Praksis 48 | Sayfa: *105-131*

Yazılım: Gerçek Boyunduruk, Genel Zekâ ve Göreli Artı Değer Arasında Bir Kavşak

Ahmet Gire[1]

Öz

Bu makalede yazılımın üretim ilişkilerindeki yeri incelenecektir. Bu soruşturmayı yaparken de öncelikle Marx'ın kavramlarına başvurulacaktır. Öncelikle, yazılım emeğinin son ürünü olan bilgisayar programlarının üretime ve tüketime nasıl dahil olduğunu göreceğiz, sonrasında ise bu ürünün üretilmesinde işleyen emek süreçlerine değineceğiz. Böylelikle yazıda hem yazılımın genel üretim ilişkilerine nasıl etki ettiğinin incelenmesi hem yazılım sektörü içinde üretim ilişkilerinin nasıl kurulduğunun görülmesi amaçlanmaktadır.

Bahsi geçen tartışma yapılırken Marx'ın genel zekâ kavramına sık sık atıfta bulunulacaktır. Yazılım üretiminin diğer sektörlere yaptığı etkinin anlaşılmasında ve bilgisayar programı üretilirken yazılımcının içinde bulunduğu ağın yazılım üretimine yaptığı etkinin anlaşılmasında genel zekânın önemli bir kavramsal dayanak olması beklenmektedir.

Diğer yandan gayri maddi emek kavramının sınırlılıkları da bu yazının tartışma konularından birisi olacaktır. Bu kavramla temel sorulara verilen yanıtlar, ofisteki üretim ilişkileri kapsamında sınanmaya çalışılmaktadır. Yazıda yapay zekâ ve Sanayi 4.0 tartışmalarının da beslediği yazılımın büyüsü, elden geldiğinde bozulmaya çalışılacaktır.

Anahtar kelimeler: *Yazılım, Marx, Genel Zekâ, Artı Değer, Gayri Maddi Emek, Agile.*

Software: A Crossroad Between Real Subsumption, General Intellect and Relative Surplus Value

Abstract

In this article, the current situation of production relations in software will be examined. In order to investigate this relations, the concepts of Marx will be discussed. First of all, it will be observed how computer programs, the final product in the field of software, are involved in the process of production and consumption, and secondly, the labor processes that operate the production of this product will be adverted. Thus, to examine how software affects general production relations, and to analyze how production relations are established within the software sector are aimed, finally.

General intelligence concept which belongs to Marx's approaches will often be used for this debate. It is expected that general intelligence will be an important conceptual basis in understanding the effect of software production on the other sectors, and the effect that the flow which is software developers are included.

On the other hand, the limitations of the concept of "immaterial labor" will also be one of the discussion topics of this article. The answers which is given by using immaterial labor will be tested with relations of production on the office.

In conclusion, this article aims to demystify the software which is blurred by artificial intelligence and Industry 4.0 discussions.

Keywords: *Software, Marx, General Intellect, Surplus Value, Immaterial Labor, Agile.*

---

1 Yıldız Teknik Üniversitesi Doktora Öğrencisi, Yazılımcı, E-posta: ahmetgire@gmail.com
Makale Gönderim ve Kabul Tarihi:11.06.2018-03.12.2018
ORCID Numarası: 0000-0003-4506-0165

Giriş

Bu çalışmada yazılımın hem genel üretime etkisi hem de kendi üretiminin nasıl gerçekleştiği Marx'ın kavramsal çerçevesine dayanarak incelenecektir. Bu nedenle özellikle 1. bölümde Marx'ın kavramlarının nasıl anlaşıldığı kısaca açıklanmaya çalışılacaktır.

Ardından yazılımın genel üretim ilişkileri içindeki yeri masaya yatırılacaktır. Bu çerçevede, yazılımın da içinde bulunduğu bilgisayar teknolojisinin tarihine hem de bu teknolojinin mevcut üretim ilişkilerindeki etkilerine odaklanılacaktır. Burada da kapitalist üretim tarzında toplumun kümülatif bilgi birikiminin sermayeyle nasıl ilişkilendiği ortaya konulacaktır. Böylece yazılım sektörünün sermayeyle ilişkisi, genel zekâ ve sermaye ilişkisinin bir biçimi olarak anlaşılmış olacaktır.

Öte yandan çalışmanın asıl odaklandığı nokta, genel zekâ ve sermaye arasında bir dolayım olarak anladığımız yazılımın üretiminin nasıl yapıldığıdır. Bu çerçevede özellikle yazılım projelerinin üretim sürecinde kullanılan ve günümüzde oldukça yaygınlaşmış olan *Agile* metodu üzerinde durulacaktır.

Son olarak da yazılım sektörünün de dahil edildiği gayri maddi emek kuramının yazılımcı emeği ile olan ilişkisi ele alınacaktır. Böylece, yazılım üretimine özgü yanlar aracılığıyla gayri maddi emek kuramının bilişsel emeği anlama biçimi tartışmaya açılacaktır.

1. Üretim Araçları ve Üretim İlişkileri: Biçimsel Boyunduruktan Gerçek Boyunduruğa

Kapitalist üretim tarzında yapılan üretimin karakteri toplumsal gereksinimlerin karşılanması amacından tamamen bağımsızdır. Sermaye ve işçiler üretim ilişkilerinin ana sınıflarını oluştururlar. Sermaye, işçilerin emeğinin ürünlerini temellük ederek, kendi yeniden üretim koşullarını başkalarının sırtına yıkar. Burada değinmeyeceğimiz tarihsel sürecin bir sonucu, ücretli emek ilişkisinin ortaya çıkışıdır. Sermayenin üretim araçlarını kendi özel mülkiyeti kılabilmesi süreci –ilkel birikim süreci- sonunda toplumun büyük çoğunluğunun elinde sadece satabilecekleri emek güçleri kalmıştır. Glassman'ın da vurguladığı gibi (2017: 80) ilkel birikim ilk olarak doğrudan üreticilerin üretim araçlarından koparılmasıdır. Emek güçlerini satabilenler işçi sınıfını oluştururken, bu emeği satın alanlar sermaye sınıfını oluşturur. Sermaye sınıfı, bu emeği satın alarak, işçilerin –varsa işçi ailesinin- yeniden üretimi için gereken zorunlu emeğin karşılığı olan değeri işçiye ücret olarak öder, emeğin ürettiği artığı da kendi cebine indirir. Bu artık, artı değer olarak kavramsallaştırılır. Kapitalist üretim tarzında üretimin ana amacı kâr üretmek, yani artı değer üretmektir.

Marx kapitalist üretim tarzında artık değerin iki biçimi olduğunu gösterir. Bunlardan ilki mutlak artık değerdir. Mutlak artık değer, işçilerin çalışma gününün uzunluğu ile ilgilidir. Marx (2011: 486) mutlak artık değeri şöyle tarifler: "İş

gününün işçinin tam kendi emek gücünün değerine eşit bir değeri ürettiği noktanın ötesine uzatılması ve bu artık emeğe sermaye tarafından el konması: mutlak artık değer üretimi denilen şey budur". Kapitalist sistemde iş günü daha baştan iki kısma bölünmüştür; zorunlu emek ve artık emek. İşçilerin sekiz saatlik çalışma gününün üç saatinde kendi ihtiyaçlarını karşılayacak üretimi yaptıklarını varsayarsak geri kalan beş saatlik emek zaman da sermayenin el koyacağı artı değerdir. Eğer günlük çalışma süresi dokuz saate çıkarılırsa işçiler 6 saatlik süreyi sermaye için çalışıyor olurlar. İşçilerin zorunlu emek miktarları tarihsel koşullara göre belirlenir. Kapitalizm, bu zorunlu emeğin fazlasını işçilerden sızdırarak var olabilir. Yani, kapitalist tarzda zenginlik varsa bu zenginlik zorunlu olarak, işçilerin kendi gereksinimlerinden fazla çalışmalarından kaynaklanır.

Diğer taraftan, gerçekte işçinin kapitaliste kazandırdığı artık, süreç boyunca artmıştır. Mutlak artı değerde işin yoğunluğu sabit kabul edilmiş, iş günü sınırsızca uzatılabilirmiş gibi düşünülmüştü, tarihsel süreç, iş yoğunluğunun da artabilir olduğunu göstermektedir. Böylece, işçi kendi zorunlu ihtiyaçlarının karşılanması için gereken işi daha kısa sürede bitirirken, artan zamanı, eskisinden daha yoğun bir tempoyla kapitalist için çalışabilir olmuştur. Böylece göreli artık değerin artmış olduğu gözlemlenir. Marx'a göre (2014: 308) göreli artı değer:

> İş gününün uzatılması yoluyla elde edilen artık değere mutlak artık değer adını veriyorum; buna karşılık, gerekli emek-zamanın kısaltılmasından ve bunun sonucu olarak iş gününün iki kısmının büyüklükleri arasındaki oranın değişiminden kaynaklanan artık değere, göreli artık değer diyorum.

İşçi ne kadar çok üretirse kapitaliste daha çok kazandırır aynı zamanda da arttırdığı üretim kadar emek gücünün değerini de düşürür. Artık, emek süreci sonunda ortaya çıkan her üründe daha az işçi emeği içerilmiştir.

Hem makinenin işçinin karşısına bağımsız bir canavar gibi çıkması hem toplumsal zekânın ürettiği bilimsel bilginin makine aracılığıyla kapitalistin hizmetinde üretim ilişkilerine dahil olması, işçinin kapitaliste olan bağımlılığını biçimsel bağımlılıktan gerçek bağımlılığa dönüşür. Marx'ın sözleriyle (2011: 487):

> Demek oluyor ki, göreli artık değer üretimi, kendine özgü yöntemleri, araçları ve koşullarıyla birlikte, ancak işçinin sermayeye biçimsel olarak bağımlı hale gelmesiyle sağlanmış bir temel üzerinde kendiliğinden ortaya çıkan ve gelişen özgül bir üretim tarzını, yani kapitalist üretim tarzını gerektirir. İşçinin sermayeye biçimsel bağımlılığı, zamanla yerini gerçek bağımlılığa bırakır.

Böylece, Narin'in de vurguladığı gibi (2017: 549) emek makineye bağımlılaşmıştır ve:

> Emeğin sermaye tarafından gerçek boyunduruk altına alınmasıyla emek süreci tümüyle değerlenmenin kontrolüne girer. Bu yüzden değerlenme süreci, eski zanaatkâr üretimindeki niteliklerden farklı olarak, örneğin işçinin özerkliğini, karar süreçlerini elinden alır, çünkü artı değeri genişletmek amaçtır ve bu, emek sürecinin dışında bir zor değil, aksine onda nesnel olarak içerilmiş bir kontroldür.

Makineleri var eden şey, toplumun genel üretkenliğiyle ilişkili olan genel zekâsıdır. Marx'a göre (2014: 549) makine, emek araçlarının tarihselliğinde kapitalist üretim aracının somutlanmış halidir:

> Emek aracının gelişerek makineye dönüşmesi sermaye için bir rastlantı değil, devraldığı geleneksel emek araçlarını sermayeye uyarlı biçime sokan tarihi biçimlendirme sürecidir. Toplumun beynindeki bilgi ve yetenek birikimi, genel üretici güçler birikimi, bu süreçte emeğe karşı sermaye tarafından özümsenir ve böylece sermayenin ya da özel olarak üretim sürecine üretim aracı olarak giren sabit sermayenin malı olarak gözükür.

Sermaye bu zekânın eseri olan, üretim ilişkilerine zuhur eden bilimsel bilgi için hiçbir şey ödemez. Ancak en başta makinenin içinde üretim ilişkilerine dahil olan bu bilgi göreli artı değeri arttırır ve emeğin içinde bulunduğu biçimsel boyunduruğunu gerçek boyunduruğa dönüştürür. Artık emek ancak kapitalist bir öznenin varlığı ile mümkündür, işçi kendi kendine iş göremez. Marx'a göre (2011: 349) "işçi, başlangıçta, emek gücünü sermayeye, bir metanın üretimi için gerekli maddi araçlara sahip olmadığı için satıyorsa, şimdi, onun bireysel emek gücü, sermayeye satılmadığı anda iş görmez hale gelir".

Üretim araçlarının yapısındaki bu değişim, kapitalist dönem boyunca hiç durmadı. Daha büyük makineler üretildi, kapitalistlerin ürettiklerini dünyanın bambaşka coğrafyalarında satabilmelerini sağlayan daha büyük ve kuvvetli ulaşım araçları icat edildi, haberleşme sistemleri son 10 yılda tahayyül edilemeyecek biçimde gelişti. Yazılım da bütün bu gelişmelerin içinde kendi yerini aldı. Ancak yazılımın gelişmesinden önce, yazılımı mümkün kılan donanımın, yani bilgisayarın gelişim sürecine de kısaca değinmek gerekir.

2. Bilgisayar ve Bilgisayar Programının Üretim İlişkilerindeki Yeri

Marx'ın yaşadığı dönemde yani endüstri devrimi çağında üretim araçlarının en güçlüsü makine olmuştu. Marx'ın ölümünden 63 yıl sonra ilk bilgisayar icat edildi. Bu bilgisayarlara *first generation computer* yani birinci kuşak bilgisayar denilmektedir; bunlar, "(t)ransistörün ve öteki yarıiletken bileşenlerin bulunmasından önce 1940'ların ortalarında gerçekleştirilip yalnızca çevirici dili bulunan, yüksek düzeyli programlama dillerinden yoksun, lambalı ilk kuşak bilgisayar"lardır (Sankur, 2002: 249).

