Serhat Obay

Rıfkı Burak Dinç

Pelin Altürk

Tuğçe Özelmacı

TANITIM

Bu bölümde, gereksinimlerin bulunabileceği, keşfedilebileceği, ele geçirilebileceği veya zorlanabileceği birçok yolu keşfedeceğiz. Bu bağlamda, bu terimlerin tümü ortaya çıkarma ile eş anlamlıdır. Ancak "toplantı" eşdeğer değildir. Gereksinimler, basitçe alınıp bir kile konacak düşen meyveler gibi değildir. Gereksinimlere ulaşmak genellikle o kadar kolay değildir, en azından hepsini değil. Daha incelikli ve karmaşık olanlardan birçoğunun, inatçı olmasa da titiz süreçlerle ayıklanması gerekir. Gereksinim tespiti yapmak için seçebileceğiniz birçok teknik vardır ve muhtemelen farklı kullanıcı/paydaş sınıfları için birden fazla ve muhtemelen farklı olanları kullanmanız gerekecektir. Tartışacağımız teknikler şunlardır:

- Beyin fırtınası
- Kart Sıralama
- Çırak olarak tasarımcı
- Domain(alan) analizi
- Etnografik gözlem
- Hedefe dayalı yaklaşımlar
- Grup çalışması
- Röportajlar
- İç gözlem
- Ortak uygulama geliştirme (JAD)
- Merdiven
- Protokol analizi
- Prototipleme
- Kalite işlevi dağıtımı (QFD)
- Anketler
- Repertuar Izgaraları (Şebeke, Sistem)
- Senaryolar
- Görev Analizi
- Kullanım Durumları
- Kullınıcı Hikayeleri
- Bakış açıları
- Atölyeler

This list is adapted from one suggested by Zowghi and Coulin (1998).

Gereksinimlerin Ortaya Çıkarılması için Hazırlanma

Tüm müşterilerin ve paydaşların belirlenmesi, gereksinimlerin ortaya çıkarılmasına hazırlanmanın ilk adımıdır. Ancak paydaş grupları ve özellikle müşteriler homojen olmayabilir ve bu nedenle her bir alt gruba farklı davranmanız gerekir. Örneğin, evcil hayvan mağazası POS sistemi için farklı kullanıcı alt sınıfları şunları içerir:

- Kasiyerler
- Yöneticiler
- Sistem bakım personeli
- Mağaza müşterileri
- Envanter/depo personeli
- Muhasebeciler (vergi bilgilerini girmek için)
- Satış departmanı (fiyat ve indirim bilgilerini girmek için)

Bu kullanıcı alt gruplarının her birinin farklı istekleri vardır ve bunların belirlenmesi gerekir. O zaman, ortaya çıkarmaya hazırlanma süreci şudur:

- Tüm müşterileri ve paydaşları tanımlayın.
- Müşterileri ve diğer paydaşları ilgi alanlarına, kapsam, yetkilendirme veya diğer ayırt edici faktörlere göre sınıflara ayırın (bazı sınıflar birden fazla bölümlendirme düzeyine ihtiyaç duyabilir).
- Her kullanıcı sınıfı ve paydaş grubu için bir şampiyon veya temsili grup seçin.
- Her sınıftan veya paydaş grubundan ilk girdileri almak için uygun teknik(ler)i seçin.

  İşte başka bir kullanıcı sınıfı bölümleme örneği. Bagaj taşıma sistemi dahil olmak üzere birçok farklı sistem vardır.

- Gezginler
- Sistem bakım personeli

- Bagaj görevlileri
- Havayolu zamanlayıcıları/göndericileri
- Havaalanı personeli
- Havaalanı yöneticileri ve politika yapıcılar

Ancak her biri farklı ihtiyaçları olan çeşitli gezginler var. Örneğin, aşağıdaki alt sınıfları göz önünde bulundurun:

- Çocuklar
- Yaşlılar
- İş adamı
- Sıradan gezginler
- Askeri Personel
- Siviller
- Sık Uçanlar

Bu alt sınıfların her birine farklı ortaya çıkarma teknikleri ile yaklaşılması gerekebilir. Örneğin, anketler çocuklar için uygun olmayabilirken, odak grupları askeri personel için daha az yararlı olabilir. Bu alt sınıfların çoğu örtüşür, örneğin bir kişi hem iş seyahatinde hem de sıradan bir gezgin olabilir ve ortaya çıkarma faaliyetlerinden elde edilen veriler analiz edilirken bu çakışmaların dikkate alınması gerekir.

Ortaya Çıkarma Teknikleri Anketi

Şimdi ortaya çıkarma tekniklerini incelemeye başlama zamanı. Bu teknikleri alfabetik sırayla sunuyoruz; herhangi bir tercih ima edilmemektedir. Bölümün sonunda, bu tekniklerin farklı durumlarda yaygınlığını ve uygunluğunu tartışacağız.

Beyin Fırtınası

Beyin fırtınası, sistemler için kapsamlı hedefler oluşturmak için müşteriler ve diğer paydaşlarla yapılan resmi olmayan oturumlardan oluşur. Beyin fırtınası, belirlenmiş bir gündemi, tutanak tutmayı ve resmi yapıların kullanımını (örneğin, Robert'ın Düzen Kuralları) içerecek şekilde resmileştirilebilir. Ancak bir beyin fırtınası toplantısının formalitesi, muhtemelen toplantıda sergilenen yaratıcı seviyeyle ters orantılıdır. Bu tür toplantılar muhtemelen gayri resmi, hatta spontane olmalı ve herhangi bir büyük keşfin bazı kayıtlarını içeren tek yapı ile olmalıdır.

Beyin fırtınası oturumları sırasında bazı ön gereksinimler oluşturulabilir, ancak bu husus sürece ikincildir. JAD(Joint Application Development) tekniği beyin fırtınasını (ve çok daha

fazlasını) içerir ve muhtemelen diğer grup odaklı uyarma tekniklerinin çoğu, dolaylı olarak bir tür beyin fırtınası içerir..

Kart Sıralama

Bu teknik, paydaşların sistem/yazılım ürünü için işlevsellik hakkında önemli bilgileri içeren bir dizi kartı tamamlamasını içerir. Paydaşların/müşterilerin her bir işlevsellik için sıralama ve gerekçe içermesi de iyi bir fikirdir.

Müşterilerin ve paydaşların kartları tamamlamasına izin verecek süre önemli bir karardır. Kart sıralama alıştırması birkaç saat içinde tamamlanabilse de, paydaşların acele etmesi önemli, eksik işlevlere yol açacaktır. Ancak paydaşlara çok fazla zaman vermek, süreci gereksiz yere yavaşlatabilir. Kartların tamamlanması için minimum 1 hafta (en fazla 2 hafta) süre verilmesi tavsiye edilir. Diğer bir alternatif ise, müşterilerin kartları 2 saatlik bir oturumda tamamlamasını ve ardından 1 hafta sonra başka bir kart tamamlama ve inceleme oturumu için geri dönmesini sağlamaktır. Her durumda, her kart oluşturma oturumundan sonra, gereksinim mühendisi bu kartları bir şekilde düzenler, genellikle işlevleri mantıksal olarak kümeler. Sıralanan kartlar, nihai kodda program sınıflarını belirlemek için CRC (yetenek, sorumluluk, işbirliği) kartları geliştirme sürecine girdi olarak da kullanılabilir. Kısaca tartışılacak başka bir teknik olan QFD , bir kart sıralama faaliyetini içerir.

Süreci açıklamak için, Şekil 3.1, müşteri tarafından evcil hayvan mağazası POS sistemi için oluşturulan ve sıralanmamış bir yığın halinde duran küçük bir kart alt kümesini göstermektedir. Bu durumda, her kart, işlevselliğin yalnızca kısa bir açıklamasını içerir ve bir öncelik derecesi dahildir (kısa olması için hiçbir gerekçe gösterilmemiştir).

Gereksinim mühendisi bu kart yığınını analiz eder ve iki kartın "müşteri yönetimi" işlevleriyle, iki kartın "vergi işlevleri" ile ve bir kartın "envanter özellikleri" veya işlevleriyle ilgili olduğuna karar verir ve kartları uygun şekilde düzenler. Şekil 3.2'de gösterildiği gibi yığınlar.

Müşteriye, düzeltme veya eksik özellikler için bu sıralanmış işlevler listesi gösterilebilir. Ardından, gerekirse yeni bir kart turu oluşturulabilir. Süreç, gereksinim mühendisi ve müşteri, sistem özelliklerinin büyük ölçüde yakalandığından emin olana kadar devam eder.

Çırak Olarak Tasarımcı

Çırak olarak tasarımcı, müşterinin ihtiyaçlarını anlamak için müşterinin işi hakkında yeterince bilgi edinmek için gereksinim mühendisinin müşterinin "omzunun üzerinden baktığı" bir

gereksinim keşif tekniğidir. Müşteri ile tasarımcı arasındaki ilişki, usta ile çırak arasındaki ilişki gibidir. Yani çırak, tıpkı gereksinim mühendisinin (tasarımcı) müşterinin yaptığı işi müşteriden öğrenmesini istediğimiz gibi ustadan bir beceri öğrenir. Çırak, ustanın bildiği her şeyi öğrenmek için oradadır (ve bu nedenle müşteriye işin bu kısımları hakkında konuşurken ve bunları gösterirken rehberlik etmelidir). Tasarımcı, belirli ihtiyaçları karşılamak için oradadır.

Not: * Bu tartışma, izin alınarak Laplante'de (2006) bulunan bir tartışmadan uyarlanmıştır.

Müşterinin bu tekniğin çalışması için bir tür öğretme becerisine sahip olması gerekiyor gibi görünebilir, ancak bu doğru değil. Bazı müşteriler işleri hakkında etkili bir şekilde konuşamazlar, ancak ortaya çıktıkça onun hakkında konuşabilirler. Ayrıca, müşterilerin onu sunmanın en iyi yolunu ya da güdülerini bulmak zorunda değiller; sadece ne yaptıklarını açıklıyorlar.

Eseri görmek de neyin önemli olduğunu ortaya koyuyor. Örneğin, insanlar yaptıkları her şeyin ve bazen neden yaptıklarının farkında değildirler. Bazı eylemler yılların deneyiminin sonucudur ve ifade edilemeyecek kadar inceliklidir. Diğer eylemler, geçerli bir gerekçesi olmayan alışkanlıklardır. Bir çırağın mevcudiyeti, ustaya (müşteriye) faaliyetler ve nasıl ortaya çıktıkları hakkında düşünme fırsatı verir.

İşi görmek ayrıntıları ortaya çıkarır, çünkü bir görevi yerine getirmedikçe, onu tarif ederken detaylandırmak zordur. Son olarak, eseri görmek yapıyı ortaya çıkarır. Çalışma kalıpları işçi için her zaman açık değildir. Bir çırak, bir görevin birden fazla örneğini gözlemleyerek ve varyasyonları dahil ederek, kendisinin nasıl yapılacağına dair bir anlayış oluşturarak işin stratejilerini ve tekniklerini öğrenir. Bu tekniğin çalışması için, gereksinim mühendisi aşağıdakiler dahil işin yapısını ve anlamını anlamalıdır:

- İşi halletmek için strateji
- Yola çıkan kısıtlamalar
- Çalışmayı desteklediği için fiziksel çevrenin yapısı
- İş bölünür
- Tekrarlayan aktivite kalıpları
- Bunların herhangi bir potansiyel sistem üzerindeki etkileri

Tasarımcı, herhangi bir yanlış anlamanın düzeltilebilmesi için müşteriye işi anladığını göstermelidir. Son olarak, tasarımcıyı çırak yaklaşımı olarak kullanmak, gereksinimleri keşfetmenin ötesinde başka proje faydaları da sağlar. Örneğin, bu tekniği kullanmak, modellenmekte olan süreci iyileştirmeye yardımcı olabilir.

