YAZILIM YAŞAM-DÖNGÜ MODELLERİ

220601059-Nida Elvin Mertoğlu

Özet

Bu yazı kapsamında, bazı yazılım geliştirme yaşam döngüsü (Software Development Life Cycle , SDLC) modelleri incelenmiştir. Ele alınan yazılım yaşam-döngü modelleri, şelale modeli (waterfall model), v-şekil modeli (v-shaped model), evrimsel geliştirme (evolutionary development) ve çevik yazılım geliştirme süreci kapsamında SCRUM ve uç değer programlamadır.(extreme programming – XP)

Yazılım Geliştirme Yaşam Döngüsü (Software Development Life Cycle ,SDLC)

Yazılım Geliştirme Yaşam Döngüsü (Software Development Life Cycle , SDLC)basitçe yazılımları geliştirmek tasarlamak ve test etmek amaçlı kullanılan bir süreç olarak ifade edilebilir. Yüksek kaliteli yazılımlar tasarlamak ve oluşturmak için uygun maliyetli ve zaman açısından verimli bir süreç olarak SDLC kullanılır. SDLC yazılımın nasıl gelişeceğini, bu sürecin nasıl ilerleyeceğini ve nasıl daha iyi hale getireleceğini kapsayan bir plandan oluşur. SDLC üretim aşamasında ve sonraki aşamalarda müşterinin beklentilerini karşılamak için ileriye dönük planlamalar aracılığıyla proje risklerini en aza indirmeyi ve maliyet, süre dâhilinde tamamlanan kaliteli yazılım üretmeyi hedefler. SDLC temel aşamalardan oluşur; Planlama, analiz, tasarım, üretim(gerçekleştirme), test etme, bakım.

Yazılım Geliştirme Yaşam-Döngü Modelleri

Şelale/Çağlayan Modeli(Waterfall Model)

Şelale/Çağlayan Modeli(waterfall model) Dr. Winston W. Royce tarafından 1970 yılında tanıtılmıştır. Geleneksel bir yöntem olup sıralı ve doğrusal bir akışta ilerler. Günümüzde kullanımı gitgide azalmasına rağmen halen endüstriyel tasarım uygulamaları için kullanılmaktadır. Waterfall, yazılım geliştirme sürecini sıralı aşamalara ayırır. Her yeni aşama önceki aşamanın sonucuna bağlı olacak şekilde sıralanır. Bir aşama tamamlanmadan diğer aşamaya geçilmez. Bir aşamadan diğer aşamaya geçilirken çoğunlukla o aşama için tanımlanan tüm hedeflere ulaşıldığına emin olmak için gözden geçirilir ve emin olunur. Şelale modeli gereksinim toplama ve analiz adımıyla başlar. Gereksinim toplama ve analiz adımında yazılım için müşterinin belirttiği gereksinimler belirlenerek bir analiz dokümanı oluşturulur. Sonra tasarım adımına geçilir. Tasarım adımında ilk aşamadaki gereksinimler ve analiz doğrultusunda bir arayüz, veritabanı vb tasarımlar yapılarak tasarım dokümanı oluşturulur. Daha sonra kodlama adımına geçilir. Kodlama adımında önceki aşamalardaki dokümanlar doğrultusuyla bir sözde kod oluşturulur. Kod küçük parçalar, üniteler doğrultusunda kodlanır daha sonra birleştirilir. Kodlama adımı bittikten sonra test adımına geçilir. Test adımında gereksinim toplama ve analiz ve tasarım dokümanlarındaki tüm fonksiyonel ve fonksiyonel olmayan gereksinimler ve tasarımlar için test senaryoları yazılır ve test yazılımları yapılır. Bu

testler sonucu gereksinim toplama ve analiz, tasarım veya kodlama adımlarından hangisinden kaynaklı bir hata oluştuysa o adıma gidilir ve hata düzeltilir. Test adımı sonucu herhangi bir hata olmazsa entegrasyon adımına geçilir. Canlı ortama entegre edilir ve müşterini kullanımına açılır veya piyasaya sürülür. Bakım aşamasında ürünü geliştirebilmek ve ürünün kullanabilirliğini artırmak için daha iyi sürümler yayınlanabilir. Gereksinimlerin çok sık değişmediği, açık ve sık belgelendiği, yerleşik bir zaman çizelgesi olan, ürünü desteklemek için gerekli uzmanlığa sahip geniş kaynakları mevcut olan, teknoloji anlaşılır ve dinamik olmadığı projelerde kullanılabilir. Avantajları arasında gereksinim toplama ve analiz detaylı bir şekilde yapıldığından sağlam bir temel yapılır, proje yöneticileri tarafından görev dağılımı kolay yapılır ve müşteriler ve kullanıcılar tarafından anlaşılan adımlar oluşturulur. Dezavantajları arasında kullanıcılar süreç içinde yer almadığından süreç bitiminde geri dönüş alınır, karmaşık ve nesne yönelimli projeler için iyi bir yöntem değildir, değişime adapte olması zor olduğundan her değişim maliyeti arttırır.

