**Software System Engineering: A Tutorial**

Sistem mühendisliği ilkeleri özellikle büyük, karmaşık yazılım sistemlerinin geliştirilmesine uygulamak, süreç ve ürün yönetimi için güçlü bir araç sağlar. Yazılım sistemleri her zamankinden daha büyük ve daha karmaşık hale geldi. Bu büyümenin bir bölümünü donanım performansındaki ilerlemelere bağlayabiliriz - bir yazılım sisteminin boyutunu ve karmaşıklığını birincil tasarım hedefi olarak sınırlama ihtiyacını azaltan ilerlemeler. Microsoft Word klasik bir örnektir: 20 yıl önce 360 Kbyte diskete sığacak bir ürün için şimdi 600 Mbyte CD gerekir. Ancak artan boyut ve karmaşıklığın başka nedenleri de var.

Spesifik olarak, çoğu teknik sistemde olmasa da, çoğu yazılımda baskın teknoloji haline geldi. Genellikle karmaşık bir sistemin problemleri çözmesini sağlayan tutarlılık ve veri kontrolü sağlar. Bir hava trafik kontrol sisteminde, yazılım bir uçağı hedefine başarıyla yönlendiren uçakları, insanları, radarları, iletişimleri ve diğer ekipmanları birbirine bağlar. Yazılım, sistemin ana teknik karmaşıklığını sağlar.

Büyük yazılım sistemlerinin büyük çoğunluğu öngörülen programlarını ya da tahmini maliyetlerini karşılamıyor ve sistem alıcısının beklentilerini tam olarak yerine getirmiyorlar. Bu fenomen uzun zamandır yazılım krizi olarak biliniyor. Bu krize cevaben, yazılım geliştiriciler, ürün geliştirmeye farklı mühendislik uygulamaları tanıttılar.

Bir geliştirme projesinin yönetsel ve teknik durumunu (kullanılan kaynaklar, kullanılan kilometre taşları tamamlandı, gereksinimler karşılandı, testler tamamlandı) izlemek, sağlığı hakkında yeterli geribildirim sağlamıyor. Bunun yerine, ürünlerinin yanı sıra teknik süreçleri de yönetmeliyiz. Sistem mühendisliği, bu teknik yönetim görevinin gerektirdiği araçları sağlar.

Sistem mühendisliği ilkelerinin bir bilgisayar yazılımı sisteminin geliştirilmesine uygulanması, yazılım sistemi mühendisliği veya SwSE denilen faaliyetler, görevler ve prosedürler üretir. Birçok uygulayıcı SwSE'yi özel bir sistem mühendisliği örneği olarak görür ve diğerleri de bunu yazılım mühendisliğinin bir parçası olarak görür. Ancak, SwSE'nin büyük yazılım projelerinin teknik gelişimini yönetmek için farklı ve güçlü bir araç olduğunu iddia edebiliriz.

Bu eğitim IEEE yazılım mühendisliği standartlarından tanımları ve süreçleri SwSE sürecine entegre eder. SwSE'yi uygulamak için adım adım ayrıntılı bir yaklaşım içeren daha uzun bir versiyon, IEEE Computer Society’nin “en iyi uygulamalar serisinin bir parçası olan Yazılım Mühendisliği Cilt 1: Geliştirme Süreci’nde mevcuttur.

**SİSTEMLER VE SİSTEM MÜHENDİSLİĞİ**

Sistem, ortak bir hedefe ulaşılmasını sağlayacak şekilde ilgili bir öğeler topluluğudur. Bilgisayar sistemlerinde, bu öğeler donanım, yazılım, insanlar, tesisler ve süreçleri içerir.

Sistem mühendisliği, operasyonel bir ihtiyacı, bu ihtiyacı en iyi şekilde karşılayan bir sistem konfigürasyonunun tanımlamasına dönüştürmek için gereken bilimsel, mühendislik ve yönetim becerilerinin pratik uygulamasıdır. Sistem geliştirme projesinin genel teknik yönetimi için geçerli olan genel bir problem çözme sürecidir. Bu süreç, bir sistemin ürün ve süreç tanımlarını tanımlamak ve geliştirmek için bir mekanizma sağlar.