Hem müşteri hem de tasarımcı bu süreç boyunca öğrenir; müşteri neyin mümkün olabileceğini öğrenir ve tasarımcı da çalışma anlayışını genişletir. Ancak tasarımcının süreci iyileştirmek için bir fikri varsa, bu hemen (o anda) müşteriye geri gönderilmelidir.

Domain Analizi

Gereksinim mühendisliğinde alan bilgisine (ihtiyaç mühendisi ve/veya müşteri tarafından sahip olunmuş olsun) sahip olmanın önemini zaten vurgulamıştık. Etki alanı analizi, tasarlanan sistemle ilgili ve rekabet eden uygulamaların "manzarasını" değerlendirmeye yönelik herhangi bir genel yaklaşımı içerir. Böyle bir yaklaşım, temel işlevlerin ve daha sonra eksik işlevlerin belirlenmesinde faydalı olabilir. Etki alanı analizi, daha sonra yeniden kullanılabilir bileşenleri (nihai tasarıma dahil edilebilecek açık kaynaklı yazılım öğeleri gibi) belirlemek için de kullanılabilir. QFD ortaya çıkarma yaklaşımı açıkça alan analizini içerir ve bu tekniği kısaca tartışacağız.

Etnografik gözlem

Etnografik gözlem, dolaylı ve doğrudan faktörlerin gözleminin gereksinim mühendisinin işini bilgilendirdiği herhangi bir tekniği ifade eder. Etnografik gözlem, sosyal bilimlerden ödünç alınan bir tekniktir; bu teknikte, insan faaliyeti ve işin gerçekleştiği çevre gözlemlerinin, bilim adamını bazı fenomenlerin çalışmasında bilgilendirmek için kullanıldığı bir tekniktir. En katı anlamıyla, etnografik gözlem, uzun gözlem periyodlarını içerir (dolayısıyla, bir gereksinim belirleme tekniği olarak kullanımına itiraz edilir).

Etnografik gözlemi örneklemek için, farklı bir kültürü inceleyen bir antropologun toplumsal dalgınlığını hayal edin. Antropolog, incelenen kültür arasında yaşar, ancak bu, minimal düzeyde müdahalecidir. Kültür içerisinde yemek yerken, uyurken, avlanırken, kutlarken, yas tutarken vb. o toplumun nasıl işlediğine ve inanç sistemlerine dair doğrudan ve dolaylı her türlü delil toplanır.

Gereksinimlerin ortaya çıkarılmasına etnografik gözlem uygularken, gereksinim mühendisi, müşterinin işyeri kültürüne kendini kaptırır. Burada, otomatikleştirilecek işi veya faaliyeti gözlemlemeye ek olarak, gereksinim mühendisi aynı zamanda, doğrudan iletilemeyen, çevreden türetilen müşteri ihtiyaçlarının kanıtlarını toplayabilecek bir konumdadır. Çırak olarak tasarımcı, etnografik gözlem etkinliğini içeren bir gereksinim ortaya çıkarma tekniğidir.

Bu tekniği pratikte göstermek için, etnografik gözlemin meydana geldiği şu durumu göz önünde bulundurun:

- Bir müşteri için akıllı bir ev için gereksinimleri topluyorsunuz.
- Müşteriyle ne istedikleri hakkında röportaj yapmak için uzun zaman harcıyorsunuz.
- Müşteriler günlerini sürdürürken ve sorular sorarken onlarla etkileşim kurmak için zaman harcarsınız ("bulaşık makinesini neden gece çalıştırıyorsunuz, neden sabah çalıştırmıyorsunuz?").
- İstekler ve arzular hakkında sözsüz ipuçları almak için mevcut evde müşteriyi "hareket halindeyken" pasif bir şekilde gözlemleyerek uzun zaman harcıyorsunuz.
- Evin kendisinden başka bilgiler edinirsiniz - kitaplıktaki kitaplar, duvardaki tablolar, mobilya stilleri, hobilerin kanıtı, çeşitli cihazlarda aşınma ve yıpranma belirtileri vb.

Etnografik gözlem çok zaman alıcı olabilir ve gözlemcinin ciddi bir eğitim almasını gerektirir. Sürecin müdahaleciliğine dayanan başka bir itiraz daha var. Fizikte, Heisenberg belirsizlik ilkesi olarak bilinen iyi bilinen bir ilke vardır; bu, sıradan kişilerin terimleriyle, ölçtüğünüz şeyi etkilemeden bir şeyi tam olarak ölçemeyeceğiniz anlamına gelir. Bu nedenle, örneğin, bir müşteri için çalışma ortamını gözlemlerken, herkes etkilemek için dışarı çıktığı için süreçler ve davranışlar değişir - bu nedenle durumun yanlış bir resmi oluşur ve hatalı kararlara yol açar.

Hedefe Dayalı Yaklaşımlar

Hedefe dayalı yaklaşımlar, gereksinimlerin, gereksinimlere yol açan bir dizi hedef yoluyla misyon bildiriminden kaynaklandığı kabul edilen herhangi bir ortaya çıkarma tekniğini içerir. Yani, misyon beyanına bakıldığında, o misyonu yerine getiren bir dizi hedef oluşturulur. Bu hedefler, daha düşük seviyeli hedeflere ulaşmak için bir veya daha fazla kez alt bölümlere ayrılabilir. Daha sonra, alt düzey hedefler belirli üst düzey gereksinimlere ayrılır. Son olarak, üst düzey gereksinimler daha düşük düzeyli gereksinimler oluşturmak için kullanılır.

Örneğin, bagaj taşıma sistemi misyon beyanını göz önünde bulundurun:

Yolcu kalkış noktasından varış noktasına kadar bagaj taşımanın tüm yönlerini otomatikleştirmek.

Bu misyonun yerine getirilmesi için aşağıdaki hedefler düşünülebilir:

- Hedef 1: Check-in'den teslim alınmasına kadar bagaj takibini tamamen otomatik hale getirmek.
- Hedef 2: Bagajın check-in kontuarından uçağa yönlendirilmesini tamamen otomatikleştirmek
- Hedef 3: Kayıp bagaj miktarını %1'e düşürmek

Bu hedefler daha sonra, hedef-soru-metrik (GQM) gibi yapılandırılmış bir yaklaşım kullanılarak gereksinimlere ayrıştırılabilir. GQM, gereksinim mühendisliği, mimari tasarım, sistem tasarımı ve proje yönetimi gibi sistem mühendisliğinin birçok alanında kullanılan önemli bir tekniktir. GQM üç adımı içerir: sistemin amaçlarını veya hedeflerini belirtin; her bir

hedeften, hedefe ulaşılıp ulaşılmadığını belirlemek için cevaplanması gereken soruları türet; soruları cevaplayabilmek için neyin ölçülmesi gerektiğine karar verin (Basili ve Weiss 1984).

Örneğin, bagaj taşıma sistemi söz konusu olduğunda, hedef 3'ü göz önünde bulundurun. Burada ilgili soru "belirli bir (havaalanı/havayolu/uçuş/zaman aralığı/vs.) için bagajın yüzde kaçı kayboluyor?"dur. Bu soru, formun bir gereksinimini ortaya koymaktadır:

Belirli bir [havaalanı/havayolu/uçuş/zaman aralığı/vb.] için kaybolan bagaj yüzdesi, %1'den fazla olmayacaktır.

Bu gereklilik için ilgili ölçü, o zaman, belirli bir (havaalanı/havayolu/uçuş/zaman aralığı/vb.) için basitçe kaybedilen pabuçların yüzdesidir. Tabii ki, kayıp bagaj için gerçekten bir tanıma ihtiyacımız var, çünkü sözde kayıp bagaj, genellikle kayıp ilan edildikten günler hatta aylar sonra tekrar ortaya çıkıyor. Ayrıca, bir havalimanının belirli bir süre boyunca veya belirli bir havayolu için bazı terminallerdeki rapor edilen bagaj kayıpları açısından bu gerekliliği çerçevelerken makul varsayımlar yapılması gerekir.

Her halükarda, burada bilerek basit bir örnek seçtik - bir hedef (gereksinim) için uygun soru her zaman çok açık değildir ve ilgili metrik sorudan bu kadar kolay elde edilemez. GQM'nin gücünü gerçekten gösterdiği yer burasıdır.

Örneğin, aşağıdakilerin bir varyasyonu olan gereksinimleri görmek yaygındır:

Sistem kullanıcı dostu olacaktır

Böyle bir gereksinimle ilgili sorun, memnuniyetini göstermenin bir yolu olmamasıdır - kabul testi ekibindeki herhangi bir kişi, sistemin kullanıcı dostu olmadığını beyan edebilir. Ancak kullanıcı dostu olmaması sistem için makul bir hedeftir. Bu nedenle, GQM'yi takiben, kullanıcı dostu olma hedefine ilişkin bir dizi soru oluşturuyoruz, örneğin:

1. Bir kullanıcının belirli işlevleri nasıl gerçekleştireceğini öğrenmesi için geçen süre
2. Bir kullanıcının belirli bir süre boyunca yardım özelliğini kaç kez kullanması gerektiği
3. Bir kullanıcının belirli bir süre boyunca belirli işlemler sırasında bir hata mesajı görme sayısı.

Son olarak, bu metrikler için kabul edilebilir aralıklar belirlemek üzere müşteriyle birlikte çalışırız. Sistem oluşturulduktan sonra, gerçek kullanıcılarla test denemeleri yoluyla gereksinimlerin tatmini gösterilebilir. Kullanıcıların özellikleri ve sayısı, test süresi vb. gibi bu testin parametreleri daha sonra tanımlanabilir ve sistem test planına dahil edilebilir. Bu şekilde kabul edilebilir bir kullanım kolaylığı seviyesi tanımlayabiliriz.

Grup Çalışması

Grup çalışması, ihtiyaç keşfi, analizi ve takibi süreçlerinde kullanılan her türlü grup toplantılarının genel adıdır. Gereksinimlerin ortaya çıkarılması için grup odaklı çalışmaların en ünlüsü, birazdan tartışacağımız ortak uygulama tasarımıdır (JAD).

Grup faaliyetleri, birçok paydaşı bir araya getirme açısından çok verimli olabilir, ancak çatışma ve bölünme potansiyelini riske atabilir. Her türlü grup çalışmasında başarının anahtarı, grup toplantılarının planlanması ve yürütülmesidir. İşte grup toplantıları hakkında hatırlanması gereken en önemli şeyler.

