Bilgi Sistemi Yazılım Geliştirme Yaşam Döngüsü Safhalarından Gereksinim Belirleme ve Sistem Tasarımında Kalite Odaklılık: Üç Proje İncelemesi

Araştırma Makalesi/Research Article

Doğan YILDIZ

Bilgi Teknolojileri Başkanlığı, Türk Havacılık ve Uzay Sanayii A.Ş. (TUSAŞ), Ankara, Türkiye
dyildiz2000@gmail.com
(Geliş/Received:30.01.2021; Kabul/Accepted:27.12.2021)
DOI: 10.17671/gazibtd.871411

Özet— Yazılım geliştirme yaşam döngüsü safhalarından gereksinim belirleme ve tasarım safhasının önemi, sistem geliştirmenin ilk başında olması ve bu safhada yapılacak olan bir hatanın maliyeti ilerleyen safhalarda veya işin bitiminde fark edilmesi durumunda çok pahalıya mal olmasındandır. Yazılım geliştirmenin bu safhasındaki tüm faaliyet ve teknikleri aktaran iş analizi bilgi birikimi (Business Analysis Body of Knowledge- BABOK)) dokümanındaki teknikler ve yöntemlerden faydalanılarak çalışma şekillendirilmiştir. Bu çalışmanın amacı, bu safhada yapılan çalışmaların ve ortaya çıkan ürünlerin kalite gereklerine uyumunun nasıl sağlanabileceğini göstermektir. Bunun için bu çalışmada yöntem olarak farklı standart ve çalışmalar incelenmiş ve bu çalışma kapsamında incelenen projeler için özel kalite gereksinimleri konularak değerlendirme yapılmıştır. Kalite gereksinimleri projeden projeye değişebileceği için özellikle standart veya daha önce ortaya konulan kalite gereksinimleri doğrudan alınmamıştır. Çalışma sonucunda, üç farklı projede, kalite kriterlerine ne derece uyulup uyulmadığı analiz edilmiş ve ortaya konulan kalite gereksinimlerinden özellikle açıklık gereksinimini her üç projenin de ihlal ettiği görülmüştür.

Anahtar Kelimeler— gereksinim belirleme, sistem tasarımı, yazılım geliştirme yaşam döngüsü, gereksinim kalite kriterleri

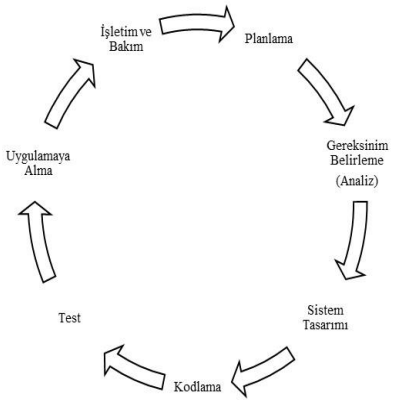
Quality Focused in Determining the Requirements and System Design from the Information System Software Development Life Cycle Phases: Review of Three Projects

Abstract— The importance of the requirement determination and design phase, which is one of the software development life cycle phases, is that it is at the beginning of the system development and the cost of an error to be made in this phase is very expensive if it is noticed in the later stages or at the end of the work. The study was shaped by utilizing the techniques and methods in the Business Analysis Body of Knowledge (BABOK) document, which conveys all the activities and techniques at this stage of software development. The purpose of this study is to show how the work done at this stage and the resulting products can comply with the quality requirements. For this, different standards and studies have been examined as a method in this study and an evaluation has been made by setting special quality requirements for the projects examined within the scope of this study. Since the quality requirements may vary from project to project, especially the standard or previously stated quality requirements were not taken directly. As a result of the study, it was analyzed to what extent the quality criteria were complied with in three different projects and it was observed that all three projects violated the quality requirements, especially the clarity requirement.

Keywords— requirement determination, system design, software development life cycle, requirement quality criteria

1. GİRİŞ (INTRODUCTION)

Yazılım, değişik ve çeşitli görevler yapma amaçlı tasarlanmış elektronik aygıtların birbirleriyle haberleşmesini ve uyumunu sağlayarak görevlerini ya da kullanılabilirliklerini geliştirmeye yarayan makine komutlarıdır [1]. Yazılım geliştirme, tanımda yer aldığı şekilde bir yazılımın geliştirilmesi için gerekli aktivitelerin yer aldığı faaliyettir. Yazılım geliştirme süreci gereksinim belirleme ve sistem tasarımı, yazılımın kodlaması ve sistem testleri şeklinde üç ana safhaya bölünebilir. En önemli safha gereksinim belirleme ve ardından sistem tasarımının ortaya konulmasıdır. Eğer bu safhada gereksinimler doğru ve net bir şekilde belirlenmez ise ortaya çıkan ürün, ihtiyaç sahibinin ihtiyacını tam olarak karşılamayacaktır. Hatta bu aşamadaki yanlışlıkların ileride geri dönüşü çok maliyetli olacaktır. Yazılım geliştirme süreci, daha geniş ve resmi bir tanım ile yazılım geliştirme yaşam döngüsü (Software Development Life Cycle – SDLC) şeklinde adlandırılmaktadır. Şekil 1'de görüleceği üzere içeriğinde planlama, gereksinim belirleme (analiz), sistem tasarımı, kodlama, test, uygulamaya alma, işletim ve bakım şeklinde adımları olan bir süreçtir. Şelale modeli, V model, döngüsel model, çevik model gibi farklı yazılım geliştirme yaşam döngüsü modelleri yer almaktadır.



Şekil 1. Yazılım geliştirme yaşam döngüsü
(Software Development Life Cycle)

Gereksinim belirleme ve sistem tasarımı safhasının önemi, çevik yazılım geliştirme yaşam döngüsünde de aynıdır şelale yaşam döngüsünde de aynıdır. Çevik yazılım geliştirmede bu safha da yapılan bir yanlışlık daha erken aşamada fark edileceği için geriye dönük düzeltmeler daha hızlı bir şekilde yapılabilecektir. Şelale döngüsünde ise bu safhada yapılan yanlışlığın fark edilmesi, projenin daha çok ilerleyen aşamalarında fark edileceği için bedeli daha ağır olacaktır. Ayrıca, her sektörün kendine has kritikliği ve önemi olduğundan dolayı tüm sektörlerdeki yazılım geliştirme faaliyetlerinde bu safha önem kazanmaktadır. Ancak, bazı sektörlerde bu safhada yapılan hatalar daha maliyetli olabilmektedir. Özellikle kritik ve sonucunun olumsuz olması durumunda bedeli insanın canı ile ödenebilir veya çok yüksek maliyetli yazılımlarda, bu safhadaki gereksinimler net olarak belirlenmeli ve ona göre yazılım geliştirmenin diğer safhalarına geçilmelidir.

Gereksinimlerin belirlenmesi ve buna göre sistem tasarımının ortaya konulması safhasında kalite odaklı bir yaklaşım sergilenmesi her zaman ilerleyen zamanda büyük kazançlar sağlayacaktır. Bunun için bu çalışmada gereksinim belirleme ve sistem tasarımı safhasında ortaya konulan kalite gereksinimlerini ve hazırlanan yazılım gereksinim spesifikasyonunun (Software Requirements Spesification – SRS) kaliteli bir şekilde oluşturulup oluşturulmadığı farklı büyüklükteki projelerde incelenerek sonuçları sunulmuştur.

