Çevik ve Emniyet Kritik: Demiryolu Sistemlerinde Çevik Yaklaşımların Uygulanabilirliğinin Araştırılması

Banu Deniz YANIK, Hacı Ali ŞAHİN, Abdul Kerim AYDIN, Serkan CEYLAN

Aselsan A.Ş. Ulaştırma Güvenlik Enerji Sistemleri Sektörü,
Yazılım Mühendisliği Müdürlüğü Macunköy / ANKARA

bdyanik@aselsan.com.tr, hasahin@aselsan.com.tr,
akaydin@aselsan.com.tr, ceylan@aselsan.com.tr

Özet. Yazılım geliştirme sürecinin savunma sanayiinden başlayıp, diğer tüm endüstriyel alanlara hızlı bir şekilde yayılması ve buna paralel olacak şekilde artan talep ve hızlı değişen sosyal ve ekonomik dinamiklerin de etkisiyle; yazılım geliştirme yöntemleri yeni yaklaşımlar ve teknikler içerecek şekilde sürekli olarak gelişmektedir. İşte bu gelişimden doğan ve dünyada pazara hızlı cevap verebilmesi, sadeliği ve motive edici etkileri vs. sayesinde ön plana çıkan çevik yaklaşımlar oldukça tercih edilen bir yöntem seti olarak göze çarpmaktadır. Bu makalede ise çevik yaklaşımların altında yer alan teknikler ve bunların emniyet kritik demiryolu sistemlerinde kullanılmasının tartışılması sağlanacaktır. Bu kapsamda EN 50128 standardının gerekleri de gözetilerek çevik yaklaşımlar değerlendirilecek; eksik kalan ve örtüşen noktalara değinilecektir. Sonuç olarak ise bu iki yöntemin birlikte çalışabilirliği ve yöntemler özetlenecektir.

Anahtar Kelimeler: Çevik Yaklaşımlar, Emniyet Kritik, Demiryolu Sistemleri, Raylı Araçlar, V-Model, EN-50128

Abstract. Software development processes created by defense industries effects all industrial areas in time. Because of increasing demands and changing social and economic dynamics, software development methods was improved and started to consist of new approaches and techniques. Due to the improvements and motivated reactions to modern marketing systems, agile systems are one of the most popular preferred methods. In this article, it is planning to explain techniques introduced by agile practices and usages of these techniques in safety integrated railway systems. In this scope, we plan to evaluate agile practices considering EN 50128 standards and lists the missing and overlapping parts. As a conclusion, methods and combination of these methods are briefly explained.

Keywords: Railway Systems, Safety Critical, CMMI, EN-50128, Process Improvement, Standard Compliance

1 Giriş

Çevik pratikler, günümüzde oldukça tercih edilen bir yazılım geliştirme disiplini olarak karşımıza çıkmaktadır. Market ihtiyaçlarına hızlı cevap verebilme, ekip ruhunu güçlü tutma, iletişim ve insani yönleri vurgulayan bu disiplin [1];

- Süreçler ve araçlardan ziyade bireyler ve etkileşimlere,
- Kapsamlı dökümantasyondan ziyade çalışan yazılıma,
- Sözleşme pazarlıklarından ziyade müşteri ile işbirliğine ve
- Bir plana bağlı kalmaktan ziyade değişime karşılık vermeye değer vermektedir.

Ancak yıllar içinde her akım ve yenilikte olduğu gibi, aranılan gümüş kurşunun [2] her koşulda çevik pratikler olmadığı gözlemlenmiştir. Bunun yerine melez [3] yöntemler, çevik uygulamaların daha plan yönelimli disiplinlerle [4] bileşimi; CMMI [5], ISO 15504 [6] gibi daha formal tanımlı süreç iyileştirme referans modelleriyle harmanlanması söz konusu olmaya başlamıştır. Bu sayede hem plan yönelimli uygulamaların disiplinli yapısı, hem de çevik pratiklerin hızlı ve pratik perspektifinin zenginliklerinden aynı anda yararlanılması söz konusu olmaktadır.

Bu çalışmada ise demiryolu sistemleri emniyet kritik projelerde aranan temel standart olan EN 50128 ile çevik pratiklerin birlikte uygulanabilirliği, eksiklikler ve önerilerden bahsedilecektir. Dokümanın bir sonraki bölümünde literatür taraması ve temel bilgilerden; devamında ise bu iki farklı yaklaşımın birlikte uygulanabilirliği ve izlenebilirliğinin sağlandığı çalışmadan bahsedilecektir. Sonuçlar kısmında ise çevik pratikler ve emniyet kritik sistemlerin melezlenmesinin sonuçları ve etki analizi aktarılacaktır.

