Güvenli Yazılım Geliştirme Süreç Modellerinin Karşılaştırılması Uygulamaları

Comparison of Secured Software Development Process Models Applications

Şeyda Ocak, Medine Yıldıztekin

Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi
Sakarya Üniversitesi

seydaocak@hotmail.com, medineyildiztekin@hotmail.com

Özet

Gelişen teknoloji ile birlikte günümüz şartlarında yazılım güvenliği ve güvenli yazılım süreç seçim kriterleri giderek önemli hale gelmeye başlamıştır. Bu çalışmada da bununla ilgili olarak yazılım süreç güvenliği modelleri ve içlerinden altı tanesi detaylı olarak anlatılmıştır. Bunlar: Çağlayan (Şelale) Modeli, Artırımsal Model, Yeniden Kullanılabilir Yazılım Modeli, Prototip (Hızlı Uygulama Geliştirme) Modeli, Helezonik (Sarmal) Model ve Evrimsel Model'dir.

Uygulama geliştirme departmanları tarafından güvenlikten noksan olarak yapılan projeleri dış kaynaklı saldırılardan korumak amaçlı yazılım güvenliğinin öneminden bahsedilmiştir. Bunun için de bir yazılımın hangi süreç modelini kullandığını bulan bir uygulama geliştirilmiştir. Altı yazılım süreç seçim kriteri çeşitli faktörler göz önüne alınarak incelenmiştir. Yazılım süreç seçim kriterleri üzerinde çalışıldıktan sonra da, bu süreçleri kullanan yazılım uygulamaları anlatılmıştır.

Abstract

In today's terms with developing tecnology, software security and secured software process selection criteria are becoming increasingly important. In this study, secured software process models and six of them described in detail. Six process models: Waterfall Model, Incremental Model, Reusable Software Model, Rapid Prototyping Model, Spiral Model and Evolutionary Model.

Security is an important property of any software for many applications are outsourced where the application deveopment lacks strong integration of software security.

For this an application which finds the process model that is used by a software, is developed. These six software process selection criterias are considered by various factors. After working on these software process selection criterias, the applications that use these criterias are told.

1. Giriş

Geleneksel yöntemler artık günümüz yazılım uygulamalarında etkili olamamaktadır. Bu konuda da şirketlerin en başarısız oldukları konuların başında yazılım güvenliği gelmektedir. Proje bolluğu, gelişmiş yazılım güvenliği, ürün çeşitliliği ve teknolojinin gelişmiş olmasına rağmen birçok uygulama günümüzde yazılım güvenliğinden yoksun şekilde geliştirilmektedir. Bu durum da şirketlerin dışarıya açılış kapılarında (ekstranet), intranet ve internet güvenliği konusunu daha da önemli hale getirmektedir.
yararlanılması yoluna başvurulur. [1]

Yazılım güvenliği konusu projeleri kötü niyetli saldırılara karşı korumak amaçlı ortaya çıkmıştır. Şuan ki organizasyon çevreleri yazılım sistemini riske atacak saldırılara karşı kritik güvenlik gereksinimlerinden yoksun şekilde oluşturulmaktadır.

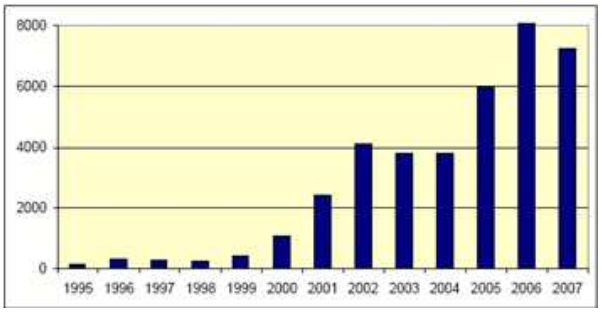
Yazılım güvenliği ölçümlerini dâhil etmek için, işletmeler var olan uygulama geliştirme yaşam metodolojilerini değiştirmek zorundadırlar. Geleneksel yöntemlerde, birçok yazılım geliştirme şirketleri yazılım yaşam döngüsünün en sonunda yazılım güvenliğine yer vermektedirler.

