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:

Gereksinim analizi sorunlar için gereksinimleri analiz etme faaliyetidir. Sorunlu Gereksinim şunları içerir:

- Komplike olma
- Konudan dışı olma
- Tekrarlanan olma (Kopya)
- Çelişki içerme
- Eksik olma

Bu kavramları Bölüm 5'te daha detaylı hale getiriyoruz.

Gereksinim analizi genellikle informal öğrenme biçimine sahiptir. Buradaki informal öğrenme, bilinçli bir öğrenme sürecine girmeksizin içgüdüler (merak, gözlem) veya ihtiyaçlar sonucunda doğal olarak gerçekleştirilen öğrenme süreci olarak nitelendirilir. Örneğin müşterilerden ortaya çıkan gereksinimler paydaş temsilcilerine, yapabilecekleri bir formatta gösterilir. Bu format, tüm müşterilerin isteklerine ilişkin gereksinim mühendisinin yorumunun doğru olduğundan emin olmak için oluşturulur. Resmi yöntemler, gereksinim analizi için daha titiz teknikler sağlar.

Farklı kaynaklardan elde edilen aynı gereksinimlerin farklarını uzlaştırma işlemine Gereksinim Anlaşması denir. Gereksinim anlaşması genellikle sistematik bir süreç olsa da informal bir süreç kullanılmalıdır. Kullanım durumları ve kullanıcı hikayeleri genellikle bu amaç için etkili araçlardır (Hull ve ark. 2011). Ekran maketleri özellikle fonksiyonel tekniklerle birlikte kullanıldığında başka bir yararlı tekniktir (Ricca et al. 2010). Diğerleri senaryolar, storyboard'lar, papers prototypes, gereksinimler anlaşması için prototipler, kodlanmış konsept oluşturucular ve işlevsel prototipler. (Sutcliffe et al. 2011).

Müşteri sözleşmesi sürecinin bir parçası olarak, Hull ve ark. (2011) şu soruları sormayı öneriyor:

- Gereksinim tamamlandı mı?

■ Gereklilik açık mı?

■ Gereklilik uygulanabilir mi?

■ Yeterlilik planı açık ve kabul edilebilir mi?

Bölüm 5'te bir gereksinim spesifikasyonunun bu soruların istenen niteliklerle nasıl ilişkili olduğunu not edeceksiniz. Ve bölüm 5 ve 6 da gereksinim mühendisliği yaşam döngüsü boyunca bu nitelikleri başarmanın yollarına bakacağız.

Gereksinim Gösterimi

Kullanım durumları, kullanıcı öyküleri, doğal diller, biçimsel diller, stilize edilmiş (sınırlandırılmış doğal) diller ve çeşitli biçimsel, yarı biçimsel ve biçimsel olmayan diyagramlar dahil olmak üzere, herhangi bir sistemdeki işlevselliği tanımlamak için çeşitli teknikler kullanılabilir. kullanımı önemli bir zorluktur ve bir bakıma bu kitabın bir temasıdır. prototipler, komut dosyasına dayalı konsept oluşturucular ve gereksinim anlaşması için işlevsel prototipler (Sutcliffe et al. 2011). 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:

Uygulanmakta olan 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, yazılım ekibinin yapısı ve yetkinliği gibi yazılım projesinin belirli özellikleri bulunur.

Proje için veya organizasyon çapında doğru temsil stilini seçme kullanımı önemli bir zorluktur. (kitapözeti)

Gereksinim Temsiline Yaklaşımlar

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

Gayri resmi gereksinimleri temsil teknikleri, kesin bir matematiksel gösterime tamamen 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 bile olsa, doğal dili kullanmak zorundadır. Bu nedenle, dikkatimizi daha çok resmi olmayan teknikler kullanarak gereksinimlerin temsiline odaklayacağız.

