir ilerleme ve iyileştirme -kaizen- getireceğini unutmayalım lütfen).

Acaba şu metodolojiyi iyi uyguluyor muyuz ? Roller tamam mı ? gibi taktiksel konular hakkında kaygılanmak yerine takım oyuncularının bir bütünlük içinde hareket edip etmediklerine dikkat etmemiz, projenin sağlığı açısından daha doğru olacaktır.

Son olarak şunu belirtmek isterim ; İyi bir takımda superstar oyuncu yoktur, ekip çalışması vardır, hayatı cennet kılan sistemler vardır, süreçler vardır. Seviye 4 ve

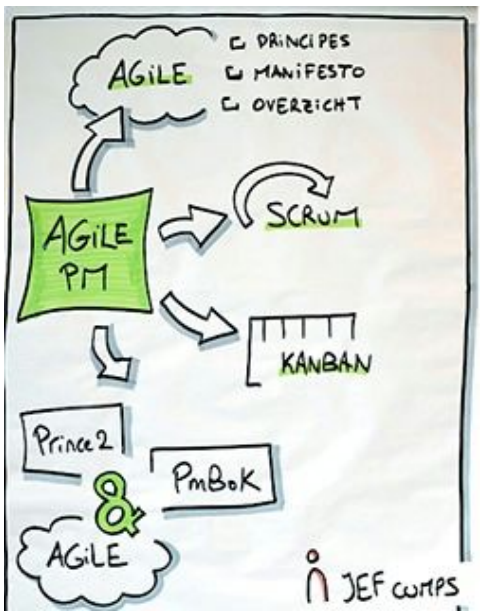
5’den bahsediyorum. Gerçek yıldız oyuncu arkadaşlarını en fazla katkıyı veren kişidir.

Bölüm 9

İnsanlar mı suçludur yoksa sistemler mi ?

Verimsiz bir organizasyon hasta ve fonksiyonlarını hakkıyla getiremeyen organizasyondur. Verimsizliğin en büyük sebeplerinden biri de israftır. Kaynakları israf etmek kötüdür ama göz görmeyince gönül de katlanıyor. İşte tamda bu yüzden süreçleri ve işleri görselleştirmek, israfın engellemek için atacağımız en önemli adımlardan bir tanesidir.

Görselleştirmeden kasttım renklendirme bile olabilir. Elle çizim bile bu noktada yeterlidir. Amaç karşı tarafa süreci en iyi şekilde anlatabilmektir. Şunu da rahatlıkla söylemek istiyorum artık dünya her konuda görselleştirmeye gitmektedir. Kanban eğitimlerimizin de bir parçası olan bu yaklaşımı sizlere de tavsiye ederiz.



Toyota ile tanışmam 2010 yılına dayanıyor. Toyota Way yani diğer adıyla Lean (yalın) kavramlarını somut olarak ilk orada gördüm. Yalın (Lean) yaklaşımı tamamen israfın yok edilmesi üzerine kuruludur desem sanırım yanlış olmaz.

Neden israf Toyota için bu kadar önemli ? Peki bir organizasyon kaynaklarını israf ediyorsa, yani verimsiz kullanıyorsa ne olur ki ? Herhalde hemen batmaz. Sonuçta hangi firma %100 verimli ki ?

Hücresel boyut

Olay başka bir metafordan bakalım : Kendi hücrelerimizin çalışma prensibinden…

Normal bir hücre ile normal-olmayan hücreyi ayıran şey verimliliktir. Yani 1 sağlıklı hücre 1 molekül glikozdan 32 ATP üretirken, normal-olmayan hücre 1 molekül glikozdan 2 ATP üretir.

Normal-olmayan hücre son derece verimsiz çalışarak kaynakları boşuna aşırı şekilde israf eder ve ortaya bir fayda koyamaz. Anormal hücreler sistemden yok edilmezse ciddi hastalıklar baş gösterebilir (bkz : kanser – Kaynak : Prof. Dr. Bülent Berkarda .)

