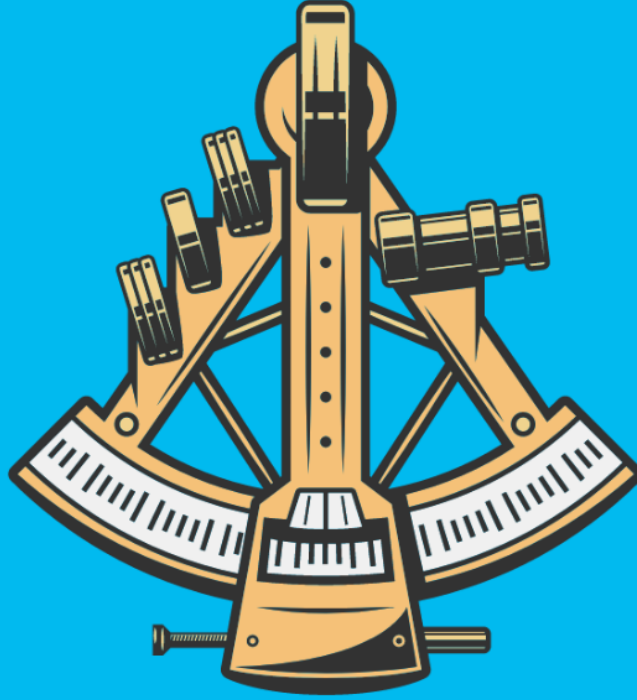


# YAZILIMCILIKTA KARIYER YOLU



Yazılımcı kariyeri hakkında kimsenin  
konuşmadıkları

**Mert Susur**

# Başlangıç

Bu kısa e-kitapta sizlere üniversiteden sonra yazılım alanındaki kariyerinizi nasıl şekillendirebileceğinizi ve bunu yaparken başınıza gelebilecek olan ileri dönem sorunlarından biraz bahsetmeye çalıştım.

Birçok yeni mezun için kısa vade hedefler sadece teknik anlamda gelişmek gibi görünse de artık modern yazılım dünyasında kariyer edinebilmek için sahip olmanız gereken iletişim ve liderlik becerilerinin de farkına varmaya başlayacaksınız. Özellikle çalıştığınız firmalarda yarattığınız etkiyi sadece yazdığınız kodla kısıtlamak istemiyorsanız bu kitapta bahsedilen yöntemlerin size rehber olacağını düşünüyorum.

Yeni mezun değil, yıllardır yazılım alanında çalışıyorsanız ama kariyerinizde bir tıkanma yaşıyorsanız, umarım bu kitaptaki iletişim becerileriyle ilgili kısımları okuduktan size biraz da olsa ilham vermeyi başarabilirim.

Gelin birlikte yazılım kariyeri hakkında kimsenin konuşmadıklarını ve kariyerinizi daha da ileriye götürebilmek için gereken marjinal farkı beraber aramaya koyulalım.

Bu alanda, yazılım kariyeriyle ilgili konuşulmayan birçok konuyu kendi blogumda yazmaya devam edeceğim. Bunun için <https://mertsusur.com> adresindeki haftalık yazılarıma ulaşmak için kayıt olmayı unutmayın. Eğer bu konularda yazdığım diğer küçük notları da takip etmek istiyorsanız bana twitter üzerinden <https://twitter.com/mertsusur> ulaşabilirsiniz.

<b>Başlangıç</b>	<b>2</b>
<b>Yurtdışı Macerası</b>	<b>4</b>
<b>Yurtdışı mülakatlarına hazırlık</b>	<b>5</b>
Teknoloji firmalarının mülakat süreçleri	5
<b>Yazılımcının Kariyer Merdiveni</b>	<b>12</b>
<b>Yazılım ekiplerinde yönetici olmadan lider olmak</b>	<b>13</b>
<b>Yazılımcılar için yöneticilik yolu</b>	<b>21</b>
Koçluk ve mentorluk	23
Liderler için koçluk	24
Peki nedir bu koçluk?	24
Geribildirim vermek	28
Bire bir görüşmeler ve ekiple kurduğunuz ilişki	28
Ekibinize onları önemseyişinizi göstermek	31
1:1 görüşmelerde ne konuşacağım?	33
Peki nasıl yöneticiliğe geçebilirim?	36
<b>Kapanış</b>	<b>37</b>
<b>Kaynaklar</b>	<b>38</b>

# Yurtdışı Macerası

Yaklaşık 15 yıldır yazılım geliştirme alanında profesyonel olarak çalışıyor olsam da kariyerimdeki en hızlı gelişmeyi yurtdışına çıkış yaptığımda yaşadığımı söylesem yalan olmaz. Özellikle en büyük tecrübeleri hızlı büyüyen, yani scale-up olarak nitelendirilen firmalarda yaşadım. Bu firmalarda yönetici olarak çalışırken kendinizi birçok farklı alanda geliştirmek için fırsat bulabileceğiniz gibi, çoğu zaman da ister istemez yaratıcı çözümler arayıp çözümleri optimize etmeyi öğrenmek zorunda kalıyorsunuz. Bu yüzden, herkese bu tarz bir firmada kısa bir süre de olsa çalışmayı kesinlikle öneriyorum.

Yurtdışı tecrübesinin kattığı bir başka artı da normal şartlarda Türkiye içerisinde karşılaşamayacağım kadar çeşitlilikte kişilerden oluşan ekipleri yönetme fırsatım olmasıydı. Farklı kültür ve geçmişlerden gelen bu kadar farklı kişiden sayısız şeyler öğrendim. Aslında öğrenmem sadece profesyonel değil, kültürel de oldu. Normalde okumayacağım birçok kitabı önerildiği için okuyarak kendimi geliştirdim, ya da zaten bildiğim düşündüğüm birçok konuda çok farklı bakış açılarını keşfetme fırsatı buldum.

Bu ekosistem içerisinde normalde karşılaşamayacağım “A Player” diye nitelendirilebilecek çok başarılı takım arkadaşlarım oldu. Bir çoğu önceden Facebook, Twitter, Google gibi firmalarda çalışmış ve o firmalardan öğrendiklerini etrafıyla paylaşan kişilerdi.

Yani demek istediğim, yurtdışında çalışmaya başlarken aslında işin bu yönünü hiç düşünmemiştim. Amacım sadece yurtdışında yazılım nasıl yapılıyor görmekken sonucunda yurtdışında çalıştığım firmalar adeta birer okula dönüştü benim için. Bu sayede kariyerimde hızlı bir ilerleme yapabildiğimi düşünüyorum.

Bu anlattıklarımla birlikte sizlere ilk başta yurtdışı mülakatlarından ve firmaların neler beklediklerinden bahsetmek istiyorum. Böylece benim gibi bu süreci deneyimlemek isteyen genç arkadaşlarıma yol göstermiş olabilirim.

# Yurtdışı mülakatlarına hazırlık

Az önce de anlattığım gibi, 2015 yılında kariyerime yurtdışında devam etmeye karar verdiğimde beni tam olarak nasıl bir sürecin beklediğini bilmiyordum. Hatta 2015 yılında Türkiye'deki teknoloji trendlerini ve hatta mülakat kültürünü düşündüğümüzde yurtdışındaki mülakatların bu kadar farklı olabileceği de aklıma gelmemişti doğrusu. Artık durum tabii ki çok farklı. Birçok yerli firma da yurtdışında uygulanan işe alım süreçlerini yavaş yavaş kullanmaya başladığı gibi işe alım yöntemleri de benzer bir hal aldı. Bunda muhtemelen birçok kişinin artık yurtdışı firmalarla görüşmeye başlayıp oradaki süreçleri çok yakından tanınmasının da katkısı var diyebiliriz.

Gelin şimdi yurtdışındaki teknoloji firmalarının mülakatlarını mercek altına alıp inceleyelim.

## Teknoloji firmalarının mülakat süreçleri

Teknoloji firmalarının mülakatları genellikle benzer bir takım adımları izliyor. Ancak bu mülakatlar kimi zaman daha kısa kimi zaman da daha uzun olabiliyor. Bu sürecin uzunluğu ve detayı biraz da firmanın hangi aşamada olduğuna bağlı olarak değişiklik gösteriyor. Bu yüzden biraz size karşılaşılabileceğiniz firmaları, biraz da çok fazla genelleyerek, anlatmaya çalışacağım. Unutmayın, bu genel bir sınıflandırma ve anlatımı basitleştirmek için köşelerinden epeyce kırıyorum. Amacım mülakat öncesi bir firmayı incelerken aklınızda bir fikir oluşması.

**Erken dönem girişimler;** A serisi yatırım turuna henüz girmemiş firmalar dersek çok da yanılmayız. Bu firmalar genellikle ölçeklenmeye uygun bir fikri uygulayabileceklerini düşündükleri bir pazarda bu fikri uygulamaya çalışırlar. Takım çoğunlukla 3-5 kişiden fazla değildir. Aradıkları profil genellikle Fullstack yazılımcılardır ancak bir alanda özelleşmiş roller de bazen bulabilirsiniz. Bu firmalar riskli işe alımlar yapabilir ve sizin potansiyelinize güvenerek size normalde alabileceğinizden daha yüksek pozisyonlar teklif edebilirler. Maaş skalası büyük rakiplerine oranla daha az olabilirken, size daha fazla düşük fiyattan (ya da bedava!) hisse alma opsiyonu verebilirler. Mülakatları genelde çok ciddi ve yorucu olmaz.

**Fonlanmış girişimler;** Genellikle fikrini ispatlamış, ya da ispatlayabileceğine yatırımcıları ikna etmiş firmalardır. Bu durumda A serisi yatırım turunu tamamlamış diyebiliriz. Artık hedefleri ürünlerini ölçeklendirmektir. Belli alanlarda özelleşmiş yazılımcılara ihtiyaç duymaya başlarlar örneğin; Front-End, Back-End ya da Infrastructure üzerinde geliştirme yapacak yazılımcılar vs. Bu aşamada firmalar hala riskli işe alımlar yapabilir ve size büyük roller teklif edebilirler. Maaşları yavaş yavaş rekabet etmeye başlayabilir ve genelde bu aşamdaki firmalar bol bol para yakarak

ilerlerler. Yani kazanç sağlamazlar hedefleri ürünlerini daha fazla kullanıcıya ulaştırıp bir sonraki yatırım turunda daha da büyümeye hazırlanmaktadır.

**Scale-up girişimler;** Fikirlerini ispatlamış ve artık ölçeklendirmeye başlayan firmalardır. Bunlar çılgınca büyüme hedeflerini hayata geçirmek için işe alım yaparlar. Artık belli alanlarda özelleşmiş takımlar oluşmuştur, Developer Enablement, Cloud Infrastructure, Data Enablement gibi value-stream dışında stratejik amaçlı takımlar kurulur. Bu gibi firmalar iyi kurulursa 100'lerce hatta 1000'lerce kişilik işe alım yapabilir ve aynı zamanda sekteye uğratmadan ürün geliştirmeye devam edebilirler. Hypergrowth aşaması da bunun bir alt kırılımıdır diyebiliriz. Bu aşamada firmaların çok güçlü liderlere ihtiyacı olur. Bu noktada, sizin liderlik yetkinliğiniz varsa teknik olarak çok güçlü olmasanız da teklif alabilirsiniz. Mülakatları, ulaşmak istedikleri ölçeklere bakarsak hızlı bir şekilde ilerleyebilir çünkü rolleri hızlı kapatmak isteyeceklerdir.

**Big Tech ve yerleşmiş girişimler;** Bu firmalar ölçeklenme hedeflerini başarmış, artık bunu sisteme oturtmuş belki de halka arzını tamamlamış firmalardır. Big Tech genelde Facebook, Twitter, Uber, Netflix gibi firmalar için kullanılırken Enterprise kavramı da çoğunlukla Microsoft, ve Oracle gibi firmalar için kullanılmaktadır. Bu firmalar artık dramatik bir şekilde büyüseler bile bunu nasıl yapacaklarını çok iyi öğrenmiş ve kültürlerinin bir parçası haline getirmişlerdir.

Tabii ki bunun kötü örnekleri de yok değil ama burada onlardan bahsetmiyorum. İyi örnekleri ise riskli kararlar vermeyi tercih etmezler. Çünkü önceki aşamalarda yaptıkları riskli kararlarla bu noktaya gelmelerine rağmen pazarda ciddi bir rekabetin içinde oldukları için her zaman "A Player" dediğimiz en iyi yetenekleri en yüksek maaşla işe almaya çalışırlar. Firmanızda Principal, Staff, Senior engineer olduğunuz halde buralarda size orta seviye yazılımcı teklifleri gelebilir, ve sürpriz bir şekilde bu teklifler sizin mevcut maaşınızdan %50 daha fazla bile olabilir!

Şimdi gelelim mülakat sürecine. Yazılımcı mülakatları 3 ana temadan oluşmasına rağmen bu mülakatlar firmaların hangi aşamada olduklarına ve risk iştahlarına göre farklılık gösterirler.

## **Sistem Tasarımı ve Mimari**

Bu görüşmede mülakatı yapan kişi sizin iletişim ve problem çözme becerinizi ölçmeye çalışır. Bunu yaparken de birçok farklı sinyalden faydalanır;

- Görüşmeyi yapan kişinin sorduğu soruyu anladınız mı?
- Anlamaya çalışırken nasıl sorular soruyorsunuz, sorduğunuz sorularla konuya hakimiyetinizi gösterebiliyor musunuz?

- Sorunu anladıktan sonra farklı “edge case”leri düşünebiliyor musunuz?
- Bilmediğiniz bir konuyla karşılaşırsanız bununla nasıl baş ediyorsunuz?
- Problem çözerken aklınızdakini net bir şekilde anlatabiliyor musunuz?
- Soruyu soran sizi sınarken ona nasıl tepki veriyorsunuz? Korumacı mısınız yoksa açık fikirli mi?
- Varsayımlar yaparken bunların varsayım olduğunu fark edebiliyor musunuz?
- Tasarımı yaparken süreci siz yönetebiliyor musunuz?
- Aklınızdaki çözüme giderken yanlış olduğunu fark edince nasıl pivot ediyorsunuz?
- Direkt çözümün detaylarına dalıp soru soranı unutup kendiniz mi çözüm üretiyorsunuz?