2 Benzer Çalışmalar ve Temel Bilgiler

2.1 Literatür Taraması

Çevik pratikler ve emniyet kritik sistemlerin birlikte çalışabilirliğine yönelik olarak yapılan bir dizi çalışma bulunmaktadır. Bunların farklı perspektifleri ve amaçları olsa da, çevik pratikler ve emniyet kritik sistemlerin bütünleşmesi için bazı karşılaştırma ve önerileri içermektedir.

[7] İlgili çalışmada, gömülü yazılımlarda emniyet kritik sistemlerin çevik pratiklerle geliştirilmesi için bazı araç destekleri ve özellikleri aktarılmıştır. Çözüm

odaklılığıyla dikkat çeken bu çalışmada toplam 12 adet tanımlanmış olan [8] çevik pratiklerden; artırımlı yinelemeli geliştirme, test yönelimli geliştirme, sürekli entegrasyon, planlama kısmına değinilmiştir ancak diğer pratiklerden bahsedilmemiştir.

Scrum ekolü ve EN 50128'in kıyaslanması ve uyumluğunu araştıran makalelere bakıldığında ise [9][10] bazı çalışmalar öne çıkmaktadır. Bu çalışmalar çevik pratiklerin temelini aldığı uç programlamadan[1][8] ziyade Scrum pratikleri ve uygulama tecrübelerini aktarmışlardır.

[11] Bahsi geçen çalışmada teker teker süreçlerin çevik ve yalın prensipler açısından incelenmesi ve gerekli değişiklikler adreslenmiştir. Bu çalışmada 12 çevik pratik yerine çevik ve yalın organizasyonların genel yapısı göz önüne alınarak bir değerlendirme yapılmıştır.

12 çevik pratiğin EN 50128'de uygulanabilirliğini doğrudan değerlendiren bu makalede ise [12] EN 50128 ile izlenebilirlikler kurulmamış ve bir çözüm önerisine ve geliştirme modeline gidilmemiştir.

2.2 Çevik Pratikler ve Tarihçesi

Yazılım projelerinde doksanlı yıllarda artan başarısızlıkların ardından [13] farklı geliştirme ve yaklaşımlar aranmaya başlamıştır. İşte uç programlama da, bu arayışlar neticesinde çıkan bir yaklaşımdır. Bu yaklaşım, güçlü iletişim, geri bildirimin önemi, basitlik ve cesaret değer yargılarını göz önüne alan, ağır dokümantasyondan imtina eden ve onu iyi iletişim kuran bir ekip, güçlü teknik altyapı ve sürecin kendisine odaklanan bir bakış açısıyla harmanlayan bir yapıdadır.

Uç programlama bahsi geçen değer yargılarını korumak ve onları sürdürmeye yönelik olarak bir dizi çekirdek çevik pratikten oluşmaktadır. Bir organizasyonun ve/veya projenin ne kadar çevik olduğu [4] bu pratikleri gerçekleştirme oranıyla ölçülmektedir. Bu bölümün devamında ilgili pratikler kategorilerine göre açıklanacaktır.

Bu kategoriler altında yer alan pratikler, biri diğerinden üstün olmamakla birlikte genel olarak aynı felsefe etrafında bütünleşen ve birbirini tamamlayan özelliktedir.

---

[1] İngilizcesi: eXtreme Programming (XP).

Kodlama Pratikleri.

1.Metafor (Ortak Bir Anlayış ve Yapı) Oluştur.

Çevik yaklaşımlarda ekibin aynı dili konuşması, aynı ortak anlayışta buluşabilmesi oldukça önemli bir pratiktir. Metaforlar yani benzeşimler aracılığıyla projenin, proje hedefleri ve tasarımının ifade edilebilmesi, üst düzey mimarinin herkes tarafından aynı şekilde anlaşılmasını sağlamaktadır.

2.Basit Tasarım ve Kodlama.

Tasarım ve kodlama aşamasında ihtiyaç duyulan kadarın tasarlanması, ihtiyaç duyulan kadarın kodlanması pratikleri sistemin küçük parçalara bölünüp daha kolay yönetilmesi ve ekibin sonuçlara daha hızlı bir şekilde ulaşmasını sağlar.

Müşterinin her iterasyonda öncelikli problemlerine odaklanmak ve bunları çözüme ulaştırmak bu pratikte temel olarak benimsenmiştir.

3.Yeniden Yapılandır.[2]

Yeniden yapılandırma, kodun sürekli iyileştirilmesi olarak da tanımlanabilir. Kodun yazılmasının ardından daha anlaşılabilir olması, daha iyi bir mimaride anlam kazanması, hatalardan ayıklanması, performans artırılması gibi sebeplerden kod yeniden yapılandırılır. Bunun sürekli göz önüne alınması ve uygulanması ise bu pratiği anlamlı hale getirir.

