Scrum

Kanban

İkisinin de En İyisini Yapmak

Kanban ve Scrum, İkisinin de En İyisini Yapmak

Henrik Kniberg & Mattias Skarin

Önsözler: Mary Poppendieck & David Anderson

Çeviri: Cihan Yılmaz

Türkçe Düzenleme: Ayşenur Yılmaz

İçerik

Çevirenin Önsözü

Kitabı çevirmemin iki nedeni var. Birinci neden Çeviklik ve Yalınlık konularında Türkçe kaynak eksikliğidir. İkinciyse Çevik Bildiri'de çoğu zaman dikkat edilmeyen, önemsenmeyen bir cümle, Çevik Bildiri'nin ilk cümlesidir. "Daha iyi yazılım geliştirme yollarını, uygulayarak ve başkalarının da uygulamasına yardım ederek, bu yolları ortaya çıkarmak…"

Kitaplarını okuduğum, konuşmalarını dinlediğim, seminerlerine katıldığım, videolarını izlediğim, Çeviklik ve Yalınlık topluluklarına büyük katkılarda bulunan insanlar var. Çeviklik'i ve Yalınlık'ı öğrenmemi sağladılar. Umarım bu kitapla birilerinin Çeviklik'i ve Yalınlık'ı öğrenmelerine yardımcı olabilirim. Umarım gelecekte Türkçe kaynak sayısı artar böylece Çeviklik ve Yalınlık herkes tarafından öğrenilebilir.

Çeviklikle tanışmam kaos halini alan işlerimizi düzene sokmak için neler yapabileceğimi düşünmekle başladı. Bir müşteri telefon ediyordu ya da bir e-posta gönderiyordu ve onun işlerini yapmaya başlıyordum. Bir şeyler geliştirmeye çalışırken daha büyük bir müşteriden baskı geliyorsa her şeyi bırakıp yangını söndürmeye çalışıyordum. On dakika içinde üç-dört farklı müşterinin projeleri arasında gidip geldiğim çok oldu. En sonunda hangisinin sesi en yüksek çıkıyorsa onunla ilgileniyordum. Bir müşterinin talebini geliştirmeye çalışırken akışa* ulaşabildiğim çok az oluyordu. Bir düzen yoktu.

Dünya üzerinde bu sorunu yaşayan tek şirket olmadığımızı biliyordum. Araştırmaya başladığımda Çeviklik ve Scrum'la tanıştım. Birkaç ay boyunca Çeviklik'in ve Scrum'ın aynı şeyler olduğunu düşündüm. Çeviklik'in özü olan deneycilik felsefesiyle on bir yaşındayken kuzenimin verdiği felsefe kitapçığında tanışmıştım ve sevmiştim. Buna rağmen iş hayatımda uzun süre determinizmi savundum. Her şeyi önceden bilip planlayabileceğimi ve planı oluşturduktan sonra herhangi bir engelle karşılaşmayacağımı savunuyordum. Bir yazılım geliştiricinin müşteriyle iletişim halinde olmaması bunun yerine arada birilerinin olması ve yazılım geliştiricinin sadece kod yazması gerektiği gibi düşüncelere sahiptim. Belki o zaman müşterilerle fazla muhatap olmak işimden zevk almamı engellediği için böyle düşünüyordum. Kanban'la tanışmamsa Scrum'la tanışmamdan birkaç yıl sonra oldu.

Çeviklik'i anladıkça bir plan oluşturup sadece planı takip ederek başarılı olamayacağımı ve müşteriyle her zaman iletişim halinde olmam gerektiğine karar verdim.

Kitapta göreceğiniz gibi Agile yerine Çevik sözcüğünü, Agility yerine Çeviklik sözcüğünü kullandım. Eğer Türkçe'de Scrum'ı ifade etmemizi sağlayacak bir sözcük olsaydı onu kullanırdım. Scrum, Rugby'de oyuncuların kafa kafaya verdikleri toplanma hareketidir. Türkçe'de bunu karşılayan bir ifade yok. Bu nedenle Scrum dedim. Kanban, Japonca görsel kart anlamına geliyor. Yine Türkçe'de bu şekilde ifade etmek anlamlı gelmiyor. Hâlbuki Çevik, Çeviklik İngilizce Agile'den çok daha güzel, okuyanın zihninde ışık yakıyor. Bu nedenle Türkçe'de anlamlı bir karşılığı olan bütün ifadeleri Türkçe karşılıklarıyla yazdım. Türkçe'ye çeviremediğim özel ifadeler için sizin aklınıza gelen bir şeyler olursa lütfen paylaşın. Yaptığım çevirilerde kullandığım Türkçe sözcükleri seçerken aslında ne kadar önemli bir iş yaptığımı hatırlatan ve yardımcı olan arkadaşım Mehmet Bozok'a teşekkür ederim.

Kitap bittikten sonra ilk okuyan ve çeviriyle ilgili "anlamak için sadece bir defa okuyorum" gibi olumlu yorumlar yapan Âdem Topal'a çok teşekkür ederim. İngilizce "waste" ifadesinin yerine israf sözcüğünü kullanabilirdim, çok uzun sürede kullandım. Alper Tonga'nın **Kanban; Evrimsel Değişim – S01E01** yazısını okuduktan sonra israf sözcüğünün yerine çöp sözcüğünü kullanmaya başladım. Alper'in çöp sözcüğünü kullanmasının nedeni çöpün somut olarak bir varlık ifade etmesidir. Çöp, kötü kokusu, çirkin görüntüsü olan ve kurtulmamız gereken her şey olabilir. Bu tanımın yanında israf sözcüğü kulağa çok soyut geliyor. Bu soyut anlam aksiyon almamız için uyarıcı bir nitelik taşımıyor. Çöp ifadesini topluluğa kazandırdığı için Alper Tonga'ya teşekkür ederim.

Kitabı çevirirken yardım eden ve Türk Dili ve Edebiyatı dersini neden sevmediğimi hatırlatan ama sonsuz dikkatiyle bütün yanlışlarımı düzelten biricik kardeşim Ayşenur'a çok teşekkür ederim.

Bu kadarcık küçük bir kitapta bu kadar çok bilgi vermelerine rağmen okuyucuyu sıkmayan bir kitap yazdıkları için Henrik Kniberg ve Mattias Skarin, çok teşekkür ederim.

Ben yeterince konuştum ☺ Zaman değerli, kitabı okuduğunuz için teşekkür ederim. Eğer geri bildirimde bulunmak isterseniz çok sevinirim.

*Belki bütün işlerde akış vardır ama yazılım dünyasında çok daha önemlidir çünkü aklınızda binlerce şeyi tutmanız ve çok kısa zamanda hatırlamanız gerekir. Akış, sizi üst benliğe ulaştırabilir. Eğer yazılım geliştirici değilseniz şuan söylediklerim anlamsız geliyor olabilir, bir yazılım geliştiriciye sorun! ☺ O, ne dediğimi biliyor.

Cihan YILMAZ
Ekim 2016 – İstanbul

site@yilmazcihan.com

http://yilmazcihan.com

Mary Poppendieck'in Önsözü

Henrik Kniberg, karmaşık bir durumun özünü çıkarabilen, tesadüfi olayların kökündeki nedenleri sıralayan ve anlaşılması çok kolay, kristal kadar net açıklama yapabilen o ender insanlardan biridir. Bu kitapta Scrum ve Kanban arasındaki farkları anlatarak mükemmel bir iş yaptı. Scrum ve Kanban'ın sadece birer araç olduğunu net bir şekilde belirtti. Gerçekte istenilen şey dolu dolu bir araçtır. Güçlü ve zayıf yönleri anlaşılan araçların nasıl kullanılacağı da anlaşılır.

Bu kitapta Kanban hakkında her şeyi -güçlü yönlerini, sınırlarını ve ne zaman kullanmanız gerektiğini- öğreneceksiniz. Aynı zamanda Scrum'ı ya da kullanabileceğiniz herhangi bir aracı ne zaman ve nasıl geliştirebileceğiniz konusunda da bilgilere sahip olacaksınız. Henrik, önemli olan şeyin kullandığınız araç olmadığını fakat kullandığınız aracı, kullanım şeklinizi sürekli olarak geliştirmek olduğunu açık bir şekilde ifade ediyor. Ayrıca zamanla kullandığınız araçların sayısını artırmakta önemlidir.

Kitabın ikinci bölümü Mattias Skarin tarafından yazılmış. Mattias hayatın içinden Scrum ve Kanban kullanımlarına örnekler vererek kitabın daha da etkili olmasını sağlıyor. Bu bölümde yazılım geliştirme sürecini iyileştirmek için Scrum ve Kanban'ın ayrı ayrı ve beraber kullanımlarını göreceksiniz. Bir işi yapmak için sadece bir tane "en iyi" yol olmadığını fark edeceksiniz. Daha iyi bir yazılım geliştirme süreci için hali hazırda bulunduğunuz durumu düşünmeli ve sıradaki adımınızın ne olması gerektiğine karar vermelisiniz.

Mary Poppendieck

David Anderson'un Önsözü

Kanban, çok basit bir fikir üzerine kuruludur. Üzerinde çalışılan iş sayısı(WIP) sınırlandırılmalıdır ve yeni bir işe ancak kendinden önceki iş bittiğinde başlanmalıdır. Kanban şu anlama gelmektedir: Yeni bir iş çekilebileceğini gösteren görsel işaret, kart. Kulağa çok devrimsel bir fikir gibi gelmiyor değil mi? Performansa, kültüre, yetkinliğe, bir takımın olgunluğuna ve bu takımı çevreleyen organizasyona derin bir etkisi olmaz gibi değil mi! Şaşırtıcı şey ise etkisinin olması. Kanban çok küçük bir değişiklikmiş gibi görünür fakat iş hakkındaki her şeyi değiştirir.

Kanban hakkında şimdilerde fark etmeye başladığımız şey yönetim işini değiştirmek için bir yaklaşım olduğudur. Kanban, bir yazılım geliştirme ya da proje yönetimi yaşam döngüsü veya süreci değildir. Kanban, hali hazırda var olan yazılım geliştirme yaşam döngüsüne ya da proje yönetim şekline değişimi getiren bir yaklaşımdır. Şimdi ne yapıyorsanız onunla değişime başlamak Kanban'ın ilkesidir. İlk olarak "değer akış haritasını" çıkararak varolan sürecinizi anlarsınız ve daha sonra aynı anda gerçekleştirilen iş sayısını(WIP) sürecinizin her adımında sınırlamaya karar verirsiniz. Bu aşamadan sonra kanban kartlarınız oluştukça kartınızı çekerek sistem boyunca işin akmasını sağlarsınız.

Kanban, Çevik yazılım geliştirme yapan takımlara kullanışlı olduğunu ispat etti. Eşit derecede geleneksel yaklaşımları tercih eden takımlara da çekici gelmeye başladı. Kanban, Yalınlık(Lean) girişiminin organizasyonların kültürünü değiştiren ve sürekli iyileştirmeyi destekleyen parçası olarak tanıtıldı.

Kanban sisteminde, aynı anda gerçekleştirilen iş sayısı sınırlandırıldığı için herhangi bir şey herhangi bir nedenle bloke olduğunda bu blokaj sistemin tıkanmasına neden olur. Eğer yeterince iş parçacığı bloklanırsa tüm süreç durma noktasına gelir. Bu, takımın ve içinde bulunduğu organizasyonun, problemin çözümüne, bloke olan iş parçacığının önündeki engelin kaldırılmasına ve iş akışının onarılmasına odaklanmalarını sağlar.

Kanban, işin takibini yapabilmek için değer akışının farklı adımları boyunca görsel kontrol mekanizmasını kullanır. Görsel kontrol için üzerinde yapışkan kâğıtlar bulunan bir beyaz tahta ya da elektronik bir duvar ile bir yazılım-excel, trello, jira, tfs- kullanılabilir. En iyi pratik muhtemelen

her ikisini-hem fiziksel hem yazılım ürünü- kullanmaktır. Bunun oluşturduğu şeffaflık kültürel değişime de katkıda bulunur. Çevik metotlar, aynı anda gerçekleştirilen işi sınırlama, tamamlanan iş, takımın hızı(bir döngüde bitirilen iş miktarı) gibi metriklerin raporlanması konularında şeffaflık sağlamada iyidirler. Ancak Kanban bir adım ileri giderek sürece ve sürecin akışına da şeffaflık sağlar. Kanban, dar boğazları, kuyrukları, değişkenliği ve çöpü ortaya çıkarır. Bunların tamamı bir organizasyonun performansına –değerli iş miktarının teslimi ve bunu teslim etme süresi açısından- etkide bulunan şeylerdir. Kanban, takım üyelerine ve paydaşlara aldıkları ya da almadıkları aksiyonların etkileri üzerine şeffaflık sağlar. Örneğin ilk durum çalışmaları, Kanban'ın davranışları değiştirdiğini ve iş yerinde daha iyi bir işbirliğinin geliştiğini gösterir. Dar boğazlar, çöp ve değişkenlik üzerindeki şeffaflık, iyileştirme fırsatlarını değerlendirme imkânı sunar. Takımlar iyileştirmeleri hızlıca süreçlerine uygulayabilirler.

Sonuç olarak Kanban varolan sürecin ilerlemeli olarak evrimleşmesini destekler. Bu evrim genel olarak Çeviklik ve Yalınlık değerleriyle aynı çizgidedir. Kanban insanların çalışma yönteminde kapsamlı bir devrim olmasını istemez, bunun yerine aşamalı bir şekilde değişimi teşvik eder. Bu değişim çalışanlar ve yöneticiler tarafından anlaşılmış ve değişim kararında fikir birliğine varılmış olmalıdır.

Çekme sisteminin doğası gereği, Kanban yeni işin önceliklendirilmesi ve varolan işin tesliminin taahhüdünü son ana kadar erteler. Bunun anlamı, işi, yapılması gereken en son ana kadar ertelemektir. Takımlar önceliklendirme ritminde anlaşır. Bu, takım ve diğer paydaşların tekrar eden bir şekilde belirlenen aralıklarda bir araya gelmesi anlamına gelir. Önceliklendirme paydaşların katıldığı ve sıradaki işin seçildiği bir toplantıda belirlenebilir. Bu toplantılar çok kısa olduğu için sık sık yapılabilir. Çok basit bir soru cevaplandırılmalıdır. Örneğin; "Son toplantımızdan beri iki iş teslim ettik. Şuan ki döngü zamanımız 6 haftadır. 6 hafta sonra teslim edilmesini istediğiniz ilk iki iş nedir?" Bu iki kat etkiye sahiptir: Birincisi basit bir soru sormak ve kaliteli bir cevap almak, ikincisi toplantıyı kısa tutmak. Bu beklentileri yöneterek çevikliğin ileri bir seviyeye ulaşmasını, taahhütten teslime kadar geçen sürenin kısaltılmasını ve önceliklerin değişmesinden kaynaklı tekrar eden işlerin en düşük seviyeye çekilmesini sağlar.

Kanban üzerine son bir söz; çalışılan iş sayısını sınırlandırma, teslim zamanının tahmin edilebilir ve çıktıların daha kaliteli olmasını sağlar. Engellere ve hatalara karşı alınan "Hattı Durdur"(stop the line) yaklaşımı kalitenin yükselmesinde ve tekrar edilen iş miktarının hızlı şekilde düşmesinde etkili olur.

Bunların hepsi bu kitapta mükemmel bir netlikte belirgin hale gelirken buraya nasıl geldiğimizse hala mat kalacaktır. Kanban sadece bir öğleden sonra oturulup tasarlanmadı aksine uzun yıllar

boyunca evrilerek ortaya çıktı. Kültür değişimi, yetkinlik ve organizasyonların olgunluğu üzerine olan psikolojik ve sosyolojik etkileri hayal dahi edilmedi. Bunlar keşfedildi. Kanban'la ortaya çıkan sonuçların çoğu doğal sezgilere aykırıdır. Çok mekanik bir yaklaşım gibi görünen aynı anda çalışılan iş sayısını sınırlandırma ve iş çekme mantığı aslında insanlar, iletişimleri ve gerçekleştirdikleri işbirliği üzerinde çok derin etkilere sahiptir. Kanban'ın ilk günlerinde ben dahil hiç kimse bu etkiyi öngöremedi.

En az dirençle karşılaşan değişim yaklaşımı olarak Kanban'ın evrimleşmesini takip ettim. 2003 yılı gibi bu benim için net bir hal aldı. Aynı zamanda mekanik faydalarını da izledim. O aralar Yalın(Lean) tekniklerin uygulanışını keşfediyordum. Eğer aynı anda çalışılan işi yönetmek mantıklı geldiyse o zaman bunu sınırlamak daha mantıklı olur. Bu, aynı anda yapılan işleri yönetim karmaşasını önler. 2004 yılında ilk ilkeden yola çıkarak çekme sistemini yazılıma uygulamaya karar verdim. Microsoft'ta bir müdür bana yaklaşıp, Microsoft içerisinde kullanılan bazı yazılımların bakımını yapan takımıyla ilgili yardım istediğinde bu şansı yakaladım. İlk uygulama Kısıtlar Teorisi'ni(Theory of Constraints), Drum-Buffer-Rope olarak bilinen çekme sistemini temel alıyordu. Çok büyük bir başarıydı, teslim zamanı %92 oranında düştü, çıktı 3 kattan fazla arttı ve tahmin edilebilirlik-işlerin zamanında teslimi- gayet kabul edilebilir bir orandı, %98!

2005 yılında Donald Reinertsen beni takip etti ve büyük bir Kanban sistem dönüşümü gerçekleştirdi. 2006 yılında Corbis, Seattle'da yazılım mühendisliği bölümünün başına geçtiğimde bir şans daha yakaladım. 2007 yılında sonuçları raporlamaya başladım. İlk sunum 2007 Mayıs'ında Chicago'da gerçekleştirilen Lean New Product Development etkinliğinde oldu. Daha sonra Ağustos ayında Washington'da gerçekleştirilen Agile 2007 etkinliğine katıldım. 25 kişi katıldı ve onlardan üçü Yahoo'dandı: Aaron Sanders, Karl Scotland ve Joe Arnold. Evlerine –Kaliforniya, Hindistan ve Birleşik Krallık- döndüler ve hali hazırda Scrum'la boğuşan takımlarıyla Kanban uygulamaya başladılar. Aynı zamanda Yahoo'da bir tartışma grubu başlattılar ve ben bu yazıyı yazarken neredeyse 800 üyesi bulunuyor. Kanban yayılmaya başlamıştı ve erken benimseyenler tecrübeleri hakkında konuşuyorlardı.

Şimdi yıl 2009 Kanban gitgide daha da yayılıyor ve sahadan-uygulayanlardan- daha çok rapor geliyor. Son 5 yılda Kanban hakkında çok fazla şey öğrendik ve her gün daha fazlasını öğrenmeye devam ediyoruz. Kanban'ı daha iyi anlamak ve diğerlerine daha iyi anlatmak amacıyla Kanban uygulamaya, Kanban hakkında yazmaya, konuşmaya ve düşünmeye odaklandım. Kanban'ı varolan Çevik metotlarla karşılaştırmaktan bilerek geri duruyorum fakat 2008 yılında Kanban'ın neden Çevik yaklaşımlardan biri olarak düşünülmesi gerektiğini anlatmak için biraz emek harcamıştım.

Kanban ve Scrum'ın karşılaştırılması konusunu daha tecrübeli kişilere bıraktım. Henrik Kniberg ve Mattias Skarin'in bu konu çerçevesinde liderler olarak ortaya çıkmasından çok memnunum. Siz, sahadaki bilgi çalışanları, bilinçli kararlar verebilmeniz ve işinizde ileri bir seviyeye geçebilmeniz

için bilgiye ihtiyacınız bulunur. Henrik ve Mattias ihtiyaçlarınızı benim karşılayamayacağım bir şekilde karşılıyorlar. Henrik'in düşünceli, gerçeklere dayalı, inatçı olmayan, dengeli karşılaştırma yaklaşımından özellikle etkilendim. Çizimleri ve örnekleri düşünceli ve sayfalarca metin okumaktan sizi kurtarıyor. Mattias'ın durum çalışması çok önemli çünkü Kanban'ın bir teoriden fazlası olduğunu ve organizasyonunuz için nasıl faydalı olabileceğini yaşamdan bir örnekle anlatıyor.

Kanban ve Scrum'ı karşılaştıran bu kitap umarım hoşunuza gider. Böylece genel olarak Çeviklik özeldeyse Kanban ve Scrum hakkında daha fazla bilgiye sahip olursunuz. Eğer Kanban hakkında daha fazla bilgi sahibi olmak istiyorsanız, lütfen Kanban topluluğumuzun web sayfasını ziyaret edin, The Limited WIP Society. http://www.limitedwipsociety.org/

David J. Anderson

Sequim, Washington, ABD
July 8th, 2009

Giriş

Normalde kitap yazmayız. Siperin derinliklerinde müşteriyle, müşterinin yazılım geliştirme süreçlerini ve organizasyonunu **optimize, debug ve refactor** ederken, ona yardım etmeyi tercih ederiz. Son zamanlarda bir moda fark ettik ve bu konuda düşüncelerimizi paylaşmak istedik. İşte örnek bir durum:

- **Jim:** "Sonunda hepimiz Scrum yapıyoruz!"
- **Fred:** "Eee nasıl gidiyor?"
- **Jim:** "Yani daha önce yaptığımızdan daha iyi..."
- **Fred:** "...ama?"
- **Jim:** "...ama biz bir destek ve bakım takımıyız."
- **Fred:** "Evet ve?"
- **Jim:** "Pekâlâ, Ürün İş Listesi'nde öncelikleri belirlemeyi, kendi kendini yöneten takımları, Günlük Scrum'ları ve retrospektifleri vb. sevdik."
- **Fred:** "Eee ne güzel, problem nerede?"
- **Jim:** "Sprint'lerde başarısız olmaya devam ediyoruz."
- **Fred:** "Neden?"
- **Jim:** "Çünkü 2 haftalık planlara taahhüt vermemiz zor. İterasyonlar bizim için çok bir şey ifade etmiyor. Bugün en acil konu neyse biz ona çalışıyoruz. Acaba 1 haftalık iterasyonlar mı yapmalıyız?"
- **Fred:** "1 haftalık işlere taahhüt verebilir misiniz? Huzur içinde 1 hafta odaklanmanıza ve çalışmanıza izin verecekler mi?"
- **Jim:** "Pek sanmıyorum. Her gün farklı bir konu ortaya çıkıyor. Belki de 1 günlük Sprint'ler yapmalıyız..."
- **Fred:** "Problemleriniz 1 günden daha kısa zamanda çözülüyor mu?"
- **Jim:** "Hayır, bazen günlerce sürenler oluyor."
- **Fred:** "Yani 1 günlük Sprint'ler de sizin için uygun değil. Hiç Sprint koşmamayı düşündünüz mü?"
- **Jim:** "Açıkçası evet, bu hoşumuza giderdi. Fakat Scrum buna karşı değil mi?"

- **Fred:** “Scrum sadece bir araç, ne zaman ve nasıl kullanacağını sen belirlersin. Scrum’ın kölesi olmaya gerek yok!”
- **Jim:** “Ne yapmalıyız?”
- **Fred:** “Kanban’ı duydun mu?”
- **Jim:** “O nedir? Scrum ve Kanban arasındaki fark nedir?”
- **Fred:** “İşte. Bu kitabı oku!”
- **Jim:** “Gerçi Scrum’ın geri kalanını seviyorum. Gerçekten vazgeçmeli miyim?”
- **Fred:** “Hayır! Teknikleri birleştirebilirsin.”
- **Jim:** “Ne? Nasıl?”
- **Fred:** “Sadece kitabı oku...”

Kitabın Amacı

Eğer Çevik yazılım geliştirmeyle ilgileniyorsanız Scrum’ı biliyorsunuzdur. Ayrıca Kanban’ı duymuş olabilirsiniz. Son zamanlarda sıkça duyduğumuz sorular: “Kanban nedir? Hangisi daha iyi Kanban mı, Scrum mı? Nerede birbirlerini tamamlarlar? Aralarında bir çelişki bulunuyor mu?”

Bu kitabın amacı ortalıktaki bu sisi kaldırarak Kanban’ın ve Scrum’ın organizasyonunuzda nasıl kullanışlı olabileceğini öğrenmeniz.

Eğer bu kitapla bunu size anlatabilirsek ve başarılı olursak lütfen bize yazın! ☺

Bölüm 1 – Kıyaslama

Kitabın ilk bölümünde Kanban ve Scrum'ın nesnel ve pratik bir kıyaslamasını yapmaya çalıştık. Bu kitap 2009 yılının Nisan ayında yayınladığımız "Kanban vs Scrum" makalesinin biraz daha güncel halidir. Bu makale çok popüler oldu. Bu nedenle makaleyi bir kitaba çevirmeye ve iş arkadaşım Mattias'tan "cepheden" biri olarak kitaba biraz renk katmasını rica ettim. Mattias'ın katkıda bulunduğu bölüm gerçekten harika, birinci bölümü okumadan ikinci bölümü okumaya başlayabilirsiniz. Söz veriyorum alınmayacağım! ☺

Belki biraz alınırım.