Yukarıdaki gibi sinyaller sizin bu görüşmelerdeki başarınızı ve hatta başarılı olursanız da sizin alacağınız teklifin seviyesini belirlemek için kullanılır. Bunlardan çok fazla “Red Flag” almadan, olumlu sinyaller toplarsanız başarılı oldunuz demektir. Genellikle büyük teknoloji firmalarında bu mülakat iki farklı round şeklinde karşınıza çıkabilir. Ancak çoğunlukla 1 saat civarında sürer. Görüşmeyi uzaktan yapacağınızı varsayarsak genelde miro, lucidchart gibi araçlar kullanarak sizden kutular çizmeniz beklenir.

Bu konuda size önerim; Alex Xu’dan System Design Interview: An Insider’s Guide<sup>1</sup> kitabını incelemeniz. Kitap size ölçeklenebilir bir sistem mimarisi kurabilmeniz hakkında yardımcı oluyor. Aynı zamanda da bu sorulara nasıl yaklaşacağınız hakkında adım adım anlatımlarda bulunuyor. Şimdi “benim bu kitaba neden ihtiyacım olsun iki kutu çizebilirim ne olacak ki” diyebilirsiniz. Eminim sorulacak soruların cevabını biliyorsunuzdur, ancak çoğu zaman bu sorunları milyarlarca kullanıcı, yüzbinlerce istek için ölçeklememiş olabilirsiniz. Dolayısıyla bunu tecrübe etmediyseniz kitabın sizin bakış açınızı değiştireceğine inanıyorum. Bu konuda kendinizi daha da geliştirmek istiyorsanız Kevin Modzelewski’nin 2012 yılında Stanford’taki “How We’ve scaled Dropbox”<sup>2</sup> sunumunu ve Educative.io üzerinden erişebileceğiniz **Grokking the System Interview** eğitimini de inceleyebilirsiniz.

Eğer hedefiniz büyük verilerle çalışan firmalarsa ve sizin de bu alanda çok fazla tecrübeniz yoksa kesinlikle Martin Kleppmann’ın “Designing Data-Intensive Application”<sup>3</sup> kitabına da göz atmanızı öneririm. Özellikle büyük teknoloji firmalarıyla yapacağınız görüşmelerde cevaplayacağınız soruların ölçekleri de çok fazla olacağı için bu kitapta anlatılanlara kulak aşinalığı bile olması sizin yaratacağınız etkiyi değiştirebilir.

---

<sup>1</sup> <https://amzn.to/3ryyFoU>

<sup>2</sup> <https://www.youtube.com/watch?v=PE4gwstWhmc>

<sup>3</sup> <https://amzn.to/3rGRw5b>

## Kodlama Mülakatı

Bu mülakatı yapan kişiler sizin kullandığınız yazılım diline olan hakimiyetinize, teknik yetkinliğinize, aklınızdaki çözümü nasıl anlattığınıza ve nasıl test ettiğinize dair sinyaller toplamaya çalışırlar. Yani konu önceki turda olduğu gibi sadece problemi çözmeniz değil, aynı zaman nasıl test ettiğiniz ya da farklı test senaryolarını nasıl gördüğünüz ile de ilgili. Bu görüşmenin yıllar içinde iki farklı çeşidini gözlemledim. Büyük teknoloji firmaları genellikle soru havuzlarından rastgele 2-3 soru seçip sizin bu soruları çözmenizi bekliyorlar. Daha küçük firmalar da daha spesifik bir problem verip sizden bu problemi uçtan uca kodlamanızı bekliyorlar. Buna bağlı olarak da bu mülakattan beklenti değişebiliyor. Eğer ilk tarz soruları merak ediyorsanız, benim önereceğim kitap Gayle Laakmann McDowell'den "Cracking the coding interview"<sup>4</sup> kitabı. Ancak bu kitaba sadece mülakata hazırlanmak olarak bakarsanız kesinlikle yanılırsınız. Çünkü kitabın size öğretmeye çalıştığı bir problem çözme metodu. Kısaca özetlemek gerekirse;

1. Problemi çok iyi anla ve varsayımlar yapıyorsan bunları açıkça dile getir ve elindeki verilere bak.
2. Varsayımlarını ay, yıl ve onlarca yıllık tahminlere oturt. Örneğin saklayacağın veriyi 100mb olarak hesapladıysan uzun yıllar sonunda nasıl bir alana ihtiyacın olacağını düşün ve problemi uzun vadeli çözmeye çalış.
3. Problemi ilk başta brute force şeklinde çözmeye çalış. Gereksiz optimizasyonlardan kaçın ancak bunun bilincinde olduğunu açıkça dile getir.
4. Problemi çözdükten ve testlerini yaptıktan sonra optimizasyon noktalarını ve hata durumlarını belirle.
5. Eğer vaktin varsa bunları uygula, yoksa bunları açıkça dile getir.

Kolay duyuluyor değil mi? Ancak zorluk detaylarda gizli. Kitap bu yüzden birçok optimizasyon yönteminden ve bunları nasıl uygulayabileceğinizden bahsediyor. Bunlar aslında computer science derslerinden aldığınız bilgiler olabilir, ancak uygulamaya geldiğinde işler biraz değişebiliyor. Eğer kitabı almak istemiyorsanız leetcode, hackerrank gibi sitelerden de sorular çözerek kendinizi geliştirebilirsiniz.

Gözlemlediğim ikinci kodlama mülakatı ise size verilen bir problemi bir ürünmüş gibi ele alıp çözüm düşünmeniz yönünde. Örneğin finansal hesap yapan uygulamanın bir servisi olabilir, ya da hazır bir API'den veri çekerek bunları anlamlandırmak üzerine olabilir ya da bir veri seti üzerindeki verileri hesaplamalardan geçirip sonunda bir çıktı üretmek üzerine olabilir. Bu gibi görüşmelerde görüşmeyi yapan aşağıdaki sinyalleri toplamaya ve buna bağlı olarak sizin tecrübenizi seviyelendirmeye çalışır.

---

<sup>4</sup> <https://amzn.to/3GzBeiB>



Bu aşamada toplanacak sinyaller aşağıdaki gibidir:

- Planlama aşamasını nasıl atlattı? Varsayımlar yaptı mı? Varsayımlarını sorular sorarak anlattı mı?
- Kodun okunabilirliği nasıl? Kullandığı değişken ve fonksiyon isimleri mantıklı mı?
- Çözüm yeterince esnek mi? Değilse nedeninden bahsetmiş mi?
- Çözümün performansını düşünmüş mü? Tercihlerini bilinçli mi yapmış?
- Test edilebilir bir kod mu yazmış? Test yazmış mı? Yazmadıysa nedenini anlatmış mı?
- Hata yönetimini düşünmüş mü? Ne gibi hatalarla karşılaşabileceğinden bahsetmiş?
- Verilen süre içinde ne kadarlık bir kısmı çözebilmiş? Çözdüğü kısım doğru sonuç veriyor mu? Tümünü çözdüyse, uygulama çalışıyor mu?

## Davranış Mülakatı

Kimi firmalar bu aşamaya “Culture fit” ya da “values fit” gibi isimler de koyabiliyorlar. Ancak genelde “culture fit” ismi farklı bir çağrışım yaptığı için tercih edilmiyor. Burada her firmanın farklı bir yaklaşımı olabiliyor, kimi firmalar kendi değerlerini baz alarak bir mülakat formatı hazırlıyor, kimisi senin sorunlar karşısındaki davranışını anlamaya çalışıyor, bazıları da tecrübenden nasıl firmalarda çalıştığını görmeye çalışırlar.

Bu aşamada önemli olan bir kaç nokta var;

1. Her zaman cevaplarınızı verirken gerçek tecrübelerle dayandırın. Mümkün olduğunca ayakları yere basan örnekler verin. Örnekler verirken de bağlamı da vermeyi unutmayın ve sonunda kesinlikle sonuçlarından bahsedin. Bunun için STAR<sup>5</sup> yöntemini kullanabilirsiniz. **S**ituation, **T**ask, **A**ction, **R**esult.
2. Unutmayın çoğu durumda bir tane doğru cevap yok. Yani firmadaki yazılımcı sayısı, içinde bulunduğu kısıtlar, çalıştığı sektör ve o an içinde bulunduğu durumu da hesaba katarak opsiyonlardan bahsedebilirsiniz. Örneğin iyi bir proje nasıl geliştirilir sorusuna birçok farklı yanıt verebilirsiniz, ya da iyi bir CI pipeline nasıl olur dersiniz o da yazılımcı sayısı, teknoloji, firmanın süreçleri gibi birçok farklı koşula göre değişebilir.
3. Her zaman idealinin ne olması gerektiğinden bahsedin, ancak cevaplarınızı gerçek hayat örnekleri vererek somutlaştırın. Bu sizin tecrübenizin bir yansıması olacaktır.
4. Asla yalan söylemeyin. Bilmediklerinizi bilmiyorum demenizin bir sakıncası yok.

---

<sup>5</sup> [https://www.vawizard.org/wiz-pdf/STAR\\_Method\\_Interviews.pdf](https://www.vawizard.org/wiz-pdf/STAR_Method_Interviews.pdf)

5. Başarısızlıklarınızı anlatmaktan çekinmeyin, ama onlardan ne öğrendiğinizi ve çıkardığınız dersleri paylaşın.
6. Mülakat öncesi firmayı inceleyin, ne iş yaptığını çok iyi anlamaya çalışın anlamadığınız yerleri bu görüşmede sorun. Aklınızda “Neden bu firmada çalışmak istiyorum” sorusuna bir cevap olsun.

## Mülakat Sonrası

Mülakat sonrasında “hire” ya da “no-hire” kararını size görüşmeyi yaptığınız işe alımcı iletecektir. Eğer başarılı olduysanız ya size başvurduğunuz pozisyonda bir teklif ile gelecektir ya da minimum çıtayı karşılıyor ancak tam o seviye değilseniz daha düşük bir seviyede teklif gelecektir. Gelecek olan teklif yıllık baz maaşın yanında bir kaç ekstra ile de gelebilir ya da gelmeyebilir. Genellikle maaştan bahsederken toplam geliri göz önünde bulundurmanızı öneririm. Bu ne demek mi?

1. Baz Maaş (Base Salary)
2. Hisse opsiyonu (Stock Options) ya da yıllık hisse miktarı (RSU)
3. Yan haklar
4. Bonus

Bu dört maddenin toplamına toplam maaş derken, baz maaş farklı olabilir. Hisse opsiyonu genellikle 1 yıl çalıştıktan sonra size tanınır ve sonraki 3 yıl boyunca aylık bir şekilde ödenir. Mesela bir firmaya girdiğinizde size 48,000 hisse opsiyonu verdilerse, ilk yılın sonunda 12,000 hisse size atanır, sonraki her ay 1,000 hisse alırsınız.

Firma bir kurum tarafından satın alınırsa ya da halka arz edilirse bu hisselerinizi kullanarak o günki hisse fiyatı kadar para kazanırsınız. Hisse opsiyonlarının bir de strike price denen bir ödemesi vardır. Mesela önceki örneğe dönecek olursak,

Firma size 48,000 hisse verdi. 18 ayın sonunda siz 18,000 hissenin hakkına sahip oldunuz. 18. Ayınızda da firma halka arz gerçekleştirdi. Bu durumda öncelikle bu hisseleri kendinize geçirmeniz gerekli. Bu noktada da  $18,000 \times \$2$  (strike price) toplamda \$36,000 ödemeniz gerekiyor. Böylece hisselerin hakkı size geçmiş olacaktır. Sonrasında da arz sırasında hisseleri \$30’dan satarsanız karşılığında \$540,000 kazanmış olacaksınız. Tabii ki bundan vergi ödemeniz gerekecektir. Bu yüzden bu gibi konuları hangi ülkede bu başınıza geliyorsa o ülkede yetkin bir vergi danışmanına danışmanızı öneririm.

Sonuç olarak gelen tekliften sonra teklifi iyice değerlendirmeniz çok önemli. Bunun yanında bir de tüm mülakatların sonunda genellikle size 10-15 dakikalık bir zaman verilir ve burada mülakatı yapana sorular sorabilirsiniz. Benim önerim bu zamanı en iyi şekilde değerlendirmeye çalışmanız olacaktır. Örneğin, “Firma kültürünüz nasıl?” “hangi teknolojileri kullanıyorsunuz” gibi sorular sormak yerine, sizinle mülakat yapacak kişilerin kariyerlerine mülakat öncesi göz gezdirip onlara da bunun sinyalini

vererek neden bu firmayı tercih ettiğini sormanız size daha güzel bir resim çizmeye yardımcı olacaktır. Örneğin “Gördüğüm kadarıyla X, Y firmalarında çalışmışsın daha önce, neden Y’yi bırakıp da buraya geçtin? Sence neden bu firmayı tercih etmeliyim?” ya da en sevdiğim zor sorulardan biri olan “Sence firmada iyi gitmeyen neler var?” sonrasında da “Neden bunu değiştirmiyorsun?”. Bir başka örnek de “Günde kaç defa deployment yapıyorsunuz?”, “Deployment yapmak ne kadar zamanını alıyor?”, “Neden o kadar yavaş?”, “Bunu neden düzeltmiyorsunuz?”. Bunu önermemin nedeni şu, eğer soruyu direkt olarak sorarsanız size firma kültürü hakkında pazarlamacı lafları söyleyecekler olmaları. Bunun yerine firmanın kültürünü anlamak için sorunuzun ayaklarını yere bastırmanız. Böylece hem karşı taraf sizin soracağınız sorularla tecrübenizi daha iyi anlayacağı gibi aynı zamanda da daha dürüst cevaplar alabileceksiniz.

Bu bölümde bahsettiklerim sanki sizi üniversite sınavına hazırlıyormuşum gibi duyulabilir. Ancak birçok firma sürecini sizi en iyi şekilde tanımak ve değerlendirmek için hazırlıyor. Yani günün sonunda sizin bu mülakatlardaki başarınız aynı zamanda sizin teknik yetkinliğinizin ve tecrübenizin bir yansıması olacak. Yani diyorum ki, bu kitapları sanki bir üniversite sınavına hazırlanır gibi değil, daha iyi bir yazılımcı olmak için incelemeye alırsanız dramatik bir şekilde farkı göreceksiniz. Bu sözüm özellikle üniversite öğrencisi arkadaşlarıma! Geriye dönüp baktığımda, keşke bu içeriklere üniversitedeyken ulaşmış olsaydım ya da okuldaki hocalarım bunları bana öğretmiş olsalardı diyorum. Ancak biliyoruz ki üniversite eğitimi çoğu üniversite için, kendi içinde çok rijit ve standartları olan bir müfredata bağlı. Dolayısıyla eğer yazılım alanında bir kariyer kovalamak istiyorsanız, hatta çok iyi bir yazılımcı olmak istiyorsanız bu bilgileri üniversitede edinip bol bol tecrübe etmek adına kod yazmanızı öneririm.