Yazılım geliştiren organizasyonların yürüttükleri projeleri düzgün bir şekilde, yani planlanan bütçe içerisinde kalarak, zamanında ve beklenen kalite seviyesini tutturarak, tamamlayabilmeleri için çeşitli yazılım araçlarına ihtiyaçları vardır. Yazılım projeleri yürütülürken karşılaşılan yönetimsel ve teknik zorluklar bu ihtiyacı daha da arttırmaktadır. Aslında, bir yazılımın ortaya çıkarılması için yapılması gereken birçok işin ilgili yazılım araçları olmadan tamamlanması pek mümkün görülmemektedir. Özellikle, yazılım geliştirme projelerinin yürütülmesine ve yönetilmesine yönelik araçların projede varlığı gittikçe daha önemli bir faktör haline gelmektedir. Bu yüzden, bu çalışmada "Yazılım Geliştirme Bilgi Sistemleri" olarak adlandırılabilecek bu tür araçlar açıklanmış ve bu araçların kullanımı bir bilgi sistemi modeli önerisi ile gösterilmiştir. Eğer ki servis organizasyonları bütünleşik güvenlik faktörleri ve güvenlik unsurlarının güvenli ölçüm modellerinden yoksun sistemlere idrak ettirilmesi konusunda başarısız olurlarsa, bu müşteri memnuniyeti ve güvenini zedeleyen bir durum olacaktır. [2]

Bu çalışmada, yaygın olarak kullanılan yazılım geliştirme süreç modelleri karşılaştırılarak, gelişen yazılım mühendisliği projelerinde uygun yazılım süreci için en iyi kriter modelinin seçilmesi ve tüm yazılım yaşam döngü sisteminin güvenli hale getirilmesi hedeflenmiştir.

2.Yazılım Geliştirme Sürecinin Tarihçesi

Bilgisayarların ilk ortaya çıkmasıyla birlikte yazılım geliştirme süreci de başlamıştır.
Bu süreç 1940'lı yıllara kadar gitmektedir.
İlk yıllarda geliştirilen yazılımlarda görülen en büyük eksiklik yazılım projelerinin zamanında tamamlanamaması ve istenilen kalitede (dokümantasyon, fonksiyonellik, harcanan fazla iş gücü) olmamasıdır.
Şekil 2.1'den görüleceği gibi son yıllarda yazılımlarda görülen açıklıklar önemli oranda artmaktadır. Bu açıklıkların artma nedenlerinin başında internetin yaygın olarak kullanılması ve bilgisayarın iş uygulamaları da dâhil olmak üzere kullanım oranının çok artması gelmektedir Diğer önemli sebep yazılımların güvenliği dikkate alınmadan sadece fonksiyonellik göz önüne alınarak geliştirilmesidir.



Şekil 2.1. Yıllara göre açıklıklar

2.1. Yazılım Geliştirme Süreci (YGS)

YGSM, herhangi bir yazılımın, üretim aşaması ve kullanım aşaması birlikte olmak üzere geçirdiği tüm aşamalar yazılım geliştirme süreç modeli olarak tanımlanır.[3].
biçiminde düşünülür. Yazılım yaşam döngüleri tek yönlü ve doğrusal olarak düşünülmemelidir. Döngü içerisinde herhangi bir aşamada geriye dönmek ve tekrar ilerlemek söz konusudur.
Temel Adımlar: Analiz, Çözümleme, Tasarım, Gerçekleştirim, Test, Bakım.

2.2. Yazılım Geliştirme Süreç Modelleri

Yazılım ile ilgilenen ve küçük-büyük projelerde yer alan-alacak olan herkesin bilmesi gereken modellerdir.
Yazılım geliştirmek için kullanacağımız bu modeller planlı çalışmamızı ve geliştireceğimiz yazılımları en iyi biçimde geliştirmemizi sağlayacaktır.