/Henrik Kniberg

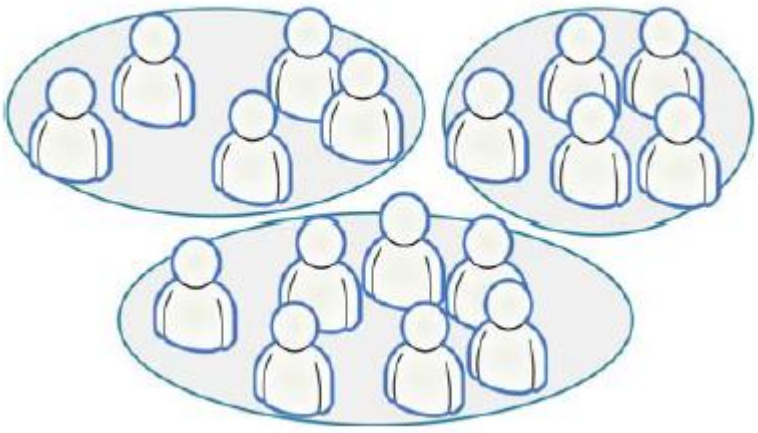
1

Scrum ve Kanban Nedir?

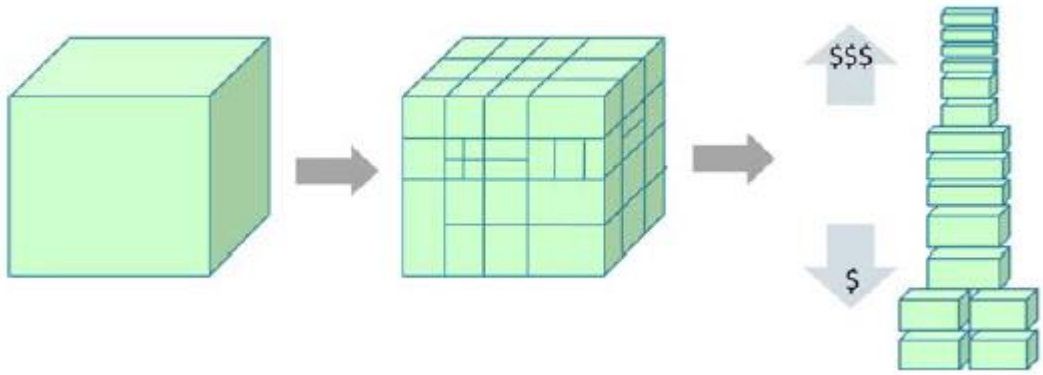
Scrum ve Kanban'ı 100 sözcükle ifade edelim.

Özetle Scrum

- **Organizasyonunuzu,** küçük, çapraz fonksiyonel ve kendi kendini yöneten takımlara bölün.



- **İşlerinizi,** küçük ve somut teslimlerden oluşan maddelere bölün.



- **Zamanınızı,** kısa, sabit uzunlukta-genellikle 1-4 hafta arasında- döngülere bölün. Döngüler sonrasında geliştirdiğiniz ve müşterinize gösterdiğiniz özellikleri teslim edin.
- **Teslim Planını en iyi hale getirin**. Müşterinizle işbirliği yaparak önceliklerinizi sürekli güncel tutun. Bunu yaparken her döngü sonrası yaptığınız teslimlerden kazandığınız tecrübeden yararlanın. Teslimlerinizi gözlemleyin ve bir sonraki tesliminizde bu gözlemlerden yararlanın.

Yani **büyük bir grup** halinde **çok uzun süre** harcayarak **büyük bir şey** –proje, ürün- geliştirmek yerine **küçük bir takım** halinde **kısa süre** harcayarak **küçük bir şey**-proje, ürün- geliştirin. Fakat düzenli olarak önceki geliştirdiklerinizle entegre etmeyi unutmayın!

104 kelime yeterince başarılı ☺

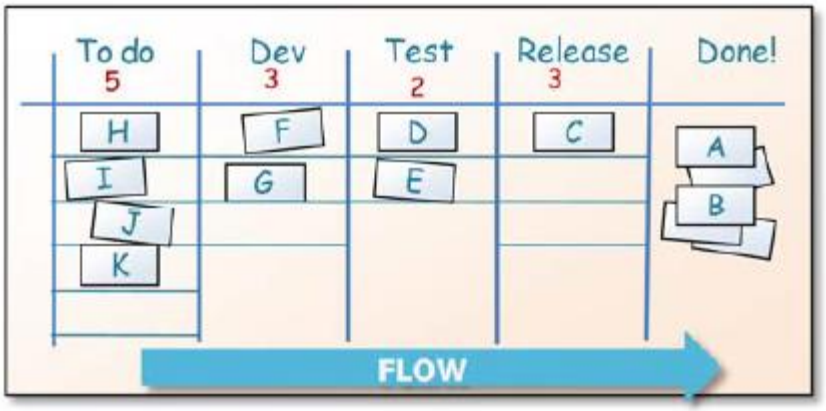
Daha fazla bilgi için "Siperden Scrum ve XP, Biz Nasıl Yapıyoruz?" e-kitabını okuyabilirsiniz. Yazarı tanıyorum, gayet iyi biri ☺

http://www.yilmazcihan.com/siperden-scrum-ve-xp-biz-nasil-yapiyoruz/

Daha fazla bilgi için bağlantıya bakabilirsiniz: http://www.yilmazcihan.com/ceviri-kitaplarim/

Özetle Kanban

- **İş akışını görselleştirin**
    - İşi küçük parçalara bölün, her küçük parçayı bir karta yazın ve bu kartları bir duvara yapıştırın.
    - Bir parçanın iş akışında nerede olduğunu görmek için iş akışınızı kolonlara bölün ve her bir kolona isim verin, bunlar iş akışınızdaki adımlar olacaktır.
- **Çalışılan İş Sayısını(WIP) sınırlayın –** İş akışınızdaki her bir adımda aynı anda **kaç** parça iş bulunabilir karar verin.
- **Teslim Süresini(Lead Time) ölçün –** Teslim Süresi, bir parça işin teslim edilmesi için gerekli olan ortalama süredir. Teslim Süresi'ni daha kısa olacak ve daha kolay tahmin edilebilecek değerlere ulaştırmak için sürecinizi iyileştirin.



Kullanışlı Kanban linklerini bu bağlantıda topladık: http://www.crisp.se/kanban

2

Scrum ve Kanban Birbiriyle Nasıl İlişkilidir?

Scrum ve Kanban, İkisi de Süreç İçin Bir Araçtır

Araç = Bir görevi yerine getirmek, bir hedefe ulaşmak için kullanılan her şey

Süreç= Çalışma şekliniz

Scrum ve Kanban *süreç araçlarıdır*. Belirli ölçüde ne yapmanız gerektiğini söyleyerek daha verimli olmanız için yardım ederler. Java'da bir araçtır. Kolay bir şekilde ürün geliştirmenizi sağlar. Diş fırçası da bir araçtır, dişlerinize ulaşmanızı ve temizlemenizi sağlar.

Araçları Anlamak İçin Kıyaslayın, Yargılamak İçin Değil

Bıçak ya da çatal? Hangisi daha iyi?

Gerçekten anlamsız bir soru değil mi? Çünkü cevap sizin içinde bulunduğunuz duruma bağlıdır. Köfte yemek için çatal muhtemelen en iyi araçtır, mantarları doğramak içinse bıçak en iyi araçtır. Masaya vurarak davul çalmak içinse ikisi de gayet yararlı olacaktır. Biftek yemek için muhtemelen

ikisini beraber kullanmak istersiniz. Pirinç yemek için bazıları çatal bazıları yemek çubuklarını tercih eder.

Yani araçları kıyaslayacağımız zaman dikkatli olmalıyız. Anlamak için kıyaslamalıyız, yargılamak için değil!

Hiçbir Araç Tam Değildir, Hiçbir Araç Mükemmel Değildir

Tıpkı her araç gibi Scrum ve Kanban'da tamamlanmış değildir, mükemmel değildir. İhtiyacınız olan *her şeyi* size söyleyemezler. Sadece yolunuzu bulabilmeniz için belirli sınırlar gösterirler ve rehberlik ederler. Örneğin Scrum kural olarak süresi önceden belirli ve sınırlı olan geliştirme zamanı-Sprint- verir, çapraz fonksiyonel takımları verir. Kanban görsel duvarlar kullanmanızı ve aynı anda geliştirdiğiniz iş sayısını kısıtlamanızı önerir.

İlginçtir ki bir aracın değeri *seçeneklerinizi sınırlamasıdır*. Her şeyi yapmanıza izin veren bir süreç aracı(process tool) kullanışlı ve faydalı değildir. Biz bu süreci "Canın Ne İstiyorsa Onu Yap Süreci" ya da "Doğru Şeyi Yap Süreci" olarak tanımlayabiliriz. "Doğru Şeyi Yap Süreci'nin" çalışacağına garanti veriliyor, sihirli değnek gibi bir şey. Eğer çalışmıyorsa, çok nettir ki süreci takip edemiyorsunuz! ☺

Doğru araçları kullanmanız başarılı olmanıza yardım edebilir. Fakat başarılı olacağınızı garanti etmez. *Projenin* başarısı ve başarısızlığıyla *aracın* başarısı ve başarısızlığını karıştırmak çok kolaydır.

- Bir proje, mükemmel bir araç kullanılarak başarıya ulaştırılabilir.
- Bir proje, kötü bir araca rağmen başarıya ulaştırılabilir.
- Bir proje, kötü bir araç yüzünden başarısız olabilir.
- Bir proje, mükemmel bir araç kullanılsa bile başarıya ulaşmayabilir.

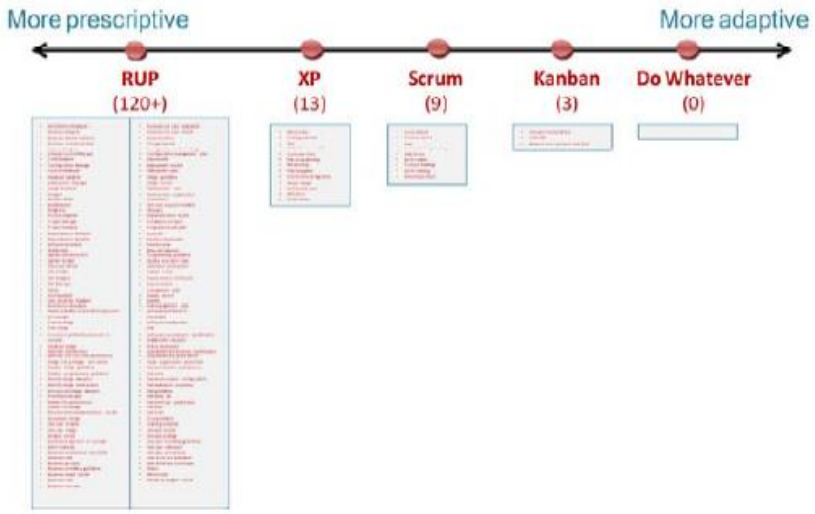
Scrum, Kanban'dan Daha Fazla Kurala Sahip

Araçları kıyaslarken kaç tane kural verdiklerine bakabiliriz. *Kuralcı*'nın anlamı **takip edilmesi gereken kural sayısının çok olması**, *Adaptif*'in anlamı **takip edilmesi gereken kural sayısının az olması** demektir. Bir süreç aracı %100 Kuralcı olduğunda beyninizi kullanma hakkınız yok demektir, her şey için bir kural bulunur. %100 Adaptif olduğundaysa Canın Ne İsterse Yap sürecidir, hiçbir kural, kısıt ve sınır yoktur. Gördüğünüz üzere iki aşırı uç örnek de biraz saçmadır.

Çevik metotlar *hafif* metotlar olarak anılırlar. Bunun nedeni gayet açıktır; geleneksel metotlardan çok daha az kurala sahiptirler. Çevik Bildiri'nin ilk değeri "Süreçler ve Araçlardan ziyade Bireyler ve İletişime değer veririz" şeklindedir.

Scrum ve Kanban ikisi de oldukça adaptiftir. Fakat göreceli olarak baktığımızda Scrum, Kanban'dan biraz daha fazla kurala sahiptir. Scrum size daha fazla kısıt verir, bu nedenle seçeneklerinizi azaltır. Örnek olarak Scrum'da süresi belirli döngüler bir kural iken, Kanban'da değildir.

Diğer metotları da Kuralcı ve Adaptif ölçeğine koyup bir göz atalım.



RUP(Rational Unified Process) oldukça kuralcıdır. 30'un üzerinde rol, 20'den fazla aktivite ve 70'ten fazla artifact(süreç içinde işinizi yaparken kullandığınız araçlar) bulunmaktadır. Öğrenmeniz gereken şeylerin altında ezilebilirsiniz. Gerçi bunların hepsini kullanmayacaksınız, projeniz için uygun bir alt kümeyi seçebilirsiniz. Ne yazık ki uygulamada bu çok zordur. "Hmmm... *Konfigürasyon Denetim Bulguları* dokümanına ihtiyacımız olacak mı? *Değişiklik Yönetimi Müdürü* rolüne ihtiyacımız olacak mı? Emin değilim, her duruma karşı bulunduralım." Belki bu nedenle RUP uygulamaları, Scrum ve XP gibi Çevik yöntemlerin uygulanmalarına göre çok ağır kalmaktadır.

XP(eXtreme Programming), Scrum'a kıyasla oldukça Kuralcı'dır. Scrum'da bulunan birçok uygulamayı içerir, artı TDD, Eşli Programlama(Pair Programming) gibi birçok Çevik Pratiğe sahiptir.

Scrum, XP'ye göre daha az Kuralcı'dır. Yani sonuçta belirli bir mühendislik pratiğini kullanmak bir kurala bağlı değildir. Scrum, Kanban'a göre daha Kuralcı'dır. Scrum'da bulunan fakat Kanban'da bulunmayan kurallardan bazıları; süresi belirli geliştirme döngüleri, çapraz fonksiyonel takımlar. Bunlar Scrum'da kural iken Kanban'da seçime bağlıdır.

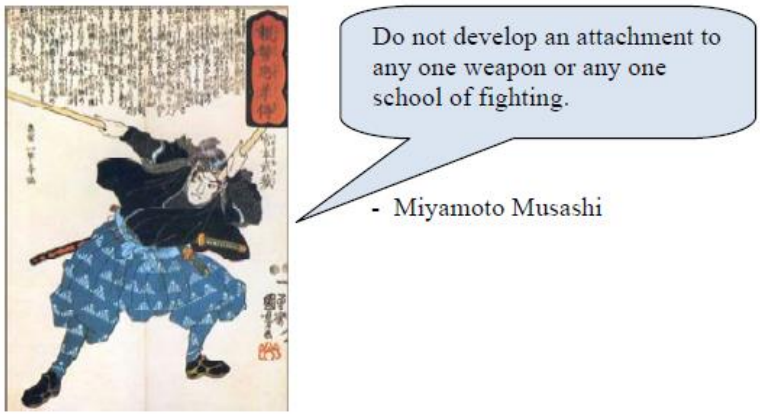
Scrum ve RUP arasındaki ana farklılıklardan biri de RUP size birçok şey verir. İhtiyacınız olmayan şeyleri belirlemeli ve siz çıkarmalısınız. Scrum size çok az şey verir, ihtiyacınız olan şeyleri siz eklemelisiniz.

Kanban neredeyse her şeyi size bırakır. Sizi sınırladığı ya da size kural olarak verdiği şeyler: İş Akışını Görselleştirme ve Çalışılan İş Sayısını sınırlamadır. "Canın Ne İsterse Yap" metoduna sadece birkaç santim uzak fakat yine de şaşırtıcı bir şekilde güçlü.

Kendinizi Sadece Bir Araçla Sınırlamayın!

İhtiyacınız olduğu sürece araçları birleştirebilirsiniz. Örneğin XP'de bulunan aktiviteleri kullanmadan başarılı olabilecek bir Scrum Takımı'nı düşünmek çok zor. Birçok Kanban Takımı günlük toplantıları(bir Scrum pratiği, Günlük Scrum) kullanır. Bazı Scrum Takımları İş Listesi Maddeleri'ni Kullanım Durumu(Use Case) olarak yazarlar, bu bir RUP pratiğidir. Bazı Scrum Takımları kuyrukta bekleyen iş sayısını limitler, buysa bir Kanban pratiğidir. Sizin için uygun olanın hangisi olduğuna karar verip onu seçmelisiniz.

Miyamoto Musashi, 17. yüzyılda yaşamış ve iki kılıç kullanma tekniğiyle ünlenmiş bir Samuray'dır. Miyamoto der ki: "Dövüş okulunda hiçbir silahla ya da hiç bir kişiyle bağ kurmayın."



Her aracın sizi sınırladığı noktalara önem verin. Eğer Scrum kullanıyorsanız ve süresi belirli döngüleri kullanmamaya karar verdiyseniz(ya da Scrum'ın getirdiği bir başka temel yaklaşımı) o zaman Scrum yapıyoruz demeyin. Scrum yeterince sadedir. Eğer bazı elementlerini kaldırır ve hala Scrum derseniz o zaman Scrum anlamını yitirir ve kafa karıştırıcı olmaya başlar. Eğer süresi belirli döngüleri kaldırıyorsanız buna Scrumish diyebilirsiniz ☺

3

Scrum Rolleri Belirler

Scrum'da 3 rol bulunur. Ürün Sahibi, ürün vizyonunu ve önceliklerini belirler. Takım, ürünü geliştirir. Scrum Master, engelleri kaldırır ve süreç liderliğini yapar.

Kanban'da belirlenen bir rol yoktur. Bu, bir Ürün Sahibi rolüne sahip olmamanız ya da olamayacağınız anlamına gelmez. Sadece bir ***zorunluluk*** olmadığını belirtir. Scrum ve Kanban kullanırken yeni roller ekleyebilirsiniz, bunda herhangi bir kısıt bulunmaz.

Gerçi rol eklerken dikkatli olmalısınız, yeni eklenen rolün gerçekten değer katacağından emin olmalısınız. Ayrıca eklediğiniz rol sürecin içinde bulunanlarla ters düşmemelidir. Örneğin; Şu Proje Yöneticisi rolüne gerçekten ihtiyacınız var mı? Belki büyük bir proje içinde bu güzel bir fikir olabilir. Belki Takımlar'ın, Ürün Sahipleri'nin senkronize olmasını sağlayan kişi, bu adam olabilir. Küçük bir proje içinde ise bu adam çöp yaratabilir ya da daha kötüsü mikro yönetim yapmaya çalışabilir.

Scrum ve Kanban'da ki genel mantık "az daha çoktur". Yani riskli bir şeyler varsa daha azla başla. Kitabın bundan sonraki bölümünde "Ürün Sahibi" terimini, kullanılan sürecin ne olduğuna bakmadan öncelikleri belirleyen kişi anlamında kullanacağım.

4

Scrum'da Döngülerin Süresi Sabittir

Scrum'ın temelinde süresi sabit döngüler bulunur. Döngünün uzunluğunu seçebilirsiniz, genel mantık döngü uzunluklarını aynı tutarak bir ***ritim*** yakalamaktır. *(Ritim sözcüğü art arda koşulan Sprint'leri çok güzel ifade ediyor. Bu nedenle kitabın kapağına kalp şeklinde bir ritim çizimi koydum.)*

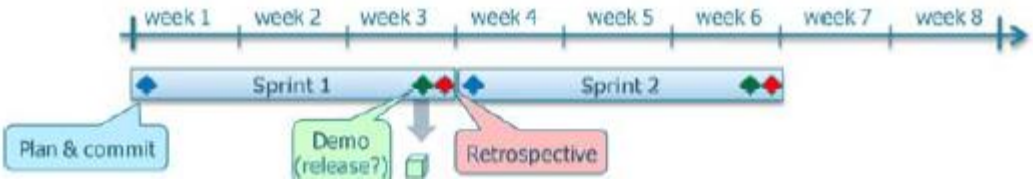
- **Döngünün başlangıcı:** Döngü planı oluşturulur. Takım belirli sayıda iş maddesini İş Listesi'nden çeker. Bu maddelerin önceliklendirilmesinin sorumluluğu Ürün Sahibi'ndedir. Takım bir döngüde ne kadar maddeyi bitirebileceğine karar verirse o kadar madde çeker.
- **Döngü boyunca:** Takım geliştirmeyi taahhüt ettiği maddeleri tamamlamaya odaklanır. Döngünün kapsamı sabitlenmiştir.
- **Döngünün sonu:** Çalışan ürün parçacığı ilgili paydaşlara gösterilir, ideal durumda bu ürün parçacığı *üretim ortamına alınabilir* olmalıdır. Daha sonra takım süreçlerini geliştirmek için retrospektif yapar.

Yani bir Scrum döngüsü: süresi sınırlı, planlama, süreç geliştirme ve teslim süreçlerini içeren bir ritimdir.

Kanban'da süresi sınırlı döngüler zorunlu değildir. Planlamayı, süreç geliştirmeyi ve teslimi ne zaman yapmanız gerektiğini kendiniz seçebilirsiniz. Bu aktiviteleri tekrar eden bir şekilde yapmayı da seçebilirsiniz. Örneğin her Pazartesi teslim yapabilirsiniz ya da bir ürün parçacığı tamamlandığında hemen teslim edebilirsiniz.

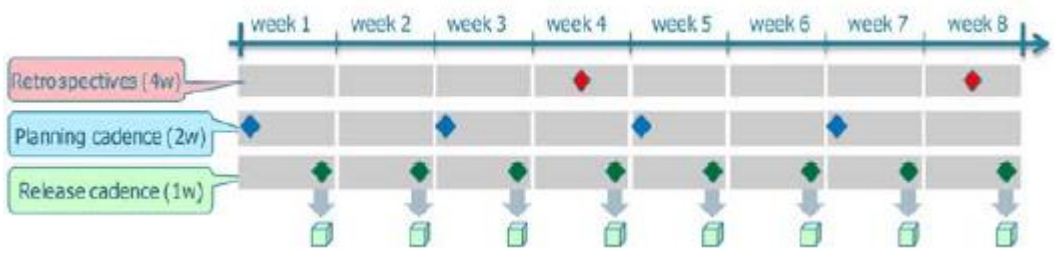
Takım #1(Tek Ritim)

"Scrum döngüleri(Sprint) yaparız".



Takım #2(Üç Ritim)

“Üç farklı ritmimiz vardır. Her hafta teslim edilmeye hazır ne varsa teslimini yaparız. Her ikinci hafta planlama toplantısı yaparız, öncelikleri ve teslim planını güncelleriz. Her dördüncü hafta retrospektif toplantısı yaparız ve sürecimizi nasıl geliştirebileceğimizi konuşuruz.”



Takım #3(Gelişen Olaylar Güdümlü)

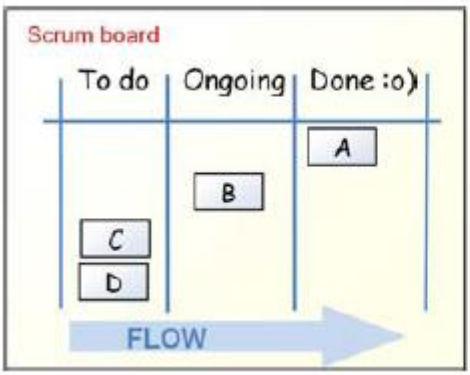
“Ne zaman geliştirilecek konularımız azalmaya başlasa bir planlama toplantısı yaparız. Ne zaman Minimum Pazarlanabilir Özelliği(MMF, Minimum Marketable Feature) geliştirmiş olsak teslimini yaparız. Ne zaman aynı problemi ikinci defa yaşarsak bir retrospektif yaparız. Ayrıca her dört haftada bir derinlemesine bir retrospektif gerçekleştiririz.”

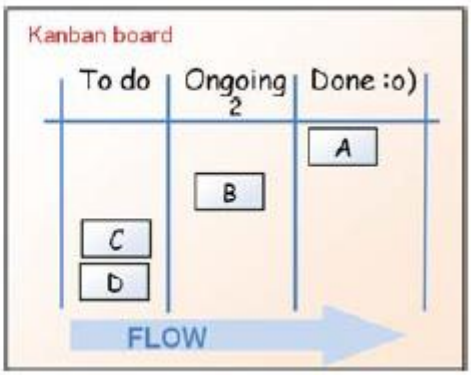
5

Kanban, *Çalışılan İşleri* İş Akışındaki Adımlara Göre, Scrum, *Çalışılan İşleri* Döngü Bazında Sınırlandırır

Scrum'da Sprint İş Listesi, içinde bulunulan döngüde(Scrum'da döngülere Sprint denir) hangi işlerin bitirileceğini gösterir. Sprint İş Listesi genelde kartlar kullanılarak bir duvar-pano- üzerinde gösterilir. Bu duvar Scrum Duvarı ya da İş Duvarı olarak bilinir.

Scrum Duvarı ve Kanban Duvarı arasındaki fark nedir? İki duvarı kıyaslamaya basit bir projeyi örnek alarak başlayalım.





Her iki durumda da iş akışı boyunca iş maddelerini takip ederiz. Biz üç aşama belirledik, Yapacağımız(To Do), Devam Eden(Ongoing), Bitti(Done) işlerin bulunduğu aşamalardır. Sizde istediğiniz aşamaları seçebilir, belirleyebilirsiniz. Bazı takımlar Entegrasyon, Test, Teslim gibi aşamalar ekler. Gerçi unutmayın ilke şu; "az daha çoktur".

