Gereksinim Anlaşması ve Analizi

Ortaya çıkarma faaliyeti boyunca ve özellikle sistem gereksinimleri spesifikasyonunu tamamlamadan önce, SRS ham gereksinimleri problemler için analiz edilmeli ve bu problemler uzlaştırılmalıdır. Gereksinim analizi, problemler için gereksinimleri analiz etme faaliyetidir. Sorunlu gereksinimler şunları içerir:

- Kafa karıştırıcı
- Yabancı
- Yinelenen
- Çakışan
- Eksik

Bu kavramları Bölüm 5'te daha katı hale getiriyoruz.

Gereksinim analizi genellikle gayri resmidir, örneğin, müşterilere ve paydaş temsilcilerine, gereksinim mühendisinin tüm müşterilerin isteklerine ilişkin yorumunun doğru olduğundan emin olmak için anlayabilecekleri bir biçimde ortaya çıkan gereksinimler gösterilir. Resmi yöntemler, gereksinim analizi için daha titiz teknikler sağlar.

Gereksinim anlaşması, farklı kaynaklardan elde edilen aynı gereksinimdeki farklılıkların uzlaştırılması sürecidir. Gereksinim anlaşması genellikle gayri resmi bir süreçtir, ancak sistematik bir süreç kullanılmalıdır. Kullanım senaryoları ve kullanıcı hikayeleri genellikle bu amaç için etkili araçlardır (Hull ve diğerleri 2011). Ekran maketleri, özellikle işlevsel prototiplerle birlikte kullanıldığında başka bir yararlı tekniktir (Ricca ve ark. 2010).

Diğerleri senaryolar, storyboard'lar, kağıt kullanmayı önerir.

prototipler, kodlanmış kavram üreteçleri ve gereksinim anlaşması için işlevsel prototipler (Sutcliffe et al. 2011).

Müşteri sözleşmesi sürecinin bir parçası olarak, Hull ve ark. (2011) aşağıdaki soruları sormanızı önerir:

- Gereksinim tamamlandı mı?
- Gereklilik açık mı?
- Gereklilik uygulanabilir mi?
- Yeterlilik planı açık ve kabul edilebilir mi?

Bölüm 5'te bu soruların bir gereksinim belirtiminin istenen nitelikleriyle nasıl ilişkili olduğunu not edeceksiniz ve Bölüm 5 ve 6'da gereksinim mühendisliği yaşam döngüsü boyunca bu nitelikleri elde etmenin daha kesin yollarını inceleyeceğiz.

Gereksinim Gösterimi

Kullanım durumları, kullanıcı öyküleri, doğal diller, biçimsel diller, stilize edilmiş (kısıtlanmış doğal) diller ve çeşitli biçimsel, yarı biçimsel ve biçimsel olmayan diyagramlar dahil olmak üzere, herhangi bir sistemdeki işlevselliği tanımlamak için çeşitli teknikler kullanılabilir. Eberlein ve Jiang (2011), spesifikasyon yaklaşımları dahil olmak üzere çeşitli gereksinim tekniklerinin seçilmesine yönelik odaklanmış bir tartışma sunar. Bu tür

seçimleri yaparken aşağıdaki faktörleri göz önünde bulundurmayı önerirler:

- Ele alınan tekniğin olgunluğu ve etkinliği gibi teknik konular
- Tekniklerin karmaşıklığı
- Organizasyonun değişime karşı çıkması gibi sosyal ve kültürel konular
- Geliştiricilerin eğitim ve bilgi düzeyi

Zaman ve maliyet kısıtlamaları, projenin karmaşıklığı ve yazılım ekibinin yapısı ve yetkinliği gibi yazılım projesinin belirli özellikleri.

Tek bir proje veya kuruluş çapında kullanım için doğru temsil stilini seçmek önemli bir zorluktur ve bir anlamda bu kitabın bir temasıdır.

Gereksinimlerin Temsiline Yaklaşımlar

Genel olarak, gereksinim gösteriminin üç biçimi vardır: resmi, resmi olmayan ve yarı resmi. Gereksinim belirtimleri, bu yaklaşımlardan birine veya diğerine sıkı sıkıya bağlı kalabilir, ancak genellikle bu yaklaşımlardan en az ikisinin (resmi olmayan ve bir diğeri) unsurlarını içerirler. Resmi temsiller titiz,

Matematiksel temel ve bunları Bölüm 6'da daha ayrıntılı olarak inceleyeceğiz. Ancak resmi gereksinim belirtimleri belgelerinde bile resmi olmayan veya yarı resmi belirtim unsurları olacaktır.

> Gayri resmi gereksinim temsil teknikleri, tam olarak katı bir matematiksel gösterime dönüştürülemez. Resmi olmayan

teknikler arasında doğal dil (yani insan dilleri), akış şemaları, özel diyagramlar ve (SRS) içinde görmeye alışık olabileceğiniz öğelerin çoğu yer alır. Aslında, tüm SRS belgelerinin bazı gayri resmi unsurları olacaktır. Bu gerçeği güvenle söyleyebiliriz, çünkü en resmi gereksinim belirtim belgeleri, yalnızca resmi bir öğeyi tanıtmak için olsa bile, doğal dili kullanmak zorundadır. Bu nedenle, dikkatimizi daha çok resmi olmayan teknikler kullanarak gereksinimlerin temsiline odaklayacağız.

Son olarak, yarı-biçimsel temsil teknikleri, gayri resmi görünse de en azından kısmi bir biçimsel temele sahip olanları içerir. Örneğin, birleşik modelleme dili (UML) veya sistem modelleme dili (SysML) metamodelleme dilleri ailesindeki,

kullanım senaryosu diyagramı da dahil olmak üzere, diyagramların çoğu gayri resmidir veya resmileştirilebilir. UML ve SysML genellikle yarı resmi modelleme teknikleri olarak kabul edilir; ancak uygun mekanizmaların eklenmesiyle tamamen resmi hale getirilebilirler (Laplante 2006).

Resmi gereksinim modellemesi Bölüm 6'da tartışılmaktadır. Endüstride kullanılan baskın yaklaşım ve en problemli olanı olduğundan, bu bölümün büyük bir kısmı doğal dili kullanan resmi olmayan modellemeye odaklanmaktadır.

Kasab et al. (2014), ankete katılanların %60'ının gereksinimlerin doğal dil cinsinden, yani gayri resmi olarak ifade edildiğini bildirdiğini saptamıştır (Şekil 4.1).

Bildirilen yarı biçimsel yaklaşımlar, UML/SysML diyagramlarını içeriyordu. Z ve

VDM dahil olmak üzere rapor edilen resmi yaklaşımlar Bölüm 6'da tartışılmaktadır.

Ankete katılanların çoğunluğu endüstride resmi olmayan şartnamelerle karşılaştıklarını veya kullandıklarını belirtirken, muhtemelen en resmi olmayan şartname belgelerinin bile bazı yarı resmi ve resmi unsurlara sahip olması muhtemeldir. Aynı şekilde, en resmi spesifikasyonlar bile bazı gayri resmi unsurlara sahip olacaktır.