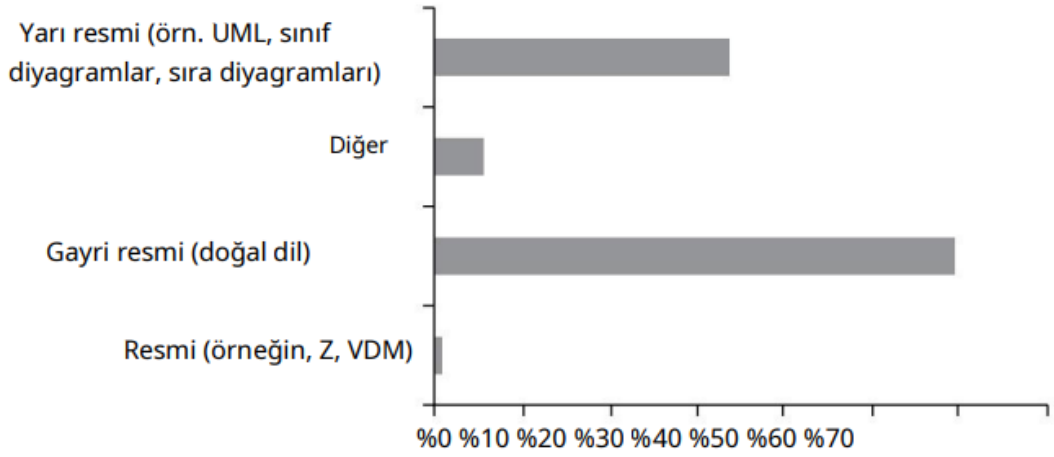
Son olarak, yarı-biçimsel temsil teknikleri, gayri resmi görünseler de, en azından kısmi bir biçimsel temele sahip olanları içerir. Örneğin, birleşik modelleme dili (UML) veya sistem modelleme dili (SysML) metamodelleme dilleri ailesindeki, kullanım senaryosu diyagramı da dahil olmak üzere, diyagramların 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 çoğunluğu doğal dil kullanılarak yapılan resmi olmayan modellemeye odaklanmaktadır. Kasab et al. (2014), ankete katılanların %60'ının gereksinimleri doğal dil olarak, yani gayri resmi olarak ifade edildi (Şekil 4.1)

Bildirilen yarı biçimsel yaklaşımlar, UML/SysML diyagramlarını içeriyordu. İçin Z ve VDM dahil olmak üzere rapor edilen kötü 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. Requirement specification notation prevalence.



Yazılım ve Sistemler için Gereksinim Mühendisliği

Nott :

Uygulamada, resmi olmayan (doğal dil), yarı resmi (örneğin, UML/ SysML modelleri) ve resmi öğelerin uygun bir karışımının olması önerilir

ISO/IEC/IEEE Standardı

) 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, tüm 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 anılacaktı

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

Alıcılar veya tedarikçiler arasındaki anlaşmanın temelini oluşturmak için ürün ne yapmalı:

- 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

- Bir ürünü yeni kullanıcılara veya yeni kullanıcılara dağıtmak için bilinçli bir temel sağlamak

- Ürün geliştirme için bir temel sağlamak için (IEEE 29148 201

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

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

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

Temsil Önerileri

İşlevsel Olmayan Gereksinimler

IEEE 29148, aşağıdakileri içeren çeşitli işlevsel olmayan gereksinimleri açıklar:

Harici 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

Amaç açıklaması

Girdi kaynağı veya çıktı hedefi

Geçerli aralık, doğruluk ve/veya tolerans

Ölçü birimleri

Zamanlama

Diğer girdi /çıktılarla ilişkiler

Ekran biçimleri

Pencere biçimleri

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 hem normal hem de en yoğun iş yükü koşulları için belirli zaman dilimlerinde işlenecek veri miktarını içerebilir.

Mantıksal veritabanı gereksinimleri, aşağıdakiler gibi çeşitli işlevler tarafından kullanılan bilgi türleridi

Kullanım sıklığı

Erişim Yetenekleri

Veri varlıkları ve ilişkileri

Bütünlük kısıtlamaları

Veri saklama gerenksinimleri

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. Genel olarak, bu tür herhangi bir nitelik, bir "iltifat" olarak adlandırılır, çünkü çoğu bir "ilgi" ile biter. Ancak güvenlik, dakiklik, güvenlik ve diğerleri gibi diğer nitelikler de illüzyonlardır.

İşlevsel Gereksinimlerin Temsil Edilmesine İlişkin Öneriler

İşlevsel gereksinimler, tüm sistem girdilerini ve her girdi olasılığı için normal ve anormal durumlara verilen işlemlerin ve yanıtların(çıktıların) tam sırasını kapsamalıdır. 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 düzenlenebilir:

- Sistem modu (ör. navigasyon, savaş, teşhis\tanı)
- Kullanıcı sınıfı (ör. kullanıcı, süpervizör, teşhis\tanı)
- Nesne (sınıfları/nesneleri, öznitelikleri, işlevleri/yöntemleri ve mesajlar)
- Özellik (sistemin kullanıcıya ne sağladığını açıklar.)
- Uyaran (ör. sensör 1, sensör 2, aktüatör 1, …)
- İşlevsel hiyerarşi (ör. görevlerin yukarıdan aşağıya ayrıştırılması)

Bu tekniklerin bir kombinasyonu tek bir SRS belgesinde kullanılabilir.