# Yazılımcının Kariyer Merdiveni

Zaman içerisinde tanıdığım neredeyse tüm yazılımcıların ister istemez kariyerleri ile ilgili kaygı duyduklarına tanık oldum. Özellikle önlerinde iyi bir örnek olmayan yazılımcılar bir zaman sonunda ya yaptıkları işlerden sıkılıyorlar, ya da firmada daha fazla ileri gidemedikleri için iş değiştirme kararı vermeye başlıyorlar.

birçok teknoloji firması yazılımcıları firma içerisindeki etki alanlarına göre farklı seviyelere ayırmayı tercih ediyorlar. Gelin şimdi bu yazılımcı kariyer adımlarını biraz tartışalım:

**Başlangıç seviyesindeki yazılımcı:** Başlangıç seviyesindeki yazılımcı ya başka bir kariyerden yazılımcılık kariyerine geçmiş ya da üniversiteden yeni mezun olmuş ve takım çalışmasına hatta belki de doğru pratiklere maruz kalmamış yazılımcılardır diyebiliriz. Genelde bu seviyedeki yazılımcılar öğrenmeye odaklanırlar, herhangi bir alanda uzmanlaşacak kadar tecrübeye sahip olmadıkları için sürekli farklı alanlara zıplarlar. Bir firmaya başlangıç seviyesinde giriş yapmış yazılımcının 1 bilemedin 2 yıl içerisinde bir sonraki seviyeye atılmasını bekleriz. Bu yazılımcının genellikle ekibindeki diğer tecrübeli yazılımcılardan çok fazla destek alması önemlidir.

**Katkı sağlayan yazılımcı:** Başlangıç seviyesini başarılı bir şekilde atlatan yazılımcılar 1-2 ya da 3 yıl içerisinde artık kendi başlarına etki yaratmaya başlarlar. Bu seviyedeki yazılımcılar artık kendilerine bir uzmanlık aramaya başlarlar. Bu uzmanlık bir yazılım dili olabileceği gibi bir platform da olabilir. Bu seviyedeki yazılımcı belirsizliği olan problemleri ufak ufak çözebilmeye başlar. Ancak zor ve belirsizliği yüksek yazılım problemleri için çözüm üretirken ekibindeki tecrübeli yazılımcılardan destek alması iyi olur. Bu yıllarda kendisini teknik yönden geliştirirken aynı zamanda iletişim becerilerine de zaman ayıranları ilerleyen süreçlerde daha hızlı ilerleyebilirler. Genelde yazılımcılar bu seviyede 2-3 yıl daha kalırlar.

**Takım seviyesinde liderlik yapan yazılımcı:** Nam-ı diğer Senior yazılımcı. Bu seviyeye genelde 5-8 yıl arasında gelinir. Bu seviyedeki bir kişi yeni bir işe başladığında nereden başlayacağını bilir ve hızlıca etki yaratmaya başlar. Yarattığı etki çoğunlukla takımının içindedir. Bu yazılımcıdan teknik liderlik yapması beklenir. Yani, takımındaki diğer arkadaşlarına mentorluk yapması, teknik problemleri çözerken yol göstericilik yapması ve belirsizliği yüksek olan problemleri çözebilecek kadar tecrübe biriktirmiş olması önemlidir. Hatta bu seviyede artık iletişim becerileri de önem kazanır.

Bu üç temel seviyeyi atlatan yazılımcılar artık hayatlarında 30'lu yaşlarına gelmiş ve ufak bir kaygıyla ve örnek eksikliği ile birlikte "50 yaşına kadar kod mu yazacağım" diye hayatı sorgulamaya başlamışlardır. Bu noktada artık önlerine iki kariyer yolu çıkar;

1. Bireysel Katılımcı Yolu (Individual Contributor)
2. Yönetici Yolu (Manager)

Bu yol ayrımı, birbirinden farklı iki kariyere açılır. Modern teknoloji firmalarında bu iki yol arasında geçiş kolaydır, ancak daha eski kafalı firmalar bu yolları öyle kesin ayırmıştır ki bu yollar arasındaki geçişi başarısızlık olarak bile nitelendirebilirler. Ancak modern teknoloji firmaları bunun bir kariyer tercihi olduğunu çok iyi bilir ve bu geçişin terfi ya da başarısızlık olmayacağı bir kariyer yolu hazırlarlar.

Şimdi ilk olarak bireysel katılımcı yolu ile başlayalım.

## Yazılım ekiplerinde yönetici olmadan lider olmak

Profesyonel olarak yazılım geliştirmeye başladığımda etrafımda gerçekten çok iyi yazılımcılar vardı. En azından o zamanki bilgimle yaptığım birçok hatada bana yardımcı oldular, hatta öyle ki basit bir XML parser yazarken bile haftalarımı almasına göz yumarak benim hata üzerine hata yapmama izin verdiler. Sonrasında her zaman gittiğim firmalarda rol model alabileceğim yazılımcıların olmasına hep dikkat ettim.

Bir yazılımcı olarak kendinizi geliştirmeniz için size neden yanlış yaptığınızı söyleyecek, sizden daha tecrübeli yazılımcıların olması çok önemli. Özellikle uzmanlaşmak istediğiniz alanda bulabildiğiniz rol modellerle çalışmak sizin ilerlemenize inanılmaz bir katkıda bulunuyor. Ancak kariyerinizde yazılımcı olarak ilerlediğinizde bir noktada, siz firmadaki “o” yazılımcıya dönüşmeye başlıyorsunuz. Durum böyle olduktan sonra da kendinizi ilerletmek, bir üst seviyeye geçmek gerçekten çok güçleşiyor. Özellikle küçük firmalarda çalışıyorsanız bu sorun önünüze daha hızlı çıkıyor.

Kariyerinizde ilerleyebilmek için tecrübeniz arttıkça yaptığınız işlerde yarattığınız etkinin de artmasını sağlamanız gerekiyor. Ancak bu yazılımcılık için o kadar da kolay olmayabilir. Örneğin kariyerinizin başında bir birim kod yazıyorsanız kariyeriniz ilerlediğinde daha çok birim kod yazıyor olabilirsiniz. Fakat yarattığınız etkiyi arttırmayı sadece yazdığınız kod ile ilişkilendirirseniz bu durumda her yıl önceki yıldan çok daha iyi kod ve fazla kod yazmayı düşünmeniz gerekir. Belli bir zaman sonra üreteceğiniz teknik çıktılar da bir üst limite ulaşacağı için artık kariyerinizde daha fazla ilerleyemezsiniz. Tabii ki bu bir tercih ama eğer daha ileriye gitmek istiyorsanız sizin etkinizi arttırmak için kullanacağınız başka bir takım çarpanlara ihtiyacınız var demektir. Bence bir yazılımcının bu üst limit kırabilmesi için kazanması gereken bir kaç tane önemli çarpan var;

1. **Projelerde liderlik yapmak:** Projelerdeki belirsiz ya da net olmayan sorunların çözülmesine ve kararlı bir hale ulaşmasına yardımcı olmak. Ekibinde bir

çarpana dönüşerek ekibin başarılı olması için çalışmak. Böyle bir yazılımcının ekibinde 5 kişi varsa, yarattığı etki bu teknik yetkinliği ve tecrübesiyle birlikte katlanarak artacaktır.

2. **Teknik mentorluk:** Ekibindeki az tecrübeli takım arkadaşlarının kariyerlerinde ilerlemesine yardımcı olarak sadece o takım içinde değil, o kişilerle kurduğu uzun süreli ilişkiler sayesinde de hatırı sayılır bir etki yaratacaktır.
3. **Uzmanlık alanı:** Yazılımcının bir alanda uzmanlaşması çalıştığı firmalarda sorusu olan herkesin ona gelmesine neden olacağı gibi bir de üstüne yukarıda saydığım diğer alanlarda da etkili olması için de fırsat yaratacaktır. Bu fırsatları da değerlendirebilmesi için gerekli donanıma sahip olması çok önemlidir.

Gelin bu bölümde size çok sevdiğim yazarlardan biri olan Will Larson'un "Staff Engineer: Leadership beyond the management track"<sup>6</sup> kitabından alıntılar yaparak bireysel katılımcı yolunu biraz daha detaylı bir şekilde tartışalım.

Kitap başlangıçta Staff Engineer rolünü tanıtarak başlıyor. Ancak tecrübeli olanlar bilecektir, staff engineer denilen rol birçok farklı firmada farklı isimlerle kullanılıyor. Hatta firmaların ihtiyaçlarına, yaptıkları işe, firmanın dinamiklerine göre de şekillenen bir rol bu. Bu yüzden yazar burada öncelikle 4 farklı arketipten bahsediyor. Bunlar kitapta anıldığı şekliyle aşağıdaki gibi tanıtılıyor.

1. Teknik Lider
2. Mimar
3. Sorun çözücü
4. Üst düzey yöneticinin sağ kolu


Tüm kitapta da bu arketipler ve aralarındaki farklılıklara vurgu yapıyor. Bu tanımları oturtuktan sonra da Staff Engineer için belirlenen çıtayı tanımlıyor. Bence kitabın etkili olduğu yerlerden birisi de burası. Bu konuyla ilgili yaşadığım sorunları yazının ilerisinde tartışmak üzere not alıyorum. Ancak burada güzel olan birebir gündelik profesyonel hayatınıza uygulayabileceğiniz güzel örnekleri barındırması. Bu bölümü okuduktan sonra aklınızda Staff Engineer rolüyle ilgili güzel bir tanım kalıyor. Bu bence çok önemli çünkü eğer neyi hedeflediğinizi ve bu hedeflediğiniz rol için gereken minimum çıtayı bilerseniz hem kendi seviyenizi daha objektif bir şekilde değerlendirebilirsiniz, hem de eksikliklerinizi daha iyi görerek hangi alanlarda kendinizi yetiştirmeniz gerektiğini keşfedebilirsiniz.


Gelin bu kısmı biraz daha açalım.

---

<sup>6</sup> <https://amzn.to/3uaAMEV>

Staff Engineer rolü çoğunlukla karşımıza orta ya da büyük ölçekli firmalarda çıkıyor. Türkiye’de birçok farklı firmada yazılım mimarı gibi rolleri görürken büyük teknoloji firmalarında (FAANG) ise Staff Engineer ya da Principal Engineer rollerini görüyoruz. Bu konuda Gergely Orosz’un güzel bir paylaşımına denk geldim ki paylaşmadan edemeyeceğim.



**Gergely Orosz** • Following  
Writing The Pragmatic Engineer. Advisor at mobile.dev. Follow me for e...  
5h • 

...

There is no "software architect" title or position in the most innovative tech companies. This is no mistake.

Everyone on the team is empowered to contribute to - and feel like they own and influence - design decisions at those companies.

How you call positions matters.

[#bigtech](#) [#startups](#) [#softwareengineering](#) [#advice](#)

Bu alıntıda Gergely, Yazılım Mimarı pozisyonunun yenilikçi firmalarda olmamasının tesadüf değil, tasarım gereği olduğunu söylüyor. Gerekçe olarak da bu rolün ekiplerdeki diğer yazılımcıları yetkilendirmeyi olumsuz etkilediğini, kararlara dahil olmalarına bir engel olduğunu söylüyor ve bu isimlendirme değişikliğinin önemine vurgu yapıyor.

### **Peki o zaman, nedir bu Staff Engineer?**

birçok firmada farklı görev ve sorumlulukları olabilir ancak ben çoğunlukla karşılaşılan tanımına göre bir tanım yapmaya çalışacağım ve kitaptaki örneklerden bahsedeceğim. Staff engineer, çoğunlukla, senior engineer rolünden sonraki kariyer adımında yer alıyor. Senior staff engineer ve sonrasında da çoğu zaman principal engineer rolü geliyor. Kimi firmalar senior engineer sonrasında principal engineer rolünü de kullanıyorlar. Ancak genellikle büyük teknoloji firmaları için konuşursak staff engineer, senior engineer rolünden bir sonraki kariyer adımıdır diyebiliriz.

Bunu bu kadar basit bir şekilde özetlemiş olsak da çoğu zaman birçok kişi bu tarz firmalarda staff engineer rolüne gelmeden önce senior engineer rolünde uzun süre kalabiliyorlar. Çünkü bu geçiş genellikle hem bakış açısı tarafından hem de çoğu zaman kişinin organizasyonda yarattığı etki açısından ciddi farklılıklar gösteriyor. Buna şimdilik İngilizce bir terim olarak "Sphere of Influence (SoI)" ya da Türkçesiyle

“Etki Alanı” diyelim. Örneğin bir senior yazılımcının etki alanı kendi takımı iken, staff rolüne geçtiğinde etki alanı grup, organizasyon ya da firma seviyesinde olabiliyor. Bu durum haliyle o kişinin kimlerle “networking” yaptığını da etkiliyor. Küçük firmalarda, staff engineer çoğunlukla CxO seviyesinde bir iletişim ağına sahipken, firmalar büyüdükçe bu Group Engineering/Product Manager seviyesinde olabiliyor. Ancak eğer senior engineer için “networking”i düşünürsek, o zaman etki alanı genellikle takımındaki yazılımcılar, diğer takımlardaki yazılımcılar, takım lideri, takımın Product Manager’ı ya da Engineering Manager’ı olabiliyor.

Bu farklılıkları düşündüğümüz zaman, Staff Engineer rolünün gündelik iletişimde olduğu kişilerin gerek tecrübesi gerekse de yetkinlikleri açısından bakarsak, teknik yetkinliğinin yanı sıra, iletişim ve liderlik becerilerinin de gelişmiş olması gerekiyor. Yani buradan yakalayabileceğimiz birinci önemli ipucu; **Liderlik ve Güçlü İletişim becerisi/tecrübesi.**

İletişim becerisi dediğimizde bu tanımın altını da doldurmak gerekli elbette! Benim gördüğüm en efektif staff engineer’ler, bir sorunu ele alırken farklı açılardan bakabiliyor, avantaj ve dezavantajlarını ürünün başarısıyla ilişkilendirip bir şekilde sonuçlarıyla anlatabiliyorlar. Yani bunu bu şekilde yaparsak, sonucunda şu kadar kazanç sağlarız ya da ekiplere şöyle bir katkısı olur gibi konuları güzelce anlatabiliyorlar. Yeterince ayakları yere basmadı değil mi? Gelin bir örnek üzerinden gidelim.

