Değişimi Normalleştirme: Normalleşme Süreci Kuramının Çevik Yazılım Geliştirmeye Uygulanması*

Normalising Change: Implementation of Normalisation Process Theory to Agile Software Development

Fatma Zehra KÖKER†

Makale Geliş Tarihi / Received : 13.09.2024
Makale Kabul Tarihi / Accepted : 30.12.2024
Araştırma Makalesi
Reserach Article

Öz

Sürekli değişime hızlı cevap veren yöntemlerden çevik yeni normal haline gelmiştir. Çeviğin normalleşmesi için günlük işlere rutin olarak yerleştirilmesi ve uygulanması gerekmektedir. Çalışmanın amacı çeviğin Yazılım Geliştirme uygulamalarına rutin olarak yerleştirilmesini ve uygulanmasını kolaylaştıran ve zorlaştıran faktörleri Normalleşme Süreci Kuramı ile tespit etmektir. Bu amaç doğrultusunda nitel araştırma yöntemlerinden çoklu durum çalışması modeli kullanılmıştır. Çeşitli yazılım geliştirme projelerinde çevikten yararlanan 5 çalışan ve yönetici ile yarı yapılandırılmış (tematik) görüşmeler yapılmıştır. Elde edilen verilere tematik içerik analizi ve karşılaştırmalı analiz uygulanmıştır. Sonuçlar çeviğin Yazılım Geliştirme uygulamalarında normalleşmesi için anlam oluşturmaya, katılıma, uygulama çabasına ve değerlendirmeye daha fazla yatırım yapılması gerektiğini göstermiştir. Çeviğin normalleşmesini kolaylaştıran başlıca faktörler: çevikten yararlanma nedenlerinin çalışanlara net bir şekilde iletilmesi, rollerin belirlenmesi, çevik bilgi-beceri eksikliklerinin giderilmesi ve çevik uygulamaların konfigürasyonu için çalışan değerlendirmelerinden yararlanılmasıdır. Tersine çeviğin normalleşmesini zorlaştıran faktörler arasında çalışanların ve yöneticilerin çeviği farklı şekillerde anlamlandırmaları ve çalışanların çevik uygulamalara dahil olmalarının meşruiyetinden şüphelenmeleri bulunmaktadır.

Anahtar Sözcükler: *Çevik, Yalın, Normalleşme Süreci Kuramı, Yazılım Geliştirme, Nitel Araştırma*.

Abstract

Agile is one of the methods that respond quickly to continuous change, has become the new normal. Normalisation requires agile to be embedded into daily work as a routine and to be implemented. The aim of the study is to identify the factors that facilitate and complicate the embedding, implementing and integrating agile into Software Development practices with the Normalisation Process Theory. For this purpose, a multiple case study model (a qualitative research method) was used. Semi-structured (thematic) interviews were conducted with 5 employees and managers who benefited from agile in various software development projects. Thematic content analysis and comparative analysis were applied to the obtained data. The results showed that more investment should be made in meaning creation, participation, implementation effort and evaluation in order for agile to become normal in Software Development practices. The main factors that facilitate the normalisation of agile are: clearly communicating the reasons for using agile to employees, determining roles, eliminating agile knowledge and skill deficiencies and using employee evaluations for the configuration of agile practices. Conversely, the factors that make it difficult to normalize agile include employees and managers interpreting agile in different ways and employees doubting the legitimacy of their involvement in agile practices.

Keywords: *Agile, Lean, Normalisation Process Theory, Software Development, Qualitative Research.*

*Bu çalışma Fatma Zehra Köker'in Prof. Dr. Recep Varçın danışmanlığında tamamladığı doktora tezinden üretilmiştir.

†Bilim Uzmanı, Ankara Üniversitesi, Sosyal Bilimler Enstitüsü, İnsan Kaynakları Yönetimi Anabilim Dalı, zehra.koker3@gmail.com

Önerilen Atıf Biçimi / Recommended Citation: Köker, F. Z. (2024). Değişimi Normalleştirme: Normalleşme Süreci Kuramının Çevik Yazılım Geliştirmeye Uygulanması. *Yönetim ve Çalışma Dergisi*. 8 (2), 241-266.

Extended Abstract

Within the scope of the research, it is aimed to identify the factors that facilitate and complicate the embedding, implementing and integrating agile into traditional Software Development with the NPT. Within the scope of the research purpose, in-depth interviews were conducted with five employees and managers working with both agile and traditional software development and management methods at least one year. A semi-structured thematic questionnaire was used during interviews. Thematic content analysis and comparative analysis were applied to the obtained data. Obtained data from interviews were coded, categorised and thematised through NPT lens. In the light of the results of the analysis the factors that facilitate and complicate the embedding, implementing and integrating of agile to traditional Software Development are shaped by investments in meaning, participation, implementation efforts and evaluation. The factors that facilitate and complicate the embedding, implementing and integrating of agile into traditional Software Devlopment are grouped under four themes; coherence, cognitive participation, collective action and reflexive monitoring.

Conceptual Framework

Normalisation Process Theory

NPT is an implementation theory that has proven useful in process evaluations. NPT provides a conceptual framework for understanding and evaluating the meaning-making, routine-embedding (operationalization), collective-implementation (normalisation) and integration of new norms into the broader context. NPT is particularly interested in the social organization of work (implementation), the routine-embedding of practices into everyday life (embedding) and the maintenance of embedded practices in their social contexts. The theory assumes that new technologies are embedded into the daily work of employees (routinely) as a result of individual and collective work. According to NPT, normalisation is achieved through the 4 constructs and 16 components of the theory. The first construct of NPT is ‘sense-making’, which is the establishment of organizational consensus about the complex intervention (new normal, norm). The second construct is ‘cognitive participation’, which is the operationalization of complex interventions. The third construct is the collective implementation of the technology or practice that is operationalized through ‘collective action’. The fourth structure, the ‘reflexive monitoring’ structure, evaluates the implementation by employees and reconfigures accordingly. Thus, the effectiveness of employees’ adoption of new technology or practice is continuously increased with NPT. It is recommended that NPT structures be applied iteratively to achieve the best technology implementation, deployment and integration results. Like other implementation theories, NPT has been criticized for being based on weak evidence-based practices. In addition, using NPT, which focuses on the social organization of work, means paying less attention to actor network structures (e.g. Actor Network Theory) or organizational psychology (e.g. Theory of Planned Behavior).

Agile Software Development and Normalisation

In cases where software requirements do not change significantly, the course of software projects is largely predictable. In this context, a well-planned and quality-focused software development process is adopted with traditional software development. In this process, which starts with great preliminary planning, requirements analysis, design, implementation, testing and maintenance activities are typically carried out in consecutive stages. However, this rigid software development model is resistant to changes in software requirements since it does not provide much flexibility to developers. On the other hand, responding to constantly changing software requirements in a dynamic change environment with this method is quite costly. Therefore, in cases where software requirements are largely uncertain or constantly changing,

software development is facilitated with agile/lean software development. In this context, with the principles and values of the Agile Manifesto, responding quickly to changing software requirements, cross-functional teams, close cooperation with customers who are the owners of software requirements, dividing problems into manageable small pieces and producing fast solutions with iterations and flexibility are encouraged. In addition to these, agile software development is facilitated with agile methods such as XP, Scrum, Kanban.

In terms of NPT, normalisation means the routinization and implementation of a complex intervention (or norm). In terms of NPT, the first step of normalisation begins with determining the differences between the two practices and is completed with the internalization of the change by the participants. The success of this stage is the determinant of the resistance to change in the next step. The second step of normalisation requires the design of the new practice brought by the intervention and organizing around the practice. For example, which agile/lean principles and methods will be adopted by whom, the tasks and responsibilities are determined in this step. The success of this stage largely depends on organizing around the new practice. The perceived legitimacy of the new practice among the participants increases the success of the next stage. In the third step of normalisation, collective action is taken with the new practice that was operationalized in the previous stage. The ability to take collective action with a new practice indicates that the practice has become normalized. In this context, skill requirements are clearly determined. The purpose of the fourth stage of normalisation is for the new practice to be evaluated and adjusted by the employees. The new practice is continuously improved from the bottom up. Because according to NPT, normalisation is not a permanent situation.

Purpose and Importance of the Research

The aim of the research is to determine the factors that facilitate and complicate the embedding, implementing and integrating agile into traditional software development applications *(normalisation)* with the Normalisation Process Theory. This study has three main contributions to practice and theory. First, the study demonstrates the novelty of applying NPT in normalizing agile software development practices. Second, this study will conduct empirical analyses on employees and managers who perform agile software development. Third, NPT is presented as a theoretical alternative to the 'digital transformation assessment' tools that have emerged in parallel with the increasing digital transformation efforts in the last five years.

Sample of the Research

Within the scope of the research, semi-structured interviews were conducted with 5 participants who could express their views on the subject objectively by using the purposive sampling method. In this context, the participants consist of two employees and three managers working in different software projects.

Data Collection Technique

In this research the data were conducted using the thematic semi-structured interview technique using a single analysis unit questionnaire.

Analysis of Data

Thematic content analysis and comparative analysis were used in the evaluation of the research data.

Findings and Discussion

Today, agile/lean methods that focus on geographically distributed high-speed virtual servers and rapid response to change stand out as two major digital transformation trends that complement each other. The agile way of thinking has become the new normal by being applied

to traditional ways of working during and after the pandemic. Agile methods are used in more than 90% of software development practices and provide well-known benefits such as responsiveness to changing requirements, higher software developer productivity, well-being, and autonomy. However, this has led to two important problems in agile software development practices. First, since there is a paradigm shift between agile thinking and Taylorist traditional management approaches, it is not easy to apply agile thinking to traditional management approaches. Naturally, many problems and resistance to change are encountered in this application process. Secondly, there is not yet a widely accepted common theory for the implementation, deployment and integration of new norms such as agile into traditional management approaches. Although there are theories in the literature such as assimilation of agile practices, routineization, change management, sensemaking, and institutionalization, dynamic capabilities, the growing scale of Software Development practices weakens the assumptions of these theories and limits their effectiveness. According to the literature review, studies on the implementation of agile methods often lack theoretical foundations, the theories used are based on weak assumptions, and excessive reliance on simple adherence to agile methods often causes problems. On the other hand, it is possible to go beyond simple adherence and wrong assumptions with an iterative normalisation. Therefore, the thesis of this study is that there is no widely accepted common theory for the (routine) implementation, implementation and integration of agile thinking (values, principles, practices) and agile methods (e.g. Scrum) into traditional Software Development practices. Literature stated that one of the ways to solve such problems is the NPT. It explains the implementation, establishment, integration (or maintenance) of new norms in various fields from health sciences to implementation sciences. Carroll et al. (2020) examined two agile transformation cases through the NPT lens and determined that more efforts should be made to better understand the differentiation, justification, values, and roles of the components, especially in the sense-making and participation dimensions, by the participants. Carroll et al. (2023b) point out that failure to invest in sustainability in transformation practices will prevent continuous normalisation. Beck (2000) could not provide sufficient explanations, especially for large-scale software development practices, since he focused on small-scale software teams while describing agile methods.