- Ödevini yap—kuruluşun tüm yönlerini, sorunları, siyaseti, çevreyi vb. araştırın.
- Toplantı gerçekleşmeden birkaç gün önce bir gündem (her madde için ayrılan süre ile) yayınlayın.
- Toplantı boyunca gündemde kalın (toplantı kapsamı kayması yok).
- Elinizde özel bir not alıcı (yazıcı) bulundurun.
- Kişisel sorunların araya girmesine izin vermeyin.
- Herkesin sesini duyurmasına izin verin.
- İlk fırsatta fikir birliği arayın.
- Gündemdeki tüm maddeler yeterince tartışılana kadar ayrılmayınız.
- Toplantı tutanaklarını toplantı kapanışından sonraki birkaç gün içinde yayınlayın ve katılımcıların değişiklik önermesine izin verin.

Bu ilkeler, gereksinimlerin ortaya çıkarılmasına yönelik JAD yaklaşımı için devreye girecektir.

Her türden grup çalışmasının birçok dezavantajı vardır. İlk olarak, grup toplantılarını organize etmek ve dahil olan birçok paydaşın konulara odaklanmasını sağlamak zor olabilir. Açıklık ve samimiyet sorunları da ortaya çıkabilir çünkü insanlar her zaman gerçek duygularını halka açık bir forumda ifade etmeye istekli değildir. Herkesin farklı bir kişiliği olduğundan, belirli kişiler toplantıya hakim olabilir (ve bunlar en “önemli” kişiler olmayabilir). Birkaç kişinin toplantıya sahip olmasına izin vermek, diğer katılımcıların çoğu için “dışlanmış” hissine yol açabilir.

Etkili toplantılar yürütmek ve dolayısıyla grup çalışmasını kullanmak son derece gelişmiş liderlik, organizasyon ve kişilerarası beceriler gerektirir. Bu nedenle, gereksinim mühendisi mümkün olduğunda bu becerileri geliştirmeye çalışmalıdır.

Mülakatlar

Mülakatlar yoluyla ortaya çıkarma, iki kişi arasındaki yüz yüze iletişimi içerir.

Bireysel paydaşlar veya küçük bir paydaş grubu (bazen odak noktası olarak adlandırılır)

grup). Mülakatlar, sistem düzeyinde gereksinimleri ortaya çıkarmak için kullanımı kolay bir tekniktir.

Özellikle kullanılabilirlik gereksinimleri yönünden.

Ortaya çıkarma faaliyetlerinde kullanılabilecek üç tür görüşme vardır ve

Bireylere veya odak gruplarına uygulanabilirler:

- Yapılandırılmamış
- Yapılandırılmış
- Yarı yapılandırılmış

Muhtemelen en yaygın tür olan yapılandırılmamış görüşmeler, doğası gereği konuşma ve katılımcıları rahatlatmaya hizmet eder. Gereksinimler ne zaman olursa olsun, herhangi bir zamanda ve herhangi bir yerde ortaya çıkabilir. Mühendis ve paydaş bir aradadır ve bu şekilde bilgi edinme fırsatı asla kaybolmamalıdır. Ancak görüşmecinin becerisine bağlı olarak, yapılandırılmamış görüşmeler isabetli olabilir veya ıskalanabilir. Bu nedenle, yapılandırılmış veya yarı yapılandırılmış mülakat tercih edilir.

Yapılandırılmış görüşmeler, doğası gereği çok daha resmidir ve önceden tanımlanmış özenle planlanmış sorulardır. Şablonlar, yapılandırılmış stili kullanarak mülakatlarda çok faydalıdır. Yapılandırılmış görüşmelerin en büyük dezavantajı, biçim çok kontrollü olduğu için bazı müşteriler bilgileri saklayabilir.

Yarı yapılandırılmış görüşmeler, yapılandırılmış ve yapılandırılmamış en iyileri bir araya getirir.

Yani, gereksinim mühendisi dikkatlice düşünülmüş bir liste hazırlar ancak daha sonra kendiliğinden yapılandırılmamış soruların görüşme sırasında içeri girmesine izin verir.

Yapılandırılmış görüşmeler tercih edilirken, hangisinin kullanılacağının seçimi çok fırsatçı bir karardır. Örneğin, müşterinin kurumsal kültürü çok resmi olmayan ve rahat ve güven yüksekse, yapılandırılmamış görüşmeler olabilir. Daha karmaşık, süreç odaklı bir organizasyonda, yapılandırılmış ve yarı yapılandırılmış görüşmeler muhtemelen daha fazla tercih edilir.

İşte üç görüşme türünden herhangi birinde kullanılabilecek bazı örnek görüşme soruları:

- Sistemin önemli bir özelliğini adlandırın.
- Bu özellik neden önemlidir?
- En önemlisi beş olmak üzere birden beşe kadar bir ölçekte bu özelliği nasıl değerlendirirsiniz?
- Bu özellik diğer özelliklere göre ne kadar önemlidir?
- Başka hangi özellikler bu özelliğe bağlıdır?
- Başka hangi özellikler bu özellikten bağımsız olmalıdır?

- Bu özellik hakkında başka hangi gözlemlerde bulunabilirsiniz?

Hangi görüşme tekniği kullanılırsa kullanılsın, tüm doğru soruların sorulduğundan emin olmak için özen gösterilmelidir. Yani, önemli soruları atlamayın ve rahatsız edici veya gereksiz soruları dahil etmeyin. Kesinlikle gerekli olduğunda, görüşmeler telefon, video konferans veya e-posta yoluyla yapılabilir, ancak bu iletişim modlarında yanıtların bazı önemli nüanslı yönlerinin kaybolabileceğini unutmayın.

İç Gözlem

Bir gereksinim mühendisi, müşterinin ne istediğini düşündüğüne dayalı olarak gereksinimler geliştirdiğinde, iç gözlem sürecini yürütüyor demektir. Özünde, gereksinim mühendisi kendini müşterinin yerine koyar ve “müşteri olsaydım sistemin bunu yapmasını isterdim…

Gereksinim mühendisinin alan bilgisi müşterininkinden çok daha fazla olduğunda, içe dönük bir yaklaşım yararlıdır. Bazen müşteri, mühendise aşağıdakine benzer sorular soracaktır: "Ben olsan ne isterdin?" İç gözlem, gereksinim mühendisinin etkileşimlerinin her yönünü bilgilendirecek olsa da, bir müşteriye ne istemesi gerektiğini söylememe konusundaki uyarımızı unutmayın.

Ortak Uygulama Tasarımı*

Ortak uygulama cihazı (JAD), sistem kullanıcıları, sistem sahipleri ve uzun bir süre boyunca belirli bir dizi soruna odaklanan analistler ile yüksek düzeyde yapılandırılmış grup toplantıları veya mini geri çekilmeleri içerir. Bu toplantılar günde 4-8 saat ve 1 günden birkaç haftaya kadar süren bir süre içinde gerçekleşir. JAD, katılımcılar aynı yerde olmadığında çok bölgeli uygulama için bile uyarlanmıştır (Cleland-Huang ve Laurent 2014). Geleneksel olarak büyük hükümet sistemleri projeleriyle ilişkilendirilse de teknik, endüstriyel ortamlarda her boyuttaki sistemlerde kullanılabilir.

JAD ve JAD benzeri teknikler, problemler, hedefler ve gereksinimler üzerinde grup mutabakatı sağlamak için sistem planlama ve sistem analizi faaliyetlerinde yaygın olarak kullanılır. Spesifik olarak, gereksinim mühendisi, operasyon tanımı kavramı, sistem hedefi tanımı, gereksinimlerin ortaya çıkarılması, gereksinim analizi, gereksinim belgesi incelemesi ve daha fazlası için JAD oturumlarını kullanabilir.

Bir JAD incelemesi veya denetim oturumu planlaması üç adımdan oluşur:

1. Katılımcıların seçilmesi

2. Gündemin hazırlanması

3. Bir yer seçme

Bu adımların her birini hazırlarken büyük özen gösterilmelidir.

İncelemeler ve denetimler, aşağıdaki katılımcıların bir kısmını veya tamamını içerebilir:

- Sponsorlar (ör. Üst yönetim)
- Bir ekip lideri (kolaylaştırıcı, bağımsız)

- Gereksinimlerin ve iş kurallarının sahibi olan kullanıcılar ve yöneticiler
- Katipler (yani toplantı tutanakları ve not tutanlar)
- Mühendislik personeli

*a.e.

Sponsor, analistler ve yöneticiler bir lider seçer. Lider kurum içi veya danışman olabilir. Normalde yazılım geliştirme ekibinden bir veya daha fazla yazar (not tutucu) seçilir. Analist ve yöneticiler, kullanıcı topluluğundan bireyler seçmelidir. Bu kişiler kendi iş alanlarında bilgili ve açık sözlü olmalıdır.

Bir oturum planlamadan önce, analist ve sponsor projenin kapsamını belirlemeli ve her oturumun üst düzey gereksinimlerini ve beklentilerini belirlemelidir.

Oturum lideri ayrıca sponsorun insanları, zamanı ve diğer kaynakları çabaya adamaya istekli olduğundan emin olmalıdır. Gündem büyük ölçüde yürütülecek incelemenin türüne bağlıdır ve yeterli zamana izin verecek şekilde oluşturulmalıdır. Gündem, kod ve belgeler ayrıca tüm katılımcılara onları gözden geçirmek, yorum yapmak ve soru sormaya hazırlanmak için yeterli zamana sahip olmaları için toplantıdan çok önce gönderilmelidir.

Aşağıda yazılım gereksinimleri, tasarım denetimleri veya kod adım adım ilerlemek için bazı kurallar yer almaktadır. Oturum lideri, bu uygulamaların uygulanmasını sağlamak için her türlü çabayı göstermelidir.

- Gündeme bağlı kalın.
- Programa bağlı kalın (gündem konularına belirli bir zaman ayrılır).
- Yazıcının not alabildiğinden emin olun.
- Teknik jargondan kaçının (inceleme teknik olmayan personeli içeriyorsa).
- Çatışmaları çözün (onları ertelememeye çalışın).
- Grup fikir birliğini teşvik edin.
- Kişilere izin vermeden kullanıcı ve yönetim katılımını teşvik edin.
- Toplantıyı kişisel olmayan tutun.
- Toplantıların gerektiği kadar uzun sürmesine izin verin.

Herhangi bir gözden geçirme oturumunun son ürünü, tipik olarak, oturum sırasında üzerinde anlaşmaya varılan öğelerin (spesifikasyonlar, tasarım değişiklikleri, kod değişiklikleri ve eylem öğeleri) bir özetini sağlayan resmi bir yazılı belgedir. Belgenin içeriği ve organizasyonu, açık bir şekilde, oturumun doğasına ve amaçlarına bağlıdır. Ancak gereksinimlerin ortaya çıkması durumunda, ana eser SRS'nin ilk taslağı olabilir.

Merdivenleme

Merdivenlemede, gereksinim mühendisi, gereksinimleri ortaya çıkarmak için müşteriye kısa yönlendirici sorular (sondalar) sorar. Daha sonra yüzeyin altında daha derine inmek için takip eden sorular sorulur. Yanıtlardan elde edilen bilgiler daha sonra ağaç benzeri bir yapı halinde düzenlenir.

Tekniği göstermek için, evcil hayvan mağazası POS sistemi için aşağıdaki merdiven sorularını ve yanıtlarını göz önünde bulundurun. "RE" gereksinim mühendisini ifade eder.

RE: Sistemin önemli bir özelliğini adlandırın.

Müşteri: Müşteri kimliği.

RE: Bir müşteriyi nasıl tanımlarsınız?