Özetle :
İsraf (verimsizlik) bozulmuş sistemin habercisidir, bozulmuş sistemler uzun süre ayakta kalamazlar.

İnsanlar suçlu değildir, sistemler suçludur

Eğitimlerin bir tanesinde katılımcılardan biri şöyle dedi :
“Hocam şirket içerisinde 5 tane pürüz adam var, onları gönderirsek gayet yalın oluruz. İsrafın kökünü keseriz. “

Bende kendisine o kişiler gitse bile şirketin kültüründen (yada sistemsizliği demek daha doğru) dolayı onlara benzeyen kişilerin çok geçmeden aynı pozisyonlara geleceğini söyledim çünkü sorun o kişilerde değil şirketin kültüründe olduğunu ifade ettim. Yani şirket kültürü insanları bir canavara dönüştürüyorsa insanlar işten çıkartmak ne işe yarar ki ? Sürekli yeni canavarlar sistem tarafından zaten oluşturulacaktır. Bu noktada W. Edwards Deming sözü hatırlatmak isterim
“ İnsanlar suçlu değildir, Sistemler suçludur” – Deming

Buna karşılık çeviklik adı altında birçok ilaç tedavisi günümüzde uygulanmaktıdır. Bu noktada hastalık yoktur, hasta vardır.

yaklaşımını benimsiyorum. Her organizasyon çevik olmak zorunda mı ? Çeviklik her zaman iyi midir ? Yada çeviklik anlayışı şirketin her birimi tarafından uygulanabilir mi ? Bu soruların cevapları bu yazının kapsamı dışında ama bildiğim tek gerçek her ilaç her bünyede aynı sonucu vermediğidir.

Bölüm 10

Çevik (Agile) yöntemler şirketlerin hastalığına iyi gelir mi ?

Theory of constraints kitabının yazarı Eliyahu M. Goldratt ifade ettiği çok güzel bir söz vardır.

Ancak evli olmayanlar duygusal dirençleri mantıksal yaklaşımla aşacaklarını zannederler. Duygusal dirençleri kırmak için daha güçlü duygusallıklar gerekmektedir.

Yani şirket organizasyonu yalın veya çevik olmak istemiyorsa ve şiddetli bir duygusal direnç varsa o zaman yapılması gereken öncelikle kişilerin korkuları ve endişelerini anlamanız ve ona göre hareket etmenizdir. Şu firma şunu kullanıyor bizde geçelim hevesiyle alınmış kararlar maalesef hüsran ile sonuçlanabilir.

Kanban veya Scrum sadece birer makaniktir. Kalite felsefesi, yalınlık, sürekli iyileşme gibi yaklaşımlar benimsemeden, bütünsel bir iyileşmenin sağlanması imkansızdır.

Olaya muhasebe departmanı açısından bakalım ?

Çevik (Agile) yöntemler daha çok IT ekiplerini hedef aldıkları aldıkları için organizasyonun tümünü yayılması düşük bir olasılıktır. Şimdi aşağıdaki resme bakın ve bu yazının şirketin muhasebe departmanının duvarında olduğunu hayal edin. Muhasebe departmanında çalışanlar için Agile manifesto birşey ifade edebilir mi ?

Agile yöntem IT için süper bir yaklaşım olabilir ama eksiktir. Verimsizlik mikrobu organizasyonun her yerinde olabilir. Sadece mutlu IT elemanlarıyla bir şirket ne kadar uzun yaşayabilir ?

Scrum mi Kanban mı ?

Verimsiz çalışan (yani hastalıklı) takımların veya bireylerin iç dengesini toparlamak için değişik ilaç tedavileri uygulanabilir. Scrum ‘a ben bu noktada yan etkisi çok fazla olan bir ilaç gözüyle bakıyorum. Neden yan etkisi fazladır ? Şahsi fikrim en başta organizasyon kültürünü darmadağan etmesidir.

Aynı şekilde Kanban'da bu noktada bir ilaç tedavisi gözüyle bakılabilir ama yan etkisi Scrum'a göre daha azdır. Ya da iki ilacı karıştırıp da uygulayabilirsiniz : Scrumban gibi.