IEEE Std. 1220-1998, sistem mühendisliği sürecini ve ürün yaşam döngüsü boyunca uygulanmasını açıklar. Sistem mühendisliği, donanım değil, belgeler üretir. Belgeler, gelişim süreçlerini projenin yaşam döngüsü modeli ile ilişkilendirmiştir. Ayrıca, proje boyunca beklenen süreç ortamlarını, arayüzleri, ürünleri ve risk yönetimi araçlarını tanımlarlar.

Sistem mühendisliği beş işlevi içerir:

• Problem tanımı, gereksinimleri analiz ederek ve edinenle etkileşime girerek ihtiyaçları ve kısıtlamaları belirler.

• Çözüm analizi, gereksinimleri ve kısıtlamaları yerine getirmenin olası yollarını belirler, olası çözümleri analiz eder ve en uygun olanı seçer.

• Süreç planlama yapılacak işler, ürünün geliştirilmesi için büyüklük ve çaba, görevler arasındaki öncelik ve projeye yönelik potansiyel riskler belirler.

• Süreç kontrolü, projeyi ve süreci kontrol etme yöntemlerini belirler, ilerlemeyi ölçer, ara ürünleri gözden geçirir ve gerektiğinde düzeltici önlem alır.

• Ürün değerlendirmesi, değerlendirme planlaması, test etme, gösterme, analiz, inceleme ve inceleme yoluyla teslim edilen ürünün kalitesini ve miktarını belirler. Sistem mühendisliği, tüm proje geliştirme süreçlerinin yanı sıra, çözüm alanını tanımlamak için bir mekanizma da sağlar - yani, sistemler ve dış sistemler ile arayüzler. Çözüm alanı, sistem gereksinimleri donanım ve yazılım alt sistemlerine bölümlenmeden önce ürünü en üst düzeyde tanımlar.

Bu yaklaşım, geliştirme sürecinde mümkün olan en geç kısıtlamaları belirleyen yazılım mühendisliği uygulamasına benzer. Bir proje bir kısıtlamayı tanımlamadan önce süreç içine girdikçe, uygulanan çözüm o kadar esnek olacaktır.

**YAZILIM SİSTEM MÜHENDİSLİĞİ NEDİR?**

Yazılım sistemi mühendisliği terimi, 1980'lerin başlarından itibaren başlar ve Winston W. Royce'a (yazılım mühendisliği alanında öncü bir lider) geçer. SwSE, sistemin genel teknik yönetiminden ve nihai sistem ürünlerinin doğrulamasından sorumludur. Sistem mühendisliğinde olduğu gibi, SwSE de belgeler değil belgeler üretiyor. Bu, bilgisayar programları ve kullanım kılavuzları üreten yazılım mühendisliğinden (SWE) farklıdır.

SwSE, sistem gereksinimleri donanım ve yazılım alt sistemlerine ayrıldıktan sonra başlar. SwSE, tüm proje yazılım geliştirme için temel teşkil eder. SwE gibi, hem teknik hem de yönetim sürecidir. SwSE teknik süreci, operasyonel bir ihtiyacı dönüştürmek için gerekli analitik çabadır.

• bir yazılım sistemi açıklaması;

• uygun boyutta, konfigürasyonda ve kalitede bir yazılım tasarımı;

• gereksinimler ve tasarımda yazılım sistemi dokümantasyonu özellikleri;

• bitmiş yazılım ürününü doğrulamak, test etmek ve kabul etmek için gerekli prosedürler;

• ve kullanmak, işletmek ve bakımını yapmak için gerekli belgeler.

SwSE bir iş tanımı değildir. Birçok insan ve kuruluşun gerçekleştirdiği bir süreç: sistem mühendisleri, yöneticiler, yazılım mühendisleri, programcılar ve göz ardı edilmemesi gereken kazanımlar ve kullanıcılar. Büyük sistem çözümleri gittikçe artan bir şekilde yazılıma bağımlı hale geldikçe, yazılım geliştirmeye yönelik bir sistem mühendisliği yaklaşımı yazılım kriziyle ilişkili sorunların önlenmesine yardımcı olabilir. Yazılım geliştiricileri, projelerinde sıklıkla sistem mühendisliği ve SwSE'yi görmezden gelir. Tamamı yazılım olan ya da kullanıma hazır bilgisayarlarda çalışan sistemleri, sistem projeleri değil, yalnızca yazılım projeleri olarak görürler. Yazılım geliştirmenin sistem yönlerini dikkate almamak, uzun süredir devam eden yazılım krizimize katkıda bulunmaktadır.