Diyelim ki ekibin ürün yöneticisi ekibe yaptığı çalışmalar sonrası keşfettiği bir fırsatla geldi. Ekip de bu ürün fırsatını değerlendirmesi için gereken çözümü aramaya koyuldu. İşte bu aşamada ekibin “scoping” dediğimiz süreci başlamış oldu. Scoping aşamasında ekip kapsamı, ellerindeki çözüm yöntemlerini inceledi ve 3 tane alternatif oluşturup tartışmaya koyuldular. Diyelim ki çözümlerden biri ekip tarafından destek görüyor ve çoğunlukla ikna olmuş durumdalar. Ancak bunun sonucunda platformun performansı da etkilenecek. Bu durum, ekip seviyesinde basit bir şekilde “bunun performans sorunları olacaktır” diyerek geçilebilir. Eminim ekip içerisindeki herkes bunun ne demek olduğunu çok iyi bir şekilde de anlayacaktır. Ancak bunu ekibin dışındaki paydaşlara anlatmak istediğinizde bunun sonuçlarını daha iyi anlatmanız gerekir. Çünkü muhtemelen ürün ekibinin paydaşları teknik kişiler olmayacaklardır. Dolayısıyla onlara teknik metrik ve yöntemlerden bahsedemezsiniz. Mesela, bu performans sorunu yıllık ciroya etki yaratacak mı? Ya da ekip bu sorunu ne kadar geç çözmeli? Bu durumda ekip bu geliştirmeye başlarken bu sorunu çözerek mi başlamak istiyor? Eğer öyleyse bunun nedeni nedir? Her ne kadar bu örnekte bir staff engineer rolü olmasa da, ekipteki teknik liderin bu sorunu etkileriyle anlatması, ekibin yaratacağı etki için çok önemli.



Gelin şimdi biraz daha geniş kapsamlı bir problem düşünelim. Bir ya da birden fazla ekibin üretkenliğini etkileyen bir problemle karşılaşıldı ve farklı ekiplerdeki yazılımcılar sorunu çözenin bir yolunun uçtan uca testlerden vazgeçilmesi olduğunu düşünüyorlar. Bu durumda bu probleme teknik olarak liderlik sağlayacak birisinin olması gerekiyor. Bu kişinin aynı zamanda böyle bir yatırım yapılacaksa bunun sebeplerini, paydaşlara anlatarak onları ikna etmesi de gerekiyor. Bu gibi bir durumda bu liderliği bir senior yazılımcının da üstlenmesi mümkünken bu durum ekibin problemi olmaktan çıkmış ve daha büyük bir soruna evrilmiş duruyor. Dolayısıyla konuya daha yukarıdan bakabilecek teknik bir kişiye ihtiyaç olabilir. Mesela bir Staff Engineer! Bu ölçekte bir problemi çözmek için sahip olunması gereken yetkinlikler de senior engineer rolüne göre daha farklı. Öncelikle ekipler arasında uzlaşma yaratabilmeli dolayısıyla iletişim becerileri güçlü olmalıdır, bunun için de ekibi aktif bir şekilde dinleyip onların fikirlerini sınavabilecek teknik yetkinlikte olmalıdır. Sonrasında da firmanın hedeflerini anlayıp, burada farklı alternatifler üretebilmesi ve nihayetinde de bunları bir şekilde farklı seviyelere, onların anlayacağı şekilde anlatabilmelidir.

Staff engineer'den çok mu şey bekliyoruz sizce? Bence evet, bu yüzden de bu rol gerçekten zor doldurulan bir rol.

Çalıştığım birçok yerde bu rolü doldurmak her zaman sorun olmuştu. Fakat gördüğüm en büyük problem her zaman bu rolün tanımını yaparken kesin kalıplara oturtmaya çalışılmasıydı. Çünkü her firmanın içinde bulunduğu dinamiklere göre bu rolün kapsamı ve etkisi ciddi farklılıklar gösterebilir. Örneğin, orta ölçekli bir startup'ta bu roldeki bir kişiden beklenti, ortak yazılım standartlarını uzlaşmacı bir şekilde belirleyip gelecekteki sorunlara hazırlık yapmakken, scale up ya da hypergrowth aşamasındaki bir startup için liderlik ve mühendislik kültürüne odaklanmak olabilir. Dolayısıyla bu rolden beklenti zaman içerisinde değişeceği için rolün tanımından çok iletişim, liderlik ve teknik yetkinliklerini iyi tanımlamak rol için gereken çitayı belirlemek için daha sağlıklı olacaktır.

### **Bir senior engineer nasıl staff engineer olabilir?**

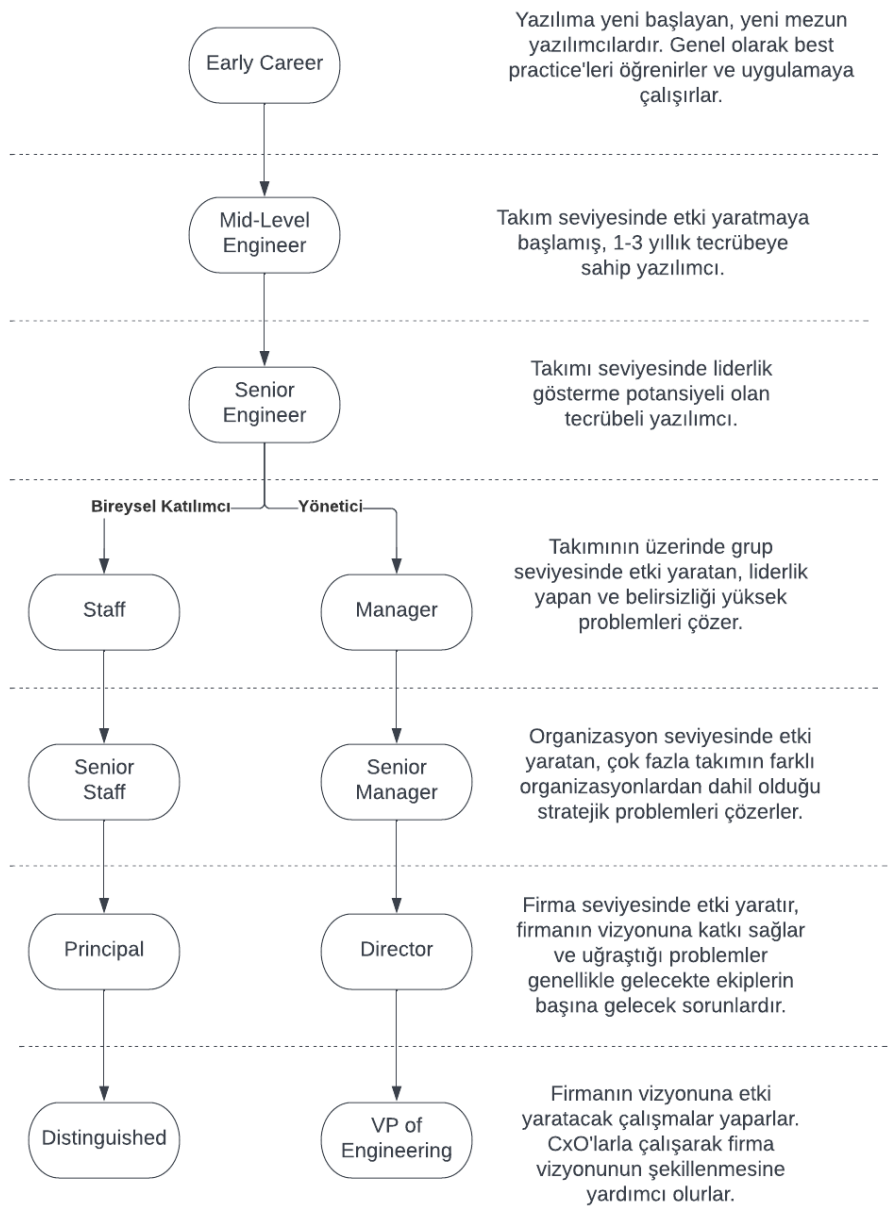
Son 4 yıldır çalıştığım bütün startup'larda her zaman bu büyük bir sorundu. Ne kadar çabalasak da her zaman bu rolün ne olduğuyla ilgili net bir açıklama yapamamak ekiplerdeki yazılımcıları her zaman bir bunalıma itti. Neden mi? Anlatayım.

Bir senior engineer, firmada 2 yıla yakın zaman geçirdikten sonra kariyerinde bir ilerleme beklemeye başlıyor. Yanlış anlamayın, bence çok yerinde bir beklenti bu. Çünkü bu adımdan önceki yılları düşünürsek, kariyerinin başında terfi almak çok basitken, ilerleyen yıllarda rollerden beklentiler artacağı için bunlar daha zor olmaya başlayacaktır. Örneğin üniversiteden mezun olan ve kariyerinin başındaki bir yazılımcı, bir yerde çalışmaya başladığında muhtemelen 1-2 yıl içinde terfi alıp orta

seviye bir yazılımcı olacaktır. Benzer şekilde orta seviye bir yazılımcı da eğer hızlı öğrenip uygulama fırsatını yakalarsa 2-3 yıl içerisinde senior rolüne gelecektir. Ancak bundan sonrası biraz daha engebeli. Çünkü bundan sonra senior engineer'ın önünde genelde iki yol olduğundan bahsetmiştim.

1. Bireysel Katılımcı Yolu (Individual Contributor)
2. Yönetici Yolu (Manager)

Genelde senior engineer'den sonraki adım IC yolunda ya principal ya da staff engineer oluyor. Bu durumda da haliyle yazılımcıların hedefi o role geçmek haline geliyor.



Burada iki farklı grubun da hatalarına çok fazla tanık oldum.

1. Görevleri ya da günlük sorumluluklarını çok iyi yaptıklarında hemen bir sonraki seviyeye geçmeleri gerektiğini düşünen senior yazılımcılar.
2. IC kariyer yolunda maaş aralıklarını dar tutan ve yazılımcılara havuç olarak terfiyi gösteren yönetici ya da firmalar.

Grup 2'nin bir temsilcisi olarak açık yüreklilikle söylemeliyim ki, bu hatayı çok kere yaptım ve sonuçlarını da ağır bir şekilde yetenekli yazılımcıları kaybederek ödedim. Ancak diğer taraftan, Grup 1'in beklentilerini iyi bir şekilde yönetmek de çok önemli tabii ki. Bu yüzden yazılımcılara terfi etmenin tek motivasyon ve daha fazla para kazanma yolu olmadığını çok iyi anlatmak ve farklı şekillerde de onları ödüllendirerek kariyerlerinde ilerlemelerini sağlamak bir yönetici için gerçekten çok önemli.

### **Peki madem bu role geçiş yapmak çok zor ve zaman alıyor, o zaman nasıl ilerlemeliyim?**

Öncelikle kariyerinizde ilerleyip bir staff engineer olmanız kolay değil. Bunu kabul etmeniz lazım. Orta seviye bir yazılımcının belki 10 yıl boyunca aynı seviyede kalması kötü bir durum olsa da, senior yazılımcının 10 yıl boyunca senior olarak kalması aslında sorun değil.

Kariyerinizde ileriye gitmek istiyorsanız öncelikle konfor alanınızın dışına çıkmayı göze alıp, toplantı yönetmek, uzun bir doküman yazarak ya da beyaz tahta karşısında diğer yazılımcıları ikna etmek, başka yazılımcıların yazdığı dokümanlara yorumlar yaparak ya da toplantılarda fikirlerini sınamak gibi daha önce deneyimlemediğiniz şeyleri yapmaya da hazır olmanız lazım. Eğer bunlar benim karakterime uygun değil diye düşünüyorsanız, üzgünüm ama bilim tam aksini söylüyor. New Scientist'te<sup>7</sup> yayımlanan Miriam Frankel'in makalesine göre sanılanın aksine içe dönük kişilerin dışa dönük birine bilinçli bir şekilde dönüşmeleri bile mümkün. Makalede yapılan araştırmalara göre bunun için alışkanlıklarınızı gözden geçirip, bunları ufak iterasyonlarla değiştirmenizin mümkün olduğundan bahsediyor.

Benim tecrübeme göre bunu başarmak bakış açınızı ve düşünme şeklinizi değiştirmekten geçiyor. Kariyerimde bir üst seviyeye çıkmayı hedeflediğimde her zaman "Şu anda bir sonraki rolde olsaydım nasıl davranırdım?" diye düşünmeye çalışıyorum ve vereceğim tepkiyi ona uyarlayıp kendimi bir nevi kalibre etmeye çalışıyorum. Ancak tabii ki herkesin farklı yetenekleri, güçlü yanları ve zayıf yanları var. O yüzden sizin maceranız kuvvetle muhtemel size özgü olacak ve size uygun bir

---

<sup>7</sup> <https://www.sciencedirect.com/science/article/abs/pii/S0262407922000641>

zaman alacaktır. Burada odağınızı iyi belirlemeniz gerekli. Bu yüzden kendinize aşağıdaki soruları sorarak başlayabilirsiniz;

1. **Teknik yetkinlikler;** bir problemi çözerken bir sistem ya da metodoloji izliyor musunuz? Bir sorun için birden fazla çözüm görebiliyor ve bunlardan birini seçtiğinizde sebeplerini anlatarak başkalarını ikna edebiliyor musunuz?
2. **Etki alanınız (Sphere of Influence);** Etki alanınızda hangi rollerde kişiler var? Takımınızdakilerin düşüncelerine etki yaratabiliyor musunuz? Peki diğer takımlardaki kişilere ilham verebiliyor musunuz? Peki ya organizasyon seviyesinde ne kadar sesiniz duyuluyor? Yazılımcılar karar verirken biz odada yoktuk diye yakınırırlar ya, eğer etkiniz yoksa o odaya çağrılmazsınız. Peki odaya nasıl girip orada kalacağınızı biliyor musunuz?
3. **Networking;** Network'ünüz yazılımcılardan mı oluşuyor yoksa yönetici ve ürün yöneticileri mi? Kaç defa yöneticilerinizle fikir alışverişi yapıp kararlara etki sağlayabildiniz? Sizi kimler tanıyor ve fikirlerinizi biliyorlar? Firmanın liderleri sizi tanıyor mu? Odadaki tüm oksijeni emmeden nasıl daha fazla görünür olabilirsiniz?
4. **Projeler;** En son büyük çaptaki hangi projede görev aldınız? Hangi fırsatları kaçıyorsunuz? Kaçırdığınız fırsatları sonradan da olsa kaçırdığınızın farkına varabiliyor musunuz?

