Bilal Semerci

Bölüm 1

Gereksinim Mühendisliğine Giriş

Yazılım geliştirme sürecinin endüstrileşmesi yolculuğunun erken dönemlerinde, Royce bu sürecin akabinde gelişen bazı gerçeklere değindi.

Gereksinim analizi yeteri kadar iyi yapılmadığında ortaya çıkan dört tür problem vardır: Yukarıdan Aşağıya dizayn imkansızdır, test etmek imkansızdır, , yönetim kontrol dışı gerçekleşir. Bu problem tartışmaları basitleştirmek için başlıklar altında toplanmıştırlar, aslında hepsi tek bir başlık altında buluşabilir - Berbat Yönetim. Yazılım üretiminin kaliteli proje yönetimi belirli ve yol gösteren gereksinimler olmadan var olamaz.

Yukarıdan aşağıya tasarımlar ne kadar yerlerini nesne tabanlı dizaynlara bırakmış olsalarda, bu doğrular geçerliliklerini hala korurlar.

Çok sayıda araştırma, mühendislik gereksinimlerine sistematik çaba harcamanın, daha sonra ihtiyaç duyulan yeniden çalışma miktarını büyük ölçüde azaltabileceğini doğrulamıştır. Sistem mühendisleri, gereksinim mühendisliğini yeterince anlamadıkları ve ya (yazılım ürünü ortaya koyma durumlarında) kod yazmak için acele ettikleri için sıklıkla vazgeçerler. Açıkçası, bu olasılıklar istenmeyen bir durumdur ve bu kitabın amacı, mühendislerin gereksinim mühendisliğinin doğru ilkelerini ve uygulamalarını anlamalarına yardımcı olmaktır.

Gereksinim Mühendisliği Nedir?

Tanımlayıcının bakış açısına bağlı olarak gereksinim mühendisliği disiplinini tasvir etmenin birçok yolu vardır. Örneğin, bir köprü sistemleri karmaşık bir sistemdir, ancak kullanılabilecek nispeten az sayıda tasarım modeline sahiptir (örneğin, asma, kirişli, kablolu). Köprüler sistemleri ayrıca yük gereksinimleri, kullanılan malzemeler ve kullanılan yapım teknikleri açısından özel sözleşmelere ve geçerli düzenlemelere sahiptir. Bu nedenle, bir müşteriyle (örneğin, ulaştırma bakanlığı) bir köprünün gereksinimleri hakkında konuşurken, işlevselliğinin çoğu kısa ve öz bir şekilde ele alınabilir:

Köprü, Chadds Ford, Pensilvanya'daki Creek Yolu'ndaki Brendibadesi Nehri üzerindeki mevcut açıklığın yerini alacak ve çelik konstrüksiyon bir konsol köprüsü olacaktır. Her yönde iki şeritli trafiği destekleyecek ve her yönde saatte minimum 100 araç kapasitesini idare edecektir.

Elbette bu "özellik belirlemede" pek çok bilgi eksiktir (örneğin ağırlık kısıtlamaları), ancak bu köprünün ne yapacağını büyük ölçüde açıklar ve bu gereksinimleri karşılamak için mevcut tasarım seçenekleri nispeten basittir.

Son derece uzmanlaşmış etki alanı diline sahip biyomekanik veya nanoteknoloji sistemleri gibi diğer sistem türleri, görünüşte egzotik gereksinimlere ve kısıtlamalara

sahiptir. Yine de diğer karmaşık sistemler, somutlaştırılması gereken pek çok davranış türüne sahiptir (kelime işlemci yazılımı bile binlerce işlevi destekleyebilir), söz konusu sistemlerin belirtimi gerçekten çok zorlayıcı hale gelir.

Yazar öncelikle bir yazılım mühendisi olduğundan, gereksinim mühendisliği için Pamela Zave'den kaynaklanan duruma uygun ve az çok evrensel bir tanım için şu disipline ulaşıyoruz:

Gereksinim mühendisliği, yazılım sistemlerinin gerçek dünya hedefleri, işlevleri ve kısıtlamaları ile ilgilenen yazılım mühendisliği dalıdır. Ayrıca, bu faktörlerin yazılım davranışının kesin özellikleriyle ve bunların zaman içinde ve yazılım aileleri arasındaki evrimiyle ilişkisiyle de ilgilenir. (Zave 1997)

Ancak gereksinim mühendisliği kavramını, yalnızca yazılım, yalnızca donanım veya donanım ve yazılım (ve birçok karmaşık sistem, donanım ve yazılımın bir birleşimidir) herhangi bir sistemi içerecek şekilde genelleştirmek istiyoruz, bu nedenle Zave'nin tanımını aşağıdaki gibi yeniden yazıyoruz:

Gereksinim mühendisliği, sistemlerin gerçek dünya hedefleri, işlevleri ve kısıtlamaları ile ilgilenen **mühendislik** dalıdır. Ayrıca, bu faktörlerin **sistem** davranışının kesin özellikleriyle ve bunların zaman içinde ve ilgili **sistem** aileleri arasındaki evrimiyle ilişkisiyle de ilgilenir.

Zave'nin tanımında yaptığımız değişiklikler kalın harflerle yazılmıştır. Bu metin boyunca “gereksinim mühendisliği”nden bahsettiğimizde bu değiştirilmiş tanıma atıfta bulunuyoruz. Ve ilerledikçe bu tanımın tüm sonuçlarını ve içerdiği faaliyetleri çok detaylı bir şekilde keşfedeceğiz.

Muhtemelen Yeterli Gereksinim Mühendisliği Yapmıyorsunuz

Araştırmalar, gereksinim mühendisliğinin endüstride iyi yapılmadığını gösteriyor. Örneğin, 3.000'den fazla uygulamalı mühendisten (250'den fazla katılımcı) oluşan küresel bir ankette, %37'si şirketlerindeki gereksinim mühendisliği uygulamalarının tatmin edici olmadığını söyledi (Kassab ve ark. 2014). Avrupa'daki yedi gömülü sistem mühendisliği şirketinin bir başka çalışmasında Sikora ve ark. (2012) benzer sonuçlar bulmuş, özellikle “mevcut gereksinim mühendisliği yöntemlerinin karmaşık gömülü sistemler için gereksinimlerin ele alınması için yetersiz olduğu” sonucuna varmışlardır.

Yetersiz gereksinim mühendisliğinin olası bir nedeni, Wnuk ve ark. tarafından önerilmektedir (2011), şirketlerin gereksinim mühendisliği uygulamalarını etkinlik kapsamı ve gereksinim yapıtlarının yapısı açısından ölçeklendirmekte zorlandıklarını gözlemlemiştir. Anketlerinde, Sikora ve ark. (2012) endüstriyel gereksinimler mühendisliğindeki bariz yetersizlik için başka bir olası neden buldu. Uygulayıcıların "genel sistem mimarisine uygun şekilde entegre edilmiş ancak belirtilen işlev veya bileşenin kendisiyle ilgili olarak çözüm içermeyen gereksinim belirtimlerini" istediklerini buldular. Ankete katılanlar, mevcut yöntem desteğinin bu hedefi desteklemede yetersiz olduğunu belirtti. Öyle görünüyor ki, gereksinim mühendisliğinin uygulayıcıların ihtiyaçlarını karşılamada kat etmesi gereken uzun bir yol var.

Gereksinimler Nelerdir?

Gereksinim mühendisliğindeki zorluğun bir kısmı, bir "gereksinimin" gerçekte ne olduğunu anlamakla ilgilidir. Gereksinimler, üst düzey, soyut ifadeler ve peçete arkası eskizlerinden resmi (matematiksel olarak titiz) spesifikasyonlara kadar değişebilir. Bu değişen temsil biçimleri, paydaşların farklı seviyelerde ihtiyaçları olduğu ve dolayısıyla farklı soyutlama temsillerine bağlı olduğu için ortaya çıkar. Paydaşlar ayrıca bu temsilleri yapmak ve okumak için (örneğin bir ticari müşteriye karşı bir tasarım mühendisi) değişen yeteneklere sahiptir ve bu da gereksinimlerde farklı kaliteye yol açar. Paydaşların doğasını ve ihtiyaçlarını ve yeteneklerini bir sonraki bölümde tartışacağız.

Gereksinimler Hedeflere Karşı

Gereksinim mühendisi için temel bir zorluk, müşterilerin gereksinimleri ve hedefleri sıklıkla karıştırdığını (ve bazen mühendisler de karıştırdığını) kabul etmektir.

Hedefler, bir işletmenin, organizasyonun veya sistemin üst düzey hedefleridir, ancak bir gereklilik, önerilen bir sistem tarafından bir hedefe nasıl ulaşılacağını belirtir. Dolayısıyla, ulaştırma bakanlığının hedefi “dünyanın en güvenli köprüsünü inşa etmek” olabilir. Ancak asıl amaçlanan, köprü malzemeleri, müteahhit ve mühendislerin nitelikleri ve güvenli bir köprüyü sağlayacak yapım teknikleri ile ilgili performans gerekliliklerini şart koşmaktır.