V-Shaped Model(V-Model)

V modeli aynı zaman doğrulama ve doğrulama modeli olarak da bilinir ve işlemin V şeklinde sıralı bir şekilde yürütüldüğü bir tür SDLC modelidir. Verification(Doğrulama) ve Validation(Onaylama) mekanizmaları eklenmiş waterfall modeline de benzetilebilir. Her bir geliştirme aşamasına karşılık test aşaması bulunur. Bir aşama geliştirmesi ve test faaliyeti bitmeden diğer aşamaya geçilmez. Geliştirme aşamaları ile paralel olan test aşamaları asıl kodlama aşamasında bir arada ele alınır. Veritification (Doğrulama) aşamasında spesifik gereksinimlerin karşılanıp karşılanmadığına bakılmak için geliştirme süreci değerlendirilir ve kod yürütülmeden statik bir analiz gerçekleştirilir. Validation(Onaylama) aşamasında ise tüm fonksiyonel ve fonksiyonel olmayan gereksinimler için dinamik bir analiz kod çalıştırılarak yapılır ve yazılım müşterinin beklentilerini ve gereksinimlerini karşılayıp karşılamadığına bakılır. Veritification (Doğrulama) belli adımlardan oluşur: gereksinim analizi, sistem tasarımı, mimari tasarım, modül tasarımı, kodlama. Validation(Onaylama) belli adımlardan oluşur: unit testi, entegrasyon testi, sistem testi, kabul testleri. Gereksinimlerin iyi tanımlandığı, teknik uzmanlığa sahip teknik kaynakların çokça olduğu projelerde kullanılabilir. Avantajları arasında anlaşılması kolay olması , proje gereksinimleri net olan küçük projelerde kullanılması , planlama ve test tasarımı gibi test metotlarının kodlamadan önce uygulanması yer alır. Dezavantajları arasında kompleks projeler için elverişli olmaması, risk çözümleme aktivitesi olmaması, aşamalar arası tekrarlamama olmaması yer alır.

Evrimsel Geliştirme (Evolutionary Development)

İlk tam ölçekli olan modeldir. Modelin başarısı ilk evrimin başarısına bağımlıdır. Geliştirme sürecini küçük parçalar böler ve sürekli değerlendirme yapılır. İki çeşit evrimsel geliştirme vardır: keşifçi geliştirme (exploratory development) ve atılacak prototipleme (throw-away prototyping). Keşifçi geliştirme (exploratory development) müşteri ile çalışılıp gereksinimleri inceleyerek son sistemi teslim etmeyi hedefler. İyi anlaşılır gereksinimler ile sürece başlanır.

Prototipleme (throw-away prototyping) sistem gereksinimlerini anlamayı hedefler. Tam anlaşılmamış gereksinimlerle sürece başlanır. Coğrafik olarak geniş alana yayılmış, çok birimli organizasyonlar için kullanılabilir. Avantajları arasında risk ve hata oranının az olması, sürekli değerlendirmeler sayesinde erken aşamalardaki geliştirme risklerini azalması ve kullanıcıların kendi gereksinimlerini daha iyi anlamasını sağlamak vardır. Dezavantajları arasında düzenli teslim edilebilir ürün olmaması, bakımı zor olması, sürekli değişikliklerden dolayı yazılımın yapısına zarar verilebilir bu yüzden sistemlerin sıklıkla iyi yapılamaması vardır.

Çevik Yazılım Geliştirme(Agile Software Development)

Çevik yazılım geliştirme yinelemeli (iterational) olarak yazılım geliştirme sürecinin uygulanması ve tedrici (incremental) olarak yazılım ürününün doğru bir şekilde, değişime açık, Agile(Çevik)Manifestosu doğrultusunda prensiplerini uygulayarak geliştirilmesidir. Agile'ı diğer yazılım geliştirme yaklaşımlarından ayıran bir özellik de işi yapan insanlara ve onların birlikte nasıl çalıştıklarına odaklanmasıdır. Agile (Çevik) Manifestosu 4 temel değer ve 12 prensipten oluşur. 4 temel değer: Süreç ve araçlardan ziyade bireyler ve etkileşim, kapsamlı dokümantasyondan ziyade çalışan bir yazılım, sözleşmeye ve pazarlıklara odaklanmak yerine müşteri iş birliğine odaklanmak, bir plana bağlı kalmak yerine talep edilen değişime karşılık vermektir. Agile(Çevik)Manifestosunun altında yatan temel prensipleri arasında müşteriye hızlı ve sürekli kullanılabilir yazılım ürünü teslim etmek, gereksinim değişiklikleri için esnekliği ön planda tutma, kısa zamana aralıklarında ekip halinde birebir etkileşimle birlikte çalışmak vardır. Amaçları arasında değişime uyum sağlamak, hızlı bir şekilde ürünü müşteriye teslim etmek, az zamanda en çok verimi elde etmek vardır. Projeyi küçük iterasyonlara ayırır. Her iterasyon sonucunda müşteriye bilgi verilir.