Şelale (Waterfall) metodu da piyasada hiç azımsanmayacak derecede kullanılmaktadır. Şelale (Waterfall) metodu eski olabilir ama yerine göre çok iyi sonuçlar halen vermektedir.

Semptomatik tedavi

Sadece popülizm ile Kanban veya Scrum'a kullanmak, hastalığın semtomlarını gidermekten öteye gidemez. Olaya bütünsel bakıp kök sebepleri yok etmemiz gerekir. Elbette bir yerden başlamak güzeldir ama sadece körü körüne bir metodolojiye bağlı kalmak en büyük hatadır.

Üst yönetimi yeni ikna ettik başka birşey uygulayamayız…

Bu noktada enteresan bir hikayemi paylaşmak istiyorum. Bir firma ile metodolojiler üzerinde konuşurken bana sıkıntılarından bahsettiler. Bende Kanban mekaniklerini değişik takımlar üzerinden hızlıca ve başarıyla uygulayabilirsiniz, rahat bir yaklaşımdır demeye kalmadı, karşı taraf bana şöyle dedi :

Üst yönetimi Scrum'a yeni ikna ettik, şimdi Kanban dersek işleri bahvederiz. ??,!! ??.

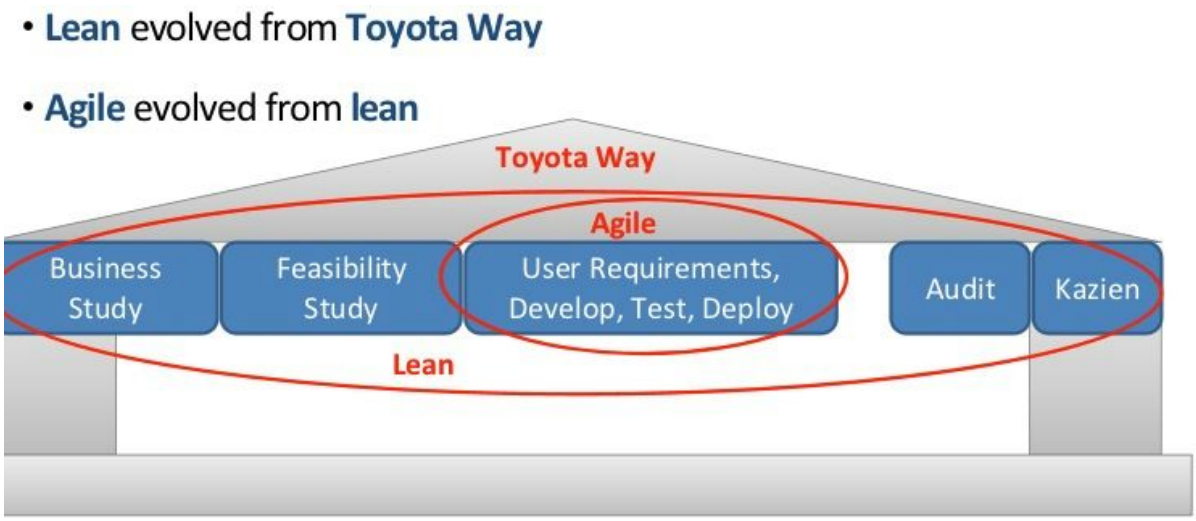
İki önemli noktayı belirtmek istiyorum; Ortalama Yalın (Lean) dönüşüm süreleri 5 ile 7 sene arasındadır. 5 sene sonra acaba hangi kaç firma Scrum kullanmaya devam edecek merak ediyorum.

Diğer bir nokta ise eğer organizasyonunuza kısa vadede aşırı baskı yaparak çevik olmasında israr ederseniz, bu organizasyon tarafında geri tepilme riskini doğurabilir. Fifth disciplineyaklaşımına göre

Prensip 1 : Sisteme ne kadar sert girersen, sistemde sana o kadar sert geri tepki gösterir.