2. GEREKSİNİM BELİRLEME VE SİSTEM TASARIMI (REQUIREMENT IDENTIFICATION AND SYSTEM DESIGN)

Gereksinimlerin belirlenmesi aşaması yazılım geliştirme yaşam döngüsünde geliştirilecek olan sistemin analizinin yapılması şeklinde de tarif edilmektedir. Burada asıl olan kullanıcının veya işin sahibinin ihtiyaçlarının ortaya çıkarılmasıdır. Tasarım ise bu ihtiyacı giderecek olan çözümün ortaya konulmasıdır. Sonuç olarak bu iki adım iç içe geçmiş durumdadır. Bir gereksinimin çözümü olan tasarımı yaparken başka gereksinimler doğabilmekte ve onlar analiz edilmektedir [2]. Analiz safhası özellikle sistem sahibi ve kullanıcılarından, gereksinimlerin görüşme, araştırma, gözlem, anket gibi farklı teknikler ile çıkarılması, bunların önceliklendirilmesi, fonksiyonel ve fonksiyonel olmayan gereksinimler şeklinde ayrıştırılması, mevcut durum (AS-IS) iş süreci modelinin çıkarılması, iş kurallarının, varsayımların ve kısıtların belirlenmesi, kullanılan yönteme göre kullanıcı hikayelerinin belirlenmesi şeklinde ana hatları ile yapılan faaliyetler sıralanabilir. Sistem tasarımı ise her bir gereksinim için sistemde yer alacak olan çözümün bulunması, sistemin kullanıcı ile hemfikir olunacak olan ekran tasarımlarının çıkarılması, konsept veri akışı ve veri modelinin belirlenmesi, prototipleme yapılması, olması gereken (TO-BE) iş süreci modelinin çıkarılması, sistemin iş kurallarının ve kısıtlarının belirlenmesi şeklinde ana faaliyetleri sıralanabilir.

Yazılım gereksinim spesifikasyonu, yazılımı kullanacak olan müşterilerin tam olarak ne elde etmek istediklerini, yazılımı üretenlerin ise müşterilerinin ne istediklerini görmelerine yardımcı olmaktadır. Bunun yanında gereksinimlerin standart bir yapıda tanımlanmasına, yazılım geliştirmenin maliyeti ve zamanının tahmin edilmesine, doğrulama ve geçerleme için temel teşkil etmeye yardımcı olmaktadır. Yazılım gereksinim spesifikasyonu (Software Requirements Spesification-SRS) dokümanının giriş kısmında amaç, kapsam, tanımlar, kısaltmalar yer almaktadır. Genel tanım kısmında ise ürün (yazılım) ile ilgili bakış açısı, ürün fonksiyonları, kullanıcı karakteristikleri, kısıtlamalar, varsayımlar ve bağımlılıklar yer almaktadır. Özel gereksinimler kısmında ise yazılım ile ilgili ihtiyaç olan dış dünya ile etkileşim, fonksiyonlar, performans gereksinimleri, mantıksal veri tabanı gereksinimleri gibi tüm diğer gereksinimlere yer verilmektedir [3].

3. LİTERATÜR TARAMASI (LITERATURE REVIEW)

Literatür taraması iki bakış açısı ile ele alınmıştır. Bunlar, yazılımların kalite gereksinimlerinin tanımlanması ve yazılım gereksinim spesifikasyonunun kaliteli bir şekilde hazırlanması ile ilgili çalışmalar şeklindedir.

Kalite gurularına göre kalitenin tanımına bakıldığında, gereksinimlerin tam ve doğru bir şekilde belirlenmesinin önemi vurgulanmıştır. Crosby, kaliteyi, müşterinin istediği gereksinimlere uyum olarak tanımlamaktadır. Bunu yazılım dünyasına uyarladığımızda yazılım geliştirmenin ilk safhası olan gereksinimlerin tam ve net bir şekilde belirlenmesi gerekir ki sonunda ortaya çıkan ürün, gereksinimleri ve dolayısı ile ihtiyacı karşılıyor olması gerekmektedir. Diğer bir kalite gurusu olan Deming ise kaliteyi, müşteri tarafından belirlenebilecek bir olgu olarak tanımlamıştır. Buna göre, aynı şekilde müşteri gereksinimlerinin ve ihtiyaçlarının tam ve net olarak belirlenmesini ifade etmektedir. Diğer bir guru olan Juran'da kaliteyi, ürünün kullanım için uygun olması şeklinde ifade etmektedir. Aynı şekilde ürünün kullanılabilir olması için ürün ile ilgili olan gereksinimlerin karşılanması gerekmektedir [4]. Kalite güvencenin iki farklı bakış açısı vardır. Bunlar, müşteri gereksinimlerinin karşılanması ve ürün kalitesidir. Bu iki bakış açısına bir tane daha bakış açısı olan yazılım geliştirme süreç kalitesini eklemek faydalı olacaktır. Bunlardan müşteri gereksinimlerinin karşılanması, doğru yazılım geliştirmeye götürecektir. Ürünün kaliteli olması önemlidir, ancak gereksinimi karşılamıyor ise bir anlam ifade etmeyecektir. Dolayısı ile ihtiyacı doğru anlayıp, doğru ürünü elde etmek, bunu yaparken de yazılım geliştirme sürecindeki kalite hususlarına dikkat etmek önem arz etmektedir [5].

Yazılım geliştirme projelerinin üçte ikisindeki başarısızlığın gereksinimlerden kaynaklanmaktadır. Ayrıca, fonksiyonel ve kullanıcı ara yüzlerindeki problemlerin nedeni olarak yetersiz gereksinim mühendisliği ve yönetimidir. Buna etki eden faktörler; gereksinimler ihtiyaç sahibinin gerçek ihtiyaçlarını yansıtmaması, tutarsız gereksinimler, tamamlanmamış gereksinimler, gereksinimler arası çatışma, gereksinimleri yanlış anlama, belirsiz ve anlaşılmaz gereksinimler, istek sahibi tarafından direkt olarak sunulan gereksinimler, sahte gereksinimler ve son olarak amaçsız gereksinimlerdir. Yazılım geliştirme aşamasında ortaya çıkan sorunların çözümlenmesinin maliyeti, bu sorunların hangi aşamada bulunulması ile ilişkilidir. Buna göre erken safhada ortaya çıkan problemler geç safhada çözümlenir ise onu düzeltmek pahalıya mal olacaktır. Bunun yanında gereksinimler önceliklendirilmelidir. Eğer zaman ve farklı nedenlerden dolayı bazı gereksinimler ele alınamayacak olursa bu şekilde önceliklendirme büyük fayda sağlayacaktır. Ayrıca gereksinimlerin fonksiyonel ve fonksiyonel olmayan gereksinimler şeklinde bölünmesi de önemli bir husustur. Ana gereksinim ve ondan türetilen gereksinimler de ayrıca gereksinimleri sınıflandırmak, öncelik vermek ve farklı amaçlar için faydalı olacaktır. Maliyet, zaman (takvim) ve performans üçlüsü önemlidir. Gereksinimlerin zayıf ve kalitesiz bir şekilde belirlenmesi bu üçlü üzerinde olumsuz yönde etki edecektir. Ayrıca, zayıf bir şekilde gereksinim mühendisliği ve yönetimi yapıldığı durumlarda; kabul testlerinde sistemin sorun çıkarabileceği, projenin bitirme tarihinde sorunlar olabileceği, sistem geliştirme maliyetinin bütçenin üzerinde olabileceği, son kullanıcı tarafından performansın istenildiği şekilde olmayacağı, ayrıca sistem tasarımını yapanlar birçok sorun ile karşılaşacağı, son kullanıcının istediği kabiliyette bir sistem olmayacağı veya istenilen şekilde kullanamayacakları veya sistemin kabulünü ret edebilecekleri, sistemin işletime verilmesinden hemen sonra çok fazla değişiklik önerisi geleceği veya bazı özelliklerin değiştirilmesi kaçınılmaz olacağı ve son olarak geliştirme zamanının bu yeni gereksinimlerden dolayı aşılabileceği şeklinde sorunlar olabilecektir [6].