ISO/IEC/IEEE Standardı 29148

2011 yılında, üç uluslararası standardizasyon kuruluşu, Uluslararası Standartlar Organizasyonu (ISO), Uluslararası Elektroteknik Komisyonu (IEC) ve Elektrik ve Elektronik Mühendisleri Enstitüsü (IEEE) birlikte ISO/IEC/IEEE Standardı 29148'i (2011) yayınladı.

Sistemler ve Yazılım Mühendisliği—Yaşam Döngüsü Süreçleri—Gereksinimler Mühendisliği için bu standart, birkaç farklı standardın uyumlu hale getirilmesinin bir sonucudur. Bu standartlardan biri olan IEEE Standard 830–1998, Yazılım Gereksinimleri Spesifikasyonları için Önerilen Uygulama, yazılım gereksinimlerinin belirtilmesi için önerilen yaklaşımları tanımlamıştır. Bu metnin önceki baskılarında IEEE Standard 830'a sıklıkla atıfta bulunulmuştur. Standart 29148, tam sistem yaşam döngüsünü dahil etmek, çevik ve yalın mühendislikteki gelişmeleri tanımak ve genel bir yazılım durumu olarak yazılım yoğun sistemlere odaklanmak için IEEE 830'da genişletildi. Kolaylık olması açısından, bundan sonra ISO/IEC/IEEE 29148, IEEE 29148 olarak anılacaktır.

IEEE 29148, aşağıdakilere yardımcı olan bir belge üreten bir modeli temel alır:

- Ürünün ne yapacağı konusunda alıcılar veya tedarikçiler arasında anlaşmanın temelini oluşturmak
- Tasarım başlamadan önce gereksinimlerin titiz bir şekilde değerlendirilmesini sağlamak ve daha sonraki yeniden tasarımı azaltmak
- Ürün maliyetlerini ve risklerini tahmin etmek için gerçekçi bir temel sağlamak ve kuruluşların doğrulama ve doğrulama planları geliştirmesini sağlamak

- Ürün geliştirme için bir temel sağlamak için (IEEE 29148 2011)

IEEE 29148 yönergelerine uymanın çeşitli faydaları vardır. İlk olarak, gereksinimler belgesinin organizasyonu için basit bir çerçeve sağlarlar. Ayrıca, IEEE 29148 taslağı, geniş bir uygulama alanı yelpazesinde yaygın olarak dağıtıldığından, gereksinim mühendisi için özellikle faydalıdır. (Şekil 4.2).

Son olarak ve belki daha da önemlisi, IEEE 29148, SRS'nin işlevsel ve işlevsel olmayan gereksinimlerini organize etmek için rehberlik sağlar.

Standart ayrıca Bölüm 3, Özel Gereksinimler altında işlevsel ve işlevsel olmayan gereksinimleri temsil etmenin yollarını açıklar. Bu gereksinimler bir sonraki tartışma konusudur.

İşlevsel Olmayan Gereksinimlerin Temsil Edilmesine İlişkin Öneriler

IEEE 29148, aşağıdakileri içeren çeşitli işlevsel olmayan gereksinimleri açıklar:

- Dış arayüzler
- Kullanılabilirlik
- Performans
- Mantıksal veritabanı konuları
- Tasarım kısıtlamaları
- Standartlara uygunluk
- Yazılım sistem özniteliği gereksinimleri

Harici arayüz gereksinimleri, aşağıdakiler dahil olmak üzere çeşitli şekillerde organize edilebilir:

- Öğenin adı
- Amacın tanımı
- Girdi kaynağı veya çıktı hedefi
- Geçerli aralık, doğruluk ve/veya tolerans

- önemsiz
- Zamanlama
- Diğer girdi/çıktılarla ilişkiler
- Ekran biçimleri/organizasyonu
- Pencere biçimleri/organizasyonu
- Veri biçimleri
- Komut biçimleri

Kullanılabilirlik gereksinimleri, kullanıcı ihtiyaçlarını karşılayan gereksinimleri

sağlar. Müşteri memnuniyeti açısından, kullanılabilirlik genellikle en önemli gereksinim sınıfıdır. Bu tür gereksinimlerin yeni sistemlerde veya yeni özelliklerle ilgili olarak keşfedilmesi zordur. Prototipleme, bu tür gereksinimleri keşfetmede yardımcı olabilir. Performans gereksinimleri, yazılıma veya bir bütün olarak yazılımla insan etkileşimine ilişkin statik ve dinamik gereksinimlerdir. Tipik performans gereksinimleri, desteklenecek eşzamanlı kullanıcı sayısını, işlem ve görevlerin sayısını ve içinde işlenecek veri miktarını içerebilir.

hem normal hem de en yoğun iş yükü koşulları için belirli zaman dilimleri.

Mantıksal veritabanı gereksinimleri, aşağıdakiler gibi çeşitli işlevler tarafından kullanılan bilgi türleridir:

- Kullanım sıklığı
- Erişim yetenekleri
- Veri varlıkları ve ilişkileri
- Bütünlük kısıtlamaları
- Veri saklama gereksinimleri

Tasarım kısıtlaması gereksinimleri, standartlara uygunluk ve donanım sınırlamaları ile ilgilidir. Kısa bir süre sonra gereksinim ifadelerinde kısıtlamaların nasıl organize edileceğini göreceğiz.

Standartlara uygunluk, geçerli yasalardan, standartlardan veya düzenlemelerden türetilen gereksinimleri belirtir. Ortaya çıkarma sürecinin başlarında bu belgelerin doğru ve eksiksiz tanımlanması çok önemlidir.

Son olarak, yazılım veya sistem öznitelikleri genellikle güvenilirlik, kullanılabilirlik, güvenlik, bakım kolaylığı, taşınabilirlik ve diğer birçok özelliği içerebilir..

İşlevsel Gereksinimlerin Temsil Edilmesine İlişkin Öneriler

İşlevsel gereksinimler, duruma göre açıklamayı veya diğer genel açıklama biçimlerini kullanabilir.

IEEE 29148, işlevsel gereksinimler için bir dizi organizasyon seçeneği sunar. İşlevsel gereksinimler şu şekilde organize edilebilir:

- Sistem modu (ör. navigasyon, savaş, teşhis)

- Kullanıcı sınıfı (ör. kullanıcı, süpervizör, teşhis)
- Nesne (sınıfları/nesneleri, öznitelikleri, işlevleri/yöntemleri ve mesajları tanımlayarak)
- Özellik (sistemin kullanıcıya ne sağladığını açıklar)
- Uyaran (ör. sensör 1, sensör 2, aktüatör 1, …)
- İşlevsel hiyerarşi (örneğin, görevlerin yukarıdan aşağıya ayrıştırılması)

Bu tekniklerin bir kombinasyonu tek bir SRS belgesinde kullanılabilir. Örneğin, Ek A'da verilen akıllı ev açıklaması bir işlevsellik odaklı açıklama. Özellik odaklı olduğuna dair bir ipucu, gereksinimlerin çoğu için “sistem gerekir” mantrasıdır. Ek

B'deki ıslak kuyu pompa sistemi spesifikasyonu da büyük ölçüde işlevsellik ile açıklanmaktadır.

Alternatif olarak, işlevsel mod tarafından düzenlenen bir sistemi düşünün - NASA WIRE (geniş alan kızılötesi gezgin) sistemi (1996). Bu sistem, Small Explorer programının bir parçasıydı ve standartlaştırılmış uzaydan yere iletişimi içeren bir milimetre-altı dalga astronomi uydusunu tanımlıyor. Uzay Veri Sistemleri Danışma Komitesi (CCSDS) ve Goddard Uzay Uçuş Merkezi standartlarına (NASA WIRE 1996) göre bilgiler yere iletilecek ve yerden alınan komutlar olacaktı.