Uç Değer Programlama (Extreme Programming – XP)

Grup içi iletişime önem verir. Geri dönüşlerin daha fazla olmasına imkân sağlar. Uç değer programlama-XP ile proje yaşam döngüsü (life cycle) çerçevesinde proje geliştirilir. Proje yaşam döngüsü bazı uygulamalara (practices) bağlı geliştirilir. Yazılım geliştirmede kolaylığı ve esnekliği sağlamak için; uç değer programlama-XP 12 farklı pratik öngörür. Bu 12 farklı pratik: Planlama oyunu, ekipte müşteri, önce test, basit tasarım, çiftli programlama, sürekli entegrasyon, kısa aralıklı sürümler, yeniden yapılandırma, ortak kod sahipleme, metafor, kodlama standardı ve haftada 40 saattir. Uç değer programlama-XP için 4 temel değer vardır bunlar iletişim, basitlik, geri bildirim, cesarettir. Avantajları arasında proje süresini kısaltma, hata oranını azaltma, değişikliklere izin verme yer alır. Küçük projelerde ve sürekli değişiklik gerektiren projeler için uygulanabilen bir modeldir.

SCRUM

Scrum, ekiplerin kendi kendini organize ederek ve ortak bir hedef doğrultusunda çalışmasını hedefleyen bir yönetim çerçevesidir. Verimli bir proje teslimi için gereken toplantı, araç ve roller tanımlanır. Scrum yazılım ekipleri tarafından karmaşık sorunları uygun

maliyetle ve sürdürülebilir bir şekilde çözmek için kullanır. Scrum sadece yazılım geliştirmede ,her kullanılmaz ,her projeye uyum sağlayabilir. Karmaşık yazılım işlemlerini küçük birimlere(sprint) bölerek tekrara dayalı bir sistemle geriştirilir. Düzenli geri bildirimler ve planlamalar ile hedefe ulaşılmak amaçlanır. Müşterinin ihtiyacına göre şekillendiğinden dolayı müşteri geri bildirimlerine göre yapılanma sağlanır. İletişim ve takım çalışması ön plandadır. Karmaşıklığı üç ilke yoluyla azaltmak hedeflenmiştir. Bu ilkeler şeffaflık, denetleme ve uyarlamadır. Scrum ekipleri beş temel değeri izler. Bu temel değerler bağlılık, cesaret, odaklanma, açıklık ve saygıdır. Scrum, öğrenmesi kolay olmasına rağmen uzmanlaşması zor bir çerçevedir. Scrum'ın kurucu ortakları olan Jeff Sutherland ve Ken Schwaber tarafından Scrum Kılavuzu'nda temel kavramlar açıklamıştır. Scrum kılavuzu, Scrum süreçlerine ve bunların nasıl etkili şekilde uygulanacağına dair ayrıntılı bir genel bilgi verir. Sprint adı verilen iki ya da dört haftalık dönemler içerisinde müşteri/kullanıcı tarafından istenilen ve tanımlanana gereksinim ve işlevler yeniden gözden geçirilir. Her sprint sonucunda yazılımın fonksiyonel bir parçası biter ve müşteriye teslime hazır bir hale gelmiş olur. Günlük toplantılar yapılarak iş takibi sağlanır. Scrumda üç temel kavram vardır. Bunlar: roller, toplantılar ve araçlardır. Toplantılar kavramı altında sprint gözden geçirme ve günlük scrum toplantısı yer alır. Araçlar kavramı altında ürün gereksinim dokümanı(product backlog),sprint dokümanı(sprint backlog), sprint kalan zaman grafiği(burndown chart) yer alır. Scrum yapıtları adlı araçlar Scrum ekipleri tarafından sorunları çözmek ve projeleri yönetmek için kullanılır. Scrum yapıtları önemli planlama ve görev bilgilerini sunar. Scrum yapıtları: Ürün iş listesi, sprint iş listesi ve ürün parçadır. Scrum ekibi üç role ihtiyaç duyar. Bunlar: Ürün sahibi, scrum lideri ve geliştirme ekibidir. Scrum etkinlikleri Scrum ekiplerinin düzenli olarak gerçekleştirdiği toplantılardır. Bunlar sprint planlama, sprint, günlük scrum veya stand-up, sprint değerlendirme, sprint retroperspektifidir. Avantajları arasında zorlu durumlarsa kaliteyi koruyabilme, daha mutlu ve üretken ekipler ve sürekli değişime adapte olan çevik bir yapı olması vardır.

Scrum günümüzde neden popüler?

Scrum, öğrenmesi kolay olmasına rağmen uzmanlaşması zor bir çerçevedir. Ama kısa sürede kaliteli ürünler çıkarabilme ve başarı garantisinin yüksek olması nedeniyle günümüzde popülerdir. Google, Microsoft, Yahoo gibi büyük şirketler tarafından tercih edilmektedir.

KAYNAKLAR

- https://scrumguides.org/scrum-guide.html

- https://www.javatpoint.com/software-engineering-v-model

- https://www.geeksforgeeks.org/software-engineering-evolutionary-model/

- https://www.geeksforgeeks.org/software-engineering-extreme-programming-xp/

- https://www.geeksforgeeks.org/software-engineering-classical-waterfall-model/

- https://aws.amazon.com/tr/what-is/sdlc/