İhtiyaç analizi metodunda sadece belirtilen talepleri değil bunun yanında ima edilen taleplerinde ele alınması gerekmektedir. Alternatifler ve çelişkiler içinde ihtiyacın doğru belirlenmesi, ayrıca farklı yöntemler ile belirlenen ihtiyaçların resmi hale getirilmesi gerekmektedir. Gereksinim mühendisliği metodunda ise gereksinimlerin objektif olarak tanımlanması, geçerleme kriterlerinin belirlenmesi, gereksinim tanımlarının herkes tarafından anlaşılabilir olması, fonksiyonel gereksinimlerin yanı sıra kalite gereksinimlerinin de belirlenmesi, bunun yanında kısıtlarında ayrıca ele alınması gerekmektedir. Kalite, gereksinim mühendisliği metotları ile ilgili ise, kalite gereksinimlerinin kritiklerine göre tüm kalite gereksinimlerinin ele alınması, kalite gereksinimlerinin hem resmi hem de geniş bir şekilde belirtilmesi, ayrıca kalite gereksinimlerinin objektif, ölçülebilir ve değerlendirilebilir olması gerekmektedir [7].

Gereksinim mühendisliği ve doğrulama arasında eşgüdüm incelenmiş ve sonuçta sekiz başlık altında sunulmuştur. Bunlar; organizasyon ve süreçlerle ilgili konular, insanlarla ilgili konular, araçlarla ilgili konular, gereksinimler ve süreçle ilgili konular, test süreciyle ilgili konular, değişim yönetimi ile ilgili konuları, izlenebilirlik konuları, ölçüm konularıdır [8]. İş süreçlerini ve gereksinim mühendisliği sonucunda elde edilen gereksinimleri eşleştirmek sureti ile gereksinimler daha kaliteli ve sürece uygun bir şekilde belirlenebilmektedir [9]. Yazılım mimarının fonksiyonel olmayan gereksinimleri nasıl dikkate almalı adlı anket çalışmasında fonksiyonel olmayan gereksinimler ile ilgili sürdürülebilirlik, tekrar kullanılabilirlik, verimlilik, güvenilirlik ve kullanılabilirlik mimarlar için daha önemli olmasına rağmen, taşınabilirlik, maliyet, standart uyumluluk ve kurumsal fonksiyonel olmayan gereksinimler daha az öneme haiz olmuştur [10].

Yazılımın gereksinimlerinin tanımlamasında olması gereken kalite ile ilgili hususlar, belirli, tam, doğru, anlaşılabilir, doğrulanabilir, dahili olarak tutarlı, harici olarak tutarlı, ulaşılabilir, özlü, tasarımdan bağımsız, izlenebilir, değiştirilebilir, elektronik depolanmış, yürütülebilir / yorumlanabilir, göreli önemle açıklamalı, göreceli istikrar ile açıklamalı, sürüme göre açıklamalı, gerekli, yeterli ayrıntı düzeyinde, hassas, yeniden kullanılabilir, izlenebilir, organize ve ilişkili şeklinde

sıralanmıştır. Ayrıca, yazılımdaki hataların geriye doğru izlendiği durumda, bir çalışmada yüzde 56, diğer bir çalışmada ise yüzde 45 yazılım gereksinim spesifikasyonunda veya erken tasarım safhalarından kaynaklandığı görülmüştür. Ayrıca, hatanın işletim ve bakım safhasında bulunması 200 kat maliyet ortaya çıkardığı da belirtilmiştir. Gereksinimlerdeki hatalar, bilgi hataları ve spesifikasyon hataları olmak üzere iki gruba ayrıştırılmaktadır. Bilgi hatalarının prototipleme ile giderilebilirken, spesifikasyon hataları için ise ilgili gereksinimlerin doğru bir şekilde tariflenmesi gerekmektedir. Yazılım gereksinim spesifikasyonunda kalite hususlarına dikkate almaz isek; ortaya çıkan yazılım, müşteri ihtiyacını karşılamayacak, geliştirici ile müşteri arasında birçok farklı yorumlama yüzünden uyuşmazlıklar olacak, test etmek çok zorlaşacak, yanlış bir sistemin ortaya çıkma durumu olabilecektir. Projeden projeye kalite gereksinimleri farklılık arz edebilmektedir. Kalitenin ulaşılabilir olduğu, mükemmelliğin ise ulaşılmaz olduğu bilinmelidir. Sonuçta, geliştirilen araçların faydaları olduğu ancak hala uzmanların yazılım gereksinim spesifikasyonlarını incelemeleri önemlidir [11].

Gereksinimler, tam, kısa, tutarlı ve anlaşılabilir gibi kalite özellikleri açısından ölçülebilmektedir. Genel olarak, SRS kalitesinin değerlendirilmesi, inceleme oturumları sırasında manuel olarak yapılmakta ve bununla birlikte, değerlendirme süreci büyük ölçüde inceleme yapan uzmanların uzmanlığına bağlıdır. Aslında, uzmanların muhakemesi, deneyim, bilgi ve alan da dahil olmak üzere çeşitli faktörler nedeniyle tutarsız olabilmekte, dolayısı ile uzmanların çalışmalarındaki amaçları SRS kalitesini ölçmek için uygulanabilir kuralları belirlemek ve gereksinim mühendisinin SRS kalitesini artırmasına yardımcı olmaktır. SRS kalite özelliklerinin analizi sonucunda, otomatikleştirilmesi mümkün olan kalite faktörleri olarak gereksinim cümle kalitesi ve gereksinim belge kalitesi olmak üzere iki tür özellik vardır [12].

Boeing şirketinde, gereksinimlerin nasıl yazılacağına dair özel bir dilbilgisi olan "yapılandırılmış doğal dil" kavramı kullanılarak, gereksinim mühendisliği, yüksek kaliteli ve standart bir yaklaşım halini almıştır. Bu yaklaşım, gereksinim mühendislerinin, ISO, ilgili ticari ve askeri standartların gerektirdiği şekilde daha kesin ve doğrulanabilir gereksinimler yazmasına yardımcı olmuştur. Ayrıca, bir gereksinimin standart kriterleri ne kadar iyi karşıladığını değerlendirmek için bir puanlama yöntemi de belirlenmiştir. Sonuçta bu yaklaşım, gereksinim yazarı olan iş analistine, kritik bilgilerin veya eksikliklerin bulunması konusunda geri bildirim sağlamıştır [13].