İki duvar arasındaki fark nedir? Kanban Duvarı'nın orta kolonundaki küçük "**2**". Bütün fark bu. Bu 2'nin anlamı: **Devam Eden** kolonunda hiçbir zaman 2'den fazla iş maddesi bulunamaz.

Scrum'da Geliştirme Takımı iş maddelerinin hepsini Devam Eden işler kolonuna koyabilir, bunu engelleyen bir kural yoktur. İterasyonun(döngünün) kendisi sabit kapsama sahip olduğu için içsel bir limit bulunur. Bu örnek için kolon başına belirlenen içsel limit 4. Çünkü duvarda sadece 4 iş bulunuyor. Yani Kanban "Çalışılan İşleri(Devam Eden)" direkt olarak sınırlandırırken Scrum dolaylı olarak sınırlandırır.

Scrum Takımları'nın çoğu sonunda anlarlar ki aynı anda devam eden birçok işin olması kötü bir fikirdir. Scrum Takımları zamanla yeni işlere başlamadan önce ellerindeki işleri bitirme kültürüne doğru evrimleşir. Hatta bazı takımlar Devam Eden işler kolonunu direkt olarak sınırlandırırlar. O zaman işte Scrum Duvarı, Kanban Duvarı'na dönüşür.

Hem Scrum hem Kanban Çalışılan İş sayısını sınırlar, sadece farklı yollarla yaparlar. Scrum Takımları genelde takımın hızını -bir iterasyonda kaç iş maddesi bitirdiklerini ya da iş maddelerine verdikleri hikaye puanlarını(story point)- ölçerler. Takım hızı belirlendiğinde aslında bu takımın Çalışılan İş sayısı limiti belirlenmiş olur. Ortalama hızı 10 iş maddesi olan bir takım bir döngüde 10'dan fazla iş almaz. Eğer takım hızını hikaye puanı bazında tutuyorsa yine bir Sprint'te bitirdikleri hikaye puanından daha fazla iş çekmezler.

Scrum'da işler belirli bir zaman-Sprint- için sınırlandırılmıştır.

Kanban'da işler iş akışındaki adımlar bazında sınırlandırılmıştır.

Yukarıdaki Kanban örneğinde, döngü uzunluğundan bağımsız şekilde Devam Eden işler kolonunda en fazla 2 iş maddesi bulunabilir. İş akışındaki her bir adım için uygulanacak sınırı belirlemelisiniz. Genel mantık iş akışındaki bütün adımlarda Çalışılan İş sayısını sınırlamaktır. Yani yukarıdaki örnekte Yapılacak(To Do) işler adımına Çalışılan İşler için bir limit eklemeliyiz. Daha sonra Teslim Süresini(Lead Time) ölçmeye ve teslim tarihini tahmin etmeye başlayabiliriz. Teslim Süresi, bir iş maddesinin Kanban Duvarı'ndaki tüm adımları geçiş süresidir. Teslim Süresini belirlememiz taahhüt vermemize yardımcı olur ve daha gerçekçi teslim planları yapabiliriz.

Eğer iş maddelerinin büyüklükleri çok değişkenlik gösteriyorsa o zaman Çalışılan İş miktarını hikaye puanı ya da kullandığınız birim neyse o seviyede sınırlamalısınız. Kimi takımlar, böyle sorunlarla uğraşmamak için iş maddelerini aynı büyüklüğe bölerler. Böylece iş maddelerini tahmin ederken harcanan süreyi azaltmış olurlar. Bazı takımlar işlerin büyüklüğünü tahmin etme işini çöp olarak değerlendirebilirler. Eğer iş maddeleri yaklaşık olarak aynı büyüklüğe sahipse düzgün iş akışı oluşturmak daha kolaydır.

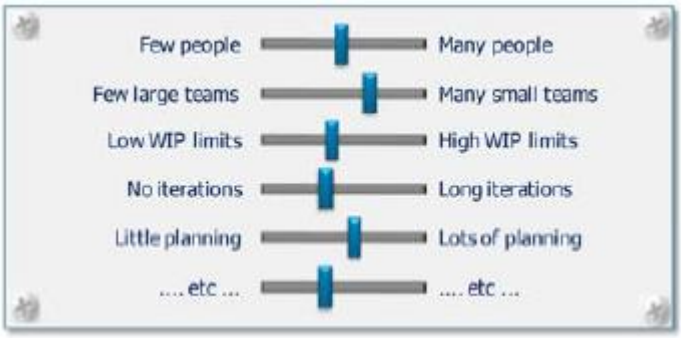
6

İkisi de Deneyseldir

Bu sayaçların üzerinde düğmeler olduğunu ve basitçe bu düğmeleri çevirerek sürecinizi yapılandırabildiğinizi hayal edin! "Yüksek kapasite, kısa teslim süresi, yüksek kalite ve yüksek tahmin edilebilirlik istiyorum. O zaman göreceli olarak düğmeleri 10, 1, 10, 10'a çeviririm".

Mükemmel olmaz mıydı? Maalesef böyle **direkt** bir kontrol yok. En azından benim bildiğim bir kontrol yok. Eğer siz bulursanız bana da haber verin. ☺

Bizim sahip olduğumuzsa birkaç **dolaylı** kontroldür.



Scrum ve Kanban, her ikisi de deneyseldir. Bu nedenle süreç üzerine deneyler yapmanız ve ortamınıza göre yeniden şekillendirmeniz beklenir. Aslında bir beklentiden öte deney yapmanız *zorunludur*. Ne Scrum ne de Kanban bütün cevaplara sahiptir, kendi sürecinizi belirleyebilmeniz için sadece temel sınırları verirler.

- Scrum der ki; çapraz fonksiyonel takımlarınız olmalıdır.

    - Eee kim, hangi takımda olmalı? Bunu söylemez, deney yapmalısınız.
- Scrum der ki; **Takım** bir Sprint'e alınacak işlerin miktarını belirler.
    - Eee Takım bir Sprint'e ne kadar iş almalı? Bunu söylemez, deney yapmalısınız.
- Kanban der ki; Çalışılan İşleri sınırlamalısınız.
    - Eee limit ne olmalı? Bunu söylemez, deney yapmalısınız.

Önceden de söylediğim gibi Kanban, Scrum'a göre daha az kısıta sahiptir. Bunun anlamı şudur; düşünmeniz gereken daha çok değişken, kullanmanız gereken daha çok düğme bulunur. İçinde bulunduğunuz duruma göre bu bazen avantaj bazen de dezavantaj olabilir. Bir yazılım IDE'sinin yapılandırma penceresini açtığınızda değişiklik yapacağınız 3 seçenek mi, 100 seçenek mi olsun istersiniz? İsteğiniz muhtemelen iki değerin ortasında bir şey olacaktır. Bu, IDE'yi ne kadar iyi bildiğinize ve yapmanız gereken değişikliğe bağlıdır.

Sürecimizi iyileştireceği hipotezine dayanarak Çalışılan İş sınırını düşürürsek kapasite, Teslim Süresi, kalite ve öngörülebilirlik gibi şeylerin nasıl değiştiğini gözlemleriz. Elde edilen verilerden sonuçlara varırız ve sonra başka şeyleri değiştiririz böylece sürekli olarak sürecimizde iyileştirmeler yapmaya çalışırız.

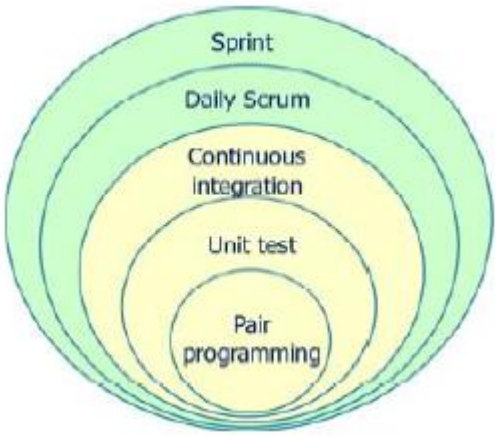
Bunun birçok adı vardır. Kaizen(Yalınlıkta bunun adı sürekli iyileştirmedir), Gözlem ve Adaptasyon(Scrum'da verilen ad), Deneysel Süreç Kontrolü ve Bilimsel Metot, hepsi aynı şeyi ifade etmektedir.

En önemli öğesi *geribildirim döngüsüdür*.

Bir şeyi değiştir => Nasıl bir sonuç verdiğini anla => Öğren => Yeniden bir şeyi değiştir

Genel mantık mümkün olduğunca kısa döngüler tutarak böylece süreci hızlıca adapte etmektir.

Scrum'da temel geribildirim döngüsü Sprint'tir. XP(eXtreme Programming) ile birleştirildiğinde daha fazlası da vardır.



Doğru yapıldığında Scrum + XP çok faydalı geribildirim döngüleri sağlar. En içteki geribildirim döngüsü Eşli Programlama(Pair Programming) birkaç saniyelik bir geribildirim döngüsüdür. Hatalar yapıldıkları andan birkaç saniye sonra bulunur ve düzeltilir.

- Hey, şu değişkenin 3 olması gerekmiyor mu?

Eşli Programlama "Geliştirdiğimiz **iş maddelerini doğru** geliştiriyor muyuz?" geribildirim döngüsüdür.

Sprint, birkaç haftalık geribildirim döngüsüdür. Bu geribildirim döngüsü "**Doğru iş maddelerini** geliştiriyor muyuz?" döngüsüdür.

Kanban'da geribildirim döngüleri? Öncelikle Kanban kullansanız da kullanmasanız da yukarıdaki geri bildirim döngülerini sürecinize ekleyebilirsiniz(ve eklemelisiniz). Kanban'ın size sağladığı şey çok kullanışlı birkaç gerçek zaman metriğidir:

- Ortalama teslim süresi. Her bir iş maddesi "BİTTİ" aşamasına-ya da en sağdaki kolonunuza ne ad verdiyseniz- ulaştığında güncellenir.
- Darboğazlar. Genel semptom, X+1 kolonu boşken X kolonunun iş maddeleriyle tıkanmasıdır.

Gerçek zamanlı metriklerin güzel yanı geri bildirim döngünüzün uzunluğunu belirleyebilmenizdir. Siz hangi sıklıkta metrikleri analiz etmeye ve değişiklik yapmaya gönüllüsünüz? Çok uzun geri bildirim döngüsünün anlamı süreci geliştirmenizin çok yavaş olması anlamına gelir. Çok kısa geri

bildirim döngüsü, iki değişiklik arasında sürecinizin stabil olması için gerekli zamana sahip olmamasıdır.

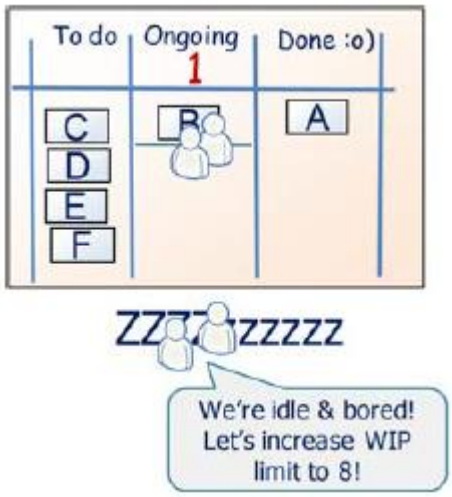
Geri bildirim döngüsünün uzunluğu bile deney yaparak doğru sonuca ulaşabileceğiniz şeylerden biridir. Bir bakıma meta-geri bildirim döngüsüdür.

Peki peki, burada duruyorum ☺

Örnek: Kanban'da Çalışılan İşi Sınırlama Deneyi

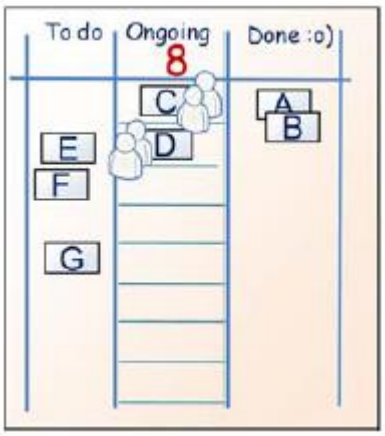
Kanban'ın "önemli noktalarından" biri Çalışılan İş'i sınırlandırmaktır. Çalışılan İş'in sınır noktasının doğru olduğunu nasıl anlarız?

Diyelim ki 4 kişiden oluşan bir takımımız var ve Çalışılan İş limitini 1 olarak belirledik.



Ne zaman bir iş maddesini çalışmaya başlasak ikinci bir maddeye çalışmaya başlayamayız ta ki çalışmaya başladığımız ilk madde BİTTİ aşamasına gelene kadar. Bu madde hızlı bir şekilde BİTTİ aşamasına ulaşır.

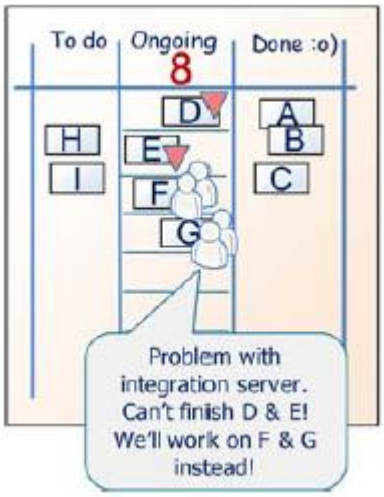
Mükemmel! Fakat sonra şöyle bir sonuçla karşılaşırız. 4 kişinin tamamının aynı iş maddesi üzerine çalışması uygun değildir. İnsanlar iş maddesinin onlara gelmesini beklerken boş oturur. Bu durum bazen oluyorsa problem olmayabilir. Fakat düzenli bir şekilde gerçekleşiyorsa ortalama teslim süresinin artmasına neden olur. Basitçe Çalışılan İş sayısını 1'e düşürmek iş maddelerinin "Devam Eden" işler aşamasını hızlıca geçmesini sağlar fakat bu işler "Yapılacaklar" aşamasında gereğinden fazla beklerler. Yani tüm iş akışı boyunca toplam Teslim Süresi gereksiz yere artacaktır.



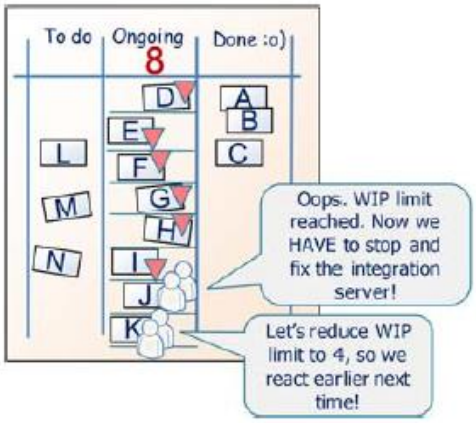
Çalışılan İş sayısının 1 olması takımı çok yavaşlatıyorsa bunu 8'e çıkarırsak ne olur?

Bu bir süre iyi olur. Çiftler halinde çalışmanın ortalamada işlerin daha çabuk bitirilmesini sağladığını ve hız kattığını keşfettik. 4 kişilik bir takımla devam eden iş sayımız her zaman 2'dir. Çalışılan İş sayısı 8 üst limittir yani çalışılmaya devam eden birkaç madde, limiti aşmadığımız için iyidir.

Şimdi düşünün, entegrasyon bilgisayarında bir problem oldu. Bizim BİTTİ Tanımı'mıza(Definition of Done) entegrasyon da dâhildir. Bu nedenle hiçbir maddeyi tamamıyla bitiremeyeceğiz. Bu tarz şeyler kimi zaman oluyor, değil mi?

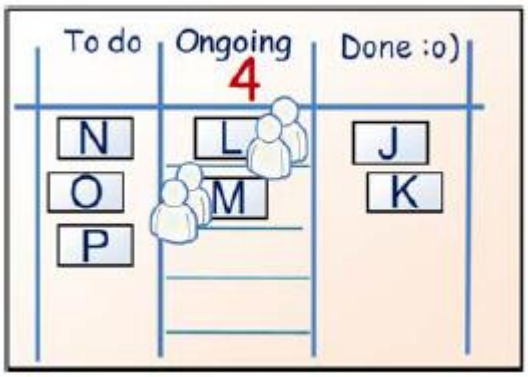


D ve E maddelerini tamamlayamadığımıza göre F maddesine çalışmaya başlayabiliriz. Bunu da entegre edemeyeceğiz. O zaman yeni bir madde alalım, G. Bir süre sonra Kanban limitine ulaşmış oluruz-Devam Eden işler kolonunda 8 madde bulunur-.



Bu noktada daha fazla iş çekemeyiz. Bozuk entegrasyon bilgisayarını düzeltsek iyi olur! Çalışılan İş limiti bizi aksiyon almaya ve bitmeyecek başka işler çekmek yerine darboğazı düzeltmeye zorladı.

Bu iyi bir şeydir. Fakat Çalışılan İş limiti 4 olsaydı, çok daha önce aksiyon alırdık. Bu da bizim ortalama teslim süremizin daha iyi olmasını sağlardı. Yani bu bir dengedir. Ortalama teslim süresini ölçeriz ve teslim süresini iyileştirmek için Çalışılan İş sınırını iyileştirmeye devam ederiz.



Bir süre sonra iş maddelerinin "Yapılacaklar" kolonunda biriktiğini görebiliriz. Belki bu kolona da bir Çalışılan İş limiti eklemeliyizdir.

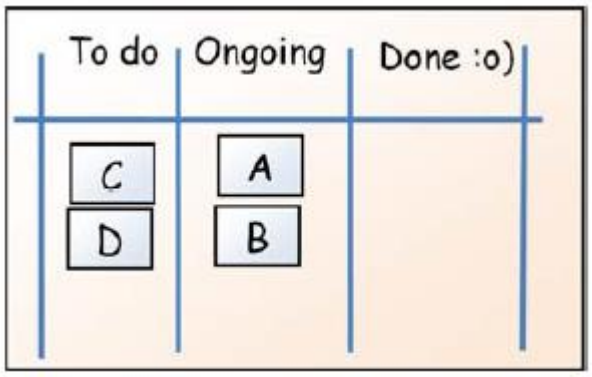
En başta "Yapılacaklar" kolonuna neden ihtiyacımız var? Eğer müşteri takıma sıradaki işin ne olduğunu her zaman söyleyebilecek durumdaysa o zaman "Yapılacaklar" kolonuna ihtiyacımız yok demektir. Fakat bu örnekte müşteri her zaman takımla iletişim halinde değildir. Yani "Yapılacaklar" kolonu müşteri takımdan uzakken takımın çekebileceği işlerin olduğu yerdir.

Deneyin! Ya da Scrum'cıların söylediği gibi, Gözlemleyin ve Adapte olun!

7

Scrum, Sprint İçindeki Değişikliklere Karşıdır

Diyelim ki Scrum Duvarımız aşağıdaki gibi:



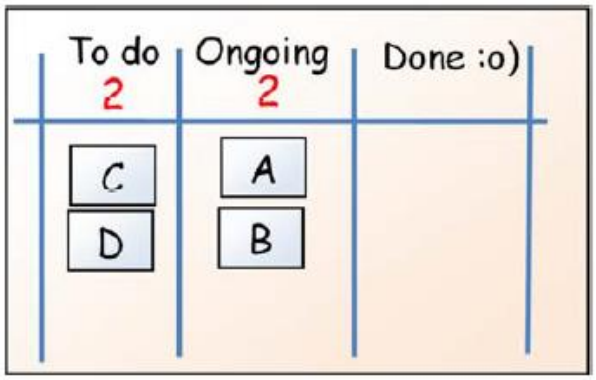
Peki, biri E maddesini eklemek isterse ne olur?

Scrum Takımı şöyle bir şey söyler.

- "Kusura bakmayın. Bu Sprint'te A+B+C+D maddelerini taahhüt ettik. E'yi Ürün İş Listesi'ne ekleyebilirsiniz. Eğer Ürün Sahibi bunun öncelikli bir madde olduğunu düşünür ve önceliğini artırırsa önümüzdeki Sprint içeri alabiliriz".

Doğru uzunluktaki Sprint'ler takımın odaklanıp bir şeyleri bitirebilmesi için yeterli uzunluktadır. Aynı şekilde Ürün Sahibi'ne öncelikleri yönetmesi ve güncellemesi için düzenli bir şekilde zaman yaratılmış olur.

Peki, bu soruya Kanban Takımı ne cevap verirdi?



E maddesini “Yapılacak” işler kolonuna ekleyebilirsin. Fakat bu kolonun limiti 2 bu nedenle C veya D’yi bu kolondan çıkarman gerekiyor. Şuan A ve B maddeleri üzerinde çalışıyoruz. Bunlardan biri biter bitmez “Yapılacak” işler kolonunun en üstündeki maddeyi çekeceğiz.

Yani Kanban takımının cevap süresi –önceliklerde bir değişiklik olduğunda bu değişikliğe verilecek cevabın aldığı süre- kapasitenin açılmasına bağlıdır. Kapasitenin açılması ise “bir madde bitti = bir madde içeri alındı” prensibine dayanır ki bunu da Çalışılan İş limiti belirler.

Scrum’da cevap süresi ortalama olarak Sprint uzunluğunun yarısı kadar bir zamandır.

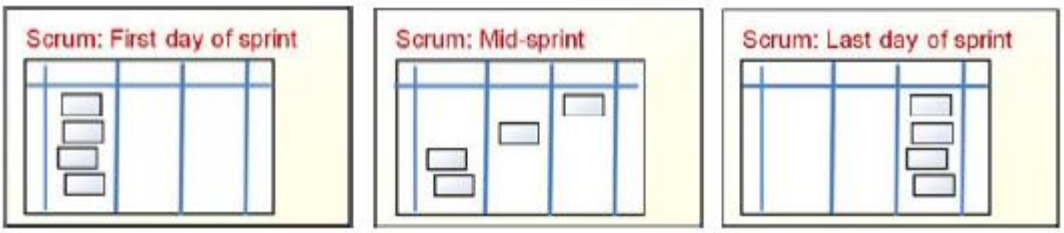
Scrum’da Ürün Sahibi, Geliştirme Takımı belirli iş maddelerini geliştirmeye taahhüt ettikten sonra Sprint içinde değişiklik yapamaz. Kanban’da değişikliği kimin yapabileceğine ya da yapamayacağına siz karar vermelisiniz. Bunun için gerekli yapıyı siz kurmalısınız. Genellikle Ürün Sahibi’ne Kanban Duvarı’nda en soldan bir kolon verilir. Bu kolonun adı “Yapılacaklar”, “Hazır”, “İş Listesi” ya da “Teklif Edilenler” olur. Ürün Sahibi bu kolon içindeki maddelerin önceliklerini istediği zaman değiştirebilir.

Bu iki yaklaşım uygulamada birbirinden çok farklı olmayabilir. Scrum Takımı, Ürün Sahibi’ne Sprint ortasında değişiklikleri yapma izni verebilir. Tabi ki bu bir istisnadır. Kanban Takımı da önceliklerin ne zaman değiştirilebileceğine dair sınırlamalar belirleyebilir. Hatta Kanban Takımı, Scrum’da olduğu gibi süresi belirli döngülerde kullanabilir.

8

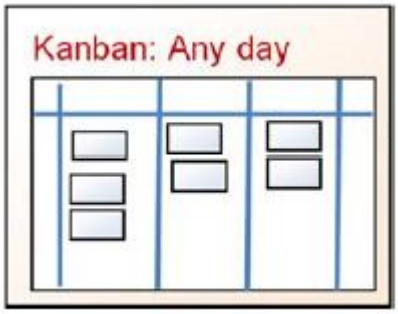
Scrum Duvarı her Sprint de Yeniden Oluşturulur

Scrum Duvarı, Sprint'in farklı evrelerinde aşağıdaki gibi görünebilir.



Sprint bittiğinde, Duvar temizlenir, bütün iş maddeleri kaldırılır. Yeni Sprint başlar ve Sprint Planlama Toplantısı'ndan sonra en soldaki kolonda yepyeni maddelerin olduğu yeni bir Scrum Duvarı'mız olur. Teknik bakımdan bu bir çöptür. Tabi ileri seviye Scrum Takımlar'ın da bu fazla bir zaman almaz. Ayrıca Duvarın yeniden oluşturulması bir başarı ve bir şeyi tamamlamış olmanın hoş duygusunu verir. Tıpkı yemekten sonra bulaşıkları yıkamak gibi, yapmak can sıkıcıdır fakat yaptıktan sonra bir rahatlama hissedersiniz.

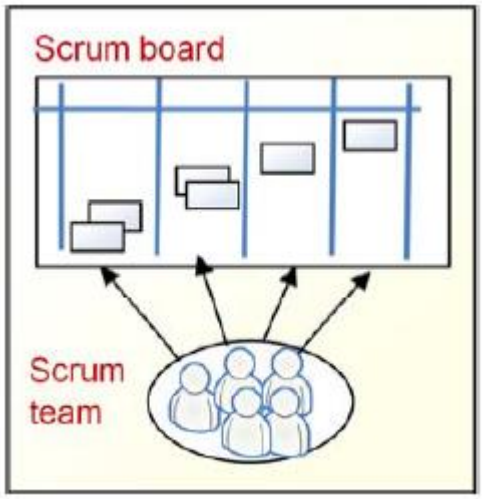
Kanban'da ise Duvar kalıcıdır. Duvarı yeniden oluşturmaz ve her şeye en baştan başlamazsınız.