Çevikliğin ötesinde Yalınlık (Lean)

Ben probleme daha bütünsellik içinde bakılması taraftarıyım. Çevik sistemler güzeldir ama organizasyon bakış açısına göre yetersiz olduğunu düşünüyorum. Çoğu şirket hasta ve bu hastalık değişik departmanlarda değişik formlarda devam ediyor. Sistemsizlik insanları hergün canından bezdiriyor.



Mehmet Tunç & James Hartley

Çevikliğin ötesinde organizasyonların yalınlaşmaya geçmelerinin orta ve uzun vadede en iyi sonuçlar getireceğine inanıyorum. Problemleri kendi kendilerine çözen israfsız ve yalın sistemler kurmak dileğiyle.

Bölüm 11

Yazılım projelerinde dikkat edilmesi gereken 5 risk noktası

Tom DeMarco ve Timothy Lister'ın yazdığı Waltzing With Bears: Managing Risk on Software Projects (Ayılarla dans/vals) kitabını okumanızı tavsiye ederim. Kitap içerisinde aklımda kalan en önemli sözlerden bir tanesi şöyle :

Riskini yönetmezsen krizi yönetmek zorunda kalırsın

Bu noktaya katılmamak sanırım mümkün değil. Kitap içerisinde dikkatimi çeken diğer bir nokta ise yazılım projelerinde dikkat edilmesi gereken en önemli 5 risk noktası konusu oldu, bu noktaları özetlersem :

1. Proje zamanlama hatası (inherent schedule flaw)
2. Gereksiz fazla gereksinim kabusu (requirement inflation)
3. Çalışanlar işten ayrılması (employee turnover)
4. Belirtimlerin analizinde oluşan ölümcül hatalar (Specification breakdown)
5. Verimsizlik (poor productivity)

- **Proje zamanlama hatası (inherent schedule flaw)**

Proje ne zaman biter ? Bu cevaplaması en zor sorulardan bir tanesidir. Eğer bu sorunun cevabı mevsimsel veya müşterinin istediğine göre verilirse proje zamanlama hatası oluşmuş olur.

Mevsimsel proje bitim süreleri vermek ülkemizde çok meşhurdur. Örneğin kış aylarındaysak, ilkbahara doğru proje bitsin diye anlamsız tahminlemeler ne yazık ki olmaktadır.

Müşteri baskısıyla oluşan proje zamanlama hatasında bir o kadar ölümcüldür. Kitap içinden bir örnek çok çarpıcıdır . Hikaye şöyledir :

Projenin sahibi olan zat gelir ve IT ekibine projeye 2 ay geç başladıkları için çatar ve der ki : Her geçen ay 110.000 dolar kaybediyorum. Projeye acil başlamanız ve şu zamanda bitirmeniz gerekir diye.

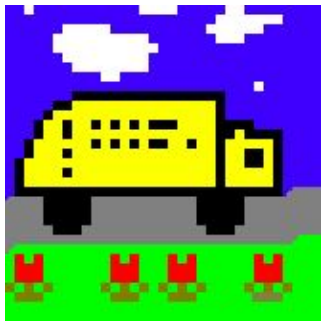
Bunun üzerine Tom DeMarco (kitabın yazarından birisi) şöyle der : Eğer projeyi 1 ay önce teslim etseydik 110.000 dolar kazancın mı olacaktı ? Projenin sahibi olan zat "evet" der.

Peki der Tom DeMarco ve ekler : Projeyi 10 ay önce sana teslim etseydik 110.000 dolar x 10 ay = 1.100.000 dolar mı kar edecektin diye sorusunu derinleştirir. Projenin sahibi olan zat biraz afallar ama yine tavrını bozmaz "evet" der.

Tom DeMarco, projenin sahibi olan zata döner ve son vuruşu yapar : Kusura bakmayın projenin IT yazılım ekibine gelene kadar zaten epey bir gecikmiş. Eğer proje bize 18 ay önce gelseydi şu an çoktan bitmişti.