According to the research results, more investment should be made in “sense making”, “participation”, “implementation” and “evaluation” for agile/lean software development practices to become more normal. Findings in the literature support this. In a qualitative study by Luig et al. (2018), it was shown that facilitating practices, sense making and supporting the adoption of innovation (intervention), and iterative evaluation of implementation processes are important. In an empirical study conducted through the NPT lens, Carroll et al. (2020) showed that the sense making structure has a decisive effect on the success of complex interventions. According to the research findings, the main factors that facilitate the normalisation of agile/lean software development include clearly communicating the reasons for using agile to employees, determining roles, eliminating agile/lean knowledge-skill deficiencies and employee evaluations for the configuration of agile/lean practices. Although these are compatible with the literature, the expected evaluations and improvements increase the workload on employees. On the other hand, some of the factors that make it difficult to normalize agile software development practices are that employees and managers interpret agile/lean in different ways and that employees doubt the legitimacy of their involvement in agile practices. The findings in the literature are parallel to this.Eurofound (2021) points out that these complicating factors can prevent the effective implementation of agile/lean practices. Axelos, (2019) and ITIL 4 (2019) have proven their effectiveness as a package for agile software development practices as 'best practice

It is important to evaluate the feedback of participants based on their experiences for the normalisation of software development practices for reconfiguration purposes. However, employee feedback is not always used for reconfiguration purposes. In order to overcome the dissatisfaction and difficulties caused by this situation, regularly requesting feedback from employees and acting accordingly makes it easier for organizations to make conscious adjustments. In addition, it ensures that agile software development practices are up-to-date and effective. On the other hand, short and systematic training should be provided to participants with training resources created to support participation in agile/lean software development practices. Training, clarification of roles and the development of a supportive environment that encourages employees to take ownership of agile/lean practices are necessary. The difference and importance of agile/lean software development practices from traditional practices should be fully and accurately understood and perceived by all stakeholders. In this way, it is possible to achieve more effective harmony between the strategic focus of managers and the operational practices of employees. In addition, it should be ensured that there are enough key employees, leaders and agile coaches to support the establishment of a common understanding of the purpose and benefits of new practices (norms) in order not to make it difficult to normalize agile/lean software development practices.

Giriş

Teknolojik altyapılar artan veya değişen gereksinimler karşısında yetersiz kaldığında sürekli olarak dijital dönüşüm veya modernizasyon uygulanmaktadır. Günümüzde başta coğrafi olarak dağıtık yüksek hızlı sanal sunuculara (bulut bilişim) dayanan teknolojik altyapılar ve değişime hızlı cevap veren dinamik düşünce tarzı (çevik/yalın) birbirini tamamlayan büyük dijital dönüşüm trendleri olarak öne çıkmıştırlar (Adam, 2023:61; Athambawa vd., 2023:20-22; OECD, 2019:3; Ogala ve Mughele, 2022:68). Bunun en önemli nedeni dijital dönüşüm ortamında karşılaşılan inovasyon içeren problemlerin dinamik düşünce tarzlarıyla (çevik/yalın) daha hızlı çözülebilmeleridir. Çevik düşünce tarzının dayandığı prensipler ve değerler genel olarak çapraz fonksiyonlu ekip çalışmasını, problemleri kolay yönetilebilir küçük parçalara bölüp yinelemeli olarak hızlı çözümler üretmeyi ve esnekliği desteklemektedirler. Çevik düşünce tarzı pandemi ve sonrasında başta geleneksel çalışma biçimlerine uygulanarak yeni normal haline gelmiştir (Adam, 2023:61; Rad ve Rad, 2021:339). Ancak bu durum iki önemli probleme yol açmıştır. Birincisi, Taylorist temellere dayanan plan odaklı geleneksel yönetim yaklaşımları ile çevik düşünce tarzı birbiriyle çelişmektedir. Literatürdeki bazı yazarlar bunu kısaca paradigma değişimi olarak adlandırmaktadırlar (Rad ve Rad, 2021:362). Bu bağlamda değişime hızlı cevap vermeye odaklanan çevik düşünce tarzını geleneksel yönetim yaklaşımlarına uygularken çok sayıda problemle ve değişime karşı dirençle karşılaşılmaktadır (Reginaldo ve Santos, 2020:5-6). İkincisi, çevik düşünce tarzı gibi yeni normların geleneksel yönetim yaklaşımlarına uygulanması, (rutin olarak) yerleştirilmesi ve bütünleştirilmesi (veya sürdürülmesi) için büyük ölçüde kabul görmüş ortak bir kuram henüz yoktur (Carroll ve Conboy, 2020:1). Yazında "çevik pratiklerin asimilasyonu, rutinleştirme, değişim yönetimi, farkındalık yaratma, kurumsallaştırma ve dinamik yetenekler" gibi kuramlar olsa da Yazılım Geliştirme uygulamalarının büyüyen ölçeği bu kuramların etkililiğini sınırlandırmaktadır (Carroll vd., 2023a:270). Bu kapsamda bu çalışmanın savı çevik düşünce tarzının (değerler, prensipler, pratikler) ve çevik yöntemlerin (örn. Scrum) geleneksel Yazılım Geliştirme uygulamalarına (rutin olarak) yerleştirilmesi, uygulanması ve bütünleştirilmesi için büyük ölçüde kabul görmüş ortak bir kuramın olmamasıdır. Çalışmanın amacı Yazılım Geliştirme uygulamalarına çevik düşünce tarzının ve çevik yöntemlerin (örn. Scrum) (rutin olarak) yerleştirilmesini, uygulanmasını ve entegre edilmesini (veya sürdürülmesini) kolaylaştıran ve zorlaştıran faktörleri Normalleşme Süreci Kuramı (NSK) ile tespit etmektir. NSK ile yeni normların günlük uygulamalara rutin olarak yerleştirilmesi, uygulanması, bütünleştirilmesi

(veya sürdürülmesi) ve değerlendirilmesi amaçlanmaktadır (Normalisation Process Theory, 2024). Bunun için NSK'nın "tutarlı anlamlandırma", "bilişsel katılım", "kolektif eylem" ve "refleksif izleme" yapılarından yararlanılmaktadır (May ve Finch, 2009:546). NSK'nın birinci yapısı (anlamlandırma) ile katılımcıların yeni normal hakkındaki algıları, düşünceleri ve ön yargıları sorgulanmaktadır (Carroll ve Maher, 2023: 104). Elde edilen bilgiler çeviğin rutin olarak uygulanmasına ve benimsenmesine ilişkin direncin öngörülmesini kolaylaştırmaktadır. NSK'nın ikinci yapısıyla (bilişsel katılım) hedeflenen pratiğe yeni normun nasıl uygulanacağı tasarlanmaktadır. Ardından yeni normun uygulandığı pratiği operasyonel hale getirmek için pratik etrafında örgütlenilmektedir. Diğer bir ifadeyle yeni normu rutin iş süreçlerine yerleştirmek için insanlar sürece dâhil edilmektedir. NSK'nın üçüncü yapısı (kolektif eylem) ile çevikleşen pratik ile kolektif olarak eyleme geçilmektedir. Çevikleşen pratik daha geniş bağlama entegre edilmektedir (veya sürdürülmektedir). NSK'nın dördüncü yapısıyla (refleksif izleme) ise çevikleştirilen pratiğin çalışan geribildirimleri doğrultusunda konfigürasyonu sağlanmaktadır (Carroll ve Conboy, 2020:2-5). Bu çalışmada "Yazılım Projesi Yönetimi pratiğini etkileyen çevik düşünce tarzının iş süreçlerine (rutin olarak) yerleştirilmesini, uygulanmasını, entegre edilmesini (veya sürdürülmesini) kolaylaştıran ve zorlaştıran faktörler nelerdir?" sorusu cevaplanmaktadır. Bu araştırma sorusunu cevaplamak için nitel araştırma yöntemlerinden çoklu durum çalışması deseni (tek analiz birimi) benimsenmiştir ve veri kaynağı (katılımcı) çeşitlemesine gidilmiştir. Çoklu durum çalışması deseninde her durum kendi içinde bütüncül olarak ele alınmıştır ve daha sonra birbirleriyle karşılaştırılmıştır (Yıldırım ve Şimşek, 2021:314). Veri kaynağı (katılımcı) çeşitlemesi için veriler hem (yazılım geliştirme projelerinde) çalışanlardan hem de yöneticilerinden toplanmıştır. Uygulanan bu araştırma stratejisi ile çalışmanın güvenilirliğinin artırılması ve bulguların daha iyi genellenebilmesi amaçlanmıştır (Akar, 2020:55). Araştırmanın gerçek ve pilot uygulaması kapsamında toplam beş çalışan ve yönetici ile yarı yapılandırılmış görüşmeler yapılmıştır. Araştırma sonuçlarına göre çevik yazılım geliştirme uygulamalarının daha fazla normalleşmesi için (rutine daha fazla yerleştirilmesi, uygulanması ve entegrasyonu için): "anlam oluşturmaya", "katılıma", "uygulama çabalarına" ve "değerlendirmeye" biraz daha yatırım yapılması gerekmektedir. Araştırma bulgularına göre çeviğin normalleşmesini kolaylaştıran başlıca faktörler arasında çevikten yararlanma nedenlerinin çalışanlara net bir şekilde iletilmesi, rollerin belirlenmesi, çevik bilgi-beceri eksikliklerinin giderilmesi ve çevik uygulamaların konfigürasyonu için çalışan değerlendirmeleri bulunmaktadır. Diğer yandan çeviğin normalleşmesini zorlaştıran faktörlerden bazıları çalışanların ve yöneticilerin çeviği farklı şekillerde anlamlandırmaları ve çalışanların çevik uygulamalara dâhil olmalarının meşruiyetinden şüphelenmeleridir. Bu çalışmanın üç temel katkısı vardır. Birincisi, çalışma çevik yazılım geliştirme uygulamalarının normalleştirilmesinde NSK'nı uygulamanın yeniliğini göstermektedir. İkincisi, çalışmada çevik yazılım geliştirme yapan çalışanlar ve yöneticiler üzerinde görgül analizler yapılacaktır. Üçüncüsü, NSK son beş yılda artan dijital dönüşüm girişimlerine paralel olarak ortaya çıkan 'dijital dönüşüm değerlendirme' araçlarına kuramsal bir alternatif olarak sunulmaktadır. Çalışma dört bölümden oluşmaktadır. Birinci bölümde kuramsal çerçeve, ikinci bölümde yöntem, üçüncü bölümde bulgular ve dördüncü bölümde sonuç ve tartışma yer almaktadır. Bu kapsamda birinci bölümde, NSK'nın amacı, varsayımları, yapıları, bileşenleri ve sınırlılıkları tartışılmıştır. Ardından çevik Yazılım Geliştirme ve normalleşme konusu incelenmiştir. İkinci bölümde, araştırma yöntemi kapsamında araştırmanın amacı, önemi, yöntemi ve yapılan çoklu durum çalışmasının diğer detayları açıklanmıştır. Üçüncü bölümde, araştırma kapsamında elde edilen bulgular doğrudan atıflarla desteklenerek ve görselleştirilerek ortaya konulmuştur. Sonuç ve tartışma bölümü ile çalışma tamamlanmıştır.

1. Kuramsal Çerçeve

1.1. Normalleşme Süreci Kuramı

"Normalleştirme Süreci Teorisi (NSK) süreç değerlendirmeleri üzerine faydası kanıtlanmış, yaygın olarak kullanılan bir uygulama kuramıdır" (Pacini vd., 2023:7). NSK yeni normların anlamlandırılmasını, günlük işlere rutin olarak yerleştirilmesini (operasyonel hale getirme), kolektif olarak uygulanmasını (normalleşme) ve daha geniş bağlam ile entegre edilmesini (veya sürdürülmesini) anlamak ve değerlendirmek için kavramsal bir çerçeve sağlamaktadır (Carroll ve Conboy, 2020:2-5; May ve Finch, 2009:546; Normalisation Process Theory, 2024). "NSK özellikle işin toplumsal örgütlenmesi (uygulama), pratiklerin günlük yaşamın rutin unsurları haline getirilmesi (gömme) ve gömülü pratiklerin toplumsal bağlamlarında sürdürülmesi (entegrasyon) ile ilgilenmektedir" (Carroll, 2020:9). Kuram yeni teknolojilerin çalışanların bireysel ve kolektif çalışmalarının bir sonucu olarak çalışanların günlük işlerine rutin olarak yerleştirildiğini varsaymaktadır (Normalisation Process Theory, 2024). NSK'ya göre normalleşme kuramın dört yapısı ve on altı bileşeniyle gerçekleştirilmektedir (Tablo 1). NSK'nın ilk yapısı olan 'anlamlandırma' ile karmaşık müdahale (yeni normal) hakkında örgütsel konsensus oluşturmaktır (May ve Finch, 2009). İkinci yapı 'bilişsel katılım' ile karmaşık müdahale operasyonel hale getirilmektedir. Üçüncüsü, 'kolektif eylem' ile operasyonel hale getirilen teknoloji veya pratik kolektif olarak uygulanmaktadır. Dördüncüsü, 'refleksif izleme' yapısı ile yapılan uygulama çalışanlar tarafından değerlendirmektedir ve doğrultusunda yeniden konfigüre edilmektedir (May ve Finch, 2009:542-546). Böylece NSK ile çalışanların yeni teknolojiyi veya pratiği benimseme düzeyinin (bağlamsal) etkililiği sürekli olarak artırılmaktadır. En iyi teknoloji uygulama, yerleştirme ve bütünleştirme sonuçlarına ulaşmak (veya sürdürülebilirlik) için NSK yapılarının yinelemeli olarak uygulanması önerilmektedir (Carroll vd., 2021:5). Diğer uygulama kuramları gibi NSK da zayıf kanıta dayalı pratiklere dayandığı için eleştirilmiştir (Oludapo vd., 2024: 11-12). Ek olarak işin sosyal organizasyonuna odaklanan NSK'yı kullanmak aktör ağ yapılarına (örn. Aktör Ağ Kuramı) veya örgütsel psikolojiye (örn. Planlı Davranış Kuramı) daha az dikkat etmek anlamına gelmektedir (Pope vd., 2013:11).