ENIAC adı verilen bu bilgisayar oldukça gelişmiş bir hesap makinesi gibidir. Aynı zamanda ENIAC günümüz bilgisayarlarından oldukça farklıydı. İçinde 17.000'den fazla vakum tüpü vardı ve ağırlığı 30 tondu. 450.000 dolara yapılabilen bu bilgisayar, skuaş kortu büyüklüğündeydi (Schwartz, 2011: 462).

Sonrasında birçok koldan bilgisayar teknolojisinin gelişmesine yönelik çalışmaların başladığı görüldü. İkinci Dünya Savaşının da etkisiyle askeri alanda faydalılığı görülen bilgisayar teknolojisine yönelik araştırmalar savaş sonrasında da devam etti. Schwartz'ın sözleriyle (2011: 463):

> Princeton, New Jersey'deki İleri Araştırmalar Enstitüsü'nde dünyaca ünlü matematikçi John Von Neumann, bütçesi kısmen ABD hükümeti ve kısmen de RCA Corporation tarafından karşılanan, paralel işlem yapma özelliğine sahip, yani hesaplamaları ardışık olarak değil de eşzamanlı olarak yapabilen ilk dijital bilgisayarı yapacak ekibin başında bulunuyordu. İngiltere'de Manchester Üniversitesi'nde ise matematikçi Alan Turing depolanmış bir programa sahip ilk bilgisayarı inşa eden ekibin bir parçasıydı. MIT'de, savaş sırasında pilotlar için uçuş eğitimi cihazları ve yeni aerodinamik tasarımlar için kararlılık analizleri geliştirmek amacıyla kurulmuş olan Servomekanizma Laboratuarı genç bir Mühendis olan Jay Forrester ile, Deniz Araştırmaları Ofisi için yeni bir aygıt geliştirmesi için anlaşma yaptı… Raytheon Copraration ABD Nüfus Bürosu için elektronik bir hesap makinesi geliştirmeye başladı.

Bu gelişmelerin sıralandığı yukarıdaki paragrafta her adımda bir kamu kurumunun adının geçmesi tesadüf değildi. İlk bilgisayarın üretim amacı askeri ihtiyaçları karşılamaktı. Bilgisayarları geliştirmek oldukça pahalı işlerdi, senelerce devam eden projeleri fonlayabilecek sadece tek bir aktör vardı: Devlet. Devletler soğuk savaş döneminde askeri güçlerini sürekli arttırmakta kararlıydılar, bunun için bilgisayar teknolojisindeki herhangi bir ilerleme önemliydi. Schwartz (2011: 464) bu konuda şunları vurgulamaktadır:

> Devlet, her yeni bilgisayar kuşağının ilk modellerinin yüksek fiyatlarını en kolay ödeyebilecek ve bu kuşakların her birinin sağladığı yeni özelliklere en büyük ilgiyle yaklaşan müşteriydi. Vergi gelirlerinin büyük bir kısmı bilgisayarların geliştirilmesine ayrıldı ve satınalmaların çoğu (silahlı kuvvetler, Atom Enerjisi Komisyonu veya NASA gibi) askeri ve yarı askeri kurumlar tarafından, küçük bir kısmı ise Nüfus Bürosu gibi sivil kurumlar tarafından gerçekleştirildi.

Sonrasında bilgisayarların hem maliyetleri düşmeye hem sayıları artmaya başladı. Yarıiletken teknolojisinin gelişimi ve transistörün icadı ile bilgisayarların boyutları küçüldü; kişisel kullanım için başvurulabilecek aletler arasına girdi.

Bilgisayarın hayatın her alanına yayılabilmesi sadece fiyatının ucuzlaması ve boyutunun küçülmesiyle alakalı değildi. Bilgisayarlar diğer makinelerden farklıydı. Özellikle Alan Turing'in programla çalışan bilgisayarı diğer makinelerin sürekli tekrara dayanan tek biçimli hareketiyle sınırlanmış değildi. Programla çalışan bilgisayar potansiyel olarak bütün makineleri taklit edebilme kapasitesine sahiptir. Weizenbaum'a göre (1995:72) bilgisayar her makineyi, programlanabilme yeteneği sayesinde taklit eder, bir hesap makinesi gibi tek bir fonksiyona mecbur değildir.

Ayrıca bilgisayar diğer makineleri taklit edebilirken, bütün makinelerin konuşabileceği ortak bir lisanı da mümkün kılmıştı. 1969'da internetin öncülü ARPANET'in kurulmasıyla bilgisayarlar ortak lisanlarını kendi aralarında konuşabilmeye başladılar. Günümüzde bütün ulaşım ve nitelikli üretim araçları bir işlemciye sahiptir. Bu da bu araçların -sınırının ne olduğu değişmekle beraber- programlanabileceğini gösterir. Bilgisayar programı sayesinde bir araç daha çok fonksiyonu yerine getirebilir olmuştur.

Yarmalı'ya göre (1995: 65) bilgisayar programının iki tanımı vardır: "1. Bilgisayar

programı, 2. Bir bilgisayarca uygulanmak üzere uygun biçimde deyimlendirilmiş bir izlence, 3. Bir bilgisayarda bir problemi çözmek ya da bir bilgiyi işlemek için kullanılan yordam." Her bilgisayar, işletim sistemine ihtiyaç duyar. İşletim sistemi, bir bilgisayar programıdır. Doğrudan kullanıcı ile bilgisayarın haberleşebilmesini mümkün kılan unsurdur.

Aslında yazılımla beraber artık makinelerle sadece fiziki -mekanik- ve kimyasal dille konuşma zorunluluğu ortadan kalkmıştır, yazılım makineye insanın direktif verilebilmesi için açılan bir kapı gibidir. Yazılım, cihazların beynine yerleşir, bu beyinler bilgisayardır ve makineler arası ortak dili anlayabilme kapasitelerine sahiptirler. Böylece makinenin fonksiyonunu değiştirmek için kesici ucu delici uçla değiştirmeye gerek kalmaz, o makinenin zihnine gönderilecek kod, makinenin talimatı algılaması ve o talimata uymasını sağlar.

Bilgisayar bunu evrensel bir dille yapar. Tarihte üretilen ikinci bilgisayarın adının UNIVAC (Universal Automatic Computer) yani Evrensel Otomatik Bilgisayar olması bir tesadüf değildir. Alan Turing ve ekibinin geliştirdiği programlanabilme özelliği evrensel bir standart sağlamıştır.[2] Bu evrensel standartlar, diğer makinelerin de standartlara uygun üretilmesiyle birlikte makineler arası ortak lisanı somutlaştırmıştır.

Bilgisayarın uyarlanabilme özelliği, onu hayatın her alanına kolayca soktu. Bu alanlardan biri de endüstriydi. Endüstri üretiminin ihtiyaçlarını giderecek bilgisayarlar yapıldı, bu bilgisayarları çalıştırabilecek bilgisayar programları yazıldı. Geliştirilmesine devletin öncülük ettiği ve bu süreçte kamu tarafından fonlanan makinenin fabrikada arttırdığı verimliliğinin yarattığı kâr, sermayenin hesabına yazılmaya başlandı. Marx *Grundrisse*'de (2014: 549) şöyle yazar:

> Toplumun beynindeki bilgi ve yetenek birikimi, genel üretici güçler birikimi, bu süreçte emeğe karşı sermaye tarafından özümsenir ve böylece sermayenin ya da özel olarak üretim sürecine üretim aracı olarak giren sabit sermayenin malı olarak gözükür.

Bilgisayar hem toplumun beyninde dolaşan kümülatif bilginin ürünüdür, hem de ilk bilgisayarların tarihi de göstermektedir ki, bilgisayar doğrudan toplumun fonladığı makinedir. Dahil edildiği iş sürecinde arttırdığı kapasite sermayenin temellük ettiği artı değeri arttırır. Halbuki, sermaye artığı arttıran bilgiye ulaşılan yolda hiçbir bedel ödememiştir.

Bilgisayarın üretimi arttırması işin yapılış biçiminde neden olduğu dönüşümlerle birlikte mümkündür. Narin'in de vurguladığı gibi (2017: 361), üretim teknolojisinde hızlı bir değişim yaşanmıştır:

> İşlem, otomasyona; iletim mekanizması ise İletişim ve Ağ teknolojilerine evrildi. Bu yüzden, bilgisayar ve otomasyonun gelişmesi ile makineler arasındaki iletişim

2 Bu standartlaşmanın bir göstergesi de yazılımcıların kolaylıkla başka ülkelerde iş bulabilmesidir. Ülkelerde konuşulan farklı dillere rağmen yazılım dilleri aynıdır. Türkiye'den başka ülkelere giden beyaz yakalıların çoğu yazılımcıdır. Avrupa ve ABD gibi zengin ülkelere yazılımcıların kolayca göçebilmesinin birinci nedeni yazılımcı emeğine bu ülkelerde yoğun talebin bulunmasıysa ikinci neden de yazılım yapmanın teknik standardının evrensel olmasıdır. Bir yazılımcı herhangi bir ülkede yazılım yapabilir, yaptığı işin teknik kısmı hepsinde aynıdır.

> ve Enformasyon kuramının, ağ mekanizmasının kurulması, bilimsel üretimdeki sıçramaya eşlik eden, onun sonucu olduğu gibi onu ivmelendiren en önemli gelişmeler oldular. Böylelikle bilimin nesneleşen gücü, gerçekten de bu yeni tip otomasyon ve iletişim "makineleri"nde, yani bilgisayar, robot ve ağlarda, üretici güç haline gelmektedir.

Sanayi 4.0 tartışmalarının çıkış noktası da bu değişimin sanayi boyutudur. TÜSİAD'ın Sanayi 4.0 raporu, yazılım sektörünün kapitalist üretim için ne anlama geldiğini ve neden sürekli bu sektörün teşviklerden faydalandığını ve gündemi nasıl işgal ettiğini göstermektedir. TÜSİAD'ın gelişmekte olan bir ekonomi perspektifi olarak gördüğü Sanayi 4.0 için metinleştirdiği fikirlerinde yazılım entegrasyonlarına dair fikirleri aydınlatıcı olabilir. Örneğin otomotiv sektörü için yazılım entegrasyonu; "Önlem: Tedarikçiler yatay entegrasyon yoluyla operasyonlarını üreticilerden gelen yeni siparişlere göre düzenleyebilecekler. Sonuç: Standartlaşmış süreçler ile zamanında iş birliği yapılması sayesinde hata asgariye indirilecek (TÜSİAD, 2016: 25)." Beyaz eşya sektörü içinse; "Önlem: Daha verimli montaj hatları oluşturmak için, ERP, ürün yaşam döngüsü yönetimi ve üretim sistemlerinin entegrasyonu. Sonuç: Operasyonların ileri düzeyde planlanması, izlenmesi ve raporlandırılması (TÜSİAD, 2016: 50)." Kimya sektörüne yönelik öngörülerinde asgari fazla mesai ifadesi, yazılımın genel üretim tarzı içinde üretimin işçiden bağımsızlaşma araçlarından biri olduğunun altını çizer; standart dışı işlerin azaltılması da işçinin öznelliğinin hiçbir şekilde üründe görünmeyeceğinin: "Önlem: Muhasebe, üretim ve envanter sistemlerinin dikey veri entegrasyonu Sonuç: Küçük miktarda üretim ve çevik operasyonlar. Gerçek zamanlı izleme. Asgari fazla mesai ve standart dışı iş (TÜSİAD, 2016: 52)."

Yazılım, sanayi üretiminde belli başlı görevleri devralacaktır. Bunun ilki iş akışının tam otomatizasyonu ile işçinin makine ve bant sistemi karşısında daha da zayıflamasıdır. *Supervisory Control And Data Acquisition* kelimelerinin ilk harfleri ile oluşturulan SCADA (Merkezi Denetleme Kontrol ve Veri Toplama) ve PLC -günümüzde PLC teknolojisi bir üst seviyeye, yani IPC'lere (endüstriyel bilgisayar) taşınmaktadır- yani *Programmable Logic Controller* (programlanabilir lojik kontrolcü) gibi teknolojiler yazılımın sanayi sistemlerine uyarlanmasında sıklıkla başvurulanlardır. Kısaca Karaçor ve Keleş'e göre (2007: 1) bu teknolojilerin işlevi esnek üretimin gerçekleşebilme koşullarını mümkün kılan verileri tutarak, kaynakların planlamasını mümkün kılmaktır.

Ayrıca işçi bilgisinden getirdiği gücünü yitirmeye devam eder. Narin'in de vurguladığı gibi (2012: 42) kapitalist üretim tarzında teknik bilginin ve üretim araçlarının gelişimi işçinin bilgisinin makineye aktarılmasının üzerinden ilerlemiştir. Sermaye makinelerin ve teknik bilginin geliştirilmesi için yaptığı desteği, işçilere olan bağımlılığını azaltma hedefi ile verir. İşçilerin ürettiği ustalık bilgisi, bu teknik geliştirmelerle, sermayenin malı gibi görünmeye başlar.

