Bölüm 1

Giriş
Gereksinim Mühendisliği

Motivasyon

Royce (1975), yazılım geliştirmeyi sanayileştirmeye yönelik atılımın çok başlarında aşağıdaki gerçeklere dikkat çekti.

> Yeterli gereksinim analizi yapılmadığında ortaya çıkan dört tür sorun vardır: yukarıdan aşağıya tasarım imkansızdır; test imkansızdır; kullanıcı dondu; yönetim kontrol altında değildir. Bu sorunlar, tartışmayı basitleştirmek için çeşitli başlıklar altında toplanmış olsa da, aslında hepsi tek bir temanın varyasyonlarıdır: kötü yönetim. Yazılım tedarikinin iyi proje yönetimi, bir tür açık (onaylanmış) ve yöneten gereksinimler olmadan imkansızdır.

Yukarıdan aşağıya tasarımın yerini büyük ölçüde nesne yönelimli tasarım almış olsa da, bu gerçekler bugün hala geçerlidir.

Çok sayıda araştırma, sistematik çabayı gereksinim mühendisliğine ayırmanın, yazılımın veya yazılım yoğun ürünün daha sonraki yaşamlarında ihtiyaç duyulan yeniden çalışma miktarını büyük ölçüde azaltabileceğini ve sistemin çeşitli niteliklerini maliyet etkin bir şekilde iyileştirebileceğini doğrulamıştır. Sistem mühendisleri, gereksinim mühendisliğinin nasıl düzgün bir şekilde yapılacağını anlamadıklarından ya da (bir yazılım ürünü söz konusu olduğunda) kodlamaya başlamak için bir acele olduğundan, genellikle yeterli gereksinim mühendisliği faaliyetinden 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 karmaşık bir sistemdir, ancak kullanılabilecek nispeten az sayıda tasarım modeline sahiptir (örneğin, asma, kirişli, kablolu). Köprüler 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 "şartnamede" pek çok bilgi eksik (örneğin, ağırlık kısıtlamaları), ancak bu köprünün ne yapacağını büyük ölçüde açıklıyor 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 esas olarak bir yazılım mühendisi olduğundan, gereksinim mühendisliği için Pamela Zave'den kaynaklanan uygun, az çok evrensel bir tanım için bu disipline ulaşıyoruz:

> Gereksinim mühendisliği, yazılım mühendisliğinin aşağıdakilerle ilgili dalıdır.*gerçek dünya hedefleri*yazılım sistemlerinin işlevleri ve kısıtlamaları. Aynı zamanda bu faktörlerin birbirleriyle olan ilişkisi ile de ilgilenmektedir.*kesin özellikler*yazılım davranışının ve onların*zaman içinde ve yazılım aileleri arasında evrim*. (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 mühendisliğinin bir dalıdır.**mühendislik**ile ilgili*gerçek dünya hedefleri*için, işlevleri ve kısıtlamaları**sistemler**. Aynı zamanda bu faktörlerin birbirleriyle olan ilişkisi ile de ilgilenmektedir.*kesin*

> *özellikler*ile ilgili**sistem**davranış ve onlara*zaman içinde ve aileler arasında evrim****ilgili sistemler***.

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 mühendisin (250'den fazla katılımcı) yer aldığı küresel bir ankette, %37'si şirketlerindeki gereksinim mühendisliği uygulamalarının tatmin edici olmadığını söyledi (Kassab ve diğerleri 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 diğerleri 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. Paydaşların ayrıca bu temsilleri yapma ve okuma konusunda farklı yetenekleri vardır (örneğin, bir ticari müşteriye karşı bir tasarım mühendisi), bu da gereksinimlerde farklı kaliteye yol açar. Paydaşların doğasını ve ihtiyaçlarını ve yeteneklerini bir sonraki bölümde tartışacağız.

Gereksinimler ve Hedefler

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 hedefleridir, ancak *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 konusunda 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 gereksinimleri bazen şu şekilde adlandırılır:*fonksiyonel özellik*veya*teknik ek*(nadiren kullanılan bir terim). 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 özellikleri, geliştiriciler tarafından uygulanması için temel olarak kullanılan analiz ve tasarım belgelerinden ortaya çıkar. Sistem tasarım spesifikasyonu, esas olarak, sistem gereksinimleri spesifikasyonunun analizinden doğrudan türetilir.

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, o zaman 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 satışın bir veya daha fazla satış kalemi olabilir.
- Her satışta bir veya daha fazla indirim olabilir.
- Her satışta yalnızca bir makbuz yazdırılabilir.

Son olarak, ilgili yazılım özellikleri

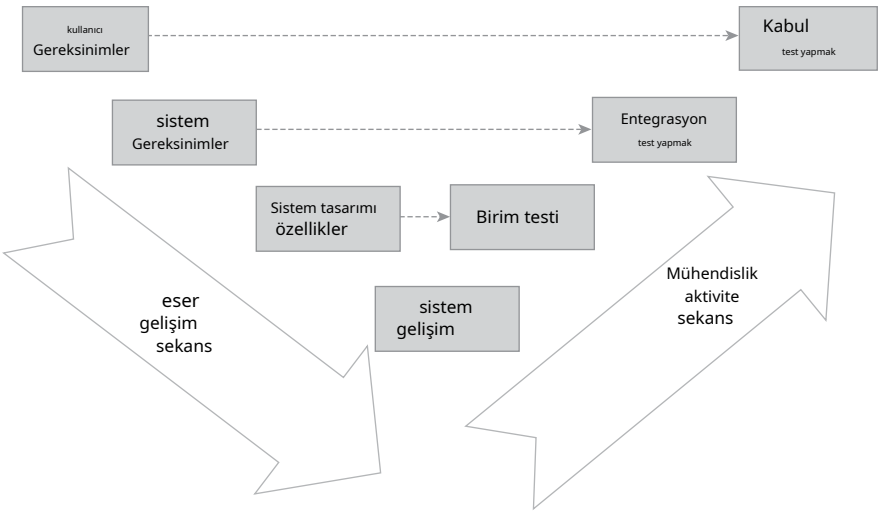
1.2 Sistem, her satış işlemine benzersiz bir satış kimlik numarası atayacaktır.

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 spesifikasyon 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 Gereksinim spesifikasyon seviyeleri ve test arasındaki ilişki.

Gereksinimler Özellikler Türler

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

- İşlevsel 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 ayrıntılı olabilir, girdileri, çıktıları, istisnaları vb. ifade edebilir (bu durumda açıklanan sistem gereksinimleridir). önce).

Doğal dilden (bizim durumumuzda İngiliz 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 düzenli 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 çarpı kalemin liste fiyatı 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.

*İşlevsel Olmayan Gereksinimler**

Yazılım sistemleri hem işlevsel davranışlarıyla (sistemin yaptığı) hem de işlevsel olmayan davranışlarıyla (sistemin güvenilirlik, yeniden kullanılabilirlik, sürdürülebilirlik, vb. gibi bazı gözlemlenebilir niteliklere göre nasıl davrandığı) ile karakterize edilir.

İşlevsel olarak eşdeğer ürünlerin aynı müşteri için rekabet ettiği yazılım pazarında, rakip ürünleri ayırt etmede işlevsel olmayan gereksinimler (NFR'ler) daha önemli hale gelir. Bununla birlikte, uygulamada, NFR'ler, FR'lere (Weber) göre hala çok az ilgi görmektedir. Bu sorun, temel olarak, NFR'lerin, geliştirme sürecinin erken bir aşamasında tedavi etme seçimini yaparken zorluk yaratan benzersiz doğasından kaynaklanmaktadır. NFR'ler özneldir, görecelidir ve gereksinimler alanından çözüm alanına eşleştirildiklerinde çoklu modüller arasında dağılma eğilimindedirler. Ayrıca, NFR'ler, bir NFR'ye ulaşma girişimlerinin diğer NFR'lerin başarılmasına yardımcı olabileceği veya engelleyebileceği anlamında sıklıkla etkileşime girebilir. Örneğin, iyileştirme girişimi

* Penn State'den Dr. Mohamad Kassab bu bölüme katkıda bulunmuştur.

yazılım güvenliği performans pahasına olabilir (örneğin, gecikmeyi artırarak performansı düşürme). Bu tür etkileşimler, NFR'ler arasında, izlenmesi veya tahmin edilmesi kolay olmayan kapsamlı bir karşılıklı bağımlılık ve değiş tokuş ağı oluşturur (Chung ve diğerleri 2000).