Gereksinim mühendisliği süreci safhasının yazılım yaşam döngüsünde büyük öneme sahiptir. Bu safhada yer alan farklı yaklaşımlar vardır [14]. Yazılım Gereksinim Spesifikasyonu dokümanındaki (SRS)'in dilsel kalite özelliklerine dayalı SRS cümleleri için bir kalite modeli geliştirilmiştir [15]. ISO 9126 kapsamındaki yazılım ürün kalitesi ana ve alt karakteristiklerini ele alarak inceleme yapılmıştır [16]. ISO / IEC25000 serisi uluslararası standartlar kullanılarak kalite gereksinimi ve değerlendirmeleri yapılmıştır. ISO 25000 serisi altında yer alan tüm standartlar ele alınarak sürecin nasıl çalıştırılması gerektiği aktarılmıştır [17]. Use Case (UC) tabanlı gereksinim oluşturma aşamasında kalite ile ilgili hususların hiç ele alınmadığını ve bunun çözümü içinde farklı bir yaklaşım ile UC oluşturma yönergelerini, UC denetimlerini birleştiren entegre bir yaklaşım ortaya konmuştur [18]. Etkili bir şekilde gereksinim spesifikasyonunu yazmanın bir araç ile yapılması durumundaki yaklaşım ele alınmış ve açıklamıştır [19]. İyi gereksinim yazmak için on sekiz maddelik teknik rapor oluşturulmuştur [20]. İyi gereksinimlerin özellikleri ile gereksinimler nasıl yazılmalı başlıkları altında iyi gereksinim yazma teknikleri ele alınmıştır [21]. Web tabanlı bilgi sistemleri için üç aşamalı gereksinimleri meydana çıkarma yöntemi olarak başlangıç analizi, anahtar kullanıcılar ile görüşme ve gereksinimleri ortaya çıkarma ve son olarak normal kullanıcılar ile görüşme şeklinde sıralamıştır [22]. On altı yazılım geliştirme şirketindeki ankette yedi çevik gereksinim mühendisliği pratiklerinin faydaları ile beraber ortaya konmuştur [23]. ISO/IEC/IEEE 29148 standardında özellikle tekil gereksinimlerin ve bir grup gereksinimin karakteristikleri sıralanmıştır. Tekil gereksinim karakteristiklerini gereklilik, uygulanabilir, belirsiz olmayan, tutarlı, tam, tekil, mümkün, izlenebilir ve doğrulanabilir şeklinde yer almaktadır. Bunun yanında birleştiklerinde yazılımda bir fonksiyon veya modül olan gereksinimlerin karakteristiklerini ise tam, tutarlı, uygun ve sınırları çizilmiş olarak yer verilmiştir [24].

Çevik yazılım geliştirmede gereksinimlerin nasıl önceliklendirileceği ile ilgili küçük sayıda gereksinim olması durumunda kullanılan teknikler ve orta ve büyük sayıda gereksinim olması durumunda kullanılan teknikler şeklinde sınıflandırmıştır. Küçük sayıda gereksinim olması durumunda grup içi önceliklendirme, yüz dolarlık tahsis tekniği, çoklu oylama sistemi, ikili analiz, ağırlıklı kriter analizi, nokta oylama tekniği ve kalite işlevsel dağıtım yaklaşımı olarak ele alınmıştır. Orta ve büyük miktarda gereksinim olması durumunda ise MoSCoW (Must, Should, Could, Won't) tekniği, İkili öncelik listesi (ikili arama ağacı tekniği), Planlama oyunu ve wiegers'in matris yaklaşımıdır [25]. İkili öncelik listesi detaylı bir şekilde analiz edilerek uygulamalı olarak ortaya konmuştur [26].

Yazılımda kalite olgusunun büyük sistemlerin geliştirilmeye başlaması ile 1960'lı yıllarda gündeme alınmıştır. Yazılım kalite elementlerini, emniyet, güvenlik, güvenilirlik, esneklik, sağlamlık, anlaşılabilirlik, test edilebilirlik, uyarlanabilirlik, modülerlik, karmaşıklık, taşınabilirlik, kullanılabilirlik, yeniden kullanılabilirlik, verimlilik, öğrenilebilirlik olarak belirtilmiştir. Kalite kapsamındaki standardizasyonun ürün ile ilgili standardizasyon ve süreç ile ilgili standardizasyon şeklinde sınıflandırılmıştır [27].

4. YÖNTEM (METHOD)

Gereksinim kalite kriterleri olarak farklı doküman ve standartlar birbirine yakın bir şekilde bu kriterleri belirlemiştir. Aşağıda, ilerleyen zamanda incelenen projelerde gereksinim kalite kriterlerini belirlemek için kullanılan ve bu anlamda öne çıkan doküman ve standartlardaki tanımlamalar ele alınmıştır.

Gereksinimlerin belirlenmesi amacı ile farklı yöntemler kullanılabilmektedir. Uygun yöntemi belirlemek her zaman daha iyi ve kaliteli bir yol alınmasını sağlayacaktır. Bu konuda İş Analizi Bilgi Birikimi (Business Analysis Body of Knowledge (BABOK)), yazılım geliştirmenin ilk safhası olan iş analizi ile ilgili tüm yöntem ve metotları açıklamaktadır. Bu dokümanda yer alan çözüm gereksinimleri gereksinim belirleme ve sistem tasarımı safhasını oluşturmaktadır. Çözüm gereksinimleri çözümün geliştirilmesi ve uygulanması için yeterli detayda olmalıdır. Fonksiyonel gereksinimler ve fonksiyonel olmayan gereksinimler şeklinde iki alt kategoriye bölünebilir. Fonksiyonel gereksinimler, sistemin fonksiyonları şeklinde ele alınması gereken gereksinimlerdir. Fonksiyonel olmayan gereksinimler ise sistemin fonksiyonlarının uygun bir şekilde çalışması için gerekli olan güvenilirlik, hız, güvenlik gibi gereksinimlerdir. Ayrıca, dokümanda, gereksinimlerin özellikleri ve tasarım kalitesi başlığı altında dokuz özellik Tablo 1'de açıklanmıştır [2]. Bu dokümana yapılan çevik uzantı (Agile Extension to the BABOK), çevik yazılım geliştirme aşamasında deneyim haritası (story mapping) ile kullanıcı hikayelerinin gerçekten istenen ürün özelliklerini sağlayıp sağlamadığı analizi yapılabilmektedir [28].

Tablo 1. Business analysis body of knowledge (BABOK) gereksinim özellikleri ve tasarım kalitesi
(Business analysis body of knowledge (BABOK) requirement properties and design quality)

Kalite kararkteristiği	Açıklama
Atomik	Diğer gereksinimler veya tasarımlardan bağımsız olarak kendi başına anlaşılabilir ve bağımsız olmalıdır.
Tam	Uygun seviyede daha fazla seviyede çalışmaya rehberlik edecek kadar ayrıntı olmalıdır.
Tutarlı	Paydaşların ihtiyaçları ile uyumlu olmalı ve diğer ihtiyaçlar ile çelişmemelidir.
Öz	Gereksiz içerik olmamalıdır.
Uygulanabilir	Kararlaştırılan risk, takvim ve bütçe dahilinde makul ve mümkün seviyede uygulanabilir olmalıdır.
Açık	Gereksinim açık ve belirli bir şekilde ifade edilmelidir.
Test Edilebilir	Gereksinim veya tasarım test edilebilir olmalıdır.
Önceliklendirilmiş	Diğer tüm gereksinimlere karşı önem açısından sıralanmış, gruplandırılmış veya müzakere edilmiş olmalıdır.
Anlaşılabilir	Okuyucunun ortak terimleri ile ifade edilmiş olmalıdır.

Elektrik ve Elektronik Mühendisleri Enstitüsü (Institute of Electrical and Electronics Engineer – IEEE) tarafından yayımlanan yazılım gereksinim spesifikasyonu (Software Requirements Spesification- SRS) dokümanında kalite gereksinimi, sekiz ana başlık altında toparlanmıştır. Bunlar, Tablo 2'de sunulmuştur [3].