Bu ustalık bilgisi, örneğin, hangi sıcaklıkta üretim sürecindeki maddeye ne yapılacağını bilmek olabilir. Artık üzerinde yazılım bulunan bir üretim bandı, daha

öncesinde kaydedilmiş üretim süreci bilgisine göre bu kararı kendi başına verir. Artık işçinin üretim sürecini gözlemesine de gerek kalmamıştır. Otomasyon sistemleri sayesinde gözetlenmesi gereken birçok üretim birimi, makinelerin gözetimine girer. Burada insan gücüne ihtiyaç kalmaz. Marx (2011: 420) telafi teorisinin eleştirisinde şöyle der: "İktisatçıların iyimserlikleri yüzünden gülünçleşen gerçek olgular şunlardır: makinelerin işlerine son verdiği işçiler iş yerlerinden çıkarılıp emek piyasasına fırlatılır ve orada kapitalistçe sömürülmek için beklemekte olan emek güçlerinin sayısını artırır."

Yazılımın sanayide daha aktif kullanılması, böylece, sanayideki canlı emek ihtiyacını azaltır. Eğer bu canlı emek kullanımının maliyeti çok çok düşük değilse sermaye yazılım sektörünün sanayide daha aktif kullanılmasını arzulayacaktır. TÜBİTAK'ın 2016'da hazırlamış olduğu *Yeni Sanayi Devrimi, Akıllı Üretim Sistemleri, Teknoloji Yol Hartitası*'nda vurgulanan hedefler, yazılım ile sanayinin olası keşişim noktalarının neler olabileceğine dair işaretler barındırır. TÜBİTAK için (2016: 10) sanayi sistemlerine yazılımın daha etkili entegrasyonu ile varılacak hedefler şunlardır:

> Her türlü ortam ve işletim sisteminde (mobil vb.) çalışabilmesi •Üretime etki eden uç noktalardaki verileri (parametreler, prosesler, makineler ve ilgili yazılımlar vb.) yönetebilmesi •Gerçek zamanlı çalışmaya imkan verecek hızda çalışması •Veri kaybı yaşanmamasının garanti altına alınması •Yüksek güvenlikte çalışabilmesi •Ürün yaşam döngüsü içerisinde ürün kalitesini izlemeye yönelik akıllı ürün ihtiyaçlarını karşılayabilmesi •Gerçek zamanlı karar destek mekanizmasına sahip olması ve yönlendirme yapabilmesi.

Yazılım entegrasyonunun bu faydalarının asıl amacının ne olduğu da yol haritasının başında tanımlanmaktadır:

> Birçok sektörde küresel marka niteliğindeki şirketlerin son yirmi yılda üretim hatlarını, işgücü maliyeti düşük olan ülkelere kaydırdığı görülmektedir. Bu durumda üretim zincirinin büyük bir bölümünün anılan ülkelere kaymasının önüne geçilmesi açısından, bilgi ve iletişim teknolojilerinin yardımıyla üretim maliyetlerinin düşürülmesi önem kazanmıştır. Sanayideki bu dönüşüm gerçekleştiğinde ülkelere önemli rekabet üstünlüğü sağlayacağı gibi, ülkemizde bu dönüşümün kaçırılması durumunda mevcut durumda ülkemizin üretimdeki lojistik avantajı ve düşük işgücü maliyetinden doğan avantajlarının geçerli olmayacağı öngörülmektedir (TÜBİTAK, 2016: 1).

Burada vurgulanan verimlilik terimini Marx'ın yukarıda vurguladığımız "göreli artık değer" kavramıyla ilişkilendirebiliriz. Marx (2011:487) değişen üretim teknolojisiyle göreli artık değeri ilişkilendirir:

> Göreli artık değer üretiminin özel bir yöntemi olarak işlev görmesi, artık yalnızca, birincisi, şimdiye kadar sermayeye sadece biçimsel olarak tabi olan sanayi kollarını ele geçirmesi, yani yayılması ölçüsünde, ikincisi, halen yerlerini almış bulunduğu sanayileri üretim yöntemlerinde değişiklikler yaparak köklü dönüşümlere uğratması ölçüsünde söz konusu olur.

Sanayi 4.0 tartışmaları, artık yapay zekânın da üretim süreçlerine dahil olmasının gerekliliğinin altını çizmektedir. TÜBİTAK raporunun birçok yerinde de üretim sürecinde yaşanan olağanüstü durumlarda makinelerin oluşturduğu ağın karar verebilecek algoritmaya da sahip olmasının elzem olduğu vurgulanmıştır. Yani Marx'ın gözlemlediği makinenin evrimi tam da Marx'ın kuramıyla öngörülebilecek biçimde yaşanmıştır. İşçiler potansiyel olarak kapitalist üretimin daha ücra köşelerine atılabilir olmuşlardır. Sermayenin en derininde yatan, üretim sürecini işçilerden tamamen bağımsız yapma hedefi ve kâbusunu bu tartışmanın satır aralarında görürüz. Marx (2014: 557) bu durumu şu şekilde ifade eder:

> Burada, belirli bir emek çeşidinin işçiden makine kılığındaki sermayeye aktarılması ve bu transfer nedeniyle işçinin emek kapasitesinin değerini kaybetmesi, dolaysız olarak görülmektedir… Eskiden canlı işçinin olan etkinlik, makinenin etkinliği haline gelmektedir.

Ancak Marx'ın vurguladığı üretim süreçlerinde sağlanan gerçek boyunduruk bile işçi sınıfının direnişini engelleyememiştir. Fabrika işçisiz çalışırsa, işçi sınıfının üretimden gelen gücü elinden alınmış olur. Ancak Marx'ın toplumsal devrimleri açıklamak için kullandığı antagonizma tekrar gün yüzüne çıkar: Üretim araçlarının üretim ilişkilerinin kabuğunu parçalaması…

Bu noktada değineceğimiz ikinci unsur ise yazılımın aynı zamanda denetim aracı olmasıdır. TÜSİAD raporunda beyaz eşya sanayisi için yazılıma biçilen görev: "Operasyonların ileri düzeyde planlanması, izlenmesi ve raporlandırılması"ydı. Bu; işçiler, makineler ve mühendislerden oluşan ham üretim sürecinin sürekli denetlenebilmesinin yazılımla mümkün kılınabileceğine dair sermaye hevesidir. Böylece yazılım *Panoptikon*'a benzer bir etki yaratır. Foucault'nun etkili bir metafor olarak kullandığı mimari distopyanın bir etkisi gözetlenip gözetlenmediğinden habersiz, kendisiyle kurduğu ilişkiye gözetlenme ihtimalini dahil eden kişinin yaratımıdır (2005: 224). Foucault, disiplin dispozitifini panoptikonla anlatır. Yazılımcının yarattığı ürün, bir gözetleme –yazılımcı buna loglama (kaydetme) der- mekanizması yaratır.[3]

Fabrika üretiminde makinelerle beraber işçinin üretimin zihinsel aşamalarından koparıldığı açıktır. Bilgisayar programıyla zihinsel süreçlerin çoğu insandan alınıp makineye verilecektir. Fabrika üretiminin otomatizasyonuyla birlikte üretim aşamasının zihinsel denetimi ve karar mekanizmaları için makine "istihdam" edilecektir.

Yazılım ve bilgisayar teknolojisi aynı zamanda hizmet sektörünü de makineleştirir. Sermayenin bu hizmetleri sağlamasında iki unsur ön plana çıkar: Birinci unsur diğer

3 Çoğu zaman bir bilgisayar programında çalışanların yaptığı bütün eylemler veritabanında kaydedilir (loglanır). Bu da bilgisayar programıyla çalışan birisinin sürekli denetlendiğini bilmesine ve buna uygun olarak davranmasına neden olur. Loglama yazılımın temel prensiplerinden biridir. Bir bilgisayar programı tasarlanırken log karakterleri de konfigürasyon dosyasının içine yazılır. Bilgi verici loglar, hata logları ve kullanıcıların eylediklerinin tutulduğu loglar ayrı ayrı istiflenir (Yuan vd, 2012).
Yazdığım loglama sisteminin attığı kayıtlar bir kişinin işten atılmasında delil olarak sunulmuştu.

sermayedarlarla girilen rekabet ilişkisidir. İnternet üzerinden sağlanan hizmetler müşterilere kolaylık sağladığı gibi, şirket için prestij unsuru da oluşturur.[4]

İkincisi müşteri ilişkilerinin sanallaştırılması, sanayi sektöründekine benzer biçimde, bu ilişkilerde istihdam edilen insanlara ihtiyaç kalmaması demektir. Bankacılık sektöründeki işten çıkarmaların bankacılık kârlarında herhangi bir düşmeye mahal vermeden sürdürülmesi, bankacılık sektöründeki dijitalleşmeyle mümkün olabilmiştir. Artık para transferi için bankaya gidip bir çalışana talimat verilmesi gerekmez. "Dijitalleşmeye odaklılık sayesinde hizmet maliyetinin azaltılması ve «gider eliminasyonunun» desteklenmesi" dijitalleşmenin bankacılık sektöründeki istihdamı düşürerek hizmet maliyetlerini azalttığı manasına gelir (Yapı Kredi Bankası, 2017: 1).

Türkiye Bankacılar Birliği'nin (TBB) hazırladığı rapor (2016: 1) da şube sayısı başına düşen personel sayısının azaldığını ancak şube başına düşen müşteri sayısının arttığını göstermektedir:

> 2005-2015 döneminde şube başına personel sayısı aşağı yönlü bir seyir izledi. 2005 yılında sektörde şube başına 21,2 olan personel sayısı, 2015 sonu itibarıyla 18'e indi. Bankacılık sektöründe şube büyümesinin hız kaybetmesinin ardından şube başına nüfus sayısı arttı. 2014 yılında bir banka şubesi ortalama 6.925 kişiye hizmet sunarken, 2015 yılında bu rakam yüzde 1,3'lük artış ile 7.012'ye yükseldi. Bir başka ifadeyle 2014 yılında 10.000 kişiye 1,44 şube düşerken, 2015 yılında bu rakam 1,43 oldu.

TBB'nin 2017 yılına dair açıkladığı raporda bankacılık alanındaki istihdamın azalmaya devam ettiği vurgulanmaktadır: "İstihdam, bir önceki yılın aynı dönemine göre 2.655 kişi (yüzde 1,3 oranında), 2016 yılsonuna göre ise 1.635 kişi azalmıştır (2017, 1)."

Banka şubelerindeki ve bu şubelerde hizmet gerçekleştiren kişi sayısındaki azalış; birinci olarak bankacılık sektöründeki dijitalleşmenin doğrudan etkisinden kaynaklanmaktadır, ikinci olarak da bu dijitalleşmenin yarattığı işsiz bankacıların sayısındaki artış bankacılara fazladan baskı kurmanın istatistiki araçlarını sunmaktadır. Böylece bankacılar daha yüksek performansla çalışmaya zorlanabilmektedir.

Bankacılık sektöründe çağrı merkezleri sıklıkla kullanılan hizmet araçlarından bir tanesidir. Çağrı merkezlerindeki telefon konuşmaları da makineye devredilmektedir. Bir müşteri bankanın çağrı merkezini aradığında ilk birkaç adımda makine ile konuşmaktadır. Böylece hizmet sektöründe yeni bir alan olarak emek gücü talebinde bulunan bu alanın makineleşmesine tanıklık etmekteyiz. Bilgisayar sesi ne kadar etkili bir biçimde algılayabilirse bu hizmetteki insan faktörüne ihtiyaç o derecede azalır. Yani teknolojik gelişmenin hizmet araçlarını arttırması üretim sektöründe gözlemlenen etkinin benzerini yaratır.

---

4 Özellikle bankaların televizyonlarda ve sosyal medyada dolaşan reklamlarının ana temasının mobil hizmetler olması dikkat çekicidir. Bankalar bu hizmetlerin pazarlanmasını yeni müşterilere ulaşmak için etkili görmektedirler.

Hizmet sektörünün makineleşmesinin diğer etkileri de e-ticaret sitelerinde de gözlemlenebilir. TÜBİSAD'ın 2016 yılı verileri 2015 ve 2016 yılları arasında e-ticaret sektörünün %30.8 büyüdüğünü, bu oranın 2016-2017 yılları arasındaki ise %37 olduğunu göstermektedir.

UBER ve AIRBNB gibi birçok yazılım da konaklama ve ulaşım gibi hizmetlerin icrasını olmasa da -sürücüsüz araçların test aşamasında olduğunu görmezden gelerek- pazarlanmasında önemli etkenler haline gelmişlerdir. Bu uygulamaların işletilmesi üzerinden kâr elde edilmektedir.

