Onur Özcan

NASIL AGİLE OLUNMAZ

Yalnızca Scrum ile değil; mühendislik,
otomasyon ve insana yatırımla dönüşüm rehberi.

sharpware

sharpware.co

Onur Özcan

Ben Onur Özcan; bankacılıktan e-ticarete, Garanti'den eBay'e uzanan 15 yılı aşkın serüvenimde, yazılım dünyasının mutfağında pek çok proje ile boğuştum. Şimdilerde **Sharpware**'in kurucu ortağı ve Bahçeşehir Üniversitesi'nde öğretim görevlisi olarak, sektörde "çeviklik" adı altında pazarlanan hayallerin değil, bizzat sahada işe yarayan gerçek mühendislik pratiklerinin peşindeyim. Burada okuyacaklarınız; süslü unvanlar, ezberlenmiş kılavuzlar veya içi boş sertifikalar değil; kan, ter ve tecrübeyle harmanlanmış, "karpuz projelerden" ve bürokrasiden bıkmış profesyonellere yönelik filtresiz gerçeklerdir. Hazırsanız, görüneni değil, olması gerekeni konuşmaya başlayalım.

1. Köprüleri Kurmadan Çeviklik Olmaz

İşin sahibini işi yapanlara -gerçekten- yaklaştırmadan çevik olmayı beklemeyin.

İşin sahibi,

- *“Benim çok işim var.”,*
- *“Ne istediğimi söyledim ya, benden daha ne bekliyorsunuz?”*
- *“Bu sorumluluk bana mevcut sorumluluklarımın yanında ilave sorumluluk olarak verildi, ancak bu kadar vakit ayırabiliyorum.”*
- *“Siz yapın, ben sonra UAT’de bakarım”*

gibi ifadeler kullanıyorsa ve siz bu durumu düzeltmek için hiçbir şey yapmıyorsanız çevik falan olamazsınız.

2. Müşteri Masada Değilse, Çeviklik Yolda Kalır

Çoğunlukla IT ekipleri önceliklendirmenin ne olduğunun ve bunun öneminin zaten farkındadır. Gidip müşteriyi doğru düzgün önceliklendirme yapmaya ikna etmediğiniz, her Sprint Genel Müdür'den, GMY'den, Dünya Başkanı'ndan araya işler sokuşturduğunuz sürece **bu iş yine olmayacak.**

Ne Yapmalı?

Takımlara gereksiz bir ton eğitim vermek yerine şunlara odaklanın:

- **Müşteriye önceliklendirme ve odağın önemini** iyice anlatın.
- Takımların odağını sürekli değiştirmenin y**an etkilerini** gösterin.
- Önceliklendirme yapma ve odağı koruma konularında onları ikna etmeye çalışın.

Müşteriye çevikliği anlatırken, *"bak bir çevik olalım her istediğiniz hemen yapılacak, işler 8 kat hızlanacak"* şeklinde ifadeler kullanıp,

mavi boncuk dağıtmayın. Bunun başarılabilmesi için kendisinin de bir şeyleri değiştirmesi gerektiğini, çevikliğin sadece IT'nin bazı pratikleri uygulamasından ibaret olmadığını öğrenmelerini sağlayın.

3. Kirli Kod ve Noel Baba Beklentisi

Elinizde kirli bir kod tabanı varsa, kapasitenizin tamamını müşteri gereksinimlerine ayırmayın.

- Her iterasyon kod tabanınızı biraz daha temiz hale getirmek için çaba harcayın.
- Az az da olsa kod kalitesini ve mimariyi düzeltin.

Kötü kalitenin üzerine kat çıkmaya devam ederseniz **daha hızlı gidemeyeceksiniz**. Çevik falan da olamayacaksınız. **Scrum falan sizi kurtarmayacak.** Kalite problemlerinizi şeffaf bir şekilde ortaya koyup, düzeltmek için efor harcayın. **Siz uyurken bir gece Noel Baba gelip kötü kodlarınızı düzeltmeyecek.** Bu konuda bir mucize beklemeyin.

4. Hızın Anahtarı: Üretimde Otomasyon

Makinaya yaptırabileceğiniz işleri insanlara yaptırdığınız sürece değişime karşı hızlı adapte olamazsınız. İnsanlar makinalara göre yavaştır ve hata yapma (rework'e neden olup, maliyeti yükseltme) olasılığı yüksektir.