2.2.1. Helezonik (Sarmal) Model

Asıl amaç her bir artırımın güvenli ve hızlı yapılmasını sağlamaktır. Sarmal model kullanıldığında her bir artırımda yazılımın yeni bir sürümü ortaya çıkar. İlk artırımda ortaya çıkarılan kesim adeta bir prototipmiş gibi düşünülebilir. Prototipten tek farkı uygulama ortamında kullanılanın gerçek yazılım olmasıdır. Bir mühendislik uygulamasının gerektirdiği özelliklerde (kapasite, verimlilik, kalite gözetilerek) geliştirilmiştir. Sarmalın her bir döngüsü kendi içinde yapılması gereken yukarıdaki etkinliklere ayrılır. Etkinlikler projenin büyüklüğüne ve karmaşıklığına göre ayarlanabilir.

2.2.2. Prototip Model

Prototip geliştirme modelinde, gereksinimler hızlıca toplanarak işe başlanılır. Geliştiriciler ve kullanıcılar aynı masa etrafında buluşarak yazılımdan elde edilecek bütün çıktılara, bu çıktılar için gerekli girdilerin nasıl sağlanacağına, nasıl korunacağına, hangi işlemlere uğrayacağına karar verirler.
Daha sonra hızlıca yapılan bir tasarım ile yazılımın kullanıcıya yansıyacak yönünü aktaran bir prototip üretilir. [4] Prototip kullanıcının kullanımına ve değerlendirilmesine sunulur. Bu değerlendirmelere bakılarak prototip üzerinde gerekli değişiklikler yapılır. Prototipin yeni hali kullanıcı tarafından yeniden değerlendirilir. Böylece kullanıcının istediği yazılıma iyice yaklaşılmış bir prototip üzerinde yazılımın neler yapacağı konusunda kullanıcı ile anlaşmaya varılır.

2.2.3. Hızlı Prototip Modeli

Kullanıcının belli gereksinimlerinin sağlanması için çok kısa süreler içinde ortaya çıkartılan kapsamı daraltılmış alt sistemlere ilişkin yazılımlardır.
Gerçekleştirimin hızlı yapılabilmesi genelde bileşen tabanlı olmasını gerektirir. Hazır bileşenler gereksinimleri karşılayabiliyorsa 60-90 gün içinde çalışan bir program ortaya çıkartılabilir. [4]

2.2.4. Çağlayan (Şelale) Modeli

Şelale yönteminde yazılım geliştirme süreci analiz, tasarım, kodlama, test, sürüm ve bakım gibi safhalardan oluşur. Geleneksel yazılım metodlarında bu safhalar şelale modelinde olduğu gibi linear olarak işler. Her safha, baslangıç noktasında bir önceki safhanın ürettiklerini bulur. Kendi bünyesindeki değişikler doğrultusunda teslim aldıklarını bir sonraki safhanın kullanabileceği şekilde degiştirir. [5]

2.2.5. Artırımsal Model

Artırımsal model bir takvime bağlı olarak yazılımı kesim kesim geliştirip teslim etmeye dayanır. Her bir yeni kesim öncekinin üstüne bazı ek işlevlerin eklenmesini öngörür.
Artırımsal model yazılım geliştirmenin kısıtlı sayıda çalışanla işin yapılmasını sağlama gibi bir üstünlüğü vardır. Ayrıca çalışanlar da her artırım geçildiğinde uygulama alanına ilişkin daha çok deneyim kazanmış olurlar. [6]

2.2.6. Evrimsel Model

İlk tam ölçekli modeldir. Coğrafik olarak geniş alana yayılmış, çok birimli organizasyonlar için önerilmektedir (banka uygulamaları). Her aşamada üretilen ürünler, üretildikleri alan için tam işlevselliği içermektedirler. Pilot uygulama kullan, test et, güncelle diğer birimlere taşı. Modelin başarısı ilk evrimin başarısına bağımlıdır.
Yazılımlar bütün karmaşık sistemler gibi zaman içinde evrimleşir.