4.Kodlama Standartları Oluştur.

Kodlamanın ekipçe belirlenmiş standartlar çerçevesinde geliştirilmesi, kaliteli ve anlaşılır kod yazmanın temelini oluşturduğu gibi, uç programlama altında yer alan çevik pratiklerden de biri olarak karşımıza çıkmaktadır. Ekibin aynı dili konuşabilmesi ve hata oranı düşük ürünler geliştirebilmesi için kullanılmaktadır.

---

[2] İngilizcesi: Refactoring.

Geliştirme Yaklaşımı Pratikleri.

1.Test Yap.

Test yönelimli yaklaşımı da içine alan bu pratikte amaç yazılan her işlev için gerekli testi de beraberinde tasarlamak, ilgili kaynak kodu yazmak ve onu çalışır hale getirmektir. Böylece yazılan her işlev çevik bir şekilde sınanabilmektedir. Test yönelimli yaklaşımlarda ise, işlevin yazılmasını daha beklemeden ilk olarak testi yazılır. Böylece hedef, testin geçmesini sağlayacak işlevleri gerçekleştirmek olarak yorumlandığı halinde de test odaklı olarak geliştirme yapılmaktadır.

2.Eş Programlamayı Kullan.[3]

Eş programlamada iki kişi aynı işlev setini kodlamak üzere birleşirler. Böylece doğan sinerji ve odaklılık sayesinde yazılan kodun kalitesi artar ve kolektif yaklaşım, ortak dilin oluşumu gibi felsefeler desteklenmiş olur.

3.Kolektif Sahiplenmeyi Sağla.

Bu pratikle hedeflenen ürüne değer katmak isteyen herkesi projenin içine çekebilmektir. Kodu belirli parçalara bölerek sahiplendirmektense bütününü herkesin sahiplenmesini, değiştirebilmesini ve anlayabilmesini sağlamak esastır.

Bu pratiğin amacı, kod üzerinde bir değişiklik yapmak gerektiğinde bağımlılığı minimuma indirmek, gereksiz beklemeleri ortadan kaldırmak ve projeyi benimsemenin ekip içinde kolaylaşmasını sağlamaktır.

4.Sürekli Entegrasyonu Sağla.

Ürünün sahip olduğu iç ve dış arayüzleriyle birlikte erken aşamalarda entegre edilmesi ve sınanması ileride çıkabilecek potansiyel hataları önceden görmemizi sağlar.

Bunun için kodun çalıştırılması, ilişkili birim ve entegrasyon testlerinin yapılması, statik kod analizlerinin gerçekleştirilmesi hatta son kullanıcı testlerinin otomatikleştirilmesini kapsayan yapılar kurularak sistemin gün içinde birden

---

[3] İngilizcesi: Pair Programming.

çok kere paketlenip hazır hale getirilmesi aktiviteleri bu pratikle gerçekleştirilir.

Yönetimsel Pratikler.

1.Planlama Oyununa Katıl.

Planlama oyununda paydaşların birleşik olarak bir sonraki adımda yapacağı işleri planlaması faaliyetleri düşünülmelidir. Ürün gözüyle ihtiyaç duyulan yetenekler kullanıcı hikayesi [14] formatında ifade edilirken, buna istinaden takım geliştirme hızı dikkate alınarak ne kadarının yapılabileceği geliştirme takımı tarafından belirlenir. Bu planlamalar ise sabit zaman kutularının içine sığacak işlerin belirlenmesi şeklinde yapılır.

Planlamalar sürüm odaklı ve bir sonraki iterasyon odaklı olarak ayrı ayrı yapılabilir. Burada planlamanın bütünsel olarak değil, projenin işlevsel parçalarına yönelik olarak yapılması beklenir. Bu sayede belirsiz olan unsurlar hakkında yanıltıcı planlar yapılmaktan kaçınılır.

2. Küçük İterasyonlar Yap.

Küçük iterasyonlarla müşteriye ihtiyaç duyduğu değerleri artırımlı parçalar hainde vermek, müşterinin değişiklikleri daha iyi takip etmesine, erken müdahale ederek yanlış anlaşılmış gereksinimleri düzeltmesine, doğru ve hızlı geribildirim verebilmesine hizmet eder.

Artırımlı yinelemeli [15] olarak da anılan bu geliştirme yaklaşımı, riski minimize etme, projeyi daha kolay yönetilebilir hale getirme, gerçekçi plan yapabilmek için gerekli ortamı hazırlamak gibi hedefleri gerçekleştirmek için tercih edilir.

3.Müşteriyi Sürece Dahil Et.