Müşteri: Sadakat kartlarını(Sadakat kartı, işletmelerin müşterilerini teşvik etmek, hizmetlerini kullanmaya devam etmesini sağlamak için ücretsiz veya küçük bir ücret karşılığında müşterilere verdiği kartlardır)okutabilirler.

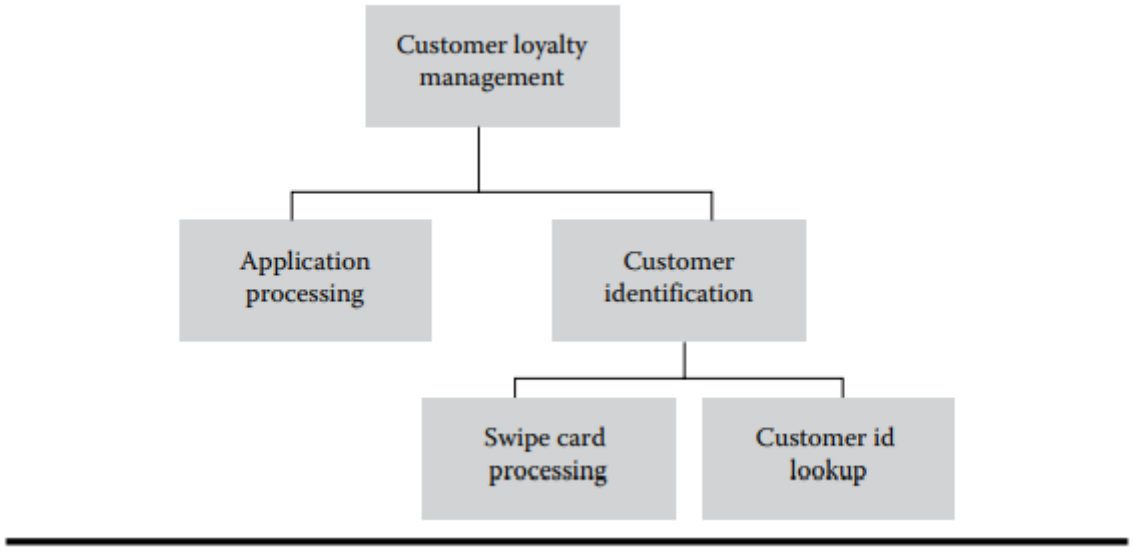
RE: Bir müşteri kartını unutursa ne olur?

Müşteri: Telefon numarasından aranabilirler.

RE: Müşterinin telefon numarasını ne zaman alırsınız?

Müşteri: Müşteriler sadakat kartı başvurusunu tamamladığında.

RE: Müşteriler başvuruları nasıl tamamlıyor? … Ve bunun gibi.



Şekil 3.3: Pet shop POS sistemi için merdiven şeması.

Şekil 3.3, sorulara verilen yanıtların bir merdiven veya hiyerarşik diyagramda nasıl organize edildiğini gösterir. Merdivenleme tekniği, bilginin hiyerarşik bir biçimde düzenlenebileceğini varsayar veya en azından bilginin hiyerarşik olarak düzenlenmesine neden olur.

Protokol Analizi

Protokol analizi, müşterilerin gereksinim mühendisleriyle birlikte otomatikleştirecekleri prosedürleri gözden geçirdikleri bir süreçtir. Böyle bir gözden geçirme sırasında müşteriler, atılan her adımın gerekçesini açıkça belirtir.

Örneğin, Bölüm 1'deki etki alanı kelime anlama bölümünde tartışılan büyük bagaj teslimat şirketi için, yoğun kış tatili sezonunda mühendislerin ve diğer destek profesyonellerinin düzenli teslimat personeliyle birlikte yolculuk yapması bir uygulamaydı. Bu uygulama, yalnızca teslim edilecek paketlerdeki dalgalanmayı ele almakla kalmadı, aynı zamanda mühendisleri, şirketin hizmetleriyle ilgili süreçler ve prosedürler hakkında, gerçekte uygulandıkları şekliyle yeniden tanıştırdı. Mühendisler tarafından sahada yapılan gözlemler genellikle süreç optimizasyonuna ve diğer yeniliklere yol açtı.

Birazdan protokol analizinin çırak olarak tasarımcıya çok benzediğini göreceksiniz, ancak ince farklılıklar var. Bu farklılıklar, çırak olarak tasarımcıdan ziyade protokol analizinde daha pasif olan gereksinim mühendisinin rolünde yatmaktadır.

Prototipleme

Prototipleme, özellikle kullanılabilirlik gereksinimleri olmak üzere yeni özellikleri keşfetmek için sistem modellerinin oluşturulmasını içerir. Prototipleme, gereksinimlerin ortaya çıkarılması için özellikle önemli bir tekniktir. Örneğin, sarmal yazılım geliştirme modelinde yaygın olarak kullanılır ve çevik metodolojiler, esas olarak, giderek daha işlevsel hale gelen bir dizi atılabilir prototipten oluşur.

Prototipler, çalışan modelleri ve çalışmayan modelleri içerebilir. Çalışan modeller, yazılım sistemleri ve simülasyonlar durumunda yürütülebilir kodu veya yazılım olmayan sistemler için geçici veya ölçekli prototipleri içerebilir. Çalışmayan modeller, storyboard'ları ve kullanıcı arayüzlerinin maketlerini içerebilir. Bina mimarları, müşteri gereksinimlerini ortaya çıkarmaya ve onaylamaya yardımcı olmak için düzenli olarak prototipleri (örneğin ölçekli çizimler, karton modeller, 3 boyutlu bilgisayar animasyonları) kullanır. Sistem mühendisleri prototipleri aynı nedenlerle kullanır.

Çalışan yazılım prototipleri söz konusu olduğunda, kod kasıtlı olarak atılacak şekilde tasarlanabilir veya kasıtlı olarak yeniden kullanılmak üzere (atılmadan) tasarlanabilir. Örneğin, grafiksel kullanıcı arayüzü kod maketleri gereksinimlerin ortaya çıkarılması için kullanılabilir ve kod yeniden kullanılabilir. Çevik yazılım geliştirme metodolojileri, atılmayan prototipleri sürekli olarak geliştirme sürecini içerir.

Son zamanlarda, 3 boyutlu baskı, belirli sistemlerin fiziksel modellerini oluşturmada önemli bir araç haline geldi. 3-D baskının diğer hızlı prototipleme teknolojilerine göre iki önemli avantajı vardır: Birincisi maliyettir—endüstriyel kalitede 3 boyutlu yazıcılar birkaç bin dolara satın alınabilirken, geleneksel bilgisayar sayısal kontrolünü (CNC) kullanan hızlı prototipleme makineleri birkaç yüz bin dolara mal olabilir. İkinci avantaj, 3 boyutlu yazıcıların, yaygın olarak kullanılan bilgisayar destekli tasarım (CAD) programları tarafından üretilen standart formattaki dosyaları girdi olarak alabilmesidir (Berman 2012).

Prototiplemeyi kullanmanın birkaç farklı yolu vardır - örneğin, dördüncü nesil bir ortamda (yani bir simülatörde), kullanılıp atılan prototipleme, evrimsel prototipleme (prototipin nihai sisteme dönüştüğü yer) veya kullanıcı arayüzü prototiplemesi. Bazı kuruluşlar birden fazla prototipleme türü kullanabilir. Kassab ve diğerleri. (2014), gereksinim mühendisliği ile ilgili çeşitli uygulamaların

yaygınlığını belirlemek için yazılım uzmanlarıyla geniş bir anket yaptı. Bu farklı prototipleme teknikleri için seçim sıklığına ilişkin sonuçları Şekil 3.4'te gösterilmektedir.

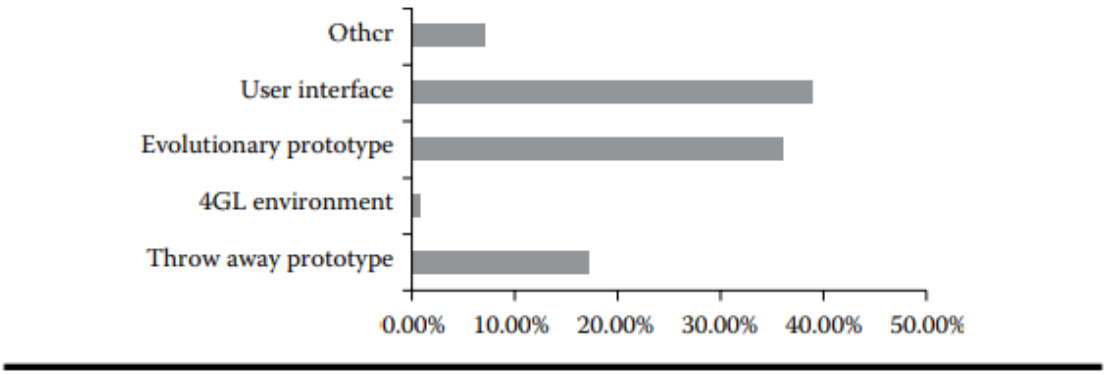
Gereksinimlerin ortaya çıkarılması için prototiplemeyi kullanırken göz önünde bulundurulması gereken en az üç tehlike vardır:

İlk olarak, bazı durumlarda, tutulması amaçlanmayan yazılım prototipleri, aslında program baskıları nedeniyle tutulur. Bu durum potansiyel olarak tehlikelidir, çünkü kod muhtemelen en titiz teknikler kullanılarak tasarlanmamıştır. Kullanılıp atılan prototiplerin istenmeyen yeniden kullanımı, endüstride sıklıkla meydana gelir.

İkinci sorun, prototiplemenin belirli işlevsel olmayan gereksinimleri keşfetmede her zaman etkili olmamasıdır. Bu, özellikle yalnızca geçerli standartların ve yasaların analiziyle elde edilebilecek gereksinimler için geçerlidir (Kassab ve Ormandjieva 2014)

Son olarak, kullanıcıların sistemle etkileşim kurma yollarını keşfetmek için prototipler kullanılırken sorunlar ortaya çıkabilir. Ana endişe, kullanıcıların bir prototiple (davranışın sonuçlarının gerçek olmadığı) gerçek sistemle farklı şekilde etkileşime girmesidir. Örneğin, bir çarpışmadan kaynaklanan gerçek bir yaralanma veya hasarın olmadığı bir araç simülatöründe kullanıcıların nasıl araç kullanabileceğini düşünün. Sürücüler

simülatörde gerçek bir araçta olduğundan çok daha agresif davranır ve muhtemelen hatalı gereksinimlerin keşfedilmesine yol açar



Şekil 3.4 Yazılım geliştirme yaşam döngüsü metodolojisi boyunca prototip yöntemleri seçimi. (Kassab, M., Neill, C & Laplante, P. State of application in gereksinim mühendisliği: çağdaş verilerden uyarlanmıştır. Innovations in Systems and Software Engineering, 10, no. 4 (2014): 235–241. İzinli.)

Kalite Fonksiyonu Dağıtımı*

Kalite fonksiyon yayılımı (QFD), müşteri gereksinimlerini keşfetmeye ve üretim aşaması boyunca kullanılacak ana kalite güvence noktalarını tanımlamaya yönelik bir tekniktir. QFD, müşterilerin ihtiyaç ve isteklerinin dikkatlice duyulmasını ve ardından analizden uygulamaya ve dağıtıma kadar doğrudan bir şirketin dahili teknik gereksinimlerine dönüştürülmesini sağlamak için bir yapı sağlar.