NFR'lerin 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:

- 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 sistemi tanıttı. Bu yeni sistem son derece verimsizdi ve ambulans müdahale süreleri önemli ölçüde arttı. Tanıtımından kısa bir süre sonra tamamen başarısız oldu ve LAS önceki manuel sisteme geri döndü. Sistemin başarısızlığı, esas olarak, sistemin tasarımında "insan ve organizasyon faktörlerinin" dikkate alınmamasından kaynaklanmaktadır (Finkelstein ve Dowell 1996).
- Bir NASA Mars Climate Orbiter uzay aracı, bir 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 aracı üzerindeki iticiler, iticilerin etkisini 4.45 kat hafife alan bir bilgisayar tarafından kontrol ediliyordu. Bu, İngiliz sisteminde standart kuvvet birimi olan pound kuvveti ile metrik sistemdeki standart birim olan Newton arasındaki orandır. Dünya'daki yazılım pound gücünde çalışıyordu, uzay aracı ise Newton'da rakamlar bekliyordu (Breitman et al. 1999).
- New Jersey Motorlu Taşıtlar Departmanı'nın lisanslama sistemi, geliştirme süresinden tasarruf sağlamak amacıyla dördüncü nesil bir programlama dilinde 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).
- Ulusal Tıp Kütüphanesi MEDLARS II sistemi başlangıçta çok çeşitli gelecekteki yayın sistemlerini desteklemek için birçok soyutlama katmanıyla geliştirilmiştir. Sistemin ilk odak noktası, "taşınabilirlik" ve "gelişebilirlik" niteliklerini geliştirmeye yönelikti. Sistem, "performans" sorunları nedeniyle iki pahalı donanım yükseltmesinden sonra rafa kaldırıldı (Boehm ve In 1996).

NFR'lerin bu bariz önemine ve alaka düzeyine rağmen, uygulama tamamlandıktan sonra neredeyse her zaman doğrulanmaya bırakılırlar, bu da şu anlama gelir:

NFR'ler, gereksinim mühendisliğinden uygulamaya doğrudan ve açık bir şekilde eşlenmez (Matoussi ve Laleau 2008). Bu durum, esas olarak, yazılımı mümkün olduğunca hızlı dağıtmaya yönelik muazzam baskıdan kaynaklanmaktadır. Bu baskı, yazılım geliştirmeyi, sürecin çok geç saatlerine kadar tespit edilmeyen, çok eskilere dayanan gereksinim hataları sorununun potansiyel olarak alevlenmesiyle karşı karşıya bırakır. Neto et al. (2000), NFR'lerin ihmal edilmesinden dolayı yazılım geliştirmenin iyi bilinen bazı problemlerini sıralamaktadır: (i) maliyet ve zamanlama aşımları, (ii) yazılım sistemlerinin durdurulması ve (iii) 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 çalışacağı ortamın dayattığı gereksinimler" olarak tanımlıyoruz. Bu durumda, "çevre", açıkça "işlevsel" olarak tanımlanmayan tüm gereksinimleri kapsayan bir şemsiye terimdir.

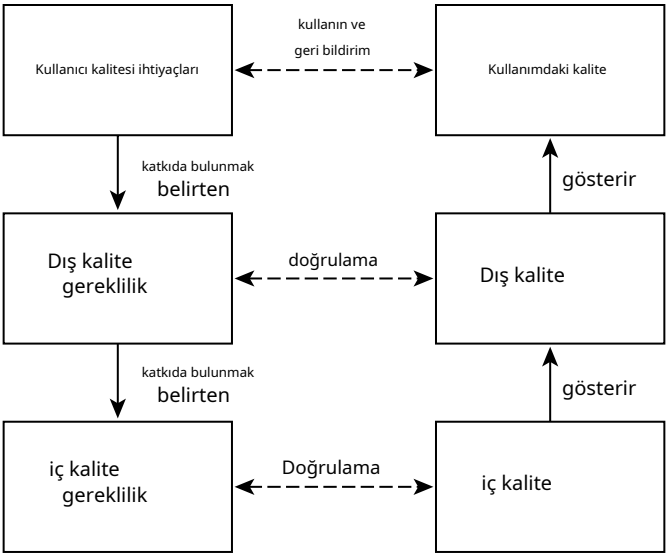
Birçok gereksinim kategorisi çevreyi oluşturur. Beş ortak NFR kategorisini tanımlıyoruz: kalite, tasarım, ekonomik, işletim ve politik/ kültürel.

Kalite gereksinimleriNFR 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, bir "ility" ile biten herhangi bir yazılım kalitesi veya niteliği, işlevsel olmayan bir gerekliliktir. Bu sözde ilişkiler, 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 kullanımdaki kalite niteliklerini (kullanıcının ürünün kalitesine ilişkin görüşünü temsil eden) ölçerek değerlendirilebilir. belirli bir ortam ve belirli bir kullanım bağlamı). Şekil 1.2, yazılım/sistem yaşam döngüsünün farklı aşamalarında ürün kalitesinin üç görünümünü sunar.

Pek çok yaklaşım, yazılım kalitesini, daha sonra alt karakteristiklere ayrıştırılan yapısal bir dizi özellik içinde sınıflandırır. Örneğin, Kassab (2009) 87 nitelik için bir kalite taksonomisi sunmuştur.

Tasarım/Uygulama Kısıtlamaları:Kısıtlamalar genellikle müzakere ve bir kez üzerinde anlaşmaya varıldığında, 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 Yazılım/sistem yaşam döngüsündeki kalite (ISO12601).

kısıtlama, kısıtlamaya uyulmaması durumunda bir tür ceza veya kaybın geçerli olmasıdır. Tasarım/uygulama kısıtlamalarına bir örnek, belirli mimari kalıpların veya belirli programlama dillerinin kullanımına ilişkin kısıtlamaları içerir.

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

Çalışma Kısıtlamaları: Bunlar, fiziksel koşulları içeren kısıtlamalardır. zorlamalar, personel mevcudiyeti, beceri düzeyindeki hususlar, bakım için sistem erişilebilirliği vb.

Siyasi/Kültürel Kısıtlamalar: Bunlar, politika ve yasal konular (örneğin, ürün için hangi yasa 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 gereken kaynaklarla ilişkilendirilebilir: "Yazılım bakımcılarının en az iki yılı olmalıdır. Oracle veritabanında deneyim."

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. Ancak bu gereksinimin karşılanması, kimlik doğrulama, yetkilendirme veya mesaj şifreleme gibi bazı teknik tasarım/mimari kararların alınmasını gerektirebilir. Üstelik, kolay

Hedefleri NFR'lerle karıştırmayın. Bir hedefin bir paydaşın genel bir niyeti olduğunu unutmayın, örneğin:

> *Sistem deneyimli operatörler tarafından kullanımı kolay olmalıdır*,

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 uzmanlaşması gerektiği (kısa olması için burada listelenmemiş olsalar da), gerekli 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 işlevsel gereksinimler üzerindeki kısıtlamalardan oluşabilir veya belirli hesaplamaların nasıl yapılması gerektiğini belirtebilirler.

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

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

- Nakit, kredi kartları ve kuponların işlenmesi
- Ekran arayüzü ve makbuz formatı
- Evcil hayvan mağazası endüstrisindeki gelenekler (örneğin, sık alıcı teşvikleri, bir alana bir bedava)
- Ürünlerin pounda göre satışı (ör. at yemi) ve ürün sayısına göre satışı (ör. köpek tasmaları)

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

Etki Alanı Kelime Bilgisini Anlama

Gereksinim mühendisi, uygulama alanı kelime dağarcığını tam olarak anladığından emin olmalıdır (veya hazırda bu kelimeyi akıcı bir şekilde bilen biri bulunmalıdır).