9

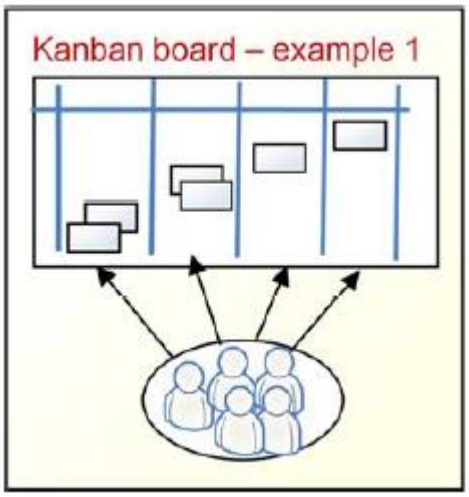
Scrum'da Çapraz Fonksiyonel Takımlar Kuraldır

Scrum Duvarı, sadece bir Scrum Takımı'na aittir. Scrum Takımı, çapraz fonksiyoneldir ve iterasyon içindeki iş maddelerinin tamamlanabilmesi için gerekli olan tüm yetkinliklere sahip üyelerden oluşur. Scrum Duvarı herkes tarafından görülebilir bir durumda olmalıdır fakat onu değiştirebilecek kişi, sahibi olan Scrum Takımı'dır. Duvar, Scrum Takımı'nın bu döngüde taahhüt ettiği işleri geliştirmek için kullandığı bir araçtır.

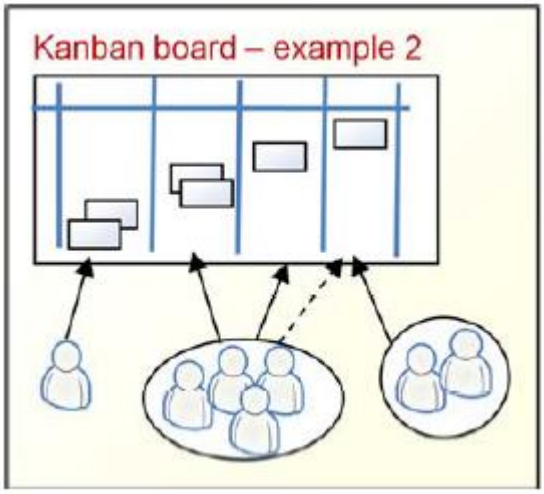


Kanban'da çapraz fonksiyonel takımlar seçime bağlıdır ve Kanban Duvarı belirli bir takım tarafından sahiplenilmez. Kanban Duvarı iş akışıyla ilişkilidir, takımla değil.

Örnekler:



Örnek 1: Tüm Duvar bir çapraz fonksiyonel takıma hizmet eder, tıpkı Scrum gibi.



Örnek 2: Ürün Sahibi birinci kolondaki maddelerin önceliklerini belirler. Çapraz fonksiyonel geliştirme takımı ikinci kolona bakarak geliştirmeyi, üçüncü kolona bakarak testi yapar. Dördüncü kolon teslim kolonudur ve buradaki iş maddeleri uzman takım tarafından yapılır. Yetkinlikler yaygınlaştırılmıştır ve teslim takımı(release team) darboğaz olduğunda geliştiricilerden biri teslim takımına yardım edecektir.

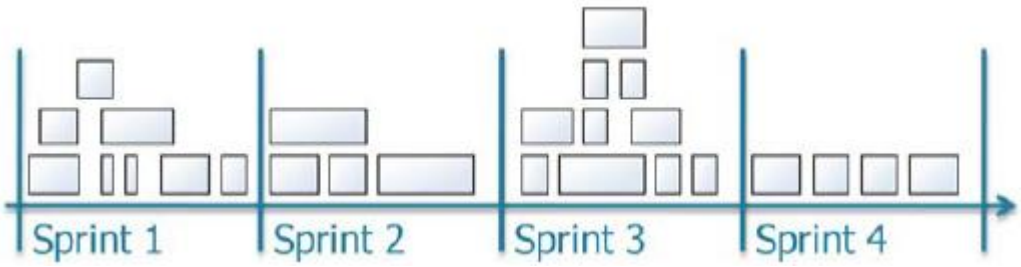
Yani Kanban'da Duvarı kimin, nasıl kullanacağına dair bazı kurallar oluşturmanız gerekir. Daha sonra bu kuralları uygulayarak deney yapmalı ve akışı iyileştirmeye çalışmalısınız.

10

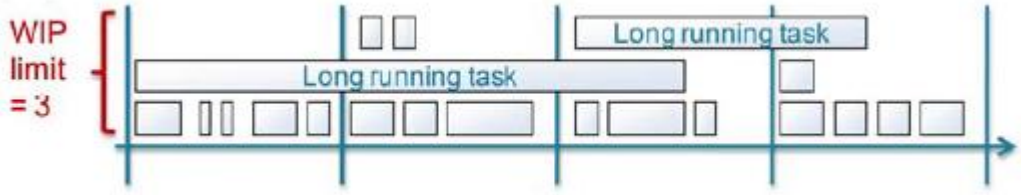
İş Maddeleri Bir Sprint'e Sığmak Zorundadır

Scrum ve Kanban artımlı(incremental) geliştirmeyi temel alır, yani işi daha küçük parçalara bölmeyi.

Scrum Takımı sadece bir iterasyonda yetiştirebileceğini düşündüğü maddelere taahhüt verir. Bunu yaparken BİTTİ Tanımı'nı göz önünde bulundurur. Eğer bir iş maddesi bir Sprint'e sığmayacak kadar büyükse Takım ve Ürün Sahibi bu iş maddesini daha küçük parçalara bölmeye çalışır.



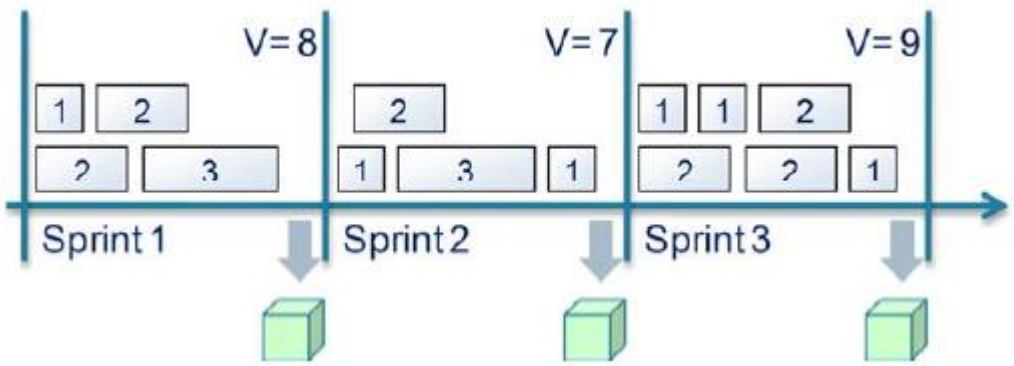
Kanban takımları teslim süresini azaltmaya, iş akışını pürüzsüzleştirmeye çalışırlar. Bu da dolaylı olarak iş maddelerini daha küçük parçalara bölmeye teşvik eder. Fakat bir iş maddesi belirli bir süreye sığmalıdır şeklinde bir kural bulunmaz. Duvar'da bir iş maddesini tamamlamak için 1 ay, başka bir iş maddesini tamamlamak için 1 gün süre verilebilir.



11

Scrum'da İş Maddelerinin Büyüklük Tahmini ve Takımın Hızı Önemlidir

Scrum'da takımlar iş maddelerinin büyüklüklerini-işin miktarını- göreceli olarak tahmin etmeye çalışırlar. Bir Sprint'te bitirilen işlerin büyüklükleri toplanarak takımın hızı hesaplanır. Takım hızı, takımın kapasitesidir, bir Sprint'te ne kadar iş teslim edebildiğimizi gösteren ölçüdür. Aşağıda ortalama takım hızı 8 olan bir takımın örneği bulunuyor.



Ortalama takım hızının 8 olduğunu bilmek güzel bir şey çünkü gelecek Sprint'ler de hangi işleri tamamlayabileceğimiz konusunda gerçekçi tahminler yapabiliriz. Bu da gerçekçi teslim planları yapmamızı sağlar.

Kanban'da işlerin büyüklüğünü tahmin etmek bir kural değildir. Bu nedenle taahhüt sizin için bir gereklilikse tahmin edilebilirliği sağlamak için ne yapmanız gerektiğine karar vermelisiniz. Kimi Kanban Takımları tıpkı Scrum Takımları'nın yaptığı gibi işlerin büyüklüğünü tahmin ederler ve takım hızını ölçerler. Kimi takımlarsa işlerin büyüklüğünü tahmin etme işini geçip iş maddelerini benzer büyüklükte iş parçacıklarına bölmeye çalışırlar. Daha sonra belirli bir süre içinde tamamlanan iş maddesi sayısını sayarak takım hızını ölçerler. Örneğin bir haftada 9 iş maddesinin

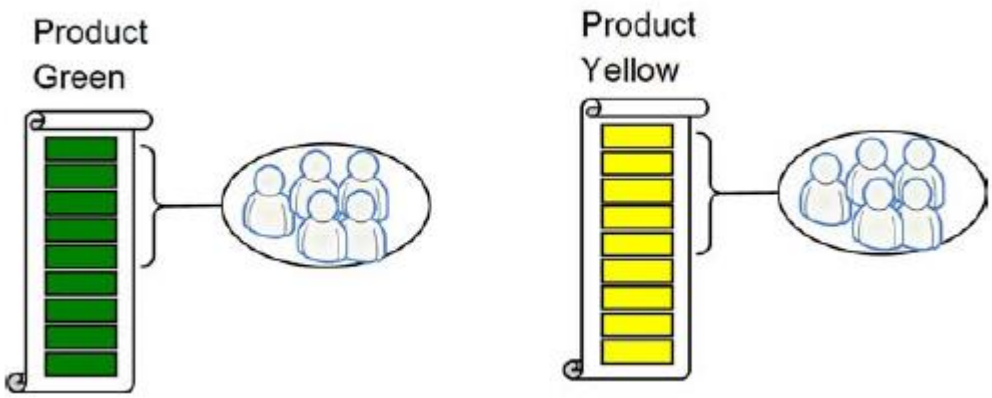
tamamlanması. Bazı takımlar iş maddelerini, MMF bazında gruplar ve bir MMF için harcanan ortalama teslim süresini hesaplarlar. Bu veri sayesinde SLA'lerini belirlerler.

Kanban stilinde teslim planlaması ve taahhüt yönetimi için birçok teknik bulunmaktadır. Kanban'la gelen herhangi bir teknik kural yoktur. Yani kendi tekniğinizi kendiniz oluşturabilirsiniz. Farklı teknikleri internette arayıp deneyebilir, sizin için en uygununu seçebilirsiniz. Zamanla iyi pratiklerin ortaya çıktığını göreceksiniz.

12

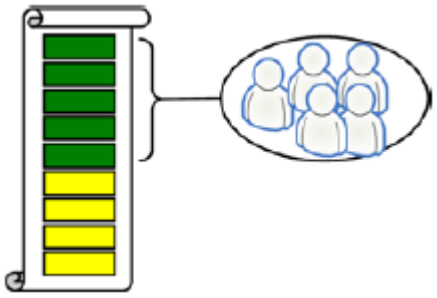
Scrum ve Kanban Aynı Anda Birçok Ürün Üzerinde Çalışmaya İzin Verir

Scrum'da "Ürün İş Listesi" bahtsız bir isimlendirmedir. Çünkü tüm iş maddelerinin aynı ürün hakkında olması gerektiğini ima eder. İşte iki farklı Ürün İş Listesi, biri yeşil, diğeri sarı ve her ikisinin de kendi takımları bulunuyor.

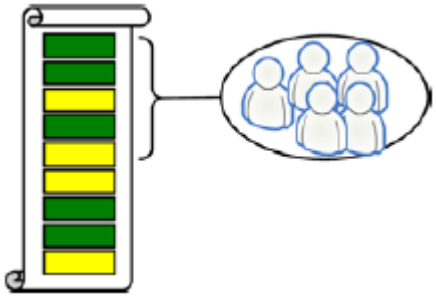


Ya bir takımınız varsa? O zaman Ürün İş Listesi'ni, Takım İş Listesi gibi düşünebilirsiniz. Bu, belirli bir takımın ya da takım grubunun gelecek iterasyonlar için iş maddelerinin önceliklendirilmiş halidir. Yani takım birçok ürünün bakımıyla ilgileniyorsa bu ürünlerle ilgili işleri bir Ürün İş Listesi'nde birleştirebilirsiniz. Bu bizi ürünler arasında da önceliklendirmeye zorlar ki bu bazı durumlarda kullanışlı olabilir.

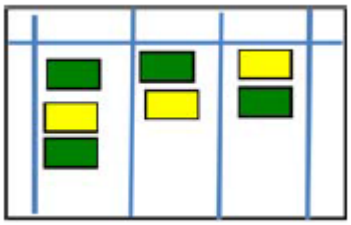
Pratikte bunu yapmanın birçok yolu vardır. Bir yolu takımın bir Sprint boyunca sadece bir ürüne odaklanmasıdır:



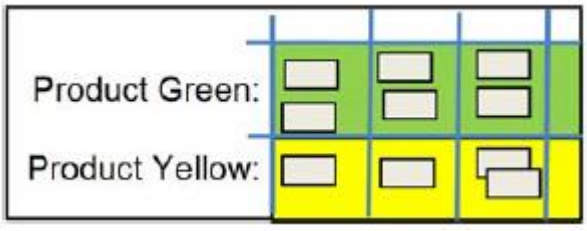
Diğer bir yoluysa takımın her iki üründen iş maddelerini çalışmasıdır:



Kanban'da da aynıdır. Aynı Duvar üzerinde birkaç ürünün akışı olabilir. Farklı renkte kartlar kullanarak birbirinden ayırabiliriz:



...ya da satırlara ayırabiliriz:



13

Scrum ve Kanban, İkisi de Yalın ve Çevik'tir

Burada Yalın Düşünce ya da Çevik Bildiri'yi anlatmayacağım. Genel olarak bahsetmek gerekirse ikisi de, Scrum da, Kanban da Yalın Düşünce ve Çevik Bildiri'de bulunan değerler ve ilkelerle aynı çizgidedir. Örneğin:

- Scrum ve Kanban, ikisi de iş çekme mantığına dayalıdır. Bu Yalınlığın anında üretim(Just In Time, JIT) ilkesine dayanır. Bunun anlamı ne kadar sürede ne kadar işi yapacağını takımın seçmesidir. Takım dışından bir etkiyle işin takıma atanmasından öte iş hazır olduğunda takım işi "çeker". Tıpkı yazıcının bastırılacak doküman hazır olduğunda sıradaki sayfayı çekmesi gibi.
- Scrum ve Kanban'ın temelinde sürekli ve deneysel süreç iyileştirmesi vardır. Bu Yalınlık ilkelerinden Kaizen'e karşılık gelmektedir.
- Scrum ve Kanban bir planı takip etmektense değişime karşılık vermeyi vurgular. Gerçi Kanban, Scrum'dan daha hızlı bir cevap süresine sahiptir. Çevik Bildiri'nin dört ilkesinden biri...

...ve daha fazlası vardır.

Bir açıdan Scrum, o kadar yalın görünmeyebilir çünkü iş maddelerini süresi belirli iterasyonlar içinde toplar. Tabi bu yalınlık iterasyon(Sprint) uzunluğuna ve neyle kıyasladığınıza göre değişkenlik gösterebilir. Yılda 2 – 4 defa bir şeyleri entegre edip, teslim yaptığınız geleneksel süreçlerle karşılaştırıldığında her 2 haftada üretim ortamına alınabilir kod geliştiren Scrum Takımı oldukça yalındır.

Ayrıca iterasyon süresini kısalttıkça Kanban'a yaklaşırsınız. İterasyonları 1 haftadan daha kısa sürelere indirmeyi konuştuğunuz zaman süresi belirli iterasyonları kaldırmayı düşünebilirsiniz.

Daha önce söyledim ve söylemeye devam edeceğim: Sizin için uygun olanı bulana dek deney yapın! Daha sonra deney yapmaya devam edin. :o)

14

Küçük Farklar

Şimdi anlatacaklarım önceki bölümlerde söylediklerime göre daha az ilgi çekici olabilir. Yine de bunları bilmek iyi olur.

Scrum'da Önceliklendirilmiş İş Listesi Kuraldır

Scrum'da Ürün İş Listesi sıralanarak öncelik belirlenir ve önceliklerdeki bir değişiklik gelecek Sprint'i etkiler, içinde bulunulan Sprint'i değil. Kanban'da herhangi bir önceliklendirme tekniğini kullanabilirsiniz, hatta hiç kullanmayabilirsiniz. Önceliklerdeki değişiklikse kapasite uygun olur olmaz etki eder. Bir Ürün İş Listesi olabilir de, olmayabilir de, Ürün İş Listesi önceliklendirilmiş olabilir de olmayabilir de.

Pratikteyse önceliklendirmedeki farkın küçük bir etkisi vardır. Kanban Duvarı'nda en soldaki kolon ile Scrum'daki Ürün İş Listesi'yle aynı amaca hizmet eder. *İş Listesi* öncelik göz önünde bulundurularak sıralanmış olsun ya da olmasın takımın işi çekerken hangi işi ilk önce çekeceğine dair bir kural oluşturması gerekir.

Örnek kurallar:

- Her zaman en üstteki iş maddesini al
- Her zaman en eski iş maddesini al, burada her maddenin üzerinde Duvar'a eklendiği tarih bulunur
- Herhangi bir iş maddesini al
- Yaklaşık olarak zamanın %20'sini bakım işlerine, %80'ini yeni özellikler geliştirmeye harca
- Takımın kapasitesini Ürün A ve Ürün B için eşit olarak böl
- Eğer kırmızı madde varsa öncelikle onu al

Scrum'da Ürün İş Listesi de Kanbanvari bir şekilde kullanılabilir. Ürün İş Listesi'nin büyüklüğünü limitleyebilir ve nasıl önceliklendirileceğine dair kurallar koyabilirsiniz.

Scrum'da Günlük Toplantılar Kuraldır

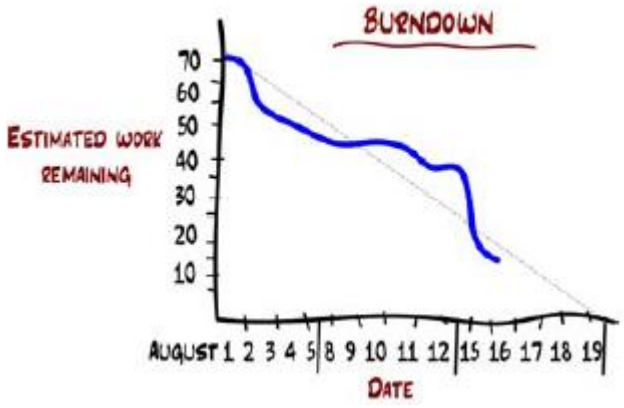
Bir Scrum Takımı her gün aynı yer ve saatte 15 dakikalık bir toplantı yapar. Bu toplantının amacı neler olup bittiğine dair bilgiyi yaymak, günün planlamasını yapmak ve varsa problemleri belirlemektir. Bu toplantı genelde ayakta yapıldığı için günlük ayakta toplantı olarak anılır. Ayakta yapılmasının nedeni kısa sürede toplantıyı tamamlamak ve toplantı süresince enerjiyi üst seviyede tutmaktır.

Günlük ayakta toplantılar Kanban'da kural değildir. Yine de birçok Kanban takımı bu pratiği kullanmaktadır. Hangi süreci kullandığınızın önemi yok, bu mükemmel bir tekniktir.

Scrum'da bu toplantının formatı insan odaklı olmasıdır-herkes sırayla rapor verir. Birçok Kanban takımı daha çok Duvar odaklıdır, darboğazlara ve görünür diğer problemleri ön plana çıkarırlar. Aynı Duvarı paylaşan 4 takım varsa ve günlük toplantılarını beraber yapıyorlarsa Kanban tarzı bir yaklaşım daha uygulanabilir görünüyor. Duvar'daki dar boğazları çözmeye odaklandığımız sürece herkesin tek tek konuşması ve rapor vermesi gereksiz olabilir.

Scrum'da Burn-down Grafikler Kuraldır

Burn-down grafik içinde bulunulan döngüde ne kadar iş kaldığını gösteren grafiktir.



Y ekseninin birimi iş maddeleri tahmin edilirken kullanılan birimle aynıdır. Bu birim genelde saat, gün ya da hikaye puanı olur. Eğer takım iş maddelerini daha küçük parçacıklara bölüyorsa- task'lara- saat ya da gün kullanılır. Eğer daha küçük parçalara bölmüyorsa hikaye puanı da kullanılabilir, bunun birçok farklı versiyonu bulunmaktadır.

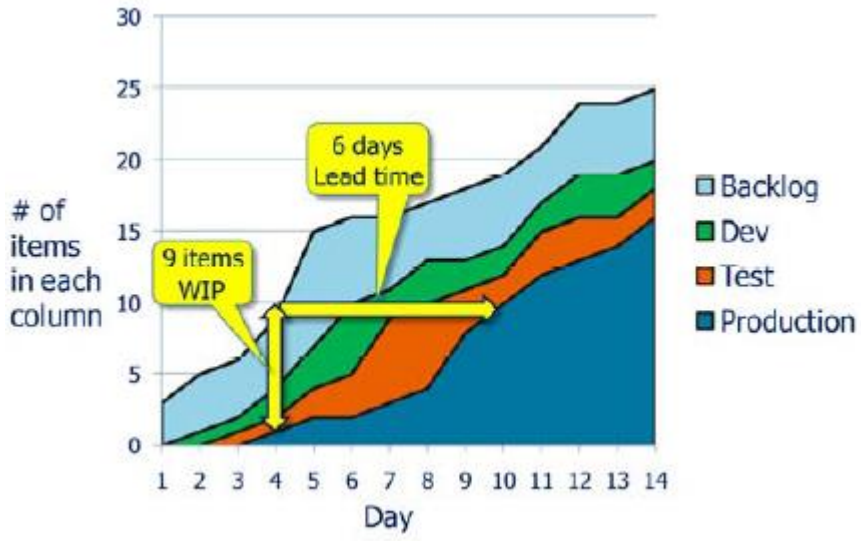
Scrum'da Sprint burn-down grafik bir döngünün takibini yapmak için kullanılan başlıca araçtır.

Bazı takımlar aynı zamanda teslim burn-down grafiklerde kullanır. Bu grafiklerde aynı formattadır fakat teslim seviyesinde oluşturulurlar. Genellikle her Sprint'ten sonra Ürün İş Listesi'nde ne kadarlık hikaye puanı kaldığını gösterir.

Burn-down grafiğin temel amacı yaptığımız planlamanın ilerisinde ya da gerisinde olduğumuzu mümkün olduğunca erken ve kolay öğrenmektir. Böylece gerekli adaptasyonu gerçekleştirebiliriz.

Kanban'da burn-down grafikler kural değildir. Aslında kural olan herhangi bir grafik yoktur. Elbette istediğiniz grafiği kullanmakta özgürsünüz, burn-down dahil.

Aşağıda Kümülatif Akış diyagram örneği bulunuyor. Bu tarz bir grafik akışınızın ne kadar pürüzsüz olduğunu ve Çalışılan İş sayısının teslim süresini nasıl etkilediğini çok güzel gösterir.



Her gün Kanban Duvarı'nda bulunan tüm kolonlardaki iş maddelerinin sayısı toplanır. Bu değer Y ekseninde gösterilir. Yani 4. günde Duvar'da 9 iş maddesi bulunur. En sağdaki kolondan başlayarak sayılır, Üretim kolonunda 1, Test kolonunda 1, Geliştirme kolonunda 2, Yapılacaklar kolonunda 5 iş

maddesi bulunur. Eğer her gün bu noktaların yerini belirlersek ve noktaları birleştirirsek yukarıdaki gibi güzel bir diyagramımız olur. Yatay ve dikey oklar Çalışılan İş ve teslim süresi arasındaki ilişkiyi gösterir.

Yatay ok, 4. günde Yapılacaklar kolonuna eklenen iş maddelerinin ortalama 6 günde Üretim kolonuna erişebileceklerini gösterir. Bu zamanın neredeyse yarısı Test kolonunda harcanır. Eğer Test ve Yapılacaklar kolonunda limit olsaydı toplam teslim süremizi önemli ölçüde düşürebileceğimizi görebiliriz.

Koyu mavi bölgenin eğimi takım hızını-örneğin her gün teslim edilen iş maddesi sayısını- gösterir. Zaman içinde yüksek takım hızının teslim zamanını nasıl düşürdüğünü, yüksek Çalışılan İş sayısının teslim zamanını nasıl artırdığını göreceğiz.

Birçok organizasyon işlerini daha çabuk bitirmek-teslim zamanını düşürmek- ister. Ne yazık ki birçoğu bunun anlamının daha fazla kişiyle çalışmak ya da daha uzun saatler çalışmak olduğunu sanma tuzağına düşerler. Bir şeyleri hızlıca bitirmenin en etkili yolu pürüzsüz bir akış ve Çalışılan İş sayısını sınırlamaktır, daha fazla kişiyi projeye dahil etmek ya da daha uzun saatler çalışmak değil.