Bunları kendinize sorduğunuzda kendinize karşı açık ve dürüst olmanız şart. Bu kimileri için zor gelebilir. Bir çoğu için belki de yöneticinizi ya da firma kültürünü suçluyor olabilirsiniz. Benim tavsiyem yapmayın! Eğer başka bir şeyi suçladığınızı farkederseniz, durun. Düşünün. Belki suçlamakta haklı olabilirsiniz, ama biraz zaman ayırıp sizin bir şeyi nasıl farklı yapabileceğinizi düşünmeye çalışın. Unutmayın, gerçek dünyadaki fırsatlar eşit bir şekilde dağıtılmaz.

Bu sorulara dürüst bir şekilde cevap verdiğinizde ikna olduktan sonra gideceğiniz en iyi kişiler yöneticiniz ya da mentorunuz olacaktır. İşte bu noktada kendinize bir sponsor bulmanızın etkisi çok fazla olacaktır. Sponsorunuzu bulabildikten sonra onunla dürüst bir şekilde ne başarmak istediğinizi ve eksiklerinizi açıkça paylaşmalısınız. Emin olun eksiklerinizi kabul ettiğinizi göstermeniz ve bunun için sizden tecrübeli birinden yardım istemeniz kadar değerli bir şey yok.

Bundan sonra kendinize bir Kişisel Gelişim Planı hazırlayın. Burada kısa ve uzun vade için hedefler belirleyin. Neleri öğrenmek ve nasıl geliştirmek istediğinizi belirledikten sonra bunlardan hangileri bilgi eksikliği yüzünden, hangisi tecrübe eksikliği yüzünden anlamaya çalışıp planınıza elle tutulur aksiyonlar ekleyin.

## Yazılımcılar için yöneticilik yolu

Yazılımcılar için kariyerlerine devam ederken önlerindeki opsiyonlardan biri de işin mutfağından ayrılmadan devam edecekleri yöneticilik seçeneği. Bu role farklı isimler verilebiliyor. Örneğin takım lideri (team lead) ya da yazılım yöneticiliği (engineering manager) olabilirken bazı firmalarda teknik lider (tech lead) rolü de bu role gibi sorumluluklar alabiliyor. Benim burada ele alacağım rol, yöneticilik rolü. Ancak burada bahsedeceklerimi az önce adını andığım diğer roller için de uygulamanız mümkün. Ancak yazı boyunca bir takımın sorumluluğunun sizde olduğunu varsayıp buna bağlı bir takım örnekler ve yöntemlerden bahsedeceğim.

Diğer taraftan bu bölümde bahsedeceğim koçluk, mentorluk, 1:1 görüşmeler gibi bölümler de iyi bir teknik lider olabilmeniz ve ekibinizle iyi bir ilişki kurabilmeniz için önemli yetkinlikler kazanmanıza yardımcı olabilecektir. Yani eğer yöneticilik yolunu seçmeyip, bireysel katılımcı olarak yolunuza devam edecekseniz temelde benzer iletişim ve liderlik yetkinliklerine de sahip olmanız gerekmektedir.

Gelin öncelikle bir ekibin yöneticisinin sorumluluğunu biraz daha iyi anlamaya çalışalım.

Diyelim ki teknik yetkinliği olmayan ama güzel bir fikri olan ve piyasa fırsatı gören bir girişimcisiniz. Teknoloji alanında bir iş yapacaksanız kendinize teknolojiyi oluşturabilecek bir teknik kişiye ihtiyacınız olacaktır. Bu kişinin temel sorumluluğu, kurduğunuz girişimin ulaşmak istediği hedefe ulaşması için gerekli teknolojiyi geliştirmek olacaktır. Bunu yaparken ürettiği teknoloji ne olursa olsun, günün sonunda nihai hedefi, organizasyona para kazandırmak olacaktır. İlk başlarda birçok sorunu kendi başına çözebilse de, firmanın büyüme hedefleri karşısında kendisini ölçeklemek için ekibine daha fazla teknoloji uzmanı alacak, farklı roller ve sorumluluk alanları oluşturacaktır. Aslında günün sonunda belli bir boyuta ulaştığında, ekibin doğru hedeflere ilerlediğinden emin olmak için belli kontrol mekanizmaları oluşturacak ve günün sonunda muhtemelen ekibine yönetici katmanı oluşturacaktır. Bu durumda bu yönetici katmanından beklentiler, fikrin ilk sahibi olan kişinin kendisinden beklentileri ile aynı olması kaçınılmazdır. Yani, firmanın ulaşmak istediği hedefe ulaşması için gerekli teknolojileri geliştirmesi ve firmaya değer üretmesi.

Bu durumda yönetici, takımın mutlu kalması için çalışacak, ancak aynı zamanda firmanın çıkarlarını korumak için gerekli politika ve stratejileri çalıştırmaktan da sorumlu olacaktır. Daniel Pink, Drive: The Surprising Truth About What Motivates Us<sup>8</sup> kitabında ekiplerin motivasyonunun dışarıdan gelen, fazla maaş, işsiz kalma korkusu, fazla bonus gibi havuç diye nitelendirilebilecek dış motivasyon öğelerinden ziyade

---

<sup>8</sup> <https://amzn.to/3LkQ3v>

bilinçsel olarak kişilerin kendi içinden ürettikleri iç motivasyonlarla daha sürdürülebilir olduğundan bahsediyor. Bu iç motivasyon öğelerini de 3'e ayırıyor;

1. Güçlü bir amaç
2. Otonomi, yani kendi kaderlerini kendilerinin belirleyebilmeleri
3. Uzmanlık alanlarına fayda sağladıklarını bilmeleri

Bu üç temel unsuru ekiplerde başarılı bir şekilde uygulayan yöneticiler de başarılı bir ekip ve sağlam bir ekip kültürü oluşturacaktır. Her ne kadar iç motivasyonun sağlanması bu üç temel unsura indirgenebiliyor olsa da bunları sağlamak için yazılım yöneticisinin işi o kadar da kolay olmayacaktır. Bunları sağlayabilmek için, firmadaki üst düzey liderlerle doğru bir şekilde anlaşması ve çoğu zaman onları yönetebilmesi (managing up) gerekirken aynı zamanda başarılı bir ürün çıkarabilmek için ürün yöneticileriyle ve iş birimleriyle de çok iyi bir ilişki kurması önemlidir.

Diğer taraftan otonomiye sağlayabilmesi için, ekibi tamamen yalnız bırakması değil doğru kontrol mekanizmalarını kurması ve ekibin doğru kararlar verdiğinden onları doğru zamanlarda sinayarak emin olması gerekmektedir. Ne yazık ki tecrübesiz yöneticiler en çok bu durumu yanlış anlarlar. Bunun sonucunda ya her işin onların denetiminden ya da onayından geçmesi gerektiğini düşünerek aşırı kontrolcü olurlar ya da ekibi tamamen kendi hallerine bırakıp ihmalkar yönetim biçimini seçerler. Bu gibi durumlarda ya ekip aşırı ilgiden sıkılır ve iş yapamaz hale gelir ya da asla bitmeyen projelerle yanlış çıktılar üreten bir ekibe dönüşür.

İşte bu yüzden bence yöneticilik kariyerine geçiş yapacak olan yazılımcıların iyi yönetici örnekleriyle yetişmeleri ve iyi örneklerini görmeleri çok önemlidir. Bu yüzden olsa gerek ki, Camille Fournier, *The Manager's Path*<sup>9</sup> kitabına şöyle başlar;

*Bu kitabı iyi bir yönetici olmak için okuyorsunuz, fakat iyi bir yönetici neye benzer biliyor musunuz? Hiç iyi bir yöneticiniz oldu mu? Eğer birisi sizi durdursa ve iyi bir yöneticiden ne beklemeniz lazım diye sorsa bu soruya cevap verebilir misiniz?*

Bu harika açılış cümlesine gerçekten bayılıyorum. Çünkü çoğu teknoloji çalışanı gibi yıllardır çalıştığım firmalarda hep yöneticim oldu. Fakat yöneticilerimin hangileri iyiydi? İyi olan yöneticilerimi düşündüğüm zaman, neden onların iyi olduğunu düşünüyordum? Her seferinde öncekinden daha iyi ya da daha kötü bir yönetici ile çalıştığımda buna neye göre karar veriyorum? Bu sorulara cevap verebilmek için Camille Fournier kitabında öncelikle beklentileri tanımlayarak başlıyor. Tabii ki bir yöneticinin şablonunu çıkarmak çok mümkün değil, bu birçok farklı dinamiğe bağlı. Ancak bunları çok geniş başlıklar altında toplamak mümkün. Buna geri geleceğiz.

---

<sup>9</sup> <https://amzn.to/34sfeJ4>

Kitap içerisinde hoşuma giden bir başka kısım ise kitabın yapısı. Konu başlıklarının içerisindeki “Ask the CTO” bölümleriyle tecrübeleri yöneticilerle yapılan sohbetler, “Good Manager, Bad Manager” ile ilgili konuda anlatılan iyi ve kötü örnekler ve sonunda da “Challenging Situations” bölümleri ile karşımıza gelebilecek farklı sorunların özetleri. Kitap adeta bir ders kitabı kadar düzenli olmasına rağmen hiç sıkıcı ve boğucu değil. Verdiği gerçek hayat örnekleri de çoğu zaman yönetici ya da yönetici adayı olarak başınıza gelebilecek sorunları size çok güzel yaşıyor. Çoğu zaman kendinizi o durumun içinde düşünüp “Ben nasıl yapardım” diye bakmaya başlıyorsunuz.

Eğer yöneticiliğe yeni başladıysanız, etrafınızda çok iyi yönetici örnekleri yoksa ve mentor bulmakta zorlanıyorsanız bu kitabı size kesinlikle öneririm!

Şimdi gelelim iyi bir yöneticiyi iyi yapan başlıkları incelemeye. Bu bölümde arada kitaptan alıntılar yaparak kendi fikrimi ve tecrübemi de paylaşacağım.

## Koçluk ve mentorluk

Çoğu zaman birbirine karıştırılan bu iki kavram bir yöneticide olmazsa olmaz yetkinliklerin başında geliyor. Bunun en önemlisi de yöneticinin ne zaman koçluk, ne zaman mentorluk yapacağını belirleyebilmesi ve buna göre hareket edebilmesi. Özellikle az önce bahsettiğim otonom bir ekip kurmak için doğru olanı belirlemek çok ama ÇOK önemli.

Gelin biraz daha detaylandıralım, koçluk; çalışanınıza yol göstererek kendi potansiyellerini keşfetmelerine yardım etmek ise, mentorluk da çalışanınıza bilgi ve tecrübenizi aktararak onların gelişimlerine yardım etmektir. İyi bir yönetici bu ikisinin farkını bilir, ve gerektiğinde birbiri arasında geçiş yapabilir. Benim gördüğüm iyi yöneticilerin çoğu bu dengeyi çok iyi biliyor, ve onlarla konuştuğunda ilk tepkileri koçluk ile başlamak oluyor. Bunların ayırdına varabilmek ve nerede giriş yapabileceğinizi bilmeniz için aktif bir dinleyici olmanız şart. Yani karşınızdakini dinleyip, sorular sorarak onları gerçekten anladığınızı göstermelisiniz. Aynı zamanda hem mentorluk hem de koçluk için konuşursak ilk başka destek verdiğiniz kişinin de sizden bu desteği alma konusunda istekli olması çok önemli. Dolayısıyla eğer yönetici olarak ekibinizden bir kişiyle çalışmaya başladığınız gün ona hemen koçluk ya da mentorluk yapabileceğinizi ve onun da size sadece ünvanınızdan dolayı saygı duyacağını düşünüyorsanız size kötü bir haberim var. Ne yazık ki yanılıyorsunuz. Bu yüzden iyi bir ilişki kurmak için, öncelikle çalışanınızı çok iyi anlamanız ve bunu ona ispatlamanız gerekir.

Bence bir yöneticinin elindeki en önemli araçlardan biri olan koçluğa biraz daha zaman ayırmamız gerektiğini düşündüğüm için buna biraz daha derinlemesine eğilmemiz gerekli!

## Liderler için koçluk

Koçluk dediğim zaman aklınıza yaşam koçu, hayat koçu gibi şeyler geliyor olabilir. Türkiye’de bunu yapan çok fazla profesyonel de dolandırıcı da var. Dolandırıcıları bir kenara bırakırsak ben de benzer bir koçluktan bahsediyorum. Ancak dışarıdan profesyonel bir koç olarak bir kişiye koçluk yapmakla, liderliğini yaptığınız bir ekibin koçluğunu yapmak arasında ciddi bir fark var. Hatta ve hatta dağlar denizler kadar fark var. Özellikle bir takım lideri ya da yönetici olarak bir ekibi yönetiyor ve ekibin çıktılarından siz sorumlu oluyorsanız, o zaman ekibi yönetirken ister istemez birçok olaya müdahale ediyor olabilirsiniz. Bu tarz bir yöneticilik az önce saydığım yetkinlik kazanma ve otonomi gibi ekibin iç motivasyonunu artırmaya yarayan unsurları zedeleyecektir. Eğer ekibin içindeyseniz, ekibe mentorluk yerine koçluk yapmanız, zamanla öğrenerek kendinizi geliştirmeniz gereken bir yetkinlik.

## Peki nedir bu koçluk?

İyi bir koç, kişinin kendi potansiyeline ulaşabilmesi için o kişiye yardım edendir. Koçun görevi o kişiye bir şey öğretmekten çok, o kişiye öğrenebilmesi ve kendini geliştirebilmesi için doğru şartları oluşturmaktır.

Eminim aranızdaki çok dikkatlıler, yazının başında okuduğum 4 kitaptan bahsettiğim halde 3 kitabın ismini vermemin farkına varmışlardır. Sizden de hiçbir şey kaçmıyor! Koçluğu anlatmak için şimdi vereceğim örnek, çok severek okuduğum Tim Gallwey’in “The inner game of tennis”<sup>10</sup> kitabından.

Gallwey burada “Inner game” derken bir benzetme yapıyor ve diyor ki, çok iyi bir tenis oyuncusu olabilmek ve sahada rakibinizi yenebilmek için öncelikle kendi içinizdeki rakibinizi yenerek sahaya çıkmanız gerekiyor. Yani “inner game” aslında kişinin kendi kendine koyduğu sınırlar, içinde onu eleştiren iç ses. Hatta yazar bundan bahsederken, çoğu zaman bu iç sesin rakibinden daha da acımasız ve zorlu olduğundan bahsediyor. Gallwey bu iç sestten bahsederken bunların sporcunun performansını düşüren bir “iç müdahale” olduğundan bahsediyor ve şöyle basit bir denklem tanımlıyor;

$$\text{Performans} = \text{Potansiyel} - \text{İç Müdahale}$$

---

<sup>10</sup> <https://amzn.to/3BZwlJi>



Bu iç müdahaleler kimi zaman kişiler tarafından kendi içlerinde çözülebiliyorken, kimi zaman da, hatta çoğu zaman, dışarıdan birilerinin sorduğu doğru sorular ve onların hazırladıkları egzersizleri yaparak çözülebiliyor. Yani bir tenisçi koçu, her ne kadar koçluk yaptığı sporcu kadar iyi tenis oynamasa da o sporcunun potansiyeline ulaşması için doğru zamanda doğru bir şekilde ilerlemesine yardımcı oluyor.