Tablo 2. IEEE yazılım gereksinim spesifikasyonu kalite gereksinimleri
(IEEE software requirement specification quality requirements)

Kalite gereksinimi	Açıklama
Doğru	SRS kapsamında belirtilen her bir gereksinimi yazılım karşılamalıdır.
Açık	SRS kapsamında belirtilen her gereksinimin sadece bir yorumu vardır.
Tam	SRS kapsamında ister fonksiyonel ister performans, isterse farklı türden oluşan tüm gereksinimler ele alınmalıdır.
Tutarlı	SRS kapsamındaki tüm gereksinimler birbiri ile çelişmeden tutarlı olmalıdır.
Önem ve / veya Olgunluk Derecesine Göre Sıralanmış	SRS kapsamındaki gereksinimler, önem derecesine göre veya olgunluklarına göre sıralanmalıdır.
Doğrulanabilir	SRS kapsamındaki tüm gereksinimler doğrulanabilir olmalıdır.
Değiştirilebilir	SRS kapsamındaki gereksinimlerdeki değişiklikler uyumlu bir şekilde kolayca yapılabilmelidir.
İzlenebilir	SRS kapsamındaki gereksinimler numaralandırılmış, ileriye doğru ve geriye doğru izlenebilir olmalıdır.

NASA, Sistem Mühendisliği El Kitabının ekinde yukarıdaki tabloda yer alan ilgili standartlardakine benzer şekilde, iyi bir gereksinim yazma kılavuzu kapsamında, ilgili kalite kriterlerini sıralamıştır. Gereksinimlerin kalite kriterlerini netlik, bütünlük, uyum, tutarlılık, izlenebilirlik, doğruluk, işlevsellik, performans, ara yüzler, sürdürülebilirlik, güvenilirlik, doğrulanabilirlik / test edilebilirlik, veri kullanımı şeklinde sıralanmıştır [29].

Gereksinim kalite kriterlerinin yanında yazılımda olması gereken kalite karaktaristikleride yine farklı doküman ve standartlarda birbirine yakın bir şekilde sıralanmıştır. Bunlardan ISO/IEC 9126 Yazılım Mühendisliği – Ürün Kalitesi adlı standart yazılım kalitesinin karakteristiklerini altı ana karakteristik ve altındaki yirmi yedi alt karakteristikte toparlamıştır. Bu altı ana karakteristik kullanılabilirlik, taşınabilirlik, verimlilik, güvenilirlik, fonksiyonellik ve bakım yapılabilirlik olarak tanımlanmıştır [30]. 1991 yılında ilk versiyonu yayımlanan ISO 9126 standardının yerini 2011 yılında ISO/IEC 25010 Sistem ve Yazılım Mühendisliği – Sistem ve Yazılım Kalite Gereksinimleri ve Değerlendirmesi (SQuaRE) – Sistem ve Yazılım Kalite Modelleri standardı almıştır. ISO 25010 standardı eski altı ana karakteristiğe iki yeni karakteristik daha eklemiştir. Bunlar güvenlik ve uyumluluk olmuştur. Mevcut karakteristiklerin iki tanesinin adı da değiştirilmiştir ve bazı alt karakteristikler eklenmiştir. Verimlilik adı performans verimliliği, fonksiyonellik adı fonksiyonel uygunluk haline

dönüştürülmüştür [31]. Tablo 3'de bu karakteristikler açıklanmıştır.

Tablo 3. ISO/IEC 9126 ve 25010 standardı yazılım kalite karakteristikleri
(Software quality characteristics of ISO / IEC 9126 and 25010 standard)

Kalite karakteristiği	Açıklama
Fonksiyonellik (Fonksiyonel uygunluk)	Belirlenmiş ihtiyaçları yazılımın fonksiyonları ile karşılayabilmesi. Doğruluk, güvenlik gibi alt başlıkları vardır.
Güvenilirlik	Belirli bir zamanda ve durumda istenilen performansı gösterebilmesi. Olgunluk, hata toleransı, düzeltilebilirlik gibi alt başlıkları vardır.
Kullanılabilirlik	Belirli kullanıcılar ile yazılımın kullanılabilir olması. Anlaşılabilirlik, öğrenilebilirlik, işletilebilirlik gibi alt başlıkları vardır.
Verimlilik (Performans verimliliği)	Belirli koşullar altında yazılımın gösterdiği performans ile tükettiği kaynaklar arasındaki ilişki. Zaman içindeki davranış ve kaynak kullanım gibi alt başlıkları vardır.
Bakım Yapılabilirlik	Yazılımda, gerektiğinde değişiklik yapılabilir olması. Analiz edilebilir, değiştirilebilir, test edilebilir, durağan olma şeklinde alt başlıkları vardır.
Taşınabilirlik	Yazılımın bir ortamdan diğer ortama taşınabilir olması. Kurulabilme, uyum sağlayabilme, yerine kurulma gibi alt başlıkları vardır.
Güvenlik	Verilerin yetki kapsamında görülebilmesi, kimin ne yaptığının izlenebilmesi alt başlıkları vardır.
Uyumluluk	Yazılımın karşılıklı uyumluluk içinde olmasıdır.

Kalite kriterleri olarak, yazılım geliştiren firmada yer alan yazılım geliştirme yaşam döngüsü dokümanındaki kalite hususları ve farklı safhalardaki kalite kriterleri incelenmiştir. Firmanın yazılım geliştirme yaşam döngüsünde ana hatları ile her bir safhadaki kalite hususlarının neler olacağı sıralanmıştır. Bunlar üç ana bölüme ayrılmıştır. Gereksinim ve sistem tasarımı ile ilgili olanlar, yazılım kodlaması ile ilgili olanlar ve geliştirilen sistemin test edilebilmesi için yazılan test senaryolarının kalite gereksinimleridir. Gereksinim ve sistem tasarımı ile ilgili olanlar; doğruluk, tamlık, tutarlılık ve uyumluluktur. Bu kriterlerin yanısıra ayrıca yukarıdaki Tablo 1, 2 ve 3'de yer alan kriterler incelenmiş ve son olarak çevik yazılım geliştirme aşamasındaki kullanıcı deneyimi kalite gereksinimleri de değerlendirilerek yeni kalite kriterleri belirlenmiştir. Yazılımı geliştiren firma, yazılım geliştirme yaşam döngüsü olarak hem çevik hem de şelale yaşam döngüsünü kullanmaktadır. Bu çalışmada, özellikle gereksinim aşamasındaki kalite hususları dikkate alınacağı için hangi yazılım geliştirme yaşam döngüsünün uygulandığından ziyade gereksinim aşamasına odaklanılmıştır. Gereksinimler yazılım geliştirme sürecinde fonksiyon listesi olarak tanımlanmış ve her bir fonksiyonda olması gereken kalite özellikleri Tablo 4'de sunulmuştur. Tablo 4'de yer alan kriterler belirlenirken firmanın yazılım geliştirme ekibindeki hem iş analisti hem yazılımcı hem de test aşamasında çalışan test sorumlularından ilgili personel arasından, kullanıcı tarafından anahtar kullanıcılardan ve son olarak projelerin yöneticileri arasından ilgili kişilerden oluşan bir grup oluşturulmuştur. Gruba bu konuda ne yapılmak istendiği bilgisi verilmiş ve ardından değerlendirme açısından hangi kalite kriterlerinin gerektiğinin açıklaması ile birlikte listelenmesi istenmiştir. Ardından her bir grubun listesi üzerinden grup ile beraber üzerinden gidilmiş ve ardından Tablo 4'de yer alan nihai listeye ulaşılmıştır. Sonuç olarak listedeki her bir kriter farklı bakış açılarını yansıtacak şekilde ele alınmıştır.

Tablo 4. Çalışma kapsamında kullanılan gereksinim kalite kriterleri
(Requirement quality criteria used in the scope of the study)

Kalite kriteri	Açıklama
Doğruluk	Her bir gereksinimin doğru bir şekilde ifade edilmesi gerekmektedir.
Açıklık	Her bir gereksinimin açık bir şekilde ifade edilmesi gerekmektedir.
Tamlık	Her bir gereksinimin tam ve eksiksiz bir şekilde ifade edilmesi gerekmektedir.
Tutarlılık	Her bir gereksinimin tutarlı olması gerekmektedir.
Uyumluluk	Her bir gereksinimin diğerleri ile uyumlu olması gerekmektedir.
Özlük	Her bir gereksinimin açıklaması öz bir şekilde ele alınması gerekmektedir.
Uygulanabilirlik	Her bir gereksinim uygulanabilir olması gerekmektedir.

