

Enerji Otomasyonu Projelerinde Yazılım

Y.Müh. Seda Canıgür
Üçgen Otomasyon EEB Ltd. Şti.,
Kontrol ve Bilgisayar

Enerji otomasyonu projeleri, PLC ve SCADA yazılım ağırlıklı olmalarından dolayı oldukça zor projelerdir. Bu projelerin başarılı olup olmamalarını etkileyen çeşitli kriterler bulunmaktadır. Yazılım tarafından baktığımızda, bunlardan ilk aklımıza gelenler yazılım ekibinin izlediği yazılım geliştirme süreci ve ekibin bilgi ve becerisi, iş kapsam ve tanımının tam olarak yapılıp yapılmadığı, seçilen yazılım geliştirme ortamının üretilmek istenen yazılım konusunda bir yetersizliğinin olmaması gibi kriterlerdir. Acaba bu tür projelerde, yazılımda başarı nedir ?

Yazılımda Başarı

Yazılımda başarıyı, aşağıdaki üç konuda projenin beklenenleri sağlaması ya da daha iyi bir performans göstermesi diye düşünebiliriz:

- Zaman planı
- Maliyet
- Kalite

Zaman planı yazılımın çeşitli hedef noktalarına (milestone) belirlenen zamanda varılmasını ve hedefleri yerine getirmesini kapsar. Bir enerji otomasyonu projesinde bulunması beklenen "analiz kapanış", "tasarım kapanış", "kod geliştirme kapanış" gibi hedeflere vaktinde ulaşılması buna örnek oluşturur.

Maliyet ise yine proje tamamlan-

dığında öngörülen bütçe sınırları dahilinde bu hedefe varmaktır.

Enerji otomasyonu projelerinde yazılım kalite beklentisi ise iki boyutludur. Bunlardan ilki, tamamlanan yazılımın başlangıçta belirlenen gereksinimleri ne kadar sağladığıdır. İdeal durumda projelerin tüm gereksinimleri yerine getirmesi beklenmektedir. Ancak çoğu durumda yazılım kısmını "hata" dediğimiz problemler ile birlikte tamamlanır. Bu nedenle kalitede başarıyı aradığımız zaman, bunun istenenleri tamamıyla yapmak ve ortaya çıkan sonucun, projenin müşterisi ve kullanım alanının kabul edebileceği bir hata düzeyinde teslim etmek olduğunu görüyoruz. Örneğin, enerji kesintisinin çok kritik olduğu büyük bir endüstriyel tesisin enerji dağıtım merkezini kontrol ve kumanda eden bir yazılımda bulunması kabul edilebilecek hata oranı, bir alışveriş merkezinde enerji analizörlerini izlemeye yönelik bir yazılımdan çok daha düşük olacaktır.

Yazılım endüstrisinde hataların sık sık dört kategoride sınıflandırıldığını görüyoruz:

- Kabul edilemez (Show stopper)
- Kritik önemli (Critical)
- Önemli (Important)
- Az önemli (Cosmetic)

Enerji otomasyonu projelerinin kabul kriterleri sırasında özellikle yazılacak program büyüklüğüne

Seviye	Olgunluk	Karakteristik	Açıklama	
5	Optimum	Geliştirilebilen	Yönetilen yöntemle ilave olarak veriye dayalı sürekli iyileştirmenin yapıldığı uygulamadır.	Verimlilik-Kalite ↑
4	Yönetilen	Ölçülebilir	Tanımlı yöntemle geliştirme sürecinin anlaşılması ve kontrol edilmesini sağlayan ölçülendirme sisteminin eklendiği uygulamadır.	
3	Tanımlı	Kurumsallaşmış	Geliştirme süreci yazılı olarak tanımlanmış ve yazılımı geliştiren ekibin prosedürleri anladığı ve izlediği uygulamadır.	
2	Tekrarlanabilir	Kişiyeye bağımlı	Proje planlaması, izleme, kalite güvence ve konfigürasyon yönetimi ilave edilerek yapılan kaotik uygulamadır.	
1	Başlangıç	Kaotik	Yazılım, üstün performanslı kişilerin bir plan ya da method olmadan, tecrübeye ve sezgiye dayanarak anında uygulama yöntemiyle geliştirilir. Proje yönetimi, proje planlama, konfigürasyon yönetimi ve yazılım kalite güvencesi bakımından eksiktir.	Risk ↓

Tablo-1 Yazılım süreçlerinin olgunluk seviyeleri

bağlı olarak yukarıdaki hata sayıları ile ilgili kabul kriterleri belirtilebilir.