Bir hedefi bir gereklilik olarak ele almak, soruna davetiye çıkarmaktır, çünkü hedefe ulaşıldığını kanıtlamak zor olacaktır. Ayrıca, paydaşlar fikirlerini değiştirdikçe ve hedefleri davranışsal gereksinimlere dönüştürdükçe ve işledikçe hedefler de gelişir.

Gereksinim Düzey Sınıflandırması

Gereksinim türlerindeki çeşitlilikle başa çıkmak için Sommerville (2005), bunları üç soyutlama düzeyinde düzenlemeyi önerir:

- Kullanıcı gereksinimleri
    - Sistem gereksinimleri
    - Tasarım özellikleri

Kullanıcı gereksinimleri, resmi olmayan diyagramlarla birlikte doğal dilde yazılmış soyut ifadelerdir. Sistemin sağlaması beklenen hizmetleri (kullanıcı işlevselliği) ve herhangi bir kısıtlamayı belirtirler. Toplanan kullanıcı gereksinimleri genellikle bir "operasyon kavramı" (Conops) belgesi olarak görünür. Birçok durumda kullanıcı hikayeleri, kullanıcı gereksinimleri rolünü oynayabilir.

Sistem gereksinimleri, hizmetlerin ve kısıtlamaların ayrıntılı açıklamalarıdır. Sistem gereksinimlerine bazen işlevsel spesifikasyon veya teknik ek (nadiren kullanılan bir terim) olarak atıfta bulunulur. Bu gereksinimler, kullanıcı gereksinimlerinin analizinden türetilir ve yapılandırılmış ve kesin olmalıdır. Gereksinimler, bir sistem gereksinimleri belirtimi (SRS) belgesinde toplanır. Kullanım senaryoları, birçok durumda sistem gereksinimi rolünü oynayabilir.

Son olarak, tasarım spesifikasyonları, geliştiriciler tarafından uygulama için temel olarak kullanılan analiz ve tasarım belgelerinden ortaya çıkar. Sistem tasarımı

spesifikasyonu, esas olarak, sistem gereksinimleri spesifikasyonunun analizinden doğrudan elde edilir.

Bu spesifikasyon seviyelerindeki farklılıkları göstermek için havayolu bagaj taşıma sisteminden aşağıdakileri göz önünde bulundurun:

Bir kullanıcı gereksinimi

- Sistem dakikada 20 torba işleyebilecektir.

Bazı ilgili sistem gereksinimleri

- İşlenen her çanta bir bagaj olayını tetikleyecektir.
- Sistem dakikada 20 bagaj olayını kaldırabilecektir.

Son olarak, ilgili sistem özellikleri

1.2 Sistem, operasyonel modda dakikada 20 bagaj olayını işleyebilmelidir.

1.2.1 Bir dakikalık aralıklarla 20'den fazla bagaj olayı meydana gelirse, sonra sistem…

1.2.2 [daha fazla istisna işleme] …

Bir evcil hayvan mağazası POS sistemi için aşağıdakileri göz önünde bulundurun:

Bir kullanıcı gereksinimi

- Sistem, indirimler, vergiler, geri ödemeler ve indirimler dahil olmak üzere satış toplamlarını doğru bir şekilde hesaplayacaktır; doğru bir makbuz yazdırın; ve envanter sayımlarını buna göre güncelleyin.

Bazı ilgili sistem gereksinimleri

- Her satışa bir satış kimliği atanacaktır.
- Her satışta bir veya daha fazla satış öğesi olabilir.
- Her satışta bir veya daha fazla indirim olabilir.
- Her satışta yalnızca bir makbuz yazdırılabilir.

Son olarak, ilgili yazılım özellikleri

1.2 Sistem, her satış işlemine benzersiz bir satış kimlik numarası atayacaktır.

1.2.1 Her satış kimliğinin kendisiyle ilişkilendirilmiş sıfır veya daha fazla satış öğesi olabilir, ancak her satış öğesi tam olarak bir satış kimliğine atanmalıdır.

Eklerdeki sistem özellikleri ayrıca, keşfetmeniz için seviyeye göre düzenlenmiş çok sayıda özellik içerir.

Farklı spesifikasyon seviyeleri, proje yaşam döngüsü boyunca aşamalı testlere rehberlik eder (Şekil 1.1).

Böylece, ilk keşfedilen kullanıcı gereksinimleri, nihai kabul testi için temel olarak kullanılır. Genellikle kullanıcı gereksinimlerinden sonra geliştirilen sistem

gereksinimleri, kabul testinden önce gelen entegrasyon testi için temel olarak kullanılır. Ve son olarak, sistem gereksinimlerinden türetilen tasarım özellikleri, her bir kod birimi uygulandıkça birim testi için kullanılır.

Şekil 1.1

Gereksinimler Özellikler Türler

Gereksinim belirtimleri için başka bir sınıflandırma, aşağıdaki olasılıklar listesinden gereksinim türüne odaklanır:

- Fonksiyonel gereksinimler (FR'ler)
- İşlevsel olmayan gereksinimler (NFR'ler)
- Alan gereksinimleri

Bunlara daha yakından bakalım.

İşlevsel gereksinimler

İşlevsel gereksinimler (FR'ler), sistemin sağlaması gereken hizmetleri ve sistemin girdilerine nasıl tepki vereceğini tanımlar. Ek olarak, işlevsel gereksinimlerin, sistemin yapmaması gereken belirli davranışları açıkça belirtmesi gerekir (bununla ilgili daha fazla bilgi ileride anlatılacaktır). İşlevsel gereksinimler yüksek düzeyde ve genel olabilir (bu durumda bunlar daha önce açıklanan anlamda kullanıcı gereksinimleridir) veya girdileri, çıktıları, istisnaları vb. ifade ederek ayrıntılı olabilir (bu durumda daha önce açıklanan sistem gereksinimleridir).

Doğal dilden (bizim durumumuzda Türkçe dili), görsel modellerden ve daha katı biçimsel yöntemlerden, işlevsel gereksinimler için birçok temsil biçimi vardır. Bölüm 4'te gereksinimlerin temsilini tartışmak için çok daha fazla zaman harcayacağız.

Bazı işlevsel gereksinimleri göstermek için, bagaj taşıma sistemi için aşağıdaki örneklemeyi göz önünde bulundurun.

2.1 Sistem dakikada 20 torbaya kadar işlemelidir

2.4 Sistem boştayken konveyör bant hareket etmeyecektir.

2.8 Ana güç kesilirse, sistem 5 saniye içinde güvenli bir şekilde kapanacaktır.

2.41 Konveyör bant motoru arızalanırsa, sistem giriş besleme mekanizmasını 3 saniye içinde kapatacaktır.

Evcil hayvan mağazası POS sistemi için aşağıdakiler bazı işlevsel gereksinimler olabilir:

4.1 Operatör “toplam” düğmesine bastığında, mevcut satış kapalı duruma girer.

4.1.1 Bir indirim, kapalı duruma girdiğinde, her satılmayan kalem için toplam, kalem sayısı ile kalemin liste fiyatının çarpımı olarak hesaplanır.

4.1.2 Bir satış kapalı duruma girdiğinde, her bir satış kalemi için bir toplam hesaplanır.

Ek A Bölüm 3'te ve Ek B Bölüm 2'de daha fazla işlevsel gereksinim bulunabilir.

Ahmet Enes Ateşsoy

Nonfunctional Requirements(İşlevi olmayan Gereksinimler)

Yazılım sistemleri hem işlevsel davranışlarıyla (sistemin yaptığı)? (What the system does) (sistem ne yapar) hem de işlevsel olmayan davranışlarıyla (sistemin güvenilirlik, yeniden kullanılabilirlik, sürdürülebilirlik vb. gibi bazı gözlemlenebilir niteliklere göre nasıl davr- andığı.) karakterize edilir.

Yazılım pazarında, işlevsel olarak eşdeğer ürünlerin aynı müşteri için rekabet ettiği işlevsel olmayan gereksinimler (NFR’LER) de rakip ürünleri ayırt etmede önemli. Bununla birlikte, uygulamada, NFR’LER (Nonfunctional Requirement), FR’LERE (Funcitonal Requirement) (Weber) göre hala çok az ilgi görmektedir.

“Bu sorun, temel olarak, NFR’LERİN geliştirme sürecinin erken bir aşamasında tedavi etme seçimini yaparken zorluk yaratan benzersiz doğasından kaynaklanmaktadır”

NFR’LER öznel, göreceli ve dağınık olma eğilimindedirler birden fazla modül arasında gereksinimlerden eşlenir çözüm uzayına etki alanı. Ayrıca NFR’LER sıklıkla etkileşimde bulunmak, girişimde bulunmak anlamında yardımcı olabilecek bir NFR’YE ulaşmak veya başkalarının başarısını engellemek NFR’LER. Örneğin, yazılım güvenliğini artırma girişimi, performans pahasına olabilir (örneğin, gecikmeyi artırarak performansı düşürme).