Farklı alanlarda terimlerin kullanımında ince ve derin farklılıklar olabilir. Aşağıdaki gerçek olay bu noktayı göstermektedir. Yazardan bir keresinde çok büyük, uluslararası bir paket dağıtım şirketine danışmanlık hizmeti vermesi istendi. Paket teslimat şirketinin mühendisleriyle birkaç saat iletişim kurduktan sonra, yazarın "kamyon" terimini yanlış kullandığı ortaya çıktı. Yazar, "kamyon"un 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 miktarda paket taşıyan herhangi bir uzun mesafeli araç (genellikle 18 tekerlekli bir kamyon) anlamına geliyordu. Dolayısıyla, bir "kamyonda" ve bir "paket vagonda" taşınan paketlerin hacmi arasında büyük bir fark vardı. O zaman, "1.000 kamyondan gelen paketlerin" işlenmesini içeren bir şartın 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
- İhtiyaç 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), pek çok 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 toplanmıştır. 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.

Gereksinim Gösterimi

Gereksinim gösterimi (veya modelleme), işlenen ham gereksinimlerin bir modele (genellikle doğal dil, matematik ve görselleştirmeler) dönüştürülmesini içerir. Uygun temsiller, gereksinimlerin iletişimini ve bir sistem mimarisi ve tasarımına dönüştürülmesini kolaylaştırır. Resmi olmayan (örneğin, doğal dil, eskizler ve diyagramlar), resmi (matematiksel olarak sağlam temsiller) ve yarı resmi (sağlam bir temsile dönüştürülebilir veya anlamsal bir çerçevenin eklenmesiyle tamamen resmi hale getirilebilir) dahil olmak üzere gereksinimlerin temsili için çeşitli teknikler kullanılır. . Genellikle gereksinimlerin temsilinde bunların bazı kombinasyonları kullanılır ve bunları Bölüm 4 ve 6'da daha ayrıntılı tartışacağız.

Gereksinim Doğrulama

Gereksinim doğrulama, spesifikasyonun müşterilerin ihtiyaçlarını doğru bir şekilde temsil edip etmediğini belirleme sürecidir. Doğrulama, “Doğru ürünü mü yapıyorum?” sorusuna yanıt verir. Gereksinim doğrulama, çeşitli yarı resmi ve resmi yöntemleri, metne dayalı araçları, görselleştirmeleri, incelemeleri ve benzerlerini içerir ve Bölüm 5'te tartışılmaktadır.

İhtiyaç Yönetimi

Gereksinim mühendisliğinin en çok gözden kaçan yönlerinden biri olan gereksinim yönetimi, zaman içinde değişen gereksinimlerin gerçeklerini yönetmeyi içerir. Aynı zamanda, gereksinimlerin uygun şekilde bir araya getirilmesi ve bunlara tabi kılınması ve gereksinimlerdeki değişikliklerin bilmesi gereken kişilere iletilmesi yoluyla izlenebilirliği geliştirmeyi de içerir.

Yöneticilerin ayrıca, kapsam kayması meydana geldiğinde akıllıca geri adım atma becerilerini de öğrenmeleri gerekir. Değişiklikleri izlemek ve izlenebilirliği sürdürmek için araçlar kullanmak, gereksinim yönetiminin yükünü önemli ölçüde azaltabilir. Bölüm 8'de gereksinim mühendisliğine yardımcı olacak yazılım araçlarını ve Bölüm 9'da genel olarak gereksinim yönetimini tartışacağız.

Bilgi Organları

Yazılım sistemlerinin gereksinim mühendisliği için üç önemli yapılandırılmış sınıflandırma veya bilgi yapısı mevcuttur: Yazılım Mühendisliği Bilgi Grubu Sürüm 3.0 (SWEBOK 2014), Lisansüstü Yazılım Mühendisliği Referans Müfredatı (GSwE 2009) ve Yazılım İlkeleri ve Uygulamaları (P&P) Mühendislik Sınav Şartnamesi (Kilcay-Ergin ve Laplante 2013). Bu bilgi yapıları, yazılım mühendisliği disiplinine odaklanır, ancak yazılım, elektrik, mekanik veya hibrit olsun, her türlü sistemin mühendisliğine uygulanabilirler.

SWEBOK, tutarlı bir yazılım mühendisliği görüşünü desteklemek ve müfredat geliştirme ile sertifika ve lisanslama sınavları için bir temel sağlamak amacıyla kurulmuştur. SWEBOK, yazılım mühendisliği yüksek lisans programından mezun olan tüm kişilerin sahip olması gereken temel bilgi ve becerileri tanımlar.

GSwE raporu, yazılım mühendisliği ve sistem mühendisliği arasındaki güçlü bağı vurgular ve sistem mühendisliği bilgi alanlarının yazılım mühendisliği müfredatına entegre edilmesi gerektiğini önerir (Kilcay-Ergin ve Laplante 2013).

Yazılım Mühendisliği İ&P sınavı, Amerika Birleşik Devletleri eyaletleri ve yargı yetki alanları tarafından, halkın sağlığını, güvenliğini ve refahını etkileyen yazılım sistemleri üzerinde çalışan kişiler için bir lisans bileşeni olarak kullanılmaktadır (Laplante ve diğerleri, 2013).

Taksonomilerin amacı farklı olduğundan, gerekli bilginin kapsamı da bazı yönlerden farklılık gösterir, örneğin, SWEBOK'ta GSwE'nin başlatılması ve kapsam tanımının net bir kapsamı yoktur. Bu bilgi grupları için bilgi kapsamının bir karşılaştırması Tablo 1.1'de gösterilmektedir.

Bu metin, SWEBOK'teki çoğu konunun makul bir kapsamını sağlamayı amaçlamaktadır; ancak diğer iki bilgi gövdesini büyük ölçüde kapsar.

Sistem mühendisliği ile ilgili yeni ortaya çıkan referans müfredatlar 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 Topluluğu'nun *Yazılım Mühendisliği Lisans Programları İçin Müfredat Esasları*(ACM/IEE 2014). Bunlar ayrıca önemli ölçüde gereksinim mühendisliği süreçleri ve faaliyetlerine odaklanır.

Tablo 1.1 GSwE, SWEBOK ve Yazılım Mühendisliği İ&P Bilgi Alanlarının Karşılaştırılması

GSwE 2009	*SWEBOK Sürüm 3.0*	*Yazılım Mühendisliği İ&P*
RE'nin Temelleri • İlişki sistemler arasında mühendislik ve yazılım mühendislik • Tanımı Gereksinimler • Sistem tasarımı kısıtlamalar • Sistem tasarımı ve gereksinimleri paylaştırma • Ürün ve süreç gereksinimleri • İşlevsel ve işlevsiz Gereksinimler • Acil mülkler • Ölçülebilir Gereksinimler	**yazılım gereksinimleri** **temel bilgiler** • Bir yazılımın tanımı gereklilik • Ürün ve süreç gereksinimleri • İşlevsel ve işlevsiz Gereksinimler • Acil mülkler • Ölçülebilir Gereksinimler • Sistem gereksinimler ve yazılım Gereksinimler	**yazılım gereksinimleri** **temel bilgiler** • Kavramı operasyonlar • Çeşitleri Gereksinimler • Ürün ve süreç gereksinimleri • İşlevsel ve işlevsiz Gereksinimler • Ölçülebilir Gereksinimler • Sistem Gereksinimler • Yazılım Gereksinimler • Türetilmiş Gereksinimler • Kısıtlamalar, hizmet seviye
RE Süreci • Süreç modelleri • Süreç aktörleri • Süreç desteği ve yönetimi • Süreç kalitesi ve iyileştirme	**Gereksinimler Süreci** • Süreç modelleri • Süreç aktörleri • Süreç desteği ve yönetimi • Süreç kalitesi ve İyileştirme	
Başlatma ve Kapsam **Tanım** • Belirleme ve müzakere Gereksinimler • Fizibilite analizi • için süreç Gereksinimler gözden geçirme/revizyon	eşdeğeri yok	