Proje tahminleme hatasının üstesinden gelebilmek için Kanban yöntemi ile birlikte öne çıkanLittle's Law, Lead time distribution ve Monte Carlo yöntemleri tercih edilmesinde fayda olduğunu düşünüyorum. Her şeyden önce bilimsel ilerlemek zorundayız. Tahminlerimizi gerçekçi ve bilimsel bir temele oturtmadığımız sürece, proje takımlarının üzerinden zaman baskısını kaldırmamız imkansızdır.

- ## Gereksiz fazla gereksinim kabusu (requirement inflation)



Pareto ilkesi %80 – %20 kuralını söyler. Yani Pareto ilkesini yazılım projeleri üzerine uygularsak, kullanılan özellik oranı %20'dir. Yani %80 kullanılMayan isteklerle dolu olduğunu görürüz.

Bu riski azalmanın yollarından biri Lean startup yaklaşımını kullanmak olabilir. Kullanılabilir en küçük ürün – minimum viable product (MVP) yaklaşımı bizi bu israftan (muda) kurtarabilir.

- ## Çalışanlar işten ayrılması (employee turnover)

Bu riski sıfıra indirmek imkansızdır. Google için bile çalışanların işten ayrılması en büyük sorunlardan bir tanesidir. Bu riski azaltmak için bilginin tek kişide toplanmasını olabildiğince azaltmak gerekir.

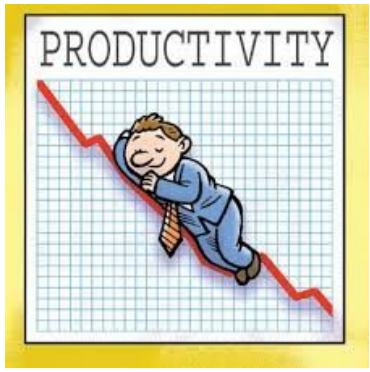
Kritik bilgi seviyesinde olan çalışanlar için bilgilerini videoya ders anlatır gibi son derece görsel bir şekilde aktarmaları son derece iyi bir yol olduğunu düşünüyorum. Ayrıca şirket için bilgi havuzları oluşturularak, bu havuzun güncelliği ve senkronizasyonu kontrol edilmesinde fayda vardır.

- **Belirtimlerin analizinde oluşan ölümcül hatalar (Specification breakdown)**

Bu çok enteresan bir risk türüdür. Belirtimlerin analizinde oluşan ölümcül hatalarda (Specification breakdown) proje bir anda sona erer. Misketlerini al ve git durumu. Bu risk eğer şelale (waterfall) türü bir metodoloji ile ilerleniyorsa başınıza gelebilir.

Örneğin gereksinimleri aldınız ve projeye başladınız. 6 ay sonra proje tamam dediniz ama bir baktınız ki müşterinin sunucuları hep Linux, siz ise uygulamayı .NET ‘de geliştirmiştiniz. Kuralları gereği de Windows sunucusu kullanmak yasak. Buyrun buradan yakın. Çevik süreçlerde bu tür bir riskin gerçekleşmesi çok düşük bir olasılıktır.

- **Verimsizlik (poor productivity)**



Verimsizlik (Muda) üstesinden gelinmesi gereken en önemli risklerden bir tanesidir. Çevik süreçler bu riskin üstesinden gelmek için iyi bir yöntemmiş gibi gözükse de, verimsizliliğin kaynağı çok daha derinlerde olabilir. Bu derinlikte problemlere salt çevik yöntemlerle uygulayarak başarılı olmak zordur. Çevik (Agile) yöntemlerin yalın (Lean) düşünceyle uygulanmasında son derece fayda vardır.

En iyi Agile (çevik) yaklaşımı olan Kanban metoduyla verimi ölçmek için öncelikle müşterinin gözünden bakmak şarttır. Müşteri gözünden en önemli KPI nedir ? Tabii ki ürünün teslim süresi (Lead Time).

Şöyle düşünün, acil kahve içmeniz gerekiyor, dükkana girdiniz ve siparişi beklemeye başladınız ve kahve bir türlü gelmiyor. Geçen süre artık (Lead Time) artık canınıza tak ettiğinde kasiyere sebebini sorarsınız.