Tablo 1 : NSK'nın dört yapısı ve on altı yapısal bileşeni

NSK Yapıları	Yapısal Bileşenler
Anlamlandırma	**Farklılaşma:** Katılımcılar müdahaleyi yeni bir iş yapma şekli olarak görüyorlar mı? **Bireysel Betimlemeler:** Katılımcılar yeniliğin kendilerinden beklenen görevleri ve sorumlulukları anlıyor mu? **Kolektif Betimlemeler:** Katılımcılar müdahalenin amacı konusunda hemfikir mi? **İçselleştirme:** Katılımcılar müdahalenin potansiyel faydalarını anlıyor mu?
Bilişsel Katılım	**Görevlendirme**: Katılımcılar kendilerinin uygulamayı ileriye taşıyacak doğru kişiler olduklarına inanıyorlar mı? **Başlatma**: Başta kilit katılımcılar başkalarını uygulamaya dahil etme konusunda istekli ve yetenekli mi? **Etkinleştirme**: Katılımcılar müdahaleyi sürdürmek için hangi görev ve faaliyetlerin gerekli olduğunu belirleyebilir mi? **Meşrulaştırma**: Katılımcılar müdahaleye katılmanın kendileri için doğru olduğuna inanıyorlar mı?
Kolektif Eylem	**Etkileşimli çalışılabilme**: Müdahale, görevlerin tamamlanmasını kolaylaştırıyor mu? **Beceri grubunun uygulanabilirliği**: Katılımcılar iş için doğru becerilere sahip mi? **İlişkisel entegrasyon**: Katılımcılar yeni çalışma şekline güveniyorlar mı? **Bağlamsal entegrasyon**: Örgütsel kaynaklar ve politikalar uygulamayı destekliyor mu?
Refleksif İzleme	**Sistemleştirme**: Katılımcılar müdahalenin etkililiğini değerlendirebilecek mi? **Bireysel değerlendirme**: Katılımcılar müdahalenin etkililiğini nasıl değerlendirecek? **Kolektif değerlendirme**: Katılımcılar müdahalenin etkililiğini kolektif olarak nasıl değerlendirecek? **Yeniden konfigürasyon**: Katılımcılar değerlendirme ve deneyime dayalı olarak müdahaleyi değiştirebilecek mi?

Kaynak: Huddlestone vd., 2020:3

1.2. Çevik Yazılım Geliştirme ve Normalleştirme

Yazılım isterlerinin büyük ölçüde değişmediği durumlarda yazılım geliştirme projelerinin seyri büyük ölçüde öngörülebilirdir. Bu bağlamda geleneksel yazılım geliştirme uygulamaları ile iyi planlanmış ve kaliteye odaklanan bir yazılım geliştirme süreci benimsenmektedir (Wysocki, 2019: 341-350). Büyük ön planlamalarla başlayan bu süreçte gereksinim analizi, tasarım, uygulama, test ve bakım faaliyetleri sıklıkla ardışık aşamalar halinde gerçekleştirilmektedir (Sommerville, 2016: 11-112). Ancak bu katı yazılım geliştirme modeli durum geliştiricilere fazla esneklik sağlamadığı için yazılım isterlerindeki değişime karşı biraz dirençlidir. Diğer yandan bu yöntem ile dinamik değişim ortamında sürekli değişen yazılım isterlerine cevap vermek oldukça maliyetlidir. Dolayısıyla yazılım isterlerinin büyük oranda belirsiz olduğu veya sürekli değiştiği durumlarda çevik yazılım geliştirme ile yazılım geliştirme kolaylaştırılmaktadır. Bu kapsamda Çevik Manifesto prensipleri ve değerleri ile değişen yazılım isterlerine hızlı cevap vermek, çapraz fonksiyonlu ekipler, yazılım isterlerinin sahipleri olan müşterilerle yakın iş birliği, problemlerin yönetilebilir küçük parçalara bölünerek yinelemelerle hızlı çözümler üretmek ve esneklik teşvik edilmektedir (Agile Manifesto Organization, 2001). Bunlara ek olarak *XP, Scrum, Kanban* (veri görselleştirme) gibi çevik yöntemlerle de çevik yazılım geliştirme kolaylaştırılmaktadır (Cohen vd., 2003: 13-15; Schwaber, 2004: 25-64). Tablo 2 incelendiğinde *Scrum* yönteminin temel özellikleri görülmektedir.

Tablo 2: Scrum yönteminin tipik özellikleri

Roller: Ürün sahibi, ürün yöneticisi, *Scrum* uzmanı, çevik süreç koçu ve geliştirme ekibi.
Kullanılan araçlar: Ürün iş listesi/yapılacaklar listesi, *Sprint* iş listesi, kalan iş-saat grafiği, Scrum/Kanban panosu.
Çevik seremoniler: *Sprint* (örn. 2-4 hafta), *Sprint* planlaması, günlük *Scrum* toplantısı, *Sprint* değerlendirmesi, *Sprint* retrospektifi.

Kaynak: Schwaber (2004: 25-64)

Geleneksel ve çevik yazılım geliştirme uygulamaları arasındaki farklar önceki paragrafta ana hatlarıyla ifade edilmiştir. NSK açısından normalleşme karmaşık bir müdahalenin (veya normun) rutinleşmesi ve uygulanması anlamına gelmektedir. NSK açısından normalleşmenin ilk adımı iki uygulama arasındaki farkların belirlenmesiyle başlamaktadır ve katılımcıların değişimi içselleştirmesiyle tamamlanmaktadır. Bu aşamanın başarısı önemlidir çünkü sonraki adımda değişime karşı direncin belirleyicisidir. Normalleşmenin ikinci adımı karmaşık müdahalenin getirdiği yeni uygulamanın tasarlanmasını ve uygulama etrafında örgütlenmeyi gerektirmektedir. Örneğin hangi çevik prensiplerin ve yöntemlerin kimler tarafından benimseneceği, görevler, sorumluluklar bu adımda belirlenmektedir. Bu aşamanın başarısı büyük oranda yeni uygulama etrafında örgütlenmeye bağlıdır. Yeni uygulamanın katılımcılar arasında kazandığı meşruiyet bir sonraki aşamanın başarısını artırmaktadır. Normalleşmenin üçüncü adımında, önceki aşamada operasyonel hale getirilen yeni uygulama ile kolektif olarak eyleme geçilmektedir. Yeni bir uygulama ile kolektif olarak eyleme geçebilmek uygulamanın normalleştiğini göstermektedir. Bu kapsamda beceri boşlukları ve beceri gereksinimleri net bir şekilde tespit edilmektedir. Normalleşmenin dördüncü aşamasının amacı yeni uygulamanın katılımcılar (çalışanlar) tarafından değerlendirilmesi ve ayarlanmasıdır. Böylece yeni uygulamanın aşağıdan yukarıya doğru sürekli olarak iyileştirilmesi sağlanmaktadır. Çünkü NSK'ya göre normalleşme kalıcı bir durum değildir (örn.May vd., 2007:3).

2. Araştırmanın Amacı ve Yöntemi

Çalışmanın amacı Yazılım Geliştirme uygulamalarına çevik düşünce tarzının (rutin olarak) yerleştirilmesini, uygulanmasını ve entegre edilmesini kolaylaştıran ve zorlaştıran faktörleri

NSK ile tespit etmektir. Bu amaç doğrultusunda yazılım geliştirme pratiğini etkileyen çevik düşünce tarzının iş süreçlerine (rutin olarak) yerleştirilmesini, uygulanmasını, entegre edilmesini (veya sürdürülmesini) kolaylaştıran ve zorlaştıran faktörler nelerdir?" araştırma sorusu cevaplanmaktadır. Bu araştırma sorusunu cevaplamak için nitel araştırma yöntemlerinden çoklu durum çalışması deseni (tek analiz birimi) benimsenmiştir. Çoklu durum çalışması deseninde her durum kendi içinde bütüncül olarak ele alınmıştır ve daha sonra birbirleriyle karşılaştırılmıştır (Yıldırım ve Şimşek, 2021:314). Çoklu durum çalışması hem olayın veya durumun derinlemesine incelenmesi ve analiz edilmesi hem de çalışmanın güvenilirliğini artırmak amacıyla yapılan bir araştırma yöntemdir (Yıldırım ve Şimşek, 2021:314). Bu çalışmada çeşitli yazılım geliştirme projelerinde çevik yöntemleri kullanan çalışanların ve yöneticilerin NSK merceğinden derinlemesine incelenmeleri sağlanmıştır. Bu tez makalesinde Ankara Üniversitesi Etik Kurulu Başkanlığınca 13/180 sayılı kararla 07.07.2022'de onaylanmış bulunan "Pandemi Krizinden Kaynaklanan Dijital Dönüşüm Kapsamında Gündeme Gelen İK ve BT Pratiklerinin Kalıcılaştırılması: Otomotiv Sektöründe Nitel Bir Araştırma" başlıklı tez çalışmasının verilerinden yararlanılmaktadır. Araştırma verileri çevik yazılım geliştirme projelerinde çalışan 5 çalışandan ve yöneticiden yarı yapılandırılmış görüşmelerle toplanmıştır. Görüşme yapılan katılımcılar amaçlı örnekleme yöntemine göre seçilmiştir. Amaçlı örnekleme yönteminin veri kaynağı (katılımcı) çeşitlendirme stratejisiyle birlikte kullanılması araştırma sorusunu aydınlatacak daha yüksek geçerliğe sahip bir çalışmanın önünü açmaktadır (örn. Akar, 2020:55).

Tablo 3: Çalışma grubunun özellikleri

	Katılımcılar	Cinsiyet	Yaş	Eğitim	Ünvan	Çevik Tecrübesi (yıl)	Mesleki Tecrübesi (yıl)
1	BT1	E	28	Yüksek Lisans (devam ediyor)	Proje Yöneticisi	6	7
2	BT2	E	30	Lisans	Proje Yöneticisi	4	7
3	BT3	E	32	Lisans	Yazılım Geliştirme Uzmanı	1	1
4	BT4	E	27	Lisans	Yazılım Geliştirme Uzmanı	3	2
5	BT5	E	56	Yüksek Lisans	BT Bölge Müdürü	10	25

Tablo 3 incelendiğinde beş katılımcının da Bilgi Teknolojileri (BT) alanında çalıştığı görülmektedir. Katılımcıların hepsinin en az bir yıl çevik çalışma deneyimi bulunmaktadır. Katılımcılardan dördü gerçek biri pilot uygulamaya katılmıştır. Pilot uygulamaya katılan katılımcıdan (BT5) elde edilen veriler gerçek uygulama dışında tutulmuştur. Veri toplama aracı olarak önceki çalışmalar için geliştirilmiş tematik yarı yapılandırılmış görüşme sorularından yararlanılmıştır (Carroll vd., 2023a: 296-298). Görüşme sorularının İngilizce-Türkçe ve Türkçe-İngilizce çevirisi yapılmıştır. Görüşme soruları 20-25 Aralık 2023 tarihinde yapılan pilot uygulamada test edilmiştir ve uzman görüşleri doğrultusunda gereken ayarlamalar yapılmıştır. Görüşme formunun son haliyle 02.04.2024 ile 27.05. 2024 tarihleri arasında araştırmacı tarafından katılımcılarla görüşmeler gerçekleştirilmiştir. Araştırma kapsamında elde edilen nitel verilerin analizinde tematik içerik analizi ve karşılaştırmalı analiz tekniklerinden yararlanılmıştır. Tematik içerik analizinde, benzer veriler NSK merceğinden belirli kategoriler (alt temalar) ve temalar altında bir araya getirilmiştir, düzenlenmiştir ve anlaşılır bir şekilde yorumlanmıştır (Yıldırım ve Şimşek, 2021: 254-257). Ardından

karşılaştırmalı analizlerde, çalışanlara ve yöneticilere ilişkin bulgular birbirleriyle karşılaştırılarak daha zengin ve açıklayıcı bulgulara ulaşılmıştır.