QFD'nin temel fikri, müşteri ihtiyaçları, teknik gereksinimler, öncelikler ve (gerekirse) rakip değerlendirmesi arasında ilişki matrisleri oluşturmaktır. Özünde, QFD, kart sıralama, merdivenleme ve alan analizini içerir.

*a.e.

Bu ilişki matrisleri genellikle bir evin çatısı, tavanı ve yanları olarak temsil edildiğinden, QFD bazen "kalite evi" olarak anılır (Şekil 3.5; Akao 1990).

QFD, 1966 yılında Yoji Akao tarafından imalat, ağır sanayi ve sistem mühendisliğinde kullanılmak üzere tanıtıldı. Ayrıca IBM, DEC, HP, AT&T, Texas Instruments ve diğerleri tarafından yazılım sistemlerine uygulanmıştır.

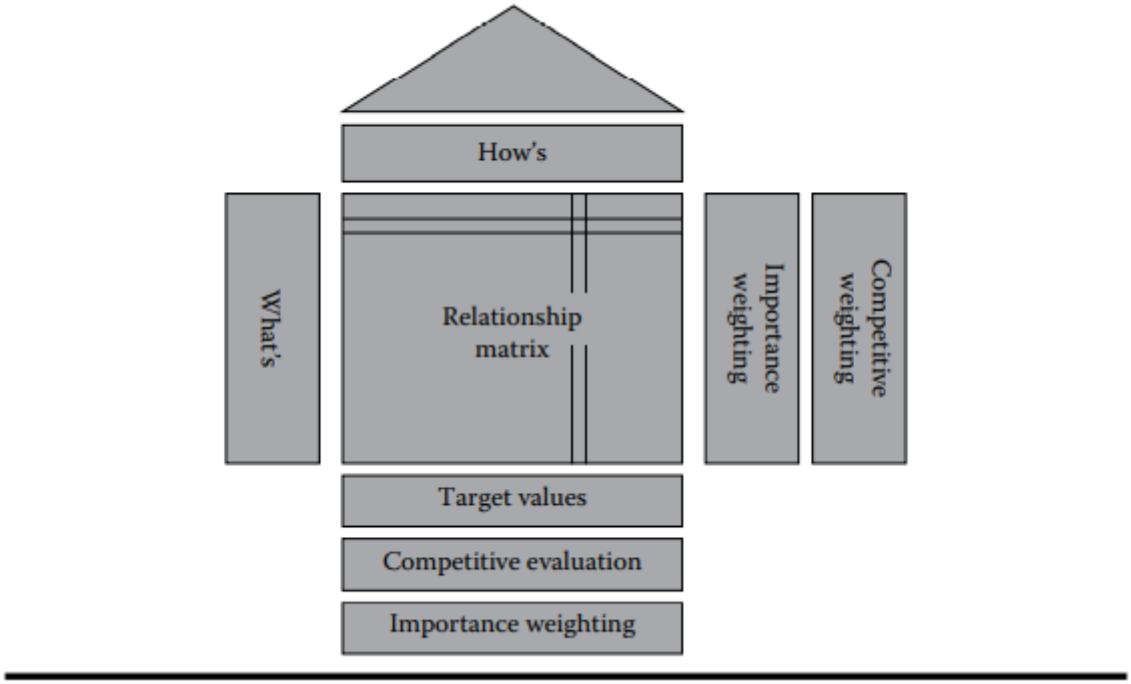
"Müşterinin sesi"ne atıfta bulunduğumuzda, müşterinin kendi sözleriyle ifade ettiği gibi, gereksinim mühendisinin üründen neye ihtiyaç duyduklarını anlamak için müşterileri empatik bir şekilde dinlemesi gerektiğini kastediyoruz. Müşterinin sesi, ürünlerin yalnızca "mühendis sesinden" geliştirilmemesini sağlamak için tüm analiz, tasarım ve geliştirme faaliyetlerinin temelini oluşturur. Bu yaklaşım, gereksinimlerin ortaya çıkarılmasının özünü somutlaştırır.

Aşağıdaki gereksinim mühendisliği süreci QFD tarafından belirlenir:

- Paydaşların niteliklerini veya gereksinimlerini tanımlayın.
- Gereksinimlerin teknik özelliklerini belirleyin.
- Gereksinimleri teknik özelliklerle ilişkilendirin.
- Rakip ürünlerin bir değerlendirmesini yapın.
- Teknik özellikleri değerlendirin ve her özellik için bir hedef değer belirleyin.
- Geliştirme çabası için teknik özelliklere öncelik verin.

QFD, rekabetçi analiz için yapılandırılmış bir yaklaşım kullanır. Yani, rakip ürünler için ilgili tüm özelliklerin birleşiminden bir özellik listesi oluşturulur. Bu özellikler, bir rekabet matrisinin sütunlarını oluşturur.

Satırlar, rakip ürünleri temsil eder ve karşılık gelen hücreler, her üründe bulunan özellikler için doldurulur. Matris daha sonra yeni veya revize edilmiş ürün için bir başlangıç gereksinimleri seti formüle etmek için kullanılabilir. Matris ayrıca, yeni sistemdeki temel özelliklerin atlanmamasını sağlamaya yardımcı olur ve istenen gereksinimlerin eksiksizlik kalitesinin iyileştirilmesine katkıda bulunabilir.



Şekil 3.5 QFD'nin "kalite evi" (Akao 1990)

Örneklemek için, evcil hayvan mağazası POS sistemi için kısmi bir rekabet analizi Tablo 3.1'de gösterilmektedir.

Yalnızca çok yüksek düzeyli özelliklerin gösterildiğine dikkat edin, ancak istenen ayrıntı düzeyine inerek matrisi büyük ölçüde genişletebiliriz. Matris bize zorunlu ve isteğe bağlı özelliklerden oluşan bir başlangıç seti sunar. Örneğin, tüm bu ürünlerde kablosuz desteğinin bulunduğunu belirtmek, yeni pet shop POS sisteminde böyle bir özelliğin zorunlu olduğunu gösterebilir.

Gereksinim mühendisliğine toplam yaşam döngüsü yaklaşımını dahil ettiğinden, QFD'nin diğer bağımsız ortaya çıkarma tekniklerine göre birçok avantajı vardır. QFD, kullanıcıların ve yöneticilerin katılımını geliştirir. Geliştirme yaşam döngüsünü kısaltır ve genel proje geliştirmeyi iyileştirir. QFD, iletişim süreçlerini yapılandırarak ekip katılımını destekler. Son olarak, bilgi kaybını önleyen önleyici bir araç sağlar.

Bununla birlikte, QFD'nin bazı dezavantajları vardır. Örneğin, zamansal gereksinimleri ifade etmede zorluklar olabilir. Ve tamamen yeni bir proje türü ile QFD'yi kullanmak zordur - var olmayan bir şey için müşteri gereksinimlerini nasıl keşfedersiniz ve rekabetçi ürünleri nasıl oluşturur ve analiz edersiniz? Bu durumlarda çözüm, benzer veya ilgili ürünlere bakmaktır, ancak yine de bilişsel bir boşluk olma eğilimi vardır.

	Competing Product		
Feature	*MyFavoritePet*	*BestFriends*	*Fido-2.0*
Maximum simultaneous users supported	100	250	Unlimited
Wireless device support	Yes	Yes	Yes
Business analytics features	Yes	Yes	No
Operating system support	Windows/ Mac/Linux	Windows/Linux	Windows/Mac
Cost (base system) ($K)	50	110	75

Tablo 3.1 Pet Mağaza Satış Noktası Sistemi için Kısmi Rekabet Analizi

Bazen belirli fonksiyonlar için ölçümler bulmak ve soyutlama seviyesini tek tip tutmak zordur. Ve ne kadar az bilirsek, o kadar az belgeliyoruz. Son olarak, özellik listesi kontrolsüz bir şekilde büyüdükçe, kalite evi bir "konak" haline gelebilir.

QFD, birincil gereksinimleri ortaya çıkarma yaklaşımı olarak kullanılmasa bile, mümkün olan her yerde rekabetçi sistem analizine yönelik yaklaşımı kullanılmalıdır. QFD rekabetçi analizinin yapılandırılmış doğası, daha eksiksiz bir gereksinim setine yol açan hiçbir önemli gereksinimin eksik olmadığından emin olmanın etkili bir yoludur.

Anketler

Gereksinim mühendisleri genellikle anketleri ve diğer anket araçlarını geniş paydaş gruplarına ulaşmak için kullanır. Anketler genellikle kapsam sınırlarını hızlı bir şekilde tanımlamak için ortaya çıkarma sürecinin erken aşamalarında kullanılır.

Her türden anket sorusu kullanılabilir. Örneğin, sorular kapalı ya da (örneğin, çoktan seçmeli, doğru-yanlış) açık uçlu—serbest biçimli yanıtları içerir.Kapalı sorular, analiz için daha kolay kodlama

avantajına sahiptir ve sistemin kapsamını bağlamaya yardımcı olur. Açık sorular daha fazla özgürlük ve yenilik sağlar, ancak analiz edilmesi daha zor olabilir ve kapsam kaymasını teşvik edebilir.

Örneğin, evcil hayvan mağazası POS sistemi için olası bazı anket soruları şunlardır:

- Envanterinizde kaç tane benzersiz ürün (SKU) taşıyorsunuz?

(a) 0–1000; (b) 1001–10,000; (c) 10.001–100.000; (d) >100.000

- Kaç farklı depo siteniz var? ___
- Kaç farklı mağaza konumunuz var? ____
- Şu anda kaç benzersiz müşteriniz var? ____

Sorular, kapalı uçlu sorular için bile yeterince çerçevelenmemişse, fazla kapsam ve alt kapsam belirleme tehlikesi vardır. Bu nedenle, anket çalışması teknikleri alan paydaşlar ve gereksinim mühendisileri tarafından da çok iyi anlaşıldığında kullanışlıdır.

Büyük ölçekli anketler yapmadan önce amaçlanan anket popülasyonunun küçük bir alt kümesi ile bir pilot çalışma yapmak önemlidir.Sonuçlar analiz edilir ve kafa karıştırıcı, eksik veya konu dışı soruları belirlemek amacıyla anket katılımcıları ile görüşülmüştür. Daha sonra araç, anketi daha büyük bir nüfusa uygulamadan önce rafine edilebilir.

Anket verilerini analiz ederken, özellikle katılımcılardan kimliklerini belirlemelerini isterken ve istenen özellikleri sıralarken, aşağıdaki etkiye dikkat edin.

Gerçekleştirilmesi gerekmeyen bir dizi seçenek verildiğine; kişi,eğer karar verildiyse çok daha fazla sayıda seçeneği arzu etme eğiliminde olacaktır.