Yazılım projeleri de çok farklı değil, müşteri siparişi en kısa sürede görmek ister bu yüzden Kanban metodunda dikkate alınan ve takip edilen en önemli gösterge işin teslim süresidir (Lead time) .

Bölüm 12

Siyasette 24 saat çok uzun zaman, peki ya Yazılım sektöründe durum nasıl ?

Geçenlerde nasıl olduysa konu konuyu açtı ve Süleyman Demirel'in efsane olmuş sözünün, yazılım sektörüne ne kadar uygun olduğunu konuştuk.

"24 saat siyasette çok uzun bir zamandır"

Siyasette 24 saat uzun bir süre, peki durum yazılım sektöründe nasıl ?
Hemen aklıma şu nokta geliyor :

Bu kaotik (tahmin etmesi zor) piyasa koşullarında : **"2 hafta biz ekip olarak odaya kapanıp proje geliştireceğiz " lüksüne çok az organizasyon sahiptir.**

Kısaca Çeviklik (Agile) ne değildir ?

Piyasadaki kavram kargaşasının farkındayım. Gelin adım adım gidelim. Sanıldığının aksine çeviklik hızlı olmak değildir. Hele hele belge yazmamak, karakucak proje geliştirme hiç değildir.

Çeviklik riski azaltmak için kullanılan bir yaklaşımdır. 1 milyon TL batmasında, 50.000 TL batsın diyeceğiniz durumlarda çevik yöntemlerle ilerlemek gayet mantıklıdır. Çeviklik riski azaltmakla ilgilidir. Önünüzü göremediğiniz ortamlarda çevik gitmek sizi büyük kayıplardan korur.

Süreçlerin kendini ispatladığı bir ortamda çevik bir şekilde ilerlemek doğru değildir. Bu durumda Şelale (Waterfall) metodu ile gitmek getiriyi (ROI – Return of investment) artıracaktır çünkü belirsizlik çok azdır.

Scrum Türkiye piyasaları için uygun değildir.

Bir firmanın nihai hedefi nakit akışını yükseltmek, net karı artırmaktır. Esas sorun bu değişken ve öngörülemeyen piyasa koşullarında firmaların son derece yalın bir yaklaşımla çevik olmaları gerekmektedir.

Son derece çeviklik demek : Riskin büyük olduğu ortamlarda sizi büyük kayıplardan koruyacak ve aynı şekilde anlık büyük fırsatları yakalamanızı sağlayacak bir yaklaşım demektir.

En çevik yöntem : Kanban metodu

En çevik (Agile) yaklaşım Kanban metodudur. Yazılımda 24 saat çok uzun süredir bu yüzden Türkiye şartlarında ekiplerin X hafta odaya kapanıp geliştirme yapıyorum demeleri zordur. İmkansız demiyorum ama zordur daha doğrusu sürdürülebilirliği şüphelidir. Türkiye'de ki firmalar için Kanban metodu en uygun modeldir.

Başarıyla uygulayan yok mu ?

Esas soru sürdürülebilir mi ? Bu çetin piyasa koşullarında Scrum sürdürülebilir olma ihtimali zayıftır. Özellikle destek yükünün ağır olduğu yazılım sektöründe Scrum öğretileri, ihtiyacı tam olarak karşılayamaz. Aşırı kaotik piyasa şartlarına karşı Scrum yapısı kırılgandır.

Diğer bir nokta ise her organizasyon bir ve aynı değildir. Tribal leadership kitabında her bir grubun bile kendi kültürü olduğundan bahseder. Siz tüm şirkete tek birşey dikte ederseniz bu geri büyük olasılıkla geri tepecektir.