Bu pratiğin en ideal formunda müşterinin geliştirme yapılan ortama dahil edilmesi veya müşterinin çalışma ortamında geliştirme faaliyetlerinin yapılması hedeflenmektedir. Ancak bu durum sağlanamadığında bile çok kolay erişilebilir, adanmış ve tanımlı işi yazılım geliştirme ekibine destek vermek olan müşteri profili bu çevik pratiğin sağlanması için gereklidir.

Bu pratikle hedeflenen müşteriyle güçlü bir iletişim kurarak, onu proje aşamalarına dahil etmek ve böylece sonradan çıkabilecek sürpriz değişiklikler, yanlış anlamalar ve gereksiz dokümantasyonun önüne geçmek, müşterinin erken

aşamalarda projeye bağlanması ve sahiplenmesini sağlayarak projenin başarı şansını artırmaktır.

4.Haftada 40 Saati Kullan.

Çevik bir ekibin verimli çalışma saatlerinin olmasının yanında, uzayan toplantılar ve fazla mesailerin azaltılması da önemli pratiklerdendir. Böylece ekibin motivasyonunun belirli bir seviyenin altına inmemesi sağlanmaktadır. Bunun için gerekli durumlarda ek zaman alanları oluşturarak üst düzey planlamaları yönetmek önerilen tavsiyelerdendir.

İnsanın doğası gereği fazla mesailerin artmasıyla birlikte yoğunlaşma ve motivasyon kaybı yaşanacağından dolayı üretilen ürünün kalitesinde düşme, uzun vadede süreçlerde aksama meydana gelebileceği gözden kaçırılmamalıdır.

2.3 Emniyet Kritik Sistemler ve EN 50128

Raylı araçlara ait uygulamalarda emniyet kritik geliştirme, yükleme ve bakım işlemlerini düzenlemek üzere EN 50128 Avrupa standart ailesi [16] kullanılmaktadır. Bu standartla uyumlu olabilecek kurumun sahip olması gereken organizasyon yapısı, kullanması gereken yaşam döngüsü ve uygulaması gereken planlama, tasarım, geliştirme ve test faaliyetleri ile kalite güvence faaliyetleri Yazılım Güvenlik Bütünlüğü Seviyelerine[4] (YGBS) göre sınıflandırılmış olarak bu standartlarda açıklanmıştır. YGBS'ler bölümün devamında detaylandırılacaktır.

Risk Analizi.

Emniyet kritik sistemlerde sistemin kullanıldığı ortamda sistem nedeniyle oluşabilecek riskler tanımlanır. Bu riskler:

- İnsan yaşamında kayıp(lar)
- Yaralanma ya da hastalık durumları
- Çevre kirliliği ve
- Taşınır-taşınmaz varlıklarda kayıp

sonuçlarına göre analiz edilerek belirlenir. İşte bu risklerin etkilerine göre belirlenecek kısım da YGBS'ler olarak anılır. Kısaca aşağıdaki tablo YBGS'leri ve anlamlarını açıklamaktadır.

---

[4] İngilizcesi: Software Safety Integrity Level (SIL).

Tablo 1. Yazılım Emniyet Bütünlük Seviyeleri ve Emniyet Kritiklikleri

Yazılım Güvenlik Bütünlüğü Seviyesi (YGBS)	Emniyetle İlişki Durumu
4	Çok Yüksek
3	Yüksek
2	Orta
1	Az
0	İlişkili Değil

Sistem ve alt parçalarına ait YGBS'ler ise EN 50128 standartlarına göre gereksinim analizi aşamasında belirlenir.

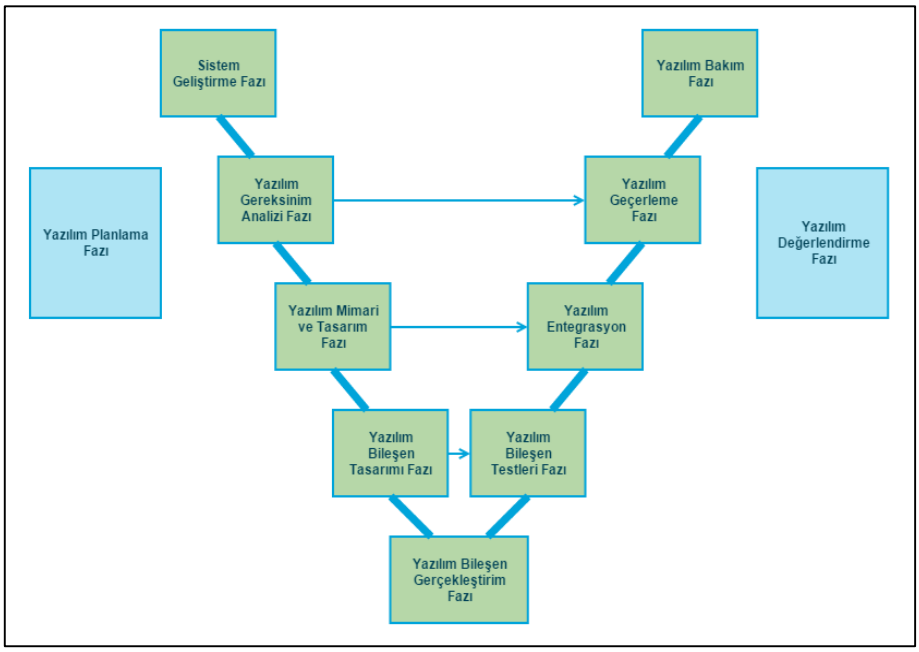
Organizasyonel Rol ve Sorumluluklar.