The Standish Group'un 2015 yılında 10.000'in üzerinde küçük, orta ve büyük çaplı yazılım projeleri ile oluşturdukları Chaos Raporuna göre çevik yazılım geliştirme modelleri ile ilerleyen yazılımlardaki başarısızlık oranı yüzde 9 seviyelerinde iken bu oran şelale modelinde yüzde 29 seviyelerindedir. Aynı raporda kısmen başarılı oranı çevik modellerde yüzde 52 şelale modelinde ise yüzde 60 seviyesindedir. Son olarak, bu raporda projelerin başarılı olma durumları ise çevik modellerde yüzde 39, şelale modellerinde ise yüzde 11 seviyelerindedir [32]. Aynı grubun 2018 yılında yayınladıkları raporda bu oranlar çevik yazılım projelerinde başarı yüzde 42, başarısızlık oranı ise yüzde 8 dir. Geriye kalan yüzde 50 oran ise ortadadır. Bu oranlara şelale modeli tarafından bakacak olursak, yüzde 26 başarılı, yüzde 21 başarısız ve yüzde 53 ortadadır. Bu raporda, bu başarısızlıkların nedenlerine baktıklarında, belirli ve tanımlı olmayan gereksinimler, değişen gereksinimler, gerçekçi olmayan beklentiler, yönetim desteğinin ve proje kaynağı yetersizlikleri şeklinde sıralanmıştır [33]. Görüleceği üzere sebeplerin çoğu gereksinimler ile ilgilidir. Dolayısı ile hangi yazılım geliştirme yaşam döngüsünü kullanırsak kullanalım eğer gereksinimleri kaliteli bir şekilde toplayamazsak projedeki başarı şansımız azalacaktır.

Çevik yazılım geliştirme aşamalarında genellikle dinamik tekniklerin kullanılmakta, şelale modelinde ise hem dinamik hem de statik yöntemler kullanılmaktadır. Ayrıca, çevik yazılım geliştirme aşamasında kalite hususları genellikle işi yaparken ki aşamaya doğru çekilmiştir ve işler daha küçük parçalara bölündüğü için kalite ile ilgili hususlar daha kontrol edilebilir durumdadır. Şelale modelindeki gereksinim kalitesi, genelde kontrol listeleri ile kontrol edilir ve ona göre raporlanırken, çevik modelde, ürün takımı içinde müşteride işin içinde olmakta ve dolayısı ile gereksinimlerin anlaşılırlığı daha açık hale getirilebilmektedir. Şelale modelinde de müşteri işin içinde olmakta ancak çevik modelde olduğu kadar olamamaktadır. Hatta gereksinim değişiklikleri, çevik modellerdeki küçük parçalardan ve döngüsel olarak ilerleme ve ürün takımlarının bir arada takım ruhu ile çalışmasından dolayı daha iyi kontrol edilebilmektedir. Ancak, çevik modellerdeki en büyük sorun ise gereksinimler bu kadar küçük parçalara bölünmesi durumunda yönetiminin de iyi bir şekilde yapılması gerektiğidir [34]. Çevik gereksinimler için kalite unsuru olarak, tamlık, bütünlük ve tutarlılık ile doğruluk olarak yer almaktadır. Bunların hem özellik hem de kullanıcı hikayesi kapsamında ele alınan kalite gereksinimleri olması gerekmektedir [35].

5. UYGULAMA (APPLICATION)

Bu çalışmada, gereksinim belirleme ve sistem tasarımı aşamasında ortaya konulan kalite gereksinimlerine ve hazırlanan yazılım gereksinim spesifikasyonunun (Software Requirements Spesification – SRS) kaliteli bir şekilde oluşturulup oluşturulmadığı, bir servis işletmesi için yapılan üç projenin tüm gereksinimleri incelenmiş ve genel olarak değerlendirilmiştir. Projeler, son bir yıl içinde tamamlanmış ve yakın zamanlarda kullanım aşamasına alınmıştır. Projeler, işletmenin bir dönüşüm içinde bulunduğu 2018 ve 2019 yılları içinde yapılmıştır. Projelerin amacı işletme içindeki dijital dönüşümün gerçekleşmesi için hayata geçirilmişlerdir. Proje A, stok yönetimi ile ilgili bilgilerin tutulduğu ve bir anlamda işletmenin elinde yer alan tüm malzemelerin stoklarının takip edildiği bir projedir. Proje A ilk aşamada 14 ay şeklinde planlanmış ancak 18 ayda gerçekleştirilmiştir. Proje B, işletmenin verdiği hizmetleri, projeleri ve onunla ilgili bilgileri tutan bir sistemin geliştirilmesi projesidir. Proje B ilk aşamada 12 ay şeklinde planlanmış ancak 13 ayda gerçekleştirilmiştir. Proje C ise işletmenin müşterilerinin sorunlarını bildirdikleri ve bu sorunların takip ve çözümlerinin bilgilerinin yer aldığı bir sistem projesidir. Proje C, 10 ay şeklinde planlanmış ancak 12 ayda gerçekleştirilmiştir. Projelerdeki gereksinimlerin, Tablo 4'deki belirlenmiş olan kalite kriterlerine uyum sağlayıp sağlamadıkları irdelenmiştir. Projelerdeki değerlendirilen gereksinimlerin tamamı fonksiyonel gereksinimlerdir. Projeler hem şelale modelinde hem de çevik yöntemlerin karması olan geliştirici firmanın kendisine göre uyarladığı bir yöntem ile geliştirilmiştir.

Aşağıdaki Tablo 5'de projelerin büyüklükleri ve her bir projenin test ve son bir yıl içindeki işletim aşamasından gelen hata sayıları yer almaktadır. Tablo 5'de yer alan proje A'daki gereksinimler ana hatları ile stok giriş, stok çıkış, stok yerleri, depo yerleri, depo içi tanımlar, stok miktarları, malzeme özellikleri gibi gereksinimlerin detaylarından oluşmaktadır. Proje B'deki gereksinimler ise ana hatları ile verilen hizmetler, hizmetler ile ilgili bilgiler, proje tanım bilgileri, proje mali bilgileri, proje müşteri bilgileri, proje kaynak bilgileri gibi gereksinimleri kapsamaktadır. Proje C'deki gereksinimler ise ana hatları ile müşteri sorun bildirimleri, müşteri sorunları, sorunların çözümleri, proje ile ilişkisi, satılan ürün ile ilişkisi gibi gereksinimleri kapsamaktadır.

Tablo 5. Projelerin geliştirme ile işletim safhaları hata sayıları
(The number of errors in the development and operation phases of the projects)

Projeler	Toplam gereksinim sayısı	Test aşaması toplam hata sayısı	İşletim aşaması toplam hata sayısı
Proje A	57	72	12
Proje B	41	83	27
Proje C	15	21	3

Tablo 5 incelendiğinde en çok hata oranı Proje B'de, ardından Proje C'de ve en az hata oranı ise Proje A'da görülmüştür. Çalışmanın devamında gereksinimler yukarıdaki kalite kriterlerine göre incelendiğinde her üç projede de Tablo 6'daki sonuçlar elde edilmiştir.