Uçuş yazılımı gereksinimleri aşağıdaki gibi düzenlenmiştir:

- Sistem yönetimi

- Komut yönetimi
- Telemetri yönetimi
- Yük yönetimi
- Sağlık ve güvenlik yönetimi
- Yazılım yönetimi
- Performans gereksinimleri

Belgeyi sistem yönetimi modu altında incelediğimizde, aşağıdaki şekilde açıklanan işletim sistemi işlevselliğini görüyoruz:

İşletim sistemi

Bu sistem, gerekli olan ortak bir mekanizma seti sağlayacaktır.

çoklu görev desteği, CPU zamanlaması, temel iletişim ve bellek yönetimi gibi

gerçek zamanlı sistemleri desteklemek için.

01.1 İşletim sistemi çoklu görev yetenekleri sağlamalıdır

02İşletim sistemi, olaya dayalı, önceliğe dayalı zamanlama sağlamalıdır.

03001.2.1 Görevin yürütülmesi, bir görevin önceliği ve yürütülmesi için gereken kaynakların mevcudiyeti ile kontrol edilmelidir.

04

0501.3 İşletim sistemi, görevler arası iletişim ve senkronizasyon için destek sağlamalıdır.

0601.4 İşletim sistemi, gerçek zamanlı saat desteği sağlayacaktır.

0701.5 İşletim sistemi yazılımı, 80387 matematik yardımcı işlemcisi için görev düzeyinde bağlam geçişi sağlamalıdır.

08 Ayrıca sistem işlevselliği altında, aşağıdaki gibi tanımlanan komut doğrulama vardır:
09
10 Komut Doğrulama
11 1e21fligh Tht yazılımı, CCSDS komut yapısı doğrulamasını gerçekleştirecektir.
12 211.1 Uçuş yazılımı, CCSDS transfer çerçevelerinin doğru ve sırayla alındığını doğrulamak için 1 numaralı CCSDS Komuta Operasyon Prosedürü (COP-1) uygulayacaktır.
13 211.2 Uçuş yazılımı, 127'lik bir sabit boyutlu çerçeve kabul ve raporlama mekanizması (ÇİFTLİK) sürgülü penceresini ve 63'lük bir sabit FARM negatif kenarını destekleyecektir.
14 211.3 Uçuş yazılımı, aşağıdaki hatalardan herhangi biri meydana gelirse durumu telemetre edecek ve

gerçek zamanlı komut paketini atacaktır:

15 Sağlama toplamı, yayınlanmadan önce doğrulamada başarısız oluyor Geçersiz bir uzunluk algılandı

16 Geçersiz bir uygulama kimliği algılandı

17 211.4 Uçuş yazılımı, bir komut bağlantısı kontrol sözcüğü (CLCW) oluşturacak ve sürdürecektir. CLCW'ye her güncelleme yapıldığında, bir CLCW paketi biçimlendirilir ve olası aşağı bağlantı için yönlendirilir (NASA WIRE 1996).

18 Ayrıca, sistem davranışını tanımlamak için nesne yönelimli bir yaklaşım benimsemek yazılım tabanlı sistemlerde çok yaygındır. Bu, özellikle yazılımın Java gibi saf nesne yönelimli bir dil kullanılarak oluşturulmasının beklendiği durumlarda geçerlidir.

19 Nesne yönelimli temsiller, nesneler olarak adlandırılan oldukça soyut sistem bileşenlerini ve bunların kapsüllenmiş niteliklerini ve davranışlarını içerir.

20 Nesne yönelimli bir tarzda düzenlenen gereksinimlerle uğraşırken, kullanıcı hikayelerini (özellikle Bölüm 7'de tartışacağımız çevik yazılım geliştirme metodolojileri ile bağlantılı olarak) kullanmak veya davranışları tanımlamak için vakaları ve vaka diyagramlarını kullanmak çok tipiktir. .

Tablo 4.1 Nesne Yönelimli ve Yapılandırılmış Temsil

Yapılandırılmış	Nesne Yönelimli
Sistem bileşenleri	Fonksiyonlar	Nesneler

Veri ve kontrol belirtimi Dahili ayrıştırma yoluyla ayrılmış Nesneler içinde kapsüllenmiş

Özellikler Hiyerarşik yapı Kalıtımnesnelerin ilişkileri

Sistemin işlevsel açıklaması Sistemin davranışsal açıklaması

Bilginin işlevler içinde kapsüllenmesi Bilginin nesneler içinde kapsüllenmesi

UML/SysML

UML, yazılım ve sistem mühendisliği gereksinimlerinin belirlenmesinde, tasarımında, sistem analizinde ve daha fazlasında kullanılan bir dizi modelleme notasyonudur. SysML, sırasıyla genel sistemlere ve iş süreçlerine odaklanan

UML'nin bir varyasyonudur. UML ve SysML oldukça benzerdir. Okuyucuya kolaylık olması için, UML standardına önemli bir katkıda bulunan James Rumbaugh tarafından UML üzerine bir başlangıç, Ek C'de bulunmaktadır.

Kullanım senaryoları, UML ve SysML'de bulunan modellerden biridir ve gereksinim mühendisleri için çok önemlidir. Bu nedenle, kullanım durumlarının kapsamlı bir tartışması Ek E'de bulunmaktadır.

Gereksinimler Belgesi

Sistem (veya yazılım) gereksinimleri spesifikasyon belgesi, sistem geliştiricilerinden nelerin istendiğinin resmi beyanıdır. Hay (2003), SRS'yi büyük bir operanın notasına benzetiyor - bir nota

olmadan kaos var, ancak bitmiş not, operadaki çeşitli sanatçıların etkinliklerinin ve katkılarının koordine edilmesini sağlıyor.

Ayrıca, sponsor ve sistemin kurucuları arasında bir müşteri-satıcı ilişkisinin olduğu durumlarda, SRS'nin bir sözleşme olduğunu ve bu nedenle medeni sözleşme hukuku (veya belirli türlerde ceza hukuku) kapsamında uygulanabilir olduğunu hatırlamak da önemlidir. dolandırıcılık veya ihmal kanıtlanabilir).

Bir Gereksinim Belgesinin Kullanıcıları

Gereksinimler belgesinin her biri benzersiz bir bakış açısına, ihtiyaçlara ve endişelere sahip birkaç kullanıcısı vardır. Belgenin tipik kullanıcıları şunlardır:

- Müşteriler
- Yöneticiler
- Geliştiriciler
- Test mühendisleri
- Bakım mühendisleri
- Paydaşlar

Müşteriler gereksinimleri belirler ve gereksinimleri karşıladıklarından emin olmak için bunları gözden geçirmeleri

gerekir. Müşteriler ayrıca gereksinimlerdeki değişiklikleri de belirtir. Müşteriler dahil olduğundan ve muhtemelen mühendis olmadıklarından, SRS belgelerine meslekten olmayan kişiler tarafından erişilebilir olmalıdır (resmi metodolojiler hariç).