Risk analiz yöntemlerinin yanı sıra, EN 50128'le birlikte organizasyonel anlamda bazı rollerin kesin sınırlarla ve belirli kişileri bağımsızlaştırarak tanımlanması gereklidir. YGBS emniyet kritikliği arttıkça, rollerin bağımsızlığı da artmaktadır.

Burada bağımsızlıkla amaçlanan, özellikle değerlendirme, denetim ve test faaliyetlerinin proje yöneticisi ve proje ekibinden ayrı tarafsız bir gözle değerlendirilmesinin sağlanmasıdır. Başka bir deyişle kalite yönetim sistemi ve geliştirme ekibinin birbirini etkilemeden çalışabilmesinin sağlanmasıdır.

Yazılım Yaşam Döngüsü ve Dokümantasyon.

Yaşam döngüleri incelendiğinde EN 50128 ile dikkat çeken nokta, her faaliyetin sonrasında ilgili bir değerlendirme adımının yer aldığıdır. Bu da aslında genel olarak V-model [17] olarak bilinen yazılım geliştirme yaşam döngüsüyle karşılanmaktadır.



Şekil 1. EN 50128 V-Model Yazılım Yaşam Döngüsü

Daha farklı yazılım yaşam döngülerinin kullanılmasına karşı çıkmayan standartta önemli görülen nokta, her faaliyetin ardından yapılması gereken doğrulama ve geçerleme adımlarının atlanmamasıdır.

3 Çevik Pratiklerle EN 50128'in Birlikte Kullanılabilirliği ve Önerilen Yöntemler

3.1 Çevik Pratiklerin EN 50128 ile Uyumluluk Durumu

Bu kısımda çevik pratiklerin EN 50128 gereksinimleriyle çelişme ve/veya onları karşılama durumu gözetilerek alt başlıklar halinde bir değerlendirme yapılacaktır. Bu 12 pratik sırasıyla aşağıda yer almaktadır.

Metafor (Ortak Bir Anlayış ve Yapı) Oluştur.

Bu pratiğin EN 50128 açısından desteklediği kısımlar mevcuttur. Ortak bir dil, ortak terim, ortak hedefler ve anlayışın oluşturulması standart üzerinde sıkça bahsedilen bir konu olarak göze çarpmaktadır.

Basit Tasarım ve Kodlama.

EN 50128'de birden çok bölüm altında bahsi geçen "*Projenin büyüklük ve karmaşıklığının dengelenmesi gereklidir.*" Felsefesini destekler nitelikte olan bu pratik sayesinde, karmaşık tasarım ve kodlama faaliyetlerini desteklemekte olan çevik pratiklerin, standart uyumluluğunu da desteklemekte olduğundan söz edilebilir.

Yeniden Yapılandır (Refactoring).

Yeniden yapılandırma sayesinde ihtiyaç duyulan kadar mimarinin tasarlanması, sonrasında kodu yeniden yapılandırarak ilerlemesi beklenir. Ancak bu pratiğin EN 50128'le doğrudan uygulanması pek olası değildir. Bunun en temel sebebi değişiklik yönetiminin çok daha formal ilerlemesi yüzünden sistemin genel olarak bu sürecin işleyişini yavaşlatan bir yanı olmasıdır.

Ek olarak, değişen her kod parçası için yeniden doğrulama ve yeniden geçerleme faaliyetlerinin tekrar edilmesi gereklidir. Tüm bunlar gözetilmeden yapılan yeniden yapılandırmalar EN 50128'e ait standart uyumluluğunu tehlikeye atabilir.

Kodlama Standartları Oluştur.

Kodlama standartlarının varlığı ve uyumluluğun sağlanması çevik pratiklerde olduğu kadar EN 50128 tarafından da aranan bir gereksinimdir. Her iki yaklaşımda da birebir karşılık bulmaktadır.

Test Yap.