Aşağıdaki örnekten dolayı bu etkiyi "dondurma marketi etkisi" olarak adlandırıyoruz. El yapımı bir dondurma dükkanı açmaya karar veren bir girişimci düşünün.Ürün araştırmasının bir parçası olarak, bir enstrümanla birkaç kişiye anket yapıyor yanıtlayanların satın alacakları dondurma lezzetlerini kontrol ediyor. Ankette listelenen 30 farklı lezzetten 20'sinin ankete katılanların %50'si veya daha fazlası tarafından seçildiğini öğreniyor. Böylece anket sonuçlarına göre bu 20 aromayı kabaca ankette belirtilen taleple orantılı olarak stoklayıp üretmeye karar verir. Yine de, dondurma dükkanını açtıktan 1 hafta sonra, işinin %90'ını çikolata, vanilya ve çilek aromalı satışların oluşturduğunu keşfediyor. Envanterinde tuttuğu diğer 17 lezzetten birkaçı daha önce hiç satılmamıştı. Ankette müşterilerin satın alacaklarını söylemelerine rağmen seçimlerini yapma zamanı geldiğinde, farklı davrandılar. Bu nedenle, müşterilere özellik

kümeleriyle ilgili seçim sunarken dondurma dükkanının etkisini unutmayın.Bir şey söylerler ama başka bir şey yaparlar.

Anketler telefon, e-posta, yüz yüze ve Web tabanlı olarak gerçekleştirilebilir. Çeşitli anket oluşturma ve kullanılması gereken sonuçları toplama ve analiz etme sürecini basitleştirmek için kullanılan ticari araçlar ve açık kaynaklı çözümler vardır.

Repertuar Izgaraları

Repertuar ızgaraları, çeşitli özellikler için yapılandırılmış sistemdeki farklı varlıklardır ve genellikle müşteriler

etki alanı uzmanları olduğunda kullanılan bir sıralama sistemi içerir.Repertuar ızgaraları, özellikle paydaş grupları içindeki anlaşma ve anlaşmazlığın belirlenmesi için yararlıdır.

Izgaralar, satırların sistemi varlıklar ve istenen nitelikleri temsil ettiği bir özellik veya kalite matrisi gibi görünür.

Ve sütunlar,paydaşların her birine dayalı sıralamaları temsil eder.

Izgaralar hem nitelikleri hem de özellikleri bünyesinde barındırabilirken,

genellikle şebekelerin, analiz ve uyuşmazlık çözümünün tutarlılığını sağlamak için tüm özelliklere veya tüm niteliklere sahip olması durumudur.

Tekniği göstermek için, Şekil 3.6 çeşitli bagaj taşıma sisteminin nitelikleri için bir repertuar ızgarasını temsil eder.

Burada, havalimanı operasyon müdürü için tüm niteliklerin esasen en yüksek öneme sahip olduğunu görüyoruz (güvenlik biraz

daha düşük, 4).

Ancak Havayolu İşçileri Sendikası temsilcisi için güvenlik en önemli şeydir (sonuçta sendika üyeliğinin sistemle günlük olarak etkileşime girmesi gerekir).

Özünde, bu derecelendirmeler, paydaşların gündemlerini veya farklı bakış açılarını yansıtır. Bu nedenle,paydaş hedeflerini içeren anlaşmazlıklarla erken yüzleşmede repertuar ızgaralarının kullanımının neden çok yararlı olabileceğini görmek kolaydır.

Ayrıca, ızgaralar paydaşların tutumlarını yakaladıkları için sistemin geliştirilmesi ve

daha sonra nitelikler ve özellikler hakkında anlaşmazlıklarla başa çıkmak için göz ardı edilmesi zor bir şekilde değerli belgeler sağlayabilir.

Yine de, repertuvar ızgaralarını kullanırken, dondurma dükkanının etkisini hatırlayın—paydaşlar halka açık bir ortamda bir şey söyleyecektir ve daha sonra farklı davranacaklardır.

Senaryolar

Senaryolar, kullanımda olan sistemin üst düzey bir performans sağlayan sistem çalışmasının tanımı, kullanıcı sınıfları ve istisnai durumların resmi olmayan açıklamalarıdır.

İşte pet shop POS sistemi için örnek bir senaryo.

Bir müşteri evcil hayvan dükkânına girer ve arabayı çeşitli ürünlerle doldurur.

Kontrol ederken, kasiyer müşterinin sadakat kartı olup olmadığını sorar.

Eğer öyleyse, kasiyer kartı okutarak müşterinin kimliğini doğrular.

Değilse, kasiyer bir tanesini yerinde tamamlamayı teklif eder.

Sadakat kartı faaliyetinden sonra, kasiyer bir kod okuyucu kullanarak ürünleri tarar.

Her ürün tarandıkça, satış toplanır ve envanter uygun şekilde güncellenir. Ürün taramasının tamamlanmasının ardından

bir ara toplam hesaplanır. Daha sonra herhangi bir kupon ve indirim girilir.

Yeni bir ara toplam hesaplanır ve geçerli vergiler eklenir. Bir makbuz yazdırılır ve müşteri nakit, kredi kartı, banka kartı veya çek ile ödeme yapar. Tüm uygun toplamlar (satışlar, vergiler, indirimler, indirimler vb.) hesaplanır ve kaydedilir.

Alan yeni olduğunda senaryolar oldukça kullanışlıdır (bir senaryo düşünün örneğin uzay istasyonu). Kullanıcı hikayeleri aslında bir tür senaryodur.

Görev Analizi

Halihazırda incelemiş olduğumuz hiyerarşik yönelimli tekniklerin çoğu gibi,görev analizi, sistem tarafından gerçekleştirilecek görevlerin işlevsel bir şekilde ayrıştırılmasını içerir. Yani, en yüksek soyutlama düzeyinden başlayarak, tasarımcı ve müşteriler daha ileri düzeyde ayrıntı ortaya çıkarır. Bu detaylı ayrıştırma, en düşük işlevsellik düzeyi (tek görev) elde edilinceye kadar devam eder.

Örnek olarak, evcil hayvan mağazası POS sistemi için kısmi görev analizini düşünün.

Şekil 3.7'de gösterilmiştir.

Burada, kapsamlı evcil hayvan mağazası POS sisteminin üç ana parçadan oluştuğu kabul edilmektedir.

görevler: envanter kontrolü, satış ve müşteri yönetimi. Altında sondaj satış fonksiyonları, bunların aşağıdaki görevlerden oluştuğunu görüyoruz: vergi fonksiyonları

ve satın alma işlemleri. Ardından satın alma işlemi fonksiyonuna geçilir,

bu görevleri satış, geri ödeme, indirim ve kupon görevlerine ayırıyoruz.

Görev analizi ve ayrıştırma, yeterli bir ayrıntı düzeyine ulaşılana kadar devam eder (tipik olarak, bir yöntem veya ayrıştırılamaz prosedür düzeyine kadar)

ve diyagram tamamlanır.

Use Case'ler*

Use caseler, daha karmaşık müşterilerin ve paydaşların ve onların isteklerinin tanımlaması için bir yoldur. Use caseler, sistem ve sistem arasındaki etkileşimleri, sistem etrafındaki ortam, özellikle insan kullanıcılar ve diğer sistemler arasındaki etkileşimleri gösterir. Saf yazılım veya hibrit yazılım donanımın davranışını modellemek için kullanılabilirler. Use caseler, tasarımcının bakış açısından sistemin çalışma senaryolarını açıklar.

Use caseler, tipik olarak, sistemin dış ortamıyla etkileşimlerini gösteren bir use case diyagramı kullanılarak temsil edilir. Use case diyagramında kutu sistemin kendisini temsil eder.

Çubuk figürler,sistem ile etkileşime giren dış varlıkları belirleyen "aktörleri" temsil eder.

Aktörler insanlar, diğer sistemler veya cihaz girdileri olabilir.

İç elipsler, aktörlerin her biri için her bir kullanım etkinliğini temsil eder (use caseler). Bu

düz çizgiler, oyuncuları her kullanımla ilişkilendirir. Şekil 3.8, bagaj kontrol sistemi için bir kullanım durumu diyagramını göstermektedir.

Üç kullanım gösterilmiştir: bagajın görüntüsünün alınması ("görüntü bagajı"),

bir güvenlik tehdidinin tespiti (bu durumda torba, çevrimdışı işleme için konveyörden reddedilir) ve ardından sistem mühendisi tarafından konfigürasyon.

Görüntüleme kamerasının, ürün sensörünün ve reddetme mekanizmasının insan benzeri bir çubuk figürle temsil edildiğine dikkat edin - bu tipik bir durumdur - çubuk figür bir

insan olsun ya da olmasın sistem "aktörü".

Ek B, Şekil B.4 başka bir örnek sağlar:

Her bir kullanım durumu, ön ve son koşulların yanı sıra incelenen sistemin çalışma senaryolarını ve istisnaları açıklayan bir belgeleme biçimidir.Yinelemeli bir geliştirme yaşam döngüsünde, bu kullanım durumları

analiz ve tasarım iş akışları ilerledikçe giderek daha rafine ve ayrıntılı hale gelir.

Daha sonra, her biri tarafından tanımlanan davranışları tanımlamak için etkileşim diyagramları oluşturulur.

İlk yinelemede, bu diyagramlar sistemi bir "kara kutu" olarak tasvir ediyor.

Ancak alan modellemesi tamamlandıktan sonra kara kutu,

daha sonra görüleceği gibi nesnelerin bir işbirliği.

İyi geliştirilmişse, bazen, use caseler bir kalıp dili oluşturmak için kullanılabilir ve bu kalıplar ve türetilen tasarım öğeleri, ilgili sistemlerde (Issa ve Al-Ali 2010) yeniden kullanılabilir.

Gereksinimleri belirtirken kalıpları kullanma

daha yüksek düzeyde tutarlılık sağlar ve belirli gereksinim özelliklerinin otomatik ölçümündeki hataları azaltabilir.

Gereksinim belirleme belgesinde gereksinimlerin belirlenmesinde ve gereksinimlerin modellenmesinde çok önemli bir araç haline geldikleri için,

Birçok örnekle birlikte kullanım durumlarının kapsamlı bir tartışması şurada bulunabilir:

Ek E.

Kullanıcı hikayeleri*

Kullanıcı hikayeleri, ilk keşif ve projelendirme gereksinimleri için kullanılan kısa konuşma metinleridir.

Kullanıcı hikayeleri çevik metodolojilerle birlikte yaygın olarak kullanılmaktadır.

Kullanıcı hikayeleri, sistemin onların ihtiyacına göre müşteriler tarafından yazılır.

Kullanıcı hikayeleri genellikle iki ila

üçe beş inçlik bir karta yazılmış dört cümleden oluşur. Yaklaşık 80 kullanıcı hikayesi genellikle

bir sistem artışı veya gelişimi için uygun, ancak kullanılacak metodoloji (örneğin, çevik ve artımlı) uygun sayı

uygulama boyutuna ve kapsamına ve geliştirmeye bağlı olarak büyük ölçüde değişecektir.

Pet shop POS sistemi için bir kullanıcı hikayesi örneği aşağıdaki gibidir:

- Her müşteri bir kasada kolayca kontrol edebilmelidir.

- Self servis desteklenecektir.
- Tüm kuponlar, indirimler ve geri ödemeler bu şekilde ele alınmalıdır.

Kullanıcı hikayeleri, yalnızca makul ölçüde düşük risk oluşturmak için yeterli ayrıntı sağlamalıdır.

hikayenin uygulanmasının ne kadar süreceğinin tahmini. zamanı geldiğinde

uygulamak için, hikaye geliştiricileri müşteriyle buluşacak ve

detaylar konuşulacaktır.