Tüm seviyelerdeki yöneticiler, sistem için bir teklif planlamak ve sistem geliştirme sürecini yönetmeyi planlamak için gereksinimler belgesini kullanacaklardır. Bu nedenle yöneticiler, SRS'de güçlü maliyet ve tamamlanma süresi göstergeleri arıyorlar.

Geliştiriciler, hangi sistemin geliştirileceğini anlamak için gereksinim belirtimi belgesini kullanır. Aynı zamanda, test mühendisleri sistem için doğrulama testleri geliştirmek için gereksinimleri kullanır. Daha sonra bakım mühendisleri,

sistemin yükseltilmesi veya sabitlenmesi için sistemi ve parçaları arasındaki ilişkiyi anlamaya yardımcı olmak için gereksinimleri kullanacaklardır. SRS'yi kullanacak diğer paydaşlar, söz konusu sistemin doğrudan ve dolaylı tüm lehtarlarını (veya muhaliflerini) ve ayrıca söz konusu sistemi kullanacak avukatlar, hakimler, davacılar, jüriler, bölge savcıları, hakemler, arabulucular vb. anlaşmazlık durumunda SRS'yi yasal bir belge olarak kullanırlar.

Gereksinimler Belge Belirtimleri

SRS için doğru format nedir? Birçok olası biçim vardır. Örneğin, IEEE 29148 standardı, bir gereksinim belirtim belgesi

için genel bir biçim sağlar. Proje Yönetim Enstitüsü (PMI) gibi diğer standart kuruluşları ve profesyonel kuruluşlar, standart gereksinim formatlarına ve şablonlarına sahiptir. Çoğu gereksinim yönetimi yazılımı, özelleştirilebilir biçimlerde belgeler üretecektir. Ancak "doğru" format, sponsorun, müşterinin, işverenin ve diğer paydaşların neye ihtiyaç duyduğuna bağlıdır.

SRS belgesinin değiştirilmesinin kolay olması gerektiği, şimdiye kadar tartıştığımız birçok nedenden dolayı açıktır. Ayrıca, SRS belgesi bakım için bir referans aracı olarak hizmet ettiğinden, sistemin yaşam döngüsü hakkında öngörüleri, yani değişiklikleri tahmin etmek için kaydetmelidir.

Genel organizasyon, yazma yaklaşımı ve söylem açısından en iyi uygulamalar şunları içerir:

- Spesifikasyon boyunca tutarlı modelleme yaklaşımları ve teknikleri kullanmak, örneğin yukarıdan aşağıya ayrıştırma, yapılandırılmış veya nesne yönelimli yaklaşımlar
- Operasyonel özellikleri tanımlayıcı davranıştan ayırma
- Modeller içinde tutarlı soyutlama düzeyleri ve modeller arasında ayrıntılandırma düzeyleri arasında uygunluk kullanma
- İşlevsel olmayan gereksinimlerin spesifikasyon modellerinin bir parçası

olarak modellenmesi – özellikle zamanlama özellikleri

- Spesifikasyonda donanım ve yazılım atamalarının atlanması (tasarımın spesifikasyondan ziyade başka bir yönü)

Bu kurallara uymak her zaman daha iyi bir SRS belgesine yol açacaktır. Son olarak, IEEE 29148, gereksinim belirtim belgeleri için istenen belirli nitelikleri tanımlar ve bunlar Bölüm 5'te tartışılacaktır.

Tercih Edilen Yazı Stili

Mühendisler (her türden), zayıf iletişim becerileri, özellikle de yazma konusunda haksız bir itibar kazanmıştır. Her durumda, gereksinim belgelerinin çok iyi yazılması gerektiği artık açık olmalıdır.

Teknik yazı üzerine her gereksinim mühendisi tarafından incelenmesi gereken mükemmel metinler vardır, örneğin, Laplante (2011). Pratik yaparak ve onlardan bir şeyler öğrenmek için iyi yazılmış SRS belgelerini inceleyerek yazınızı geliştirmeye teşvik ediliyorsunuz. Ayrıca edebiyat, şiir ve haber okumalısınız, bu yazılardan ekonomi ve sunum netliği hakkında çok şey öğrenilebilir. Hatta bazıları senaryo yazımı, gereksinim belgeleri (veya kullanıcı hikayeleri ve kullanım durumları) yazmak için uygun bir paradigma olarak önerdi (Norden 2007).

Her durumda, gereksinim belgesine herhangi bir yazı gibi yaklaşın - tekrar tekrar yazmaya ve yeniden yazmaya hazır olun.

Metin Yapısı Analizi

Metrikler, temel yazma özelliklerini denetlemede yardımcı olabilir. Çoğu kelime işlem aracı ortalama kelime, cümle ve paragraf uzunluğunu hesaplar ve bunlar çok basit veya anlaşılması çok zor olan yazıları vurgulayabildikleri için toplanması değerlidir. Bununla birlikte, standart kelime işlemciler tarafından hesaplanmayan yazının önemli bir özelliği, numaralandırma yapısının derinliğidir.

Numaralandırma yapısı derinliği, kaynak belgenin her seviyesindeki numaralandırılmış ifadeleri sayan bir ölçüdür. Örneğin, 1.0, 2.0, 3.0 vb. numaralandırılmış birinci düzey gereksinimlerin çok yüksek düzeyde olması beklenir.

(soyut) gereksinimler. 1.1, 1.2, 1.3, … numaralı ikinci seviye gereksinimler

2.1, 2.2, 2.3, … 3.1, 3.2, vb. daha düşük seviyedeki ikincil gereksinimlerdir.

detay. Daha da ayrıntılı gereksinimler üçüncü düzeyde bulunacaktır.

1.1.1, 1.1.2 vb. şeklinde sıralanmıştır. Bir belirtim, dördüncü ve hatta beşinci düzey gereksinimlere kadar devam edebilir, ancak normal olarak, üçüncü veya dördüncü ayrıntı düzeyleri yeterli olmalıdır. Her durumda, 1, 2, 3 vb. düzeylerdeki gereksinimlerin sayısı, belgenin organizasyonu, tutarlılığı ve ayrıntı düzeyi hakkında bir gösterge sağlar.

İyi organize edilmiş bir SRS belgesi tutarlı bir ayrıntı düzeyine sahip olmalıdır ve her düzeydeki gereksinimleri listeleyecekseniz, ortaya

çıkan şekil bir piramit gibi görünmelidir, çünkü 1. düzeyde ve her birinin altında birkaç numaralı ifade olmalıdır. seviye, üstündeki seviyeden daha fazla numaralı ifadelere sahip olmalıdır (Şekil 4.3a).

Öte yandan, gereksinimleri kum saati şeklini andıran (Şekil 4.3b) her düzeyde önemli olan gereksinim belgeleri, genellikle büyük miktarda tanıtıcı ve idari bilgi içeren belgelerdir. Son olarak, bir piramit tarafından temsil edilen baklava biçimli belgeler, ardından daha düşük seviyelerde azalan ifade sayıları (Şekil 4.3c), tutarsız bir ayrıntı temsil düzeyine işaret etmektedir (Hammer ve diğerleri, 1998).

Bölüm 5'te tanıtılan NASA ARM aracı, belirli bir SRS belgesi için metin yapısını uygun bir biçimde hesaplar.