Çevik yazılım geliştirme yaklaşımlarında test ve kod bir bütün olarak değerlendirilir ve geliştirme yapan kişinin ilgili test betiklerini de yazması beklenir. Bu kapsamda da testi oluşturan kişi ve kodu yazan kişinin aynı olması kaçınılmaz bir sonuç olabilmektedir. Ancak bu sonuç, EN 50128 standartlarına göre farklı ele alınması gereken bazı noktaları açığa çıkarır.

EN 50128'e göre testçilerin geliştirme ekibinden bağımsız olarak çalışması ve ürünün doğruluğunu sınaması beklenir. Bu durumda bu çevik pratiğin doğrudan karşılandığı söylenemez. Bu durum bir çelişki oluşturmaktadır. Bir öneri olarak testlere ait prosedür ve gereksinimlerin testçiler tarafından yazılması

ancak bu yazılan gereksinimlere ait test betiklerinin kodlanması işin yine geliştiriciler tarafından yapılması önerilebilir.

Eş Programlamayı (Pair Programming) Kullan.

Eş programlama sayesinde kod kalitesinin artacağı, ortak dişin oluşmasına katkı sağlanacağı daha önceki bölümlerde aktarılmıştır. Bu sayede bazı gözden geçirmelerin de dolaylı olarak yapıldığı söylenebilir ancak bu durumda dahi üzerine yapılması gereken formal gözden geçirmeler tamamlanmış sayılmayacaktır ve bunların da planlanması ve yönetilmesi ayrıca gereklidir.

Kolektif Sahiplenmeyi Sağla.

EN 50128 açısından bir çelişki ya da ilişki içermemektedir.

Sürekli Entegrasyonu Sağla.

Entegrasyon EN 50128 için oldukça önemli bir aktivite başlığı olmaktadır. Hem donanımsal hem de yazılımsal bütünlüğün sağlanması ve sınanması faaliyetleri Entegratör tarafından yürütülmelidir. Ancak standartta entegrasyonun önemi vurgulanmış olsa da sürekli entegrasyon kavramından çok da söz edilmemiştir.

Bir öneri olarak, EN 50128'de direk tarifli edilmiş olmasa bile, sürekli entegrasyon konularının ele alınması işi Entegratör üzerinden yapılabilir. Bu durumda Entegratör'ün konfigürasyon yönetim işlerinde de görevlerinin olması gerektiği göz ardı edilmemelidir.

Ayrıca EN 50128'in gereği olarak sürekli entegrasyon için kullanılan araçların emniyet kritiklik açısından değerlendirilmesi, sınıflandırılması ve ilgili YBS'ye göre yönetilmesi [16][5] gereklidir. Bununla birlikte, bu araçlar üzerinde koşacak olan test, derleme, yükleme ve diğer yardımcı betiklerin de sahip olması gereken bir dizi emniyet kritik gereksinim ve özellik bulunmaktadır.

Planlama Oyunu.

EN 50128 kapsamında farklı boyutlarda ve çok sayıda planlama yapıldığı söylenebilir. Bu standart, iterasyonlara yönelik olarak planlama yapılmasına karşı

[5] 6.7 Destek Araçları ve Dilleri başlığı incelenebilir.

çıkmamakla birlikte, dokümantasyon ihtiyacının çevik süreçlere göre fazla olması sebebiyle ek maliyet getireceği gözden kaçırılmamalıdır.

EN 50128 kapsamında, kalite güvence, konfigürasyon yönetimi, doğrulama, geçerleme, değerlendirme ve bakım planlarının dokümante edilmesi beklenmektedir ve bu dokümanların ihtiyaca yönelik olarak birlikte oluşturulmasında bir sakınca yoktur. Bu noktada bakıldığında işlevsel olmayan gereksinimler için beklenen planlama yaklaşımının yanında, esas işin planlanmasına yönelik kısmın standart kapsamında oldukça esnek bırakıldığını söyleyebiliriz. Bu da planlama oyunu pratiğinin gerçekleştirilmesinde bir çelişki oluşturmayacağı anlamına gelmektedir.

Küçük İterasyonlar Yap.

EN 50128'le küçük artırımlar üretme pratiği doğrudan bir çelişki oluşturmamaktadır ancak bir artırım oluşturmak, çevik süreçlere göre daha maliyetlidir. Doğrulama, geçerleme ve risk değerlendirme ve denetleme faaliyetleri bu iterasyonlardaki ek işleri artırmaktadır.

Diğer yandan bu faaliyetleri daha sık yapmış olmanın verdiği avantaj sayesinde kodun daha sağlam temeller üzerinde yükseldiğinden de bahsedilebilir.

Müşteriyi Sürece Dahil Et.