Hız istiyorsanız kaliteyi korumanız (yüksek teknik borç taşımamanız) ve fikirden çalışan çözüme ulaştığınız üretim sürecinizde uygun olan şeyleri insan yerine makinalara yaptırmanız gerekir.

- **Zor Olan:** Yazılım üretimi sürecinde otomatikleştirilmesi en zor olan kısım gereksinimlerle ilgili çalışmalardır. “Makina bana şu projenin gereksinimlerini üretiversin” demek en azından bugün çok mümkün değil. Bu kısım hala yoğun insan emeği gerektiriyor ve gereksinimlerle ilgili çalışmaların maliyetinden yırtma şansımız pek yok.
- **Mümkün Olan:** Ancak dağıtım ve test gibi aktiviteleri -tümü olmasa bile büyük bir bölümünü- otomatikleştirmek mümkün ve çok

da zor değil. Hatta kod üretimini bile belli koşullarda belli araçlarla otomatikleştirmek söz konusu.

O zaman ne yapıyoruz?

Hızlı gidebilmek için üretim sürecimizde otomasyonu arttırıyoruz. Gerçek çeviklik için muhtemelen bir ordu büyüklüğünde olan ve işleri tekrar tekrar test datası hazırlayıp ekranların sağına soluna tıklamak olan test ekiplerinizi **"otomasyon" konusunda eğiterek** işe başlayabilirsiniz.

5. "Light" Yöntemler ve Mühendislik Gerçeği

Çeviklik sadece Scrum gibi proje yönetimine odaklanan **light yöntemler** kullanılarak elde edilebilecek bir şey değil. Kendinizi ve takımlarınızı kandırmayın. Scrum, Kanban gibi yöntemleri **iyi mühendislikle beslemediğiniz**, işi yapan insanların yetkinliklerini geliştirmek için yatırım yapmadığınız sürece 2 haftalık Sprintlerinizin sonunda ancak *"yine doğru düzgün bir şey teslim edemedik"* dersiniz.

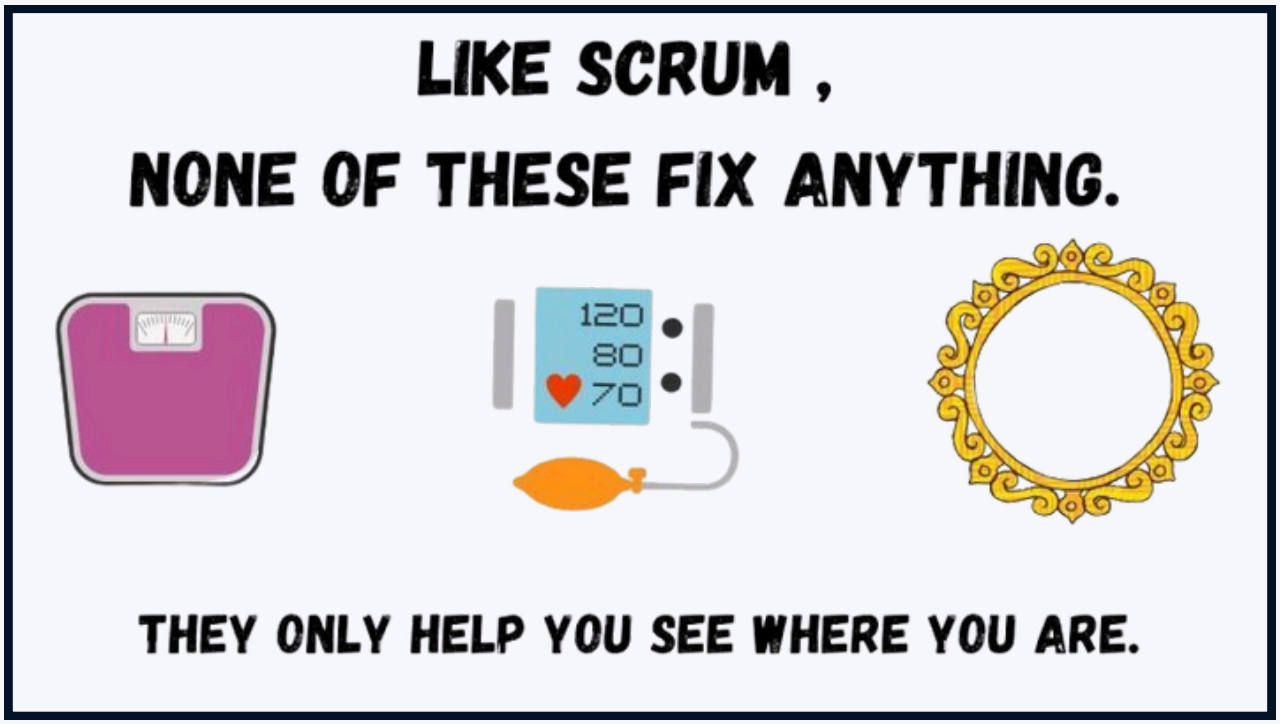
İnsanların gelişimlerine destek olun. Çalışanlarınız için **20-30 dolarlık birkaç kitap** ve/veya e-learning içeriklerine yatırım yapmaktan çekinmeyin. Bu paralar hiçbir şirketi batırmamış veya zengin etmemiştir. Şirketinizde 10 dakika turlasanız bunun kat ve katı israf görürsünüz, onları azaltıp, çalışanların gelişimine yatırım yapın. **İşi yapan insanların gelişimi** en az Scrum Master veya Product Owner'ın gelişimi kadar önemlidir.