Bütün bu makineleştirme yani üretim süreçlerinden insanların dışlanmasını mümkün kılacak teknolojik gelişimler, öncelikle başka bir alanda -genel üretim süreci düşünüldüğünde çok daha az- emek gücü talebi yaratır. Bu alan yazılım alanıdır. Üretim süreçlerinin ve hizmet süreçlerinin makineleştirilebilmesi, çok miktarda yazılımcının emeğiyle mümkün olabilmektedir. Bu süreçte yazılımcılara ödenen ücret, bu işler için eğer canlı emek kullanılsaydı bu kimselere ödenecek toplam ücretten daha az olmak zorundadır. Aksi takdirde sermaye üretim ve dolaşım süreçlerini asla dijitalleştiremezdi. Marx (2011: 378) makineleşmenin gerçekleşme koşulunu kendi güncel örneğiyle şu şekilde vurgular:

> Yankee'ler taş kırma makineleri icat etti. İngilizler bunları kullanmıyor; çünkü, bu işi yapan "zavallı" (İngiliz ekonomi politiğinin tarım işçisi için kullandığı teknik terim "wretch"tir), emeğinin o kadar küçük bir kısmının karşılığını alır ki, makine kullanılması üretimi kapitalistler için pahalılaştırırdı. İngiltere'de kanallarda kullanılan tekneleri çekmek vb. işler için zaman zaman hâlâ beygirler yerine kadınlar çalıştırılır, çünkü, beygirlerin ve makinelerin üretimleri için gerekli emek miktarı belli bir matematiksel büyüklüktür; oysa, surplus-population (artık nüfus) içindeki kadınların ayakta tutulmaları için gereken emek miktarı her türlü hesabın altında kalır. Bundan ötürü, insan gücü, hiçbir yerde, en değersiz işler için, makineler ülkesi İngiltere'de olduğundan daha utanmazca çarçur edilmez.

TÜBİSAD'ın (Bilişim Sanayicileri Derneği) 2017 yılı için hazırladığı rapor göstermektedir ki, bilişim sektörü Türkiye'de yılda ortalama %15 oranında büyümektedir. 2016 ve 2017 yılları arasında sektör %18.2 oranında büyümüştür. 2017 yılında sektörün büyüklüğü 30-32 Milyar dolar arasındadır, 120 milyar TL'lik bir sektör hacmi vardır. UİS (Ulusal İstihdam Stratejisi) İzleme Değerlendirme raporu incelendiğinde de 2012-2013 yılları arasında en çok büyüyen BİT sektörü bileşeninin yazılım olduğu görülebilir. Türkiye'deki BİT sektöründeki Pazar bileşenlerinin oranları da son 5 senede gelişmiş ülke dağılımlarına benzemeye başlamıştır. BİT sektörünün Pazar bileşenlerinin %16'sını hizmet, %45'ini yazılım ve %39'unu da donanım oluşturmaktadır. 128.000 kişi bilişim sektöründe çalışmaktadır.

Genel olarak yazılımın da bir parçası olduğu BİT (Bilişim ve İletişim Teknolojileri) sektörünün günümüz kapitalizmi için anlamı Savul'a göre (2017: 189) şöyledir:

> Bankacılık işlemlerinden, havayolu şirketlerinin uçuş planlamalarına kadar birçok etkinlik günümüzde BİT sektörünün sağladığı olanaklar olmadan olmaksızın

> sürdürülememektedir. Genel olarak kapitalizmin devamlılığı günümüzde hem BİT sektöründeki meta üretimine hem de BİT sektörünün diğer sektörlere altyapı sağlamasına bağlıdır.

3. Genel Zekâyı Üretim Araçlarına Dahil Etme: Yazılım Üretim Süreçleri

Yazılım sektöründe nihai ürün bilgisayar programıdır. Söz konusu programların asli üreticileri de yazılımcılardır. Bir yazılım geliştirme süreci genel olarak dört parçadan oluşur. Öncelikle talep edilen programın son kullanıcının ihtiyaçlarını nasıl karşılayabileceğini gösteren bir analiz yapılır. Geniş yazılım ekiplerinde bu işi analistler yapar. Bu durum yazılımcıların işin bütününe dair bilgisini azaltan, onları sadece teknik bilgi sahibi kılan süreçlerin başında gelir. Read'e göre (2014: 161) Marx daha kapitalizmin başlarında görülen manifaktür tipi iş bölümünden beri planlama ve iş arasındaki ayrımın doğurduğu iktidar ilişkilerini incelemiştir. Marx (2011: 349) parça iş ile bütün arasındaki kopukluğun işçinin karşısında mistikleşmiş bir güç gibi dikildiğinin altını çizer:

> Maddi üretim sürecinin düşünsel güçlerinin, işçilerin karşısında bir yabancının mülkü ve kendilerine hükmeden bir kudret olarak yer alması, manifaktür tipi iş bölümünün bir sonucudur. Bu ayrılma süreci, kapitalistin, tek tek işçilerin karşısında, toplumsal toplam işçinin birliğini ve iradesini temsil ettiği basit el birliği aşamasında başlar. Bu süreç, işçiyi parça-işçi şeklinde güdükleştiren manifaktürde gelişir, bilimi bağımsız bir üretim gücü olarak emekten ayıran ve sermayenin hizmetine sokan büyük sanayide tamamlanır.

Fabrika işçileri için bu ayrım yabancılaşmaya neden olan etkenlerden birini doğurur. Ürün başından sonuna kadar işçinin kafasında canlanmaz, sanki işçinin emeğinin ürünü değilmiş de başka bir varlığı varmış gibi "şey"leşir. Ancak yazılım ürünü değişkendir, kimi zaman işin bütününün yazılımcının kafasında da netleşmesi gerekir. Kimi zaman da şirketlerin güvenlik sınırlamaları işi öyle böler ki, yazılımcı standart kodlamanın dışında kalan ama ihtiyacı olan her teknik mesele için başka birimlerden yardım istemek zorunda kalır. Kod geliştirmesi için yazılımcıya gelen analiz teknik açıdan ne kadar detaylı olursa, yazılımcının işin bütününü kavrama gereksinimi o kadar az olur.

Yazılım süreci en genel biçimde analiz, geliştirme, test ve bulunan hataların giderilmesi aşamalarından oluşur. Öncelikle işin analizi yapılır. Sonrasında ise yazılımcının geliştirme yaptığı süreç başlar. Program üretiminin asıl yapıldığı aşama ikinci bölümdür. Sonrasında ise test yapan kişilerin programı son kullanıcıymış gibi kullandıkları ve programdaki hataları[5] yazılımcılara raporladıkları aşama gelir.

Yazılım üretim süreçlerinin verimliliği için, bu süreçlerin yapılandırılması fikri ortaya çıkmıştır. Örneğin, *Agile* ve *Waterfall* gibi proje yönetme metotları yaygın biçimde kullanılır ve farklı ihtiyaçlara yanıt verir. *Waterfall* büyük kapsamlı projeler için kullanılırken, *Agile* bölünebilmesi mümkün ya da küçük kapsamlı projeler için tercih edilir. Bu çalışmada *Agile* metotu incelenecektir.

---

5 Bu hatalara "bug" denir, bu hataları giderme işlemine de "bug fix" denmektedir.

4. Yazılım Yapmada Öne Çıkan Bir Metot: "Agile" Metodu

Neoliberalizm politika ve üretimin kıstasını benzer bir biçimde değiştirdi. Politikada hızlı karar alma, bürokrasinin bir yük olarak tasvir edilmesi ve piyasanın, kârın tek referans noktası olarak kurulmasıyla beraber iş yaşamında da bürokratik süreçlerin ortadan kaldırılması, sonuç odaklılık, performans gibi kavramlar görünür hale geldi. Bürokrasinin tasfiyesi, yanında hukukun da tasfiyesini getirdi. Böylece neoliberalizm süreklileşmiş muğlaklık çağının adı oldu.

Neoliberalizmle değişen sınıfsal güç ilişkileri ile beraber üretimin planlanma biçimi de değişti (Harvey, 2010: 206). Stoksuz üretim daha kârlı hale geldi. Yurtiçi hasılayı yükseltmenin yolu iç tüketim değil, ihracat malı üretmekten geçmekteydi. Bu da dünya piyasasında rekabet gücü yüksek şirketlerin olmasını zorunlu kılıyordu. Bu şirketler de ancak yüksek kârlılıkla var olabilirlerdi. Yani yüksek kârlılık elde etmek için satış fiyatlarını arttırmak nasıl bir yol olabiliyorsa, maliyetleri düşürmek de bir yoldu. Mevcut politik süreçlerin işçi sınıfı aleyhine geliştiği o dönemde de maliyetleri düşürmek daha kolay bir yol olarak görüldü. Yukarıda saydığımız ana akım iktisadi gerekçelere de bir yanıt olarak *kaizen* üretim (yalın üretim) modeli ortaya çıktı.[6]

Günümüzde neredeyse tek referans noktası haline gelen müşteri memnuniyeti *Kaizen* modelinin çıkış noktasıdır. Fordist üretimin anonim bir kitleye yaptığı üretime karşı *Kaizen* muhatap aldığı bir müşteriye üretim yapar gibi üretim süreçlerini sürekli esnek planlar. Müşteri istek değiştirirse hemen üretim süreci modifiye edilir. *Kaizen* üretiminde Ford'un fabrikalarındaki gibi bir üründen binlerce üretilmez, çünkü bu seri üretim yaşanan bir hatanın maliyetini geri dönülmez boyutlara taşıyabilir hem de stokta bekleyen ürün Meta - Para - Meta' - Para' döngüsünde sıkışma yaratır. Bu döngü yavaşladığında sermaye birikimin hızı da yavaşlamış olur.

Agile (Çevik) modeli de *Kaizen* sisteminin yazılım süreçlerine uygulanmasıdır. 2001 yılında bir grup yazılımcının ABD'nin Utah kentinde yayımladığı bir manifestoyla yazılım camiasında uygulanmaya başlamıştır. Elbette birkaç yazılımcının yayınladığı bir manifesto sadece kendinde bir şey olarak yaygınlaşmadı; bu manifestonun maddeleri ekonomik ve siyasal süreçlere büyük oranda uygunluk gösteriyordu. Bürokrasinin, detaylı dokümantasyonun zaman kaybı olduğunu bildiren manifesto yeni siyasal yapı gibi tek bir kıstas koyuyordu bütün yazılım üretim sürecine: Çalışan program. Yayınlandığı konjonktürde yeni sağın siyasal kabulleri politik alanda hegemonik düzeye çoktan erişmişti ve *Kaizen* üretim modeli Batılı şirketlerde yaygınlaşmaya başlamıştı.

Agile manifestosu (2001) temel olarak şu maddeleri öne sürmektedir:

> Bizler şu ilkeleri izliyoruz: En önemli önceliğimiz değerli yazılımın erken ve devamlı teslimini sağlayarak müşterileri memnun etmektir. Değişen gereksinimler

6 Konumuz dışında olması sebebiyle bu üretim modelinin tarihine değinilmeyecektir. Ancak modelin doğum yeri Japonya'dır. II. Dünya Savaşı sonrasında icat edilmiştir.

> yazılım sürecinin son aşamalarında bile kabul edilmelidir. Çevik süreçler değişimi müşterinin rekabet avantajı için kullanır. Çalışan yazılım, tercihen kısa zaman aralıkları belirlenerek birkaç haftada ya da birkaç ayda bir düzenli olarak müşteriye sunulmalıdır. İş süreçlerinin sahipleri ve yazılımcılar proje boyunca her gün birlikte çalışmalıdırlar. Projelerin temelinde motive olmuş bireyler yer almalıdır. Onlara ihtiyaçları olan ortam ve destek sağlanmalı, işi başaracakları konusunda güven duyulmalıdır. Bir yazılım takımında bilgi alışverişinin en verimli ve etkin yöntemi yüz yüze iletişimdir. Çalışan yazılım ilerlemenin birincil ölçüsüdür. Çevik süreçler sürdürülebilir geliştirmeyi teşvik etmektedir. Sponsorlar, yazılımcılar ve kullanıcılar sabit tempoyu sürekli devam ettirebilmelidir. Teknik mükemmeliyet ve iyi tasarım konusundaki sürekli özen çevikliği artırır. Sadelik, yapılmasına gerek olmayan işlerin mümkün olduğunca azaltılması sanatı, olmazsa olmazlardandır. En iyi mimariler, gereksinimler ve tasarımlar kendi kendini örgütleyen takımlardan ortaya çıkar. Takım, düzenli aralıklarla nasıl daha etkili ve verimli olabileceğinin üzerinde düşünür ve davranışlarını buna göre ayarlar ve düzenler.

Maddelerden de anlaşılacağı gibi, bürokrasinin azaltılması, dokümantasyon gibi evrak-kağıt işleriyle vakit kaybedilmemesi, en hızlı şekilde, her durumda çalışan bir yazılım ürününün üretilmesi ana hedeftir. Müşteri projenin sonunda dahi ürünü değiştirmeyi talep edebilir. *Agile* için çalışan program esastır. İletişim enformaldir, doküman yoluyla yapılmaz. Metot, proje sürecinde periyodik toplantılar dayatır.