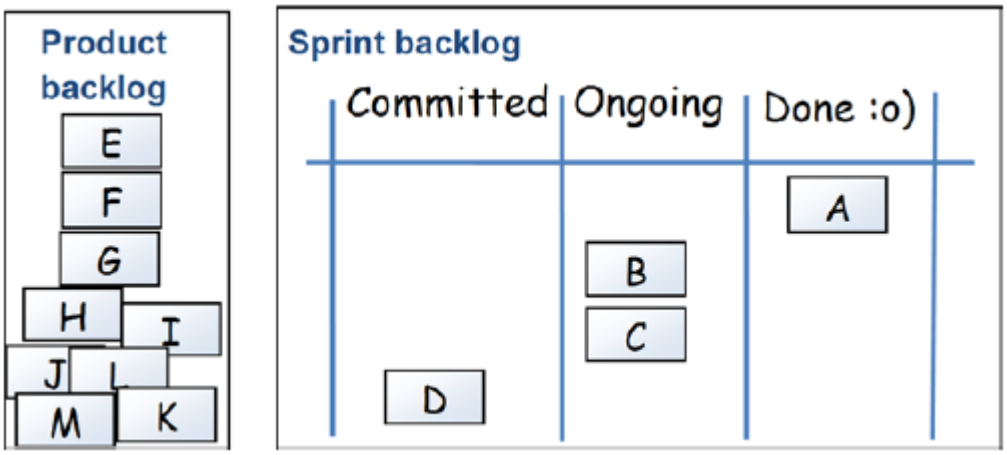
Kuyrukta bekleyenleri-test için bekleyenler- ve çalışılanları-test edilenler- şeklinde ayırmaya başlarsak çok daha net bir hal almaya başlıyor. Kuyrukta bekleyen iş miktarını en düşük seviyede olmasını isteriz ve kümülatif diyagram bunun için gerekli bilgiyi sağlıyor.

15

Scrum Duvarı, Kanban Duvarı'na Karşı

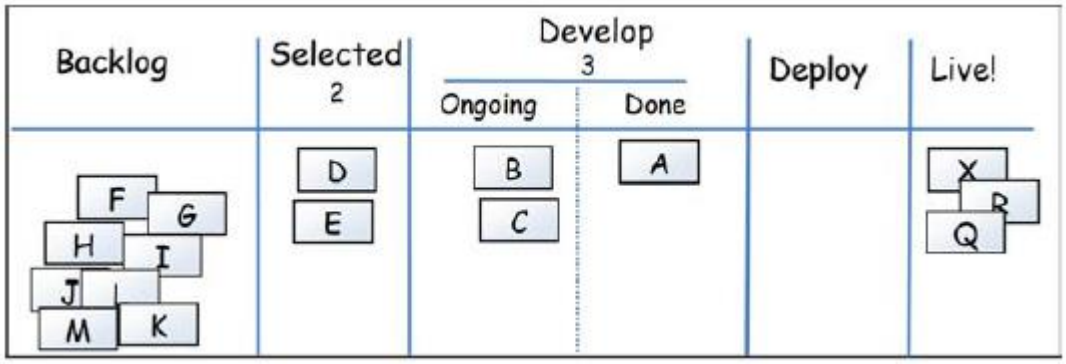
Sprint İş Listesi, resmin sadece bir parçasıdır- içinde bulunulan Sprint'te takımın ne yaptığını gösteren parça. Diğer parçasıysa *Ürün İş Listesi*'dir-Ürün Sahibi'nin gelecek Sprint'lerde sahip olmak istediği şeylerin listesi.

Ürün Sahibi, **Sprint İş Listesi**'ni görebilir fakat dokunamaz. Ürün Sahibi, **Ürün İş Listesi**'ni istediği zaman değiştirebilir fakat yaptığı değişiklik ancak gelecek Sprint'te etkisini gösterir.



Sprint bittiğinde, Geliştirme Takımı üretim ortamına alınabilir işlevsellikleri teslim eder. Geliştirme Takımı, Sprint'i tamamlar ve Sprint Değerlendirme Toplantısı'nı yapar ve gururla tamamladığı A, B, C ve D özelliklerini Ürün Sahibi'ne sunar. Bu aşamadan sonra Ürün Sahibi geliştirilen bu özelliklerin üretim ortamına alınıp alınmamasına karar verir. Son bölüm –geliştirilen özellikleri üretim ortamına alma bölümü- genellikle Sprint içine dahil edilmez, bu nedenle Sprint İş Listesi'nde bulunmaz.

Bu senaryoda Kanban Duvarı aşağıdaki gibi görünebilir:



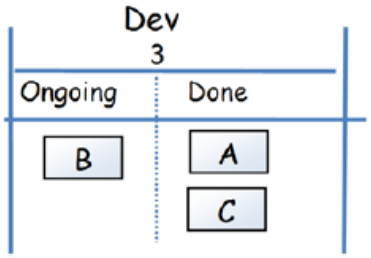
Şimdi tüm iş akışı Duvar'da, sadece bir Scrum Takımı'nın bir iterasyon içinde yaptığı işe bakmıyoruz.

Yukarıdaki örnekte "İş Listesi" kolonu özel bir sıralaması olmayan sadece isteklerin bulunduğu listedir. Seçilenler(Selected) kolonu önceliği yüksek iş maddelerini içerir. Bu kolonun limiti 2'dir. Yani bu kolonda herhangi bir anda sadece 2 tane yüksek öncelikli iş maddesi bulunabilir. Takım ne zaman yeni bir iş maddesi üzerinde çalışmaya hazır olursa "Seçilenler" kolonunda en üstte bulunan iş maddesini çeker. Ürün Sahibi, "İş Listesi" ve "Seçilenler" kolonlarında istediği zaman değişiklik yapabilir fakat diğer kolonlarda değişiklik yapamaz.

"Geliştirme" kolonu iki alt kolona bölünür, Geliştirme kolonunun limiti 3'tür. Şuanda geliştirilen işlerin gösterildiği kolondur. Network terimiyle Kanban limiti, "bant genişliğine", teslim süresiyse "ping" yani cevap süresine denk gelir.

Neden "Geliştirme" kolonunu "Devam Eden" ve "Bitti" diye ikiye böldük? Bu, teslim ekibinin neyi üretim ortamına alabileceklerini daha kolay görmeleri için yapılır.

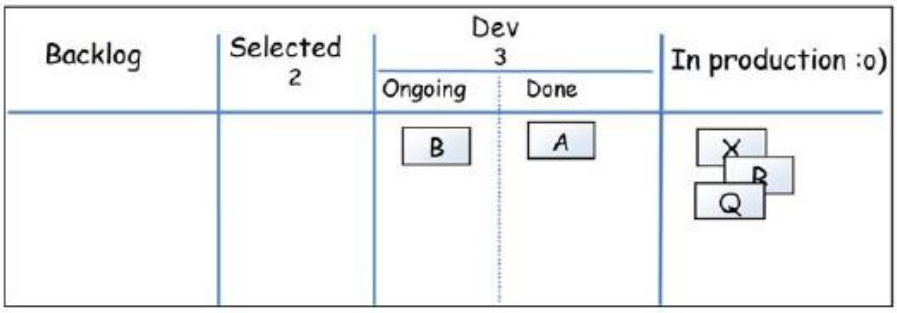
"Geliştirme" kolonunun 3 limiti iki alt kolon tarafından paylaşılır. Neden? Diyelim ki "Bitti" kolonunda 2 tane iş maddesi buluyor.



Bunun anlamı “Devam Eden” kolonunda sadece 1 iş maddesi bulunabilir olmasıdır. Bunun anlamı da geliştiricilerden biri yeni bir iş maddesine başladığında kapasite aşılıyor demektir. Fakat Kanban limitinden dolayı geliştiricilerin yeni bir işe başlamaya izni yoktur. Bu, takımı harekete geçirir ve teslimi yapacak kişilerden bir şeyleri hemen üretim ortamına alabilmek için yardım isterler. Böylece “Bitti” kolonunu temizlemiş ve akışı en yüksek seviyeye çıkarmış olurlar. Bu etki iyidir ve karşılıklıdır. Ne kadar fazla iş maddesi “Bitti” kolonuna girerse o kadar az iş “Devam Eden” işler kolonuna girebilir. Buda takımın doğru noktaya odaklanmasına yardım eder.

Tek Parça Akış

Tek parça akış, bir iş maddesinin herhangi bir kuyrukta beklemeden iş akışını uçtan uca geçmesidir, bir bakıma mükemmel akış senaryosudur. Bunun anlamı her an birileri bu iş maddesi üzerinde çalışıyor demektir. Bu durumda Kanban Duvarı aşağıdaki gibi görünebilir:

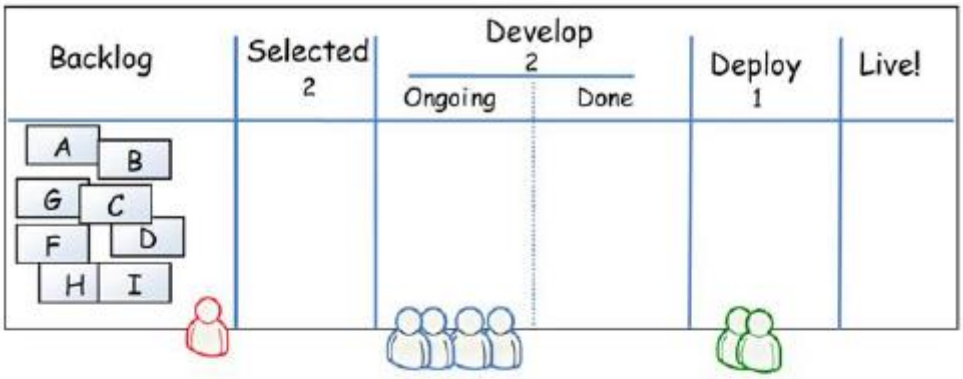


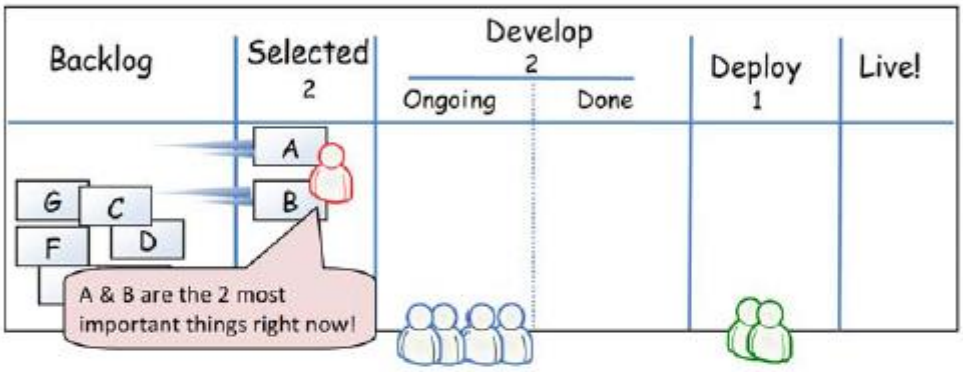
Şuan B maddesi geliştiriliyor, A maddesi üretim ortamına alınıyor. Takım ne zaman yeni bir iş maddesine hazır olsa Ürün Sahibi’ne en önemli maddenin hangisi olduğunu sorar ve hemen cevap alır. Eğer bu ideal senaryo devam ederse iki kuyruk kaldırılabilir, “Yapılacaklar” ve “Seçilenler” ve gerçekten çok çok kısa bir teslim süresi elde edilebilir.

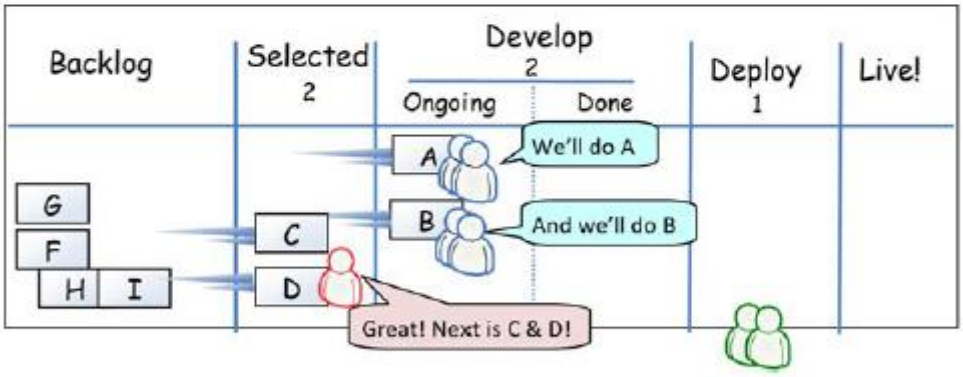
Cory Ladas bunu çok güzel ifade eder: “İdeal çalışma süreci, geliştirme takımına sıradaki çalışılacak en iyi şeyi sağlamalı, ne daha fazlası ne daha azı”.

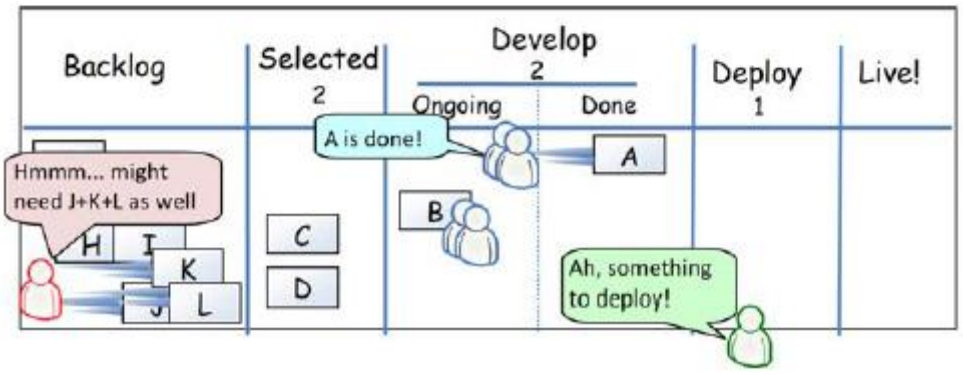
Çalışılan İş limitleri problemlerin çığırından çıkmasını engellemek için vardır. Yani iş maddeleri pürüzsüz bir şekilde akıyorsa Çalışılan İş limitleri gerçekte kullanılmıyordur.

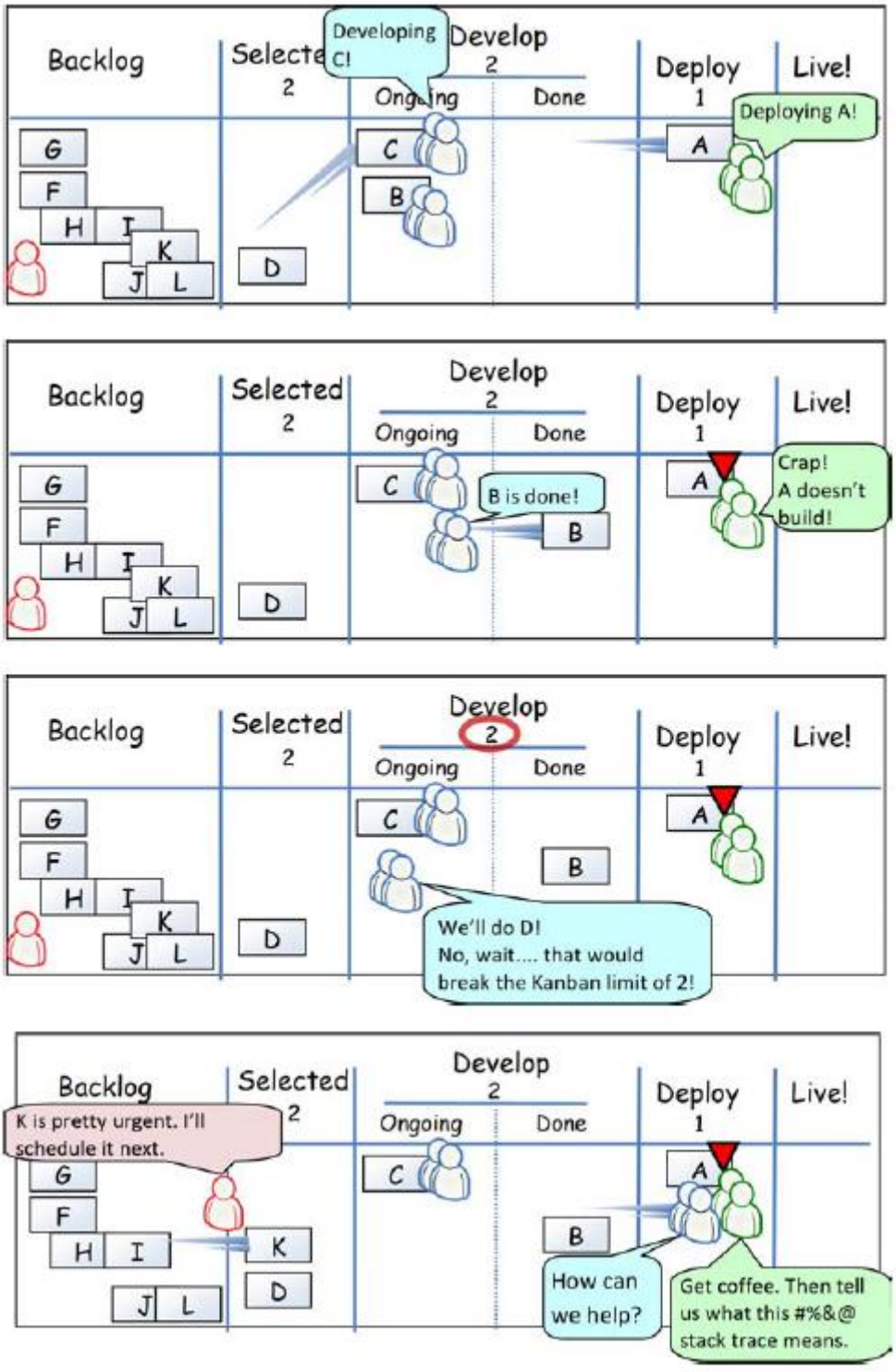
Kanban'da Bir Gün

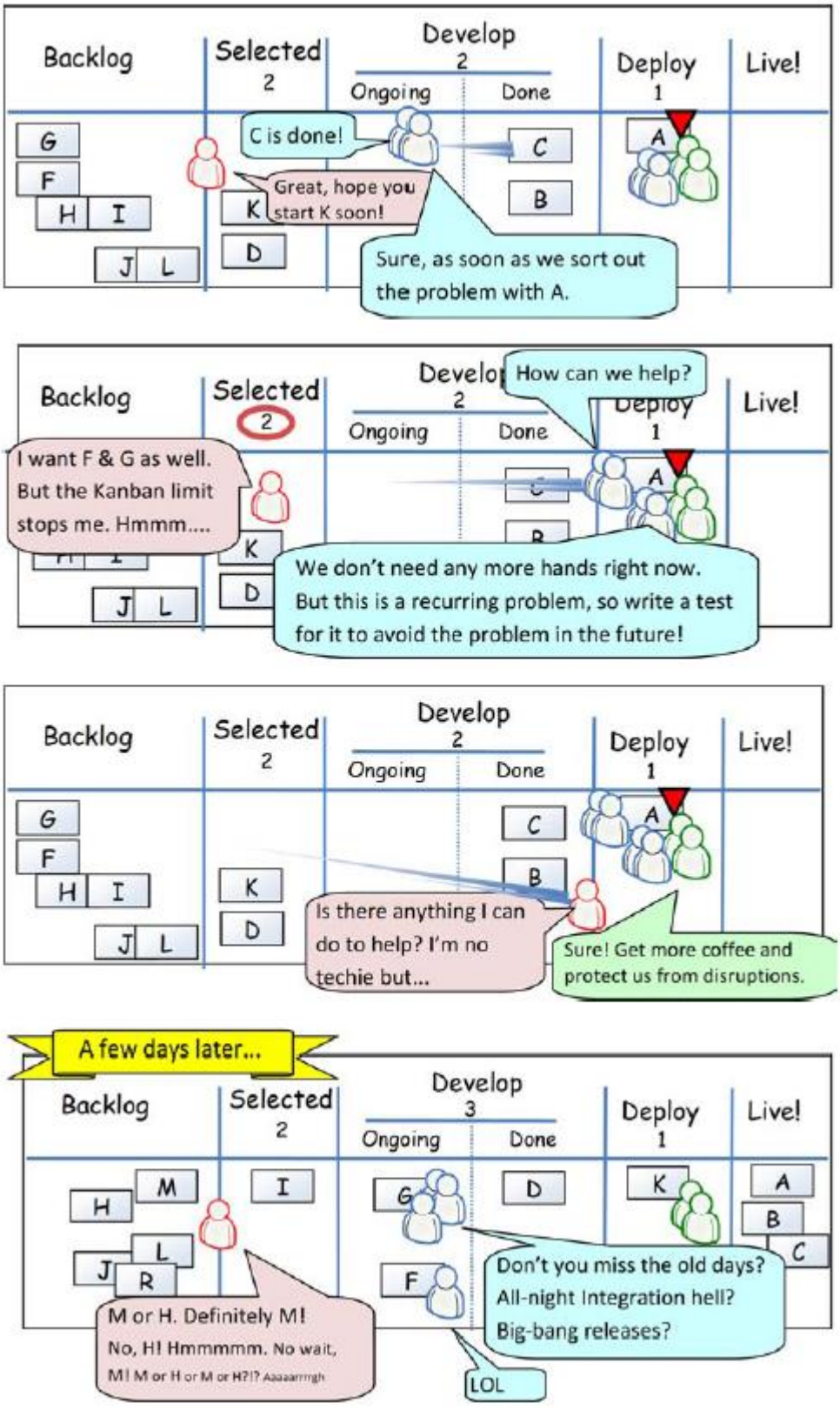









Backlog
Selecte
2
Developing C!
Develop
2
Ongoing
Done
Deploy
1
Live!
Deploying A!
G
F
H
I
K
J
L
D
C
B
A
Backlog
Selected
2
Develop
2
Ongoing
Done
Deploy
1
Live!
B is done!
Crap!
A doesn't
build!
Backlog
Selected
2
Develop
2
Ongoing
Done
Deploy
1
Live!
We'll do D!
No, wait.... that would
break the Kanban limit of 2!
Backlog
Selected
2
Develop
2
Ongoing
Done
Deploy
1
Live!
K is pretty urgent. I'll
schedule it next.
How can
we help?
Get coffee. Then tell
us what this #%&@
stack trace means.


Backlog
Selected 2
Develop 2
Ongoing
Done
Deploy 1
Live!
G
F
H
I
J
L
K
D
C
B
A
C is done!
Great, hope you start K soon!
Sure, as soon as we sort out the problem with A.
How can we help?
I want F & G as well. But the Kanban limit stops me. Hmmm....
We don't need any more hands right now. But this is a recurring problem, so write a test for it to avoid the problem in the future!
Is there anything I can do to help? I'm no techie but...
Sure! Get more coffee and protect us from disruptions.
A few days later...
Develop 3
M
R
M or H. Definitely M!
No, H! Hmmmmm. No wait, M! M or H or M or H?!? Aaaaarrrrgh
Don't you miss the old days? All-night Integration hell? Big-bang releases?
LOL

Kanban Duvarı Bu Şekilde Olmak Zorunda Mı?

Hayır! Yukarıdaki Duvar sadece bir örnek!

Kanban'ın kural olarak belirlediği şey, iş akışının görsel olması ve Çalışılan İşin sınırlanmasıdır. Amaç, sistem boyunca pürüzsüz bir akış oluşturmak ve teslim süresini azaltmaktır. Bu nedenle aşağıdakilere benzer soruları kendinize düzenli olarak sormalısınız:

Kanban Duvarı'mızda Hangi Kolonlar Olmalı?

Her kolon iş akışındaki bir adımı ya da bir kuyruğu-iş akışındaki iki aşama arası- ifade etmektedir. Basitten başlayın ve gerektikçe yeni kolonlar ekleyin.

Kanban Duvarı'mızdaki Kolonların Limitleri Kaç Olmalı?

"Sizin" kolonunuz için Kanban limitine ulaştığınızda ve yapacak bir işiniz kalmadığında sizin kolonunuzdan önceki kolonlarda darboğaz aramaya başlamalısınız. Eğer bir darboğaz bulunmuyorsa o zaman sizin kolonunuzdan önceki kolonların limiti olması gerekenden düşük olabilir.

Bir limit olmasının amacı darboğazları besleme riskini engellemektir.

Eğer birçok iş maddesini üzerinde çalışılmadan beklediğini görürseniz, bu Kanban limitinin çok yüksek olduğunu gösterir.

- Düşük Kanban limiti => Bekleyen insanlar => Kötü verimlilik
- Yüksek Kanban limiti => Bekleyen işler => Kötü teslim zamanı

Kanban Limitleri Ne Kadar Katı Olmalı?

Kimi takımlar Kanban limitlerine askeri bir disiplinle uyar, takım asla limitleri aşamaz. Kimi takımlarsa bir rehber veya sorunu görüp tartışma için bir tetikleyici gibi görürler, takımın Kanban limitini aşmaya izni vardır ancak sağlam bir neden gösterip bunu yapar. Yani bir kere daha söylüyorum, size kalmış! Söylemiştim Kanban'da çok fazla kural yok.