(Devam etti)

Tablo 1.1 (Devamı) GSwE, SWEBOK ve Yazılım Mühendisliği İ&P Bilgi Alanlarının Karşılaştırması

GSwE 2009	*SWEBOK Sürüm 3.0*	*Yazılım Mühendisliği İ&P*
Gereksinimlerin Ortaya Çıkarılması • Gereksinimler kaynaklar • Ortaya Çıkarma teknikler	**Gereksinimlerin Ortaya Çıkarılması** • Gereksinimler kaynaklar • Ortaya Çıkarma teknikler	**Gereksinimlerin Ortaya Çıkarılması** • Gereksinimler kaynaklar • Ortaya Çıkarma teknikler • Gereksinimler temsil
Gereksinimlerin analizi • Gereksinimler sınıflandırma • Kavramsal modelleme • Sezgisel yöntemler • Resmi yöntemler • Gereksinimler müzakere	**Gereksinimlerin analizi** • Gereksinimler sınıflandırma • Kavramsal modelleme • Mimari tasarım ve gereksinimler paylaştırma • Gereksinimler müzakere • Resmi analiz	
Gereksinimler **Şartname** • Gereksinimler Şartname teknikler	**Gereksinimler** **Şartname** • Sistem tanım belgesi • Sistem Gereksinimler Şartname • Yazılım Gereksinimler Şartname	**Gereksinimler** **Şartname** • Sistem tanımı belge • Sistem/alt sistemler Şartname • Yazılım Gereksinimler Şartname • Arayüz Gereksinimler Şartname
Gereksinim Doğrulama • Gereksinimler yorumlar • Prototipleme • Model geçerliliği • Kabul testleri	**Gereksinim Doğrulama** • Gereksinimler yorumlar • Prototipleme • Model geçerliliği • Kabul testleri	**Gereksinimler** **Doğrulama ve doğrulama** • Gereksinimler yorumlar • Prototipleme • Model geçerliliği • Simülasyon

(Devam etti)

Tablo 1.1 (Devamı) GSwE, SWEBOK ve Yazılım Mühendisliği İ&P Bilgi Alanlarının Karşılaştırması

GSwE 2009	*SWEBOK Sürüm 3.0*	*Yazılım Mühendisliği İ&P*
Pratik Hususlar • Gereksinimlerin yinelemeli yapısı işlem • Değiştirmek yönetmek • Gereksinimler Öznitellikler • Gereksinimler izleme • Ölçüm Gereksinimler	**Pratik Hususlar** • Gereksinimlerin yinelemeli yapısı işlem • Değiştirmek yönetmek • Gereksinimler Öznitellikler • Gereksinimler izleme • Ölçüm Gereksinimler	**Gereksinimler Yönetmek** • Gereksinimlerin yinelemeli yapısı işlem • Değiştirmek yönetmek • Gereklilik Öznitellikler • Gereksinimler izlenebilirlik • Ölçüm Gereksinimler • Yazılım gereksinim araçları

Kaynak: Kilcay-Ergin, N. ve Laplante, PA'dan uyarlanmıştır, *IEEE Trans. eğitim*, 56: 199-207, 2013.

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 (süreci yönetmek için), iyi bir dinleyici, adil, iyi bir müzakereci ve çok disiplinli (örneğin, iletişim ve yönetimle güçlendirilmiş geleneksel sabit bilimler ve mühendislik geçmişine sahip) olması gerektiğini öne sürüyorlar. Beceriler). Son olarak, gereksinim mühendisi problem alanını anlamalıdır.

Gorla ve Lam (2004), mühendislerin Myers-Briggs anlamında düşünmeleri, algılamaları ve yargılamaları gerektiğini ima eder. Bu gözlemi, gereksinim mühendislerinin yapılandırılmış ve mantıklı olduğu (düşündüğü), toplanan bilgilere odaklandığı ve onu yorumlamaya çalışmadığı (algılamadığı) ve her şeyi açık bırakmak (yargılamak) yerine sonuç arayışında olduğu şeklinde yorumlayabiliriz.

Son olarak, Ebert (2010), gereksinim mühendislerinin aşağıdaki alanlarda yetkinliğe sahip olması gerektiğini önermektedir:

- Gereksinim mühendisliği
- Sistem Mühendisi

- Yönetmek
- İletişim
- Biliş
- Sosyal etkileşim

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

Gereksinim Mühendisi 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 (KOBİ)
- Mimar olarak gereksinim mühendisi
- İş süreci uzmanı olarak gereksinim mühendisi
- Cahil gereksinimler 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. Durum böyle olduğunda, 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, ya sorun alanını anlamaya yardım etmede ya da müşterilerin kendi istek ve arzularını anlamada uzmanlık için gereksinim mühendisinin bir KOBİ olmasını ister. Bazen gereksinim mühendisi bir KOBİ değildir; gereksinim mühendisliğinde uzmandırlar. Gereksinim mühendisinin bir KOBİ olmadığı durumlarda, bir KOBİ ile güçlerini birleştirmeyi düşünün.

Gereksinim Mühendisi Mimar olarak

Bina inşaatı genellikle yazılım inşaatı için bir metafor olarak kullanılır. Yazarın deneyiminde, mimarlar ve peyzaj mimarları gereksinim mühendislerine benzer roller oynarlar (ve bu benzerlik genellikle yazılım mühendislerinin lisanslanması gerektiğine dair bir argüman olarak kullanılır). Daniel Berry bu konu hakkında kapsamlı bir şekilde yazmıştır (Berry 1999, 2003). Berry, benzer faaliyetlerin kapsam kaymasını azalttığını ve müşterileri gereksinim mühendisliği sürecine daha iyi dahil ettiğini kaydetti. Ek olarak, Zachman (1987), modeli sunulmak üzere olandan önemli ölçüde farklı olmasına rağmen, bilgi sistemleri için mimari bir metafor ortaya koydu.

Mimari (ev spesifikasyonu şeklinde) ve yazılım/sistem spesifikasyonu arasındaki benzerlikler Tablo 1.2'de özetlenmiştir. Bir ev inşa etme veya yenileme sürecinden geçtiyseniz, benzerlikleri takdir edeceksiniz.

İş Süreçleri Uzmanı olarak Gereksinim Mühendisi

Gereksinim mühendisliğinin faaliyetleri bir tür problem çözmeyi içerir - müşterinin bir problemi vardır ve sistem onu çözmelidir. Çoğu zaman, eldeki sorunu çözmek, iş süreçlerindeki değişiklikleri tavsiye eden gereksinim mühendisini içerir.

Tablo 1.2 Sistem Mühendisliği için Mimari Model

Ev inşa etme	*Yazılım/sistem oluşturma*
Mimar ile tanışır ve röportajlar müşteriler. Turlar özelliği. Notlar ve resimler alır.	Gereksinimler mühendisi ile buluşuyor müşteriler ve görüşmeler ve diğer ortaya çıkarma tekniklerini kullanır.
Mimar kaba eskizler yapar (müşterilere gösterir, geri bildirim alır).	Gereksinim mühendisi, müşterilere gösterilecek gereksinim modellerini yapar (örneğin, prototipler, SRS taslağı).
Mimar daha fazla eskiz yapar (örn., yükseklikler) ve belki daha karmaşık modeller (örn. karton, 3B sanal modeller, geçiş animasyonları).	Gereksinim mühendisi inceler gereksinimleri ve resmi ve yarı resmi öğeleri ekler (örneğin, UML). Daha fazla prototipleme kullanılır.
Mimar ile modeller hazırlar ek detay (kat planları).	Gereksinim mühendisi kullanır eksiksiz SRS geliştirmek için yukarıda belirtilen bilgiler.
Gelecek modeller (örneğin, inşaat çizimler) yüklenicilerin kullanımı içindir.	Gelecek modeller (örneğin, yazılım tasarımı belgeler) geliştiricilerin kullanımı içindir.

sistem davranışının ifadesini basitleştirmek için. İş süreci iyileştirmesini yürütmek gereksinim mühendisinin rolü olmasa da, bu yan fayda sıklıkla gerçekleştirilir.