Bu iç müdahale konusuna başka bir bakış da beyin kimyamız ile alakalı. Çoğu zaman bizi yeni şeyler öğrenmekten alıkoyan belirsizlik korkusu çoğu zaman potansiyelimize ulaşmamıza da engel oluyor. Özellikle etrafınızda sizin gitmek istediğiniz zorlu yoldan geçmiş kimse yoksa kendinizi başarısız olma korkusuyla durduruyor olmanız çok insani bir durum. Çoğu zaman beynimiz bu belirsizliklerle baş etmemek için başarımla aramıza korku ve kaygı yoluyla engeller koyuyor. Bu yüzden olsa gerek ki çoğu kişi başarısız olmayı göze alamadığı için birçok şeyi denemeden “ben zaten yapamam ki” denemekten bile vazgeçiyor.

Sir John Whitmore, Coaching for Performance<sup>11</sup> kitabında, bunun için koçluk yapan liderin doğru bir düşünce yapısında olması gerektiğini anlatıyor. Bunu da “Coaching mindset” olarak tanımlıyor ve temelde bir koçun, koçluk yaptığı kişinin;

1. Yaptığı iş için yeterli kabiliyette olduğunu,
2. Her işin altından kalkabileceğini,
3. Çok yüksek bir potansiyelde olduğunu,

kabul ederek koçluğa başlaması gerektiğini söylüyor. Bu düşünce yapısını iyi sindirmiş bir koç böylelikle, koçluk yaptığı kişinin kendine inanarak, kendini motive ederek gerçek potansiyeline ulaşmasını sağlayabiliyor.

Whitmore aynı kitapta GROW adında bir yöntemden de bahsediliyor. Bu model tahmin edeceğimiz üzere bir kısaltma: **Goal, Reality, Options, Will.**

**Goal (Hedef):** Net bir şekilde belirlenmiş, kısa ve uzun vadeden oluşan bir hedef belirleyerek başlamalısınız.

**Reality (Gerçeklik):** Karşınızdakinin hedefini belirledikten sonra içinde bulunduğu gerçekliği anlaması o hedefi gerçekleştirebilmesi önemlidir. Nasıl kısıtları var? Hedeflerinden ne kadar uzaktalar? Neden bu hedefe ulaşamadılar?

**Options (Seçenekler):** İçinde bulunduğu durumun farkına varmasını sağladıktan sonra karşınızdakine önündeki seçenekleri keşfetmesi konusunda yardımcı olmalısınız. Hedefine ulaşması için yapması gereken ilk adım nedir? Bunu başarmak

---

<sup>11</sup> <https://amzn.to/3BXqfyt>

için başka neler yapabilir? Bu adımları yapınca sonuçları neler olabilir? (bkz: Second Order thinking<sup>12</sup>)

**Will (Niyet, İstek):** Beraber keşfedilen seçenekleri son aşamada aksiyonlara dönüştürmemiz gerekiyor. Bunu yine beraber keşfederek yapmanız gerekiyor. Bu seçenekleri hayata geçirmek için neler yapılmalı? Nasıl kaynaklara ihtiyacı var? Başkalarından neler bekliyor?

Tüm bunları yaparken ne kadar zor olursa olsun kendi fikirleriniz değil, karşınızdakinin ne düşündüğüne odaklanmanız gerekiyor. Soru sormaya ve karşınızdakini anlamaya çalışmanız sizi daha iyi bir koç yapacaktır.

Benim önerim, koçluk konusunda tecrübe kazanmak için başlangıçta bunu etrafınızdakilerle mümkün olduğunca denemeniz. Örneğin bir arkadaşınız ya da kendi ekibinizden olmasa bile bir çalışma arkadaşınızda bunu deneyebilirsiniz. Bu arada koçluk yapabilmek için illaki 1:1 görüşmeyi beklemek zorunda da değilsiniz. Zaten bu davranış biçimini alışkanlık haline getirmeye başlarsanız ancak bunun ekibinizdeki etkisini görebilirsiniz.

Şimdi de gördüğü fırsatı iyi bir koçluk fırsatına çevirebilen ve bunu ıskalayan bir lideri inceleyelim.

**Ahmet:** Geçen gün konuştuğumuz gibi denedim ama sorunu çözemedim.

**Buket:** Peki servisleri kendi ortamında kurup denedin mi?

**Ahmet:** Hayır, hemen öyle deneyeyim.

Yukarıdaki diyalogun kötü örnek olduğunu söylememe gerek bile yok ama neden kötü olduğunu incelemenin yine de önemli olduğunu düşünüyorum. Yukarıdaki örnekteki Ahmet, açıkça bir sorun yaşıyor. Belki önceden yöneticisi Buket ile bu konuyu tartışmış, ama görünüşe göre başarılı olamamışlar. Bunun sonucunda da Buket'e geri gelmiş. Buket sorunun nasıl çözüleceğine dair tecrübeli. Tecrübesini aktarmak yerine karşısındakine farklı seçenekleri denemesini söylüyor. Bunun sonucunda Ahmet belki bunları deneyecek ama başarılı olsa bile neden onu seçtiğini sormazsa Buket'ten öğrenemeyecek. Daha da kötüsü Ahmet, Buket'e bağımlı hale geliyor. Bir yönetici olarak bu durum Buket'in yoğun günlerinin daha da yoğunlaşmasına sebep oluyor. Peki Buket bunu iyi bir fırsata nasıl çevirebilirdi?

**Ahmet:** Geçen gün konuştuğumuz gibi yaptım ama yine başaramadım.

**Buket:** Hadi ya, o kadar da uğraşmıştın bu noktaya getirmek için. Peki şimdi ne yapmayı düşünüyorsun?

---

<sup>12</sup> <https://fs.blog/second-order-thinking/>

**Ahmet:** Servisleri kendi bilgisayarımda kurarsam sorunu daha iyi anlayabilirim diye düşünüyorum.

**Buket:** Güzel fikir, peki bunu yapmak için bir şeye ihtiyacın var mı?

**Ahmet:** Servislerin dokümantasyonu çok iyi değil, o yüzden bunu nasıl yaparım emin değilim. Ama belki ekip kanalında sorarsam birinden yardım alabilirim.

**Buket:** Bence çok güzel bir plan. Bazen bu tarz problem zor gibi gelebilir ama bence çözüme çok yakınsın.

**Ahmet:** Tamamdır, şimdi deneyeceğim.

Bu konuşmada Buket, Ahmet'i anlıyor, harcadığı zamanı takdir ediyor ve destek olarak sorunu çözmesine yardımcı oluyor. Bunu yaparken ona ne yapacağını söylemiyor ve sadece sorular soruyor.

Bu belki gerçekçi görünen bir senaryo gibi duruyor olabilir ama az tecrübeli bir ekiple çalışırken bu senaryo işlemeyebilir ya da "Options" kısmında daha fazla zaman harcayabilirsiniz. Zaten daha tecrübeli kişilere koçluk yapmak, daha az tecrübeli kişilere koçluk yapmaktan daha kolay olabilir. Ancak bu gibi durumlarda mentorluğun da etkisini azımsamamanız lazım.

Eğer ekibimde az tecrübeli birisi varsa onunla aramızdaki koçluk ilişkisini zedelememek için genellikle organizasyon içinde tecrübeli bir mentor bulmasına yardımcı oluyorum. Mesela ekibimde az tecrübeli bir yazılımcı işe başlayacaksa onlar işe başlamadan önce mentorlarını planlıyorum ve mentorunu onun geleceği günden önce hazırlanmaya başlıyorum. Böylece söylediğim gibi aramızdaki koçluk ilişkisini zedeledikten, ekibimdekilerin de gerekli yardımı aldıklarından emin oluyorum.

Yukarıdaki konuşma gerçekleştiikten sonra Buket'in bir sonraki 1:1 görüşmesinde bu yaşananı Ahmet ile baştan değerlendirip, neleri daha farklı yapabileceğini, neler öğrendiğini ve bir daha aynı olay başına gelirse neleri farklı yapması gerektiğini keşfetmesine yardım etmesi gerekiyor. Böylece Buket, Ahmet'in kariyerinde uzun vadeli bir etki yaratabileceği gibi ekibini daha da güçlü bir hale getirmeye devam edecektir.

Bu yazıyı okurken, "AMA benim durumum çok farklı", "AMA ekibim bunu yapmıyor", "AMA bir şey bilmiyorlar" diye düşündüğünüz yerler olabilir. Liderlik tarzınızı değiştirmek kolay değil. Özellikle artık içselleştirdiğiniz bazı "kötü" alışkanlıklara sahipseniz bu alışkanlıkları değiştirmek daha da zor olacaktır. Bunu değiştirmeyi ciddiye alıyorsanız, bence başarabilirsiniz. Unutmayın, yeni bir yöntemi öğrenirken öncelikle geleneksel yöntemi deneyimlemeniz, sonra eleştirmeniz ve sonunda da kendi tarzınızı çıkartmanız lazım.

## Geribildirim vermek

Çalıştığım en iyi yöneticilerin hepsi zamanında ve çok net şekilde geri bildirim vermeyi çok iyi biliyorlar. Örneğin bir toplantıda çalışanınız bir sunum mu yaptı ve sunum çok mu iyiydi? Toplantı sonrasında ona kesinlikle bunu söyleyin, mümkünse herkesin ortasında! Sunumu daha iyi nasıl yapabiliirdi? Bunu ona hemen söyleyin, ancak bu sefer baş başayken. İşte bunu yaptıktan sonra ona elle tutulur örnekler ve sizce daha iyi nasıl olabileceğini de anlatın. Sonrasında da koçluk yaparak onun bu yönünü geliştirmesine yardımcı olun. Örneğin ekibimdeki yeni bir yönetici sabah toplantılarını yönetirken belli bir plana uygun ilerlemiyordu ve bu da ekibin kafasının karışmasına sebep oluyordu. Toplantı bittikten sonra bu geribildirimi kendisiyle hemen paylaştım ve belki de toplantılara katılmadan önce birer cümlelik konuşma notlarıyla hazırlık yaparak gelmesinin iyi olacağını söylemiştim. Bu gibi ufak geri bildirimler eğer zamanında yapılırsa o kişinin kariyeri ve gelişimi için inanılmaz önemli bir etki yaratacaktır. Bu sizin sorumluluğunuz!

## Bire bir görüşmeler ve ekiple kurduğunuz ilişki

Türkiye’de yöneticilik kültürünün yurtdışında çalıştığım son 6 yılda değiştiğine eminim, ancak geriye bakarak kendi tecrübelerimi düşündüğümde ne yazık ki Türkiye’deki yöneticilerimle 1:1 görüşmeler yapmadığımı hatırlıyorum. Belki bir kriz olduğunda arada yöneticimle oturup konuşuyorduk, ancak benim kariyerime destek olacak fırsatı ne yazık ki yakalayamamıştık. Bu benim, tecrübeme dayanarak konuşuyorum, yöneticim ile aramdaki ilişkiyi de derinleştirmeye fırsat sağlayamadı. Aynı firmadaki ekibimde birlikte çalıştığım teknik liderle aramdaki ilişki de bir o kadar harikaydı. Ona gidip her türlü derdimi açık ve filtresiz bir şekilde söyleyebiliyordum. Üstelik başka bir iş aradığımda ondan yardım bile almıştım, bu durum da beni firmadan ayrılmak konusunda kendimi daha da kötü hissettiriyordu hatta bir kaç defa ayrılmama kararı vermeme sebep olmuştu.

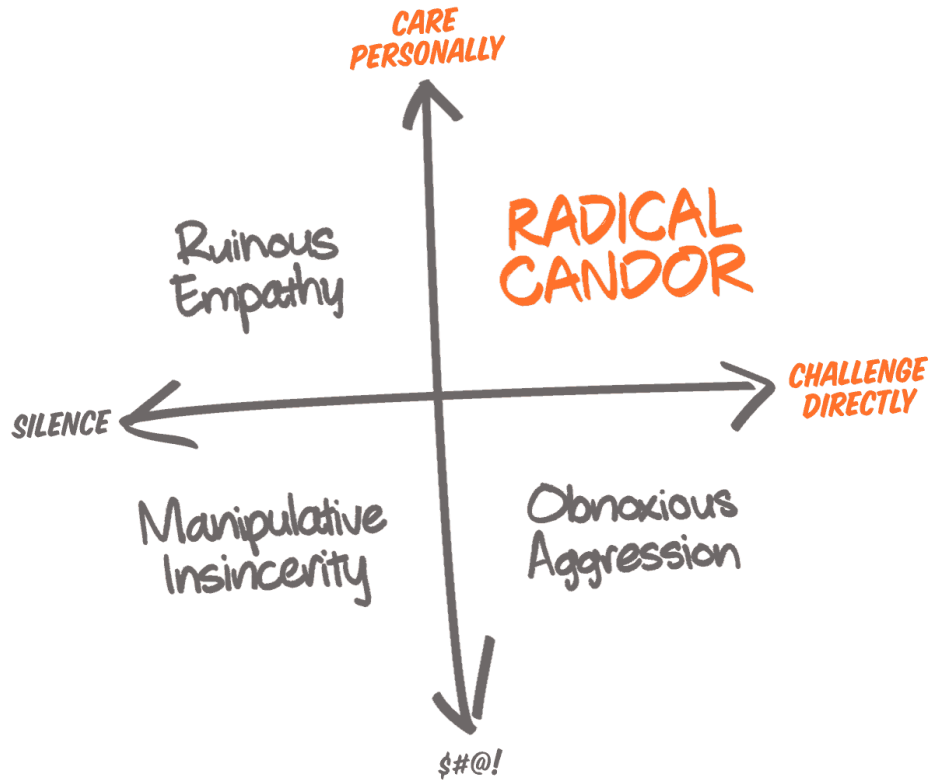
Konuyu daha fazla uzatmadan sadede geleyim, yani demek istediğim, bence, çok iyi bir 1:1 görüşmenin asıl sırrı ekibinizdekilerle nasıl bir ilişki kurduğunuz. Bunun için Silikon Vadisi’nin en kutsal kitaplarından biri kabul edilen Radical Candor<sup>13</sup> kitabını ve bana öğrettilerinden bahsedeceğim.