Tablo 6. Gereksinimlerin kalite kriterlerine uyumu (sağlamıyor)
(Compliance of requirements with quality criteria) (not met))

	Gereksinimler'den kaç tanesi kriteri sağlamıyor		
Gereksinim kalite kriterleri	*Proje - A*	*Proje - B*	*Proje - C*
Doğruluk	15	26	7
Açıklık	34	29	8
Tamlık	23	22	9
Tutarlılık	13	16	5
Uyumluluk	5	13	6
Özlük	28	20	7
Uygulanabilirlik	7	13	4

Tablo 6'daki sonuçlar her bir proje bazında incelendiğinde Tablo 5'deki hataların nedeni anlaşılabilmektedir. Görüleceği üzere gereksinim sayısına göre Proje – B'deki kalite kriterlerini karşılama oranı bayağı düşüktür. Projelerin büyükten küçüğe doğru kalite kriterlerine uyumsuzluk sıralaması ise Tablo 7'de verilmiştir.

Tablo 7. Proje gereksinimlerinin kalite kriterlerine uyumsuzluk sıralaması
(Ranking of non-compliance of project requirements with quality criteria)

Proje - A uyumsuzluk sıralaması	Proje - B uyumsuzluk sıralaması	Proje - C uyumsuzluk sıralaması
Açıklık	Açıklık	Tamlık
Özlük	Doğruluk	Açıklık
Tamlık	Tamlık	Özlük
Doğruluk	Özlük	Doğruluk
Tutarlılık	Tutarlılık	Uyumluluk
Uygulanabilirlik	Uygulanabilirlik	Tutarlılık
Uyumluluk	Uyumluluk	Uygulanabilirlik

Her üç projede de açıklık kriterinin ihlal edildiği görülmektedir. Ardından sırası ile ihlal edilen kriterler tamlık, özlük, doğruluk, tutarlılık, uygulanabilirlik ve uyumluluk şeklinde sıralanabilir. Projelerdeki gereksinim kalite kriterlerini sağlaması açısından bakıldığında ise Tablo 8'de yer alan değerlere ulaşılmıştır.

Tablo 8. Gereksinimlerin kalite kriterlerini uyumu (sağlıyor)
(Compliance of requirements with quality criteria) (met))

	Gereksinimler'den kaç tanesi kriteri sağlıyor		
Gereksinim kalite kriterleri	*Proje - A*	*Proje - B*	*Proje - C*
Doğruluk	42	15	8
Açıklık	23	12	7
Tamlık	34	19	6
Tutarlılık	44	25	10
Uyumluluk	52	28	9
Özlük	29	21	8
Uygulanabilirlik	50	28	11

Projelerin büyükten küçüğe doğru kalite kriterlerine uyumluluk sıralaması ise Tablo 9'da verilmiştir.

Tablo 9'dan görüleceği üzere Tablo 7'nin tam tersi bir sıralama yer almaktadır. Bu sıralamaya göre Proje – A da yer alan gereksinimler ve Proje -B'de yer alan gereksinimler uyumluluk, uygulanabilirlik ve tutarlılık açısından aynı sırlamayı almıştır. Proje-C'de ise yine ilk üçte aynı kriterler yer almış sadece Proje-C'de uygulanabilirlik ilk sıraya yerleşmiştir.

Bu çalışmanın sonucunda gereksinim kalite kriterlerine uyumsuzluk olarak bakıldığında açıklık, tamlık ve doğruluk kriterleri ihlal edilmiş ancak uyumluluk, uygulanabilirlik ve tutarlılık kriterlerinde ise daha yüksek bir uyum sağlandığı görülmüştür.

Tablo 9. Proje gereksinimlerinin kalite kriterlerine uyumluluk sıralaması
(Ranking of compliance of project requirements quality criteria)

Proje - A uyumluluk sıralaması	Proje - B uyumluluk sıralaması	Proje - C uyumluluk sıralaması
Uyumluluk	Uyumluluk	Uygulanabilirlik
Uygulanabilirlik	Uygulanabilirlik	Tutarlılık
Tutarlılık	Tutarlılık	Uyumluluk
Doğruluk	Özlük	Özlük
Tamlık	Tamlık	Doğruluk
Özlük	Doğruluk	Açıklık
Açıklık	Açıklık	Tamlık

6. SONUÇ VE ÖNERİLER (CONCLUSION AND RECOMMENDATIONS)

Bilgi teknolojilerinin ve sistemlerinin kullanıldığı en basit işlemden, uzaya uydu fırlatma gibi en karmaşık işleme kadar yazılım kullanılmaktadır. Bazı işlemlerde yazılımdaki hata ihmal edilebilirken, bazı durumlarda bunu ihmal etmek imkansızdır. Hatta insanoğlunun canı ile bu hataları ödemek zorunda kalınabilir. Dolayısı ile yazılım geliştirme yaşam döngüsünde her bir safhanın büyük bir önemi vardır. Ancak, bunlardan gereksinim tanımlama ve sistem tasarımı safhası belki de en büyük ve kritik safhalardan birisidir. Çünkü bu aşamada yapılan bir hata veya eksiklik ilerleyen zamanda daha pahalıya mal olacağı aşikardır. Dolayısıyla, iş analistinin hazırlamış olduğu gereksinimler, kalite kriterlerine uyumlu bir şekilde hazırlanması gerekmektedir.

Bu çalışmada, kalite kriterleri ile ilgili olarak standartlar, yapılan çalışmalar ve ayrıca üç projedeki gereksinimler, belirlenmiş olan kalite kriterlerine göre incelenmiştir. Kalite kriterleri doğruluk, açıklık, tamlık, tutarlılık, uyumluluk, özlük, uygulanabilirlik olarak farklı standart ve çalışmalar incelendikten sonra bir ekip tarafından belirlenmiştir. Sonuç olarak gereksinimlerin hazırlanması esnasında üç projede de en büyük kalite kriterinde yaşanan sorunun açıklık olduğu görülmüştür. Kalite kriterlerinden uyumluluk açısından da daha az sorun ile karşılaşılmıştır.

Açıklık, iş analistinin gereksinimi açık, anlaşılabilir bir şekilde ifade etmesidir. Bunu yapamaması durumunda sistem tasarımı, yazılım geliştirme ve test safhalarında yanlış bir şekilde ilerlemiş olacaktır. Dolayısı ile iş analistlerinin, gereksinimleri nasıl yazmaları konusunda bazı kurallara uymaları gerekmektedir. Bunun için kendilerini geliştirmeleri ve bu konuda ilerleme kat ederek bu tür hatalara düşmemeleri gerekmektedir. Gereksinimler yazılırken belirsiz ve açık olmayan kelime ve cümlelerden kaçınmak gerekmektedir. Bunun yanında, yazılan gereksinimi hedef kitlemizdeki paydaşların okuyup, tek bir anlam çıkarıp anlayabilecekleri bir dil ile oluşturmak gerekmektedir. Farklı anlam çıkarılacak ifadelerden kaçınmak, projenin ve ilgili fonksiyonun çerçevesi içinde kalmak en büyük netlik olacaktır. Ayrıca, gereksinimleri

ifade ederken ulusal, uluslararası veya var ise şirket içindeki standarda uygun bir şekilde yazılması da önem arz etmektedir. Bu standartlarda yer alan kurallara uyulması, gereksinimlerin anlaşılabilir olmasını kolaylaştıracaktır. Eğer anlaşılmayan hususlar var ise öncelikle ne ifade edildiğini açıklamak veya tanımlamak da faydalı olacaktır. Sonuç olarak, iş analistinin en büyük rolü iş dünyası ile bilişim dünyasının arasında köprü kurmaktır. Bu görev, yanlış anlatılan veya ifade edilen gereksinimler ile olursa bu köprü yıkılmaya veya iki grup arasındaki iletişim kopmaya ve yanlış sonuçlar üretilmeye mahkûm olunması kaçınılmazdır. Dolayısı ile bu köprünün temellerini teşkil eden gereksinimlerin iyi bir şekilde tanımlanması gerekmektedir.