Bu tür etkileşimler, NFR'LER arasında izlemesi veya tahmin etmesi kolay olmayan kapsamlı bir karşılıklı bağımlılık ve değiş tokuş ağı oluşturur. NFR’LERİN zorlu doğasına rağmen, raporlar sürekli olarak, bunların ihmal edilmesinin feci proje başarısızlıklarına veya en azından önemli gecikmelere ve sonuç olarak nihai maliyette

önemli artışlara yol açabileceğini göstermektedir. Aşağıdaki liste birkaç örnek sunmaktadır:

1=>1992'de Londra Ambulans Servisi (LAS), halktan ve acil servislerden gelen çağrılara yanıt olarak ambulansları sevk eden sistemi otomatikleştirmeyi amaçlayan yeni bir bilgisayar destekli sevk sistemini tanıttı. Bu yeni sistem son derece verimsizdi ve ambulans müdahale süreleri önemli ölçüde arttı. Piyasaya sürüldükten kısa bir süre sonra tamamen başarısız oldu ve LAS önceki manuel sisteme geri döndü. Sistemin başarısızlığı, esas olarak sistemin tasarımında "insan ve organizasyon faktörlerini" dikkate almamaktan kaynaklanıyordu (Finkelstein ve Dowell 1996).

2=>Bir NASA Mars Climate Orbiter uzay aracı, yazılım "birlikte çalışabilirlik" sorunu nedeniyle başarısız oldu. Araç, yolculuğu sırasında rotasından saptı ve planlanandan çok daha düşük bir yörüngeye girdi ve atmosferik sürtünme tarafından yok edildi. Gemiyi yok eden metrik/İngiliz birimleri karışımı, Dünya'daki bir yazılım hatasından kaynaklandı. Dönme hızını kontrol etmesi amaçlanan uzay aracındaki iticiler, iticilerin etkisini 4.45 kat hafife alan bir bilgisayar tarafından kontrol edildi. Bu, bir pound kuvveti ile İngilizce'deki standart kuvvet birimi arasındaki orandır. sistem ve Newton, metrik sistemdeki standart birim. Dünya'daki yazılım pound gücünde çalışıyordu, uzay aracı ise Newton'da rakamlar bekliyordu (Breitman ve diğerleri 1999)

3=>New Jersey Motorlu Araçlar Departmanı'nın lisanslama sistemi, geliştirme süresinden tasarruf sağlamak amacıyla 4. nesil bir programlama dili ile yazılmıştır. Ancak uygulandığında, sistem o kadar yavaştı ki, bir noktada, milyonlarca New Jersey aracı işlenmemiş lisans yenilemeleriyle sokaklarda dolaştı. Proje, "uygun fiyat" ve "zamanlılık" hedeflerini karşılamayı amaçladı, ancak "performans ölçeklenebilirliği" sorunları nedeniyle başarısız oldu (Babcock 1985).

4=>Ulusal Tıp Kütüphanesi MEDLARS II sistemi başlangıçta çok çeşitli gelecekteki yayın sistemlerini desteklemek için birçok soyutlama katmanıyla geliştirilmiştir. Sistemin ilk odak noktası, "taşınabilirlik" ve "gelişe bilirlik" niteliklerini geliştirmeye yönelikti. Sistem, "performans" sorunları nedeniyle iki pahalı donanım yükseltmesinden sonra hurdaya çıkarıldı (Boehm ve In 1996)

NFR'lerin bu bariz önemine ve alaka düzeyine rağmen, neredeyse uygulama tamamlandıktan sonra her zaman doğrulanmaya bırakılır, bu da NFR'lerin gereksinim mühendisliğinden doğrudan ve açık bir şekilde eşlenmediği anlamına gelir. Uygulanması (Matoussi ve Laleau 2008). Bu durum, esas olarak, yazılımı mümkün olduğunca hızlı dağıtmaya yönelik muazzam baskıdan kaynaklanmaktadır. bu basınç Yazılım geliştirme, sürecin çok geç saatlerine kadar tespit edilmeyen, çok eskilere dayanan gereksinim hataları sorununun potansiyel alevlenmesiyle baş başa bırakır. Neto et al. (2000), NFR'lerin ihmal edilmesinden dolayı yazılım geliştirmenin iyi bilinen bazı problemlerini sıralamaktadır:

- Maliyet ve program aşımları,
- Yazılım sistemlerinin durdurulması ve

- Yazılım sistemleri kullanıcılarının memnuniyetsizliği.

Tüm bunlar için, NFR'lerin yazılım/sistem yaşam döngüsünün tüm seviyelerini etkilemesi ve mümkün olan en kısa sürede tanımlanması ve bunların ortaya çıkarılmasının doğru ve eksiksiz olması gerektiğini teyit etmek önemlidir.

NFR'lerle uğraşmanın ilk adımı, NFR teriminin nasıl tanımlanacağı konusunda bir anlaşmaya varmaktır. Literatürde bu tür birçok tanım varken, NFR'yi "sistemin içinde bulunduğu ortamın dayattığı gereksinimler" olarak tanımlıyoruz. İşletmektir." Bu durumda, "çevre", açıkça "işlevsel" olarak tanımlanmayan tüm gereksinimleri kapsayan bir şemsiye terimdir.

Birçok gereksinim kategorisi çevreyi oluşturur. Beş ortak NFR kategorisini tanımlıyoruz: kalite, tasarım, ekonomik, işletim ve politik/kültürel.

Kalite gereksinimleri, NFR dünyasındaki en önemli kategoridir. Kalite, bir işletmenin belirtilen ve ima edilen ihtiyaçları karşılama kabiliyetine dayanan özelliklerin toplamıdır (ISO12601). Sistem kalitesi, nihai ürünün temel ve ayırt edici bir özelliğidir. Tipik kalite gereksinimleri arasında güvenlik, gizlilik, güvenilirlik, kullanılabilirlik ve sürdürülebilirlik gereksinimleri bulunur. Genel olarak, herhangi "beceriksizlik" ile biten yazılım kalitesi veya özelliği, işlevsel olmayan bir gerekliliktir. Bu sözde ??ilitler, yasalar ve yönetmelikler, standartlar, çevresel kısıtlamalar ve başka yerler dahil olmak üzere birçok kaynaktan türemiştir. Yazılım/sistem ürün kalitesi, dahili nitelikleri (tipik olarak ara ürünlerin statik ölçümleri), harici nitelikleri veya kullanım niteliklerini (kullanıcının özelliklerini temsil eden) ölçerek değerlendirilebilir. Belirli bir ortamda ve belirli bir kullanım bağlamında kullanıldığında ürünün kalitesinin görünümü). Şekil 1.2, yazılım/sistem yaşam döngüsünün farklı aşamalarında ürün kalitesinin üç görünümünü sunar. Birçok yaklaşım, yazılım kalitesini, daha sonra alt özelliklere ayrıştırılan yapılandırılmış bir dizi özellik içinde sınıflandırır. Örneğin, Kasab (2009) 87 nitelik için bir kalite taksonomisi sundu.

Tasarım/Uygulama Kısıtlamaları: Kısıtlamalar genellikle müzakereye tabi değildir ve bir kez üzerinde anlaşmaya varıldıktan sonra tasarım değiş tokuşları sırasında sınırsızdır. Kısıtlamalar, sistemin dış davranışını etkilemeyen, ancak teknik, ticari veya sözleşme yükümlülüklerini yerine getirmek için yerine getirilmesi gereken, sistemin tasarımı veya bir sistemin geliştirildiği süreç üzerindeki kısıtlamalar olarak tanımlanır (Leffingwell ve Widrig). 2003). Bir anahtar özelliği.

Şekil 1.2

Kısıtlama, kısıtlamaya uyulmaması durumunda bir tür ceza veya kaybın geçerli olmasıdır. Tasarım/uygulama kısıtlamalarına bir örnek, belirli mimari kalıpları veya belirli programlama dillerini kullanma kısıtlamalarını içerir.

Ekonomik Kısıtlamalar: Bunlar, acil ve/veya uzun vadeli geliştirme maliyetini içeren kısıtlamalardır.

İşletim Kısıtlamaları: Bunlar, fiziksel kısıtlamaları, personel mevcudiyetini, beceri düzeyindeki hususları, bakım için sistem erişilebilirliğini vb. içeren kısıtlamalardır.

Politik/Kültürel Kısıtlamalar: Bunlar, politika ve yasal konuları içeren kısıtlamalardır (örneğin, ürün için hangi kanun ve standartların geçerli olduğu)

Varlıkları her zaman proje bağlamındaki diğer kavramlara/gerekliliklere bağlı olduğundan, NFR'lerin bağımsız gereksinimler olmadığını bilmek önemlidir. Örneğin, NFR'ler işlevsel gereksinimlerle ilişkilendirilebilir: "Yalnızca yetkili kullanıcılar sistemin X işlevine erişebilmelidir" veya proje için gerekli kaynaklarla ilişkilendirilmeli: “Yazılım bakımcıları Oracle veritabanında en az iki yıllık deneyime sahip olmalıdır.