2.2.7. Yeniden Kullanılabilir Model

Organizasyon tarafından daha önce hazırlanmış veya dışarıdan temin edilmiş yazılımların (veya yazılım parçalarının) kullanılması ile geliştirme yapılması son yıllarda popüleritesi artan bir yaklaşımdır. [7] Organizasyonların olgunlukları arttıkça, bu tür uygulamalar yapabilmek için alt yapı kurulmuş olmaktadır.

3.Yazılım Geliştirme Süreç Modellerinin Karşılaştırılması

SPSM için en iyi seçim kriterleri analizi ölçekleme işlemi beş adımda anlatılmıştır:

Adım 1: Projenin özellikleri çok dikkatli bir biçimde analiz edildikten sonra projeyi en iyi tanımlayabilecek her kriter için 0 veya 1 lik ikili gösterge belirlenir. Bir (1) , özel yazılım projesi için sürecin o fonksiyon kriterinin uygun olduğunu gösterirken sıfır(0) ise o fonksiyon kriterinin uygun olmadığını gösterir.

Adım 2: Her satır için, Ci fonksiyon nokta değerlerinin toplam puanlarını hesaplar. Düşük-orta değerleri satırındaki ilk üç fonksiyon noktası değerleri ile yüksek-orta değerleri satırındaki son üç fonksiyon noktası değerlerini alır ve her üç fonksiyon noktası değerlerinin maksimum puanı ile toplar.

Adım 3: Düşük-orta değerleri ile yüksek-orta değerlerinin toplamları hesaplanır ve kriterlere bağlı istenilen model ortaya çıkarılır. İstenilen modelin türetimi tamamen müşterinin kendi projesine bağlı olarak düşünülür.

Adım 4: Bütün bu işlemlerden sonra süreç modelleri için bir sıralama yapılır. Bütün elde edilen düşük-orta puanlar ile istenilen düşük-orta puanlar ve elde edilen yüksek-orta puanlar ile istenilen yüksek-orta puanlar karşılaştırılır.

Adım 5: Final puanları sıralanır. Türetilen sonuçlar ölçekleme sıralarının bulunması için değerlendirilir. İstenilen yüksek-orta puanına yakın en iyi türetilen yüksek-orta puanlı süreç en iyisi seçilir. İstenilen yüksek-orta puanın altında kalan puanlar en az dikkate alınanlardır ve güvenli bir yazılım süreç modeli seçiminde istenilen düşük-orta puanlar pek yararlı değildir. [8]

Bir yazılım projesinin karakteristiğini anlayabilmek için, yazılımın geliştirildiği ortamın, etrafını saran çevresel faktörlerin ve yazılımı geliştiren organizasyonun dikkatli bir şekilde analiz edilmesi gerekmektedir.

	Süreç Modeli	Elde Edilen Düşük-Orta	Elde Edilen Yüksek-Orta	Toplam
1	Hızlı Prototip	36	65	101
2	Yeniden Kullanılabilir	33	73	106
3	Sarmal	22	96	118
4	Şelale	51	87	138
5	Evrimsel Prototip	42	98	140
6	Artırımsal	48	97	145

Şekil 3.1. Süreç modelleri değerlendrme sırası

İncelenen projenin kriter toplamının hangi modele yakın olduğunu görmek için yapılan matematiksel işlemleri ve oluşturulan algoritmanın adım adım açıklanması şu şekildedir:

1. Hızlı Prototip Model ile hemen altındaki Yeniden Kullanılabilir Model'in sonuçlarının ortalaması bulundu: (101 + 106) / 2 =103.5
103. 5 ondalıklı bir değer olduğu için şuna karar verildi: Eğer bulunan değer (sonuç) 0 ile 103 arasında ise; yani <= sonuç <= 103) ise bu model "Hızlı Prototip Model' e yakındır."