16

Scrum ve Kanban Benzerlikler

- İkisi de Yalın ve Çevik.
- İkisi de işi çekme mantığına sahip.
- İkisi de Çalışılan İş sayısını limitler.
- İkisi de süreci iyileştirmek için şeffaflıktan yararlanır.
- İkisi de ürün parçacığını erken ve sık teslim etmeye odaklanmıştır.
- İkisi de kendi kendini yöneten takımlara sahiptir.
- İkisi de işi daha küçük parçalara bölmek gerektiğini söyler.
- İkisi de deneysel veriyi(takımın hızı/teslim süresi) dikkate alarak teslim planında sürekli iyileştirme yapılır.

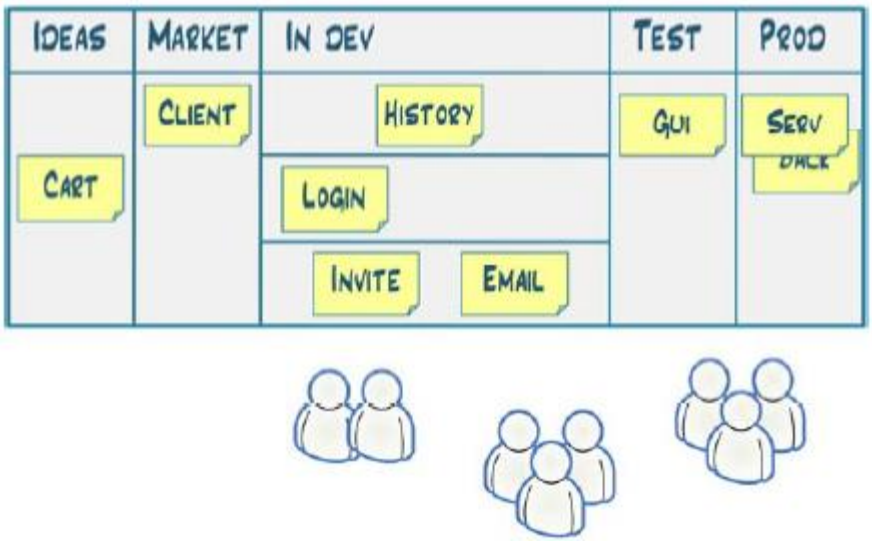
Scrum ve Kanban Farklılıklar

Scrum	Kanban
Süresi belirli ve sabit döngüler kuraldır.	Süresi belirli ve sabit döngüler isteğe bağlıdır.
Takım, bir döngü için belirli bir miktar işe taahhüt verir.	Taahhüt vermek isteğe bağlıdır.
Takım hızı, planlama yapmak ve süreci iyileştirmek için bir metrik olarak kullanılır.	Teslim Süresi, planlama yapmak ve süreci iyileştirmek için bir metrik olarak kullanılır.
Çapraz fonksiyonel takımlar kuraldır.	Çapraz fonksiyonel takımlar isteğe bağlıdır. Uzmanlık takımlarına izin verilir.
İş maddeleri küçük parçalara bölünmelidir. Böylece bir Sprint içinde tamamlanabilirler.	İş maddeleri için bir büyüklük belirlenmemiştir.
Burn-down grafik bir kuraldır.	Herhangi bir grafik kural değildir.
Çalışılan iş dolaylı yoldan sınırlanır. Her Sprint'te farklı olabilir.	Çalışılan İş direk olarak sınırlanır. İş akışındaki her adım için farklı olabilir.
İşlerin büyüklüğünü tahmin işi bir kuraldır.	İşlerin büyüklüğünü tahmin isteğe bağlıdır.
Devam eden döngü içinde yeni iş maddeleri eklenemez.	Kapasitede boşluk oluştuğunda yeni iş maddeleri eklenebilir.
Sprint İş Listesi'nin sahibi **bir** takımdır.	Kanban Duvarı **birden çok** takım ya da kişi tarafından paylaşılabilir.
3 rol-Ürün Sahibi, Scrum Master, Geliştirme Takımı- kuraldır.	Herhangi bir rol yoktur.
Scrum Duvarı, her Sprint'te yeniden oluşturulur.	Kanban Duvarı kalıcıdır.
Önceliklendirilmiş Ürün İş Listesi'nin olması kuraldır.	Önceliklendirme isteğe bağlıdır.

İşte bu kadar, şimdi farkları da biliyorsunuz. Fakat bitmedi. Şimdi sıra en iyi bölümde! Botlarınızı giyin! Şimdi zaman, Mattias ile birlikte sipere girme ve pratikte nasıl olduğunu öğrenme zamanı.

Bölüm 2 – Durum Çalışması

Gerçek Hayatta Kanban



Bu hikaye, bizim Kanban'ı kullanışımızın hikayesidir. Biz başladığımızda etrafta çok fazla bilgi yoktu ve Dr. Google ilk defa elimizi boş bıraktı. Bugün Kanban çok başarılı bir şekilde evrimleşiyor ve etrafta birçok bilgi var. David Anderson'un çalışmalarına göz atmanızı şiddetle öneririm. İlk ve son tavsiyem şudur: Çözümlerinizin belirli problemlerinizi karşıladığından emin olun! İşte bizim hikayemiz.

/Mattias Skarin

17

Operasyonel İşlerin Doğası

Eğer 7/24 çalışıyorsanız bir üretim ortamının yönetim sorumluluğu üzerine bilginiz vardır. Bir gece yarısı problemi anlamanız ve çözmeniz beklenir, problemin kaynağı olun ya da olmayın. Hiç kimse çözümü bilmez, bu nedenle sizi ararlar. Donanımı siz yapmadığınıza, sürücüleri, işletim sistemini ya da yazılımı siz yazmadığınıza göre oldukça zor bir durumdur. Seçenekleriniz, problemi belirli bir noktaya kadar daraltmak, problemin etkisini sınırlamak, problemi yeniden oluşturmak için gerekli bilgileri saklamak ve problemden sorumlu kişinin gelip sorunu yeniden oluşturup çözmesiyle sınırlıdır.

Teknik operasyonlarda cevap verebilirlik, problem çözmenin hızı ve doğruluğu en önemli noktalardır.

18

Neden Değişim?

2008 yılında müşterilerimizden biri, -bir İskandinav oyun geliştirme organizasyonu- bir dizi süreç iyileştirme çalışması gerçekleştirdi. Bunlardan biri Scrum'ın yazılım geliştirme bölümünde yaygınlaştırılması ve yazılım geliştirmenin önündeki engellerin parça parça ortadan kaldırılmasıydı. Yazılımda işler akmaya ve performans yükselmeye başladıktan sonra teknik operasyonlar tarafındaki baskı kümülatif bir şekilde arttı. Önceleri yandan seyreden operasyon takımı şimdilerde geliştirme sürecinin aktif bir ortağıydı.

Şekil 1. Teknik operasyonlar ekibi 3 takımdan oluşur: Veritabanı Yöneticileri(DBA), Sistem Yöneticileri ve Acil Destek Ekibi.

Sonuç olarak süreç iyileştirme çalışmasında sadece geliştirme takımlarına yardım etmek yeterli değildi. Eğer sadece geliştirme takımlarına odaklanmaya devam etseydik, bu teknik operasyonlar ekibi tarafından geliştirilen kritik altyapı iyileştirmelerinde gecikmeye neden olurdu.

Ayrıca geliştirme takımlarındaki ilerleme, geliştirme takımlarının analizler ve yeni fikirler hakkında geri bildirim ihtiyaçlarını artırdı. Bunun anlamı yöneticilerin gerçek zamanlı işlerin önceliklendirilmesine ve problem çözümüne ayırabilecekleri daha az zamana sahip olmalarıdır. Yönetim takımı, durum yönetilemez hale gelmeden aksiyon almaları gerektiğini fark etti.

19

Nereden Başladık?

Teknik operasyonların müşterisi olan geliştirme takımlarına sormak iyi bir başlangıç noktası oldu.

Geliştirme Açısından Operasyonun Görünümü

“Operasyonları düşündüğünüzde aklınıza gelen en önemli üç şey nedir” diye sordum? En yaygın cevaplar şöyleydi:

“Değişken bilgi”

“İş akış sistemleri berbat”

“İş alt yapıya geldiğinde çok yetkin”

“O adamlar ne yapıyor?”

“Yardım isterler ama söz konusu bize yardım etmek olduğunda etmezler”

“Basit işleri yapmak için birçok e-posta göndermek gerek”

“Projeler çok uzun zaman alıyor”

“İletişime geçmek zor”

Özetle bu geliştirme takımlarının operasyonu nasıl gördüğüdür. Şimdi bunu operasyon takımlarının geliştirme takımlarını nasıl gördüğüyle karşılaştıralım...

Operasyon Açısından Geliştirmenin Görünümü

"Neden var olan platformun avantajlarını kullanmıyorlar?"

"Keşke teslimleri(release) daha az sorunlu yapabilseler!"

"Kalitesizlerinden biz zarar görürüz!"

"Değiştirilmeleri gerek", her iki tartışmada da ana temaydı. Eğer ortak problemleri çözeceksek bu düşünce şeklinin değişmesi gerektiği açıktı. Toplantılardaki pozitif eleştirilere bakarsak "İş alt yapıya geldiğinde çok yetkin", çekirdek yetkinliklerine güvenildiğini gösterir. "Bize karşı onlar" yaklaşımı eğer doğru çalışma koşulları yaratılırsa düzeltilebilir. Bu nedenle organizasyon fazla mesaiyi kaldırmanın ve kaliteye odaklanmanın geçerli tek seçenek olduğunu anladı.

20

Başlarken

Başlamalıyız, fakat nereden? Bildiğimiz tek şey başladığımız yer ulaştığımız yer olmayacak.

Geçmişte yazılım geliştiriciydim yani operasyonel işler hakkında biraz bilgim vardı. “Fırtına gibi içeri girip bir şeyleri değiştirme” yaklaşımında olmamam gerektiğini biliyordum. İlgili şeyleri bize öğretecek, ilgisiz şeyleri ortadan kaldırabilecek, kolayca öğrenebileceğimiz ve daha az çatışmacı bir yaklaşıma ihtiyaç vardı.

Adaylar:

Scrum – Geliştirme takımlarında gayet iyi gidiyordu.

Kanban – Yeni ve denenmemiş fakat organizasyonumuzda olmayan Yalınlık ilkeleriyle çok uyumlu.

Müdürlerle yaptığımız toplantılarda Kanban ve Yalınlık ilkeleri adreslemeye çalıştığımız problemlerle örtüşüyor gibi göründü. Onların bakış açısından Sprint’ler operasyonel işler için uygun değildi çünkü önceliklendirmeleri günlük bazda yapıyorlardı. Yani Kanban yeni bile olsa mantıklı bir başlangıç noktasıydı.

21

Takımlar Başlarken

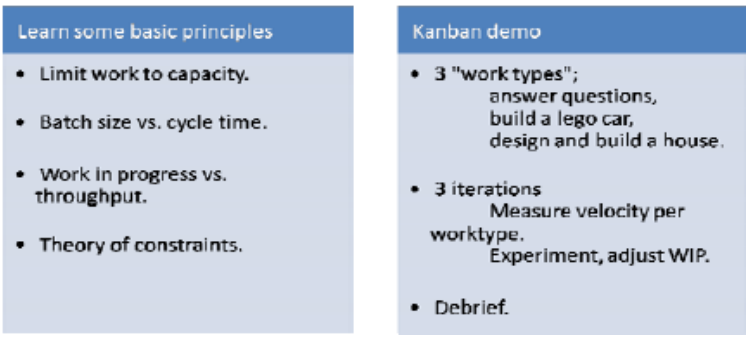
Danışman olarak takımları Kanban'a nasıl başlatabileceğim konusunda herhangi bir fikrim yoktu, bunun için bir el kitabı da yoktu. Bu başlangıcı yanlış yapmak birçok şeyi riske atabilirdi. İyileştirme noktalarını kaçırmanın dışında yerine birileri zor bulunacak, konusunda uzman ve yetenekli kişileri kaçırmak istemiyorduk, bu kişilerin kaçışı sorun olabilirdi. Onları soğutmak oldukça kötü olurdu.

- Devam etmeli ve sonuçlarla, onlar ortaya çıktığında mı baş etmeliyiz?
- Ya da ilk önce bir atölye çalışması mı yapmalıyız?

Çok net bir şekilde görülüyor ki ilk önce atölye çalışmasını yapmalıyız. Nasıl? Bütün teknik operasyon ekibinin bir atölye çalışmasına katılmasını sağlamak oldukça zor, ya biri ararsa? Sonunda yarım günlük, basit ve gerçek hayattan örneklerin oldukça çok anlatıldığı bir atölye çalışması yapmaya karar verdik.

Atölye Çalışması

Atölye çalışmasının faydalarından biri problemlerimizin erkenden gün yüzüne çıkmasına yardımcı olmasıdır. Aynı zamanda değişimin etkilerinin takım üyeleriyle güven içerisinde tartışılabileceği bir ortam sağladı. Söylemek istediğim şey, herkes içinde bulunduğu çalışma şeklini değiştirmeye aşırı hevesli değildir ve bu değişim sıkıntı yaratabilir. Fakat takım üyelerinin çoğu denemeye açıktı. Bizde en önemli ilkeleri anlattığımız küçültülmüş bir Kanban simulasyonu çalışması yaptık.



Atölye çalışmasının sonunda takımların bunu gerçekten denemek isteyip istemediğini öğrenmek için bir oylama yaptık. Bu aşamada herhangi bir itiraz gelmedi bizde onay almış olduk.

22

Paydaşlara Hitap

Sadece operasyon ekibi üyeleri değil aynı zamanda paydaşlarda Kanban uygulanmasından etkileneceklerdi. Değişim daha iyi olmak için yapılır. Bunun anlamı takım tamamlayamayacağı işe hayır demeye, kalite için diretmeye ve düşük öncelikli iş maddelerini takım iş listesinden çıkarmaya başladı. Tabi bunu yapmadan önce paydaşlarla konuşmak iyi bir fikirdir.

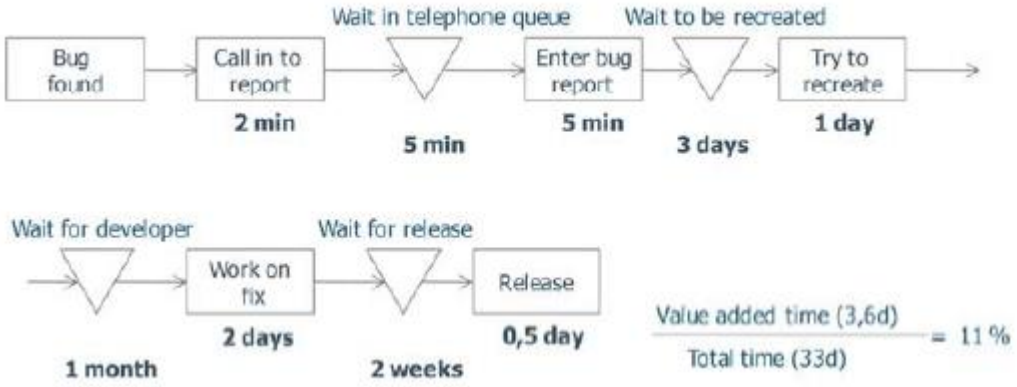
En yakın paydaşlar birinci basamak destek ekibi ve bölüm yöneticileriydi. Onlarda atölye çalışmasına katıldılar ve değişim konusunda pozitiftiler. Aynı şekilde geliştirme takımları az çok bir ilerleme bekliyorlardı. Fakat destek takımının görüşleri farklıydı. Destek takımının en büyük problemi fazla mesaiye boğulmuş olmalarıydı. Aynı zamanda müşteriden gelen problemlerle ilgileniyorlardı ve şirketin müşterilerden gelen tüm konulara cevap vermek gibi bir taahhüdü bulunuyordu. Kanban'ı uygulamaya başlamak ve Çalışılan İş sayısını sınırlamaya zorlamak oldukça büyük bir değişiklikti.

Yakın çalışacağımız paydaşlarla bir toplantı ayarladık ve amaçlarımızı, beklenen faydaları ve muhtemel sonuçları anlattık. Neyse ki önerilerimiz iyi karşılandı hatta bazen şöyle ifadeler kullanıldı, "sonunda bu konuları geride bırakabilirsek mükemmel olur!".

23

İlk Duvarın Oluşturulması

Kanban Duvarı'nı oluşturmaya başlamanın iyi bir yolu değer akış haritası oluşturmaktır. Bu, temelde değer zincirinin görselleştirilmesidir. Ayrıca bunu yapmak iş adımları, akış ve sistem boyunca geçen zaman hakkında fikre sahip olmamızı sağlar.



Biz çok daha basit başladık; basit bir Kanban Duvarı'nı yöneticilerden biriyle beraber kağıt üzerine çizdik. Birkaç defa gözden geçirdik ve tamamladık. Bu aşamada aklımıza gelen sorular:

- Ne tür işler yapıyoruz?
- Bu işleri yapanlar kimler?
- Farklı iş tiplerinde sorumluluğu paylaşmalı mıyız?
- Farklı iş tiplerinde sorumluluğu paylaşırsak farklı yetkinlik seviyeleriyle nasıl başa çıkmalıyız?

Farklı iş tipleri farklı SLA'lere sahip olduğundan her takımın kendi Duvarı'nı oluşturması mantıklı geldi. Takımlar kolonları ve satırları kendileri oluşturdular.

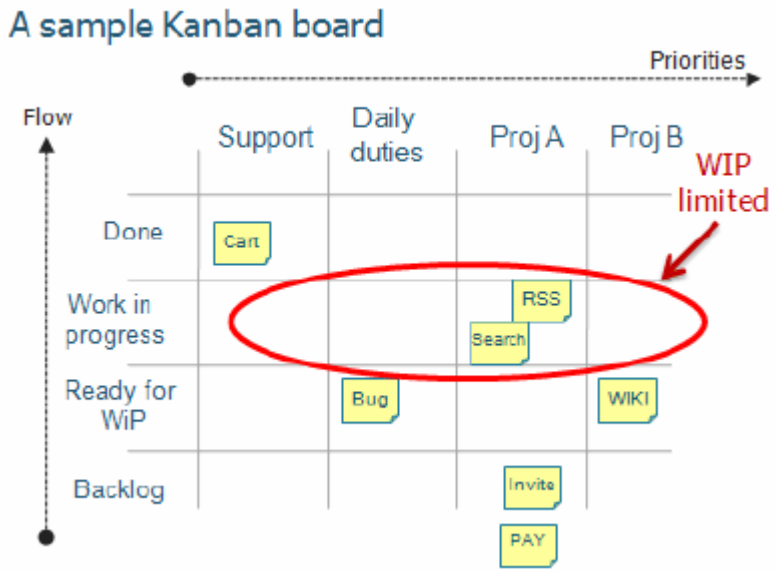
Sonraki önemli karar farklı iş tipleri arasında paylaşılmış sorumluluk olup olmayacağıydı. "Takımın bir bölümü gelen soruları hemen cevaplarken-reaktif iş- kalan bölümüyse projelere-proaktif iş- mi odaklanmalıydı?" İlk önce sorumluluğu paylaşmaya karar verdik. Çünkü kendi kendini yönetmeyi,

sürekli öğrenmeyi ve bilgi paylaşımının sürekli büyüme için önemini anlamıştık. Bu kararın sakıncalı yönü herkesin zarar görebilecek olmasıydı fakat o an düşünebildiğimiz en iyi fikir buydu.

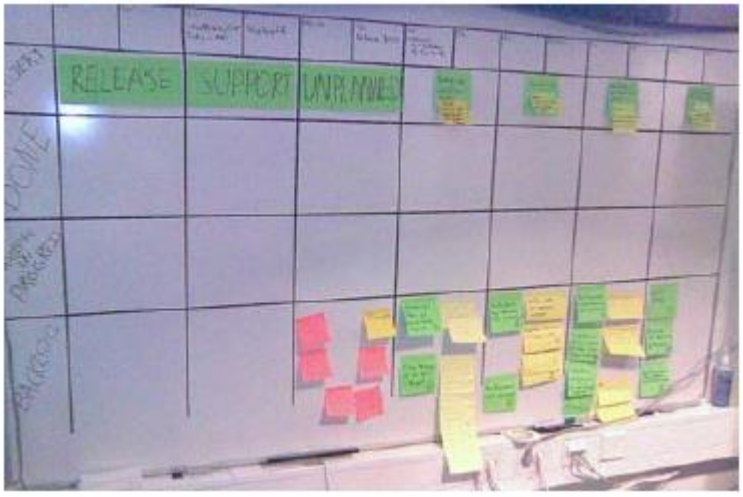
Küçük bir not: Atölye çalışmasında bu konuyu gündeme getirdiğimizde takım aslında kendi kendine bu problemin çözümünü aradı. Acil isteklerle bir kişinin ilgilenmesinin ve kalanların daha büyük konularla ilgilenmesinin daha doğru olduğunu düşündüler.

İlk Kanban Modeli

Aşağıdaki resim ilk Kanban modelini gösterir. Ayrıca dikkat edin Takım iş maddelerinin aşağıdan yukarı doğru akmasını istedi, tıpkı denizin içindeki su baloncukları gibi, tipik modelse iş maddelerinin soldan sağa doğru aktığı modeldir.



Resim 2. Bu ilk Kanban Duvarı'nın görünümüdür. Öncelikler soldan sağa doğru giderken iş akışı aşağıdan yukarı doğrudur. Çalışılan İş sayısı limiti, Çalışılan İş satırındaki işlerin toplamının hesaplanmasıyla bulunur. Bu model oluşturulurken Linda Cook'tan etkilenilmiştir.



Resim 3. Sistem Yöneticileri takımının ilk Kanban Duvarı.

Resimdeki Satırlar

İş Akış Adımları	Nasıl tanımladık?
İş Listesi	Yöneticinin gerekli olduğuna karar verdiği kullanıcı hikayeleri.
Hazır	Büyüklükleri belirlenmiş ve en fazla 8 saatlik küçük işlere bölünmüş kullanıcı hikayeleri.
Çalışılan İş	Çalışılan İş sayısı limitinin bulunduğu kolondur. Limit = (Takım Üye Sayısı * 2) – 1 olarak belirledik. -1 iş birliği için ayırdık. Yani 4 kişilik takımda Çalışılan İş limitini 7 olarak belirledik.
Bitti	Teslim edilmeye hazır.

Resimdeki Kolonlar

İş Tipi	Nasıl tanımladık?
Teslim	Geliştirme takımlarına teslim için yardım edilen işler.
Destek	Diğer ekiplerden gelen küçük istekler.
Planlanmamış	Beklenmeyen işler, örneğin altyapısal küçük işler.
A Projesi	Büyük operasyonel işler, örneğin staging ortamında donanımsal değişiklik.
B Projesi	Bir başka büyük proje.

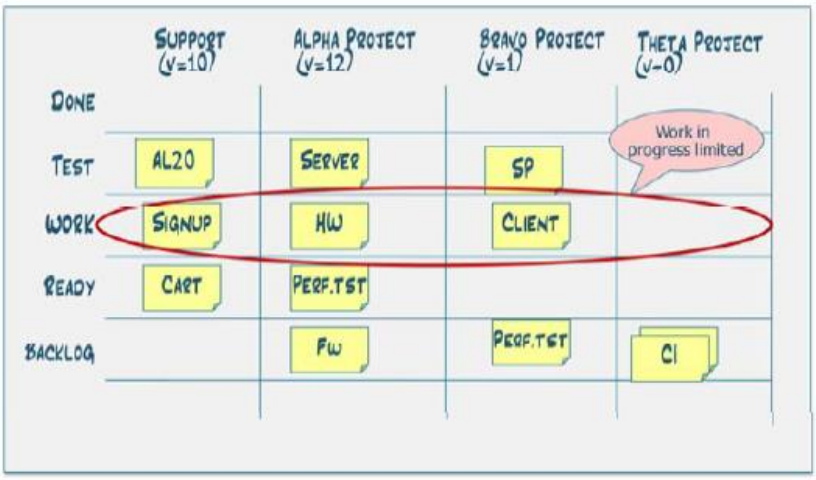
Bütün Kanban Duvarları aynı değildi. Hepsi basit bir çizimle başladı ve zamanla evrimleşti.

24

İlk Çalışılan İş Limitini Belirleme

Çalışılan İş limitini ilk belirlediğimizde oldukça cömert davrandık. İş akışını görselleştirerek, neler yaşadığımızı görebileceğimizi ve deneyimlerimizden faydalanabileceğimizi düşündük. Ayrıca daha başlangıçta bizim için uygun olan limiti doğru belirlememiz pek olası değildi. Zaman geçtikçe ve iyi nedenler buldukça(tek yapmamız gereken Duvar'da bunu göstermekti) Çalışılan İş limitinde iyileştirmeler yaptık.

Belirlediğimiz ilk limit = 2n – 1'di. (n= Takım üye sayısı, -1 iş birliğini desteklemek için çıkardığımız iş). Neden böyle bir değer verdik? Çok basit, aklımıza daha iyi bir fikir gelmedi! ☺ Formül, takıma iş veren herkese basit ve mantıklı geldi. "... buna göre her bir takım üyesi aynı anda en fazla iki iş üzerinde çalışabilir, bir aktif ve bir bekleyen. Neden daha fazla almalarını bekliyorsunuz?" Geriye doğru bakın, yeni başlayanlar için böyle cömert limitler hep olur. Kanban Duvarı'nı gözlemleyerek doğru limitler zamanla bulunabilir.