Yazılım geliştirme, genellikle yalnızca kod yazımından ibaret görülüyor. Oysa bu temelden yanlış bir yaklaşım. Tüm dünyada 365 bin üyesi bulunan ve 150 ülkede faaliyet gösteren meslek örgütü IEEE'nin bu konuda yön gösteren çalışmaları bulunuyor.

IEEE'nin katkısıyla oluşturulan Yazılım Mühendisliği Bilgi Tanımı (Software Engineering Body of Knowledge - SWEBOOK) on ayrı kategoriyi kapsıyor. Konfigürasyon yönetimi, üretim, tasarım, mühendislik altyapısı ve mühendislik yönetimini kapsayan bu kategorilerin tümü, ortaya sağlıklı bir yazılım çıkması açısından önem taşıyor. Kodlama ise bu kategorilerden birinin alt başlığı konumunda. Bu açıdan bakığımızda, yazılım yapan şirketlerin izlediği süreçlerin olgunluğu konusunda modellemelerle karşılaşmaktayız.

Yazılım Olgunluk Modeli

Yazılım olgunluk modeli (Humphrey, Kitson & Kasse 1989), yazılım yapan şirket ve organizasyonların izlediği süreç ve yöntemleri sınıflandıran bir modeldir. Bu modele göre yazılım geliştirme süreci 5 farklı olgunluk seviyesin-

“ Özellikle enerji kalitesi ve sürekliliğinin önem arzettiği tesislerdeki enerji otomasyon projelerinde, yazılım geliştirme önemle takip edilmesi gereken bir süreçtir. Bu tür projelerde kaotik ya da altındaki bir seviyede çalışan ekiplerden başarı beklemek, üstün performanslı birisinin mucizeler yaratarak projeyi kurtarmasını beklemekle ve projeyi gelecekte o kişiye bağımlı kılmakla eşanlamlıdır ”

de değerlendirilebilir. Seviye 1 en düşük dereceyi (risk fazla, verimlilik ve kalite az), Seviye 5 ise en yüksek dereceyi (risk az, verimlilik ve kalite fazla) ifade etmektedir. Bu modele göre, diğer sektörlerde de olduğu gibi, enerji otomasyonu sektöründe yazılım faaliyeti gösteren her bir şirket bu seviyelerden birisi ile özdeşleştirilebilir.

Yazılım Gelişmemişlik Modeli

Yazılım Gelişmemişlik Modeli (Finkelstein 1992) Yazılım Olgunluk Modeli Seviye 1'in de altındaki seviyeleri tanımlar. SEI (Software Engineering Institute – Yazılım Mühendisliği Enstitüsü) istatistiklerine göre dünya üzerinde yazılım yapan kuruluşların %70'i en alt seviye olan başlangıç seviyesindedir. Aslında bunların çoğu kaotik yapının da altında, Seviye 0, Seviye -1 ve Seviye -2 olarak tanımlanmayı hak etmektedirler. Yazılım Olgunluk Modeli Seviye 1 şirket ve organizasyonların uyguladığı kaotik

Seviye	Gelişmemişlik	Karakteristik	Açıklama	Anlayış---^ ---Ahmakklık ^---Anlayış
0	Savsak	İhmalkar	Başarılı geliştirme sürecine ulaşmayı engelleyen uygulamadır.	
-1	Sersem	Zorluk Çıkarıcı	Verimsiz geliştirme sürecinde ısrar eden uygulamadır.	
-2	Ahmak	Hor Gören	Yazılım geliştirme süreçlerinde kurumsallaşmayı umursamayan, hatta karşı çıkan uygulamadır.	