Gereksinimlerin tam ve doğru bir şekilde belirlenmesi çok büyük önem arz etmektedir. Dolayısı ile yazılım geliştirme projelerinin tamamında, bu yazılım ister gömülü yazılım olsun ister arayüz ile etkileşim içinde olan yazılım olsun, gereksinim belirleme ve bu gereksinimlerin kalitesine dikkat edilmesi gerekmektedir. Ayrıca kalite gereksinimlerininde net bir şekilde tanımlanması gerekmektedir. Bu tür projeleri sadece kod geliştirme safhasından ibaret olarak görmek en büyük yanılgıdır. Ayrıca gereksinim safhasında harcanacak zamanı kayıp zaman olarak görmek yerine, bu zamanın ileride geri kazanılacağını da unutmamak gerekmektedir.

Bu çalışmaya ilave olarak, ilerleyen zamanda daha farklı tipteki ve sektördeki projelerde de benzer çalışılmanın yapılması ve en sık yapılan gereksinim belirleme hatalarının yayımlanarak, bunlardan ders çıkarılması gerekmektedir.

KAYNAKLAR (REFERENCES)

[1] Internet: Yazılım, https://tr.wikipedia.org/wiki/Yaz%C4%B1l%C4%B1m, 25.07.2021.

[2] Business Analysis Body of Knowledge (BABoK), A Guide to Business Analysis Body of Knowledge, International Institute of Business, 2015.

[3] IEEE Recommended Practice for Software Requirements Specifications, IEEE Computer Society Sponsored by the Software Engineering Standards Committee, 1998.

[4] Internet: B. Aktaş, Kalite Yaklaşımları ve Yazılım Kalitesi, https://medium.com/@aktasburak/kaliteyakla%C5%9F%C4%B1mlar%C4%B1-ve-yaz%C4%B1l%C4%B1m-kalitesi-23d395754bc1, 03.04.2020.

[5] Internet: M. Kirikova, J. A.Bubenko, Enterprise Modelling: Improving the Quality of Requirements Specifications. IRIS 17, https://pdfs.semanticscholar.org/9657/5d5fcb0bba481eb399758040274186e555ac.pdf, 03.04.2020.

[6] J. J. Carr, "Requirements Engineering and Management: The Key to Designing Quality Complex Systems", *The TQM Magazine*, 12(6), 400-407, 2000.

[7] M. Azuma, "Applying ISO/IEC 9126-1 Quality Model to Quality Requirements Engineering on Critical Software**", 12th IEEE International Requirements Engineering Conference (RE '04)**, 3-10, Kyoto, Japan, 06 Eylül 2004.

[8] G. Sabaliauskaite, A. Loconsole, E. Engstrom, M. Unterkalmsteiner, B. Regnell, P. Runeson, R. Feldt, "Challenges in Aligning Requirements Engineering and Verification in a Large-Scale Industrial Context." **Requirements Engineering: Foundation for Software Quality, 16th InternationalWorking Conference, REFSQ 2010**, 128-142, Essen, Germany, 2010.

[9] T. Arao, E. Goto, T. Nagata, "Business Process Oriented Requirements Engineering Process" **Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering (RE'05),** IEEE Computer Society, August 2005.

[10] A. David, X. Franch, "How Do Software Architects Consider Non-Functional Requirements: A Survey", **Requirements Engineering: Foundation for Software Quality, 16th International Working Conference, REFSQ 2010**, 276-278, Essen, Germany, 2010.

[11] A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, M. Theofanos, "Identifying and Measuring Quality in a Software Requirements Specification", **Proceedings First International Software Metrics Symposium**, 141-152, 1993.

[12] A. Nordin, N. H. Zaidi, N. A. Mazlan, "Measuring Software Requirements Specification Quality", *Journal of Telecommunication, Electronic and Computer Engineering*, 9(3-5), 123-128, 2017.

[13] R. S. Carson, **Implementing Structured Requirements to Improve Requirements Quality**, The Boeing Company, 2015.

[14] A Chakraborty, M..K. Baowaly, A. Arefin, A. N. Bahar, "The Role of Requirement Engineering in Software Development Life Cycle", *Journal of Emerging Trends in Computing and Information Sciences*, 3(5), 723-729, 2012.

[15] F. Fabbrini, M. Fusani, S. Gnesi, G. Lami, "Quality Evaluation of Software Requirement Specifications", **Proceedings of the Software and Internet Quality Week 2000 Conference**, 1-18, 2000.

[16] Internet: A. Herrmann, B. Paech, Quality Misuse Semantic Scholar, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.6249&rep=rep1&type=pdf, 06.04.2020.

[17] K. Esaki, "System Quality Requirement and Evaluation, Importance of Application of the ISO/IEC25000 series", *Global Perspectives on Engineering Management*, 2(2), 52-59, 2013.

[18] C. Denger, B. Paech, "An Integrated Quality Assurance Approach for Use Case Based Requirements", **Lecture Notes in Informatics**, 59-74, 2004.

[19] W. M. Wilson, "Writing Effective Qequireents Specifications" *CrossTalk: The Journal of Defense Software Engineering*, 16-19, 1999.

[20] Internet: N. Boulila, Guidelines for Good Requirements Writing with Examples, Researchgate. https://www.researchgate.net/publication/284173600, 10.04.2020.

[21] Internet: K. Şen, İyi Gereksinim Yazma Teknikleri, http://www.emo.org.tr/ekler/41bea7a76881d2c_ek.pdf, 06.04.2020.

[22] H. L. Yang, J. H. Tang, "A Three-Stage Model of Requirements Elicitation for Web-Based Information Systems" *Industrial Management & Data Systems*, 103(6), 398-409, 2003.

[23] L. Cao, B. Ramesh, "Agile Requirements Engineering Practices: An Empirical Study", *IEEE Software*, 60-67, 2008.

[24] System and Software Engineering – Life Cycle processes – Requirements engineering, ISO/IEC/IEEE 29148, 2011.

[25] Z. Racheva, M. Daneva, L. Buglione, "Supporting the Dynamic Reprioritization of Requirements in Agile Development of Software Products", **Proceedings of the Second International Workshop on Software Product Management**, 49-58, 2008.

[26] T. Bebensee, I. Van De Weerd, S. Brinkkemper, "Binary Priority List for Prioritizing Software Requirements", **Requirements Engineering: Foundation for Software Quality, 16th InternationalWorking Conference, REFSQ 2010**, 67-78, Essen, Germany, 2010.

[27] I. Sommerville, **Software Engineering**, Tenth Edition, Global Edition. Pearson, 2016.

[28] Agile Extension to the Business Analysis Body of Knowledge (BABOK), IIBA, Agile Alliance, 2017.

[29] Systems Engineering Handbook, Appendix C: How to Write a Good Requirement, NASA, 2020.

[30] Software Engineering - Product quality (ISO/IEC 9126), 2001.

[31] Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models (ISO/IEC 25010: 2011), 2011.

[32] Chaos Raporu, The Standish Group, 2015.

[33] Chaos Raporu, The Standish Group, 2018.

[34] M. Huo, J. Verner, L. Zhu, M. A. Babar, "Software Quality and Agile Methods", **Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)**, 2004.

[35] P. Heck, A. Zaidman, **A Quality Fraework for Agile Requireents: A Practitioner's Perspective**, 2014.