Kullanıcı hikayeleri ayrıca kabul testinin temelini oluşturur. Örneğin, bir veya daha fazla

kullanıcı hikayesinin verilip verilmediğini doğrulamak için otomatik kabul testleri oluşturulabilir.

Kullanıcı hikayeleri Bölüm 7'de ve Ek D'de daha ayrıntılı olarak tartışılmaktadır.

Bakış açıları

Bakış açıları, bilgileri farklı seçim bölgelerinden(bakış açılarından) organize etmenin bir yoludur.

Örneğin, bagaj taşıma sisteminde

Aşağıdaki paydaşların her biri için farklı sistemin perspektifleri:

- Bagaj taşıma personeli
- Gezginler
- Bakım mühendisleri
- Havalimanı yöneticileri
- Düzenleyici kurumlar

Bu paydaşların her birinin ihtiyaçlarını ve çelişkileri tanıyarak bu bakış açılarıyla ortaya çıkan çatışmalar, çeşitli yaklaşımlar kullanılarak uzlaştırılabilir.

Gerçek bakış açıları, işletmeden alınan çeşitli bilgileri içerir.

etki alanı, süreç modelleri, işlevsel gereksinim özellikleri, organizasyonel

modeller, vb.

Sommerville ve Sawyer (1997), aşağıdaki bileşenlerin olması gerektiğini önerdi:

her bakış açısında:

- Belirtimde kullanılan gösterimi tanımlayan bir temsil stili
- "Bakış açısının ele aldığı ilgi alanı" olarak tanımlanan bir alan
- Tanımlanmış tarzda ifade edilen bir sistem modeli olan bir spesifikasyon
- Sistemin nasıl oluşturulacağını ve kontrol edileceğini tanımlayan bir süreç modeline sahip bir iş planı.

Şartname

- Yapım, kontrol, denetim,

ve spesifikasyonun değiştirilmesi

Bakış açısı analizi genellikle önceliklendirme, anlaşma ve gereksinimleri sıralama için kullanılır.

Atölyeler

En genel düzeyde, çalıştaylar, gereksinimlerin sorunlarını çözmek için herhangi bir paydaş toplantısıdır.

Çalıştayları iki tür olarak ayırt edebiliriz: resmi

ve gayri resmi.

Resmi çalıştaylar iyi planlanmış toplantılardır ve genellikle "teslim edilebilir" etkinliklerdir.

sözleşme ile zorunlu kılınanlardır. Örneğin, DOD-MIL-STD-2167 dahil

birden fazla gerekli ve isteğe bağlı atölye çalışması (kritik incelemeler). İyi bir örnek

resmi atölye stili, JAD'de somutlaştırılmıştır.

Gayri resmi çalıştaylar genellikle yüksek düzeyde yapılandırılmış toplantılardan daha az sıkıcıdır. Ancak

gayri resmi toplantılar çok özensiz olma ve yanlış bir güvenlik duygusuna ve

kayıp bilgi eğilimindedir. Bir tür çalıştay gerekiyorsa,daha önce tartışılan başarılı toplantılar için parametreler kullanılarak bir tane yapılabilir.

İşlevsel Olmayan Gereksinimlerin Ortaya Çıkarılması*

İşlevsel olmayan gereksinim (NFR) ortaya çıkarma teknikleri, işlevsel gereksinimleri ortaya çıkarma tekniklerinden farklıdır.

NFR'ler genellikle gereksinim analizi sırasında ve yazılım geliştirme sürecinde gayri resmi olarak belirtilir, genellikle çelişkilidir ve uygulanması zordur.

. Borg ve ark. (2003), taşınan

farklı kuruluşlarda NFR ile ilgili sorunların köklerini belirlemeyi amaçlayan görüşmeler. Sonuç, NFR ile ilgili sorunların dörtte ortaya çıkmasıydı.

geliştirme sürecinin aşamaları: ortaya çıkarma, dokümantasyon, yönetim ve

Ölçek; Ortaya çıkarma aşamasında NFR ihmali tüm geliştirme süreci boyunca yayıldığından, ortaya çıkarma NFR ile ilgili olası sorunların ana kaynağı olarak işaretlenir. Örneğin, nedenler şunlardır:

(i) belirli kısıtlamalar gereksinimler aşamasında bilinmiyorsa,

(ii) NFR'ler birbiriyle çelişme eğilimindedir,

(iii) FR'leri ve NFR'leri ayırmak, aralarındaki bağımlılıkların izlenmesini zorlaştırır.

Bunlar, tüm gereksinimler birlikte karıştırılırsa işlevsel ve işlevsel olmayan hususları ayırmak zordur.

Ortaya çıkarmak, detaylandırmak ve belgelemek için bir yöntem seçmek kolay değildir.

Mevcut yöntemlerin çeşitliliği arasında NFR'ler. Aşağıdakiler arzu edilir

NFR ortaya çıkarma yöntemlerinin özellikleri (Herrmann ve ark. 2007):

1. Daha az deneyimli personel tarafından yöntem kullanımını kolaylaştırmak için rehberli bir süreç ve

sonuçların tekrarlanabilirliğini desteklemek için

2. Kalite güvencesini kolaylaştırmak için ölçülebilir NFR'lerin türetilmesi

3. Desteklemek için türetilmiş NFR'lerin eksiksizliğini desteklemek için eserlerin yeniden kullanımı

öğrenmek ve tekrar çalışmaktan kaçınmak

4. Gizli gereksinimleri de yakalamak ve böylece eksiksizliği desteklemek için kalitenin sezgisel ve yaratıcı şekilde ortaya çıkarılması

5. Etkin ortaya çıkarma için odaklanmış çaba ve desteklemek için NFR önceliklendirmesi

takas kararları

6. Takas kararlarını desteklemek için NFR'ler arasındaki bağımlılıkların ele alınması

7. NFR'lerin fonksiyonel gereksinimlerle entegrasyonu

NFR'leri keşfetmenin yaygın yolları, sistem niteliklerinin rekabetçi analizini içerir: NFR'ler, rekabet halindeki ürünlerin niteliklerini analiz ederek keşfedilebilir.

Örneğin, rakip bir ürün için tepki süresi nedir? Ve

daha iyisini mi yapmamız gerekiyor?

NFS'yi keşfetmek için başka bir teknik, önceden oluşturulmuş bir gereksinim mühendisinin paydaşlardan ve geliştirme ekibinden sorulacak bir anket kullanmaktır.

Örneğin: "Sistem aşağıdakilere nasıl yanıt vermeli?

giriş hataları? Sistemin hangi bölümleri daha sonra değiştirilmeye aday olabilir?

Sistemin hangi verileri güvenli olmalı?" Bu sorular, odaklanmak ve sormak için bir şablon veya standart (örneğin, ISO 9126) izlenerek hazırlanabilir.

Açıklama Özeti

Bu tur, birçok ortaya çıkarma tekniği içeriyor ve her birinin avantajları var

ve yol boyunca tartışılan dezavantajlar. Açıkçası, bunlardan bazıları

teknikler çok genel, bazıları çok spesifik, bazıları paydaşlara çok fazla güveniyor,bazıları yeterli değil, vb. Bu nedenle, gereksinimlerin ortaya çıkarılmasını başarılı bir şekilde ele almak için bazı teknik kombinasyonlarının gerekli olduğu açıktır.

Gereksinimlerin Hangi Çıkarma Teknikleri Kombinasyonu

Kullanılmalı?

Uygun bir gereksinimleri ortaya çıkarma tekniği seçimi konusunda rehberlik sağlayacak çok az araştırma var.

Dikkate değer bir istisna, bilgi temelli

gereksinim mühendisliği tekniklerinin seçimine yönelik kullanıcıları bir kitaplıktan bir gösterim kombinasyonu seçmeye yönlendiren(Eberlein

ve Jiang 2011). yaklaşım (KASRET).

Teknikler kütüphanesi ve atama algoritması, bir

endüstriyel ve akademik uzmanların literatür taraması ve anketini baz almıştır.

Buna rağmen henüz yaygın olarak kullanılmamıştır.

Uygun ortaya çıkarma teknikleri hakkında biraz rehberlik sağlamak için,

önce daha önce tartışılan teknikleri kategorilere veya eşdeğerliğe göre tekniklerin ortaya çıkaracağı bilgi türlerine dayalı sınıflara gruplandırın.

Sınıflar (röportajlar, alan odaklı, grup çalışması, etnografi, prototip oluşturma,hedefler, senaryolar, bakış açıları) ve dahil edilen ortaya çıkarma teknikleri tablo 3.2'de gösterilmiştir.

Şimdi çeşitli tekniklerin bunlarla başa çıkmada ne kadar etkili olduğunu özetleyebiliriz.

Tablo 3.3'te ortaya çıkarma sürecinin çeşitli yönleri gösterilmiştir (Zowghi ve Coulin 1998 çalışmasına dayalı olarak).

Örneğin, görüşmeye dayalı teknikler, gereksinimlerin ortaya çıkarılmasının tüm yönleri için faydalıdır (ancak çok zaman alıcıdır). Öte yandan, prototipleme

teknikleri, paydaşları analiz etmek ve gereksinimleri ortaya çıkarmak için en iyi şekilde kullanılır.

Etnografik teknikler, problem alanını anlamak, paydaşları analiz etmek, gereksinimleri talep etmek vb. için iyidir.

Son olarak, bu ortaya çıkarma teknikleri (kümeler) arasında bazılarının aynı şeyi başarması ve dolayısıyla birbirinin alternatifi olması bakımından açık bir örtüşme vardır.

Diğer durumlarda, bu teknikler birbirini tamamlar. Tablo 3.4'te alternatif

(A) ve tamamlayıcı (C) ortaya çıkarma grupları gösterilmiştir.

Uygun bir ortaya çıkarma teknikleri seti seçmenizde size rehberlik etmesi için Tablo 3.2 ila 3.4'ü kullanabilirsiniz. Kullanılacak bir dizi teknik seçerken, bir dizi tamamlayıcı teknik seçersiniz. Örneğin, bir kombinasyonun

bakış açısı analizi ve bir tür prototipleme istenebilir. Tersine,

hem bakış açısı analizini hem de senaryo oluşturmayı kullanmak, muhtemelen aşırı derecede gereksiz bilgi verecektir.

Örnek olarak,bir hastanedeki insanları izlemek için bir IoT sağlık sistemi örneğini düşünün.

Burada, sistemin genel hedeflerini ve istenen sonuçları tanımlamaya yönelik bir girişimle başlamak uygun olacaktır. Daha sonra, bir etki alanı analizi kullanılır.

Sistem için geçerli olan yasalar, düzenlemeler ve standartlar araştırılır. Kullanıcı hikayeleri, senaryolar ve röportajlar, gereksinimlerin ortaya çıkarılması sistemin kullanıcıları için uygun olacaktır. Şüphesiz, gereksinimleri keşfetme ve iyileştirme süreci boyunca bir miktar prototipleme olacaktır. Bu tekniklerin her biri gereksinim keşfi ilerledikçe gayri resmi olandan daha katı olana geçmelidir. Bu yaklaşım genellikle federal, eyalet ve belediye

yönetimleri için ve hatta büyük endüstriyel projeler için kullanılır.