Danışmanlar size bu gerçekleri söylemek yerine:

- *"Scrum master'ları geliştirmeye odaklanalım,"*
- *"Product owner'lar için bir kamp yapalım,"*
- *"Buradaki asıl problem takımlar arası dependency yönetimi"*

falan diyip 1-2 senenizi heba ederler. Şenliklerle, çarşaf çarşaf CEO mailleri ve *"yaşasın biz de agile oluyoruz"* temalı LinkedIn postları ile başlayan agile dönüşüm insiyatiflerinizin **mutsuz son ile bitmemesi** için benden söylemesi.

6. Liderlik ve Halının Altındaki Engeller



Sevgili dostum Hamdi Küçük paylaşmıştı bu görseli daha önce.

Bir çevik dönüşümde takımlara eğitim aldırıp Scrum veya Kanban kullanmaya başlamalarını sağlayan şirketler dönüşümün %95'ini hallettiklerini zannediyorlar. **Bu yöntemler sizin yerinize problemlerinizi fixleyecek sihirli değnekler değiller.** Asıl iş bu yöntemleri kullanmaya başladıktan sonra farkına varılan engelleri ortadan kaldırabilmek veya minimize edebilmek.

Özellikle şirketlerin liderlerine çok fazla iş düşer bu aşamada. Takımlar veya bir Scrum Master size bir engelden bahsettiğinde bunu halının altına süpürürseniz, **Scrum'ın yapabileceği tek şey o engelin hala orada olduğunu size göstermeye devam etmektir.** Sevgili liderler, yöneticiler; **elinizi kirletmeden, takımların sorunları / engelleri ile ilgilenmeden Scrum'ın bir mucize yaratmasını beklemeyin.** Sadece masanızda oturup *"işte bizim çocuklar da eğitim aldılar, 2 haftalık Sprintler yapıyorlar"* dediğinizde çalışanlarınız da dönüşüme olan inancınızı ve samimiyetinizi sorgulamaya başlayacaktır.

Takımlarınıza kulak verin ve onlar için mücadele edin.

7. Gereksinimlerin Metodolojisi Olmaz

Her ne kadar "Agile Software Requirements" isimli kitaplar/makaleler olsa da bir Agile projenin gereksinimleri diğer yazılım geliştirme yaşam döngüsü modellerini takip eden projelerden **niteliksel olarak farklı değildir.**

Software Requirement'ın Agile'ı Waterfall'ı olmaz. Bununla birlikte, çevik ve geleneksel projeler, özellikle gereksinimlerle ilgili çalışmaların **zamanlaması ve derinliği** ile gereksinimlerin **dokümantasyon şekli** açısından elbette farklılıklar içerir.

Hangi SDLC modelini kullanırsanız kullanın tüm projelerde doğru işlevselliği doğru şekilde geliştirebilmek için işi yapacak geliştiricilerin önüne aynı bilgileri/detayları koymanız gerekir. İster **user story formatı** ile ifade edin, ister **use case tekniğini** kullanın, isterseniz beyaz bir tahtada görseller çizin **işi yapacak insanlar "ne" üreteceklerini iyi anlamalıdırlar**. Agile Coachlarınıza veya Scrum Masterlarınıza boş beleş eğitimler aldırana kadar tüm takım üyelerine **Yazılım Gereksinimleri, Görsel Analiz Modelleri (UML vb.), Prototipleme Araçları** gibi konularda eğitimler aldırın.