Tablo-2 Yazılım süreçlerinin gelişmemişlik seviyeleri

yöntemde, şahısların üstün gayretleri ile yazılım geliştirme yapılabilir. Seviye 0'dakiler bu gayreti engelleyici tutum içerisindedirler. Oluşturulan dokümanı ve tanımlanan yazılım özelliklerini önemsemeyenler, hatta kaybederler. Yazılmış programlarla ilgili olarak konfigürasyon yönetimini yapamaz ve yanlış versiyonları uygularlar.

Gelişmemiş şirket ve organizasyonlar, süreç yönetimlerinin oldukça çarpık olduğunun farkında değildirler. Teknik bir düzeltmenin sorunları ortadan kaldıracığına inançları tamdır. Bu tür kuruluşlar için yönetsel meseleler "öncelikli konular listesi" içerisinde yer almamaktadır.

Seviye 0 kuruluşları öncelikli teknik problemin yazılımların tekrar kullanılması sırasında ortaya çıkacağına inanırlar. Bir yazılımın tekrar kullanılması sırasında önceki projede geliştirme sürecinde yaptıkları hatalardan daha büyük bir hata yapmayacaklarının garantisinde olduğuna inanırlar. Seviye 0 kuruluşları ihmalkarlıkları sebebiyle etkili geliştirmenin önüne geçerler. Seviye -1 kuruluşları ise yazılım geliştirme düzenini bozucu yönde hareket etmektedirler. Bunlar karmaşık ya-

pılarda ısrarcı davranmakta, standart dışı dökümantasyon yapmaktadırlar.

Seviye 0 kuruluşları için en önemli teknik kıstas yazılım geliştirme ortamları ve kaynaklarıdır. Uygun bir ortamda politikalarını ve prosesi sürekli geliştirebileceklerine inanmaktadırlar. Tüm dökümantasyonu oluşturmak ve kontrol etmek için kendi standartlarını geliştirebileceklerini, aynı zamanda bu tür bir ortamda karmaşık sorunlara çözüm getiren uygulamaları yapabileceklerini düşünmektedirler.

Seviye -1 organizasyonları benzer davranışlarla yazılım geliştirmeyi engellerken, aslında yardımcı olduklarını düşünmektedirler. Seviye -2 kuruluşları ise yazılım geliştirme süreçlerinin iyileştirilmesi konusunda oldukça küçümseyici yaklaşmaktadırlar. Zayıf yazılımlar yapmakta olduklarına aldırılmazlar çünkü muhtemelen daha sonradan bakım ve servis hizmetleri ile ilk yaptıkları yazılımdan daha fazla para kazanacaklardır. Bu tür kuruluşlarda, yazılım geliştirme sürecini tarif eden döküman (eğer varsa tabii ki) uzun süre önce işten ayrılmış bir mühendis tarafından yıllar önce oluşturulmuştur. Bu dökümanı

kimsenin okumamasıyla gurur duyarlar, zaten okumak isteyen çıksa da bu dökümanı bulamazlar. Bu tür organizasyonlar başarısızlığı hak etmektedirler. Bunların işe yarar yazılım üretmeleri mucizelere bağlıdır.

Sonuç

Özellikle enerji kalitesi ve sürekliliğinin önem arzettiği tesislerdeki enerji otomasyon projelerinde, yazılım geliştirme önemle takip edilmesi gereken bir süreçtir. Bu tür projelerde kaotik ya da altındaki bir seviyede çalışan ekiplerden başarı beklemek, üstün performanslı birisinin mucizeler yaratarak projeyi kurtarmasını beklemekle ve projeyi gelecekte o kişiye bağımlı kılmakla eşanlamlıdır. Bu tür projelerde yer alan mühendislik ekiplerinin en az Seviye 3 yazılım geliştirme sürecini takip ediyor olması gerekliliği açıktır. Ancak ülkemizde Seviye 3 süreç yönetimini izleyen otomasyon mühendislik grubuna rastlamak bile sık karşılaşılan bir durum değildir.

Kaynaklar

- 1- Finkelstein, A., SIGSOFT Yazılım Mühendisliği Notları, 1992
- 2- Erener, Ö., BTİnsan, 2003