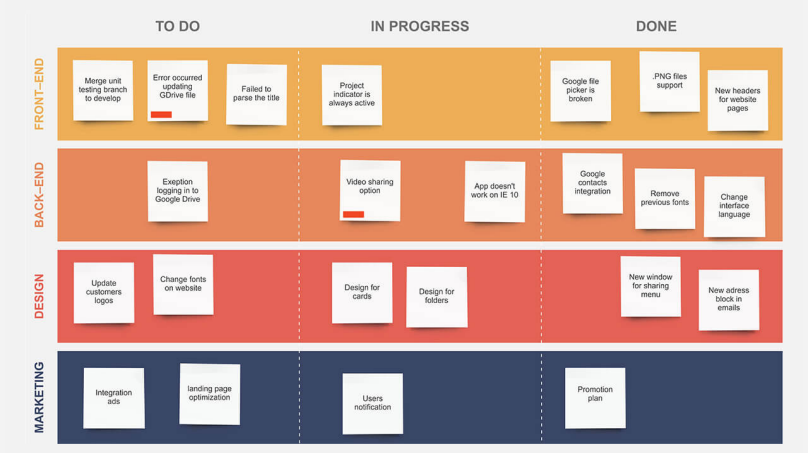
1980'li yıllarda (örneğin uçakların kontrol sistemlerinin yazılımı gibi) 10 yıllık yazılım projeleri yapılırken, *Agile* sistemi gibi bir ihtiyaçtan söz edilemezdi. Öncelikle bu projelerin iyi dokümante edilmesi zorunluluktu çünkü projede çalışanlar sürekli değişmekteydi. İyi planlanmalıydı, belirli bir sürenin ardından bir ihtiyacın karşılanması imkânsız ise proje çöp olabilirdi. Ancak yazılım teknolojisi geliştikçe yazılım geliştirme süreçleri kısaldı. Diğer yandan Sommerville'in vurguladığı gibi (2009: 3) internet gibi birçok teknoloji gündelik yaşama girdikçe yazılım ürünlerine olan talep arttı, uçak yazılımından çok daha basit olan e-ticaret gibi birçok yazılım ürünü piyasada istenir hale geldi.

Agile sistemi bütün projeleri küçük projelere bölerek işe başlar. 1 sene boyunca sürecek bir proje 26 farklı küçük projenin bir sonucu olarak ortaya çıkacaktır. Her iki haftada bir, küçük projelerden birine başlanır ve iki hafta sonunda bunlardan birisi bitirilir (ACM, 2010: 6). Böylece müşteri ile sürekli iletişim halinde olunur. Yazılım sürecinde yapılan bir hatanın maliyeti en fazla iki haftalık emek-zaman kaybına neden olur. Diğer taraftan zaten müşteri projenin başlama aşamasında proje sonunda ne elde etmek istediğini tam olarak bilemez, bu nedenle süreci yakından takip etmesi ve fazla ilerlenmeden isteklerini belirtmesi tasarruflu bir üretim süreci oluşturur.

Agile sistemi enformal iletişim biçimlerini aktif olarak kullanır. Yüz yüze iletişime öncelik verilir. Her sabah "Daily" adı verilen 15'er dakikalık toplantılar yapılır. Her mini proje bitişinde -iki haftada bir- müşteri ile bir önceki projenin kapanışı ve yeni projenin planlaması yapılır. Bu toplantı sonrasında da "retrospektif" denilen grup

içindeki anlaşmazlıkların ve sorunların konuşulduğu bir toplantı yapılır.

Müşteri ile mini projelerin başlangıcından önce, alınacak yeni işlerin tartışıldığı "Grooming" adı verilen bir başka toplantı da gerçekleştirilmektedir. Bütün ekibin katılmasının zorunlu olmadığı bu toplantılarda yeni işlerin yapılıp yapılamayacağı, iki haftalık sürenin yeterli olup olmayacağı konuşulur.



Şekil 1: Agile tahtası

Planlama toplantılarından sonra işler yukarıdaki gibi bir *Agile* tahtasına geçirilir. İşler bölüşülür ve başlanılan işler "in progress" bölümüne alınır. Sonrasında da bu işler yapılacaklar, yapılmakta olanlar ve yapılanlar olarak her gün güncellenir.

Agile sisteminin teorik anlatısı bu şekildedir. Ancak pratikteki uygulamalarında bu anlatının hesaba katmadığı birçok çatışma alanı ortaya çıkar ve çoğu zaman normal süreç istisnai durumun varlığıyla işyerinde askıya alınır. *Agile* yazılımcının ofisteki bütün zamanını işe kanalize edebilmesinin bir yöntemini sunmaya çalışır ancak Türkiye'deki birçok ofis az yazılımcıyla daha fazla proje yapma niyetinde olduğu için *Agile* ile sıkıştırılan zaman zorunlu olarak fazla mesailere neden olur.

Çoğu firmada neredeyse her iki haftada bir kriz ortamı yaşanır. Halihazırda çalışmakta olan projede –çünkü Agile sisteminin en önemli özelliği bir sürecin sonunda çalışan projenin teslimi değildir, her iki haftada bir yeni fonksiyon eklene eklene ilerlenir- yaşanan bir sorun doğrudan kriz ortamı doğurur. Bu kriz ortamı sorgusuz sualsiz kabul edilen fazla mesai demektir. Halbuki *Agile*'ın yazılımcıları ikna etmek için öne sürdüğü ilk iddia verimli çalışarak fazla mesailerin önüne geçileceğidir. Ancak çalışan bir programa yapılan her ilave, o programın halihazırdaki çalışmasının sorumluluğunu almak anlamına gelir.

Diğer taraftan iki haftalık projenin sonunda son tarihe yetiştirme baskısı yaşanır. Bu da fazla mesaiyi meşrulaştırıcı etki yapar. Yazılımcıların en çok fazla mesaiye kaldıkları zamanlar projelerin son tarihlerinin yaklaştığı zamanlardır. İşin

planlanmasında yazılım ekibi de olduğu için fazla mesai tüm yazılımcılar için meşru görülmektedir. Yani *Agile* sistemi –diğer birçok sistem gibi- ücret almadan daha fazla çalışmanın kişiler arasındaki meşru zeminini yaratmaktadır. Çünkü planlama toplantısında yazılım ekibiyle müşterinin ortak kararının ihlali gerçekleşmek üzeredir, bu da hem ahlaki açıdan hem firmalar arasındaki anlaşma gereği, mali açıdan kabul edilemez, tek çözüm fazla mesai yapılmasıdır. Yazılım sektöründe fazla mesainin ücretlendirilmesi istisnadır.

Hele ki sektörde çoğu iş yerinde az çalışanla daha fazla ürün elde ederek kârlılığını arttırmak gibi bir iş politikası uygulanırsa, sözü edilen fazla mesailer sayıca artmaya devam eder. Çalışılan işyerindeki bir ekipte 2 yazılımcı 3 analist varken diğer ekipte 2 analist 3 yazılımcı vardır. Halbuki süreç başlangıcında her ekipte 4 yazılımcı 2 analist olacağı söylenmişti. Yazılımcının fazla olması önemlidir çünkü her *sprint*[7] en az 8 yazılım geliştirmesinin çalışır düzeyde müşteriye teslim edilmesi şart koşulmaktadır. 2 yazılımcının 4'er iş yapması zorunlu olarak fazla mesaiye kalmaları demektir.

Öte yandan *Agile*'ın en önemli özelliği işin ölçümüne getirdiği sistemdir. Her iş ekip tarafından müşterinin gözü önünde puanlanır. Verilen puanlara *efor* denir. *Efor* bir kişinin bir saatlik çalışma süresi üzerinden hesaplanan bir emek-zaman birimidir. Her ekibin 400 efor alma zorunluluğu vardır. Bu, *Agile* sisteminin yarattığı bir çatışma alanıdır. Ekip daha fazla mesaiye kalmamak ve stres altında çalışmamak için her işe daha yüksek *efor* verme niyetindedir. Müşteri ise[8] işlerin az *efor*lanmasını sağlayıp daha az sürede daha fazla iş yapılmasını sağlama niyetindedir. İlk bakışta yazılımcıların teknik bilgisi burada belirleyici bir faktör gibi dursa da şirketler arasındaki anlaşmalar ve *Agile* sisteminin sürekli daha fazlasının yapılması için yapılandırdığı kurallar yazılımcının teknik iktidarını elinden alır. Öncelikle ne kadar efor çıkarsa çıksın anlaşma gereği 2 haftada 8 iş teslim edilmelidir. Diğer yandan yüksek verilen eforlar sürekli aşılması gereken hedefler doğurur. Yani 8 işe 600 efor verilen bir süreç sonrasında 8 işe 400 efor verilmesi düşük performans göstergesi kabul edilir.

Bu süreçte sürekli iletişim hali, ekip içindeki ilişkilerin kuvveti ve *Agile*'ın ölçülmesini kolaylaştırdığı yazılımcının emek-zamanının yarattığı disipline edici etki, sermaye lehine üretim sürecine dahil edilir. Sıradan *Agile* işleyişinin yanında süreklileşmiş krizlerin yarattığı baskı mekanizması da yazılımcıların ücretini alamadıkları fazla mesailerin süresini uzatmaya yarar. Yazılımcının fazla mesaide çalıştığı her süre mutlak artı değerin arttırılmasını sağlar.

Öte taraftan da *Agile* sisteminin getirdiği *efor*lama uygulaması, yazılım projelerinde soyut emeğin somutlanmasına yarar. Yazılımcılar arasındaki tecrübe farkının yarattığı teknik kapasite de böylece nötrlenmiş olur. Her yazılımcı emeği eşitmiş gibi kabul edilir. Nitelik farklılıklarına rağmen işler eşit dağıtıldığında,

7 İki haftalık mini proje süreçlerine *sprint* denmektedir.

8 Toplantıya katılan müşteri temsilcine Product Owner (Ürün sahibi) denilmektedir.

deneyimsiz yazılımcıları daha kolay fazla mesaiye zorlanırlar.

Nesnel bir emek zaman ölçümünün olduğunun kabulü proje içindeki kontrolü sağlar, her hafta istatistikler çıkarılarak yöneticilere gönderilir. Bu istatistiklerin yukarı yönlü bir grafik yaratmaması çalışanların suçu olarak dillendirilir. Bu da üretim ilişkilerinde ahlaki, etik ve duygusal birçok mekanizmanın etkin şekilde bulunduğunu gösterir. Ölçümlerin getirdiği suçluluk duygusu daha kolay ücretsiz fazla mesainin yazılımcı tarafından kabul edilebilmesinin ahlaki, duygusal meşruiyetini yaratır.

Agile'ın getirdiği sürekli iletişim durumu da yazılımcının denetlenmesini kolaylaştırır. İki haftada bir yazılımcının yaptıkları ölçülür. Bu haftalar arttıkça yazılımcının emek hızı kavranabilir hale gelir. Bu ölçümlerle beraber, emeğin niteliği ne olursa olsun, yazılımcıdan beklenen bir emek zaman değeri elde edilir. Şimdi yanıtlanması gereken soru, tasvirini yaptığımız *Agile* süreci ile yazılımcı emeğinin içine dahil edildiği gayri maddi emek literatürün birbiriyle ilişkisidir.

5. Yazılımcı Emeğin Maddi Tabiatı: Gayri Maddilik ve Maddilik Arasındaki Salınım

Yazılımcı emeği çoğu zaman, ürettiği ürünün maddiliğiyle ilişkili olarak gayri maddi emek –maddi olmayan emek- olarak görülür.

Ancak yazılımcının gözünde yazılımcının ürettiği şey çoğu zaman açıktır ve bu ürüne ancak yoğun emek süreci sonrasında erişilebilir olduğu yazılımcılar tarafından kolaylıkla anlaşılabilir. Bir bilgisayar programının en temel özellikleri vurgulanırken yazılım mimarisi teriminin kullanılması ya da oldukça deneyimli yazılımcılara yazılım mimarları denmesi tesadüf değildir. Yazılımcıların gözünde bilgisayar programı kodlamak bina inşa etmek gibidir. Yazılım işinin temeli insanlığın bin yıllardır hemhâl olduğu matematiğe dayanır. Yaygın biçimde kullanılan yazılım dillerinden biri olan "Java", Nesneye Yönelik Programlama (*Object Oriented Programming*) türünde bir dildir. Aslında yazılım teknolojileri geliştikçe programlama dilleri insan dillerine yakınsamaktadır.[9]

Bir yazılım ürününe harcanan emek, doğrudan ürünün kendisinden anlaşılamaz, ürün içinde barındırdığı emeği doğrudan göstermez. Metanın bir bölümü göz önündedir. Programın arkasında çalışan milyonlarca kod satırının içine, yazılımcıların emek zamanı aktarılmıştır. Bir internet sitesinde tıklanan bir buton saniyeden daha kısa sürede istenilen işi yerine getirir, ancak sadece o butonun yapacağı işin kodlanması haftalar alabilir. Sitede işlem yapan kullanıcının bu butona ne kadar emek harcandığını kestirebilmesi çok zordur. Bunun birinci nedeni, telif hakları gibi hukuki düzenlemelerle yazılım ürününün ticari bir meta olabilmesi için gerekli koşulların yaratılmasıdır. Böylece para ödenmeden kaynak kodların görülebilmesi imkânsız hale gelir. İkinci neden yazılım ürünü açık

9 "C" gibi nesneye yönelik olmayan programlama dilleri makine diline daha yakınken "Java" ve ".Net" gibi ileri teknoloji programlama dilleri insan diline daha yakındır. Yazılım dillerinin teknolojik hiyerarşisi –yüksek teknoloji ve düşük teknoloji ayrımı- insan diline yakınsaması kıstasına göre belirlenir (Chen, 2010: 45).

kaynak kodlu olsa[10] bile, ürüne ne kadar emek harcandığını anlamak için bu kaynak kodları inceleyebilecek teknik donanıma sahip olmamadır. Ürüne harcanan emeğe dair sağlıklı bir tahmine ancak teknik inceleme yoluyla ulaşılabilir.