Gereksinim Biçimi

Her gereksinim, dilin kullanımında açık, özlü ve tutarlı bir biçimde olmalıdır. Tutarlı bir dil kullanmak, gereksinim belgelerinin kullanıcılarına yardımcı olur (Hull ve diğerleri, 2011) ve SRS belgesinin yazılım araçlarıyla analizini çok daha kolay hale getirir.

Basitleştirilmiş bir standart gereksinim formu:

- [isim tamlaması] [fiil tamlaması] olacaktır (değil)

[isim tamlaması] gereksinimin ana konusunu tanımlıyorsa, gerekir (veya

yapılmaz) ifadesi gerekli veya yasaklanmış bir davranışı belirtir ve [fiil ifadesi] gereksinimin eylemlerini belirtir.

Bagaj taşıma sistemi için şu iki gereksinimi göz önünde bulundurun:

- Sistem, etiketlenmemiş bagajı reddedecektir.
- Sistem hasarlı bagajı reddetmemelidir

"Bagaj", "reddet", "etiketlenmiş" ve "hasar" terimlerinin SRS belgesinde bir yerde tanımlandığını varsayıyoruz.

Gereksinimlerin standart olmayan biçimde yazılması nadir değildir. Örneğin, gösterilen ilk gereksinim şu şekilde yeniden yazılabilir:

- Etiketlenmemiş bagajlar sistem tarafından reddedilecektir.

    Bu gereksinimin her iki versiyonu da eşdeğerdir, ancak gereksinimlerin standart biçimde temsil edilmesi, daha önce belirtilen nedenlerden dolayı arzu edilir.

Ayrıca, gereksinimlerdeki makalenin çıkarılması nadir görülen bir durum değildir, bu nedenle iki örnek gereksinim şöyle olacaktır:

- Sistem, etiketlenmemiş bagajı reddedecektir
- Sistem hasarlı bagajı reddetmemelidir

Bu gereksinimlerin formülasyonunda yanlış bir şey yoktur.

Referans kolaylığı için gereksinimleri bir şekilde, genellikle hiyerarşik

numaralandırma ile belirlemek önemlidir. Bu nedenle, bir gereksinim için standart form aşağıdakileri kapsar:

- [tanımlayıcı] [isim tamlaması] [fiil tamlaması] olacaktır (değil)

Önceki iki gereksinime dönersek ve bazı tanımlayıcı numaralar icat ederek, aşağıdaki iki örneğe sahibiz:

1.2.1 Sistem, etiketlenmemiş bagajı reddedecektir.

1.2.2 Sistem, hasarlı bagajı reddetmeyecektir.

Son olarak, standart gereksinim formunu aşağıdakiler için zenginleştirerek, mümkün olduğunda işlevsel gereksinimler için

performansa ölçülebilir kısıtlamalar koymak arzu edilir:

■ [tanımlayıcı] [isim tamlaması] [fiil tamlaması] [kısıtlama ifadesi] olacaktır (değil)

• Nitelik: Bir tarayıcı biriminin bir bagaj parçasını taramak için ihtiyaç duyduğu ortalama süre.

• Ölçek: Saniye (tür: oran ölçeği).

• Prosedür: Bir paketi yasaklı içerikler için taramak için gereken süreyi ölçün: çeşitli türlerin ortalama 1.000'den fazla parçasını alın.

• Planlanan değer: Referans değerinden %50 daha az.

• Kabul edilebilir en düşük değer: Referans değerinden %30 daha az.

• Referans değeri: Rakip veya benzer ürünlerin bir parça bagajı taramak için ihtiyaç duyduğu ortalama süre.

Ölçülebilir bir hedef içeren bir gereksinime örnek olarak bagaj taşıma sistemine dönelim. Aşağıdaki gereksinimi sağlamak üzere gerçekçi bir kısıtlama dahil edilebilir:

1.2.1 Sistem, tanımlandıktan sonra 5 saniye içinde etiketlenmemiş bagajı reddedecektir.

Bir gereksinimin ölçülebilir hedeflerinde farklılıklar ve toleranslar olabilir. Örneğin, bagaj taşıma sistemindeki varsayımsal bir

gereklilik 3.4.2'yi ele alalım: 3.4.2 Her bir bagaj tarayıcı birimi, dakikada ortalama 10 adet bagaj işleyecektir.

Bu gereksinimdeki bu hedef, Şekil 4.4'te (Glinz 2008) gösterilen formatta belirli kısıtlama nitelikleriyle genişletilebilir.

“Nihai” bireysel gereksinim standart formunun başka varyasyonları da vardır, örneğin Hull ve diğerleri tarafından verilenler. (2011) ve IEEE 29148'de. Ancak bu varyasyonların daha önce tanıtılan formata indirgenebilir olduğu gösterilebilir.

Zorunlulukların Kullanımı

“Shall”, gereksinim belirtimlerinde sıklıkla kullanılan bir komut sözcüğü veya buyruktur. Sık kullanılan diğer komut sözcükleri ve deyimleri arasında “olmalı”,

“zorunlu”, “olacak”, “sorumlu” ve daha fazlası var. Wilson ve ark. (1997), Tablo 4.2'de özetlenen bu zorunluluklardaki ince farklılıkları tanımlamaktadır.

Gereksinimlerin zorunlu olduğu durumlarda zorunluluk olarak "yapılacaktır" ifadesinin kullanılması ve isteğe bağlı gereksinimler için olması gerektiği gibi zayıf zorunlulukların kullanılmaması iyi bir uygulama olarak kabul edilir. Gereksinimler önem derecesine göre sıralandığında, olması gerektiği gibi, düşük sıralı standart formdaki bir gereksinim, esasen isteğe bağlı bir gereksinim olarak kabul edilebilecek bir gereksinimdir.

Yapacak mı, Yapmayacak mı?

Gereksinimler yazılırken genellikle olumsuz biçim yerine olumlu biçimin kullanılması tercih edilir. Yani gereklilik, “shall not” ifadeleri yerine “shall” ifadeleri kullanılarak yazılmalıdır. IEEE 29148 yazmaya karşı tavsiyede bulunur

gereksinimleri tamamen “yapmayacaktır”. Ancak, özellikle tehlikelerle ilgili gereklilikler için istisnalar olması mümkündür ve bazen arzu edilir. Aşağıdaki örneği düşünün. Gereklilik

- Sistem sadece yetkili kullanıcıların erişimine izin vermelidir. eşdeğer olarak yeniden yazılabilir
- Sistem, yetkisiz kullanıcıların erişimine izin vermeyecektir.

Bu durumda, ilk biçimin anlaşılması daha kolaydır çünkü bir çift negatif içermez. Ancak, gereksinimin karşılandığını kanıtlamak için test senaryoları tasarlama sorununu düşünün. İlk formda, test senaryoları yetkili ve yetkisiz kullanıcıların alt kümelerini içermelidir. O zaman kullanıp kullanmayacağımıza karar vermeliyiz

hem yetkili hem de yetkisiz kullanıcılar için kapsamlı testler veya eşdeğerlik sınıfı testleri veya ikili testler vb. Tersine, ikinci biçimde, gereksinimin karşılandığını göstermek için yalnızca bir veya daha fazla yetkisiz kullanıcıyı içeren bir dizi test senaryosunun oluşturulması gerekir. Yetkisiz kullanıcılar evreninin bir alt kümesini nasıl seçeceğimize hala karar vermemiz gerekiyor, ancak gereksinimin

ikinci biçimi, testi önemli ölçüde basitleştiriyor.