2. Yeniden Kullanılabilir Model ' in değeri, Hızlı Prototip Model' in sınırı 103 olarak belirlendiğinden 104 ile başlatıldı ve (106 + 118) / 2= 112 olduğundan 112 ile sınırlandırıldı. Alınan karar:
(104<= sonuç <= 112) ise "bu model Yeniden Kullanılabilir Model' e yakındır."

3. Helezonik (Sarmal) Model için:
(118 + 138) / 2= 128 olduğundan (113 <= sonuç <= 128) ise "bu model Helezonik (Sarmal) Model' e yakındır."

4. Şelale (Çağlayan) Modeli için;
(138 + 140) / 2= 139 olduğundan (129 <= sonuç <= 139) ise "bu model Şelale (Çağlayan) Model' e yakındır."
5. Evrimsel Prototip Model için:
(140 + 145) / 2= 142,5 olduğundan (140 <= sonuç <= 142) ise "bu model Evrimsel Prototip Model' e yakındır."

6. Artırımsal Model için:
(sonuç >= 143) ise "bu model Artırımsal Model' e yakındır." Denilebilir.

Toplamları birbirlerine yakın olan süreç modellerinde elde edilen Düşük-Orta ve elde edilen Yüksek-Orta değerlerine de bakılması gerekir.

	Kriter (Ci)	F1	F2	F3	F4	F5	F6
1	Uygulamanın Olgunluğu	Güçlü	Yeni	Benzer	Standart	İyi-Anlaşılır	Ana-Kapalı
2	Problem Karmaşıklığı	Önemsiz	Basit	Titiz	Zor	Karmaşık	Zorlu
3	Kısmi İşlevsellik Şartı	Arzu Edilmiyor	İsteğe Bağlı	Arzu Edilir	Kritik	Kaçınılmaz	Zorlu
4	Değişim Frekansı	Seyrek	Az	Orta	Hızlı	Hızlı	Gösterişli
5	Değişim Büyüklüğü	Önemsiz	Küçük	Küçük	Orta	Büyük	Aşırı
6	Tanımlı Güvenlik Sorunu	Önemsiz	Önemli	Daha Önemli	Önemli	Daha Önemli	En Önemli
7	Kullanıcı Deneyimi	Acemi	Bilgili	Deneyimli	İyi Deneyimli	Uzman	Deneyimli-Uzman
8	Kullanıcı İfade Yeteneği	Pervasız	Kararsız	Sessiz	İletişimli	Etkileyici	Tanımlayıcı
9	Uygulama Geliştirici Deneyim Alanı	Acemi	Bilgili	Deneyimli	İyi Deneyimli	Uzman	Deneyimli-Uzman
10	Yazılım Mühendisliği Deneyim Gelişimi	Acemi	Bilgili	Deneyimli	İyi Deneyimli	Uzman	Deneyimli-Uzman
11	Sistem Güvenliği Farkındalığı	Önemsiz	Kıt	Sınırlı	Yeterli	Yeterli	Çok
12	Kullanıcı Katılımı & Güvenlik	Önemsiz	Kıt	Sınırlı	Yeterli	Yeterli	Çok
13	Finansman	Düzensiz	Düşük-Sabit	Düşük-Yüksek	Yüksek-Düşük	Yüksek-Kararlı	Yüksek-Artış
14	Para Kullanılabilirliği	Önemsiz	Kıt	Sınırlı	Yeterli	Yeterli	Çok
15	Personel Durumu	Düzensiz	Düşük-Kararlı	Düşük-Yüksek	Yüksek-Düşük	Yüksek-Kararlı	Yüksek-Artış
				……..			
33	Güven Yönetimi	Önemsiz	Çok Düşük	Düşük	Yüksek	Çok Yüksek	En Yüksek
34	Risk Yönetimi	Önemsiz	Çok Düşük	Düşük	Yüksek	Çok Yüksek	En Yüksek
35	Güvenlik Kontrol Yönetimi	Önemsiz	Çok Düşük	Düşük	Yüksek	Çok Yüksek	En Yüksek

Şekil 3.2. Uygulama için otuz-beş parçalık ve altı fonksiyonel nokta değerli kriter seti

4.Uygulama

İncelenen Proje: MRP Uygulaması

Malzeme ihtiyaç planlama, üretim aşamasından gelen ham maddenin just in time a göre tezgâhlara alınmasını sağlayan tedarik ve malzeme yönetim sistemi.