3. Bulgular

3.1. Çevik Yazılım Geliştirme Uygulamasının Anlamlandırılmasına İlişkin Bulgular

Yeni uygulamanın (örn. çevik) anlamlı özelliklerinin katılımcılar tarafından anlaşılmasını kolaylaştıran veya zorlaştıran faktörler tespit edilmektedir (Carroll vd., 2023a: 273; May ve Finch, 2009:542). Katılımcılara Tablo 4'te yer alan tematik sorular sorulduğunda Tablo 5'te bulunan dört alt tema ortaya çıkmıştır.

Tablo 4: Çeviğin yazılım geliştirme uygulamalarına yerleştirilmesinin anlamlandırılmasıyla ilgili sorular

No	Çalışanlara (Ç) ve/veya Yöneticilere (Y) Sorulan Sorular	
2	Y	Çevik yazılım geliştirme uygulamasının geleneksel yazılım geliştirme uygulamasından farkını çalışanlara nasıl aktarıyorsunuz?
3	Y	Çevik yazılım geliştirme uygulamasının gerekçesini çalışanlara nasıl aktarıyorsunuz?
4	Ç Y	Çevik yazılım geliştirme uygulamasının amaçları ve beklenen faydalarına ilişkin düşünceleriniz nedir?
5	Ç Y	Sizce çevik yazılım geliştirme uygulamasının gerekçesi nedir? Sonda sorusu: Beklenen fayda nedir?
6	Ç Y	Çevik yazılım geliştirme uygulamasına ilişkin görev ve sorumluluklarınız hakkında ne düşünüyorsunuz?
7	Ç Y	Çevik yazılım geliştirme uygulamasının (önerilen) değerini ve faydalarını çalışanlara iletmek için ne gibi bir anlamlandırma çalışması yapıldı?
8	Ç Y	Çevik yazılım geliştirme uygulamasının temel değerleri, faydaları ve önemi hakkında farkındalık yaratmak için ne tür eğitim verildi?
9	Ç Y	Çevik yazılım geliştirme uygulamasının değeri ve faydaları konusunda beklentilerinizi ve hedeflerinizi neler yönlendirmektedir?

Kaynak: Carroll vd., 2023a: 296-298'den esinlenilmiştir.

Katılımcıların çevik çalışmayı anlamlandırma ile ilgili görüşlerinin analiz edilmesiyle elde edilen alt temalar Tablo 5'te bulunmaktadır. Tablo 5 incelendiğinde NSK'nın farklılaşma, kolektif açıklamalar, bireysel açıklamalar ve içselleştirme alt temalarını içerdiği görülmektedir.

Tablo 5: Katılımcıların çevik çalışmayı anlamlandırma ile ilgili görüşlerinin oluşturduğu alt temalar

Tema	Alt Tema (Kategoriler)	Katılımcılar
Anlamlandırma	Farklılaşma	BT1, BT2, BT3, BT4
	Kolektif Açıklamalar	
	Bireysel Açıklamalar	
	İçselleştirme	

Katılımcıların anlamlandırmayı oluşturan alt temalarla ilgili görüşlerine ait değerlendir-meler Tablo 6'da yer almaktadır. Tablo 6 incelendiğinde çevik uygulamalara yönelik olumlu ve olumsuz anlamlandırmalar olduğu görülmektedir.

Tablo 6: Çalışanların ve yöneticilerin anlamlandırmayı oluşturan alt temalarla ilgili görüşlerine ait değerlendirmeler

	Çevik Yazılım Geliştirme Uygulamalarını Anlamlandırmak			
	Geleneksel (Plan Odaklı) ve Çevik Uygulamalar Arasındaki **Farklılaşma**	Çevik Uygulamalara ilişkin **Kolektif Açıklamalar**	Çevik Uygulamalara ilişkin **Bireysel Açıklamalar**	Çevik Uygulamaları **İçselleştirme**
BT1	*Scrum*/Ustası olan Proje Yöneticisi, çevik yazılım geliştirmenin geleneksel yazılım geliştirmeden farkını büyük oranda ifade etmiştir.	*Scrum* Ustası olan Proje Yöneticisine göre çevik yazılım geliştirme yazılım kalitesini ve hızını artırmaktadır.	*Scrum* Ustası olan Proje Yöneticisine göre çevik yöntemler proje yönetimini ve ekibin organizasyonunu kolaylaştırmaktadır.	*Scrum* Ustası olan Proje Yöneticisi için çevik yazılım geliştirme çok faydalıdır. Kendisi çevik yazılım geliştirmeyi etkili ve verimli bulmaktadır.
BT2	Proje Yöneticisi, çevik yazılım geliştirmenin plan odaklı yazılım geliştirmeden temel farklarını hakkında çok net bir anlayışa sahip değildir	Proje Yöneticisine göre çevik yöntemler ekibin yazılım geliştirme hızını ve çalışanın yeteneklerini kullanma miktarını artırmaktadır.	Proje Yöneticisine göre çevik yöntemlerle geliştirme yaparken fazla dokümantasyon hazırlamaktan kurtulmaktadır.	Proje Yöneticisine göre çevik yöntemler öngörülmeyen problemleri erken ele almayı ve hızla gidermeyi kolaylaştırmaktadır.
BT3	Çalışana göre çevik ve geleneksel yazılım geliştirme arasındaki farklardan çok emin değildir.	Çalışana göre çevik yazılım geliştirme ekibin koordinasyonunu ve hızını artırmaktadır	Çalışana göre çevik geliştirmede görevlerinin çok sayıda küçük parça halinde verilmesi kendisi için daha konforludur.	Çalışan çevik yazılım geliştirmeyi tam olarak içselleştirememiştir (benimseyememiştir).
BT4	Çalışana göre çevik ve geleneksel yazılım geliştirme arasında belirgin bir farklılık yoktur.	Çalışana göre çevik yöntemler çalışanın iş yükünü artırmaktadır.	Çalışan çevik yöntemlerle yazılım geliştirmektedir ancak çevik bireysel olarak ilgisini çekmemektedir.	Çalışan çevik yazılım geliştirmeyi tam olarak içselleştirememiştir (benimseyememiştir).

Tablo 6 incelendiğinde katılımcıların çevik yazılım geliştirme uygulamalarını nasıl anlamlandırdıkları dört alt tema açısından ayrı ayrı görülebilmektedir. Bunlara ilişkin birinci alt tema olan "farklılaşma" (eski ve yeni uygulama arasındaki farklar) (Carroll, 2023 vd., 273) üzerine örnek görüşler aşağıda verilmiştir:

> *"Çevik [Scrum] ile değişen müşteri isteklerine hızlı ve etkili bir şekilde cevap verebiliyoruz…Özellikle az bilgiyle yeni başladığımız projelerde kısa planlama ve çevik geliştirmeyi tercih ediyoruz…Hakimiyetimizin bayağı yüksek olduğu projelerde V-model'e veya Waterfall'a dönüyoruz…Bazen çevik projelerin sonuna doğru da Waterfall yaptığımız oluyor…Duruma bağlı yani."* (BT1)

Proje Yöneticisi olan BT1 hangi durumlarda çevik hangi durumlarda geleneksel yaklaşımları tercih ettiklerini açıkça ortaya koymuştur.

> "Çevik (Scrum) galiba kalabalık ekipler için gerekiyor…D***'nin söylediğine göre çevik olmasaydı bu kadar kişiyle [yazılımcıyla] birlikte çalışamazdık." (BT3)

> *"Çevik yazılım geliştirme ve destek operasyonları ile önceki süreçler arasında önemli bir fark olduğunu sanmıyorum…"* (BT4).

İkinci alt tema olan "kolektif açıklamalar" (ortak vizyon, amaçlar, hedefler, beklenen faydalardır) (Carroll, 2023 vd., 273) üzerine örnek görüşler aşağıda verilmiştir:

> *"Çevik metodoloji yazılım geliştirme hızımızı artırıyor. Bu nedenle çoğu zaman çevik çalışıyoruz."* (BT2)

> *"Çeviğin tam faydasını bilmiyorum ama çevikte müşteri memnunsa dokümantasyon az veya yok."* (BT2)

> *"Çevik hızlı olabilir ama iş yükünü artırıyor…Müşteri her dakika geribildirim gönderiyor."* (BT4)

Üçüncü alt tema olan olumsuz "bireysel açıklamalar" (kişiye özel görevler ve sorumluluklardır) (Carroll, 2023 vd., 273) üzerine örnek görüşler aşağıda verilmiştir:

> *"Çeviğin benim için faydalı olabileceğini düşünmüyorum."* (BT4).

BT4 çevik yazılım geliştirme yapmaktadır ancak çeviğe karşı bireysel ilgisi düşüktür.

Dördüncü alt tema olan "içselleştirme" (önerilen yeniliğin faydasını, değerini, önemini anlamaktır) (Carroll, 2023 vd., 273) üzerine örnek görüşler aşağıda verilmiştir:

> *"Çevik yazılım geliştirme önemlidir çünkü bize hızlı geliştirme yapabilme, çalışan takibi, süreç takibi, talep sahibine potansiyel ve rakip şirketlerden daha kaliteli ve sürekli yeni özellik sunabilme fırsatı vermektedir…Ekip olarak son kullanıcıya ulaşabilmek ve değişen taleplerine cevap verebilmek…"* (BT1).

3.2. Çevik Yazılım Geliştirme Uygulamasına Bilişsel Katılıma İlişkin Bulgular

Yeni uygulamaya (örn. çevik) katılımı kolaylaştıran veya zorlaştıran faktörler tespit edilmektedir (May ve Finch, 2009:543). Katılımcılara Tablo 7'de yer alan tematik sorular sorulduğunda Tablo 8'de bulunan dört alt tema ortaya çıkmıştır.

Tablo 7: Çevik yazılım geliştirme uygulamasına katılımla ilgili sorulan sorular

No	Çalışanlara (Ç) ve/veya Yöneticilere (Y) Sorulan Sorular	
10	Ç Y	Çevik yazılım geliştirme uygulamasının amacını ve hedeflerini nasıl tanımlıyorsunuz?
11	Ç Y	Çevik yazılım geliştirme uygulamanızın optimum çevikleşme ve dijitalleşme derecesi nedir?
12	Ç Y	Sizce firmanın çevik yazılım geliştirme ve destek uygulaması ile değer yaratmaya öncelik verdiği nasıl anlaşılır?
13	Y	Çevik yazılım geliştirme kapsamında ile ekibi koordine etmek için hangi süreçler uygulanıyor?
14	Ç Y	Çevik yazılım geliştirme uygulaması ile belirli bir ekip yapısının, rollerinin ve sorumluluklarının benimsemesinin mantığı nedir?
15	Y	Ekip üyeleri çevik yazılım geliştirme uygulamasına katkıda bulunabileceklerine nasıl inanıyorlar?
16	Y	Ekip üyelerinin çevik yazılım geliştirme uygulamasına nasıl katkıda bulunduğunu değerlendirmek için hangi mekanizmaları kullanıyorsunuz?
17	Y	Çevik yazılım geliştirme uygulamasını uygulayan çalışanlar arasında öğrenme kültürünü nasıl oluşturuyorsunuz?
18	Ç Y	Çevik yazılım geliştirme uygulamasını sürdürmek için gereken eylem ve prosedürleri kolektif olarak nasıl gözden geçiriyorsunuz ve tanımlıyorsunuz?
19	Ç Y	Size göre çevik yazılım geliştirme uygulamasına olan bağlılığı hangi faktörler etkiler?
20	Ç Y	Çevik yazılım geliştirme ve destek uygulamasını gerçekleştiren ekibin performansını ölçmek için hangi metrikler kullanılıyor?
21	Ç Y	Ekipler, çevik yazılım geliştirme uygulamasını sürdürmek için iş birliğini nasıl sağlamaktadırlar?
22	Ç Y	Ekip üyeleri, çevik yazılım geliştirme uygulamasını operasyonel hale getirmek için geleneksel ve çevik çalışma şekillerini birlikte kullanarak nasıl çalışmaktadır?