Bilgisayar programlarının gayri maddi ürünler olarak anlaşılmasının bir diğer sebebi bir kere yazılıp derlendikten sonra sınırsız ve kolayca çoğaltılabilmelerinden kaynaklanır. Bu üründen bir tane daha üretmek için programın kopyalanması yeterlidir. Bant üretimi yapılan bir fabrikada, seri üretime geçildikten sonra her ürünün yapılması için belirli bir sabit sermaye –makineler ve ürün için gereken hammadde-, emek gücü ve *t* süresi gerekirken, yazılım ürününün çoğaltılması için bunlara gerek yoktur.

Gayri maddi ürün olarak düşünülen bilgisayar programını üreten emek de gayri maddi emek olur. Lazzarato'ya göre (1996: 132) gayri maddi emek ürününe bakılarak şu şekilde ayırt edilebilir:

> Maddi olmayan emek kavramı emeğin iki farklı yönüne işaret eder. Bir yandan, metanın "enformasyonel içeriği" bakımından, maddi olmayan emek; dolaysız emek için gerekli olan becerilerin artan şekilde sibernetik ve bilgisayar hakimiyetine (ve yatay ve dikey iletişim) dair beceriler gerektirdiği endüstri sektörü ve üçüncü sektörlerdeki büyük şirketlerde çalışan işçilerin emek süreçlerinde gerçekleşen değişimlere doğrudan bir gönderme yapar. Öte yandan, maddi olmayan emek metanın "kültürel içeriğini" üreten etkinlik bakımından, normalde "iş" olarak kabul edilmeyen bir dizi etkinliği içerir; diğer bir deyişle, kültürel ve sanatsal standartları, modayı, zevkleri, tüketici normlarını ve daha stratejik olarak kamuoyunu belirlemeye ve kurmaya yönelik etkinlikleri.

Lazzarato, gayri maddi emeğin enformasyonel ve kültürel içerik ürettiğinin altını çizmiştir. Gayri maddi emeğin 21. yüzyılda daha fazla tartışılmasının nedeni ise bu ürettiği unsurların sermaye birikimindeki etkisidir. Bu kavramın günümüz kapitalizminde görünür olmasının nedeni, artı değer üretiminde bilginin ve kültürel içeriğin hegemonik olmaya başlamasıdır. Negri ve Hardt (2004: 122) maddi olmayan emeğin üretiminin yükselişini şöyle ifade eder: "XX. yüzyılın son on yıllarında, endüstriyel emek hegemonyasını yitirdi ve onun yerine 'maddi olmayan emek,' yani bilgi, enformasyon, iletişim, ilişkiler veya duygusal ifade gibi maddi olmayan ürünler üreten emek geçti." Bu hegemonik hale gelen "maddi olmayan emek"in ürünlerini de şu şekilde tanımlarlar: "Maddi olmayan emek, fikirler, semboller, kodlar, metinler, dilsel figürler, imajlar gibi ürünler üretir. Maddi olmayan emeğin diğer ana biçimine de duygulanımsal emek diyoruz (Hardt ve Negri, 2004: 122)."

Gayri maddi emek kavramı bunca tanımlama çabasına rağmen muğlak kalmaktadır. Yani onu diğer emek biçimlerinden ayıran nedir? Negri ve Hardt'a

---

10 Açık kaynak kodlu programlar (open-source program) kaynak kodlarına erişilebilir programlardır. Bu nedenle ücretsiz olarak erişilebilirler ve kaynak kodları açık olduğu için ihtiyaç halinde yazılım uzmanları tarafından revize de edilebilirler. Kapalı kaynak kodlu bir yazılım ürünü bazı yöntemlerle ücretsiz -korsan- kullanılabilir ancak revize edilmesi çok zordur –imkansız değildir-. Böylece programın kullanılabilmesi ve programda istenilen değişikliklerin yapılabilmesi için, programı oluşturan şirketlere para ödenmesi gerekmektedir.

göre (2004: 123) maddi olmayan şey emek değildir, emeğin ürünüdür. Negri ve Hardt (2004: 123) bu muğlaklığı "biyopolitik emek" kavramıyla aşmaya çalışırlar:

> Yeni hegemonik biçimi "biyopolitik emek," yani sadece maddi mallar üretmekle kalmayıp ilişkileri ve de toplumsal yaşamın kendisini de üreten emek olarak algılamak daha doğru olabilir. Dolayısıyla *biyopolitik* terimi, ekonomik, siyasal, toplumsal ve kültürel arasındaki geleneksel ayrımların giderek bulanıklaştığını anlatır.

Nasıl "ekonomik, siyasal, toplumsal ve kültürel arasındaki geleneksel ayrımlar giderek bulanıklaş"ıyorsa iş ve iş dışı zamanları da bulanıklaşır. Bir adım daha öteye gidersek, gayri maddi emek icra eden kişinin yaşamının iş, işinin yaşamı olduğunu iddia edebilir hale geliriz. Gerçekten de gayri maddi emek, tanımı gereği işle iş dışı arasında bir ayrımı tanıyamaz. Negri ve Hardt (2004: 126) bu muğlaklığı şu satırlarla vurgularlar:

> Örneğin maddi olmayan emek paradigmasındaki iş gününün değişimini, yani iş ve dinlenme zamanının giderek belirsizleşmesini ele alalım. Endüstriyel paradigmada, işçiler üretimi neredeyse sadece fabrikada yapardı. Oysa üretim bir problemi çözmeye ya da bir fikir ya da ilişki yaratmaya odaklandığı zaman, iş zamanı yaşamın tüm zamanına yayılmaya başlar. Bir fikir ya da imaj insanın aklına sadece ofiste değil, duşta ya da rüyada da gelebilir.

Böylece, Marx'ın emeğin ölçüsü olarak zamanı parametre alması boşa düşmüş olur, diğer taraftan mutlak artık değer ve göreli artık değer kavramları da bütün bir yaşama yayılmış iş ve emek süreci yüzünden kullanışsızlarşır. Ancak, Yıldırım ve Türkcan'ın yaptıkları (2017: 132) nitel çalışmada yazılımcıların fazla mesaiden dert yandıkları gözlemlenmiştir:

> Fazla mesai ya da eve iş götürme sıradan karşılanmaktadır. Bunun karşılığında hiçbir ücret almadıkları gibi takdir de görmemektedirler. İzin günlerinde işe ne zaman çağrılacakları bilinmemektedir, işyerini ev gibi kullanma, tatile iş götürme gibi durumlarla çoğu zaman karşılaşmışlardır. Mühendisler, müdürlerinin bir arkadaşlarına balayında kod yazdırdıklarını bile söylemişlerdir.

Tam da Negri ve Hardt'ın biyopolitik veya gayri maddi emek kavramlarıyla hegemonikleştiğini iddia ettikleri şey, bir çatışmanın konusudur ve hegemonik değildir, çünkü yazılımcıların buna rızası olduğunu ön kabul olarak ele almak mümkün değildir. Bu durumun göz ardı edilmesi, yazılım alanındaki direniş biçimlerini gözden kaçırmaya neden olur ve bir diğer genel kabulü doğurur: Yazılımcılar –genel olarak beyaz yakalılar-, sınıflarının bilincinde değildir. Ancak yazılımcıların hem mücadele ettikleri yönetim stratejilerinin hem de bu stratejilere karşı geliştirdikleri direniş biçimlerinin farklılaştığı düşünülürse ancak o zaman sağlıklı bir sınıf tartışması yürütülebilir ve erken verilmiş teorik kararlar sorgulanabilir.

Bu çalışmada biz yazılımı anlamak için mutlak artı değer ve göreli artı değer kavramlarının yine de işlevli olduğunu düşünüyoruz. Marx'ın iş süreçlerini anlamak için kullandığı kavramları biz de yazılım geliştirme süreçlerini anlamak için kullanıyoruz.

Ancak mutlak artı değer kavramında yazılım sektöründe gözlemlenen bir değişim olduğu da doğrudur. Çalışan bir bankacılık uygulamasını düşünelim. Bu uygulama para transferleri, borsa işlemleri, döviz alım satım, kredi talep etme gibi eskiden kişilerin yaptığı işlerin birçoğunu yapabilmektedir. Yani, işi yapan artık bu uygulamadır. Yazılımcının görevi bu uygulamaya eklemeler çıkarmalar yapmak, aynı zamanda uygulamanın 7/24 çalışır olmasını sağlamaktır. Yazılımcı artık uygulamaya tabidir, hizmetin gerçekleştirilmesi uygulamanın işidir, yazılımcının işi uygulamaya "göz-kulak" olmaktır. Yazılımcı aynı zamanda uygulamanın yazıldığı teknolojiye tabidir. Böylelikle yazılımcı edindiği niteliklerin bir kısmından da mahrum kalır. Çünkü uygulama hangi programlama dillerinde yazıldıysa, yazılımcı o programlama dillerini kullanmak zorundadır.

Diğer taraftan uygulamanın sürekli "ayakta" olması zorunluluğu yazılımcının mesai saatlerini muğlaklaştırır. Ancak burada durum, otonomist Marksistlerin toplumun fabrika haline geldiği ve bu yüzden değer üretiminin bütün mekânı ve toplumu kapsadığı iddiasından farklıdır. Buradaki neden uygulamanın sağladığı hizmetlerin 7/24 olmasıdır. Burada muğlaklaşmış biçimde mutlak artık değer vardır. Yazılımcının kendisini yeniden üretmesi için gereken zaman, sürekli ayakta olan bilgisayar programına bağımlılaşmıştır. Gece çalan telefonlar, mesai saatleri dışında müdahale edilmesi zorunlu olan hatalar, bütün bunlar iş saatlerini muğlaklaştırır, ancak bu işle yaşamın ayırt edilemez olduğu anlamına gelmez.

Marx'ın üretim ilişkilerini anlamak için kullandığı bir diğer kavram olan göreli artı değer kavramı ile yazılım geliştirme süreçlerini düşündüğümüzde ise, yazılımcının dahil olduğu iş süreçlerinde göreli artı değeri arttırmanın üç yolu olduğunu görürüz. İlki yazılım teknolojisinde biriktirilen kümülatif bilginin üretim süreçlerine dahlidir. İkincisi ise, *Agile* metodunun çıkış nedeni olan, projenin ilerleyiş biçiminin organizasyonundaki değişimdir. Üçüncüsü ise yazılımcılar arasındaki ağdaki bilgi birikiminin metaya ya da sabit sermayeye aktarılmasıdır.

İlk maddedeki en önemli faktörlerden biri, topluluklarının ücretsiz olarak ürettikleri, yazılımcılar için oldukça kullanışlı olan geliştirmelerdir. "Java" programlama dili için geliştirilmiş yapılar (*framework*) yazılımcının hızlı kod geliştirmesine yarayan kod parçacıklarıdır. Örneğin "Spring" yapısı açık kaynak (*open source*) olarak geliştirilmiş bir yapıdır ve günümüz yazılım geliştirmelerinin üretiminde ücretsiz bir biçimde bu yapının avantajlarından yararlanılabilir. Her yazılım dili için Spring gibi ücretsiz ve kod yazma hızını arttıracak yapılar bulmak mümkündür. Bunu sermaye doğrudan finanse etmemiştir ancak yazılım geliştirme süreçlerinde yazılımcıların sermaye adına daha hızlı ve verimli bu bilgi ile mümkün olmuştur. Terranova (2015) internetin gelişimiyle var olan dijital ekonominin büyük bölümünün karşılığı ödenmemiş emeğe dayandığının altını çizer. Bu emeğin öncelikle göze çarpan iki türü vardır. Birincisi, internet sitelerinin kullanıcısı olmak yoluyla harcanan emektir. İkinci emek türü ise "Spring" gibi yapıları oluşturan

yazılım uzmanlarının bu yapıları oluştururken harcadıkları ve karşılığı ödenmeyen, keyif için icra edilen emektir. Bu emeğin ürettiği yapıların şirketlerin yazılım ofislerinde arttırdığı verimlilik,[11] sermaye birikiminin hızlanmasına ve sermayenin karşılamak zorunda kaldığı emek maliyetinin azalmasına neden olur.

Yazılımcı topluluklarının yazılım süreçlerini kısaltmak için ürettikleri yapılar genel zekânın yazılım sektörüne uyarlanması olarak anlaşılabilir. Marx (2014: 570), göreli artı değeri arttıran en önemli unsurlardan birinin makine içinde fabrikaya giren genel zekâ olduğunu göstermiştir. Makinede sabit sermaye biçimini almış genel zekâ, sermayenin kısa zamanda daha çok ürün elde edebilmesini mümkün kılmıştır. Yazılım sektöründe de bu yapılar, sermaye birikiminin hızlanmasında etkilidir.

Yazılımcıdan elde edilen göreli artı değeri arttırmanın ikinci yolu ise, yazılım geliştirme sürecinin yapılandırılmasındaki hamlelerdir. Yukarıda değindiğimiz *Agile* manifestosunun maddelerini tekrar incelendiğinde kişiler arasındaki iletişimin metaya aktarılmasının planlandığı kolaylıkla görülebilir: "Bir yazılım takımında bilgi alışverişinin en verimli ve etkin yöntemi yüz yüze iletişimdir." *Agile* metodu iletişimi zorunlu kılar: "İş süreçlerinin sahipleri ve yazılımcılar proje boyunca her gün birlikte çalışmalıdırlar." Böylece garanti altına alınmış sürekli iletişimin ürettiği değer ürünün içinde sermayeye aktarılır.