Erdem olarak cehalet

Berry (1995), gereksinim mühendisliği sürecine dahil olan problem alanında hem acemilerin hem de uzmanların olmasını önerdi. Onun gerekçesi aşağıdaki gibidir. "Cahil" insanlar "aptal" sorular soruyor ve uzmanlar bu soruları yanıtlıyor. Grubun en cahil kişisi olarak gereksinimler mühendisine sahip olmak, en azından problem alanıyla ilgili olarak, mutlaka kötü bir şey değildir, çünkü onu zor soruları sormaya ve geleneksel inançlara meydan okumaya zorlar. Elbette cahil ihtiyaç mühendisi, KOBİ'nin rolüne tamamen karşıdır.

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

Müşterinin Rolü

Gereksinim mühendisi müşterinin hangi rolü oynamasını beklemeli? Müşterilerin rolleri çeşitlidir ve şunları içerir:

- Gereksinim mühendisinin neye ihtiyaç duyduklarını ve ne istediklerini anlamasına yardımcı olmak (ortaya çıkarma ve doğrulama)
- Gereksinim mühendisinin ne istemediklerini anlamasına yardımcı olmak (ortaya çıkarma ve doğrulama)
- Gerektiğinde ve mümkün olduğunda alan bilgisi sağlamak
- Gereksinim mühendisini, kendisinin veya başkalarının hata yaptığını keşfettiklerinde hızlı ve net bir şekilde uyarmak
- Değişikliklerin gerekli olduğunu belirlediklerinde (gerçekten gerekli) gereksinim mühendisini hızlı bir şekilde uyarmak
- Büyük kapsam kaymasına neden olan "aha anları" yaşama dürtülerini kontrol etme
- Tüm anlaşmalara bağlı kalmak

Müşteri, özellikle, gereksinim mühendisinin de yardımıyla aşağıdaki dört soruyu yanıtlamaktan sorumludur:

1. İstediğim sistem uygulanabilir mi?
2. Eğer öyleyse, ne kadara mal olacak?
3. İnşa etmek ne kadar sürer?
4. Sistemi inşa etmek ve teslim etmek için plan nedir?

Gereksinim mühendisi, müşterilerin bu konularla ilgili beklentilerini yönetmelidir. Bir sonraki bölümde müşterilerin ve paydaşların doğasını ve rolünü keşfedeceğiz.

Geleneksel Gereksinimler Mühendisliği ile İlgili Sorunlar

Geleneksel gereksinim mühendisliği yaklaşımları, birçoğunu halihazırda ele aldığımız (ve ele alınacak olan diğerlerini) ve bunların çoğu kolayca çözülmeyen bir dizi sorundan muzdariptir. Bu sorunlar şunları içerir:

- Doğal dil sorunları (örneğin, belirsizlik, belirsizlik)
- Alan anlama
- Karmaşıklıkla başa çıkmak (özellikle zamansal davranış)
- Zarflama sistemi davranışındaki zorluklar
- Eksiklik (eksik işlevsellik)
- Aşırı eksiksizlik (altın kaplama)
- Aşırı genişleme (tehlikeli "tümü")
- Tutarsızlık
- yanlışlık
- ve dahası

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

Doğal dil sorunları, doğal (insan) dillerin belirsizliklerinden ve bağlam duyarlılığından kaynaklanır. Bu dil sorunlarının sadece gereksinim mühendisleri için değil, herkes için var olduğunu biliyoruz ve avukatlar ve yasa koyucular, doğal dilde yazılmış herhangi bir yasa ve sözleşmede bulunan boşlukları bularak, istismar ederek 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 belirleme ve eksik davranış sorunları da çok zordur, ancak en azından bir sistemdeki en belirgin işlevselliği kaçırmaya yardımcı olabilecek bazı teknikler vardır.

karmaşıklık

Çoğu sistem için gereksinim mühendisliği etkinlikleriyle, özellikle ortaya çıkarma ve temsille uğraşmanın en büyük zorluklarından biri, bunların karmaşık olmalarıdır. Karmaşıklık için bir tanım bulmaya çalışmadan, herhangi bir türden önemsiz davranışı yakalamanın zorluklarının ve karmaşıklığının açıklayıcı olduğunu iddia ediyoruz.

karmaşık kavramı. Bu tür zorluklar, en basit tekrarlanabilir insan çabalarında bile bulunur.

Örneğin, birisinin uyandığınız andan itibaren sabahınızın ilk 5 dakikasını tanımlamanızı istediğini hayal edin. Kesin olarak yapabilir misiniz (deneyin)? Hayır, yapamazdın. Niye ya? 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 gidip gitmemenize, tatil olup olmamasına vb. göre farklı uyanıyorsunuz. Bunu açıklamanızda hesaba katmanız gerekir. Ama ya hasta uyanırsan? Bu, olayların sırasını nasıl değiştirir? 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 gülünç bir noktaya kadar sınırlayana kadar, herhangi bir önemsiz olmayan insan faaliyetini tam olarak yakalamanın zorlayıcı hatta imkansız olduğunu göreceksiniz.

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, gereksinimleri ortaya çıkarmayı ve tanımlamayı bu kadar zorlaştıran (ve diğer tüm gereksinim faaliyetlerini de zorlaştıran), zamansal davranışın karmaşıklığı ve ayrıca beklenmeyen olayların sorunlarıdır.

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

- Kötü bir sorunun kesin bir formülasyonu yoktur.
- Kötü sorunların durma kuralı 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 soruna her çözüm, "tek seferlik bir işlemdir"; deneme yanılma yoluyla öğrenme imkanı olmadığı için her denemenin önemi büyüktür.
- Kötü problemlerin sayısız (ya da ayrıntılı olarak betimlenebilir) bir dizi potansiyel çözümü yoktur ve plana dahil edilebilecek iyi tanımlanmış bir izin verilebilir işlemler dizisi de yoktur.
- Her kötü sorun özünde benzersizdir.
- Her kötü sorun, başka bir sorunun belirtisi olarak kabul edilebilir.
- Kötü bir sorunu temsil eden bir tutarsızlığın varlığı çeşitli şekillerde açıklanabilir. Açıklama seçimi, sorunun çözümünün doğasını belirler.
- Planlayıcının (tasarımcının) yanılma hakkı yoktur.

Rittel ve Webber, ekonomik, politik ve toplumsal nitelikteki kötü sorunların (örneğin, açlık, uyuşturucu kullanımı ve Orta Doğu'daki çatışmalar) olduğu anlamına geliyordu; Öyleyse,

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 seçkin kitabında Brooks (2010), "saçma gereksinimlere", yani teklif edildikleri zamanda teknolojinin durumu göz önüne alındığında sağlanması zor olan gereksinimlere karşı uyarıda bulunur. Başka bir tür gülünç gereksinim, gerçekleştirilmesi mümkün olsa da kullanılması pek mümkün olmayan bir gereksinimdir.

Brooks örneğini veriyor *Comanche, "kendi kendine hareket eden bir helikopter"* kendini Atlantik üzerinde uçabilmesi gerekiyordu. Bu özelliğin sık kullanılması amaçlanmamıştı, ancak tasarımı önemli ölçüde karmaşıklaştırdı. Uygulanması zor olmasa bile, kullanılması muhtemel olmayan özelliklerin belirtilmesine bazen alaycı bir şekilde altın kaplama denir.