**SwSE ve YAZILIM MÜHENDİSLİĞİ**

Hem SwSE hem de SwE teknik ve yönetim süreçleridir, ancak SwE yazılım bileşenleri ve destekleyici belgeler üretir.

Özellikle, yazılım mühendisliği

• bilgisayar bilimi, yönetim ve diğer bilimlerin, yazılımın ve ilgili belgelerin analizine, tasarımına, yapımına ve bakımına pratik olarak uygulanması;

• büyük, özel yapım bilgisayar programlarının zaman ve bütçe kısıtlamaları altında başarılı bir şekilde tamamlanmasında analiz, tasarım, kodlama, test, dokümantasyon ve yönetim kavramlarını uygulayan bir mühendislik bilimi; ve

• Etkili ve verimli bir yazılım sistemi için belirtilen bir gereksinimi veya hedefi elde eden yöntem, araç ve tekniklerin sistematik olarak uygulanması. Şekil 2, sistem mühendisliği, SwSE ve SwE arasındaki mühendislik ilişkilerini göstermektedir.

Geleneksel sistem mühendisliği ilk analiz ve tasarımın yanı sıra son sistem entegrasyon ve testini yapar. Yazılım geliştirmenin ilk aşamasında, SwSE, yazılım gereksinimleri analizinden ve mimari tasarımdan sorumludur. SwSE ayrıca yazılım sisteminin son testini yönetir. Son olarak, SwE, sistem mühendislerinin bileşen mühendisliği dedikleri şeyi yönetir.

**SwSE ve PROJE YÖNETİCİSİ**

Proje yönetimi süreci, yazılım sisteminin risklerini ve maliyetlerini değerlendirmeyi, bir program oluşturmayı, çeşitli mühendislik uzmanlıklarını ve tasarım gruplarını entegre etmeyi, yapılandırma kontrolünü sürdürmeyi ve projenin masrafları ve programları karşıladığını ve teknik gereklilikleri karşıladığını ve karşıladığını sağlama çabasını sürekli denetlemeyi içerir. Şekil 3, proje yönetimi, SwSE ve SwE arasındaki yönetim ilişkilerini göstermektedir. Proje yönetimi, proje ve kaynak sağlama yetkisi için genel yönetim sorumluluğuna sahiptir. SwSE, teknik yaklaşımı belirler, teknik kararlar alır, teknik alıcıyla ara yüz kurar ve nihai yazılım ürününü onaylar ve kabul eder. SwE, yazılım tasarımını geliştirmek, tasarımını kodlamak ve yazılım bileşenlerini geliştirmekle sorumludur.

**YAZILIM SİSTEM MÜHENDİSLİĞİNİN FONKSİYONLARI**

Tablo 1, her bir SwSE fonksiyonunun kısa bir genel açıklamasının yanı sıra, SwSE ile ilişkili sistem mühendisliğinin beş ana fonksiyonunu listeler.

***Gereksinimlerin Analizi***

Herhangi bir yazılım geliştirme aktivitesindeki ilk adım, bir sistem gereksinimleri spesifikasyonu (SRS) veya bir yazılım gereksinimleri spesifikasyonu veya her ikisinde de sistem seviyesi şartlarını belirlemek ve belgelendirmektir. Yazılım gereksinimleri, bir kullanıcının bir problemi çözmek veya bir hedefe ulaşmak için ihtiyaç duyduğu yetenekleri ve ayrıca bir sistemin veya bileşenin bir sözleşmeyi, standardı veya diğer resmi olarak uygulanan bir belgeyi yerine getirmek için ihtiyaç duyduğu yetenekleri içerir.

Yazılım gereksinimlerini şu şekilde kategorize edebiliriz:

• İşlevsel gereksinimler, bir sistem veya sistem bileşeninin gerçekleştirmesi gereken işlevleri belirtir.