EN 50128'de direk olarak müşteri odaklı bir gereksinim adreslenmemiş olsa da, geliştirme ekibinin yakınında bir müşterinin olması geçerleme faaliyetlerini olumlu etkileyecek ve destek olacaktır.

Haftada 40 Saati Kullan.

EN 50128 açısından bir çelişki ya da ilişki içermemektedir.

3.2 Çevik Pratiklerin EN 50128 ile İzlenebilirliği

Bu kısımda çevik pratikler ve onların ilişkili olduğu EN 50128 gereksinimleri maddeler halinde özetlenecektir.

Tablo 2. Çevik Pratikler ve EN 50128 Gereksinim İzlenebilirlikleri

Çevik Pratik Adı	**İlgili EN 50128 Gereksinim Numaraları**
Metafor Oluştur	*5.3 Yaşam Döngüsü ve Dokümantasyon* başlığı altında *5.3.26, 7, 8, 9, 10* gereksinimleri ortak anlayış ve ortak bir dil oluşturmanın ekip için gerekliliğini vurgulamaktadır.
Basit Tasarım ve Kodlama	*7.3 Mimari* başlığı altında *7.3.4.3* ve *7.3.4.15* maddeleri doğrudan açık ve anlaşılır bir mimariye duyulan gereksinimi özetlemektedir. *7.4 Bileşen Tasarımı* altında *7.4.4.5* maddesiyle bileşenlerin seviyesinde ve *7.5 Bileşen Gerçekleştirim ve Test* başlığı altında *7.5.4.2* ile gerçekleştirim seviyesinde açık ve anlaşılır mimari ihtiyacı özetlenmiştir.
Yeniden Yapılandır	*6.6 Düzeltme ve Değişiklik Kontrolü* başlığı altında hangi koşullarda ve ne şekilde yazılım üzerinde değişiklik yapılabileceği tariflenmiştir. Burada yer alan sürecin ağır dokümantasyon ve bir dizi kontrol makamlarınca onaylanarak değişikliklerin varlık bulması yeniden yapılandırma çevik pratiğini yavaşlatmakta ve hatta çoğu zaman engeller durumda olmaktadır.
Kodlama Standartları Oluştur	*7.3 Mimari* başlığı altında yer alan *7.3.4.25, 26, 27, 28* direk kodlama standartlarının varlığını ve sahip olması gereken prensipleri açıklamaktadır.
Test Yap	*6.1 Yazılım Testleri* başlığı altında testler ve beklenen özellikleri genel olarak aktarılmıştır.
Eş Programlamayı Kullan	İzlenebilirlik kurulamamıştır.
Kolektif Sahiplenmeyi Sağla	İzlenebilirlik kurulamamıştır.
Sürekli Entegrasyonu Sağla	*7.6 Entegrasyon* başlığı altında yer alan gereksinimler doğrudan sürekli entegrasyon aracılığıyla kolaylaşacak gereksinimlerdir.
Planlama Oyunu	*6.2 Yazılım Doğrulama* altında *6.2.4.2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ve 12* nolu gereksinimler; *6.3 Yazılım Geçerleme* altında *6.3.4.2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 ve 13* nolu gereksinimler; *6.4*

	Yazılım Değerlendirme altında *6.4.4.4 ve 5* nolu gereksinimler; *6.5 Yazılım Kalite Güvence* altında *6.5.4.3, 4, 5, 6, 7 ve 8* gereksinimleri planlamayla direk ilişkili gereksinimlerdir.
Küçük İterasyonlar Yap	İzlenebilirlik kurulamamıştır.
Müşteriyi Sürece Dahil Et	İzlenebilirlik kurulamamıştır.
Haftada 40 Saati Kullan	İzlenebilirlik kurulamamıştır.

Çevik pratikler ve emniyet kritik sistemlerin geliştirme yaklaşımı dikkate alındığında birbirini bütünleyen, birbiriyle ortaklaştırılabilecek ve prensipte birbiriyle çelişki oluşturabilecek aktivitelerin yer aldığı gözlemlenebilir.