Aynı zamanda *Agile* metodunda bireylerin duyguları da üretim sürecine dahil edilmeye çalışılır; arzu, istek ve motivasyonu önemlidir: "Projelerin temelinde motive olmuş bireyler yer almalıdır. Onlara ihtiyaçları olan ortam ve destek sağlanmalı, işi başaracakları konusunda güven duyulmalıdır."

Agile metodunun göreli artı değeri arttırmak için iş sürecine dayattığı son unsur da "önceliklendirme"dir. İşlerin önem sırasının belirlenmesi, her işin yapılması değil, müşteriye ya da patrona en fazla kâr getirecek işin en önce yapılması: "Sadelik, yapılmasına gerek olmayan işlerin mümkün olduğunca azaltılması sanatı, olmazsa olmazlardandır."

Agile sistemi takımların otonomisini de önemser. Böylelikle üretim sürecinde oluşan deneyim, bu otonomi yoluyla sermayeye kolaylıkla aktarılabilir: "Takım, düzenli aralıklarla nasıl daha etkili ve verimli olabileceğinin üzerinde düşünür ve davranışlarını buna göre ayarlar ve düzenler."

"Retrospektif" toplantılarıyla da takım içi potansiyel çatışma noktaları yönetilmeye çalışılır. *Agile*, takım içinde çatışma olasılığını her zaman kabul eder ve bu olası çatışmanın yönetimini de üstüne alır. Takımın otonomisi, iyi planlanmış bir takvimle sınırlandırılmıştır. Takım içi duyguların sermaye adına üretim ilişkilerine dahil edilmesi de takvimlendirilmiş toplantılar vasıtasıyla olur. Bu potansiyel

11 Bir yazılımcı Spring kullanmadan 3 günde yapabileceği bir işlemi, Spring yapısı sayesinde yarım saat gibi kısa bir sürede yapabilir. Spring'in modüler yapısı güvenlik, veri tabanı işlemleri gibi standartlaşmış işlemleri paket halinde yazılımcıya sunar (Acar, 2014: 36). Özellikle "dependency injection" teknolojisi birbirine bağımlı olan sınıfların ve metotların yönetimini tek bir konfigürasyon dosyasında toplayarak yazılımcının her bir sınıfı kullanılan her ayrı sınıfta tanımlamasının önüne geçer. Böylece enjekte edilen sınıfta yapılan bir değişiklik bu sınıfa bağımlı olan diğer bütün sınıflarda da görülebilir.

çatışma, *Agile* tarafından üretkenliği arttırabilecek bir ihtimalmiş gibi kabul edilir ve insanların çatışabileceği toplantılar kurgulanır.

Proje yönetim metodunun dışında yazılımda göreli artı değeri arttıran üçüncü yol ise, yazılımcıların çözdükleri sorunların çözüm yollarını birbirleriyle paylaşmalarıdır. Ofis yaşamında fazlaca gözlenen şey bu ağın aktif üretime katılma biçimidir. Çoğu yazılımcının kullandığı, üyesi olduğu internet sitesi grupları vardır.[12] Bu internet siteleri "ağ" yaratır ve dünya üzerindeki neredeyse bütün yazılımcılar bu siteleri bilirler. Bu internet siteleri aracılığıyla kamusallaşan bilginin ofis içinde üretilen yazılımla üretim ilişkilerine dahil edilmesi zamandan tasarruf sağlar. Yazılımcı ağ olmadığı takdirde günlerce çözmek için çabalayacağı bir sorunu dakikalar içinde çözüp başka bir işe devam edebiliyorsa, bu durum ağın yazılımcı üzerinden elde edilen göreli artı değeri arttırdığını gösterir.

Göreli artı değer kavramıyla beraber Marx, biçimsel boyunduruktan gerçek boyunduruğa geçişin yaşandığını vurgulamıştır. Gerçek boyunduruk işçinin sermaye olmadan iş görememesi ve aynı zamanda iş üzerindeki bütün hakimiyetini yitirmesi demektir. Gerçek boyunduruğa geçiş süreci aynı zamanda bir niteliksizleşme sürecidir. Ancak yazılımcının verimlileştirilmesi sürecinin aynı zamanda bir niteliksizleştirme süreci olup olmadığı tartışmalıdır. Bu tartışma yazılımcıların homojen bir bütün oluşturmamalarından ve işlerinin hem nitelik isteyen hem de nitelik istemeyen yanlarının bulunmasından kaynaklanmaktadır.

Yazılımcının niteliksizleşme sürecini anlayabilmek için yazılımcı emeğini anlamak gerekir. Bu yolda Franco "Bifo" Berardi (2014: 110) endüstri emeği ile bilişsel emek arasında keskin bir ayrım yapar ve gayri maddi emek bağlamındaki tartışmalara "kogniterya" kavramı ile müdahil olur. Berardi'ye göre (2012: 82) emek biçimleri arasındaki temel ayrım endüstriyel emeğin durağanlığı ile bilişsel emeğin akışkanlığı arasındadır:

> Endüstriyel emek iletişimi akla getirmezken ve arzulayıcı enerjileri çekmezken, bilişsel emek için tersi söylenebilir. Enformasyon işçileri bilgilerini ve yaratıcılıklarını ağların yaratılması sürecine yatırdıkları içindir ki zaman zaman zanaatkar olarak nitelenirler. Sürekli değişen genel bir çerçeve içinde yeniden birleştirebilmek için enformasyon parçalarını yakalamaya çalışan bu işçiler, enerjilerini üretim ağının bir noktasından başka bir noktasına kaydırır.

"Bifo" Berardi (2012: 83) emek biçimleri arasındaki bu değişimin altını çizdikten hemen sonra, bu değişimin ağ içinde işleyen bütün emeklerin karakteristik bir özelliği olmadığının da altını çizer: "…bir şekilde zihinsel emek faaliyeti olarak tanımlanabilecek tüm iş biçimleri iletişimle, yeniliklerle ve yaratıcılıkla bağlantılı değildir."

Öyle ki yazılımcı edindiği nitelikleri iş yaşamında kullanabilme imkanını da

12 Bunların en ünlüsü ve en çok kullanılanı; https://stackoverflow.com/ sitesidir. Yazılımcıların sıklıkla karşılaştıkları sorunları buradan paylaşırlar ya da burada daha önce bu sorunlarla boğuşmuş ve çözmüş kişilerin yazdıkları çözümlere ulaşırlar. Bu sitenin kendisi de bu ağı yaratarak para kazanmaktadır. Ağ yaratmanın kendisi günümüzde etkili kazanç yöntemlerinden biridir.

kolay kolay bulamaz. Upadhya ve Vasavi'nin (2008, Aktaran: Fuchs, 2015: 298) Hintli yazılımcılar üzerine yürüttükleri alan çalışması da Hintli yazılımcıların niteliklerinin yaptıkları işin çok üstünde olduğunu belirtmektedir. Ancak Lazzarato (1996: 133) gayri maddi emeğin sermaye birikiminde öneminin artmasıyla üretim ilişkilerindeki emek biçiminde görülen değişimi şöyle tarif eder:

> 1970'li yıllarda başlayan 'büyük dönüşüm', sorunun ortaya konuluş koşullarının kendisini değiştirmiştir. Kol emeği artan şekilde "entelektüel" olarak tanımlanabilecek zihinsel işlemler içermeye başlamıştır ve yeni iletişim teknolojileri artan şekilde bilgi açısından zengin öznellikler gerektirmektedir.

Türkiye'deki yazılımcılar da Hintli yazılımcılarla benzer niteliklere sahiptir. Her iş ilânında altı çizilen "yeni teknolojileri takip etme" özelliği ofis yaşamında çoğu zaman körelmektedir. Türkiye'de şirketlerin kullandığı ve bakım-onarım yapılan bilgisayar programlarının çoğu eski yazılım teknolojileriyle yazılmışlardır. Bunların yenilenmesi şirketler için yüklü maliyet demektir. Çoğu zaman yazılımcılar bu projelerde çalışmak zorunda kalmaktadırlar. Bu da onların teknik bilgi birikimlerini sekteye uğratmaktadır.

Bu teknik bilgi birikiminin sekteye uğramasının yanında *Agile* gibi proje yürütme metotları projenin nasıl takvimlendirileceği konusunda yazılımcıları vasıfsız kılar. Proje yönetimindeki bu otoritesizlik yazılımcının her anlamda vasıfsızlaşacağı bir süreci tetiklemez ancak yazılımcı metodun takvimine bağımlılaşmıştır.

Hem teknik bilgisinin köreltilmesine hem proje yönetimindeki otoritesizliğine rağmen yazılım sektörü hemen her gün yeni teknolojilerin doğduğu bir alandır. Az sayıda da olsa yaratıcılık isteyen işler vardır. Yazılımcının bu işlerde çalışabilmesi için yeni yazılım teknolojilerini takip edebilmesi gerekir. Böylelikle yazılımcı, *Agile* ile daha mekanize bir emek sürecinin parçası haline gelirken; projenin gereksinimlerinin getirdiği yeni teknolojileri ve "know-how"ı öğrenmek zorundadır, bu da emeğin niteliksizleşmesi sürecinin tersine bir etki yaratır. Yazılımcı, işçi ve zanatkâr arasında salınmaktadır. Berardi'nin vurguladığı zihinsel emeğin farklı bileşenleri, tek bir yazılımcının iş hayatında görülebilir. Berardi'nin önerisi (2012: 83) ve bilişsel emek sınıflandırması şöyledir:

> Bir bütün olarak zihinsel emek kategorisi altında entelektüel enerjilerin sürekli olarak yaratıcı bir yersiz yurtsuzlaşmaya tabi olduğu bilişsel emek ile bugün halen nicel olarak hâkim konumdaki esasen uygulamaya dönük olan zihinsel emek arasında düzgün bir ayrım yapmamız gerekir.

Bu ayrım, tek bir yazılımcının üzerinde çalıştığı işlerden yola çıkılarak bile yapılabilir. Bir projede yeni teknolojilerle çalışırken diğer projede yıllar öncesinin teknolojisini kullanarak kod geliştiren yazılımcı, zihinsel emeğin yukarıda bahsi geçen farklı yakalarında çalışır. Eski teknoloji yıllardır kullanıldığı ve yazılım süreçlerine dair çok fazla deneyim biriktirildiğinden, yazılımcının çözmesi gereken sorunların çoğu kaydedilmiştir. Yazılımcıya kalan, bu çözümleri "uygulamaktır".

Ama bu kadar deneyimin biriktirilmediği dillerde yazılımcının çözümü "icat" etmesi gerekir. Savul'un ifadeleriyle (2018: 332):

> BİT sektörünün gayri-maddi üretim süreçleri ayrı ayrı ele alınarak değerlendirilirse, sektördeki gayri-maddi emekçilerin vasıfsızlaşma aracılığıyla değil, sürekli vasıf geliştirme baskısı aracılığıyla sermaye tarafından denetim altına alınmaya çalışıldığı ifade edilebilir.

Agile ile işleri küçük parçalara bölünen yazılımcının niteliği, yazılımcının üzerinde çalıştığı teknolojiyle beraber düşünüldüğünde anlam kazanır. Proje yönetim sürecinin mekanizasyonu ve projenin içeriğinin etkisinin bileşkesi, yazılımcının emeğinin niteliği üzerinde etki yapar.

"Bifo" Berardi'nin uyarılarına rağmen gayri maddi emek literatürü zihinsel emeğin bütün bileşenlerini homojenmiş gibi kabul etmeye meyillidir. Bunun en önemli nedeni, gayri maddi emek tartışmasında gayri maddi emek biçimini tanımlamak için kullanılan verilerin kaynağı sorunudur. Gayri maddi emek tartışmalarında vurgulanan birçok faktörün bizatihi yazılım sektöründe geçerli olup olmadığının sorgulanması için üretimin yapılış biçiminin ele alınması gerekir. Marx'ın üretim ilişkileri tarifi, biçimsel boyunduruk ve gerçek boyunduruk ayrımı yapılan işten yola çıkılarak oluşturulmuştur. İşin yapılış biçimi, işte aletlerin ve aygıtların kullanılış biçimi ve işin örgütlenme biçimi emek süreçlerini anlamada önemli faktörlerdir. İtalyan otonomist marksistler gayri maddi emek kavramının içini doldururken bu faktörlere üstün körü değinirler. Koşar'ın da altını çizdiği gibi (2017: 120) asıl olarak gayri maddi emek literatüründeki:

> … veri olarak kabul ettiği "işçi sınıfının önemini yitirdiği" ve "maddi olmayan emeğin belirleyici olduğu" tezi; "bilişsel kapitalizm" ve "maddi olmayan emek" paradigmasının "bilgi toplumu" fetişizmi ve yeterli ampirik çalışmayla birleşmeyen ideolojik/teknolojist ön kabullerine dayanır.