Resim 4. Veritabanı Yöneticileri ve Sistem Yöneticileri takımları için Çalışılan İş limitlerini nasıl uyguladığımızın resmi, her bir iş tipi için limit 1.

Yaptığımız bir gözlem, hikaye puanlarını kullanarak Çalışılan İş limitini belirlemenin faydasız olduğunu gösterdi. Bunu takip etmesi çok zordu, takip edilecek kadar kolay olan yöntemse sadece Duvar'daki iş maddelerini saymaktı.

Destek Takımı için kullandığımız limitler kolon bazında tanımlandı çünkü limit aşılıyorsa daha hızlı reaksiyon vermeliydik.

25

Çalışılan İş Limitine Saygı Gösterme

Çalışılan İş limitine saygı göstermek teoride basit gibi dururken pratikte bunu uygulamak oldukça zordur. Bunun anlamı bazen “hayır” diyebilmektir. Bunun için birçok farklı yol denedik.

Duvarda Konuşma

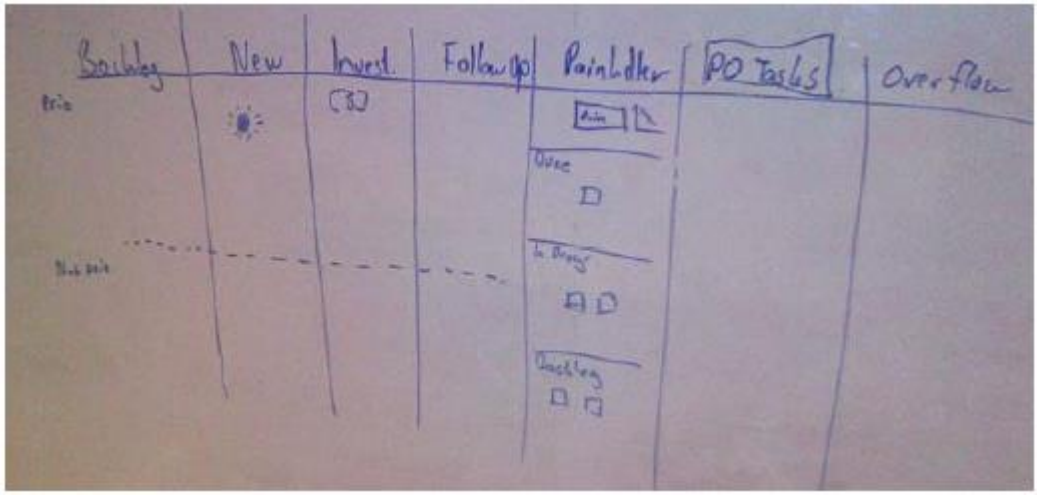
Eğer bir ihlal görülürse paydaşları Kanban Duvarı’na getiririz ve neyi başarmak istediklerini sorarız. Başlangıç döneminde ihlallerin en büyük nedeni deneyimsizlikti. Bazı durumlarda önceliklendirme üzerine farklı görüşlerin olduğunu da gördük, bunun nedeni genellikle uzman geliştiricilerden birinin belirli bir alanda bilgiye sahipken diğer bir alanda yeterince bilgi sahibi olmamasıydı. Bunlar anlaşmazlığın olduğu ender zamanlardı. Çoğu zaman problem Duvar önünde anlaşıldı ve çözüm yolları tartışıldı.

Taşma Bölümünün Belirlenmesi

Eğer “hayır” demek çok fazla çatışmaya yol açıyorsa ve iş maddelerini Duvardan kaldırmak zorsa düşük öncelikli iş maddelerini “Taşma” bölümüne taşıdık. Taşma bölümündeki işlere iki kural uyguladık:

1. Bu işler unutulmadı, ne zaman uygun olursak bu işlerle ilgileneceğiz.
2. Eğer bu maddeleri Taşma bölümünden de kaldırmaya karar verirsek, tüm paydaşları bilgilendireceğiz.

Sadece iki haftadan sonra Taşma bölümündeki maddelerle hiç ilgilenilmediği açıkça görüldü. Takım yöneticisinin yardımıyla Taşma bölümü tamamıyla kaldırıldı.

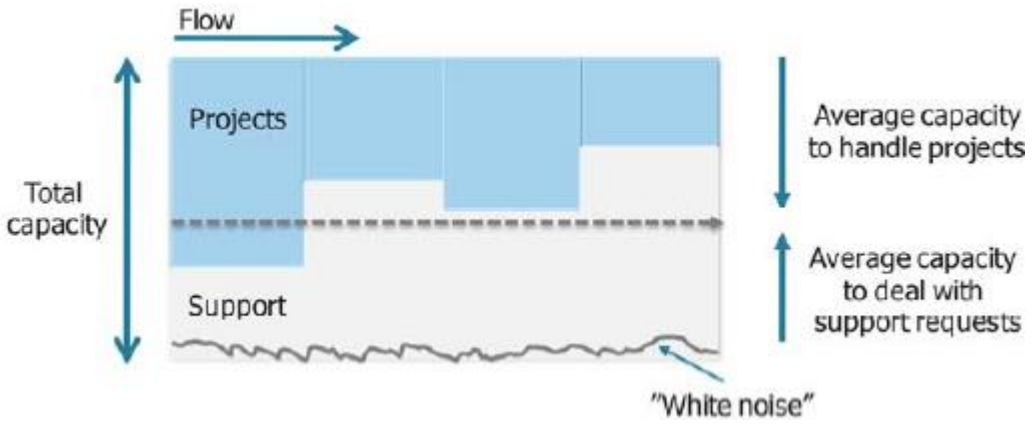


Resim 5. Destek Takımı için Kanban Duvarı çizimi; en sağdaki kolon Taşma bölümüne ayrıldı.

26

Hangi İşler Duvarda

Takım tarafından yapılan bütün işlerin Duvarda olmamasına karar verdik. Telefon görüşmesi, kahve arası gibi şeyleri takip etmek Kanban'ı bir yönetim kâbusuna dönüştürür. Problem çözmek için buradayız, yaratmak için değil ☺ Bizde 1 saatten büyük işleri Duvara yansıtmaya karar verdik, 1 saatten küçük her şey **parazit**-ses ve elektrik frekanslarında olduğu gibi- olarak düşünüldü. 1 saatlik limit gerçekten başarılıydı ve değişmeden kalan birkaç şeyden biriydi. (Biraz sonra arka plan gürültülerinin etkisi hakkındaki varsayımımızı gözden geçireceğiz.)



Resim 6. Toplam kapasitenin kabaca iki iş tipi-büyük projeler ve destek işleri- tarafından tüketildiğini varsayarak çalışmaya başladık. Takım hızını-takımın büyük projelere harcadığı kapasiteyi- hesaplayarak teslim tarihini tahmin edebileceğimizi düşünmüştük. "Parazit", 1 saatten küçük işler, toplantılar, kahve arası ya da bir arkadaşa yardım her zaman olacak şeyler diye düşündük.

27

İşlerin Büyüklüğünü Nasıl Belirledik?

İşlerin büyüklüğünü tahmin etme işi sürekli devam eden bir konu ve kesinlikle birden fazla cevabı var:

- Düzenli bir şekilde işlere büyüklük ver.
- İhtiyaç olduğunda işlere büyüklük ver.
- İşlerin büyüklüğünü belirlerken ideal gün / hikaye puanı kullan.
- Tahminler kesin değildir bu yüzden tişört büyüklüklerini kullan(small, medium, large).
- İşlerin büyüklüğünü hiç tahmin etme ya da "gecikme maliyetini" belirlemek senin için değerliyse tahmin yap.

Biraz Scrum'dan etkilenerek(sonuçta geldiğimiz yer orasıydı) hikaye puanlarıyla başlamaya karar verdik. Pratikteyse takımlar hikaye puanlarını adam/saat'in eşiti olarak görmeye başladılar. Başlangıçta bütün kullanıcı hikayelerine büyüklük verildi. Zamanla yöneticiler aynı anda gerçekleştirilen proje sayısını düşük tutarlarsa paydaşları bekletmediklerini öğrendiler. Ayrıca acil bir değişiklik olduğunda önceliklendirmeyi yeniden yapıp problemi adresleyebildiklerini öğrendiler.

Teslim tarihini tahmin etmek artık büyük bir sorun değildi. Bu, yöneticilerin iş gelmeden önce teslim tarihini sormalarının önüne geçti. Daha önce işler bitmediği için yöneticiler sürekli olarak ne zaman biteceğini soruyordu.

Yeni başladığımız zamanlarda yöneticilerden biri telefonda stres oldu ve "bu hafta sonunda" projenin yetiştirileceği sözünü verdi. Kanban Duvarı'nda bir projenin ilerleyişini tahmin etmek kolaydır, tamamlanan kullanıcı hikayelerini sayarak bile yapılabilir. Kullanıcı hikayeleri sayıldığında bir haftanın sonunda projenin yaklaşık olarak %25'inin tamamlanabileceği görüldü. Dolayısıyla fazladan 3 haftaya gerek vardı. Bu gerçekle erkenden yüz yüze kalan yönetici işlerin önceliklerini değiştirdi, devam eden bütün işleri durdurdu ve projenin teslimini mümkün kıldı. ☺

Tahmin Edilen Büyüklük Ne Demek? Teslim Süresi Mi, Çalışma Zamanı Mı?

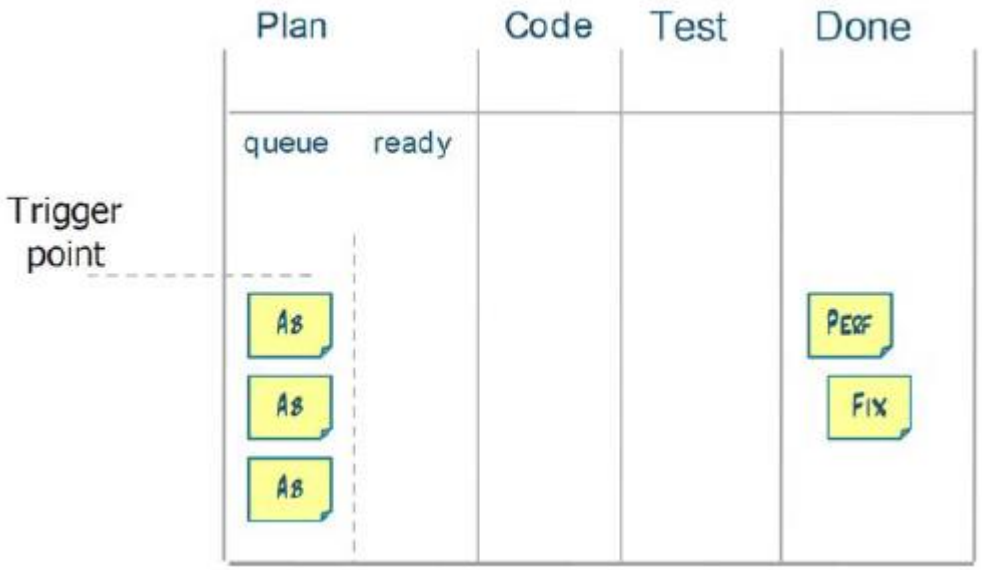
İş maddelerine hikaye puanı ile büyüklük verme, takım üyelerinin çalışma zamanını düşünmelerine neden oldu. Örneğin; bu kullanıcı hikayesi bölünmeden kaç saatte-teslim süresi, takvim süresi ya da bekleme süresi değil- tamamlanır? BİTTİ olan hikaye puanlarını-takımın hızını- her hafta ölçerek teslim sonucunu hesaplayabiliriz.

Her kullanıcı hikayesinin büyüklüğünü sadece bir defa tahmin ettik, işler yapılırken kullanıcı hikayelerinin büyüklüklerini yeniden gözden geçirmedik. Bu tahmin yapma işine harcadığımız zamanı en aza indirgememizi sağladı.

28

İşleri Nasıl Yaptık?

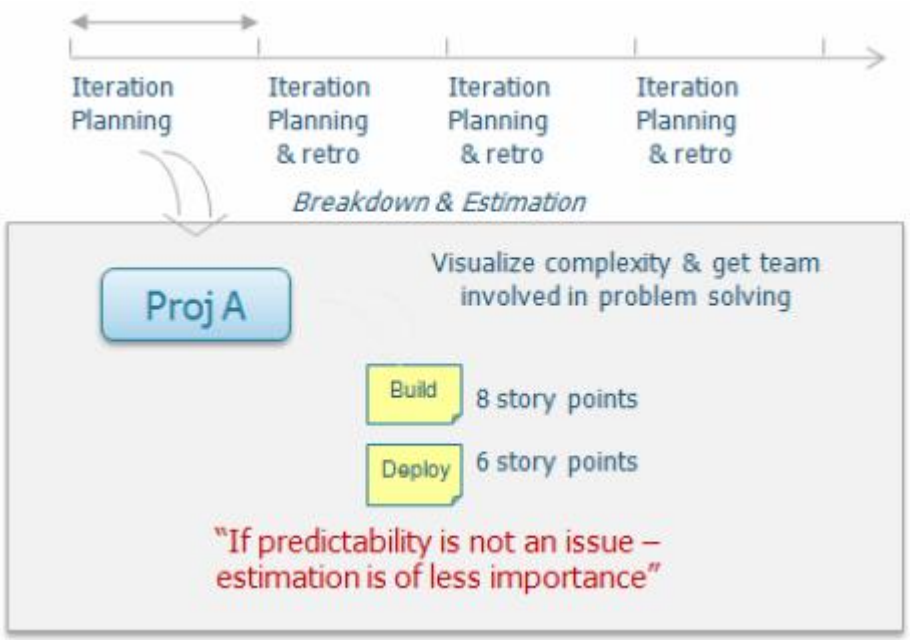
Kanban gerçekten sizi sınırlamayan ve özgür bırakan bir yapıya sahip, istediğiniz şekilde çalışabilirsiniz. Takımlara zaman bağımlı çalışmaları için izin verebilirsiniz ya da aktivitelerin yapılması için doğru zamanın geldiğini düşünüp o zaman yapabilirsiniz.



Resim 7. İş Listesi kolonuna üç iş maddesi girdiğinde, bu planlama ve işlerin büyüklüğünü tahmin etme aktivitelerinin yapılması için bir tetikleyici olabilir.

Bir ritim şeklinde tekrar eden iki etkinlik yaparız:

- Günlük toplantı – Duvarın önünde takımca problemleri ortaya çıkarmak ve bilgi paylaşımı için…
- Haftalık iterasyon planlaması, planlama ve sürekli iyileştirmeyi gerçekleştirme amaçlarıyla…



Bu yaklaşım bizim için gayet başarılı bir şekilde işledi.

Günlük Toplantılar

Günlük toplantılar, Günlük Scrum Toplantıları'nın benzeriydi. Bu toplantı tüm yazılım geliştirme takımlarının Günlük Scrum Toplantıları yapıldıktan sonra yapılıyordu ve bütün takımlardan katılım oluyordu(geliştirme, test, operasyon). Geliştirme Takımları arasında yapılan Scrumlar'ın Scrum'ı(Scrum of Scrums) Toplantısı Kanban Takımları için değerli girdiler sağlıyordu. Örneğin; ilk önce hangi problem çözülmeli ya da şuan hangi geliştirme takımı en büyük sıkıntıyı yaşıyor. Başlangıçta yöneticiler sıkça bu toplantılara katıldılar, çözüm önerilerinde bulundular ve önceliklere karar verdiler. Zamanla takımlar kendi kendini yönetme konusunda daha iyi oldukça yöneticiler daha az katılım gösterdiler fakat ihtiyaç olması durumunda ulaşılamaz değillerdi.

İterasyon Planlama

Haftada bir iterasyon planlama toplantısı yaptık. Sabit bir zaman-haftalık- aralığı belirledik çünkü planlama yapıp öncelikleri belirlemezsek önceliğimiz başka şeylere kayıyordu. Ayrıca takımca konuşmaya ve bilgi paylaşımı yapmaya ihtiyacımız olduğunu fark ettik. Toplantı formatı:

- Grafikler ve Duvarın güncellenmesi(Tamamlanan projeler "Bitti Duvarına" taşınır).
- Geride kalan hafta değerlendirilir. Ne oldu? Neden böyle oldu? Bunu geliştirmek için ne yapılabilir?
- Eğer ihtiyaç varsa Çalışılan İş limitinin güncellenmesi.
- Eğer ihtiyaç varsa işlerin büyüklüğü belirlenir ve işler daha küçük parçalara bölünür.

Temelde iterasyon planlama işlerin büyüklüğünü belirleme ve sürekli iyileştirme için bir ritimdi. Küçük ve orta ölçekli işler ilk hat yöneticilerinin yardımıyla çözümlendi. Fakat karmaşık, altyapısal konular başa çıkması zor işlerdir. Bununla başa çıkmak için takımlar "iki takım engelini" yöneticilerine atadılar. İki takım engeli takımların karşılaştıkları ve çözemedikleri sorunları ifade eder. Çözemedikleri sorunları yöneticilerine atarlar ve yöneticileri takım için çözmeye çalışır.

Kurallar:

1. Yönetici herhangi bir anda sadece iki engelle ilgilenebilir.
2. Eğer ikisi de doluysa önceliği düşük olanı çıkarıp yeni bir tane ekleyebilirsiniz.
3. Problemin çözüldüğüne takım karar verir.

Bu olumlu bir değişiklikti. Takımlar, yöneticilerinin zor konuların çözümünde bile onlara yardım ettiğini görünce şaşırdılar. Takım üyeleri engel listesini göstererek "nasıl gidiyor?" diye sordular. Bu problemler unutulmadı ya da yeni gelen önceliği yüksek maddeler eklenmesi nedeniyle iptal edilmediler.

Ciddi engellerden biri, bir hata bulunduğunda operasyon takımı üyelerinin geliştiricilerden yardım alamama konusuydu. Problemin neden kaynaklandığını bulmak için bir geliştiricinin yardımı gerekliydi fakat geliştiriciler Sprint'ler içinde yeni şeyler geliştirdikleri için problemler sürekli beklemekteydi. Şaşırtıcı değil operasyon takımı üyeleri geliştiricilerin kaliteyi çok önemsemediklerini düşünmeye başladılar.

Bu engel ortaya çıktığı zaman ilk önce kaynak yöneticisine iletildi daha sonra bölüm müdürüne. Daha sonra bölüm müdürü, geliştirme bölümünün müdürüyle bir toplantı ayarladı. Tartışmanın sonunda müdürler kaliteye öncelik vermeye karar verdiler ve bir çözüm üretildi. Her Sprint, geliştirme takımlarından biri aranmaya ve operasyona her an yardım etmeye hazır oldu. Yazılım geliştirme müdürü operasyon ekibine iletişim kurabilecekleri kişilerin bir listesini verdi. Daha sonra operasyon ekipleri bunun gerçek olmadığını düşünüp çözümü test etmek istediler. Fakat bu kez gerçekti, gerekli çalışmalar geliştirme ekipleri tarafından yapılmıştı ve engel ortadan kaldırılmıştı. Bu operasyon takımlarına büyük bir rahatlama getirdi.

29

İşe Yarar Bir Planlama Şekli Bulmak

Bir Hikaye

Takımlardan biri için bir dönüm noktası hatırlıyorum. İş büyüklüklerini belirleme toplantılarından ikincisiydi, takım, büyüklüğünü nasıl vereceğini bilmediği bir projeye takılmıştı. Birçok bilinmeyen vardı ve bu bütün toplantının aksamasına neden oldu. Sorumluluk alıp, problemi çözmektense süreci detaylandırmalarını ve daha iyi bir çözüm bulmalarını rica ettim. Yöneticilerinin liderliğiyle zorluğa göğüs gerdiler ve kendi çözümlerini tasarlamaya başladılar. Bu olay önemli bir dönüm noktasıydı, kendi geliştirdikleri önemli bir "kazanımdı". Böylece takım olma adına güven elde ettiler. Bundan sonra o kadar hızlı evrimleşmeye başladılar ki yollarından çekilmek zorunda kaldık.

İki ay sonra bir retrospektif toplantısının bitişinde yöneticileri bana yaklaştı ve "bir problemi" olduğunu söyledi. Takımının Kanban Duvarı'nı göstererek, "gerçek bir problemimiz yok, ne yapmalıyız?"

Planlamanın Yeniden Keşfi

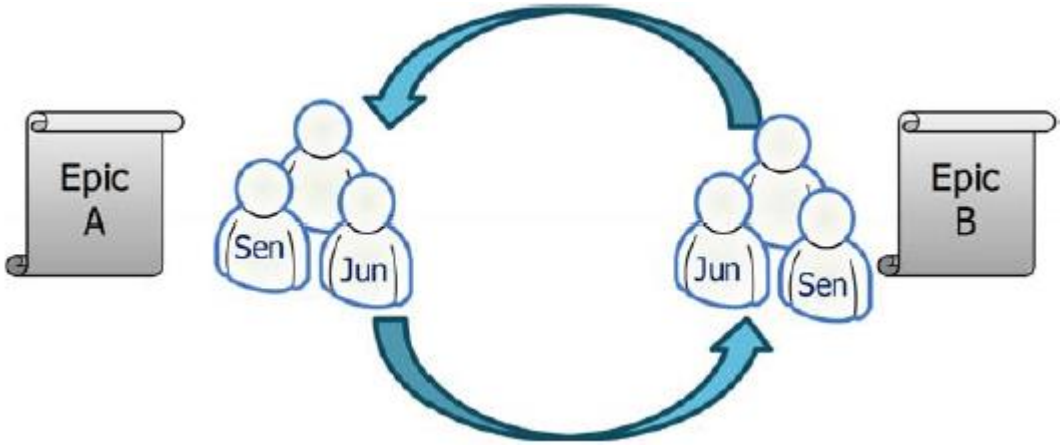
Planlama toplantılarında takım üyelerinin tamamının katılımıyla gerçekleştirdiğimiz poker kartlarıyla işlerin büyüklüğünü belirleme toplantıları üç takım içinde başarılı sonuç vermedi.

Nedenler:

1. Bilgi, takım üyeleri arasında eşit bir şekilde dağılmamıştı.
2. Genelde sadece bir kişi konuşuyordu.
3. Takım üyeleri masalarında bıraktıkları sorunları biran önce çözmek için toplantının kısa sürmesini istiyorlardı.

Takımlar deneyimleyerek iş büyüklüğünü iki farklı şekilde tahmin etme süreci geliştirdiler. Takımların oluşturdukları süreç kendileri için gayet başarılı bir şekilde işledi.

Yaklaşım 1 – Değiştirmek ve Yeniden Gözden Geçirmek



- Her projenin/kullanıcı hikayesinin büyüklüğünü tahmin etmek için bir uzman ve bir çıraktan oluşan çift atandı. Örneğin; belirli kullanıcı hikayesini çok iyi bilen biri ve bilmeyen biri). Bu bilginin yayılmasına yardımcı oldu.
- Kalan takım üyeleri büyüklüğünü belirlemek istedikleri kullanıcı hikayelerini seçtiler. Her kullanıcı hikayesini en fazla dört kişinin değerlendirmesi kuraldı. Bu tartışmayı verimli kılmak için alınmış bir karardı.
- Her "tahmin takımı" kendi kullanıcı hikayelerini daha küçük parçalara böldü.
- Daha sonra tahmin takımları kullanıcı hikayelerini birbirleri arasında değiştirdiler ve diğer takımların kullanıcı hikayelerini ve büyüklük tahminlerini gözden geçirdiler. Bu gözden geçirme sırasında takım üyelerinden biri geride kaldı ve gözden geçiren kişilere açıklamalarda bulundu.
- Bitti!

Ortalama bir iterasyon planlama toplantısı 45 dakika sürüyordu ve enerji seviyesi toplantı boyunca yüksekti. Kullanıcı hikayeleri gözden geçirildikten sonra genelde 1-2 ayarlama yapılırdı.

Yaklaşım 2 – Uzmanlar Gözden Geçirir ve Sonra Tahmin Yapılır

Planlama toplantısı öncesinde iki uzman takım üyesi projeyi/kullanıcı hikayesini gözden geçirir, mimarisel çözümleri analiz eder ve bir çözüm üzerine karar verirler. Bir kere karar verildikten sonra takım olaya dahil olur ve kullanıcı hikayesini daha küçük parçalara böler.

Resim 8. İterasyon planlama toplantısında yeniden değerlendirme ve işi daha küçük parçalara bölme.