Eski Gereksinimler

Bir *eski gereksinim* önemli ölçüde değişmiş (sistem geliştirme sırasında) veya söz konusu yeni sistem için artık geçerli olmayan bir sistemdir. İkinci, üçüncü ve benzeri üretim sistemleri ve ilgili ürünlerden veya ürün hatlarında türetilen sistemler için (bkz. Alıştırma 1.12), gereksinim özelliklerinin bölümlerini yeniden kullanmak yaygın bir uygulamadır. Gereksinim belirtimleri yeniden kullanıldığında, genellikle eski gereksinimlerin yayılması söz konusudur. Wnuk et al. (2013) modası geçmiş gereksinimlerle ilgili 219 katılımcıdan anket verilerini bildirdi. Eski gereksinimlerin, yazılım yoğun ürünler ve büyük projeler için önemli bir sorun olduğunu bulmuşlardır.

Şüpheli gereksinimler, eski gereksinimlerle ilgilidir. Şüpheli gereksinimler, kaynağı 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, gereksinim mühendisliği yaşam döngüsünün tüm aşamalarında agresif bir şekilde tanımlanmalı ve ayıklanmalıdır. Gereksinimler için kaynaklarına ve ilgili gereksinimlere yönelik izlenebilirlik bağlantılarının sürdürülmesi, eski ve şüpheli gereksinimleri önlemenin önemli bir yoludur. Tarih damgalama gereksinimleri (oluşturdukları andan itibaren 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 (5. Bölümde tartışılmıştır).

*Gereksinimler*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 olarak) tanımlamak gerekli veya arzu edilmez.

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

   *Özellikler**ne**sistem tarafından değil, sistem tarafından elde edilecek**nasıl**.*

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

   Tüm eylem türleri gereksinim mühendisliği ile ilgilidir ve resmi olarak tanımlanmaları veya sınırlandırılmaları gerekebilir.
   Resmi tanımlar devletlere odaklanırsa, aynı temel ilkeler biraz farklı bir biçimde uygulanır.

   Resmi temsil yöntemi, sistemin temel organizasyonunu takip etmelidir. Örneğin, devlete dayalı bir sistem, en iyi şekilde devlete dayalı bir formalizasyonla temsil edilir.

4. Gereksinim mühendisliğinde alan bilgisinin birincil rolü, gereksinimlerin uygulanabilir spesifikasyonlara uygun hale getirilmesini 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.

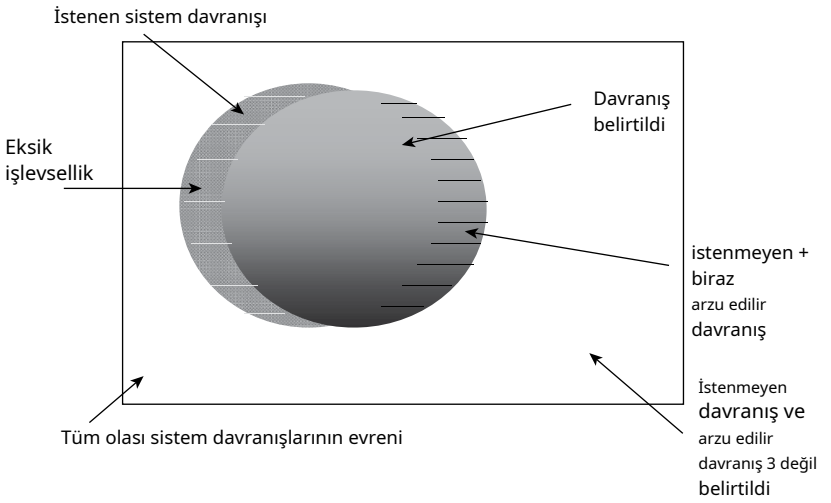
Zarflama Sistemi Davranışındaki Zorluklar

Şekil 1.3'te gösterilen keyfi sistemi şu şekilde hayal edin: *n* girdiler ve *m* çıktılar.

Bu sistemin girdi setinin olduğu bir ortamda çalıştığını hayal edin, Ben, insan operatörlerden, sensörlerden, depolama cihazlarından, diğer sistemlerden ve benzerlerinden türemiştir. Çıkışlar, O, görüntüleme cihazları, aktüatörler, depolama cihazları, diğer sistemler vb. ile ilgilidir. Sisteme getireceğimiz tek kısıtlama, giriş ve çıkışların sistem içinde dijital olarak temsil edilmesidir (eğer bunlar analog cihazlardan veya sistemlerden/sistemlerden ise uygun bir dönüştürme gereklidir). Sistemin bir davranışını giriş/çıkış çifti olarak tanımlarız. Girdiler ve çıktılar ayrık olduğundan, bu sistem sonsuz ama sayılabilir sayıda davranışa sahip olarak düşünülebilir. $B \subseteq i \times Ö$.

Davranış alanını hayal edin, *B*, Şekil 1.3'teki Venn diyagramı ile temsil edilmektedir. Diyagramdaki en soldaki daire, müşterinin anladığı şekliyle istenen davranış setini temsil eder. Bu çemberin dışındaki alan, müşterinin herhangi bir nedenle keşfetmediği istenmeyen davranışlar ve istenen davranışlardır.

Gereksinim mühendisi işine devam eder ve müşteri tarafından istenen davranışı temsil etmesi amaçlanan bir belirtim üretir (Şekil 1.3'ün en sağdaki çemberi). Kusurlu olduğundan, bu belirtim istenen davranışın bazılarını (hepsini değil) ve ayrıca bazı istenmeyen davranışları yakalar. bu Belirtilen davranışta yakalanan, eksik işlevsel yakalanan davranış yasaktır işlevsellik.



Şekil 1.3 Tüm olası davranışların, istenen davranışın ve belirtilen davranışın evreni.

Özetle gereksinim mühendisinin amacı, sol ve sağ dairelerin mümkün olduğunca üst üste gelmesini sağlamak ve başlangıçta belirtim tarafından kapsanmayan (en soldaki daire) eksik istenen davranışları keşfetmektir.

Sistemin istenen davranışlarından dışlanabilecek sonsuz sayıda davranış olduğunu kabul ederek, yine de bunların en kötüsüne odaklanmamız gerekiyor. Bu nedenle, keşfedeceğimiz ve ileteceğimiz bazı gereksinimler şu şekilde olacaktır:

> "Sistem,..."

Bu istenmeyen işleve bazen tehlike veya "davranmaz" veya "yasak davranış" denir. Tehlike terimi genellikle önemli veya yıkıcı başarısızlığa yol açabilecek yasaklanmış davranışı ifade ederken, "yasak davranış" daha az uğursuz bir çağrışıma sahiptir.

Bilinen "yapmamalı" bir sorun değil, peki ya bilinmeyen yasak davranışlar? Bu bilinmeyen yasak davranışlar, bilinen yasak davranışların bilinen risklerle ilişkili olduğu gibi, bilinmeyen riskler veya belirsizliklerle ilişkilidir (Voas ve Laplante 2010). Aşağıdaki trajik skeç, bu iki kavram arasındaki farkın bir örneğidir.

13 Ocak 1982'de karlı bir havada, bir Air Florida Boeing 737 jeti Washington DC'nin Ulusal Havaalanından ayrıldı ve havaalanından bir milden daha yakın bir mesafede 14. Cadde Köprüsü'ne çarptı. Uçakta bulunan 79 kişiden sadece 5'i hayatta kaldı. Uçak ayrıca köprüde yedi araca çarparak dört sürücüyü öldürdü ve dört kişiyi de yaraladı.

Uçaktaki talihsiz yolcular, bu risk ne kadar küçük olursa olsun, uçağın düşme riski olduğunu bilmeliydi. Bazı durumlarda yolcular özel kaza sigortası bile yaptırmış olabilir. Peki ölen ya da yaralanan köprüdeki araçlarda bulunan sürücülerin beklentileri nelerdi? Elbette bir uçağın köprüye çarpması düşüncesi akıllarından hiç geçmedi. Bu nedenle, sürücüler için bu kaza, bilinmeyen ama belirgin bir belirsizliği temsil ediyordu.

Yasak davranışa örnek olarak bagaj taşıma sistemi için aşağıdakileri göz önünde bulundurun:

> "Konveyör sıkışması" sinyali yüksek duruma ayarlandığında, besleme mekanizması konveyör sistemine ilave maddelerin girmesine izin vermeyecektir.