Bu üstünkörülük neoliberalizmin ürettiği bilgiden dolayı maluldür. Negri ve Hardt'ın bütün toplumu artı değer üretim sathı olarak kavramsallaştırmasıyla neoliberalizmin bütün toplumu sermaye sahibi olarak anlaması arasında benzerlik vardır. Bu benzerlik de toplumsala dair yürütülen bir kuramsallaştırma çabasında sınıfların yerinin olmadığıdır. Gayri maddi emeğin öne çıkan özellikleri ile iş ilanları arasındaki benzerlik de sorunludur. İşin kendisi ile işin ilanı arasındaki farklılık, yazılım sektöründe icra edilen emeği kuramsallaştırabilmek için değinilmesi gereken konulardan biridir. Burada da bizatihi yazılım sektöründeki üretim ilişkilerinin masaya yatırılması daha isabetli soyutlamaları mümkün kılabilir. Çünkü gayri maddi emek kavramı, aynı kazanda eritmeye çalıştığı birçok farklı emek biçiminin özgül yanlarını, yaptığı genellemenin neden olduğu şiddetle görünmez kılma tehlikesi taşır.[13] Otonomist marksistlerin kavramsallaştırdığı gayri maddi emeğin, ofisteki

13 Çağrı merkezi çalışanlarının emeği de yazılımcıların emeği de gayri maddi emek kapsamında değerlendirilir. Çağrı merkezi çalışanlarının emeği esas olarak zamanla ölçülür ve sıkı şekilde takip edilebilirken yazılımcının emeğinin takibi çok daha farklı yöntemlerle kontrol edilmeye çalışılır ve ölçülür.

üretim ilişkileriyle eleştirel bir okumaya tabi tutulması elzem görünmektedir.

Sonuç

Yazılım sektörü üretim araçlarının gelişimi sürecinin bir parçasıdır. Günümüzde toplumun kolektif ve kümülatif biçimde ürettiği genel zekânın üretim araçlarına dahil edilmesinin sıkça başvurulan yöntemlerinden biridir. Üretim araçlarının tarihsel gelişim yörüngesinin güncel duraklarından en önemlilerindendir.

Yazılımın genel kapitalist üretim tarzına etkisi göreli artı değerin arttırılabilmesini mümkün kılmasıyla ilişkilidir. Yazılım sayesinde iş süreçleri daha da verimlileştirilebilmiş ve üretilen değer ile işçinin emeği arasındaki ilişki daha güçlü bir biçimde koparılabilmiştir. Yazılımla geliştirilen yapay zekâ gibi araçlar, üretim ilişkilerinde işçinin elinde kalan son noktaları da temizlemiş, kritik anlarda karar vericiliği üstüne almıştır. Diğer taraftan yazılımın yarattığı denetleme mekanizması da işçinin daha çok çalışmasına neden olur. İşçinin bilgisayar dolayımıyla yaptığı her işlem gözlem altındadır. Üçüncü olarak yazılım emek gücü talebini azaltacak gelişmelerin önünü açar işsiz sayısını arttırır. İşsizlikteki bu artış işçilerin önüne sermayenin silahı olarak çıkar.

Yazılımcının içinde bulunduğu üretim ilişkileri belirli tahakküm mekanizmaları içerir. Yazılımcıdan mesai saatlerini raporlaması talep edilir, proje yönetim metotlarıyla yazılımcının her günü ve saatini dolu geçirmesinin ön hazırlığı yapılmaya çalışılır. Böylece yazılımcının teknik bilgisinin yarattığı güç alanı, zihinsel emeğin içsel iş bölümünün yaratılması, iş süreçlerinin yazılımcının denetiminden çıkarılarak belirli metotlara göre yapılması ve işin çok fazla parçaya ayrılarak bütünlüğünün yazılımcının zihninde canlanamamasıyla yok edilir. Öyle ki, yazılımcının zihinsel becerilerini geliştireceği, "inovasyon" kavramıyla popülerleşmiş teknik yaratıcılığı işin sınırlılığıyla zaten yok edilmiş olur. Az sayıda yazılımcı "creative" işlerde çalışır, çoğunluğu eski teknolojiyle yazılmış projeleri ayakta tutmaktan sorumludur.

Bu çalışmada yazılımcı emeğinin nasıl kavramsallaştırılacağını, kendine özgü yanlarının ve bu özgünlüklerin yarattığı üretim ilişkilerinin nasıl bir genellemeyle daha iyi anlaşılabilir kılacağına dair sağlam adımlar atamadık. Gayri maddi emek gibi, bu alanı kapsadığını iddia eden kuramlara itirazlarımızı dile getirdik. Yazılımcıların ve yazılım teknolojisinin tarihsel materyalizmle de anlaşılabileceğine dair bir tartışma başlatabilmeyi denedik. Asıl amacımız, neoliberalizmin hegemonik hale getirdiği düşünsel paradigmanın yarattığı "kuramsal gözlüğün" bir gözlük olduğunun farkında olarak yazılımcının zihinsel emeğinin Marx'ın yöntemiyle anlaşılabilir olduğunu göstermektir. Böylece kapitalizme dair bütün çelişkilerin, biçimsel farklılıklarla beraber yeni üretim araçlarının içine doğduğu ve sınırlı olarak dönüştürdüğü ilişkilerde de var olduğunu gösterebiliriz, hem Marx'ın altını çizdiği gibi üretim araçlarının gelişiminin zorunlu olarak kapitalist ilişkileri var etmediği, bu üretim araçlarının komünist bir örgütlenmede kapitalist olmayan üretim ilişkileri içinde de kullanışlı olabileceğini iddia edebiliriz.

Kaynakça

Acar, Ö. (2014) *Pratik Spring Core*, Pratik Programcı Yayınları.

ACM Yazılım Çözümleri. (2010) *Çevik Yazılım Geliştirme: "Agile". http://www.acm-software.com/Pdf/AboutAgile.pdf,* indirilme tarihi: 1 Eylül 2018.

Agile Manifestosu. (2001) *http://agilemanifesto.org/,* indirilme tarihi: 1 Eylül 2018.

Berardi, F. (2012) *Ruh İşbaşında: Yabancılaşmadan Otonomiye,* çev. F. Genç, İstanbul: Metis.

Berardi, F. (2014) *Gelecekten Sonra*, çev. Şişman, S. Özer. İstanbul: Otonom.

Chen, H. (2010) *Comparative Study of C, C++, C# and Java Programming Languages Information Technology*, Yayımlanmamış Yüksek Lisans Tezi, Vaasan Ammattikorkeakoulu University of Applied Sciences, University Of Applied Sciences, Vaasa. *opera.ucsd.edu/paper/log_icse12.pdf,* indirilme tarihi: 8 Eylül 2018.

Foucault, M. (2005) "Hakikat ve Hukuki Biçimler", Keskin, F. (der.), *Büyük Kapatılma* içinde*,* çev. I. Ergüden, İstanbul: Ayrıntı, 163-278.

Fuchs, C. (2015) *Dijital Emek ve Karl Marx*, çev. Oğuz, T. E. Kalaycı, İstanbul: NotaBene.

Glassman, J. (2017) "İlk Birikim, Mülksüzleştirerek Birikim ve "Ekonomi –dışı" Araçlarla Birikim", Adaman, F. ve B. Akbulut ve U. Kocagöz (der.), *Herkesin Herkes İçin: Müşterekler Üzerine Eleştirel Bir Antoloji* içinde, İstanbul: Metis, 80-106.

Hardt, M. Negri, A. (2004) *İmparatorluk*, çev. A. Yılmaz, İstanbul: Ayrıntı.

Harvey, D. (2010) *Postmodernliğin Durumu*, çev. S. Savran, İstanbul: Metis.

Koçar, M, Keleş, K. (2007) Otomasyon Sistemlerinin Bileşenleri, *IV Otomasyon Sempozyumu,* İstanbul, Elektrik Mühendisleri Odası. *http://www.emo.org.tr/ekler/8822633894235ea_ek.pdf,* indirilme tarihi: 8 Eylül 2018.

Koşar, A. (2018) "Maddi Olmayan Emeği Sahada Aramak", Koşar, A. (der.), *Maddi Olmayan Emek Teorisi: Kuramsal Bir Eleştiri* içinde, İstanbul: Kor, 115-136.

Lazzarato, M. (1996) 'Immaterial Labor', çev. P. Colilli and E. Emery, in M. Hardt and P. Virno (der.), *Radical Thought in Italy: A Potential Politics* içinde, Minneapolis and London: University of Minnesota Press, 133-147.

Marx, K. (2011) *Kapital I*, çev. M. Selik ve N. Satlıgan, İstanbul: Yordam.

Marx. K. (2014) *Grundrisse: Ekonomi Politiğin Eleştirisi için Ön Çalışma*, çev. S. Nişanyan, İstanbul: Birikim.

Narin, Ö. (2012) Teknoloji, Üretim Sürecinde Bölünme ve Emek, *Elektrik Mühendisleri Odası Dergisi*, 445: 40-45.

Narin, Ö. (2017) "*Kapital*'den Sonra Grundrisse: Marx ve Genel Zekâ", Öztürk, Ö. ve M. Yaman ve Ö. Narin (der.), *Grundrisse'den Kapital'e Patikalar* içinde, İstanbul: Sav, 347-409.

Read, J. (2014) *Sermayenin Mikropolitikası: Şimdiki Zamanın Tarih Öncesi ve Marx*, çev. A. D. Temiz, İstanbul: Metis.

Sankur, B. (2002) *İngilizce – Türkçe Ansiklopedik Bilişim Sözlüğü*, İstanbul: Pusula Yayıncılık.

Savul. G, (2017) "Bilişim ve İletişim Sektörü Emekçilerinin Sınıf Mücadelesi Kapasitesi Üzerine Değerlendirmeler", Kutlu, D. ve Ç. Kaderoğlu Bulut (der.), *Sınıfın Suretleri: Emek Süreçleri ve Karşı Hareketler* içinde, İstanbul: NotaBene, 187-198.

Savul, G. (2018) *Sınıfın Yeni Görünümleri ve Bilişim Sektörü*, İstanbul: NotaBene.

Schwartz, C. R. (2011) "Elektroniğin Toplumsal Gelişimi", Crowley, D. ve P. Heyer (der.), *İletişim Tarihi* içinde**,** çev. B. Ersöz, Ankara: Siyasal Kitabevi, 456-466.

Sommerville, I. (2009) *Agile Software Development*. *https://www.uio.no/.../sommerville.ch_03_agile_sw_dev.pdf,* indirilme tarihi: 10 Eylül 2018.

Terranova, T. (2015) Bedava Emek: Dijital Ekonomide Kültür Üretimi, çev. Bulut, A. Ateş. *Folklor/Edebiyat Dergisi*, 21(83).

Türkiye Bankalar Birliği (TBB) (2016) *Bankacılık Sektöründe Şube ve Personel Sayılarına İlişkin Bilgi Notu*. *https://www.tbb.org.tr/Content/Upload/.../TBB_sube_personel_bilgi_notu_100316.pd,* indirilme tarihi: 8 Eylül 2018.

Türkiye Bankalar Birliği (TBB) (2017) *Bankacılık Sisteminde Banka, Çalışan ve Şube Sayıları*. *https://www.tbb.org.tr/Content/.../Banka_Calisan_ve_Sube_Sayilari-Eylul_2017.pdf,* indirilme tarihi: 7 Eylül 2018.

Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) (2016) *Yeni Sanayi Devrimi: Akıllı Üretim Sistemleri Teknoloji Yol Haritası*. *http://www.tubitak.gov.tr/sites/default/files/akilli_uretim_sistemleri_tyh_v27aralik2012.pdf,* indirilme tarihi: 7 Eylül 2018.

Türkiye Bilişim Sanayicileri Derneği (TÜBİSAD), (2017) *Türkiye'de E-Ticaret: 2016 Pazar Büyüklüğü*, İstanbul: TÜBİSAD.

Türkiye Bilişim Sanayicileri Derneği (TÜBİSAD) (2018) *Bilgi ve İletişim Sektörü: 2017 Pazar Verileri*, İstanbul: TÜBİSAD.

Türkiye Sanayici ve İşinsanları Derneği (TUSİAD) (2016) *Türkiye'nin Küresel Rekabetçiliği için Bir Gereklilik Olarak Sanayi 4.0: Gelişmekte Olan Ekonomi Perspektifi*, İstanbul: TUSİAD.

Weizenbaum, J. (1995) *Bilgisayar Gücü ve İnsan Aklı: Yargıdan Hesaplamaya*, çev. T. Yöney. İstanbul: Sarmal Yayınevi.

Yapı Kredi Bankası. (2017) *2017 Makro ve Bankacılık Sektörü Beklentileri: Süregelen Dalgalanmaya Rağmen Temel Bankacılık Göstergelerinin Korunması*, *https://www.kap.org.tr/tr/ek-indir/4028328d5988e2630159ff44460c170c,* indirilme tarihi: 7 Eylül 2018.

Yarmalı, E. S. (1995) *Bilgisayar Terimler Sözlüğü*, İstanbul: Birsen Yayınevi.

Yıldırım, T. Ö. Türkcan, M. (2017) "Teknokentte Sınıf İçi İletişim", Kutlu, D. ve Ç. Kaderoğlu Bulut (der.), *Sınıfın Suretleri: Emek Süreçleri ve Karşı Hareketler* içinde, İstanbul:NotaBene, 187-198.

Yuan, D. vd. (2012) "Characterizing Logging Practices in Open-source Software," *34th International Conference on Software Engineering (ICSE)*, Zurich, http://opera.ucsd.edu/paper/log_icse12.pdf, indirilme tarihi:7 Eylül 2018.