Bazı durumlarda, şartın "yapmalı" formu yerine "yapmamalı" formunu kullanmanın paydaşlar (özellikle avukatlar) üzerinde psikolojik bir etkisi olabilir. Bu örneği düşünün:

- Sistem insanlara zarar vermeyecektir.

Bu gereksinim, gereksinimden daha basit görünüyor

- Sistem insanlara zarar vermeyecektir.

Her durumda, mümkün olduğunda ve açıklıktan ödün verilmediğinde, gerekli ifadeleri kullanarak gereksinimleri yazmalısınız. Ancak “yapmayacak” gereklilikleri, bazen “yapılacak” eşdeğer biçimlerine tercih edilebilir.

Gereksinimlerde Belirsizliği Önleme

Bir spesifikasyon belgesindeki tüm gereksinimler kesin olmalıdır. Bu, ölçülebilir miktarlar kullanmak ve sayısız, biraz, yaklaşık, çok büyük, küçük, mikroskobik vb. gibi belirsiz değiştiricilerden kaçınmaktır. Bu belirsiz kelimelerin ölçümlerle değiştirilmesi gerekiyor.

Örneğin, bagaj taşıma sistemi için aşağıdaki gereksinimi göz önünde bulundurun:

4.3.1 Sistem tarafından kaybedilen bagaj sayısı minimum olacaktır.

Bu ifade belirsizdir ve tasarımcılara gerçek bir rehberlik sağlamaz. Aşağıdaki gereksinim bir iyileştirmedir:

4.3.1 Sistem tarafından kaybolan bagaj sayısı, girilen bagajların %0,001'inden az olacaktır.

Bu gereksinim artık ölçülebilir ve dolayısıyla daha kesindir.

Kesin olmayan kelimelerin daha fazla örneği "birkaç", "bazıları", "çok" ve "çoğu"dur. Bir gereklilik olarak, bu kelimeler kesirler veya aralıklar ile değiştirilmelidir. Örneğin,

1/100, 1/20, 1/2 ve 9/10 kesirler sırasıyla yukarıda belirtilen kelimelerin yerini alabilir. Uygun aralıklar "<10", "10 ila 20", "21 ila 50" ve "51 ila

99” sırasıyla.

Bulanık sözcüklerin kullanıldığı diğer belirsizlik biçimleri, gereksinimlerde belirsizliğe yol açabilir. Örneğin, bir

gereksinimde "sıklıkla" olacak bir şeyin belirtilmesi

bir aralık veya yaklaşık bir oran ile değiştirilmelidir. Örneğin, "sıklıkla" yerine "zamanın %75'inden fazlasını" kullanın. Sabit bir sayı veya aralık sağlanması, bilgilerin bulanık bir sayıdan (veya aralıktan) daha ince bir aralık veya tam sayıya gelecekte iyileştirilmesi için bir referans noktası sağlar. Gereksinimlerde diğer riskten korunma kelimelerinden de kaçınılmalıdır. Mümkün olduğunda bir gereksinim bildiriminde bir olasılık veya aralık ile değiştirilmesi gereken bu kısa riskten korunma sözcükleri listesini göz önünde bulundurun:

- Muhtemelen

- Neredeyse
- Muhtemelen
- Belki
- Olabilir

Riskten korunma kelimelerini her zaman hassasiyetle değiştiremezsiniz. Bazen sadece bir tahmin mümkündür. Örneğin, önceden istatistikleriniz olmayan ve hatta tahmine dayalı istatistikler oluşturmak için bir olasılık modeli bile olmayan bir olayın olasılığına ilişkin bir tartışmayı düşünün. Bu, bazı kabul kriterlerini kabul etmek için paydaşlar tarafından müzakere edilen olasılıkların gerekli olduğu bir durum olacaktır.

Gereksinimler Belge Boyutu

Gereksinim sayısı, sayfa uzunluğu ve diğer ölçüler açısından gereksinim belirtimi belge boyutunun istatistiklerini kapsayan az sayıda çalışma vardır. Yine de şu soru sıklıkla sorulur: SRS belgesi ne kadar uzun olmalıdır? Yazılım gereksinimleri mühendisliği uygulamalarıyla ilgili yazılım uzmanlarıyla yapılan bir ankette, katılımcılara SRS belge uzunluğu soruldu (Neill ve Laplante 2003). Ankete katılanlar, belgelerin yaklaşık %40'ının 25 sayfadan az olduğunu ve yaklaşık %30'unun 25 ila 50 sayfa arasında olduğunu bildirdi. SRS belgelerinin kabaca %25'inin 51 ila 100 sayfa arasında, %17'sinin 101 ila 250 sayfa arasında olduğu ve bakiyenin 251 veya daha fazla sayfa uzunluğunda olduğu bildirildi.

Yukarıda belirtilen sayılar, 1990'larda 2000'lerin başlarına kadar geliştirilen 56

NASA gereksinimleri spesifikasyon belgesinin bir çalışmasında bulunanlarla tutarlıdır (Wilson ve diğerleri 1997). İncelenen projeler için, SRS belge boyutları, medyan 2.200 ve ortalama 4.700 metin satırı olmak üzere yaklaşık 140 ila 28.000 metin satırı arasında değişiyordu. Sayfa başına 60 satırlık bir metin olduğu varsayıldığında, bu rakamlar 2 ila 470 sayfa aralığına, ortalama 37 sayfaya ve ortalama 78 sayfaya dönüşmektedir.

Açıkça, endüstriyel norm, nispeten kısa gereksinimler spesifikasyon belgeleri içindir. Her bir SRS belgesinin uzunluk dışındaki niteliklere göre değerlendirilmesi gerektiğinden, bu bulgunun iyi mi yoksa kötü mü olduğu konusunda bir yargıya varılamaz.

Davranışsal Özellikler

Bazı durumlarda, gereksinimler mevcut olmadığında, eksik olduğunda, güncel olmadığında veya yanlış olduğunda gereksinim mühendisinden mevcut bir sistem için gereksinimleri tersine çevirmesi istenebilir. Test spesifikasyonlarının oluşturulması amacıyla açık kaynaklı yazılım (bir lisans koşulları altında kullanımı ve/veya yeniden dağıtımı ücretsiz olan yazılım) için gereksinimlerin oluşturulması da gerekli olabilir. Bu durumlarda, davranışsal belirtim olan bir SRS biçiminin oluşturulması gerekir.

Davranışsal belirtim, gereksinim belirtimiyle tüm yönleriyle aynıdır, ancak ilki, mühendislerin, kullanıcıların sistemin ne yapmayı amaçladığına dair en iyi anlayışını temsil ederken, ikincisi, kullanıcıların sistemin ne yapması

gerektiğine dair en iyi anlayışını temsil eder. . Davranışsal belirtim, ek bir çıkarım katmanına sahiptir ve bu nedenle, bir gereksinim belirtiminden bile daha az eksiksiz olması beklenebilir.