NFR'nin ne olduğu ile yerine getirilmesi için neler gerektirebileceği arasında ayrım yapmak da önemlidir. Örneğin, e-posta mesajlarını okurken güvenli etkileşim gereksinimi NFR olarak sınıflandırılır. Fakat bu ihtiyacı karşılamak kimlik doğrulama, yetkilendirme veya mesaj şifreleme gibi bazı teknik tasarım/mimari kararların uygulanmasını gerektirebilir. Ayrıca, hedefleri NFR'lerle karıştırmak kolaydır. Bir hedefin bir paydaşın genel bir niyeti olduğunu unutmayın,

Örneğin:

Sistemin deneyimli operatörler tarafından kullanımı kolay olmalı,

doğrulanabilir bir NFR, bazı nesnel ölçüleri kullanan bir ifadedir:

Deneyimli operatörler, %0,5'ten fazla olmayan bir hata oranıyla iki saatlik uygulamalı eğitmen liderliğindeki eğitimden sonra aşağıdaki 18 sistem özelliğini kullanabilecektir.

Burada, "deneyimli" bir kullanıcının hangi özelliklerde ustalaşması gerektiği (kısa olması için burada listelenmemiş olsalar da), gerekli olan eğitim türünün tanımı (örneğin, kendi kendine eğitimin aksine, eğitmen tarafından yönetilen) konusunda bir kesinlik vardır. çalışma) ve hiçbir insanın mükemmel olmadığını anlamak için yeterli boşluk.

Ek A ve B, işlevsel olmayan gereksinimlerin birkaç başka örneğini içerir.

Alan gereksinimleri

Etki alanı gereksinimleri, uygulama etki alanından türetilir. Bu tür gereksinimler, yeni işlevsel gereksinimlerden veya mevcut işlevsellik üzerindeki kısıtlamalardan oluşabilir. Gereksinimleri veya belirli hesaplamaların nasıl yapılması gerektiğini belirtebilirler.

Örneğin bagaj taşıma sisteminde çeşitli alan gerçeklikleri gereksinimler yaratır. Endüstri standartları vardır (yeni sistemin diğer havayollarının sistemlerine kıyasla daha düşük performans göstermesini istemeyiz). Mevcut mevcut donanım (örneğin, konveyör sistemleri) tarafından dayatılan kısıtlamalar vardır. Ve kısıtlamalar olabilir bagaj görevlileri sendikası ile yapılan toplu iş sözleşmeleri tarafından zorunlu kılınan performans hakkında.

Evcil hayvan mağazası POS sistemi için alan gereksinimleri, geleneksel mağaza uygulamaları tarafından empoze edilir.

>>Nakit, kredi kartları ve kuponların işlenmesi.

>> Ekran ara yüzü ve makbuz formatı.

>> Evcil hayvan mağazası endüstrisindeki sözleşmeler (ör. sık alıcı teşvikleri, bir alana bir tane bedava.

>>Ürünlerin pounda göre satışı (örneğin at yemi) karşı ürün sayısına göre (örneğin, köpek tasmaları)

Ek A ve B ayrıca işlevsel olmayan gereksinimlerin başka örneklerini de içerir.

Etki Alanı Kelime Bilgisini Anlama

Farklı alanlarda terimlerin kullanımında ince ve derin farklılıklar olabileceğinden, gereksinim mühendisi uygulama alanı kelime dağarcığını tam olarak anladığından emin olmalıdır (veya bu kelime dağarcığına hakim birinin hazır olması gerekir). Takip ettiğiniz gerçek olay bu noktayı göstermektedir. Yazardan bir kez istendi çok büyük, uluslararası bir paket dağıtım şirketine danışmanlık hizmetleri sağlamak. Paket teslimat şirketinin mühendisleriyle birkaç saat iletişim kurduktan sonra, yazarın “kamyon” terimini yanlış kullandığı ortaya çıktı. Yazar, "kamyonun", paketleri doğrudan müşterilerin evlerine teslim edecek tanıdık araca atıfta bulunduğuna inanırken, şirket bu araçlara atıfta bulunmak için "paket araba" terimini kullandı. "Kamyon" terimi, bir dağıtım merkezinden diğerine büyük miktarlarda paket taşıyan herhangi bir uzun yol aracı (genellikle 18 tekerlekli bir kamyon) anlamına geliyordu. yani arada çok büyük fark vardı bir "kamyonda" ve bir "paket vagonunda" taşınan paketlerin hacmi. O zaman, "1.000 kamyondan gelen paketlerin" işlenmesini içeren bir gerekliliğin yazıldığını hayal edin.Gerçekten “1.000 paket araba” anlamına geldiğinde). Açıkça, alan terminolojisi anlayışındaki bu fark, önemli ve potansiyel olarak maliyetli olurdu.

Gereksinimler Mühendislik Faaliyetleri

Gereksinim mühendisi bir dizi faaliyetten sorumludur. Bunlar şunları içerir:

>> Gereksinimlerin ortaya çıkarılması/keşfi

>> Gereksinim analizi ve mutabakatı

>> Gereksinim gösterimi/modelleme

>> Gereksinim doğrulama ve doğrulama

>>Gereksinim yönetimi

Bu faaliyetlerin her birini aşağıdaki bölümlerde kısaca ve sonraki bölümlerde ise daha çok inceleyeceğiz.

Gereksinimler Ortaya Çıkarma/Keşif

Gereksinimlerin ortaya çıkarılması/keşfi, müşterinin neye ihtiyaç duyduğunu ve istediğini ortaya çıkarmayı içerir. Ancak ortaya çıkarma, bir ağaçtan alçakta asılı meyve hasat etmek gibi değildir. Bazı gereksinimler açık olsa da (örneğin, POS sisteminin satış vergisini hesaplaması gerekecek) birçok gereksinimin iyi tanımlanmış yaklaşımlarla müşteriden çıkarılması gerekecektir. Gereksinim mühendisliğinin bu yönü, paydaşların kim olduğunu keşfetmeyi de içerir; örneğin, gizli paydaşlar var mı? Ortaya çıkarma, genellikle gözden kaçan NFR'lerin belirlenmesini de içerir.

Gereksinim Analizi ve Anlaşma

Gereksinim analizi ve gereksinim anlaşması, gereksinimlerle ilgili bir dizi sorunu "ham" biçiminde, yani müşterilerden toplandıktan sonra ele almaya yönelik teknikleri içerir. Ham gereksinimlerle ilgili sorunlar şunları içerir:

>> Her zaman mantıklı değiller.

>> Genellikle birbirleriyle çelişirler (ve her zaman açıkça böyle değildir).

>> Tutarsız olabilirler.

>> Eksik olabilirler.

>> Belirsiz veya sadece yanlış olabilirler.

>> Etkileşim içinde olabilirler ve birbirlerine bağımlı olabilirler.

Daha sonra tartışacağımız ortaya çıkarma tekniklerinin çoğu, bu sorunları önlemeye veya hafifletmeye yöneliktir. Formel yöntemler de bu konuda faydalıdır.

Gereksinim analizi ve anlaşma Bölüm 4'te tartışılmaktadır.

Mustafa Enes Tepe

Gereksinimlerin Gösterimi

Gereksinimlerin gösterimi(ya da modellenmesi), ham gereksinimlerin bir modele (genel olarak doğal dil, matematik ve görselleştirme) dönüştürülür. Uygun gösterimler, aşağıdakilerin iletişimini kolaylaştırır:

Gereksinimleri bir sistem mimarisine ve tasarımına dönüştürmeyi.

Gereksinimlerin gösterimi için resmi olmayan(doğal dil ve diyagramlar), resmi(matematiksel dayanağı olan temsiller), yarı resmi(sağlam bir temele dönüştürülebilir veya anlamsal bir çerçevenin eklenmesiyle biçimsel hale getirilebilir). Genellikle bunların bazı kombinasyonları gereksinim gösteriminde kullanılır ve biz bunu konuşacağız. Bölüm 4 ve 6'da verilmiştir.

Gereksinimleri Doğrulama

Gereksinim doğrulama, özelleştirmenin müşterinin ihtiyaçlarını doğru bir şekilde temsil edip etmediğini belirlemesi sürecidir. Doğrulama "Doğru ürünü mü yapıyorum?" sorusuna cevap verir. Gereksinim doğrulama, çeşitli yarı resmi ve biçimsel yöntemler, metin tabanlı araçlar, görselleştirmeler, denetimler vs. Bölüm 5'te konuşuldu

Gereksinimlerin Yönetimi

Gereksinim mühendisliğinin en çok gözden kaçan yönlerinden biri olan gereksinim yönetimi, zaman içerisinde değişen gereksinimlerin yönetimini içerir. Aynı zamanda, gereksinimlerin uygun bir şekilde bir araya getirilmesi, bunlara yönelik devam etmesi ve değişikliklerin değişiklerle ilgilenen ilgili kişilere iletilmesi yoluyla, izlenebilirliği geliştirmeyi de içerir.