8. Kod Yazmadan Değişebilmek

Değişen müşteri gereksinimlerine hızlı bir şekilde adapte olabilmek istiyorsanız, **kolaylıkla değiştirilebilir uygulamalara** sahip olmalısınız. Müşterinizden gelen her gereksinim için kod yazıp, test yapmak zorunda kalıyorsanız çeviklik konusunda epeyce bir yolunuz var demektir.

Uygulamalarınızın en sık change alan bölümlerini herhangi bir kod geliştirmenize ve test yapmanıza gerek kalmadan değiştirmenize imkan veren yapılar tasarlayın. **Imperative yapılar yerine Domain-Specific Language / Rule Engine gibi Declarative yapılar** kullanmanız değişiklik taleplerini minimum maliyetle karşılamanızı sağlayarak çevikliğinizi arttırır.

He tabi bunları tasarlamak ve geliştirmek tecrübe ve emek ister. Bu yollar biraz daha **kan, ter, gözyaşı gerektirir.**

Bunlarla uğraşmak yerine;

> "Çok keyifli bir Retrospective tekniği denedik, tüm takım üyelerinin çocukluğuna indik. Meğer bizim Ahmet'in ilkokul öğretmeni eline cetvelle vurmuş ondan bu kadar agresifmiş, bu Sprint bunu öğrendik."

şeklinde de devam edebilirsiniz çevik dönüşümünüze.

9. Danışmanlık ve ”Hands-on” Gerçeği

Eğer çevik dönüşüm için bir danışmanlık desteği alıyorsanız **danışmanları oyunun içerisine - gerçekten- dahil edin.** Sadece takımlarınıza ve yönetiminize "ne" yapılması gerektiğini söyleyip, kendisi hiç bir işe elini sürmeyen danışmanlarla çevik dönüşüm falan yapamazsınız.

Sınıf eğitimlerinde size ideal kurgular üzerinden product backlog oluşturmayı, sıralamayı anlatan arkadaşların çoğu gerçek bir projede *"hadi gel beraber bir product backlog oluşturalım"* dediğinizde kalem oynatamazlar. Size sadece

kitabi doğruları söyleyen arkadaşlar çoğu zaman söylediklerini herhangi bir şirkete uygulamış durumda değiller.

> **Acı Gerçek:** Takımlarınızla birlikte -**hands on**- iş yapmayan, sadece size ahkam kesen danışmanların tek faydası şirket giderlerinizi arttırıp daha az vergi ödemenizi sağlamalarıdır.

Bir de hayatında bir kez dahi bir yazılım projesinin içerisinde bulunmamış, kariyerini İK, satış gibi fonksiyonlarda geçirmiş danışmanların yazılım takımlarınıza fayda sağlama olasılığı yoktur. Faydanın aksine, işi gerçekten yapan insanlara işin nasıl yapılması gerektiği konusunda bol miktarda ahkam kesip canlarını sıkarlar, üretkenliklerini düşürürler.

Takımlarınız, insanlarınız bu kadar değersiz değil. Önünüze danışman diye koyulan her insanı kabul etmeyin.

10. Sertifika Fetişizmi ve Gerçek Dönüşüm

Sevgili şirketler/teknoloji emekçileri, bir konuda sizi uyarmak istiyorum. Şirketinizdeki herkese;

- **PSM, PSPO, PSD, SPS,**
- **PAL, PAL-E, PSK, PSU,**
- **CSM, CSPO, PMI-ACP**

sertifikalarını aldırdığınızda **dönüşmüş olmuyorsunuz.** Linkedin'de boy boy paylaştığınız sertifikalar günün sonunda işi yapmıyor. **Kağıt parçalarına odaklandığınız kadar işinizi nasıl daha iyi yapacağınıza odaklansanız** projelerinizin başarılı olma şansı artar. Sertifikayı veren kurumlar basılı halini göndermeye bile tenezzül etmiyor, pdf üretip üzerine isminizi yazıyor, ne kadar değerli olduğunu buradan anlayabilirsiniz.