Neyse ki, davranışsal belirtimi oluşturmaya yönelik bir yaklaşım var. Teknik, sistemin amacını açıklayabilecek mümkün olduğunca çok sayıda eserden oluşan bir koleksiyon oluşturmayı içerir. Ardından, bu eserler sistemin amaçlanan davranışını en iyi anlaşıldığı şekilde yeniden yapılandırmak için kullanılır. Yukarıdaki açıklama, Elcock ve Laplante (2006) tarafından hazırlanan orijinal makaleden uyarlanmıştır.

> Davranışsal bir belirtim türetmek için kullanılabilecek eserler şunları içerir (ancak bunlarla sınırlı olmamalıdır)

- Güncel olmayan, eksik veya yanlış olduğu bilinen herhangi bir mevcut gereksinim belirtimi
- Kullanım kılavuzları ve yardım bilgileri (programı çalıştırırken mevcuttur)
- Sürüm notları
- Hata raporları ve destek talepleri
- Uygulama forumları
- İncelenen yazılımın ilgili sürümleri

Bu yapılardan bazılarının müşteri dosyaları, e-postalar, açık kaynak topluluk havuzları veya arşivler gibi çeşitli kaynaklardan temizlenmesi gerekebilir. Spesifikasyonun oluşturulmasında bu eserlerin nasıl kullanıldığına dair kısa bir açıklama aşağıda verilmiştir.

Kullanıcı kılavuzlarından başlayarak, kullanıcı girdisine yanıt olarak bir uygulamanın davranışı hakkındaki ifadeler, doğrudan davranış belirtimi ile ilgili olabilir. Yazılımın kullanıcı uyaranlarına tepkisini açıklamak onların varlığıyla ilgili olduğundan, kullanıcı kılavuzları bu amaç için özellikle uygundur. Destek web siteleri ve uygulama yardım menüleri gibi yardım bilgileri, davranışsal gereksinimleri tanımlamak için doğrudan kullanılabilen veya özetlenen bilgiler açısından da zengindir.

Ardından, sürüm notlarına başvurulabilir. Sürüm notları, belirli bir sürümde hangi özelliklerin uygulandığını açıklamaya odaklanma eğiliminde olduklarından, test senaryoları geliştirme sürecine genellikle sınırlı yarar sağlar. Genellikle açıklama yoktur

desteklenen özelliklerin nasıl çalışması gerektiğine ilişkin Bununla birlikte, sürüm notları, bir uygulama kısmen uygulanan veya gelecekteki uygulamadan kaldırılan özellikler için beklendiği gibi yanıt vermediğinde ortaya çıkan çatışmayı çözmede özellikle önemlidir.

Hata raporları, genellikle kullanıcılar tarafından yazıldığından, yazılımın sorunlu alanlarını belirlediklerinden ve genellikle bir geliştiricinin belirli davranışa yönelik niyetini açıklığa kavuşturduklarından davranışsal yanıtları çıkarmak için harika bir kaynak olabilir. Hata raporları, en azından açık kaynaklı sistemler için, Bugzilla gibi açık depolarda kolaylıkla bulunur. Bazı durumlarda hata raporlarının davranışsal yanıtları çıkarmak için yararlı olamayacak kadar çok uygulama ayrıntısı

içerdiği doğru olsa da, davranış ayrıntılardan tahmin edilemezse bunlar atılabilir.

Birçok yönden, destek isteklerinin içeriği, beklenmeyen davranışları belirlemeleri bakımından hata raporlarına benzer. Bununla birlikte, hata raporlarının aksine, destek talepleri bazen tam olarak uygulanmayan özellikleri belirlemede ve ayrıca kullanıcıların beklentilerini aydınlatan bilgiler sağlamada yardımcı olabilir. Her ikisini de araştırmak önemlidir, çünkü bahsedilen içgörüleri sağlamanın yanı sıra, beklenmeyen davranışların rasyonelleştirilmesine de yardımcı olabilirler.

Birçok açık kaynaklı proje ve bazı kapalı kaynaklı projeler için projeyle ilişkili Web tabanlı forumlar vardır. Bu forumlarda, çeşitli miktarlarda davranışsal bilgi

çıkarılabilir. Yararsızdan alakalıya kadar, açık tartışma gönderilerinin dikkatlice filtrelenmesi ve yalnızca diğer geliştirme eserleri eksik olduğunda uygulanması gerekir. Diğer eserlerde olduğu gibi, bu ilanlar da davranışı netleştirmek için kullanılabilir.

Son olarak, başka herhangi bir yapaylığın yokluğunda, test edilen yazılımın kendisi yapısal (cam kutu) test senaryolarının geliştirilmesi için bir girdi olabilir. Bu yaklaşımın gerekli olduğunu varsayarsak, gerçek şu ki, test senaryolarının tanımlanmasında büyük ölçüde hakim olacak olan, gerçekten de test edenin doğru davranışı karakterize etmesi olacaktır.

Keşif süreci sona erdiğinde, davranışsal belirtim yazılabilir. Davranış spesifikasyonunun formatı, gereksinim

spesifikasyonunun formatı ile aynıdır (Elcock ve Laplante 2006) ve tüm IEEE 29148 kuralları uygulanmalıdır.

En İyi Uygulamalar ve Öneriler

Etkili gereksinim spesifikasyonları yazmak, belirttiğimiz birçok zorluk nedeniyle önemsiz sistemler için bile çok zor olabilir. Zayıf SRS belgeleri yazmanın daha yaygın tehlikelerinden bazıları şunlardır:

- Operasyonel ve tanımlayıcı özelliklerin karıştırılması
- Düşük düzeyli donanım işlevselliği ile üst düzey sistemler ve yazılım işlevlerini aynı işlevsel düzeyde birleştirmek
- Zamanlama bilgilerinin ihmal edilmesi

Diğer kötü uygulamalar, doğrulanabilecek bir dilin kullanılmamasından kaynaklanır. Örneğin, bu gereksinimler grubunu göz önünde bulundurun:

- Sistem tamamen güvenilir olacaktır.
- Sistem modüler olacaktır.
- Sistem hızlı olacaktır.
- Hatalar %99'dan az olacaktır.

Bunların nesi var? Sorun şu ki, tamamen belirsiz ve ölçülemezler ve bu nedenle memnuniyetleri kanıtlanamıyor. Örneğin, “tamamen güvenilir” ne anlama geliyor? Herhangi bir keyfi kişi, belirli bir sistem için güvenilirlik için farklı bir anlama sahip olacaktır. Modülerliğin (yazılımda) belirli

bir anlamı vardır, ancak nasıl ölçülebilir? "hızlı" ne demek Tren kadar hızlı mı? Bir mermiden daha hızlı? Bu gereksinim çok belirsizdir. Ve son olarak, “hatalar %99'dan az olacak” bir dava reçetesidir. neyin %99'u? Hangi zaman diliminde?

Yukarıdaki gereksinimler için geliştirilmiş sürüm:

- Birinci seviye eylemlerin tümü için yanıt süreleri 100 ms'den az olacaktır.
- Her modülün döngüsel karmaşıklığı 10 ila 40 aralığında olmalıdır.
- İşlemlerin %95'i 1 saniyeden daha kısa sürede işlenecektir.
- İlk arızadan önceki ortalama süre 100 saat sürekli çalışma olacaktır.

Ancak bu set bile kusurludur, çünkü bazı önemli ayrıntılar eksik olabilir - SRS belgesinin geri kalanının bağlamı dışında neyin eksik olduğunu gerçekten bilemeyiz. Glintz'in öznitelik şablonunu kullanmak (Şekil 4.4) bu soruna iyi bir çözümdür.