Kaynak: Carroll vd., 2023a: 296-298'den esinlenilmiştir.

Görüşmeler sonucunda elde edilen verilerden katılımcıların çevik çalışmaya bilişsel katılımları hakkındaki görüşlerinin analiziyle ilgili alt temalar Tablo 8'de bulunmaktadır. Tablo 8 incelendiğinde NSK'nın görevlendirme, başlatma, meşruiyet ve aktivasyon alt temalarını içerdiği görülmektedir.

Tablo 8: Katılımcıların çevik çalışmaya bilişsel katılımlarıyla ilgili görüşlerinin oluşturduğu alt temalar

Tema	Alt Temalar (Kategoriler)	Katılımcılar
Bilişsel Katılım	Görevlendirme	BT1, BT2, BT3, BT4
	Başlatma	
	Meşrulaştırma	
	Aktivasyon	

Tablo 9: Çalışanların ve yöneticilerin bilişsel katılımı oluşturan alt temalarla ile ilgili görüşlerinin karşılaştırmalı analiz sonuçları

	Çevik Yazılım Geliştirme Pratiklerine Bilişsel Katılım			
	Görevlendirme (Uygulamaya Katılma)	**Başlatma** (Katılımcıların katkılarını belirleme)	**Meşruiyet**	**Etkinleştirme**
BT1	*Scrum* Ustası olan Proje Yöneticisine göre çalışanın işe dahil edilmesinde kabiliyet ve gönüllülük esas alınmaktadır.	*Scrum* Ustası olan Proje Yöneticisinin liderliğindeki toplantılarda ekip üyelerinin rolleri ve görevleri (projenin isterlerine göre) belirlemektedir ve planlamaktadır.	*Scrum* Ustası olan Proje Yöneticisi, çevik yazılım geliştirmenin meşruiyetini sağlamak için kararları herkesin katıldığı toplantılarda almaktadır. İsteyen çalışanlara çevik eğitim paketleri hediye etmektedir.	*Scrum* Ustası olan Proje Yöneticisi, çevik yazılım geliştirme pratiklerinin sürdürülmesi için gereken prosedürlerin ve standartların ihtiyaç duyuldukça ekip tarafından tek tek tespit edildiğini belirtmektedir.
BT2	Proje Yöneticisine göre görev paylaşımını ekip üyeleri kendi arasında yapar. Herhangi bir sorun yoktur	Proje Yöneticisine göre her proje ekibinin ihtiyaç duyduğu insan kaynağını, eğitimi, yapılacak işlerin taslağını ve kullanacakları yazılım geliştirme araçlarını hazırlamaktadır.	Proje Yöneticisine göre çevik yazılım geliştirme pratiklerine ekip üyelerinin desteğini ve kabulünü sağlamak için sürekli toplantılar (kısa iş planlamaları) yapılmaktadır ve eğitimler sunulmaktadır.	Proje Yöneticisine göre çevik yazılım geliştirme pratiklerinin sürdürülmesi için gereken eylemler çevik çalışmaya başlamadan önce yönetim ekibi tarafından topluca tespit edilmiştir
BT3	Çalışana göre çevik çalışmayı ileriye taşıyan bir kilit çalışan vardır. Kilit çalışan görevlere katılımı organize ediyor.	Çalışana göre *Scrum* Ustası olan Proje Yöneticisi tarafından çevik yazılım geliştirme için detaylı bir hazırlık yapılmıştır. Gereken dijital araçlar (*Jira, Discord, Visual Studio Framework., Elasticsearch, Grafana, Prometheus, Jenkins, Github*) ve gereken insan kaynağı ile bir araya getirilmiştir. Görevlendirilmişlerdir.	Çalışana göre çalışanlar işe girmeden önce veya sonrasında çevik çalışma yapılacağı konusunda bilgilendiriliyor ve gönüllü bir katılım sağlanıyor.	Çalışana göre çevik yazılım geliştirme pratiklerinin gerçekleştirilmesi ve sürdürülmesi için gereken eylemler ve araçlar çevik çalışmaya başlamadan önce kilit çalışan tarafından topluca tespit edilmiştir. Karşılaşılan bir problem durumunda tüm ekip birlikte çözüm aramaktadır.
BT4	Ekip içinde görev paylaşımı çok zaman alıyor ve yıpratıcı oluyor.	Çalışana göre çevik yazılım geliştirmeye yapılacak katkıların belirlenmesi çok zaman alıcı ve yorucu olmaktadır.	Çalışana göre çevik çalışmanın meşruiyeti şüphelidir. Çünkü öncelikle çevik iş sözleşmesi yapılması gerekmektedir.	Çalışana göre çevik yazılım geliştirme pratiklerinin etkinleştirilmesinde (PMBOK veya agile PRINCE2 gibi) çevik standartların benimsenmesi faydalıdır.

Tablo 9 incelendiğinde katılımcıların çevik yazılım geliştirme uygulamalarını nasıl katıldıkları dört alt tema açısından ayrı ayrı görülebilmektedir. Bunlara ilişkin birinci alt tema görevlendirmedir. Görevlendirme ile yeni uygulamaya katkıda bulunacak kilit çalışanların yeniliğe dahil edilirken karşılaşılan zorlaştırıcı ve kolaylaştırıcı faktörler incelenmektedir (Carroll vd., 2023a:273). Birinci alt tema "görevlendirme" üzerine örnek görüşler aşağıda verilmiştir:

> *"Gereksinimleri en iyi algılayıp karar verebilen Project Manager olur. Ekipte bunu herkes bilir…"* (BT1)
>
> *"Ekip olarak planlama yapıyoruz… Planlamada her seferinde kimin hangi pozisyona katkıda bulunacağını back-end, front-end, middle tier (securiy) hep birlikte belirleriz."* (BT1)
>
> *"Arkadaşlar rol ve görev paylaşımını kendileri yapar… Ben kimin hangi görevi aldığını sisteme (Azure DevOps) girerim…Tahmini görev tamamlama süreleri genelde bellidir…Şimdilik herhangi bir sorun yok"* (BT2)
>
> *"Kendi aramızda görev paylaşımı yapmak çok fazla zaman alıyor ve çok zor oluyor… Özellikle yeni bir projenin başında kimin ne yapacağını bilmeden saatlerce görev paylaşımı yapmaya çalışıyoruz... Çevik çalışmada bunu bizim için birilerinin yapması veya kolaylaştırması gerekmiyor mu? Zaten bizim bir çalışan eksiğimiz var… Scrum Master olsa iyi olurdu ama sanmıyorum"* (BT4)

Çalışan (BT2), diğerlerinin (BT1, BT2) aksine görevlendirme ile ilgili sürekli karşılaştığı bir olumsuzluğu belirtmiştir.

İkinci alt tema başlatmadır. Başlatma ile katılımcıların yeni uygulamayı kullanarak yapabilecekleri spesifik katkılar belirlenmektedir (Carroll vd., 2023a:273). İkinci alt tema "başlatma" üzerine örnek görüşler aşağıda verilmiştir:

> *"Başlangıçta D** (Scrum Ustası olan Proje Yöneticisi) bir Sprint nasıl yapılır bize adım adım gösterdi."* (BT3)
>
> *"Çevik yazılım geliştirme uygulamalarını başlatmak için genel müdür tarafından kısa bir toplantı yapıldı. Yaklaşık iki saat falan sürdü. Çeviğin ne farkı var, eskisiyle ortak yönleri neler belirtildi. Bir örnek uygulama yapıldı. Sonuçta bunları hesaba katarak artık çevik yazılım geliştirebileceğimiz söylendi…Bunun dışında herhangi bir eğitim olmadı."* (BT2)
>
> *"Çevik çalışmaya başladığımızda Azure Devops lisansı satın alındı. Genel müdürün yaptığı toplantıda bunu nasıl kullanacağımız da anlatıldı"* (BT2)

BT3 ve BT2'nin açıklamalarına göre her ekipte en az bir kilit çalışanın (yönetici) başlatma kapsamında bir çalışma yapıldığı anlaşılmaktadır.

Üçüncü alt tema olan "meşrulaştırma" (katılımcıların çeviğe geçerli katkılar yapabileceklerine inanmalarıdır) (Carroll, 2023 vd., 273) üzerine olumlu ve olumsuz örnek görüşler aşağıda verilmiştir:

> *"Meşruiyeti sağlamak için tüm kararları herkesin katıldığı toplantılarda alıyoruz. Bir kişi bir şeye toplantıda itiraz etmediği sürece o şeyin meşru olduğunu kabul etmiş sayılıyor..."* (BT1)
>
> *"Çevik iş sözleşmesi nasıl yapılıyor?... Çevik iş sözleşmesi yapmadan çevik yazılım geliştirmek bana doğru gelmiyor."* (BT4)

Dördüncü alt tema olan "etkinleştirme" (çevik geliştirmeyi sürdürmeyi destekleyen eylemlerin ve prosedürlerin topluca belirlenmesidir) (Carroll, 2023 vd., 273) üzerine örnek görüşler aşağıda verilmiştir:

> *"Çevik çalışmayı etkinleştirmek için gereken plan ve prosedürleri neye göre kim yaptı bilmiyorum. Bunların PMBOK veya agile PRINCE2 gibi çevik standartlara uygun olarak yapılması iyi olmaz mı?"* (BT4)
>
> *"Çevik yazılım geliştirme için gereken tüm etkinleştirme gereksinimlerini en başta topluca öngöremiyoruz. Ancak ihtiyaç duydukça ekip olarak oturup araştırıyoruz ve tespit ediyoruz."* (BT1)
>
> "*Bizim plan ve prosedür işini galiba V** Bey [proje koordinatörü] yapar. Sonra herkesin katıldığı bir toplantıda bize anlatır."* (BT2)

3.3. Çevik Yazılım Geliştirme Uygulamasıyla Kolektif Eyleme Geçmeye İlişkin Bulgular

Yeni uygulamayla (örn. çevik) kolektif olarak eyleme geçmeyi kolaylaştıran veya zorlaştıran faktörler tespit edilmektedir (May ve Finch, 2009:542). Katılımcılara Tablo 10'da yer alan sorular sorulduğunda Tablo 11'de bulunan dört alt tema (kategori) ortaya çıkmıştır.

Tablo 10: Çevik (yazılım geliştirme) ile kolektif eyleme geçme konusundaki tematik sorular.