• Performans gereksinimleri, bir sistem veya sistem bileşeninin hız, doğruluk ve frekans gibi sahip olması gereken performans özelliklerini belirtir.

• Dış arabirim gereksinimleri, bir sistemin veya bileşenin arabirim oluşturması gereken donanım, yazılım veya veritabanı öğelerini belirtir veya bu tür bir arabirimin neden olduğu biçimler, zamanlamalar veya diğer faktörlerle ilgili kısıtlamalar getirir.

• Tasarım kısıtlamaları, örneğin bir dil gereksinimleri, fiziksel donanım gereksinimleri, yazılım geliştirme standartları ve yazılım kalitesi güvence standartları gibi bir yazılım sisteminin veya yazılım sistemi bileşeninin tasarımını etkiler veya sınırlar.

• Kalite özellikleri, yazılımın doğruluğu, güvenilirliği, bakımı yapılabilirliği ve taşınabilirliği gibi kaliteyi etkileyen özelliklere sahip olma derecesini belirtir. Yazılım gereksinimleri analizi, sistem mühendisliği edinen ve kullanıcı sistem gereksinimlerini tanımladıktan sonra başlar. İşlevleri, hepsinin (veya mümkün olduğunca çok sayıda) yazılım sistemi gereksinimlerinin tanımlanmasını içerir ve sonucu, bazen tahsis edilmiş temel olarak adlandırılan mevcut gereksinimler temelini işaretler.

***Yazılım Tasarımı***

Yazılım tasarımı, birlikte yazılım sistemi gereksinimlerini uygulayacak en etkili ve verimli sistem öğelerini seçme ve belgeleme işlemidir. Tasarım, yazılım gereksinimlerini karşılamak için özel ve mantıklı bir yaklaşımı temsil eder.

Yazılım tasarımı geleneksel olarak iki bileşene ayrılır:

• Mimari tasarım, geliştiricinin sistem düzeyinde yapı seçtiği ve yazılım gereksinimlerini yapının bileşenlerine tahsis ettiği sistem tasarımına eşdeğerdir. Mimari tasarım - bazen üst seviye tasarım veya ön tasarım olarak adlandırılır, genellikle bilgisayar programı bileşenlerini ve verilerini tanımlar ve yapılandırır, arayüzleri tanımlar ve zamanlama ve boyutlandırma tahminlerini hazırlar. Genel işleme mimarisi, işlev tahsisleri (ancak ayrıntılı açıklamaları değil), veri akışları, sistem programları, işletim sistemi arayüzleri ve depolama verimi gibi bilgileri içerir.

• Ayrıntılı tasarım, bileşen mühendisliğine eşdeğerdir. Bu durumda bileşenler bağımsız yazılım modülleri ve eserlerdir. Burada önerilen metodoloji, mimari tasarımı SwSE'e, ayrıntılı tasarımı ise SwE'ye tahsis eder. Süreç planlama Planlama, proje hedeflerini ve amaçlarını ve bunlara ulaşmak için stratejileri, politikaları, planları ve prosedürleri belirtir. Ne yapılacağını, nasıl yapılacağını, ne zaman yapılacağını ve kimin yapacağını önceden belirler. Bir SWE projesinin planlanması, alternatif olasılıklardan bir eylem kursu seçilmesine ve bu eylemlerin tamamlanması için bir program tanımlanmasına yol açan SWSE yönetimi faaliyetlerinden oluşur. Proje yönetiminin tüm proje planlamasını gerçekleştirdiğine dair yanlış bir varsayım var. Gerçekte, proje planlamanın iki bileşeni vardır - biri proje yönetimi ve diğeri SwSE tarafından gerçekleştirilir - ve proje planlamasının büyük kısmı bir SwSE işlevidir. (Bu, proje yöneticilerinin her iki işlevi de yerine getiremeyebileceğini söylemek değildir.)

Tablo 2, bir yazılım sistemi projesi için planlama fonksiyonlarının bir bölümlemesini göstermektedir.