Ne yaparsan yap Yalınlıkla (Lean) yap

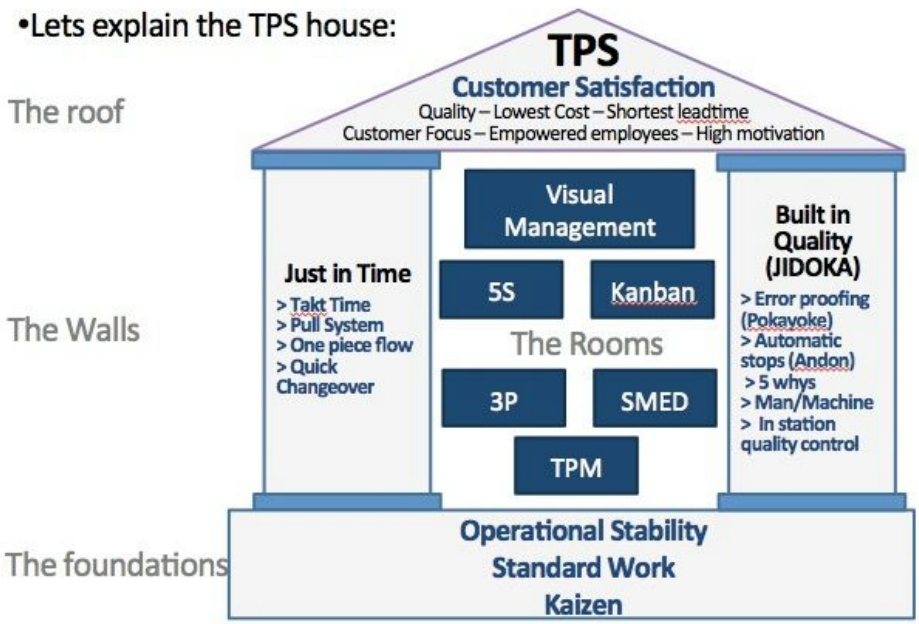
İster Çevik (Agile) ister Şelale (Waterfall) yaklaşın ama her iki durumda da yalın (Lean) dönüşümü gerçekleştirmek zorundasınız çünkü israfları yok etmek hayatta kalmak demektir. Büyüklerimizin dediği gibi:

İşten artmaz, Dişten artar.

Yalın olunmadığı ve israflara bir sınır getirilmediği sürece kalıcı başarı yakalanmış olmaz.

Özetlersek

Kanban'a geçin rahat edin gibi bir tavır içerisinde değilim. Hedef yalın dönüşümü gerçekleştirmek olmalıdır.



TPS yani Toyota Production System, yani Lean yani Türkçesi Yalınlık.
Yukarıda ki grafik özetle şunu der : Kanban yönetimi yalın dönüşümün sadece bir parçasıdır. Yol uzun ama getiriler büyüktür. Hedef kültür değişimi olmalıdır yani Yalın dönüşüm olmalıdır.

Kaynaklar

1. David Anderson -Kanban: Successful Evolutionary Change for Your Technology Business
2. Henrik Kniberg – Lean from the Trenches: Managing Large-Scale Projects with Kanban
3. Jez Humble – Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Addison-Wesley Signature Series (Fowler))
4. Hakan Forss – http://hakanforss.wordpress.com/2011/06/23/control-chart-how-to-create-one-in-excel-2010/
5. Pawel Brodzinski – http://brodzinski.com/2009/11/kanban-story-kanban-board.html
6. David Anderson – Kanban: Successful Evolutionary Change for Your Technology Business
7. http://en.wikipedia.org/wiki/Recognition_primed_decision
8. Thinking fast and slow – bölüm 22
9. http://atam.gov.tr/basari-ve-basari-yolu/#
10. Kanban: Successful Evolutionary Change for Your Technology Business by David J Anderson
11. Lean from the Trenches: Managing Large-Scale Projects with Kanban by Henrik Kniberg
12. Toyota Kata: Managing People for Improvement, Adaptiveness and Superior Results by Mike Rother
13. Liz Keogh – http://www.youtube.com/watch?v=G2X_7ojZwtU
14. Toyota Way – http://www.amazon.com/The-Toyota-Way-Management-Manufacturer/dp/0071392319
15. Cynefin – http://cognitive-edge.com/library/more/video/introduction-to-the-cynefin-framework/