No	Çalışanlara (Ç) ve/veya Yöneticilere (Y) Sorulan Sorular	
23	Y	Çevik yazılım geliştirme uygulamalarında ekibinin performansını artırmak için geleneksel çalışma şekillerini kullanma konusunda ekipleri nasıl yetkilendirmektesiniz?
34	Y	Çevik yazılım geliştirme uygulamasını operasyonel hale getirmek ve bunlardan yararlanmak için görevleri nasıl paylaştırıyorsunuz?
25	Y	Ekiplerin çevik yazılım geliştirme uygulamasına güven nasıl oluşturuyorsunuz?
26	Ç Y	Çevik yazılım geliştirme uygulaması kapsamında size sunulan otonomi ne kadar tatmin edicidir?
27	C Y	Çevik yazılım geliştirme uygulamasına ilişkin iş bölümü nasıl yapılmaktadır?
28	Y	Çevik yazılım geliştirme uygulamasının temel eylemlerini ve metriklerini nasıl tanımlıyorsunuz?
29	Y	Çevik yazılım geliştirme uygulaması için gereken kaynakları nasıl tahsis ediyorsunuz? Sonda: Protokolleri, politikaları ve prosedürleri nasıl yürütmektesiniz?
30	Y	Çevik yazılım geliştirme uygulaması için kaynak tahsisine rehberlik eden kontrolleri nasıl gerçekleştiriyorsunuz?
31	Ç Y	Çevik yazılım geliştirme uygulaması için hangi teknikleri kullanıyorsunuz?
32	Ç Y	Çevik yazılım geliştirme uygulaması ne kadar etkili ve kullanışlıdır?

Kaynak: Carroll vd., 2023a: 296-298'den esinlenilmiştir.

Görüşmeler sonucunda elde edilen verilerden katılımcıların çevik çalışmayla kolektif olarak eyleme geçmeleri konusundaki görüşlerin analiziyle elde edilen alt temalar Tablo 11'de bulunmaktadır. Tablo 11 incelendiğinde NSK'nın etkileşimli çalışabilme, ilişkisel bütünleşme (ilişkisel entegrasyon), yeni beceri setiyle çalışabilme ve bağlamsal entegrasyon alt temalarını içerdiği görülmektedir.

Tablo 11: Katılımcıların çevikle kolektif eyleme geçme hakkındaki görüşlerinin oluşturduğu alt temalar

Tema	Alt Temalar (Kategoriler)	Katılımcılar
Kolektif Eylem	Etkileşimli Çalışabilme	BT1, BT2, BT3, BT4
	İlişkisel Betimleme	
	Yeni Beceri Seti ile Çalışma	
	Bağlamsal Bütünleşme	

Katılımcıların kolektif eylemi oluşturan alt temalarla ilgili görüşlerine ait karşılaştırmalı analizler Tablo 12'de yer almaktadır. Tablo 12 incelendiğinde kolektif eyleme yönelik zorlaştırıcı ve kolaylaştırıcı faktörler görülmektedir.

Tablo 12: Çalışanları ve yöneticilerin kolektif eylemi oluşturan alt temalarla ile ilgili görüşlerinin karşılaştırmalı analiz sonuçları

	Çevikle Kolektif Olarak Eyleme Geçmek			
	Çevik Uygulamalarla **Etkileşimli Çalışabilme**	Çevik Katılımcılar Arasındaki **İlişkisel Entegrasyon**	Çevik **Beceri Setiyle Çalışabilme**	Çevik Uygulamaların **Bağlamsal Entegrasyonu**
BT1	*Scrum* Ustası olan Proje Yöneticisi, yazılım geliştirme ekibi üyeleri çevik yazılım geliştirme uygulamasıyla ve birbirleriyle ekip içinde etkili bir şekilde çalışabilmektedirler.	*Scrum* Ustası olan Proje Yöneticisi, eğitimler, iş birliği, yazılım testleri ve psikolojik destek ile ekip üyelerinin birbirine karşı güveni sürdürmesi ve hesap verebilirliği sağlaması kolaylaşmaktadır.	*Scrum* Ustası olan Proje Yöneticisi, çalışanlara hediye edilen üçüncü taraf eğitim abonelikleriyle eğitim gereksinimleri karşılanmaktadır. Ekip üyelerinin görev ve sorumluluk paylaşımları kısa planlamalarla kolektif olarak yapılmaktadır.	*Scrum* Ustası olan Proje Yöneticisine göre uzaktan çevik çalışma maliyetlerini düşürmek için *Discord* gibi ücretsiz araçlardan yararlanmaktadırlar.
BT2	Proje Yöneticisine göre çevik yazılım geliştirme uygulamalarıyla ve ekip üyeleriyle etkili bir şekilde çalışabilmektedirler.	Proje Yöneticisine göre çevik yazılım geliştirme pratikleri müşteri iletişimi geliştirmiştir ve güçlendirmiştir. Ayrıca müşteri ilişkileri yeni dijital araçlarla desteklenmiştir.	Proje Yöneticisi görevler çalışanların uzmanlığına ve gönüllüğüne göre paylaştırılmaktadır.	Proje Yöneticisine göre çevik yazılım geliştirme için yeni dijital araçlara, çalışan ve yönetici eğitimine finansal olarak yatırım yapılmıştır
BT3	Çalışana göre çevik yazılım geliştirme nedeniyle sürekli seremonilerle uğraşmak ve toplantıdan toplantıya koşmak biraz yorucudur.	Çalışana göre çevik çalışmayı yeni öğrenen yazılım geliştiricilere verilen sınırlı otonomi, öğrenme sürecinin geliştirilmesine katkı sağlarken yakın kontrol çalışanı biraz olumsuz etkilemektedir.	Çalışana göre çevik çalışma nedeniyle eğitim/ beceri gereksinimleri arttı. Hem de yazılım geliştiricilerin müşterilerle görüşmek zorunda kalması mevcut iş yükünü artırmıştır.	Çalışana çevikleşen pratiklerin iş ilişkileri, prosedürler ve örgütsel kaynaklar üzerindeki etkisini bilmemektedir.
BT4	Çalışana göre otonom BT ekipleri çevikleşen BT pratikleriyle ve birbirleriyle ekip içinde birlikte çalışabilmektedirler.	Çalışan çevik çalışmadaki artan otonomiden memnundur. Ayrıca çevik çalışmada ustalaştıkça otonomisinin artacağını bilmektedir.	Çalışana göre çevik çalışma için kendisinden beklenen özel beceriler bulunmamaktadır.	Çalışana çevik uygulamanın iş ilişkileri, prosedürler ve örgütsel kaynaklar üzerinde herhangi bir etkisi olmadığını düşünmektedir.

Tablo 12 incelendiğinde katılımcıların çevik yazılım geliştirme uygulamaları ile nasıl kolektif olarak eyleme geçtikleri dört alt tema açısından görülebilmektedir. Birinci alt tema "etkileşimli çalışabilme" (aktörlerin işi operasyonel hale getirmesini zorlaştıran ve kolaylaştıran faktörler belirlenmektedir) (May ve Finch, 2009:544) üzerine örnek görüşler aşağıda verilmiştir:

> "*Ekibin bir kısmı iş yerinde bir kısmı uzaktan çalıyor. Bu nedenle ekip olarak çalışırken ekran paylaşımı yapan open source bir uygulama kullanıyoruz (Discord). Mikrofonu açıyoruz bütün gün hiç kapatmıyoruz. Herkes yan yana çalışıyormuş gibi birbirini görebiliyor…İsterse ekranda yazışabiliyor."* (BT1)
>
> "*Sistemde (Azure Devops) online iş listesi var. Herkes kimin hangi aşamada olduğunu ve ilerlemeleri sistemden takip edebiliyor."* (BT2)
>
> "*Eski ve yeni yazılım geliştirme yöntemlerini birlikte kullanıyoruz*." (BT2)
>
> "*Her sabah saat on birde yaklaşık on dakika Scrum toplantıları yapıyoruz. Herkes önceki gün ne yaptığını ve bugün ne yapacağını birbirine söylüyor. Ekip olarak ilerlemeleri ve işin neresinde olduğumuzu sürekli görebiliyoruz*." (BT1)

İkinci alt tema "ilişkisel entegrasyon" (yeni uygulamanın etrafındaki çalışanlar tarafından anlaşılmasıdır) (örn. hesap verebilirliği ve güveni oluşturmak) (Carroll vd., 2023a:273; May ve Finch, 2009:545) üzerine örnek görüşler aşağıda verilmiştir:

> *"Müşterileri dinlemek için onlarla yüz yüze görüşmeye gidiyorum…Diğerleri de [diğer yazılım geliştiriciler de] gidiyor, gitmesi de lazım…"* (BT2)
>
> *"Geçen yıl çevik çalışmayı öğrenme aşamasındayken bana çok sınırlı bir otonomi verilmişti. Sürekli ne yaptığım kontrol ediliyordu…Ama bu yıl daha fazla otonom hale geldim. Herkesle çok daha iyi anlaşabiliyorum."* (BT3)

Üçüncü alt tema "yeni beceri setiyle çalışabilme" (yeni becerileri ve iş bölümünü etkileyen faktörler incelenmektedir) (Carroll vd., 2023a:273) üzerine örnek görüşler aşağıda verilmiştir:

> *"Çalışanın çevik kabiliyetlerini artırmak için ekip olarak yardımcı olmaya çalışıyoruz. Gerekirse hatalarından ders almasını kolaylaştırmak için psikolojik destek veriyoruz. Ayrıca Udemy'den veya istediği bir yerden eğitim abonelikleri hediye ediyoruz."* (BT1)
>
> *"Eskiden yazılım geliştirici olarak sadece yazılımı geliştirirdik şimdi çevik yüzünden yazılımı geliştirirken müşteriyle de görüşmek zorunda kaldık."* (BT3)
>
> *"Çevik çalışmak için ya benden beklenen olan özel bir beceri yok ya da benim haberim yok"* (BT4)

Dördüncü alt tema "bağlamsal entegrasyon" (yeni uygulamaların mevcut örgütsel yapıları, prosedürleri ve kaynakları nasıl etkilediği incelenmiştir) (Carroll vd., 2023a:273) üzerine örnek görüşler aşağıda verilmiştir:

> *"Uzaktan çevik yazılım geliştirme maliyetlerimizi azaltmak için Discord gibi ücretsiz araçlardan yararlanıyoruz."* (BT1)
>
> *"Çevik yazılım geliştirmenin bir maliyeti var. Başta DevOps lisansı satın alındı. Daha sonra uzaktan çalışanları Scrum toplantılarına daha iyi bir şekilde bağlamak için çeşitli telekonferans araçları alındı*." (BT2)
>
> *"Ben çevik yazılım geliştirmenin finansal olarak desteklenip desteklenmediğini bilmiyorum."* (BT4)

3.4. Çevik Yazılım Geliştirme Uygulamasının Yeniden Konfigürasyonu için Değerlendirilmesine İlişkin Bulgular

Yeni uygulamanın (örn. çevik) yeniden ayarlanması için çalışanlar tarafından yapılan değerlendirme çalışmalarını zorlaştıran ve kolaylaştıran faktörler incelenmektedir (Carroll vd., 2023a: 274). Katılımcılara Tablo 13'te yer alan tematik sorular sorulduğunda Tablo 14'te bulunan dört alt tema (kategori) ortaya çıkmıştır.

Tablo 13: Çevik yazılım geliştirme uygulamasının değerlendirilmesi ve iyileştirilmesiyle ilgili tematik sorular

No	Çalışanlara (Ç) ve/veya Yöneticilere (Y) Sorulan Sorular	
33	Y	Çevik yazılım geliştirme uygulamasının olgunluğunu nasıl değerlendiriyorsunuz?
34	Ç Y	Çevikleşen yazılım geliştirme uygulamasını değerli buluyor musunuz? Neden?
35	Ç Y	Çevik yazılım geliştirme uygulamasının faydalarını nasıl değerlendiriliyorsunuz?
36	Ç Y	Çevik yazılım geliştirme uygulamasını değerlendirmek için hangi metrikleri benimsediniz?
37	Y	Çevik yazılım geliştirme uygulamasının değerini çalışanlara iletilmek için ne tür bilgi paylaşımı yöntemlerini kullanıyorsunuz?
38	Y	Çevik yazılım geliştirme uygulamasının değerini çalışanlara iletmek için kanıtları nasıl oluşturuyorsunuz?
39	Ç Y	Çevik yazılım geliştirme için çevik yöntemlerle ve dijital araçlarla çalışmanın üzerinizdeki etkisi nedir?
40	Y	Çevik yazılım geliştirme uygulamasının faydalarını değerlendirebilmek için ekip üyelerinin birlikte çalışmasını nasıl destekliyorsunuz?
41	Ç Y	Çevik yazılım geliştirme uygulamasını sürdürmek için prosedürleri nasıl yeniden tanımlıyorsunuz? Sonda: Yazılım geliştirme uygulamasını nasıl çevikleştiriyorsunuz
42	Ç Y	Çevik yazılım geliştirme uygulamasını sürdürmek ve ivme kazanmak için hangi teknikleri benimsiyorsunuz?
43	Ç Y	Çevik yazılım geliştirme uygulamasını sürdürmek ve gelecekteki performansını iyileştirmek için ortaya çıkan yeni teknolojileri nasıl değerlendiriyorsunuz?
44	Ç Y	Ekibin çevik yazılım geliştirme uygulamasını sürdürmesi ve değişen yazılım geliştirme uygulamasına bağlı kalması için ne gibi teşvikler sunuluyor?