Yazılım ve Sistemler için Mühendislik Gereksinimleri

Yöneticilerin ayrıca, kapsam kayması meydana geldiğinde akıllıca geri adım atma becerisini de kazanmaları gerekir. Değişiklikleri izlemek ve izlenebilirliği sürdürmek amacıyla araçlar kullanmak, gereksinim yönetiminin yükünü önemli ölçüde hafifletebilir. Yazılım araçlarını konuşacağız. Bölüm 8'deki gereksinim mühendisliğine ve bölüm 9'daki gereksinim yönetimine yardımcı olmak için.

Bilgi Organları

Yazılım sistemlerinin gereksinim mühendisliği için üç önemli yapılandırılmış sınıflandırma veya bilgi gövdesi mevcuttur: Yazılım Mühendisliği Bilgi Grubu Versiyon 3.0 (SWEBOK 2014), Lisansüstü Yazılım Mühendisliği Referans

Müfredatı(GSwE 2009) ve Yazılım Mühendisliği Sınav Şartnamesi İlkeleri ve Uygulamaları(P&P). Bu bilgi yapıları yazılım mühendisliği disiplinine odaklanır, fakat yazılım, elektrik, mekanik yada hibrit olsun her türlü sistemin mühendisliğine uygulanabilirler. SWEBOK (Yazılım Mühendisliği Bilgi Grubu Versiyon 3.0), yazılım mühendisliğinin tutarlı bir görünümünü desteklemek ve müfredat gelişimine, sertifikasyon ve lisanslama sınavları için bir temeldir. SWEBOK, yazılım mühendisliği yüksek lisans programından mezun olan herkesin sahip olması gereken temel beceri ve bilgileri tanımlar. GSwE(Lisansüstü Yazılım Mühendisliği Referans Müfredatı) raporu, yazılım mühendisliği ile sistem mühendisliği arasındaki güçlü bağı vurgular ve sistem mühendisliği bilgi alanlarının yazılım mühendisliği müfredatına entegre edilmesi gerektiğini söyler. Yazılım Mühendisliği P&P sınavı, Amerika Birleşik Devletleri Eyaletleri ve yargı bölgeleri tarafından, halkın sağlığını, güvenliğini ve refahını etkileyen yazılım sistemleri üzerinde çalışan kişiler için bir lisans bileşeni olarak kullanılmaktadır. Sınıflandırmaların amacı farklı olduğundan, gerekli bilginin kapsamı da bazı yönlerden farklılık gösterebilir. Örneğin, SWEBOK'ta GSwE'nin başlatılması ve kapsam tanımının net bir kapsamı yoktur. Bu bilgi grupları için bilgi kapsamının karşılaştırılması Tablo 1.1'de gösterilmiştir. Bu metin SWEBOK'taki çoğu konunun makul kapsamını sağlamayı amaçlandırmaktadır; ancak diğer iki bilgi gövdesini büyük ölçüde kapsar.

Sistem mühendisliği ile ilgili yeni çıkan referans müfredat ve bilgi yapıları da vardır. Örneğin Sistem Mühendisliği Bilgi Grubu Kılavuzu(SEBoK 2012), Sistem Mühendisliği Lisansüstü Referans Müfredatı (GRCSE 2012) ve Bilgisayar Makineleri Derneği(ACM), Elektrik ve Elektronik Mühendisleri Enstitüsü (IEEE), Bilgisayar Topluluğunun Yazılım Mühendisliği Lisans Derecesi Programları için Müfredat Yönergeleri (ACM/IEEE). Bunlar ayrıca önemli ölçüde gereksinim mühendisliği süreçleri ve görevlerine odaklanır.

Tablo

Gereksinim Mühendisi

Bir gereksinim mühendisinin sahip olması gereken beceriler nelerdir? Christensen ve Chang, gereksinim mühendisinin organize olması, (yazılım) mühendislik yaşam döngüsü boyunca deneyime sahip olması, ne zaman genel ve ne zaman spesifik olması gerektiğini bilecek olgunluğa sahip olması ve gerektiğinde müşteriye karşı durabilmesi gerektiğini öne sürüyorlar (Christensen ve Chang 1996). Christensen ve Chang ayrıca gereksinim mühendisinin iyi bir yönetici, iyi bir dinleyici, adil, iyi bir müzakereci ve çok disiplinli iletişim ve yönetim becerileri yüksek biri olması gerektiğini söylemektedir. Son olarak, gereksinim mühendisi problem alanını anlamalıdır. Gorla

ve Lam (2004), mühendislerin Myers-Briggs anlamında düşünmeleri, algılamaları ve yargılamaları gerektiğini ima eder. Bu gözlemi şu anlama yorumlayabiliriz gereksinim mühendisleri yapılandırılmış ve mantıklıdır (düşünür), toplanan bilgilere odaklanır ve onu yorumlamaya (algılamaya) çalışmaz ve işleri açık bırakmak (yargılamak) yerine kapanış arar. Son olarak, Ebert (2010), gereksinim mühendislerinin aşağıdaki alanlarda yetkinliğe sahip olması gerektiğini önermektedir:

Gereksinim Mühendisliği

Sistem Mühendisliği

Yazılım ve Sistemler için Gereksinim Mühendisliği

Yönetim

İletişim

Biliş

Sosyal Etkileşim

Ebert ayrıca akademik programların yetersiz olduğunu ve bunların yıllarca uygulama ve çalışma yoluyla kazanılması gerektiğini de belirtiyor. Ancak akademik kurslar, yeni gereksinim mühendislerini doğru yönde başlatabilir.

Gereksinim Mühendisinin Rolleri

Gereksinim mühendisliğinin doğasını anlamanın başka bir yolu da gereksinim mühendisinin rollerine bakmaktır. Aşağıdaki rol modellerini belirledik:

- Yazılım sistemleri mühendisi olarak gereksinim mühendisi
- Konu uzmanı olarak gereksinim mühendisi (SME)
- Mimar olarak gereksinim mühendisi
- İş süreci uzmanı olarak gereksinim mühendisi
- Cahil gereksinim mühendisi

Gereksinim mühendisi için yukarıdan gelen hibrit roller de vardır.

Yazılım veya Sistem Mühendisi olarak Gereksinim Mühendisi

Birçok gereksinim mühendisinin muhtemelen yazılım mühendisi, elektrik mühendisi veya sistem mühendisi olması muhtemeldir. Bu durumda, gereksinim mühendisi, modellerin (örneğin, yazılım tasarımı) sonraki gelişimini olumlu yönde etkileyebilir. Bu durumdaki tehlike, gereksinim mühendisinin gereksinim özelliklerini geliştirmesi gerektiğinde bir tasarım oluşturmaya başlayabilmesidir.

Konu Uzmanı olarak Gereksinim Mühendisi

Çoğu durumda müşteri, gereksinim mühendisinin, ya sorun alanını anlamada ya da müşterilerin kendi istek ve arzularını anlamada uzmanlık için bir konu uzmanı olmasını ister. Bazen gereksinim mühendisi bir konu uzmanı değildir; gereksinim

mühendisliğinde uzmandırlar. Gereksinim mühendisinin bir konu uzmanı olmadığı durumlarda, bir konu uzmanı ile güçlerini birleştirmeyi düşünün.

Metin Oğulcan Koca

Mimar Olarak Gereksinim Mühendisi

Bina inşaatı genellikle yazılım inşaatı için bir metafor olarak kullanılır. Yazarın deneyimlerine göre mimarlar ve peysaj mimarları gereksinim mühendisleriyle benzer rollere sahiptir(ve bu benzerlikte yazılım mühendisleri genelde lisanslı olanlardır). Daniel Berry bu konu hakkında kapsamlı bir şekilde yazmıştır (Berry 1999, 2003). Ek olarak Zachman(1987) sunulmak üzere olandan önemli ölçüde farklı olsa da bilgi sistemleri için bir mimari metafor tanıttı.

Mimari (ev spesifikasyonu şeklinde) ve mimari arasındaki benzerlikler yazılım/sistem özellikleri Tablo 1.2'de özetlenmiştir.

İş Süreçleri Uzmanı Olarak Gereksinim Mühendisi

Gereksinim mühendisliğinin faaliyetleri, bir problemi çözmeyi içerir. Müşterinin bir problemi varsa sistem bunu çözmelidir. Çoğu zaman sorunun çözümü , iş süreçlerindeki değişimleri tavsiye eden gereksinim mühendisini de içerir.