Bununla birlikte, ortaya çıkarma tekniklerinin "gümüş kurşun" kombinasyonu yoktur. Doğru karışım, uygulama alanına, müşteri organizasyonunun kültürüne ve gereksinim mühendisinin kültürüne, projenin boyutuna ve birçok şeye bağlı olacaktır.

Bu konuda deneyimlerden ders almak çok önemlidir.

Gereksinimlerin Ortaya Çıkması Tekniklerinin Yaygınlığı

Bu tartışmayı bitirmeden önce, endüstride yaygın olarak kullanılan çeşitli çıkarımların nasıl olduğu hakkında bir fikir edinelim. Bunu yapmak için ankete geri dönüyoruz

yazılım profesyonelleri (Kassab ve ark. 2014).

"Hangi gereksinimleri ortaya çıkarma tekniğini/tekniklerini kullanıyorsunuz?" sorusuna verilen cevapların özeti Şekil 3.9.da gösterilir.

Yanıtlar, beyin fırtınası, görüşmeler ve prototip oluşturmanın

en sık kullanılan ortaya çıkarma teknikleri (her biri yaklaşık %10).

Bu bölümde tartışılan diğer teknikler nispeten seyrek kullanıldı.

Tehlikeleri Ortaya Çıkarmak

Daha önce "yapmamalı" davranışlarının, istenmeyen ve tehlikelerin neden olma eğiliminde olan davranışların bir alt kümesi olduğu ciddi veya yıkıcı arızalardan ortaya çıkan çıktı davranışları kümesi olduğunu belirtmiştik.

"Ciddi" ve "felaket" terimleri özneldir, ancak genellikle can kaybını, ciddi yaralanmayı, büyük altyapı hasarını,ya da büyük mali kaybı içerir.

Örneğin, evcil hayvan mağazası POS'u için bazı "yapılmaz" gereksinimleri şunları içerir:

- Sistem, müşteri bilgilerini harici sistemlere ifşa etmeyecektir.
- Sistem yetkisiz erişime izin vermeyecektir.
- Sistem, müşterilerin mağaza kredisini aşmasına izin vermeyecektir.

Bu örneklerde, ilk iki gereksinim tehlike olarak kabul edilebilir, çünkü şirkete mali zarar verme potansiyeli üçüncüsünden çok daha fazladır.

Tehlikeler, doğal olarak meydana gelen girdi anormalliklerinin (donanım arızaları gibi) veya yapay olarak meydana gelen (davetsiz girişlerden gelen saldırılar gibi) (Voas ve Laplante 2010)bir fonksiyonudur.

Bu anormal giriş olaylarının tanımlanması gerekir ve bunların ortaya çıkan arıza modları ve kritikliklerinin süreç boyunca belirlenmesi gerekir.

uygun bir "yapmamalı" seti geliştirmek için gereksinimlerin ortaya çıkarılması aşamasında diğer gerekliliklerde olduğu gibi, "yapmayacak" gerekliliklerine öncelik verilmesi gerekir.

Tehlike belirlemeye yönelik tipik teknikler arasında, kötüye kullanım durumlarının geleneksel olarak geliştirilmesi, antimodeling ve resmi yöntemler yer alır (Robinson 2010).

Sistemin veya ilgili sistemlerin önceki sürümlerinden oluşturulmuş istenmeyen davranışların kontrol listeleri de istenmeyen davranışların belirlenmesinde yardımcı olur. Geçerli standartlar ve düzenlemeler ayrıca belirli

"değil" gereksinimleri içerebilir, örneğin Amerika Birleşik Devletleri'nde Sağlık Sigortası.

Taşınabilirlik ve Sorumluluk Yasası (HIPPA 1996), belirli kişisel bilgilerin yetkisiz taraflara verilmesini ve standart yapıyı yasaklar, tüm yargı alanlarındaki kodlar, belirli inşaat uygulamalarına karşı çok sayıda yasak içerir.

Kötüye Kullanım Durumları

Kullanım senaryoları, istenen davranışın yapılandırılmış, kısa açıklamalarıdır.

İstenen davranışı tanımlayan vakalar, yanlış kullanım vakaları (veya kötüye kullanım vakaları) vardır.

Çoğu sistem için tipik istenmeyen davranışlar suistimaller, güvenlik ihlalleri ve diğer kötü niyetli davranışların yanı sıra eğitimsiz, yönünü şaşırmış veya beceriksiz kişilerce kötüye kullanım içerir. Cleland-Huang ve ark. (2016),tehdit modelleme ve beyin fırtınasına dayalı vakalarla kötüye kullanımı belirlemenin birkaç yolunu açıklar.

Kullanım senaryoları oluşturmanın kolay bir yolu, istenmeyen bir kişi rolünü üstlenmektir.Bu kişi sistemin istenmeyen bir kullanıcısıdır ve daha sonra böyle bir kişinin davranışlarını model alır.

(Cleland-Huang 2014). İstenmeyen kişiler, bilgisayar korsanlarını, davetsiz misafirleri, casusları, ve hatta iyi niyetli, ancak beceriksiz kullanıcıları temsil eder.

Bu kişilerin belirlenmesinin tehlike gereksinimleri oluşturmaya nasıl yardımcı olduğunu görmek için,

aşağıdaki örnekleri göz önünde bulundurun. Pet shop POS sisteminde, bir hacker'ın bu sisteme nasıl sızacağını düşünmek ve daha sonra hacker'ın niyetini bozacak gereksinimler yaratmak uygun olacaktır. Bagaj taşıma sisteminde,

bir gereksinim mühendisi, beceriksiz veya dalgın bir gezgin rolünü üstlenebilir ve ardından bu tür kişilerin güvenliğini sağlayacak gereksinimleri belirleyebilir.

Kötüye kullanım vakaları yaratma ihtiyacı, tüm olumsuz paydaşları tamamen tanımlamak için bir nedendir çünkü bu kişilerin çok sayıda iyi olmayan kişiyi içermesi muhtemeldir.

Antimodeller

İstenmeyen davranışı elde etmenin başka bir yolu, sistem için antimodeller yaratmaktır.

Antimodeller, hata ağaçlarıyla ilgilidir, yani model, bir

sistem hatasına yol açan istenmeyen davranışlar için neden ve sonuç hiyerarşisidir. Sistem arızasının nedenleri "yapmamalı" gerekliliklerini oluşturmak için kullanılır.

Örneğin, bagaj taşıma sisteminin güvenlik işlevini düşünün

Şekil 3.10'da gösterilen hasarlı bagajın istenmeyen sonucudur.

Şekil bizi aşağıdaki ham gereksinimleri yazmaya yönlendiriyor:

- Bagaj sıkışması algılanırsa konveyör hareket etmemelidir.
- Bagaj besleyici sıkışmışsa, konveyör hareket etmemelidir.
- Bagaj kapısı sıkışmışsa konveyör hareket etmemelidir.
- Konveyör sıkışmışsa, bagaj besleyici hareket etmemelidir.

Bu gereksinimlerin daha fazla analize ve muhtemelen basitleştirilmesine ihtiyacı vardır, ancak

antimodel, bu ham gereksinimleri sistematik bir şekilde elde etmemize yardımcı oldu.

Resmi Yöntemler

Bölüm 6'da tartışılacağı üzere, matematiksel formalizmler operasyonları ile ilgili olarak sistemin ve çevresinin bir modelini,amaçlarını, gereksinimlerini ve kısıtlamalarını oluşturmak için kullanılabilir. Bu formalizmler daha sonra otomatik model denetleyicileri birlikte sistemin çeşitli özelliklerini incelemek için ve istenmeyenlerin bulunmadığından emin olmak için kullanılabilir.

Örneğin, UML/SysML model davranışı kullanıyorsanız resmi etkinlik diyagramları güvenlik endişeleriyle açıklanabilir. Aktivite diyagramları Ek C, UML'de açıklanmıştır.

Standart, güvenlik gereksinimlerinin aşağıdakiler aracılığıyla izlenebilirlik için etiketlenmesini şart koşar:

Gereksinimlerdeki etkilerin ve değişikliklerin değerlendirilmesini sağlamak için ihtiyaç duyulan gereksinim mühendisliği yaşam döngüsü. Benzersiz olarak tanımlanmış tehlikeler,

gereksinimler belgesinde özel bir bölümde listelenmeli veya

gereksinimin yanında bir bayrak veya bir gereksinim yönetimi içinde etiketlenmelidir.

Yazılım güvenlik gereksinimleri, güvenli olmayan sistemlere karşı koruma sağlamaktan fazlasını yapabilir. Yazılım, sistemi izlemek, kritik analizleri yapmak için,verileri, eğilimleri arayan ve tehlikeli bir durumun habercisi olabilecek olaylar meydana geldiğinde sinyal verecek veya harekete geçecek proaktif olarak kullanılabilir

Böyle bir gösterge algılandığında, yazılım bu tehlikenin etkilerini önlemek veya azaltmak için kullanılabilir. Kaçınma veya hafifletme sistemin tamamen veya kısmen geri yüklenmesini veya sistemin güvenli bir duruma getirilmesini içerebilir.

Egzersizler

3.1 Akıllı ev sistemi için aşağıda açıklanan bazı farklı kullanıcı sınıfları nelerdir?

Ek A?

3.2 Yüz yüze etkileşim olmadan gereksinimler ortaya çıkarmaya çalışırken karşılaşılabilecek bazı zorluklar nelerdir?

3.3 Heisenberg belirsizlik ilkesi, aşağıdakiler dışındaki teknikler için de geçerli midir?

etnografik gözlem? Heisenberg belirsizlik ilkesini hafifletmenin bazı yolları nelerdir?

3.4 Etnografik gözlem sırasında zamanı ve yapılan gözlemin gününü kaydetmenin amacı nedir?

3.5 Gereksinimler gelecekteki ölçeklenebilirlik ve geliştirmeleri hesaba katmalı mı?

3.6 Bu bölümde açıklanan tekniklerin hangi alt kümesi, müşterilerin coğrafi olarak dağıldığı bir ortam için uygun olur?

3.7 “Aktif dinleme” kavramını araştırın. Bu teknik gereksinimlerin ortaya çıkarılmasında nasıl yardımcı olur?

3.8 Sistem gereksinimlerini ortaya çıkarmak için aşağıdaki paydaş gruplarından hangi ortaya çıkarma tekniklerini kullanırsınız?

3.8.1 Bagaj taşıma sistemindeki yolcular

3.8.2 Pet shop POS sistemindeki kasiyerler

3.8.3 Bölüm 2'nin sonunda açıklanan IoT sağlık sistemindeki doktorlar

3.9 Bir kurs projesi üzerinde çalışıyorsanız, uyguladığınız ortaya çıkarma tekniklerini listeleyin.

Paydaş gruplarının her birinden sistem gereksinimlerini ortaya çıkarmak için kullanır mıydınız?

3.10 Ek A'daki SRS'de birkaç "yapılmaması gereken" gereklilikler vardır.

varsa, bunları tehlike olarak kabul eder misiniz?

3.11 Türkiye'deki SRS'de neden “yapmamalı” gereklilikleri olmadığına dair spekülasyon yapın.

Ek B.

*3.12 Evcil hayvan mağazası satış noktası sistemi için, envanter kontrolüne ilişkin bir antimodel geliştirin. Örneğin, sistem negatif envanter kaydetmemelidir.

Bu antimodel için ilgili "yapmayacak" gereksinimlerini yazın