***Süreç Kontrolü***

Kontrol, projenin plana uygun olarak yürütülmesini sağlamak için kullanılan yönetim faaliyetlerinin toplamıdır. Süreç kontrolü, performansı ve planları karşı sonuçları ölçer, sapmaları not eder ve planlar ile gerçek sonuçlar arasında uyumu sağlamak için düzeltici önlemler alır.

Süreç kontrolü, projenin ne kadar iyi gittiğine dair bir geri bildirim sistemidir. Süreç kontrolü aşağıdaki gibi sorular sorar: Bütçe ve program dahilinde belirli bir gereksinimi karşılamada gecikmelere neden olacak herhangi bir potansiyel sorun var mı? Herhangi bir risk sorunlara dönüşmüş mü? Tasarım yaklaşımı hala yapılabilir mi?

Kontrol, durumu tekrar plana uygun hale getirmek, planı değiştirmek veya projeyi sonlandırmak için düzeltici faaliyete yol açmalıdır. Proje kontrolü ayrıca iki ayrı bileşene sahiptir: proje yönetiminin başardığını ve yazılım sistemleri mühendisliğinin başardığını kontrol eder. Tablo 3, bir yazılım sistemi projesi için kontrol fonksiyonlarının bir bölümlemesini göstermektedir.

***Doğrulama, Onaylama ve Test Etme***

Doğrulama, doğrulama ve test etme (VV&T) çabası, mühendislik işleminin doğru olup olmadığını ve ürünlerin gereksinimlerine uygun olup olmadığını belirler. Aşağıdaki kritik tanımlar geçerlidir:

• Doğrulama, yazılım geliştirme döngüsünün belirli bir aşamasındaki ürünlerin önceki aşamada belirlenen gereklilikleri karşılayıp karşılamadığını belirler. Doğrulama “Ürünü doğru yapıyorum mu?” Sorusunu yanıtlıyor.

• Onaylama, nihai programın veya yazılımın, kullanıcının ihtiyaç ve gereksinimlerine göre doğruluğunu belirler. Doğrulama, “Doğru ürünü mü inşa ediyorum?” sorusunu yanıtlar.

• Test, hataları bulmak amacıyla hem öngörülen hem de gözlemlenen bilinen giriş ve çıkışlarla bir programın veya kısmi bir programın yürütülmesidir. Testler genellikle onaylamanın bir parçası olarak kabul edilir.

V&V, teknik ve yönetim planlarını, şartnameleri, standartları ve prosedürleri takip ettiklerini belirlemek için sürekli olarak sistem mühendisliği, SwSE, SwE ve proje yönetimi faaliyetlerini takip eden bir süreçtir. V&V ayrıca, SwE projesinin geçici ve nihai ürünlerini değerlendirir. Ara ürünler, gereksinim spesifikasyonlarını, tasarım açıklamalarını, test planlarını ve inceleme sonuçlarını içerir. Nihai ürünler arasında yazılım, kullanım kılavuzları, eğitim kılavuzları vb. Bir yazılım geliştirme projesi içindeki herhangi bir kişi veya işlev V&V yapabilir. SwSE, ihtiyaç özelliklerini, tasarım tanımlarını ve SwSE sürecinin diğer geçici ürünlerini değerlendirmek için V&V tekniklerini ve araçlarını kullanır. Nihai ürünün proje gereksinimleri spesifikasyonlarını karşılayıp karşılamadığını belirlemek için test kullanır. Herhangi bir yazılım geliştirme aktivitesindeki son adım, nihai yazılım ürününü yazılım gereklilikleri spesifikasyonuna göre doğrulamak ve test etmek ve nihai sistem ürününü SRS'ye göre doğrulamak ve test etmektir.

Sistem mühendisliği ve SwSE, öncelikle sistem yaşam döngüsünün önündeki teknik planlama için ve planların projenin sonunda karşılandığını doğrulamak için kullanılan disiplinlerdir.

Ne yazık ki, bir proje, özellikle tamamen yazılımdan oluşuyorsa veya kullanıma hazır ticari bilgisayarlarda çalışıyorsa, genellikle bu disiplinlere bakar. Herhangi bir yazılım projesinin sistem yönlerini dikkate almamak, seçilen donanım üzerinde çalışmayacak veya diğer yazılım sistemleriyle bütünleşmeyecek bir yazılımla sonuçlanabilir.