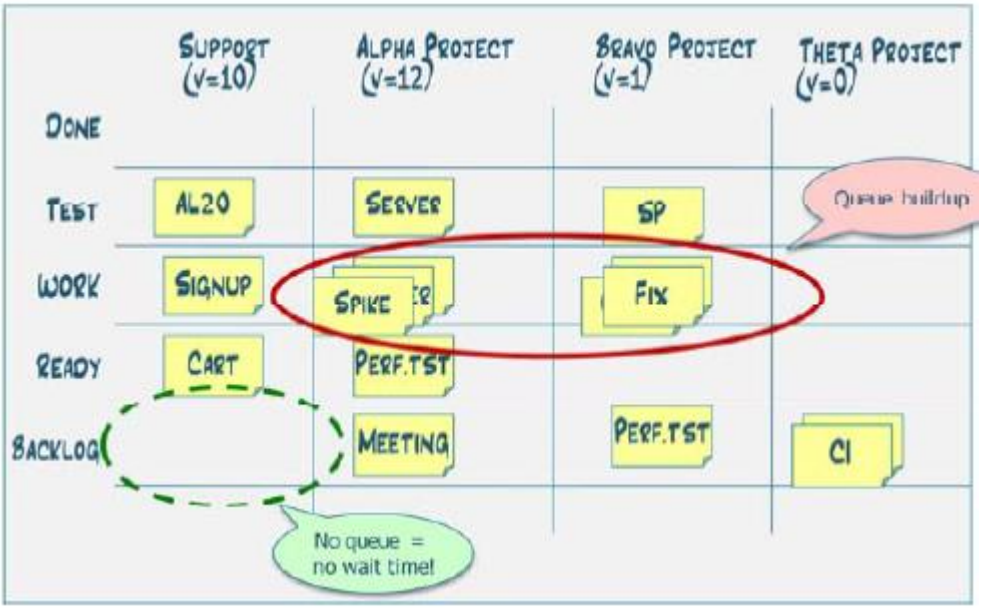
30

Neyi Ölçmeli?

Ölçülebilecek birçok şey vardır, ihtiyacın belirlendiği ve ihtiyacın karşılanması arasındaki zaman farkı, takım hızı, kuyruklar...önemli soru süreci iyileştirmek için hangi metrik kullanılabilir. Benim tavsiyem deneyimlemeniz ve sizin için uygun olanı bulmanızdır. Burn-down grafiklerin 4 haftadan kısa projeler için fazlasıyla çöp oluşturduğunu öğrendik. İlerlemenin tümü basitçe Kanban Duvarı'na bakarak görülebilir.(İş Listesi'nde kaç iş maddesi bulunuyor ve kaç iş maddesi Bitti bölümünde bulunuyor).

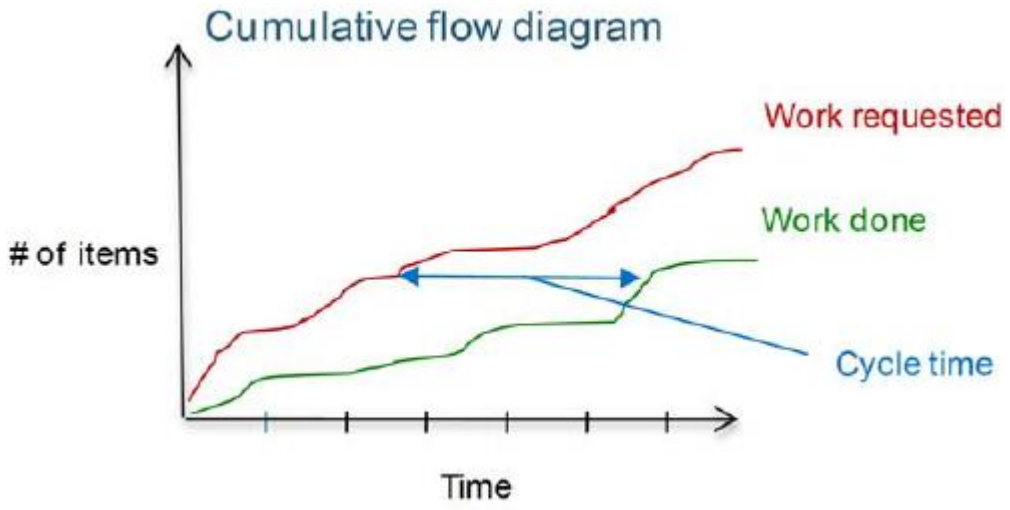
Metrik	Avantajı	Dezavantajı
Teslim zamanı	Ölçmesi kolaydır. Tahmin yapmaya gerek yok, müşteriyle başlar ve müşteriyle biter.	İşin büyüklüğü dikkate alınmaz.
Takımın toplam hızı(Bütün iş tipleri için)	Kaba fakat ilerlemenin ve değişimin basit göstergesidir.	Belirli iş tipleri için teslim zamanını tahmin etmeye yaramaz.
İş tipi bazında takımın hızı	Takımın toplam hızından daha detaylı olduğu için doğruya daha yakın sonuç verir.	Kullanışlı olması için müşterinin ihtiyacının başladığı ve ihtiyacın karşılandığı zaman dahil olmalıdır. Takımın toplam hızını belirlemekten biraz daha zordur.
Kuyruk uzunlukları	Talep dalgalanmalarının hızlı bir göstergesidir. Görselleştirmek kolaydır.	Beklemenin talep fazlalığından mı kapasite azlığından mı kaynaklandığını söylemez. Hiç kuyruk olmaması kapasitenin yüksek olduğunu gösterebilir.

"İş tipi bazında takım hızını" ve "kuyruk uzunluklarını" ölçerek başladık. İş tipi bazında takımın hızını ölçmek kolay ve beklediğiniz faydayı sağlar. Kuyruk uzunlukları, hemen fark edildikleri için iyi göstergelerdir.(Nereye bakmanız gerektiğini bilirseniz☺)



Resim 9. Darboğazlar ve fırsatlar. Kırmızı alan testin nasıl bir darboğaza dönüştüğünü gösterir. Destek kolonunda hiç kuyruk olmaması yeni gelen işlerde bekleme olmayacağını gösterir. Bu da yüksek müşteri memnuniyetine işarettir.

Kümülatif diyagramlar kullanmadık fakat kullanmak ilginç olabilirdi.



Kanban Duvarı ve takım hızını gösteren grafikler yeterli bilgiyi verdiği için kümülatif diyagramları kullanmadık, en azından takımın erken dönemlerinde. Darboğazlar, eşitsizlik ve fazla çalışma kümülatif diyagramlar kullanılmasa da kolayca tanımlanabilir ve çözümlenebilir.

31

Değişim Nasıl Başladı?

Kanban uygulamaya başlamamızdan 3 ay sonra Sistem Yöneticileri takımı, Bilgi Teknolojileri organizasyonu içinde en iyi performans gösteren takım ödülünü aldı. Aynı zamanda Sistem Yöneticileri takımı şirket içinde en iyi üç pozitif deneyimden biri olarak gösterildi. Pozitif deneyimler şirket genelinde 6 haftada bir yapılan bir etkinlikle seçilirdi ve ilk defa bir bilgi işlem takımı en iyi üç listesine girmişti. 3 ay önce bu takımlar herkesin dert yandığı darboğazlardı.

Hizmet kalitesi net bir şekilde artmıştı. Peki, bu nasıl oldu?

Dönüm noktası herkesin aynı anda çekmeye başlamasıydı. Yöneticiler açık bir odak sağladılar ve takımı, takıma ait olmayan işlerden korudular. Takımlar kalite ve teslim tarihleri için sorumluluk aldılar. Bunun ortaya çıkması yaklaşık olarak 3-4 ay aldı fakat bundan sonra iş pürüzsüzce aktı. Tabi ki dünyadaki bütün problemler biranda kaybolmadı, bu hepimizi işsiz bırakırdı ☺ Yeni problemlerle karşılaştık. Örneğin; artık takımlar bir darboğaz olmadığına göre takımların motivasyonunu nasıl yüksek tutarız?

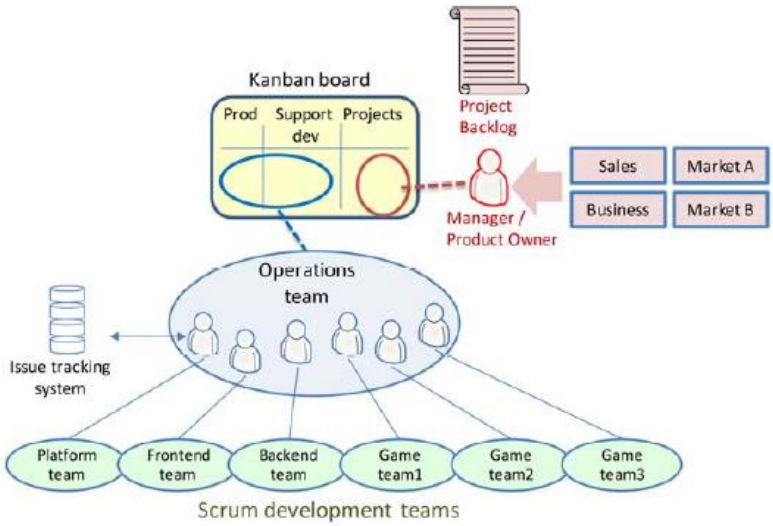
Kendi kendini yönetme yapbozunun önemli bir parçası her geliştirme takımına operasyon içinde iletişim kurabilecekleri birinin belirlenmesiydi. Kanban, operasyon takımı üyelerinin iş odaklı kendi kendini yönetmesini, fazla çalışmayı engellemesini ve sürekli iyileştirmeyi olanaklı kılarak bunun olmasını sağladı. Önceden rastgele bir kişi kuyruktan işi çeker, çözer ve yeni işe başlardı. İletişimde herhangi bir yanlış anlama bütün işin yeniden yapılması demekti. Bire bir kavramı ortaya çıktıktan sonra Destek Takımı biranda daha hızlı cevap verme olanağına sahip oldu.

Hızlıca iletişim protokolü evrimleşti; operasyon çalışanları iyi tanıdıkları yazılım geliştiriciler ile anlık mesajlaşmaya, yazma yetenekleri konuşma yeteneklerinden daha iyi olanlarla e-posta üzerinden iletişim kurmaya başladı. Eğer problemin en hızlı şekilde çözülmesi için uygun yol telefonsa telefon kullanmaya başladılar.

Önceden

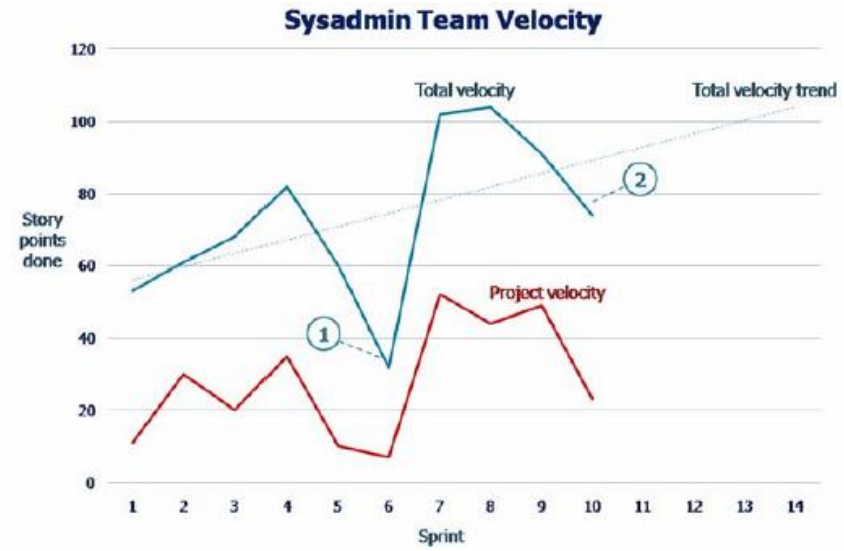
Resim 10. İlk hat yöneticileri her takım için ilk iletişim noktasıydı. Yapılması gereken her önemli şey ilk önce onların filtresinden geçiyordu. Küçük konular, yazılım geliştiricilerin problemleri bir takip sistemi aracılığıyla alındı. Kişiden kişiye iletişim çok azdı.

Sonradan



Resim 11. Her geliştirme takımına operasyon içinde iletişim kurabilecekleri biri belirlendikten sonra geliştirme takımları ihtiyaç duyduklarında direkt olarak bu kişiyle iletişime geçti. Kişiden kişiye birçok iletişim başladı. Operasyon takım üyeleri Kanban Duvarı'nda işlerini yönettiler. Yöneticiler büyük projelerin önceliklerini belirlediler ve büyük bir sorunla karşılaşıldığında takıma destek oldular.

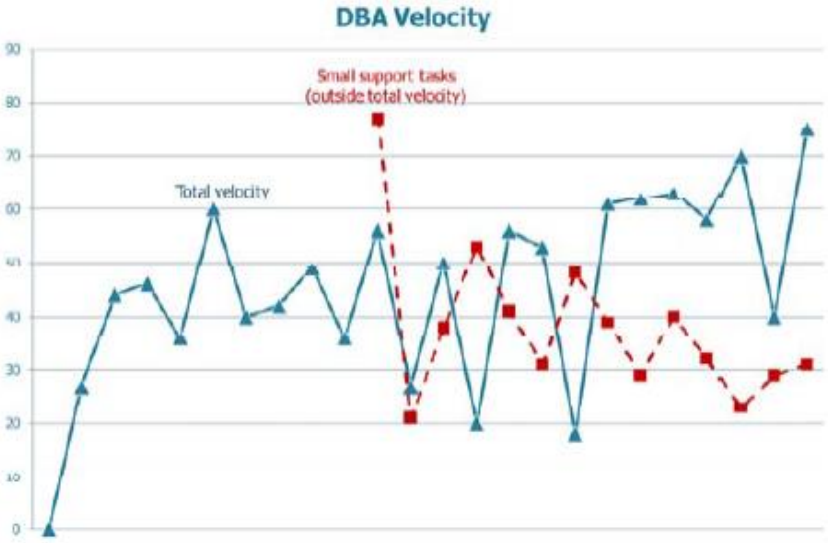
Bu, takımların performansına nasıl bir etkide bulundu?



Resim 12. Toplam takım hızı ve takım proje hızı, her hafta biten kullanıcı hikayelerinin büyüklükleri toplanarak ölçüldü. Toplam takım hızı değeri, bütün kolonların toplanmasıyla elde edildi. Takım proje hızı değeri, projelere bağlı takım hızı olarak hesaplandı. İki düşüş noktası aşağıdakilerle ilişkilidir:

1. **Takım üyelerinin tamamına yakını seyahate çıkmıştı**
2. **Geliştirme tarafında büyük bir teslim yapıldı**

Resmin tamamına bakıldığında takımın pozitif bir eğilimi olduğu görülür. Aynı zamanda Eşli Programlama yaparak bilgi paylaşımına da yatırım yapılmış oldu.



Resim 13. Toplam takım hızı ve küçük destek işleri. Ortadaki küçük düşüş yılbaşından kaynaklıdır.

Toplam takım hızı yukarı doğru bir eğilim göstermesine rağmen buradaki varyans önemlidir. Varyansın büyüklüğü takımın küçük destek işlerini izlemesi için bir işaret oldu. Gördüğünüz gibi toplam takım hızı ve küçük destek işleri arasında ters bir orantı bulunmaktadır.

Önce Destek Takımı daha sonra diğer iki takım Kanban uygulamaya başladı. Bu nedenle diğer iki takım için şuan elimizde yeterli veri bulunmuyor.

Olgunluğun Oluşması

Başladığımızda problemli bölgeleri bulmak kolaydı fakat iyileştirme için en büyük fırsatı bulmak zordu. Kanban Duvarı her şeye yeni bir şeffaflık verdi. Sadece problemleri nokta atışı göstermekle kalmadı, iş akışı, varyans ve kuyruklar hakkında önemli soruların sorulmasını da sağladı. Kuyrukları, problemleri belirlemek için bir araç olarak kullandık. Kanban'a başladıktan 4 ay sonra yöneticiler takımlarına zarar veren varyans kaynaklarını birer birer engellemeye başladılar.

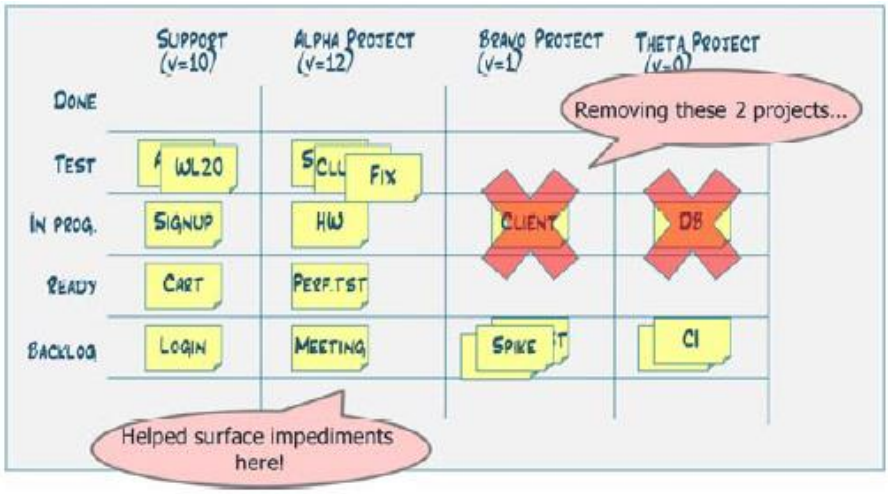
Takımlar bireylerden kendi kendini yöneten birimlere doğru evrimleştikçe yöneticiler, liderlik konusunda yeni zorluklarla karşılaşmaya başladılar. İnsani problemlerle-şikâyetler, ortak bir amacın belirlenmesi, çatışmaların çözümlenmesi ve pazarlık anlaşmaları- daha çok uğraşmaya başladılar. Yöneticiler açıkça belirttiler "bu vasıfları kazanmak yetenek ve enerji gerektirir", değişim ağrısız olmaz. Bu zorlukları gördüler ve kabul ettiler daha sonra gerekli yetkinlikleri kazanarak daha iyi bir lider oldular.

32

Çalışılan İş Sayısı Düşerken Kısıtlar Ortaya Çıkar

Bütün takımlar, başlangıçta Çalışılan İş limitlerini belirlerken büyük değerler seçtiler. O evrede enerjinin çoğu iş akışını oluşturmaya ve organizasyonun ihtiyaç duyduğu desteği vermeye harcandı. Başlangıçta yöneticiler birden fazla projenin aynı anda geliştirilmesini istediler. Birkaç hafta içinde önceliği düşük projelerle ilgilenilebilecek kapasitenin olmadığı ortaya çıktı. Kanban Duvarı'na hızlı bir bakış düşük öncelikli işlerin hiçbirine başlanılamadığını görmeyi sağladı. Bu yöneticilerin takım başına düşen proje sayısını düşürmelerini sağladı.

Zamanla yüksek öncelikli işler için akış daha sabit bir hal aldı. Çalışılan İş sayısını daha da düşürmeye başladık. Devam eden projeler için oluşturduğumuz kolon sayısını 3'ten 2'ye daha sonra da 1'e düşürdük. Bu olurken takım dışındaki kısıtlar ortaya çıkmaya başladı. Takım üyeleri takım dışındaki kişilerden yardım alamadıklarını bildirmeye başladılar. Böylece yöneticiler dikkatlerini bu konuya verdiler.



İlerleyen günlerde başka konularda gündeme geldi. Örneğin diğer takımlardan gelen bilgilerin yanlış olması ve bunun takım performansı üzerindeki etkileri. Gelen iş maddelerinin sürekli olarak düzeltilmesi gerektiğinden pürüzsüz ve hızlı iş akışını devam ettirmek zordu.

Biz başlamadan önce bu problemler görünür değildi. Problem, “acaba ilk önce hangi konuya destek vermeliyiz?” idi. Kanban Duvarı’yla belirli bir problemin iş akışını nasıl etkilediğini herkes kolayca görebildi. Herkesin kolayca gördüğü konu için organizasyonel sınırlar çerçevesinde aksiyon almak daha kolay oldu.

Kanban Duvarı Zamanla Değişecektir, Duvar Tasarımınızı Taşı Oyarak Yapmayın

Kanban Duvarları zamanla değişir. Takımın kendine uygun bir Duvarı bulması iki üç tasarımdan sonra olur. Bunun anlamı ilk tasarıma uzun zaman harcamak çöp yaratmaktır. Dikkat edilmesi gereken konu Duvar’ın kolayca yeniden tasarlanıyor olabilmesidir. Kolonların ve satırların belli olması için siyah bant kullandık. Bantları çıkarıp yeni bir tasarım yapmak kolay olur. Diğer bir uygulamaysa beyaz tahta kullanmaktır. Kolonları ve satırları kolayca çizip silebilir ve gerekli değişiklikleri yapabilirsiniz. Fakat kullandığınız kalemlerin silinebilen kalemlerden olduğuna emin olun ☺

Aşağıda tasarım iyileştirmesi yapılmış bir Duvar bulunuyor. Öncelikler sıkça değişiyorsa bir kolon boyunca yapışkan kağıtları bir ileri bir geri taşımaktansa her kolona bir öncelik numarası vermek gayet mantıklıydı.

Resim 14. Üzerinde o anki öncelikler bulunan ilk Kanban Duvarı.

Denemekten ve Başarısız Olmaktan Korkmayın

Bu maceradan çıkardığım ders bir “son” noktası olmamasıdır. Sadece sonsuz deney ve öğrenme var. İlerlediğimiz yolda birkaç başarısızlığımız oldu, kötü duvar tasarımı, işlerin büyüklüğünü

verirken yaptığımız tahminler, gereksiz burn-down grafikler gibi. Her seferinde yeni ve önemli bir şey öğrendik. *Eğer denemeyi bıraksaydık yeni ve önemli şeyler nasıl öğrenebilirdik?*

Kanban'ın başarısı yönetim takımlarını büyüledi ve Scrum Takımları da Kanban Duvarı'nı kullanmaya başladılar. Belki bu kitap yardımcı olur.

Retrospektifle Başlayın!

Düşünülmesi gereken birçok şey var değil mi? Umarım bu kitap sisin birazını ortadan kaldırır.

Eğer sürecinizi değiştirmeyi ve iyileştirmeyi düşünüyorsanız, düzenli retrospektiflere başlayın. Bu retrospektiflerin **değişime** yön verdiğinden emin olun. Gerekirse dışarıdan danışman getirin.

Bir kere başarılı retrospektifleriniz olduğunda kendi ortamınız içinde süreciniz evrimleşmeye başlayacaktır. Başarılı retrospektifler yapıyorsanız kullandığınız sürecin -Scrum, XP, Kanban, bunların bir karışımı ya da hiçbiri- ne olduğu önemli değildir.

Deney Yapmaktan Vazgeçmeyin!

Kanban ya da Scrum ulaşmak istediğimiz hedef değildir, ulaşmak istediğimiz hedef sürekli iyileştirmedir. Çevik yazılım geliştirmenin en güzel yanlarından biri kısa geribildirim döngüleridir, kısa döngüler öğrenmenin anahtar noktasıdır. Her şeyi sorgulayın, deney yapın, başarısız olun, öğrenin ve yeniden deney yapın. İlk seferde başarılı olmalıyım diye endişeye kapılmayın çünkü olamayacaksınız! Sadece bir yerden başlayın daha sonra evrimleşmeye başlayacaksınız.

Tek <u>gerçek</u> başarısızlık, başarısızlıktan bir şeyler öğrenemediğimiz zaman ki başarısızlıktır.

Bundan da bir şeyler öğrenebilirsiniz! ☺

/Henrik & Mattias, Stockholm 2009-06-24

Yazarlar Hakkında

Henrik Kniberg ve Mattias Skarin, Stockholm'de bulunan Crisp şirketinde danışman olarak çalışmaktadırlar. Şirketlere yazılım geliştirmenin hem teknik hem de insani yönünde yardımcı olurlar. Ayrıca onlarca şirkete Yalın ve Çevik ilkelerin pratiğe dökülmesinde yardım etmişlerdir.

Henrik Kniberg

Geçen on yıl içinde Henrik üç İsveç şirketinde teknolojiden sorumlu genel müdür yardımcısı olarak görev aldı ve birçoğuna süreçlerini iyileştirmeleri konusunda yardımcı oldu. Ayrıca Henrik lisanslı Scrum eğitmenidir, Yalınlık ve Çeviklik yaklaşımlarının öncülerinden Jeff Sutherland, Mary Poppendieck ve David Anderson'la beraber çalışır.

Henrik'in "Siperden Scrum ve XP" adlı kitabı 150.000 üzerinde okuyucuya ulaşmış, bu konudaki en popüler kitaplardan biridir. Uluslararası konferanslarda en iyi konuşmacı ödülünü birçok defa kazanmıştır.

Henrik Tokyo'da büyüdü ve şimdi Stockholm'de eşi ve üç çocuğuyla yaşıyor. Boş zamanlarında müzikle uğraşan Henrik, beste yapar ve yerel gruplarda bas ve klavye çalar.

henrik.kniberg@crisp.se

http://blog.crisp.se/henrikkniberg

http://www.crisp.se/henrik.kniberg

Mattias Skarin

Mattias, Yalınlık hakkında koçluk verir, yazılım şirketlerine Yalınlık ve Çeviklik konusunda destek olur. Yazılım geliştiricilerden yönetim seviyesine kadar farklı ölçülerde koçluk yapar. Yardım ettiği oyun geliştirme şirketi, oyun geliştirme süresini 24 aydan 4 aya düşürdü. Geliştirme bölümüne olan güvenin yenilenmesine yardım etti. Kanban'ın ilk uygulayıcılarından biri olarak öne çıktı.

Girişimci olarak iki şirket kurdu.

Kalite Yönetimi üzerine yüksek lisans eğitimini tamamladı ve 10 yıl süreyle kritik görevlerde yazılım geliştirici olarak bulundu.

Müzik, dans, araba yarışı ve kayaktan hoşlanıyor, Stockholm'de yaşıyor.

mattias.skarin@crisp.se

http://blog.crisp.se/mattiasskarin

http://www.crisp.se/mattias.skarin