Spesifikasyon belgelerinin yazılması için bazı son öneriler şunlardır:

- Standart bir format icat edin ve kullanın ve tüm gereksinimler için kullanın.
- Dili tutarlı bir şekilde kullanın.
- Zorunlu gereksinimler için "yapmalı" ifadesini kullanın.
    - Arzu edilen gereksinimler için "should" ifadesini kullanın.

- Gereksinimin önemli kısımlarını belirlemek için metin vurgulamayı kullanın.
- Garanti edilmedikçe teknik dil kullanmaktan kaçının.

Dick (2010), gereksinimler belgesini yazarken aşağıdakilerden kaçınılmasını da önerir:

- Saçma sapan ifadeler
- “Gerekirse” gibi riskten korunma maddeleri
- “ve”, “veya” ve “ama” gibi bağlaçlar
- tek bir ifadede birden fazla gereksinime yol açabilir

- “Belki” ve “bazen” gibi skülatif terimler
- Doğrulanamadığı için “kullanıcı dostu” ve “genel olarak” gibi belirsiz terimler.

Ancak, gereksinim belirtimini yazma işlemimizi henüz bitirmedik. Bölüm 5, belirli gereksinimlerin yazılmasını mükemmelleştirmeye ayrılmıştır.

VIGNETTE 4.1 Ürün Hattı Mühendisliği

Ürün hattı sistemleri için gereksinim spesifikasyonlarının (bkz. Alıştırma 1.12) ortak gereksinimlerin bir temel çizgisinden veya şablonundan oluşması sıklıkla karşılaşılan bir durumdur. Ardından, hattaki ilgili ürünlerin çeşitli örnekleri için

farklı gereksinimler eklenir. Bir ürün hattı için amaca yönelik gereksinim mühendisliği, gelişigüzel yaklaşımlara göre önemli maliyet tasarrufları anlamına gelebilir. Aşağıdaki örneği düşünün. Çok büyük (milyar dolarlık) çok uluslu bir firma, küçük gazla çalışan motorlardan çok büyük jet türbinlerine kadar her türden motor üretmektedir. Şirketin, her birinde 50.000'den fazla gereksinim bulunan 150'den fazla büyük projesi var. Bu projelerin çoğu çok benzer gereksinim setlerine sahiptir. Yine de ürün hattı gereksinimlerinin kaldıraç etkisi

mühendislik yapılmaz.

Araç kullanımı projeler ve mühendislik sahaları arasında tutarsızdır, bu nedenle yeniden kullanılabilirlik, değişiklik takibi ve kaçırılan veya aşırı gereksinimlerin belirlenmesi gücü kaybolur. Büyük bir

organizasyonun işleri daha iyi yapması beklenebilir, ancak bu şirket için sorunun bir kısmı, şirketin ilk günlerinde hiçbir araç bulunmaması ve eski mühendislerin ürün hattı mühendisliği potansiyelini fark etmemiş olmalarıdır. ve işlerin yapılma şeklini değiştirme ihtiyacı. Şirket onlarca yıl içinde birçok farklı araç kullanan ve çok çeşitli mühendislik uygulamalarına sahip şirketleri satın alarak büyüdükçe başka sorunlar da ortaya çıktı.

Elbette, şirket genelinde alet kullanımını ve mühendislik uygulamalarını standart hale getirmek çözümün büyük bir parçasıdır, ancak eğitim de öyle, böylece tüm mühendisler aleti aynı ve doğru şekilde kullanabilirler. Mühendislerin hepsinin “aynı sayfada” olması için bir kurumsal ve proje “metaforu” oluşturulmalıdır. Katı ürün hattı

mühendisliği uygulamaları o zaman mümkün olacaktır.

Az önce açıklanan durum büyük şirketlere özgü değildir.

Egzersizler

4.1 Bir SRS'yi yalnızca resmi olmayan teknikleri kullanarak hangi koşullar altında sunmak uygundur?

4.2 Davranış spesifikasyonu, bir gereksinim belgesinin sağlayamadığı neyi sağlayabilir?

4.3 Müşteri, gelecekteki büyüme ve iyileştirme fikirlerinin saklanmasını isterse, bu fikirler nereye yerleştirilebilir?

4.4 SRS'de "veri saklama" kapsamına dahil edilecek bazı öğeler nelerdir?

4.5 Burada, gerçek gereksinim belirtimlerinde fiilen ortaya çıkan bazı

belirsiz ve belirsiz gereksinim örnekleri verilmiştir. Neden belirsiz, eksik veya belirsiz olduklarını tartışın. Bu gereksinimlerin geliştirilmiş versiyonlarını sağlayın (gerekli varsayımları yapın).

4.5.1 Araç, sonraki raporlar için gerekli olan önemli veri alanlarına hızlı veri girişine izin verecektir.

4.5.2 Sistem, istenmeyen arıza modlarını ve/veya kabul edilebilir sınırların altındaki performans düşüşünü belirlemek ve ortadan kaldırmak için etkili bir araç sağlayacaktır.

4.5.3 Veritabanı, gerekli kişileri anında uyaran otomatik bir olay raporu oluşturur.

4.5.4 Mühendis, veritabanı da dahil olmak üzere sistem etkinliğini yönetecektir.

4.5.5 Rapor, gereksinimleri karşılamaya yetecek miktarda ve ayrıntıda verilerden oluşacaktır.

4.5.6 Sağlanan veriler, arızayı hızlandıran olay ve koşulların sağlıklı bir şekilde belirlenmesine olanak sağlayacaktır.

4.5.7 Dokümante edilmiş analiz raporu, uygun olduğu şekilde, araştırma bulgularını, mühendislik analizini ve laboratuvar analizini içerecektir.

4.6 Ekteki SRS Bölüm 9.4'te, “ölçülebilir hedeflerin” Şekil 4.4'te gösterilen formatta kullanılması için hangi gereksinimler uygundur?

4.7 Ekteki gereksinimlerin çoğu çeşitli şekillerde geliştirilebilir. Listelenen on gereksinimi seçin ve bunları geliştirilmiş bir biçimde yeniden yazın. IEEE 29148'in kelime dağarcığını kullanarak yeniden

yazılmış gereksinimlerinizin neden üstün olduğunu tartışın.

4.8 Ek A'daki aşağıdaki “yapmamalı” gerekliliklerini “yapmalı” formatına dönüştürün: 4.1.1, 5.2.2, 6.1.8, 8.1.17.

4.9 Ek A'daki aşağıdaki “yapmamalı” gerekliliklerini “yapmalı” formatına dönüştürün: 9.1.2.6, 9.3.3, 9.3.5, 9.3.13, 9.5.10.

*4.10 Bağlam diyagramları oluşturmak için farklı teknikleri araştırın ve her birinin güçlü ve zayıf yönlerini vurgulayan bir rapor hazırlayın.

*4.11 Ek A ve B'deki gereksinim özellikleri, zorunlulukları tutarsız bir şekilde kullanır. ARM gibi bir metinsel analiz aracı kullanarak veya manuel analiz yoluyla, bu belgelerdeki emirlerin kullanımını değerlendirin ve iyileştirmeler önerin