Bu istenmeyen veya "yapmamalı" davranışların nasıl tanımlanacağını ve karakterize edileceğini Bölüm 3'te tartışacağız.

Spesifikasyonlarda "Hepsi" Tehlikesi

Gereksinim belirtimi cümleleri genellikle bazı evrensel nicelemeleri içerir (örneğin, "tüm kullanıcılar...").Ancak "tüm" spesifikasyonların kullanımı

tehlikelidir çünkü genellikle doğru değildirler (Berry ve Kamsties 2000). Örneğin, bagaj taşıma sistemi için bir gereklilik şunları belirtiyorsa: *Tüm bagajlara benzersiz bir tanımlayıcı atanacaktır*.

Bu gerekliliğe meydan okumakta fayda var. Yani, benzersiz bir tanımlayıcı vermek istemediğimiz bir tür "bagaj" var mı? Bir havaalanı check-in acentesine şüpheli bir paket verilirse, bir barkod etiketi yazdırıp paketin üzerine koymaları gerekir mi? Yoksa güvenlik personeli tarafından teftiş gibi daha sonraki işlemler için bir kenara mı koymalılar? Cevabın "evet" veya "hayır" olması önemsizdir - bu tür durumları dikkate almak ve konuyu gereksinim analizi gerçekleşene kadar ertelemek yerine taslak halinde gereksinimleri yazarken ele almak istiyoruz. Bu nedenle, “hepsi” kelimesini içeren gereklilikler sorgulanmalı ve mümkün olduğunda gevşetilmelidir.

"Hepsi" belirtimleriyle ilgili olarak "asla", "her zaman", "hiçbiri" ve "her" (çünkü bunlar resmen evrensel nicelemeyle eşitlenebilecekleri için), bu sözcüklerden ve matematiksel eşdeğerlerinden de kaçınılmalıdır.

VIGNETTE 1.1 Devlet Alımları

Amerika Birleşik Devletleri'nde, bir devlet kurumuna bir sistem sağlayan bir yüklenicinin ihtiyaç duyduğu mühendislik süreci oldukça titizdir. Savunma Bakanlığı veya Enerji Bakanlığı gibi devlet kurumu, genellikle önceden iyi tanımlanmış bir dizi gereksinime dayalı olarak proje teklifleri isteyecektir. Örneğin, yeni bir nükleer enerji üretim tesisi için gereksinimlerin çoğu, sırasıyla büyük ölçüde standartlara, yönetmeliklere ve yasalara dayanan önceki sistemlere dayanacaktır. Bu gereksinimler, sözleşmeyi yapan kurum (ve diğer paydaşlar) tarafından iş sözleşmesi verilmeden önce belirlenir.

Müteahhitler, endüstri günleri aracılığıyla sözleşme teklifi oluşturmaya katılabilir. Bir endüstri günü boyunca, devlet kurumu, sistemin mevcut gereksinimlerini tartışmak için potansiyel müteahhitleri bir araya getirecek. Bu süreç yinelemelidir ve önceden belirlenmiş gereksinimler olgunlaştıkça sistem daha iyi tanımlanmış hale gelir.

Sözleşme verildikten sonra gereksinim mühendisliği süreci devam eder. Kazanan yüklenici, sistem gereksinimlerini iyileştirmek için kitapta açıklanan tekniklerin birçoğunu kullanacaktır. Bu süreçte yüklenici müşteri iletişiminin iyi yapılandırılmış olması esastır. Bu nedenle, gereksinim mühendisliğine yönelik daha resmi olmayan bazı yaklaşımlar yasaklanacaktır (ve genellikle sözleşme,

metodolojiler kullanılabilir veya kullanılamaz). Örneğin, birkaç JAD oturumu düzenlemek (Bölüm 3'te ele alınmıştır) genellikle sözleşmeye dayalı bir gerekliliktir.

Açıklanan süreç, birçok ülkede hükümet sözleşmelerine benzer.

Egzersizler

1.1 Uygun gereksinim mühendisliği faaliyetlerine yönelik başlıca itirazlardan ve caydırıcılardan bazıları nelerdir?

1.2 "Küçük" sistemler için gereksinim mühendisliği nasıl farklıdır?

1.3 Müşterilerin gereksinimleri değiştirmesine neden olabilecek bazı faktörler nelerdir?

1.4 Konu uzmanı olmayan bir gereksinim mühendisi gereksinimleri tanımlamaya yardımcı olması için bir konu uzmanı görevlendirdiğinde hangi sorunlar ortaya çıkabilir?

1.5 için bazı temsili kullanıcı gereksinimlerini, sistem gereksinimlerini ve yazılım özelliklerini listeleyin.

1.5.1 Evcil hayvan mağazası POS sistemi

1.5.2 Bagaj taşıma sistemi

1.6 Aşağıdakiler için beş tipik işlevsel gereksinimi listeleyin:

1.6.1 Evcil hayvan mağazası POS sistemi

1.6.2 Bagaj taşıma sistemi

1.7 Aşağıdakiler için yasaklanmış beş işlevsel gereksinimi listeleyin:

1.7.1 Evcil hayvan mağazası POS sistemi

1.7.2 Bagaj taşıma sistemi

1.8 Aşağıdaki işlevsel olmayan gereksinimleri tanımlayın:

1.8.1 Ek A'daki akıllı ev sistemi özellikleri

1.8.2 Ek B'deki ıslak kuyu pompalama sisteminin özellikleri

1.9 Aşağıdakiler için beş olası işlevsel olmayan gereksinimi listeleyin:

1.9.1 Evcil hayvan mağazası POS sistemi

1.9.2 Bagaj taşıma sistemi

1.10 Akıllı ev sistemleri için geçerli düzenlemeleri veya standartları (NFR) keşfetmek için bazı Web araştırması yapın.

1.11 Akıllı ev sistemi için, Alıştırma 1.9'da keşfettiğiniz düzenlemelere ve sahip olabileceğiniz diğer bilgilere dayanarak bazı tehlikelerin (bu sistemin ne yapmayacağını) bir listesini yapın.

* 1.12 Ürün grubu, belirli özellikleri paylaşan ilgili ürünler kümesidir. Ürün grupları planlanabilir veya ortaya çıkabilir. Ürün hatları için gereksinim mühendisliği zorluklarından bazıları nelerdir? Ritter ve Webber'in kötü sorununun “tek seferlik çözüm” yönü nasıl uygulanır? Bu soruyu cevaplamak için biraz araştırma yapmak isteyebilirsiniz.

Referanslar

ACM/IEEE. (2014). *Lisans Derecesi için Bilgisayar Topluluğu Müfredat Esasları Yazılım Mühendisliği Programları*. https://www.acm.org/education/se2014.pdf (Ocak 2017'de erişildi).

Babcock, C. (1985). Software Jam'deki New Jersey sürücüleri. *Bilgisayar Dünyası*, 30: 1–6. Berry, D. (1995). Gereksinim mühendisliğinde cehaletin önemi. *Dergisi Sistemler ve Yazılım*, 28(1): 179–184.

Berry, DM (1999). Yazılım ve ev gereksinimleri mühendisliği: Öğrenilen dersler mücadele gereksinimleri sürünme. *Gereksinim Mühendisliği Dergisi*, 3(3&4): 242–244.

Berry, DM (2003). İnşaat müteahhitleri ile daha fazla gereksinim mühendisliği macerası. *Gereksinim Mühendisliği Dergisi*, 8(2): 142–146.

Berry, DM ve Kamsties, E. (2000). Spesifikasyonlardaki tehlikeli 'Tümü'. İçinde *Bildiriler Onuncu Uluslararası Yazılım Spesifikasyonu ve Tasarımı Çalıştayı (IWSSD'00)*), San Diego, CA, 5-7 Kasım.

Brooks, FP, Jr. (2010). *Tasarımın tasarımı: Bir bilgisayar bilimcisinden denemeler*. Pearson Eğitim. Addison-Wesley Profesyonel, Boston.