“Candor” kelimesi Türkçe “samimiyet, içtenlik” anlamına geliyor. Kitap bize net bir şekilde içindekileri söylemenin, yani samimi olabilmenin, yolunun karşındakini umursadığını, değer verdiğini göstermekten geçtiğini savunuyor. Bunu yaparken de basit bir anlatımla 4 farklı iletişim tipine odaklanıyor.

---

<sup>13</sup> <https://amzn.to/3rUe3v8>

Kim Scott kitabındaki bu 4 farklı iletişim tipini iki eksende inceliyor. Yatay eksenle direkt olarak karşındakini sına, dikey eksenle ise önemseme değeri bulunuyor.



Yukarıdaki grafiği anlamaya çalışırsak aşağıdaki gibi yorumlanabilir.

- Karşınızdakine değeri veriyor ama onları direkt olarak sınıyorlarsanız "Ruinous Empathy", yani "zarar verici empati",
- Hiç önemsemiyor ve direkt olarak sınıyorlarsanız, "Manipulative Insincerity" yani "Manipülatif içtensizlik",
- Hiç önemsemiyor ama direkt olarak onları sınıyorlarsanız "Obnoxious Aggression" yani "Çirkin Saldırgan",
- Son olarak, hem önemsiyorlarsanız hem de onları direkt bir şekilde sınıyorlarsanız "Radical Candor"

Gelin bir örnek üzerinden inceleyelim.

Diyelim ki ofisteki arkadaşlarınızla kalabalık bir öğle yemeğindesiniz, karşınızdaki iş arkadaşınız içinde ıspanak olan bir yemeği söyledi. Yemeği yerken iş arkadaşınızın dışında ıspanak olduğunu fark ettiniz. Ama o kadar kocaman ki, bilinçli bir şekilde çabalamadığı sürece bundan kurtulması mümkün değil gibi duruyor. Ne yaparsınız? Radical Candor grafiğindeki davranış biçimlerine göre farklı yapılacak şeyleri yerleştirmeye çalışalım;

- **Radical Candor**; Hemen bir bahane bulur, arkadaşınızın yanına gider ve kulağına “Kusura bakma ama bir şey söylemem lazım, dışında kocaman bir ıspanak var ve çok kötü görünüyor. Bende olsa birisinin bana söylemesini isterdim. Tuvalete gidip dışını temizlemek isteyebilirsin.” diyebilirsiniz. Bunu derken karşınızdakini umursadığınızı, onun kendini kötü göstermesini istemediğinizi ve daha da önemlisi sorununu nasıl çözeceğini direkt bir şekilde söylediniz. Sizce arkadaşınız nasıl hissetti?
- **Obnoxious Aggression**; Dışındekini fark ettiğinizde masada yüksek bir sesle “ŞUNA BAKIN DIŞINDE KOCAMAN YEŞİL BİR ŞEY VAR HAAAAHA DIŞ FIRÇASI OLAN VAR MI!?” diye bağırırken arkadaşınızı gösterebilirsiniz. Arkadaşınız için harika bir deneyim olmasa da sorununu öğrenebildiği için sorunu çözüyor. Ancak aranızdaki ilişkiyi kötüye götürürken belki de o kişi için günü rezil etmiş olabilirsiniz.
- **Ruinous Empathy**; Dışındekini fark ettiğinizde kendi kendinize “Ya dışındaki kocaman şeyi söylesem mi? Ama söylersem onu üzebilirim. Neyse ya o kendisi farkına varır” diye düşünerek hiç sesinizi çıkartmadığınızı düşünün. Bu durumda sorunu çözmediğiniz gibi arkadaşınızı da dışarıdan gelecek farklı “Obnoxious Aggression” tavırlarına karşı açık bir şekilde bıraktınız. Yani bence en kötüsü bu, çünkü önemseydiğiniz birisinin sorununu çözme fırsatını kaçırmaya sebep oldunuz. Arkadaşınız bunu fark etmedi ve tüm öğleden sonra toplantılara katıldı. Akşam ofisten çıkarken bir şekilde farkına vardı ve sizce ne düşündü?
- **Manipulative Insincerity**; Dışındekini fark ettiğinizde yanınızdaki arkadaşınıza döndünüz ve onun kulağına “Şuna bak ya dışında kocaman bir ıspanak var, işte ben bu yüzden öğle yemeklerinde ne yediğime dikkat ediyorum sürekli dişimle kontrol ediyorum. Ne yani ıspanak yiyip dişimle kontrol de mi etmiyor?” dediniz.

Şimdi bu noktada geçtiğiniz haftayı düşünün. Siz iş hayatınızda hangisisiniz? Eminim çoğu zaman farklı kişiler arasında bu spektrumda farklı geçişler yapıyorsunuzdur. Sizin bu geçişleri yapmanızı ne sağlıyor? Neden bir kişiyle ilişkinizde Obnoxious Aggression olurken diğerine karşı Radical Candor’sunuz?

Bu tamamen karşılıklı ilişkilerinizle alakalı bir durum. Eğer ekibinizdeki herkesle bir şekilde “Radical Candor” olamıyorsanız bir şeyleri sorgulamanın zamanı gelmiş olabilir! Bu örnekte birçok kişinin Radical Candor davranacağını düşündüğüm için iş hayatından başka bir senaryo ile ilerleyeceğim:

Ekibinizdeki ürün yöneticiniz tüm firmanın katıldığı bir etkinlikte sizin projenizin sunumunu yapmak için gönüllü oldu. Sunum günü geldi. Tüm firma toplandınız ve arkadaşınızın sunumunu dinliyorsunuz. Sunum sırasında arkadaşınızın adeta fırtına gibi kelimeleri peşi sıra söylediğini ve normalden çok hızlı konuştuğunu fark ettiniz.

Ne yaparsınız? Ben onun bir akranı olarak ne yaptığımı söyleyeyim, belki siz de yorumlara kendi bakış açınızı eklemek istersiniz.

Sunum bittikten sonra kısa bir soluklanma zamanı bırakıp kendisine slack üzerinden ulaştım ve müsait bir zamanda ona sunum hakkında geribildirim vermek istediğimi söyledim. Hemen dönüş yaptı ve hızlı bir görüşme yaptık. Görüşme şöyle gelişti.

– *Sunumdan hemen sonra böyle seni sıkıştırmak istemezdim, hala geri bildirim almak istiyorsun değil mi? Biraz rahatlayıp dinlenmek istersen sonra da konuşabiliriz.*

– *Ne düşündüğünü merak ediyorum, duymak çok iyi olur.*

– *Peki. Bence sunumun içeriği ve slaytlar çok güzel tasarlanmıştı. Ama sunumu bir çırpıda o kadar hızlı bitirdin ki birçok kişi o slayttaki görsel ne anlatıyor tam olarak düşünme fırsatı bulamadan diğerine geçti. Bence böyle yapman özenerek hazırladığın slaytların yaratacağı etkiyi ciddi anlamda düşürdü. Projeyi çok iyi bilen birisi olarak ben bile bazen yakalamakta zorluk çektim. Belki cümle aralarında 1-2 saniyelik boşluklar verebilirdin, böylece hem kendini dinlendirirdin hem de insanların seni yakalaması ve slaytları anlaması için zaman kazandırabilirdin.*

– *Haklısın, ben de biraz onu fark ettim ama zamanı yönetemeyeceğim diye çok korktuğum için biraz telaş yaptım sanırım.*

– *Benim de başıma çok geliyor bu, istersen bir sonraki sunumdan önce seninle birlikte bir prova yaparız böylece hem heyecanı üzerinden atarsın hem de yetiştirip yetiştiremeyeceğini gerçekten deneyimlemiş olursun.*

– *Güzel fikir teşekkürler, hatta belki tüm takıma sunumu yaparım böylece onları da dahil etmiş olurum. Hem onlar da bu sunumu büyük toplantıdan önce görmüş olurlar.*

– *Harika fikir!*

Tabii bu konuşmayı hiç görüşmediğim ve ilişkim olmayan birisine karşı bu şekilde yapamazdım. Bunun gibi geri bildirimleri yüz yüze vermek için önce iyi bir ilişki kurmanız gerekiyor.

**Ekibinize onları önemseyişinizi göstermek**

Bunun için yapılacak birçok şey var, bu tamamen sizin karakterinize ve güçlü yanlarınıza kalmış. Ancak etkili 1:1 görüşmeler yapmak için biraz altyapı çalışmasına ihtiyacınız var. Temel bazı prensipleri tartışarak başlayalım, ancak unutmayın sadece

yöneticileri değil aynı zamanda ekipteki teknik liderleri de ilgilendiren prensipler bunlar.

- Ekibinizin hayatını takip edin. Onlara sadece firma için çalışan “kaynak” değil, ekibinizin başarısı için en önemli kişi gibi davranın. Eskiden “İşe gelirken kişisel hayatını ve sorunlarını bir kenara bırak” denirdi, ancak bunun mümkün olmadığını çok iyi biliyoruz. Bir lider olarak, eğer karşınızdaki anlatmak isterse dinleyin. İş ile ilgili olmasa da bu sorunun çözümü için nasıl yardımcı olabileceğinizi düşünün. Partneriyle sorunlar mı yaşıyor? Bir iki günlük izne çıkıp kafasını toplamasına yardım edebilir misiniz? Ya da evinde sorunlar mı var? Sessiz bir dinleyici olarak sorununu dinleyebilir misin? Akıl sağlığı ile ilgili sorunları mı var? Firmanızın psikoterapi masraflarını karşılaması için çaba harcayabilir misiniz?
- Uzaktan çalışıyorsanız, arada iletişim kurduğunuz araç üzerinden nasılsın her şey yolunda mı diye hal hatır sorun. Özellikle ofisten çalışmaya alışkınsanız, uzaktan çalışmaya geçtiğinizde bu tarz şeyler çok kolay atlanabiliyor. Bu yüzden fazlaca hassas olmanızın size hiçbir zararı olmaz.
- Gördüğünüz bir sorun varsa bir sonraki 1:1 görüşmesini beklemeyin. Hemen söyleyin. Çözebiliyorsanız çözün, ama uzun sürecekse 1:1 görüşmelerinize taşımayı ve derinlemesine tartışmayı önerin.
- Bir başarı sağladığında hemen herkesin önünde tebrik edin. Ama yaptığı hatalarını gizlice ona hemen söyleyin.
- İlgi alanlarını dinleyin ve ortak ilgi alanlarınızı keşfetmeye çalışın. İş dışında vakit geçirmekten çekinmeyin.
- Ekibinizle iş konuşmanın dışında vakit geçirebileceğiniz görüşmeler planlayın ve bunu günlük rutinelere ekleyin. Örneğin ofisteyseniz öğle yemeklerini beraber yemek, haftalık fika<sup>14</sup> buluşmaları ya da belli günler beraber spor yapmak gibi.
- 1:1 görüşmeleri sık ve düzenli yapın. Google’ın yaptığı araştırmaya<sup>15</sup> göre sıklıkla 1:1 yapan yöneticilerin ekiplerinin performansının arttığı görülüyor. Bence en güzel sıklık haftada en az 1 defa 30-60 dakika arasında buluşmak.
- 1:1 görüşmelerinizi sakın iptal etmeyin ve asla geç kalmayın. Eğer çok zorundaysanız başka bir güne alın. Ama yoğunluğunuzu bahane ederek sakın haftayı atlamayın.

<sup>14</sup> **Fika:** İsveççe kahve anlamına geliyor. İsveç kültüründe rahatlamak ve sohbet etmek için yapılan buluşmalara da fika deniyor. Firma içinde uygulaması da her hafta ekibin fika saatinde 30 dakikalık bir buluşma yapılıyor, iş konuşmak yasak ve her hafta bir kişi kekleri alıyor.

<sup>15</sup>

<https://rework.withgoogle.com/guides/managers-coach-managers-to-coach/steps/hold-effective-1-1-meetings/>



## 1:1 görüşmelerde ne konuşacağım?

Ben genellikle bu görüşmelerin gündemini karşı tarafın belirlemesini bekliyorum. Bunun için toplantı gündemiyle ilgili bir dosya yaratıp bunu paylaşıyorum. Her hafta, o hafta içinde yaşanan ve konuşmak istediklerini bu dokümana yazmalarını istiyorum. Eğer benim konuşmak istediğim bir şey varsa da bunu oraya ekliyorum. Böylece görüşmeden önce bakıp hazırlanma şansımız oluyor. Ancak bazen karşımdaki hazırlanmadan gelirse diye elimde hazırda beklettiğim sorular olmuyor değil. Bunları aşağıdaki gibi ikiye ayırarak sınıflandırıyorum;

### **Eğer ilk defa görüşüyorsam ya da işe yeni başladıysa;**

- Bana biraz kendinden bahseder misin?; Neden yazılımcılığı seçtin? Seni en çok ne motive etti? Yazılımcılık kariyerinden önce neler yapıyordun?
- Kişisel hayatın ve profesyonel hayatın için başarmak istediğin hırsların neler?
- Seni en çok ne demotive ediyor?
- Senin gözünde iyi bir lider/yönetici nasıl olmalı?
- Benden beklentilerin nedir? Sana en iyi şekilde nasıl yardımcı olabilirim?
- Senin için iyi bir çalışma ortamı nedir?
- (Eğer ben işe yeni başladıysam) Senin için firmada düzeltilmesi gereken en önemli sorun nedir?
- 1:1 görüşmeler seni tanımam ve yardımcı olmam için çok önemli, bu görüşmeleri en faydalı hale getirmek için sence nasıl yapmalıyız?

### **Zaten düzenli olarak 1:1 yapıyorsak;**

- Nasılsın? Haftan nasıl geçiyor?
- Bugün ne hakkında konuşmak istersin?
- Eğer bugün ayrılmak istersen, neden ayrılırdın?
- Firmanın vizyonunu/yeni stratejisini nasıl buluyorsun?
- Sence ekibin çalışmaları nasıl gidiyor? – Burada proje, ürün hakkında geribildirim alabilirsiniz.
- Bana vermek istediğin bir geri bildirim var mı?

Tabii bu paylaştığım sorular çok kesin ve net görülebilir. Ancak görüşmelerde sanki bir röportaj yapar gibi soru sormamalısınız. Yapmanız gereken doğal bir konuşma içerisindeyken karşınızdakini dinlemek. Unutmayın eğer 1:1 sırasında siz karşınızdakinden daha çok konuşuyorsanız bir şeyi yanlış yapıyor olabilirsiniz. Ya da henüz karşınızdakiyle iyi bir ilişki kuramamış olabilirsiniz.