Otuz beş parçalık kriter seti uygulanarak aşağıdaki sonuçlar elde edilmiştir.

Ci	F1	F2	F3	F4	F5	F6	Toplam
C1	0	0	0	1	1	0	2
C2	0	0	0	1	1	1	3
C3	1	1	1	1	1	1	6
C4	1	1	1	0	0	0	3
C5	0	0	1	1	1	1	4
C6	0	1	1	1	1	0	4
C7	1	1	1	1	1	1	6
C8	1	1	1	1	1	1	6
C9	0	0	0	1	1	1	3
C10	0	0	0	0	1	1	2
C11	0	0	0	1	1	1	3
C12	0	0	0	1	1	1	3
C13	0	0	1	1	1	0	3
C14	0	1	1	1	0	0	3
C15	0	0	1	0	0	0	1
C16	0	0	0	1	1	1	3
C17	0	0	1	1	1	0	3
C18	0	0	0	0	0	1	1
C19	0	0	0	1	1	1	3
C20	0	0	0	1	1	1	3
C21	0	0	0	0	1	1	2
C22	0	0	0	1	1	1	3
C23	0	0	1	1	1	1	4
C24	0	0	1	1	1	1	4
C25	0	1	1	1	0	0	3
C26	0	0	0	0	1	1	2
C27	0	0	0	0	1	1	2
C28	1	1	1	0	0	0	3
C29	0	0	0	1	1	1	3
C30	0	0	0	1	1	1	3
C31	0	0	0	1	1	1	3
C32	1	1	0	0	0	0	2
C33	0	0	0	0	1	1	2
C34	1	1	0	0	0	0	2
C35	0	1	1	0	0	0	2
Toplam							105

Şekil 4.1. MRP uygulaması

Elde Edilen Düşük-Orta: 33
Elde Edilen Yüksek-Orta: 72

İncelenen uygulamada elde edilen Düşük-Orta, Yüksek-Orta ve toplam değerlerine bakılarak Yeniden Kullanılabilir Yazılım geliştirme Süreç Modeli kullanıldığını söyleyebiliriz.

İncelenen Proje: CRM Uygulaması

Gelen işlerin projelendirilip ürün ağaçlarına göre parçalanarak personele dağıtıldığı; personelin yapılan işleri, gereken malzemeleri ve tarihlerini girdiği; her proje için kullanılan malzemenin depoda takip edildiği; bunlara ek olaraktan bütçe planlamasının yapıldığı yazılım.

Otuz beş parçalık kriter seti uygulanarak aşağıdaki sonuçlar elde edilmiştir.

Ci	F1	F2	F3	F4	F5	F6	Toplam
C1	0	0	0	1	1	1	3
C2	0	0	0	1	1	1	3
C3	0	1	1	1	1	1	5
C4	0	1	1	1	1	0	4
C5	0	1	1	1	1	1	5
C6	0	0	0	1	1	1	3
C7	0	1	1	1	1	1	5
C8	1	1	1	1	1	1	6
C9	0	0	0	1	1	1	3
C10	0	0	0	1	1	1	3
C11	0	0	0	1	1	0	2
C12	0	0	0	1	1	1	3
C13	0	0	0	1	1	1	3
C14	0	0	0	0	1	1	2
C15	0	1	1	1	0	0	3
C16	0	1	1	1	0	0	3
C17	0	1	1	1	1	0	4
C18	0	0	0	0	1	1	2
C19	0	0	0	1	1	1	3
C20	0	0	0	1	1	1	3
C21	0	0	0	0	1	1	2
C22	0	0	0	1	1	1	3
C23	0	0	1	1	1	0	3
C24	0	0	1	1	1	1	4
C25	0	1	1	1	1	0	4
C26	0	0	0	1	1	1	3
C27	0	0	0	0	1	1	2
C28	0	0	0	1	1	1	3
C29	0	0	1	1	1	1	4
C30	0	0	0	0	1	1	2
C31	0	0	1	1	1	1	4
C32	0	0	0	0	1	1	2
C33	0	0	0	1	1	1	3
C34	0	0	0	1	1	1	3
C35	0	0	0	1	1	1	3
Toplam							113