Kaynak: Carroll vd., 2023a: 296-298'den esinlenilmiştir.

Görüşmeler sonucunda elde edilen verilerden katılımcıların çevik çalışmayı yeniden ayarlamak için değerlendirme yapmaları konusundaki görüşlerinin analiziyle elde edilen alt temalar Tablo 14'te bulunmaktadır. Tablo 14 incelendiğinde NSK'nın sistemleştirme, kolektif değerlendirme, bireysel değerlendirme ve yeniden konfigürasyon alt temalarını (kategorilerini) içerdiği görülmektedir.

Tablo 14: Katılımcıların çevik çalışmayı refleksif olarak izlemelerine ilişkin görüşlerinin oluşturduğu alt temalar

Tema	Alt Temalar (Kategoriler)	Katılımcılar
Refleksif Gözlem	Sistemleştirme	BT1, BT2, BT3, BT4
	Kolektif Değerlendirme	
	Bireysel Değerlendirme	
	Yeniden Konfigürasyon	

Katılımcıların çevik çalışmayı refleksif izlemeyi oluşturan alt temalarla ilgili görüşlerine ait karşılaştırmalı analizler Tablo 15'te yer almaktadır. Tablo 15 incelendiğinde katılımcıların refleksif izlemelerini zorlaştırıcı ve kolaylaştırıcı faktörler görülmektedir.

Tablo 15: Çalışanların ve yöneticilerin çevik yazılım geliştirme faaliyetlerini iyileştirme amacıyla değerlendirilmelerine ilişkin karşılaştırmalı analiz sonuçları

	Refleksif İzleme			
	Sistemleştirme	**Kolektif Değerlendirme**	**Bireysel Değerlendirme**	**Yeniden Konfigürasyon**
BT1	*Scrum* Ustası olan Proje Yöneticisine göre çevik yazılım geliştirme yöntemlerinden *Scrum* on kişilik bir ekip ile yazılım geliştirmek için etkili ve çok kullanışlıdır.	*Scrum* Ustası olan Proje Yöneticisine göre ekibin uyguladığı çevik yazılım geliştirme pratiği başta müşteri memnuniyeti, yazılım testleri, ilerleme hızı, geçen zaman gibi metriklerle sürekli olarak değerlendirilmektedir. Çeşitli ISO değerlendirmeleri de bunlardandır.	*Scrum* Ustası olan Proje Yöneticisine göre uzaktan çevik yazılım geliştirme yapan bir çalışanın iletişim becerisinin kuvvetli olması gerekmektedir.	*Scrum* Ustası olan Proje Yöneticisine göre ekip olarak her konuda (çevik yazılım geliştirme dahil) yeniden yapılandırma amacıyla değerlendirme yapılabilmektedir
BT2	Proje Yöneticisine göre çevik çalışma yöntemleri sayesinde her ay fazladan 4-5 günü doküman yazmak için harcamıyoruz.	Proje Yöneticisine göre çevik çalışmayı değerlendirmek için zaman ve hata odaklı performans metrikleri esas alınmaktadır.	Proje Yöneticisine göre çevik yazılım geliştirme için kullanılan dijital araçlardan çok az yararlanılmaktadır.	Proje Yöneticisine göre herhangi çalışan uygulanan çevik çalışmayı yeniden yapılandırmak için formel bir değerlendirmede bulunmamıştır.
BT3	Çalışana göre eskiden bir köşede tek başına bir haftada geliştirdiği kodu çevik yöntemlerle üç günde geliştirebiliyor.	Çalışana göre çevik yazılım geliştirme faaliyetleri ikili toplantılarda ve grup toplantılarında (*Sprint* değerlendirme toplantıları ve retrospektifler) formel ve enformel olarak değerlendirilmektedir.	Çalışana göre çevik yazılım geliştirmeyi öğrenmek zaman alıcıdır ve fazla performans gerektirmektedir.	Çalışana göre kişi çevik yazılım geliştirmeyi değerlendirmek veya yapılandırmak istememektedir
BT4	Çalışana göre çevik uygulamalar kişinin işi küçük parçalara bölerek daha hızlı tamamlamasını kolaylaştırıyor.	Çalışana göre çevikleşen BT pratikleri ekip içinde enformel olarak değerlendirilmektedir.	Çalışana göre ekipte *Scrum Master* eksikliği vardır. Ekip her kararı uzun süre tartışarak çok zaman kaybetmektedir.	Çalışana göre yeniden yapılandırma amacıyla değişen BT pratikleri değerlendirilmemektedir.

Tablo 15 incelendiğinde katılımcıların çevik yazılım geliştirme uygulamalarını nasıl refleksif olarak izledikleri dört alt tema açısından görülebilmektedir. Birinci alt tema "sistemleştirme" (ekip üyelerinin yeni uygulamanın kendileri için ne kadar etkili ve kullanışlı olduğunu kurumsal açıdan değerlendirmeleridir) (Carroll vd., 2023a:274; May ve Finch, 2009:545) üzerine örnek görüşler aşağıda verilmiştir:

> *"Çevik (Scrum) olmadan on- on beş kişilik bir ekip olarak nasıl yazılım geliştirme yapabilirdik bilmiyorum. Son beş yıldaki başarılarımızı değerlendirdiğimizde çevik yazılım geliştirme yöntemlerinin bizim için çok etkili ve kullanışlı olduğu söylenebilir."* (BT1)

> "*Eskiden detaylı dokümantasyon yazmakla uğraşırdık. Neredeyse her ay 8-10 günü geliştirdiğimiz yazılımın dokümantasyonunu hazırlamak için harcardık. Çevikte bu süre net yarıya indi."* (BT2)

İkinci alt tema "kolektif değerlendirme" (katılımcıların deneyimlerine dayanarak yeni uygulamanın çıktılarını ve değerini kolektif olarak değerlendirmeleridir) (Carroll vd., 2023a:274; May ve Finch, 2009:546) üzerine örnek görüşler aşağıda verilmiştir:

> *"Müşteri memnuniyeti, proje süreci takibi, proje takibi, Sprint tamamlama süreleri, hız…Çalışanı sık sık değerlendiriyoruz ve geliştiriyoruz. Yazılım Test sonuçları, otomatik ve manuel testler hepsini değerlendirme kriteri olarak kullanıyoruz…Çıktılarımızın kalitesini ve çevik çalışma kabiliyetimizi sürekli olarak iyileştiriyoruz."* (BT1)

> *"Çevik proje geliştirme etkililiğimizi geliştirmek için yazılım testlerinden ve tahmini tamamlama sürelerinden yararlanıyoruz."* (BT2)

Üçüncü alt tema "bireysel değerlendirme" (bireylerin deneyimlerine dayanarak yeni uygulamanın çıktılarını ve değerini bireysel olarak değerlendirmeleridir) (Carroll vd., 2023a:274; May ve Finch, 2009:546) üzerine örnek görüşler aşağıda verilmiştir:

> *"Çevik yöntemleri sürekli değerlendirmek için daha fazla zaman ve çaba harcamamız gerekiyor…Bunun için zaman ayırmakta zorlanıyoruz." (BT3)*

> *"Ekip içinde her kararı sürekli uzun süre tartışıyoruz…Ekipte Scrum Master eksikliği olabilir." (BT4)*

Dördüncü alt tema "yeniden konfigürasyon" (yeni uygulamanın kullanımı veya kullanışlılığı ile ilgili hangi özelliklerinin değiştirilmesi veya yeniden yapılandırılması hakkındaki düşünceler) (Carroll vd., 2023a:274; May ve Finch, 2009:546) üzerine örnek görüşler aşağıda verilmiştir:

> *"Küçük ölçekli bir yazılım geliştirme ekibi olarak henüz her konuda kolektif olarak sürekli geliştirme ve iyileştirme yapabiliyoruz…Kişinin sağlam bir gerekçesi olduğu sürece istediği herhangi bir şeyi düzeltebilir. Herkes gördüğünü düzeltmezse gelişme yolumuz kapanır."* (BT1)

> *"Çevik ile ilgili herhangi bir şeyi değerlendirmekle veya geliştirmekle ilgilendiğimi sanmıyorum…Bence bunlar yazılım geliştiricilerin işi olmamalıdır…Scrum Ustası değerlendirmeyi ve geliştirmeyi en iyi şekilde yapıyor başka işi yok zaten." (BT3)*

Çevik yazılım geliştirmenin çalışanlar tarafından değerlendirilmesi ve sürekli iyileştirilmesi konusuna BT1 (yönetici) olumlu bir görüş bildirirken BT3 (çalışan) olumsuz bir görüş belirtmiştir.

Sonuç ve Tartışma

Günümüzde coğrafi olarak dağıtık yüksek hızlı sanal sunuculara ve değişime hızlı cevap vermeye odaklanan çevik/yalın yöntemler birbirini tamamlayan iki büyük dijital dönüşüm trendi olarak öne çıkmıştır (Adam, 2023:61; Athambawa vd., 2023:20-22; OECD, 2019:3; Ogala ve Mughele, 2022:68). Çevik/yalın düşünce tarzı pandemi ve sonrasında başta geleneksel çalışma biçimlerine uygulanarak yeni normal haline gelmiştir (Adam, 2023:61; Rad ve Rad, 2021:339). "Çevik yöntemler yazılım geliştirme faaliyetlerinin %90'ından fazlasında kullanılmaktadır ve değişen gereksinimlere yanıt verme …, daha yüksek yazılım geliştiricisi üretkenliği, refahı ve otonomisi gibi iyi bilinen faydalar sağlamaktadır" (Carroll vd., 2023a:267). Ancak bu durum uygulamada ve teoride iki önemli probleme yol açmıştır. Birincisi, çevik düşünce tarzı ile geleneksel yönetim yaklaşımları arasında paradigma değişimi olduğu için çevik düşünce tarzını geleneksel yönetim yaklaşımlarına uygulamak kolay değildir. Bu uygulama sürecinde çok sayıda problemle ve değişime karşı dirençle karşılaşılmaktadır (Reginaldo ve Santos, 2020:5-6). İkincisi, çevik gibi yeni normların geleneksel yönetim yaklaşımlarına uygulanması, (rutin olarak) yerleştirilmesi ve bütünleştirilmesi için büyük ölçüde kabul görmüş ortak bir kuram henüz yoktur (Carroll ve Conboy, 2020:1). Literatürde "çevik pratiklerin asimilasyonu, rutinleştirme, değişim yönetimi, farkındalık yaratma, kurumsallaştırma ve dinamik yetenekler" gibi kuramlar olsa da Yazılım Geliştirme uygulamalarının büyüyen ölçeği bu kuramların etkililiğini ve varsayımlarını sınırlandırmaktadır (Carroll vd., 2023a:270). Bu alanda yapılan yazın taramalarına göre çevik yöntemlerin uygulamaları üzerine yapılan çalışmaların sıklıkla kuramsal temelleri eksiktir, kullanılan teoriler zayıf varsayımlara dayanmaktadır ve "çevik yöntemlere basit bir şekilde bağlı kalmaya aşırı derecede güvenmek çoğu zaman sorun yaratmaktadır" (Carroll vd., 2023:267). Diğer yandan yinelemeli bir normalleşme (uygulamanın rutinleşmesi) ile bir şeye basit bir şekilde bağlı kalmanın ve yanlış varsayımların ötesine geçilebilmektedir (Carroll vd., 2023:270). Bu bağlamda bu çalışmanın savı çevik düşünce tarzının (değerler, prensipler, pratikler) ve çevik yöntemlerin (örn. *Scrum*) geleneksel Yazılım Geliştirme pratiklerine (rutin olarak) yerleştirilmesi, uygulanması ve bütünleştirilmesi için büyük ölçüde kabul görmüş ortak bir kuramın olmamasıdır. Carroll ve Conboy (2019:1-12); Carroll ve diğerleri (2023a:99-127), Carroll ve diğerleri (2020:75-83) bu tür sorunlara çözüm olabilecek yollardan birisinin Normalleşme Süreci Kuramı (NSK) olduğunu ifade etmektedir (örn. May ve Finch, 2009: 538-540). NSK başta sağlık bilimlerinden uygulama bilimlerine kadar çeşitli alanlarda yeni normların uygulanmasını, yerleştirilmesini, entegre edilmesini (veya sürdürülmesini) açıklamaktadır (Finch vd., 2013: 1; Huddlestone vd., 2020). Carroll ve diğerleri (2020:82) biri başarılı diğeri başarısız iki çevik dönüşüm vakasını NSK merceğinden inceledikleri bir çalışmada özellikle anlamlandırma ve katılım boyutlarında yer alan farklılaşmanın, gerekçelendirmenin, değerlerin ve rollerin katılımcılar tarafından daha iyi anlaşılması için daha fazla çabalanması gerektiğini tespit etmiştir. Carroll ve diğerleri (2023b:348) dönüşüm uygulamalarında sürdürülebilirliğe yatırım yapılmamasının sürekli normalleşmenin önüne geçeceğine işaret etmektedir. Beck (2000) çevik yöntemleri anlatırken küçük ölçekli yazılım ekiplerine odaklandığı için özellikle büyük ölçekli yazılım geliştirme uygulamaları için yeterli açıklamalarda bulunamamıştır.