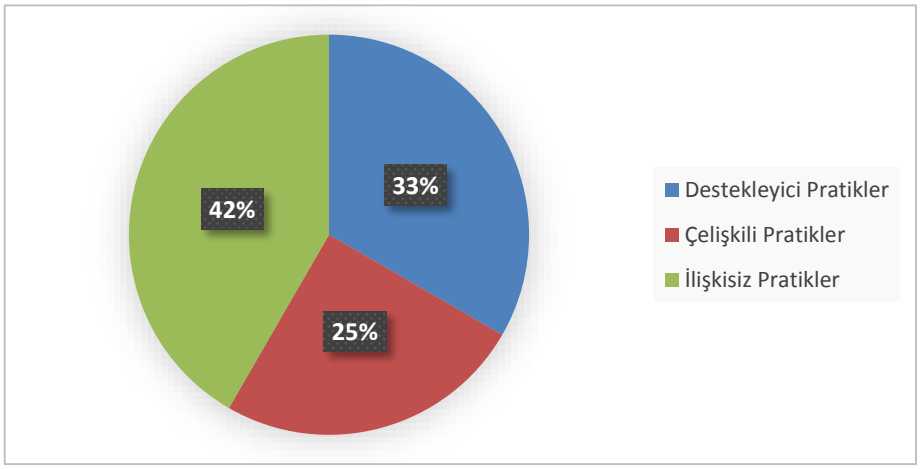
4 Sonuçlar

Çevik yazılım geliştirme pratikleriyle EN 50128 standartları temel anlamda birlikte çalışabilir yaklaşımlar olarak (Bakınız Tablo 3) karşımıza çıkmaktadır. Aşağıda çalışmanın sonucu olarak EN 50128'i destekleyen, çelişki içeren ve ilişkisiz pratikler ayrı ayrı listelenmiştir.

Tablo 3. EN 50128 ile Destekler, Çelişkili ve İlişkisiz Çevik Pratikler

	EN 50128 ile		
	Destekleyici	**Çelişkili**	**İlişkisiz**
Metafor Oluştur	X	-	-
Basit Tasarım ve Kodlama	X	-	-
Yeniden Yapılandır	-	X	-
Kodlama Standartları Oluştur	X		-
Test Yap	-	X	-
Eş Programlamayı Kullan	-	-	X
Kolektif Sahiplenmeyi Sağla			X
Sürekli Entegrasyonu Sağla	X	-	-
Planlama Oyunu	-	X	-
Küçük İterasyonlar Yap	-	-	X
Müşteriyi Sürece Dahil Et	-	-	X
Haftada 40 Saati Kullan	-	-	X

Burada atlanmaması gereken bir nokta ilişkisiz pratiklerin genel olarak ekip motivasyonunu artıran, kod kalitesini artıran, ekip içindeki kişi bağımlılıklarını azaltan, yönetilebilirliği artıran, riski minimize eden yaklaşımlar olması sebebiyle direk standartla ilişki kurulamasa bile, projenin başarısını doğrudan etkileyen pratikler olduğu göz ardı edilmemelidir.



Şekil 2. Çevik Pratiklerin Uyumluluk Dağılımı

5 Referanslar

[1] M. Fowler and J. Highsmith, "The agile manifesto," *Softw. Dev.*, vol. 9, pp. 28–35, 2001.

[2] F. Brooks, *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, 1982.

[3] B. Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*, vol. 22. 2003.

[4] B. Boehm, "Get Ready for Agle Methods, with Care," *Int. J. Eng. Sci. Technol.*, vol. 4, no. 1, pp. 23–29, 2012.

[5] CMMI, "CMMI for Development, Version 1.3," 2010.

[6] S. International Organization for Standardization, Geneva, "ISO/IEC 15504." p. Information technology – Process assessment, 2004.

[7] B. P. Douglass, "Adopting Agile Methods for Safety Critical Embedded Systems," no. October, 2012.

[8] K. Beck, *Extreme Programming Explained: Embrace Change*, no. c. 1999.

[9] T. Stålhane–IDI, “Safety standards and Scrum–A synopsis of three standards,” *Nbl.Sintef.No*.

[10] T. Myklebust, T. Stålhane, and N. Lyngby, “Application of an agile development processes for EN50128 / Railway conformant software,” no. 2012, pp. 4037–4044, 2015.

[11] M. Vuori, *Agile Development of Safety-Critical Software*. 2011.

[12] H. Jonsson, S. Larsson, and S. Punnekkat, “Agile practices in regulated railway software development,” *Proc. - 23rd IEEE Int. Symp. Softw. Reliab. Eng. Work. ISSREW 2012*, pp. 355–360, 2012.

[13] R. L. Glass, *Facts and Fallacies of Software Engineering*. 2002.

[14] M. Cohn, *User Stories Applied: For Agile Software Development (Addison Wesley Signature Series)*, vol. 1, no. 0. 2004.

[15] C. Larman and V. R. Basili, “Iterative and incremental developments. a brief history,” *Computer (Long. Beach. Calif).*, vol. 36, no. 6, pp. 47–56, 2003.

[16] B. Standard, “BS EN 50128:2011,” 2011.

[17] S. Balaji, “Waterfall vs v-model vs agile : A comparative study on SDLC,” *WATERFALL Vs V-MODEL Vs Agil. A Comp. STUDY SDLC*, vol. 2, no. 1, pp. 26–30, 2012.