Ev inşaatı	*Yazılım/Sistem İnşaatı*
Mimar müşteriler ile tanışır ve görüşmeler yapar. Notlar alır ve resimler çeker.	Gereksinim mühendisi müşteriler ile buluşur ve görüşmeler yapar. Konuşmaları ve diğer eleme tekniklerini kullanır
Mimar kaba eskizler yapar. (Müşteriye gösterir ve geri dönüşler alır).	Gereksinim mühendisi müşterilere gereksinim modelleri yapar. (Örneğin: Prototipler, SRS taslağı)
Mimar daha fazla eskiz yapar (Örneğin; yükseklikler) ve belki de daha fazlası (Örneğin; Karton, 3B modeller, geçiş animasyonları)	Gereksinim mühendisi gereksinimleri inceler ve daha fazla fotoğraf ve yarı resmi öğeler (Örneğin; UML). Daha fazla prototip kullanır
Mimar ek detaylarla (Kat planları) modeller hazırlar	Gereksinim mühendisi tam SRS geliştirmek için yukarıda belirtilenleri kullanır
Gelecek modeller (Örneğin; İnşaat çizimleri) müteahhitlerin kullanımı içindir	Gelecek modeller (Örneğin; yazılım tasarımı, dökümanlar) geliştiricilerin kullanımı içindir

İş süreci iyileştirmesini yürütmek gereksinim mühendisinin görevi olmasa da genellikle yardımcı olur.

Erdem Olarak Cehalet

Berry (1995) gereksinim analizi sürecine dahil olan problem alanında hem acemilerin hem de uzmanların olmasını önerdi. Gerekçesi ise; "Cahil" insanlar "aptal" sorular soruyor ve uzmanlar yanıtlıyor. Gruptaki en cahil gereksinim mühendisine sahip olmak kötü bir şey değildir, çünkü onu zor sorular sormaya ve geleneksel inançlara meydan okumaya zorlar. Tabii ki, cahil ihtiyaç mühendisi, KOBİ'nin rolüne tamamen karşıdır.

Berry ayrıca gereksinim mühendisliğinde biçimsel yöntemleri kullanmanın bir tür cehalet olduğuna dikkat çekti, çünkü bir matematikçi genellikle bir uygulama alanı hakkında onu modellemeye başlamadan önce cahildir.

Müşterinin Rolü

Gereksinim analizinde müşterinin rolü nedir? Müşterinin rolleri çeşitlidir ve şunları içerir:

- Gereksinim mühendisinin neye ihtiyaç duyduklarını ve ne istediklerini anlamasına yardımcı olmak (ortaya çıkarma ve doğrulama)
- Gereksinim mühendisinin ne istemediğini anlamasını sağlamak (ortaya çıkarma ve doğrulama)
- Gerektiğinde ve mümkün olduğunda alan bilgisi sağlamak
- Gereksinim mühendisinin hata yaptığını farkettiklerinde hızlı ve net bir şekilde onları uyarmak
- Değişikliklerin gerekli olduğunu belirlediklerinde (gerçekten gerekli) gereksinim mühendisini hızlı bir şekilde uyarmak
- Tüm anlaşmalara bağlı kalma

Özellikle müşteri, gereksinim mühendisinin de yardımıyla aşağıdaki dört soruyu yanıtlamaktan sorumludur:

1. İstediğim sistem uygulanabilir mi?

2. Uygulanabilirse, ne kadara mal olacak?

3. Yapımı ne kadar sürecek?

4. Sistemi inşa etmek ve teslim etmek için plan nedir?

Bu konularda gereksinim mühendisi müşterilerin beklentilerini saygıyla yönetmelidir. Müşterilerin ve paydaşların doğasını ve rolünü sonraki bölümde keşfedeceğiz.

Geleneksel Gereksinimler Mühendisliği ile İlgili Sorunlar

Geleneksel gereksinim mühendisliği yaklaşımları bir takım problemlerden muzdariptir, birçoğuna halihazırda değindiğimiz (ve ele alınacak olan diğerleri) bunların çoğu kolayca çözülmez. Bu sorunlar şunları içerir:

- Doğal dil sorunları (Örneğin; belirsizlik)
- Alan anlama
- Karmaşıklıkla başa çıkmak (özellikle zamansal)
- Eksiklik (eksik işlevsellik)
- Aşırı eksiksizlik (altın kaplama)
- Aşırı genişleme

- Tutarsızlık
- Yanlışlık
- Ve daha fazlası

Bu sorunların çözümünü kitap boyunca ve özellikle Bölüm 5'te inceleyeceğiz.

Doğal dil sorunları, doğal (insan) dilinin belirsizliklerinden kaynaklanır. Bu dil sorunları sadece gereksinim mühendisleri için değil bütün herkes için geçerlidir. Avukatlar doğal dil ile yazılmış herhangi bir yasa veya sözleşmede bulunan açıkları bularak, sömürerek veya kapatarak geçimlerini sağlıyorlar.

Alan anlama konusunu zaten ele aldık ve diğerleri gibi gereksinim mühendisinin kurulacak sistemin çalışacağı alanda uzman olabileceğini gözlemledik. Sistem karmaşıklığı, tüm sistem mühendislerinin karşılaştığı yaygın bir sorundur ve bu birazdan tartışılacaktır. Sistem davranışını tam olarak ve eksiklerini bulmak çok zordur ancak en azından sistemdeki belirgin işlevselliği yakalamaya yardımcı olabilecek bazı teknikler vardır.

Karmaşıklık

Gereksinim mühendisliğinde çoğu sistem için ortaya çıkarma ve temsil etmedeki en büyük zorluklardan biri karmaşıklıktır. Karmaşıklık için bir tanım bulmadan, herhangi bir önemli davranışı yakalamanın zorluğu ve karmaşıklığının karmaşıklık kavramını açıkladığını iddia ediyoruz. Bu tür zorluklar en basit insan çabasında bile bulunur.

Örneğin, birisinin uyandığınız andan itibaren sabahınızın ilk 5 dakikasını tanımlamanızı istediğini hayal edin. Kesin olarak yapabilir misiniz? Hayır, yapamazdın. Peki neden? Faaliyetlerinizin alabileceği çok fazla olası farklı yol ve çok fazla belirsizlik var. Atomik bir saatle bile her gün aynı saatte uyandığınızı iddia edemezsiniz (çünkü atom saatleri bile kusurludur). Ama tabii ki haftanın gününe, işten tatile çıkıp çıkmadığınıza, tatil olup olmadığına göre farklı şekillerde uyanıyorsunuz. Bunu açıklamanızda hesaba katmanız gerekir. Ama ya hasta uyanırsan? Olayların sırası nasıl değişir? Kalkarken komodininizdeki su bardağını yanlışlıkla devirirseniz ne olur? Bu, aktivitenin özelliklerini değiştirir mi? Ya da tuvalete giderken köpeğinize takılırsanız? Bu örnekle devam edebilir ve bu alıştırmayı çim biçme veya yiyecek alışverişi gibi diğer basit görevlerle tekrarlayabiliriz. Aslında, sorunu şu noktaya kadar sınırlayana kadar saçmalık, herhangi bir önemsiz hareketi bile tam olarak yakalamayı zor hatta imkansız bulacaksınız.

Şimdi karmaşık bir bilgi veya gömülü işleme sistemi düşünün. Böyle bir sistem muhtemelen insanlarla etkileşime girmek zorunda kalacak. Doğrudan insan etkileşimine bağlı olmayan sistemlerde bile, zamansal davranışın yanı sıra yanı sıra gereksinimlerin ortaya çıkarılmasını ve belirtilmesini çok zorlaştıran beklenmeyen olayların sorunları karmaşıklıktır

Rittel ve Webber (1973), "kötü" olarak adlandırdıkları bir dizi karmaşık problem tanımladılar. Kötü problemlerin 10 özelliği vardır:

- Kötü bir problemin kesin bir formülü yoktur.
- Kötü problemlerin durma noktası yoktur.
- Kötü sorunların çözümleri doğru ya da yanlış değil, iyi ya da kötüdür.
- Kötü bir sorunun çözümünün acil ve nihai bir testi yoktur.
- Kötü bir sorunda her çözüm, "tek seferlik bir işlemdir"; deneme yanılma yoluyla öğrenme imkanı olmadığı için her denemenin önemi büyüktür.

- Kötü problemlerin sayısız (ya da ayrıntılı olarak tanımlanabilen) bir dizi olası çözümü yoktur ve plana dahil edilebilecek iyi tanımlanmış bir izin verilebilir işlemler dizisi de yoktur.
- Her kötü sorun özünde benzersizdir.
- Kötü bir sorunu temsil eden bir tutarsızlığın varlığı çeşitli şekillerde açıklanabilir. Açıklama seçimi, sorunun çözümünün doğasını belirler.
- Planlayıcının (tasarımcının) yanılmaya hakkı yoktur.

Rittel ve Webber, ekonomik, politik ve toplumsal nitelikteki kötü sorunların (örneğin, açlık, uyuşturucu kullanımı ve Orta Doğu'daki çatışmalar) olduğunu kastetmişti; dolayısıyla gereksinim mühendisliği için uygun bir çözüm stratejisi sunmazlar. Bununla birlikte, gereksinim mühendisliğini kötü bir problem bağlamında görmek faydalıdır çünkü görevin neden bu kadar zor olduğunu açıklamaya yardımcı olur - çünkü çoğu durumda gerçek sistemler kötü bir problemin birçok özelliğini bünyesinde barındırır.