Şekil 4.2. CRM uygulaması

Elde Edilen Düşük-Orta: 23
Elde Edilen Yüksek-Orta: 90

İncelenen uygulamada elde edilen Düşük-Orta, Yüksek-Orta ve toplam değerlerine bakılarak Helezonik(Sarmal) Yazılım geliştirme Süreç Modeli kullanıldığını söyleyebiliriz.

5. Sonuçlar

İstenilen süreç modelindeki güvenlik ölçümleri Yüksek-Orta alana doğru eğimlidir. Verilen yazılım projesindeki takım yönetimine olan güvenin derecesi yüksek olmalıdır. Eğer güven düşük olursa, proje başlamadan önce bütün takım yönetimi değiştirilmelidir yoksa sözde güvenilir çalışan sistem veritabanı bütünlüğüne ve sistem mimarisine zarar verecek hatalar oluşturur. Güvenlik proje sistem yaşam döngüsü içinde yazılım geliştiren tüm katılımcılar arasında sağlanmalıdır.
Kullanıcıların davranışları ve anlatım kabiliyetleri, Yüksek-Orta alana dayalı bir takım tasarlamak için çok yardımcı ve takdir edilen olacaktır. Ayrıca problem alanında deneyimli ve uzman olmak; yazılım araçları, yöntemleri, teknikleri, ürün ve geliştirme aşamasındaki diller hakkında temel bilgi sahibi olmak ile de daha iyisi yapılacaktır.[8]
Yazılım süreç seçim kriterlerinin seçilebilmesi için herhangi basit bir yaklaşım bulunmamaktadır. Bu çalışmada geliştirilen uygulama ile otuz beş parçalık kriter seti yazılımlara uygulanarak kullanılan yazılım süreç modeli bulundu. Çeşitli yazılım uygulamaları ve süreçleri incelendiğinde, değişik yazılım projeleri için bir seçim yöntemi tanımlamak için bazı kurallar tanımlanmış olduğu görülmektedir. Bir gerçek vardır ki bir proje için yeterli olan diğer bir proje için yetersiz olmaktadır.

6. Kaynaklar

[1] T.G. Olson, N.R. Reizer, J.W. Over, Handbook CMU/SEI-94-HB-01: A Software Process Framework for the SEI Capability Maturity Model, SEI, Eylül 1994

[2] C. Ghezzi, M. Jazayeri, D. Mandrioli, Fundamentals of Software Engineering, Prentice Hall, Pearson Educatio Inc., New Jersey, 2003.

[3] Defining an Adaptive Software Security Metric from a Dynamic Software Failure Tolerance Measure A. Ghosh, G.McGraw, F. Charron E. Miller

[4] http://www.mehmetduran.com/detay.aspx?detay=299

[5] http://www.bilgisayarkavramlari.com/2008/11/29/selale-modeli-waterfall-model

[6] L.Alexander and A. Davis, 1991. “Criteria For Selecting Software Process Models”. In Proceedings of COMPSAC’91 Tokyo, Japan, 09-11-1991 to 09.13.1991, pp. 521-528

[7] web.iku.edu.tr/~gyilmaz/Notes/ YazilimMuhendisligiYonetimi

[8] A. M. Davis, E.H. Bersoff and E. R. Comer. 1988. Astrategy for comparing alternative software development life cycle models. Software engineering. IEEE Transactions on 14, (10): 1453-1461.