Beni yanlış anlamayın, her 1:1 görüşme harika geçecek diye bir şey yok. Bazen sizi duygusal olarak ciddi anlamda yorabileceği gibi aynı zamanda bazen derin sessizlikler de olabilir. Gelin farklı bir kaç senaryoyu inceleyelim;

**İçini dökme görüşmesi;** Benim en sevdiğim 1:1 senaryosu. Karşınızdaki şirkete, takımdan birine, ürün yöneticisine, bir direktöre ya da size çok kızmış ve size durmadan şikayette bulunuyor. Bu biriyle gelebileceğiniz en samimi durum. Ancak bu durumu çok iyi yönetmeniz lazım. Genelde az tecrübeli liderlerin yaptığı hata burada başkasını suçlayarak durumu kurtarmaya çalışmak ya da her problemi o toplantıda çözerek bir süper kahraman olmaya çalışmak. Bunun yerine bu gibi durumlarda karşı tarafı çok iyi anladığınızdan emin olmanız ve hiçbir varsayımda bulunmamanız lazım. Eğer bir görüşmede birisi size “Ekipteki kimse yeterince çalışmıyor” diyorsa ona sorun; “Seni ne böyle düşündürüyor?”, “Daha iyi anlamam için bir örnek verebilir misin?”. Ya da bir gün çalışanınız size “Firma bizi hiç iyi anlamıyor” diyerek şikayette bulunabilir, firmanın sizi de anlamadığını düşünüyor olabilirsiniz. Ama sizin göreviniz bir anda gıybet yapmaya başlamak değil aksine karşınızdakinden elle tutulur deliller toplamak. Bu yüzden sorular sorarak daha iyi anlamaya çalışmalısınız. Bu fikrinizi beyan etmemeniz anlamına gelmiyor, ama sakın bir taraf gibi davranarak karşınızdakiyle bir birlik ya da ona karşı olmayın. Sizin göreviniz bu değil çünkü. Bu tarz görüşmeleri sevmemin sebebi, 1) karşımdakinin bana güvendiği sinyalini almaya başlamanın 2) açıkça problemleri konuştukça yardımcı olabilme fırsatını yakalamak. Yani bu görüşmeleri iyi kullanırsanız uzun vadede hem sizin, hem ekibinizin hem de firmanızın yararına büyük fırsatlar yakalayabilirsiniz. Bu yüzden taraf olarak, karşınızdakini sorun olmadığına ikna etmeye çalışarak bunu berbat etmeyin. Benim bu senaryodaki son kişisel önerim en kolay çözebileceğinizi düşündüğünüz ve kontrolü sizde olan konulara ilk başta odaklanmanız. Böylece karşınızdakinin size olan güvenini artırma konusundaki ihtimalinizi arttırabilirsiniz. Geri kalanları da, “benim kontrolümde olanlar” ve “yöneticiden yardım almam gerekenler” diye sınıflandırarak üzerinde çalışabilirsiniz.

**Durum raporu görüşmesi;** Bazı zamanlar 1:1 görüşmelerde karşınızdaki bir anda size o hafta ne yaptığını anlatmaya başlayabilir. Unutmayın, bu görüşmenin amacı karşınızdakinin ne iş yaptığını anlamak değil. Onun için stand-up’lar var, task management araçları var, github pull-request’leri var. Bu görüşmenin amacı karşınızdakine yardım etmek, geri bildirim vermek ve onların kariyerlerine katkıda bulunmak. Bu yüzden bu durumu kontrol altına almanız önemli olabilir. Bu senaryo ile karşılaşmanız her şeyin iyi gittiğine de işaret ediyor olabileceği gibi diğer taraftan da karşınızdakinin sizden ne beklediğini anlamadığı anlamına da geliyor olabilir. Bu durumu kontrol altına alırken bunu karşınızdakine güzel bir şekilde anlatmanız gerekli. Ben bunu genelde tartışmanın doğallığını koruması için dolaylı yoldan

yapmayı tercih ediyorum. Bu durumda karşılaştığınızda aklınızdaki acil durum sorularını kullanabilirsiniz. Bunu yapmak için de verdiği durum raporunu iyi dinleyip kelime aralarından sorular çıkartarak konuşmayı döndürebilirsiniz. Mesela spesifik olarak yaptığı ya da yapamadığı şeylerden teknik olarak ne kadar zorlandığını sorabilirsiniz, ya da bir başkasıyla yaşadığı bir konuşmayı anlatırken ilişki kurmakta ne kadar başarılı olup olmadığını anlamak için sorular sorabilirsiniz. Günün sonunda direkt olmasa da bu zamanı sizden geri bildirim alarak nasıl daha faydalı bir hale getirebileceğine ikna etmenizde fayda var.

**Firmadan kopuş görüşmeleri;** Bazen karşınızdakinin firmadan yavaş yavaş ayrılmayı aklına koyduğunu fark edebilirsiniz. Bu durum genelde kendini umursamazlık, heyecansızlık ya da aşırı kızgınlık olarak gösterebilir. Öncelikle bu durumu gözlemlerken biraz içgüdülerinize ve tecrübenize güvenmenizde fayda var. Eğer bunu hissederseniz, bence, bu konuşmayı bir iç dökmeye dönüştürmek iyi bir fikir olabilir. Bunun için çoğu zaman yapmanız gereken susup karşı tarafı dinlemek ve içini dökmesine izin vermek. Böylece duygusal kısmı atlatıp, mantıksal tarafla konuşmaya başlayabilirsiniz. Bunu yaparken de durumun ciddiyetine göre pozitif şeylere odaklanmayı deneyebilir, kendinize karşınızdakini ikna etmek için aksiyonlar çıkarmaya çalışabilir ve çözüm üretmeyi deneyebilirsiniz. Ama unutmayın, her zaman herkesin fikrini değiştiremezsiniz ve herkesi kurtaramazsınız. Çalışanınızın işten ayrılması dünyanın sonu değil ya da üstesinden gelemeyeceğiniz bir başarısızlık da değil. Hatta her zaman sizin hatanız da olmayabilir. Önemli olan sürekli ilişkileri güçlü ve samimi hale getirmeye çalışmak olmalı.

**Sessiz görüşmeler;** Bunlar benim en sevmediğim ve en çok zorlandığım görüşmeler. Bu görüşmelerde karşınızdaki sizinle ne konuşacağını bilmiyor olabilir, sizi umursamıyor olabilir, sizinle bir şey konuşmak istemiyor olabilir. Sizin durumu kontrol altına alarak ona nasıl yardım edebileceğinizi keşfetmeniz gerekli. Bu gibi durumlarla karşılaştığımda yaptığım şey genellikle ortak bir payda bulmaya çalışmak oluyor. Küresel ısınma, enerji krizi, spor, yazılımcılık, filmler, diziler her şey olabilir. Bunu anlamak biraz sizin kişisel iletişim becerilerinizle de alakalı. Doğrusu yöneticilik yapmaya başlamadan önce asla havadan sudan konuşmayı beceremezdim. Ancak İstanbul taksicileri ve berberleri üzerinde uzun süre çalışarak bu konuda kendimi geliştirmeyi başardım. Herkese tavsiye ederim! Ehem, konuya dönelim. Sessiz görüşmelerle ilk defa karşılaştıktan sonra 1:1'lerde bir süre iş hakkında konuşmamaya hazırlayın kendinizi, karşınızdaki mental olarak hazır olmaya başladığında sizinle bir şeyler paylaşmaya başlayacaktır. Eğer bu durum 3-4 görüşmeden uzun sürüyorsa taktik değiştirmeyi düşünebilirsiniz. Fakat karşınızdakinin performansını iyi görüyorsanız, topladığınız geri bildirimler de iyiye belki de bu görüşmelerin böyle ilerlemesi daha sağlıklı bile olabilir.

Bu bölümde anlatmak istediklerimi iki başlıkta toplayabiliriz, umarım bunları aktarmakta başarılı olmuşumdur. Özetlemem gerekirse,

1. Her zaman Radical Candor olmaya çalışın. Bunun için çalışma arkadaşlarınızı umursadığınızı onlara gösterin.
2. Ekibinizi dinleyin. Dinleyin. Dinleyin. Soru sormaktan çok siz bir şeyler anlatıyorsanız bu sözümü hatırlayın. Belki gerçekten anlatmanız gerekiyor olabilir, fakat bir öz kontrol yapmakta fayda da olabilir.

## Peki nasıl yöneticiliğe geçebilirim?

Eğer bir yazılımcı olarak kariyerinize yönetici olarak devam etmek istiyorsanız benim önerim bunu öncelikle kendi yöneticinizle tartışmanız ve kendiniz hakkında geribildirim almanız. Bu yazıda bahsettiğim konuları düşünerek bunlardan hangilerini rahat yapabilirsiniz, hangilerini yapmak için tecrübe kazanabilirsiniz bunu düşünün. Kendinize firma içinden, ya da firma dışından bir mentor bulun. Ancak mentorla buluştuğunuzda onu dinleyin, her dediğine karşı çıkıp bu böyle olmaz demeyin. Anlamaya çalışın ve gerçek örneklerden bahsedip onun bunu nasıl çözeceğini dinleyin.

Bol bol okuyun ve öğrenmeye çalışın. Özellikle blog’umda (<https://mertsusur.com>) bu konuyla ilgili Türkçe içerik eksikliğini gidermek için düzenli yazılar üretmeye çalışıyorum! Eğer yabancı dilde yayın dinlemek ya da okumak sizin için sorun olmayacaksa Managing Up Podcast<sup>16</sup>, Will Larson’un blog<sup>17</sup> ve Lena Reinhard’ın<sup>18</sup> konuşmasına göz atın.

Yöneticiliğin birden fazla çeşidi var, ve bu çalışacağınız ekipten tutun da çalıştığınız firmaya göre ciddi değişiklikler gösteriyor. Örneğin bir firmada yönetici sadece “People Manager” sorumlulukları alırken, farklı bir firmada kod yazmayı gerektirebilir. Hatta bu firmanın hangi aşamada olduğuna bile bağlı olabilir! Mesela erken dönem bir startup’ta yönetici olarak işe başladığınızda sorumluluğunuz çoğunlukla ekibi koordine edip teknik liderlik yapmak hatta belli bir zamanınızı kod yazarak geçirmenize kadar gidebilir. Ancak firma büyümeye başladığında, özellikle bir scale-up hatta hyper-growth yapan bir scale-up ise o zaman yöneticiden beklenti organizasyon ve süreç tasarımı dönüşürken bol bol yangın söndürmekle uğraşabilirsiniz. Firma daha da büyüyüp, halka arz geçirmişse ya da artık süreçler oturmuşsa bu durumda daha da stratejik bir beklenti içinde olabilirler.

Genellikle Avrupa’daki birçok firma bu yüzden yöneticilerini seçerken onların öncelikle Senior Engineer seviyesinde olacak kadar teknik yetkinliğe sahip olmasını bekliyorlar. Sonrasında da temel yöneticilik yetkinliklerine sahip olup olmadığına ve neleri

<sup>16</sup> <https://www.managingup.show/>

<sup>17</sup> <https://lethain.com/>

<sup>18</sup> [https://www.youtube.com/watch?v=Q\\_bJVokYLRl](https://www.youtube.com/watch?v=Q_bJVokYLRl)

tecrübe ettiklerine odaklanırlar. Özellikle Google, Facebook ya da Twitter gibi firmalarda yöneticilik rolleri için görüşürseniz sizi aynı yazılımcılarda olduğu gibi kod yazdığınız ve sistem tasarımı yaptığınız zorlu bir mülakat bekleyecektir.

## Kapanış

Umarım bu kitabı okumaya başladığınızda okuduklarınız size biraz da olsa ilham vermiş, hatta umarım kafanızı biraz daha karıştırmıştır. Unutmayın, yazılım teknolojileri gelişmeye ve dolayısıyla kariyer yolları da bu teknoloji ile birlikte evrimleşmeye devam ediyor. Bu yüzden yazılımcı topluluklarıyla buluşmanızı, twitter ve diğer sosyal medya mecralarındaki sesli ya da görüntülü sohbetlere katılmanızı ve en önemlisi ihtiyacınız olduğunu düşünüyorsanız mentorluk arayışına başlamanızı kesinlikle öneririm. Tecrübeli bir yazılımcı ya da sizin geçtiğiniz yoldan geçerek yönetici olmuş tecrübeli bir yöneticinin eminim kariyerinize faydası olacaktır.

Bana aşağıdaki bağlantılar üzerinden ulaşabilirsiniz:

Haftalık yazılarım için: <https://www.mertsusur.com/>

Mentorluk ve Koçluk için: <https://superpeer.com/msusur>

Linkedin: <https://www.linkedin.com/in/msusur>

Twitter: <https://twitter.com/mertsusur>

# Kaynaklar

1

System Design Interview – An insider's guide



2

Youtube - How We've Scaled Dropbox



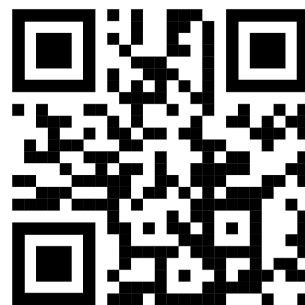
3

Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems



4

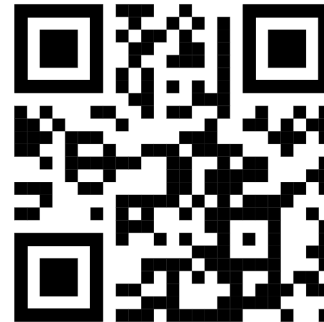
Cracking the Coding Interview, 6th Edition: 189 Programming Questions and Solutions



5 STAR Method



6 Staff Engineer: Leadership beyond the management track



7 New Scientist: How to hack your personality



8 Drive: The Surprising Truth About What Motivates Us



9

The Manager's Path: A Guide for Tech Leaders Navigating Growth and Change



10

The Inner Game of Tennis: The Classic Guide to the Mental Side of Peak Performance



11

Coaching for Performance, 5th Edition: The Principles and Practice of Coaching and Leadership



12

Second-Order Thinking: What Smart People Use to Outperform





13

Radical Candor: How to Get What You Want  
by Saying What You Mean



15

Google reWork: Hold effective 1:1 meetings



16

Managing Up



17

Blog: Will Larson



18

Youtube: What Engineering Managers  
Should Do (and Why We Don't)