Altın Kaplama ve Saçma Gereksinimler

Sistem tasarımı üzerine olağanüstü kitabında Brooks (2010), "gülünç gereksinimlere", yani teklif edildikleri zamanda teknolojinin durumu göz önüne alındığında sağlanması zor olan gereksinimlere karşı uyarıda bulunur. Başka türlü gülünç gereksinim, gerçekleştirilmesi mümkün olsa da, kullanılması pek mümkün olmayan bir gereksinim olacaktır.

Brooks, Atlantik üzerinde uçabilmesi beklenen bir “kendi kendine hareket eden helikopter” olan Comanche örneğini veriyor. Bu özelliğin sık kullanılması amaçlanmamıştı, ancak tasarımı önemli ölçüde karmaşıklaştırdı. Uygulanması zor olmasa bile, kullanılması muhtemel olmayan özelliklerin belirtilmesi, altın kaplama olarak adlandırılır.

Eski Gereksinimler

Eski bir gereksinim, önemli ölçüde değişmiş (sistem geliştirme sırasında) veya söz konusu yeni sistem için artık geçerli olmayan gereksinimdir. İkinci, üçüncü vb. nesil sistemler ve ilgili sistemlerden türetilen sistemler için ürünlerde veya ürün hatlarında gereksinim gereksinimlerinin bazı kısımlarını yeniden kullanmak yaygın bir uygulamadır. Gereksinim özellikleri yeniden kullanıldığında, genellikle eski gereksinimlerin yayılması söz konusudur. Wnuk et al. (2013) modası geçmiş ürünlerle ilgili 219 katılımcıdan anket verilerini bildirdi. Eski gereksinimlerin, yazılım yoğun ürünler ve büyük projeler için önemli bir sorun olduğunu bulmuşlardır. Ankete katılan şirketlerin, eski yazılım gereksinimleriyle başa çıkmak için süreçlere veya otomatik araçlara sahip değildi - ankete katılanların yalnızca %10'u, eski gereksinimlerin belirlenmesini desteklemek için araçlar kullandığını bildirdi.

Şüpheli gereksinimler, eski gereksinimlerle ilgilidir. Şüpheli gereksinimler, kökeni bilinmeyen ve/veya bilinmeyen bir amaca hizmet eden gereksinimlerdir. Bu nedenle, şüpheli bir gereklilik mevcut sistemde geçerli olmayabilir ve bu nedenle eskimiş olabilir.

Eski ve şüpheli gereksinimler agresif bir şekilde belirlenmiş ve gereksinim mühendisliği yaşam döngüsünün tüm aşamalarından geçmiştir. Gereksinimler için kaynaklarına ve ilgili gereksinimlere yönelik izlenebilirlik bağlantılarının sürdürülmesi, eski ve şüpheli gereksinimleri önlemenin önemli bir yoludur. Tarih damgalama gereksinimleri (oluşturulmaları üzerine ve herhangi bir değişiklik için), eski gereksinimlerin önlenmesine

yardımcı olabilir. Eski ve şüpheli gereksinimleri yönetmenin daha emek yoğun bir başka yolu da düzenli gereksinim incelemeleri/incelemeleri yapmaktır. (Bölüm 5'te tartışılmıştır)

Gereksinim dalgalanması, gereksinimler üzerinde anlaşmaya varıldıktan sonra gereksinimlerde meydana gelen değişiklikleri ifade eder. Gereksinim dalgalanması, birim zaman başına değişen gereksinimlerin oranı veya belirli bir zamanda toplam gereksinim sayısı başına değişen gereksinimlerin oranı olarak ölçülebilir. Gereksinimler bir sistemin yaşam döngüsü boyunca değişecektir, ancak bu değişikliklerin dikkatli bir şekilde yönetilmesi gerekir. Çeşitli aşamalardaki projeler için kayıp oranı için geçmiş istatistiklerin tutulması ve bu oranların izlenmesi, çok hızlı değişen gereksinimler gibi sorunların belirlenmesine yardımcı olabilir. Yüksek bir kayıp oranı, modası geçmiş gereksinimler için önde gelen bir gösterge olabilir.

Dört Karanlık Köşe

Geleneksel gereksinim mühendisliği ile ilgili sorunların çoğu "dört karanlık köşeden" kaynaklanmaktadır (Zave ve Jackson 1997). Burada göze çarpan noktaları italik yorumlarla kelimesi kelimesine tekrarlıyoruz.

1. Gereksinim mühendisliğinde kullanılan tüm terminoloji, bir makinenin üretileceği ortamın gerçekliğine dayanmalıdır.

2. İnşa edilecek makineyi (her ne kadar soyut olsa da) tanımlamak gerekli veya arzu edilen bir şey değildir.

Bunun yerine çevre, makine olmadan veya makineye rağmen olacağı ve makine sayesinde olacağını umduğumuz gibi iki şekilde tanımlanır.

*Spesifikasyonlar, sistemin **nasıl** başarılacağı değil, **neyin** başarılacağıdır.*

3. Resmi tanımların eylemlere odaklandığını varsayarsak, hangi eylemlerin çevre tarafından, hangi eylemlerin makine tarafından kontrol edildiğini ve çevrenin hangi eylemlerinin makineyle paylaşıldığını belirlemek esastır.

Her tür eylem, gereksinim mühendisliği ile ilgilidir ve bunlar resmi olarak tanımlanması veya sınırlandırılması gerekebilir. Resmi açıklamalar devletlere odaklanırsa, aynı temel ilkeler biraz farklı bir biçimde uygulanır.

> *Resmi temsil yöntemi, sistemin altında yatan organizasyonu takip etmelidir. Örneğin, duruma dayalı bir sistem, en iyi duruma dayalı bir resmileştirme ile temsil edilir.*

4. Gereksinim mühendisliğinde alan bilgisinin birincil rolü, gereksinimlerin uygulanabilir spesifikasyonlara göre iyileştirilmesini desteklemektir.

Uygun alan bilgisi ile birlikte doğru spesifikasyonlar, gereksinimlerin karşılandığını ima eder.

> *Alan bilgisinin rolünün tanınmaması, doldurulmamış gereksinimlere ve yasaklanmış davranışlara yol açabilir.*

Bu karanlık dönüştürücüleri yönetmek, gereksinim mühendisleri ve proje yöneticileri için önemli bir görevdir.

İzzet Talha Çalışır

Mühendisliğin Gereksinimleri Nelerdir?

Tanımlayıcının bakış açısına bağlı olarak gereksinim mühendisliği disiplinini tasvir etmenin birçok yolu vardır. Örneğin, köprü karmaşık bir sistemdir, ancak kullanılabilecek az sayıda tasarım modeline sahiptir. Asma, kafes, kablolu gibi. Köprüler aynı zamanda yükleme gereksinimlerinde farklı sözleşmelere ve düzenlemelere sahiptir. Kullanılan materyaller ve yapım teknikleri olarak da aynı şekildedir. Müşteriler ile köprünün gereksinimleri hakkında konuşurken kısaca işlevselliğinin çoğunun belirlenip yakalanmasıdır.

Pensilvanya Chadds Ford Creek yolundaki Brandywine nehri boyunca yayılan köprü değiştirilecek, ve çelik bir konsol üstüne kurulacaktır. Her birinde iki şeritli trafiği destekleyecektir. Her bir yönde minimum 100 araçlık kapasitesi olacak.

Görüldüğü üzere burada spesifikasyonla ilgili bir sürü bilgi eksik ama esasen bir köprünün ne yapacağını söyler.

Biyomekanik veya nanoteknolojik sistemler gibi diğer sistem türleri son derece uzmanlaşmış etki alanı dili, görünüşte egzotik gereksinimlere ve kısıtlamalara sahiptir. Yine de diğer karmaşık sistemler, yapılması gereken çok sayıda davranışa sahiptir. somutlaştırılabilir olarak (kelime işlemci yazılımı bile binlerce işlevi destekleyebilir) bahsedilen sistemlerin spesifikasyonunun gerçekten çok zorlayıcı hale geldiğini görebiliyoruz Yazar öncelikle bir yazılım mühendisi olduğu için bu disipline ulaşıyoruz. gereksinim mühendisliği için uygun, aşağı yukarı evrensel bir tanım için bu Pamela Zave'den kaynaklanıyor:

Gereksinim mühendisliği, gerçek dünya hedefleri, işlevleri ve kısıtlamaları ile ilgilenen yazılım mühendisliği dalıdır. Ayrıca, bu faktörlerin yazılım davranışının kesin özellikleriyle ve bunların evrimiyle ilişkisini de zaman ve yazılım aileleri arasındaki ilişkisiyle ilgilenir.