Araştırma sonuçlarına göre çevik yazılım geliştirme uygulamalarının daha fazla normalleşmesi için "anlamlandırmaya", "katılıma", "uygulamaya" ve "değerlendirmeye" daha fazla yatırım yapılması gerekmektedir. Literatürdeki bulgular bunu desteklemektedir. Luig ve diğerlerinin (2018:431-432) yaptığı nitel bir çalışmada uygulamaları kolaylaştırmanın, anlamlandırmanın ve yeniliğin (müdahalenin) benimsenmesini desteklemenin, uygulama süreçlerinin yinelemeli değerlendirmesinin önemli olduğunu kanıtlamıştır. Carroll ve diğerleri (2020) NSK merceğinden yaptığı görgül çalışmada anlamlandırma yapısının karmaşık müdahalelerin başarısı üzerinde belirleyici etkisi olduğunu göstermiştir. Araştırma bulgularına göre çevik yazılım geliştirmenin normalleşmesini kolaylaştıran başlıca faktörler arasında çevikten

yararlanma nedenlerinin çalışanlara net bir şekilde iletilmesi, rollerin belirlenmesi, çevik bilgi-beceri eksikliklerinin giderilmesi ve çevik uygulamaların konfigürasyonu için çalışan değerlendirmeleri bulunmaktadır. Bunlar literatür ile uyumlu olsa da yapılması beklenen değerlendirmelerin ve iyileştirmelerin çalışanlar üzerindeki iş yükünü artırdığı bir gerçektir (Carroll vd., 2020; Carroll vd.,2023b; Luig vd. 2018). Diğer yandan çevik yazılım geliştirme uygulamalarının normalleşmesini zorlaştıran faktörlerden bazıları çalışanların ve yöneticilerin çeviği farklı şekillerde anlamlandırmaları ve çalışanların çevik uygulamalara dahil olmalarının meşruiyetinden şüphelenmeleridir. Literatürdeki bulgular bunu desteklemektedir. Eurofound (2021:2-27) söz konusu zorlaştırıcı faktörlerin çevik uygulamaların etkili bir şekilde uygulanmasını engelleyebileceğini işaret etmektedir. Axelos, (2019) ve ITIL 4 (2019) çevik yazılım geliştirme uygulamaları için etkililiği kanıtlanmış hazır iyi uygulamalar (*best practices*) çözümleri önermektedir.

Yazılım geliştirme uygulamalarının normalleşmesi için katılımcıların deneyimlerine dayanan geri bildirimlerinin yeniden konfigürasyon amacıyla değerlendirilmesi önem arz etmektedir. Çalışan geri bildirimleri her zaman yeniden konfigürasyon amacıyla kullanılmamaktadır. Bu durumun yol açtığı memnuniyetsizliği ve zorlukları aşmak için düzenli olarak çalışanlardan geri bildirim istemek ve buna göre hareket etmek, örgütlerin bilinçli ayarlamalar yapmasına yardımcı olmaktadır. Bunla ek olarak çevik yazılım geliştirme uygulamalarının güncel ve etkili kalmasını kolaylaştırmaktadır. Diğer yandan, çevik yazılım geliştirme uygulamalarına olan katılımı desteklemek için oluşturulan eğitim kaynakları ile katılımcılara kısa ve sistemli eğitimler sunulmalıdır. Eğitim, rollerin netleştirilmesi ve çalışanları çevik uygulamaları sahiplenmeye teşvik eden destekleyici bir ortamın geliştirilmesi gerekmektedir. Çevik yazılım geliştirme uygulamalarının geleneksel uygulamalardan farkının ve öneminin tüm paydaşlar tarafından tam ve doğru bir şekilde anlaşılması ve algılanması sağlanmalıdır (Eurofound, 2021). Böylece yöneticilerin stratejik odaklanması ile çalışanların operasyonel uygulamaları arasında daha etkili bir uyum yakalanabilmektedir. Ayrıca çevik yazılım geliştirme uygulamalarının normalleşmesini zorlaştırmamak için yeni uygulamaların (normların) amacı ve faydaları konusunda ortak bir anlayış oluşturmayı destekleyecek yeterli sayıda kilit çalışan, lider ve koç olduğundan emin olunmalıdır.

Hakem Değerlendirmesi: İki bağımsız hakem tarafından değerlendirilmiştir.
Çıkar Çatışması: Yazar çıkar çatışması bildirmemiştir.
Mali Destek: Yazar bu çalışma için mali destek almamıştır.

Peer Review: Externally peer-reviewed.
Conflict of Interest: The author declares no conflict of interest.
Grant Support: The author did not received financial support for this study.

Kaynakça

Adam, P.A. (2023) *Outlook: Agile as the New Normal?* Agile in ISO 9001 (Ed.) Adam, P.A. Switzerland.: Springer, 61-67.

Akar, A. (2020) Sağlık Turizmi Kapsamında Termal Oteller ve Dış Turizm Talebinin Yapısal Analizi: Aydın İli Örneği. Yüksek Lisans Tezi, Adnan Menderes Üniversitesi, Sağlık Bilimleri Enstitüsü ve Sağlık Turizmi Anabilim Dalı.

Athambawa, A., Gapar, J. Md. ve Khatibi, A. (2023) *COVID-19 and the Level of Cloud Computing Adoption: A Study of Sri Lankan Information Technology Organisations*. Journal of Information Engineering and Applications, 13:2, 20-28.

Axelos (2019) ITIL 4 Foundation Revision Guide. UK: The Stationary Office.

Beck, K. (2000) Extreme Programming Explained. Boston: Addison-Wesley.

Carroll, N. (2020) *Theorizing on the Normalization of Digital Transformations*. Twenty- Eight European Conference on Information Systems (ECIS2020). June, 2020. Morocco. 1-17.

Carroll, N., Bjornson, F.O., Dingsoyr, T., Rolland, K., Conboy, K. (2020) *Operationalizing Agile Methods: Examining Coherence in Large-Scale Agile Transformations*. Agile Processes in Software Engineering and Extreme Programming, 8–12 June, Denmark. 75-83.

Carroll, N. ve Conboy, K. (2020) *Normalising The "New Normal": Changing Tech-Driven Work Practices Under Pandemic Time Pressure*. International Journal of Information Management, 55, 1-5.

Carroll, N., Lafferty, B., Conboy, K. ve Donnellan, B. (2021) Normalising a Digital Transformation. Forty-Second International Conference on Information (ICIS). December, 2021. USA.

Carroll, N. ve Maher, M. (2023) How Shell Fueled Digital Transformation by Establishing DIY Software Development. MIS Quarterly Executive. 22:2.

Carroll, N., Conboy, K. ve Wang, X. (2023a) From transformation to normalisation: An exploratory study of a large-scale agile transformation. Journal of Information Technology, 38:3, 267–303.

Carroll, N., Hassan, N.R., Junglas, I. Et. All. (2023b) *Transform or be transformed: the importance of research on managing andsustaining digital transformations*, European Journal of Information Systems, 32(3), 347-353.

Cohen, D., Lindvall, M. and Costa, P. (2003) *Agile Software Development A DACS State- of-the-Art Report*. Data and Analysis Center for Software (DACS). University of *Maryland*.

Eurofound (2021) *The digital age: Implications of automation, digitisation and platforms for work and employment*, Challenges and prospects in the EU series (Ed.) Eurofound, Luxembourg: Publications Office. 1-27

Finch, T.L., Rapley, T., Girling, M. et al. (2013) *Improving the normalization of complex interventions: measure development based on normalization process theory (NoMAD): study protocol*. Implementation Science, 8 (43), 1-8.

Huddlestone, L., Turner, J., Eborall, H., et. All. (2020) *Application of normalisation*

process theory in understanding implementation processes in primary care settings in the UK: A systematic review, BMC Family Practice, 21 (52), 1-16.

ITIL 4 (2019) ITIL Foundation ITIL 4 Edition. United Kingdom: The Stationery Office.

Luig, T., Jodie A., Arya M. S. Et. All. (2018) *Understanding Implementation of Complex Interventions in Primary Care Teams*, *The Journal of the American Board of Family Medicine*, 31 (3), 431-444.

May, C.R., Finch, T., Mair, F. Et. All. (2007). *Understanding the implementation of complex interventions in health care: the normalization process model*. BMC Health Services Research, 7(148), 1-7.

May, C.R. ve Finch, T. (2009) *Implementing, embedding, and integrating practices: an Outline of normalization process theory*, Sociology, 43:3, 535-554.

Normalisation Process Theory (2024) "*Core Propositions of NPT*" [https://normalization- process-theory] (Erisim Tarihi: 14.01.2023)

OECD (2019) Measuring the Digital Transformation: A Roadmap for the Future. Paris: OECD Publishing.

Ogala, J. ve Mughele, S. (2022) *Agile Software Development Methodologies in Cloud Computing*. Requirement Enginering in the Cloud. Nigeria: ACity-IEEE- SMART-- Foundations Series, 67-80.

Oludapo, S., Carroll, N., ve Helfert, M. (2024) Why do so many digital transformations fail? A bibliometric analysis and future research agenda. Journal of Business Research, 174, 1-17.

Pacini, A., Stickland, A., Iniguez, K. Et. All (2023*) Implementation of home practice support strategies for older adults attending an online mindfulness based cognitive therapy course: An adaptive intervention protocol*. 5th UK Implementation Science Research Conference, The Open University. July 2022.

Pope, C., Halford, S. Turnbull, J. Et all. (2013) *Using computer decision support systems in NHS emergency and urgent care: ethnographic study using normalisation process theory*. BMC Health Services Research. 13(111), 1-13.

Rad, D. ve Rad, G. M. (2021) *Going Agile, a Post-Pandemic Universal Work Paradigm: a Theoretical Narrative Review*. Postmodern Openings, 12(4), 337-388.

Reginaldo, F. ve Santos, G. (2020) *Challenges in Agile Transformation Journey: A Qualitative Study*. 34th Brazilian Symposium on Software Engineering. 19-23 October 2020, Natal.

Schwaber, K. (2004) Agile Software Development with Scrum. USA: Microsoft Press.

Sommerville, I. (2016) Software Engineering. Boston: Pearson Education.

Wysocki, R.K. (2009) Effective Project Management: Traditional, Agile, Extreme. Canada: Wiley.