Boehm, B., & In, H. (1996). *Kalite-Gereksinim Çatışmalarının Belirlenmesi*. IEEE Yazılımı, IEEE Computer Society Press, Los Alamitos, CA. s. 25–35.

Breitman, KK, Leite JCSP ve Finkelstein, A. (1999). Dünya sahnesi: Bir anket gerçek hayattan bir vaka çalışması kullanarak gereksinim mühendisliği üzerine. *Brezilya Bilgisayar Topluluğu Dergisi*, 1(6): 13–37.

Christensen, M. ve Chang, C. (1996). İdeal gereksinim mühendisi için plan. *Yazılım*, Mart, s. 12.

Chung, L., Nixon, BA, Yu, E., & Mylopoulos, J. (2000). *İşlevsel Olmayan Gereksinimler Yazılım Mühendisliği*. Kluwer Akademik Yayıncılık. Boston, MA.

Ebert, C. (2010). Gereksinim mühendisliği: Yönetim. P. Laplante'de (Ed.), *Ansiklopedi Yazılım Mühendisliği*, s. 932-948. Taylor ve Francis. Boca Raton, Florida. Finkelstein, A., & Dowell, J. (1996). Bir hatalar komedisi: Londra Ambulans Servisi Vaka Analizi. İçinde *8. Uluslararası Çalıştay Yazılım Spesifikasyonları ve Tasarımı Bildiriler Kitabı*, s. 2–5.

Sistem Mühendisliği Bilgi Grubu (SEBoK) Kılavuzu. (2012). http://www.sebokwiki.org/ (Ocak 2017'de erişildi).

Sistem Mühendisliği (GRCSE) için Lisansüstü Referans Müfredatı. (2012). http://www.bkcase.org/grcse/ (Ocak 2017'de erişildi).

Lisansüstü Yazılım Mühendisliği Referans Müfredatı (GSwE). (2009). https://www.acm.org/binaries/content/assets/education/gsew2009.pf (Ocak 2017'de erişildi).

Gorla, N. ve Lam, YW (2004). Kim kiminle çalışmalı?: Etkili yazılım oluşturma proje ekipleri. *ACM'nin İletişimi*, 47(6): 79-82.

Kasab, M. (2009). *İşlevsel Olmayan Gereksinimler: Modelleme ve Değerlendirme*. VDM Verlag. Saarbrücken, Almanya

Kassab, M., Neill, C., & Laplante, P. (2014). Gereksinim mühendisliğinde uygulama durumu: Çağdaş veriler. *Sistem ve Yazılım Mühendisliğinde Yenilikler*, 10(4): 235–241. Kilcay-Ergin, N., & Laplante, PA (2013). Çevrimiçi bir lisansüstü gereksinim mühendisliği kurs. *Eğitimde IEEE İşlemleri*, 56(2): 199–207.

Laplante, PA, Kalinowski, B. ve Thornton, M. (2013). İlkeler ve uygulamalar sınavı Amerika Birleşik Devletleri'nde yazılım mühendisliği lisansını desteklemek için spesifikasyon. *Yazılım Kalitesi Profesyoneli*, 15(1): 4–15.

Leffingwell, D. ve Widrig, D. (2003). *Yazılım Gereksinimlerini Yönetme: Birleşik Bir Yaklaşım*. Addison-Wesley Nesne Teknolojisi Serisi. Boston, MA.

Matoussi, A. ve Laleau, R. (2008). *Yazılımda İşlevsel Olmayan Gereksinimlere İlişkin Bir Araştırma Gelişme süreci*. Departement d'Informatique Universite, Paris, s. 12. Neto, D.,

Leite, J. ve Cysneiros, L. (2000). Nesne için işlevsel olmayan gereksinimler odaklı modelleme İçinde *Gereksinim Mühendisliği Üzerine Üçüncü Çalıştay*, Rio de Janeiro, Brezilya, s. 109–125.

Rittel, H. ve Webber, M. (1973). Genel bir planlama teorisinde ikilemler. *Politika Bilimleri*, 4: 155–169.

Royce, W. (1975). *Büyük Yazılım Sistemleri Geliştirmek İçin Pratik Stratejiler*. Addison-Wesley, Boston, MA, s. 59.

Sikora, E., Tenbergen, B. ve Pohl, K. (2012). Sanayi ihtiyaçları ve araştırma yönleri gömülü sistemler için gereksinim mühendisliği. *Gereksinim Mühendisliği*, 17: 57-78.

Sommerville, I. (2005). *Yazılım Mühendisliği*. 7. baskı Addison-Wesley, Boston, MA.

Yazılım Mühendisliği Bilgi Gövdesi Sürüm 3.0 (SWEBOK). (2014). http://www. computer.org/portal/web/swebok/html/contentsch2#ch2 (Ocak 2017'de erişildi). Voas, J.

ve Laplante, P. (2010). Etkili bir şekilde tanımlama gereksinimleri olmamalıdır. *BT Uzmanı*, 12(3): 46–53.

Wnuk, K., Gorschek, T., & Zahda, S. (2013). Eski yazılım gereksinimleri. *Bilgi ve Yazılım Teknolojisi*, 55(6): 921–940.

Wnuk, K., Regnell, B. ve Berenbach, B. (2011). Gereksinim mühendisliğini büyütme—Pazar odaklı yazılım geliştirmede artan boyut ve karmaşıklığın zorluklarını keşfetmek. D. Berry & X. French'de (Ed.), *Gereksinim Mühendisliği: Yazılım Kalitesinin Temeli*. REFSQ 2011, LNCS, Cilt. 6606, s. 54–59.

Zachman, JA (1987). Bilgi sistemleri mimarisi için bir çerçeve. *IBM Sistemleri günlük*, 26(3): 276–292.

Zave, P. (1997). Gereksinim mühendisliğinde araştırma çabalarının sınıflandırılması. *ACM Bilgisayar Anketleri*, 29(4): 315–321.

Zave, P. ve Jackson, M. (1997). Gereksinim mühendisliğinin dört karanlık köşesi. *ACM Yazılım Mühendisliği Metodolojisine İlişkin İşlemler*, 6(1): 1–30.

Bölüm 2

Hazırlık için

Gereksinimlerin Ortaya Çıkarılması

Ürün Misyon Beyanı

Yeni bir sistemin geliştirilmesini veya eski bir sistemin yeniden tasarlanmasını üstlenirken yapmamız gereken ilk şey, ne yapması gerektiğine dair kısa bir açıklama elde etmek veya geliştirmektir. Böyle bir ifadeye genellikle ürün misyonu beyanı (veya sistem misyon beyanı) denir. Bazı kuruluşlar, daha uzun olma eğiliminde olsa da, misyon beyanına benzer bir operasyon kavramı (Conops) beyanına atıfta bulunur. Bazı ortamlarda Conops, çok yüksek seviyeli bir gereksinim belirtimine benzeyen bir belgedir.

Ürün misyon beyanı veya Conops, sisteme dahil olan herkes için bir odak noktası görevi görür ve “bu işlevsellik amacına nasıl hizmet ediyor?” Sorusunu sorarak çeşitli özelliklerin önemini tartmamıza izin verir. Çevik metodolojilerde, daha sonra tartışılacağı üzere, misyon ifadesinin veya Conops'un “sistem metaforu” rolünü oynadığını söyleyebiliriz.

Örneğin, Ek A'daki Bölüm 1.1, Smith Akıllı Ev için bir görev bildirimi içerir ve Ek B'deki Bölüm 1, atık su pompalama kontrol sistemi için bir Conops bildirimi içerir.

Misyon beyanları yazmak çekişmeli bir iş olabilir ve birçok insan, küçük ayrıntılarda çıkmaza girme eğilimi olabileceğinden, bunu yapmaktan çekinir veya korkar. Bazen, ürün misyon beyanları çok uzun sürebilir ve aslında bir Conops belgesine dönüşebilir. Bir ürün misyonu ifadesi çok kısa, açıklayıcı, ilgi çekici ve asla ayrıntılı olmamalıdır.