Ancak gereksinim mühendisliği kavramını herhangi bir şeyi içerecek şekilde genelleştirmek istiyoruz. Sistem, yalnızca yazılım, yalnızca donanım veya donanım ve yazılım (ve birçok karmaşık sistem, donanım ve yazılımın bir birleşimidir), bu yüzden yeniden yazıyoruz. Zave'nin tanımı şu şekilde:

Gereksinim mühendisliği, konuyla ilgili mühendislik dalıdır. Sistemlerin gerçek dünya hedefleri, işlevleri ve kısıtlamaları ile ilgilenir. Aynı zamanda, bu faktörlerin sistem davranışının kesin spesifikasyonları ve bunların zaman ve zaman içindeki evrimi ile ilişkisi ile de ilgilenir.

Zave'nin tanımında yaptığımız değişiklikler kalın harflerle yazılmıştır. Buna atıfta bulunarak bu kısımda “gereksinim mühendisliği”nden bahsettiğimizde tanımı değişitirilmiş olarak karşımıza çıkıyor. Ve bu tanımın tüm sonuçlarını ve etkinliklerini büyük ayrıntılarla keşfedeceğiz

Muhtemelen Yeterince Gereksinim Mühendisliği Yapmıyorsun

Araştırmalar, gereksinim mühendisliğinin endüstride iyi yapılmadığını gösteriyor. Örneğin, 3.000'den fazla uygulamalı mühendisin katıldığı küresel bir ankette (250 katılımcı) %37'si kendi alanlarında gereksinim mühendisliği uygulamalarına yanıt verdi. Rakamlar tatmin edici değildi (Kassab ve diğerleri 2014). Başka bir çalışmada Avrupa'daki yedi gömülü sistem mühendisliği şirketi, Sikora ve ark. (2012) benzer sonuçlar bulmuşlar, özellikle "Mevcut gereksinimler mühendislik yöntemleri, karmaşık gereksinimlerin üstesinden gelmek için yetersizdir." sonucuna varmışlardır.

Yetersiz gereksinim mühendisliğinin olası bir nedeni aşağıdaki şekilde olduğu düşünülmektedir:Wnuk et al. (2011) şirketlerin ölçeklendirmede gözlemlenen faaliyet kapsamı ve yapı açısından gereksinim mühendisliği uygulamaları gereksinimlerinde zorluk yaşamışlardır. Onların anketlerinde Sikora(2012) endüstriyel gereksinim mühendisliğindeki bariz yetersizlik için başka bir ihtimali bulmuştur.Uygulayıcıların, "Gereksinim özellikleri uygun şekilde genel sistem mimarisine entegre edilmiş, ancak belirtilen işlev veya bileşenin kendisi olması gerektiğini" arzuladıklarını bulmuşlardır. Ankete katılanlar oluşan bu metodun yeteri kadar bu amacı desteklemediğini not etmişlerdir. Öyle görünüyor ki, gereksinim mühendisliğinin uygulayıcıların ihtiyaçlarını karşılamada kat etmesi gereken uzun bir yol var.

Gereksinimler Nelerdir?

Gereksinim mühendisliğindeki zorluğun bir kısmı, bir "gereksinimin" gerçekte ne olduğunu anlamakla ilgilidir. Gereksinimler üst düzey, soyut ifadeler ve peçete arkası eskizleri resmi (matematiksel olarak) özellikler olarak karşımıza çıkar. Bu değişen temsil biçimleri, paydaşların farklı düzeylerde ihtiyaçları olduğu ve dolayısıyla farklı soyutlamalara bağlı olduğu için ortaya çıkar.Paydaşlar ayrıca bunları yapmak ve okumak için çeşitli yeteneklere sahiptir.(Örneğin, bir ticari müşteriye karşı bir tasarım mühendisi) gereksinimlerdeki kalitedeki farklılıklara liderlik ederler. Sonraki bölünde paydaşların doğasını, ihtiyaçlarını ve yeterliliklerini tartışacağız.

Gereklilikler vs Amaçlar

Gereksinim mühendisi için temel bir zorluk, müşterilerin gereksinimleri ve hedefleri sıklıkla karıştırdığını (ve bazen mühendislerin de karıştırdığını) kabul etmektir. Hedefler, bir işletmenin, organizasyonun veya sistemin üst düzey amaçlarıdır, ancak gereksinim, önerilen bir sistem tarafından bir hedefe nasıl ulaşılacağını belirtir.Bu nedenle, ulaştırma bakanlığının hedefi, "en güvenli olan köprüyü inşa etmek" olabilir. Gerçekten amaçlanan köprü malzemeleri,yüklenicinin nitelikleri,mühendisler ve inşaat tekniklerinin uyum içinde güvenli bir köprü yapması performans gerekliliklerini belirlemektir. Bir hedefi bir gereklilik olarak ele almak, hedefe ulaşıldığı için belaya davetiye çıkarmaktır çünkü hedefi kanıtlamak zor olacaktır. Buna ek olarak, paydaşlar değiştikçe hedefler de gelişir. Zihinleri ve hedefleri davranışsal gereksinimlere göre hassaslaştırın ve operasyonel hale getirin.

Gerekliliklerin Seviye Sınıflaması

Sommerville gereksinim türlerindeki çeşitlilikle başa çıkmak için şunları önerir: Bunları üç soyutlama düzeyinde organize etmek:

Kullanıcı gereksinimleri,Sistem Gereksinimleri,Dizayn Gereksinimleri

Kullanıcı gereksinimleri, doğal dilde yazılmış soyut ifadelere eşlik eden resmi olmayan diyagramlardır. Sistemin hangi servise izin vereceği ve hangisini kısıtlayacağına servisler karar verir. Toplanılan kullanıcı gereksinimleri bazen operasyon konsepti belgesi olarak karşımıza çıkar. Birçok durumda kullanıcı hikayeleri kullanıcı gereksinimi rolü oynar. Sistem gereksinimleri servislerin ve kısıtlamaların detaylı tanımlarıdır. Sistem gereksinimleri bazen fonksiyonel özellik veya teknikal ek olarak refere eder.Bu gereklilikler kullanıcı gerekliliklerin analizi sonucu türemiş ve yapılandırılmış ve kesin olmalıdırlar. Gereksinimler bir sistem gereksinimleri spesifikasyonu (SRS) belgesinde toplanır. Use-caseler birçok olayda sistem gereksinmi rolü oynarlar.Son olarak, tasarım özellikleri, geliştiriciler tarafından uygulanması için temel olarak kullanılan analiz ve tasarım belgelerinden ortaya çıkar. Sistem tasarımı spesifikasyonu, esasen doğrudan sistem gereksinimlerinin analizinden elde edilir.Bu spesifikasyon seviyelerindeki farklılıkları göstermek için aşağıdakileri göz önünde bulundurabiliriz. Havayolu bagaj taşıma sisteminden:

Sistem dakikada 20 çanta taşıyabilmeli-kullanıcı gereksinimi

Her çanta bir bagaj olayını tetiklemeli-Sistem her dakikada 20 bagaj olayınının üstesinden gelmeli-bazın bağlantılı sistem gereksinimleri

Son olarak bazı balı sistem özellikleri-Sistem, dakikada 20 bagaj olayını operasyonel modda işleyebilecektir. Bir dakikalık aralıklarla 20'den fazla bagaj olayı meydana gelirse, o zaman sistem- daha fazla istisna işleme-

Bir evcil hayvan dükkanı pos sistemi için;

Sistem, indirimler, vergiler, geri ödemeler ve indirimler; doğru bir makbuz yazdırın; ve envanter sayımlarını güncelleme buna göre yapar- kullanıcı gereksinimi

Bazı bağlantılı sistem gereksinimleri

Her satış bir satış Id'sine atancaktır

Her satış 1 veya daha fazla item satacaktır

Her satış 1 veya daha fazla indirim içerecektir

Her satış sadece bir fiş içerecektir

Son olarak bağlantılı yazılım özelliği olarak

Sistem her satışta farklı bir ıd numarası atacaktır.

Her satış Idsi sıfır veya daha fazla satılmış itemle alakalı olmalı, ama her satılan item sadece bir ıd ile ilişkilendirilmeli.

Eklerdeki sistem özellikleri ayrıca, keşfetmeniz için seviyeye göre düzenlenmiş çok sayıda spesifikasyon içerir.

Farklı spesifikasyon seviyeleri, proje yaşam döngüsü boyunca aşamalı testlere rehberlik eder.

Böylece ilk keşfedilen kullanıcı gereksinimleri son kabul testinin temelleri olarak kullanılır.

Genellikle kullanıcı gereksinimlerinden sonra geliştirilen sistem gereksinimleri, entegrasyon testi için temel olarak kullanılır. Bu, kabul testinden önce gelir. Ve son olarak, tasarım özellikleri sistem gereksinimlerinden türetilen, her bir kod birimi olarak birim testi için uygulanmaktadır.

Bilal Semerci

Ahmet Enes Ateşsoy

Mustafa Enes Tepe

Metin Oğulcan Koca

İzzet Talha Çalışır