Örneğin , Ek A'da verilen akıllı ev açıklaması, işlevselliğe yönelik özellik odaklı bir açıklamadır. Özellik odaklı olduğuna dair bir ipucu, gereksinimlerin çoğu için "sistem gerekir" sloganıdır. 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 Gereklilikleri

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

İşletim Sistemi

01 İşletim sistemi, çoklu görev desteği, CPU zamanlaması, temel iletişim ve bellek yönetimi gibi gerçek zamanlı sistemleri desteklemek için gerekli ortak bir mekanizma seti sağlamalıdır.

01.1 İşletim sistemi, çoklu görev yetenekleri sağlamalıdır.
01.2 İşletim sistemi, olaya dayalı, önceliğe dayalı zamanlama sağlamalıdır.
01.2.1 Görevin yürütülmesi, bir görevin önceliği ve yürütülmesi için gereken kaynakların mevcudiyeti ile kontrol edilecektir.
01.3 İşletim sistemi, görevler arası iletişim ve senkronizasyon için destek sağlayacaktır.
01.4 İşletim sistemi gerçek zamanlı saat desteği sağlayacaktır.
01.5 İşletim sistemi yazılımı, 80387 matematik yardımcı işlemcisi için görev düzeyinde bağlam değiştirme sağlayacaktır.

Ayrıca sistem işlevselliği altında, aşağıdaki gibi tanımlanan komut doğrulama vardır:

Komut Doğrulama

211 Uçuş yazılımı, CCSDS komut yapısı doğrulamasını gerçekleştirecektir.

211.1 Uçuş yazılımı, CCSDS transfer çerçevelerinin doğru ve sırayla alındığını doğrulamak için 1 numaralı CCSDS Komuta Operasyon Prosedürü (COP-1) uygulayacaktır.

211.2 Uçuş yazılımı, 127'lik bir sabit boyutlu çerçeve kabul ve raporlama mekanizması (FARM) sürgülü penceresini ve 63'lük sabit FARM negatif kenarını destekleyecektir.

211.3 Uçuş yazılımı, aşağıdaki hatalardan herhangi biri meydana gelirse durumu telemetre edecek ve gerçek zamanlı komut paketini atacaktır:

- Sağlama toplamı, yayınlanmadan önce doğrulamada başarısız oluyor.
- Geçersiz bir uzunluk algılandı.
- Geçersiz bir uygulama kimliği algılandı.

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)

Ayrıca, sistem davranışını tanılamak 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.

Nesne yönelimli temsiller, nesneler olarak adlandırılan oldukça soyut sistem bileşenlerini ve bunların kapsüllenmiş niteliklerini ve davranışlarını içerir. Sistemlerin geleneksel yapılandırılmış tanımları ile sistemlerin nesne yönelimli tanımları arasındaki farklar Tablo 4.1' de özetlenmiştir.

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

	Yapılandırılmış	Nesne-Yönelimli
Sistem Bileşenleri	Fonksiyonlar	Nesneler
Veri ve kontrol belirtimi	Dahili ayrıştırma yoluyla ayrıştırılmış	Nesne içinde kapsüllenmiş
Özellikler	Hiyerarşik yapı	Nesnelerin kalıtım ilişkisi
	Sistemin fonksiyonel açıklaması	Sistemin davranışsal açıklaması
	Bilginin işlevler içinde kapsüllenmesi	Bilginin nesneler içinde kapsüllenmesi

Tablo 4.1 Nesne Dayalı Temsil ve Yapılandırılmış Temsil

UML/SYSML

(Birleştirilmiş Modelleme Dili/Sistem Modelleme Dili)

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' in bir varyasyonudur. UML ve SysML oldukça benzerdir. Okuyucuya kolaylık olması için, UML standardına önemli bir katkı sağlayan 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.

Gereksinim 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' in bir sözleşme olduğunu ve bu nedenle medeni sözleşme hukuku (veya belirli türlerde ceza hukuku) kapsamında uygulanabilir olduğunu hatırlamak da önemlidir. Dolandırıcılık veya ihmal kanıtlanabilir.

Gereksinim Belgesinin Kullanıcıları

Gereksinim belgesinin, her biri benzersiz bir bakış açısına, gereksinimlere ve endişelere sahip birkaç kullanıcısı vardır.

Belgenin tipik kullanıcıları şunları içerir:

- 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 tamamlama göstergesi arıyorlar. Ve elbette geliştiriciler, hangi sistemin geliştirileceğini anlamak için gereksinim belgesini kullanır. Aynı zamanda, test mühendisleri sistem için doğrulama testleri geliştirmek için gereksinimleri kullanır. Daha sonra bakım mühendisleri, sistemin yükseltilebilmesi veya sabitlenebilmesi için sistemi ve parçaları arasındaki ilişkiyi anlamaya yardımcı olmak için gereksinimleri kullanacaklardır. SRS' i kullanacak diğer paydaşlar, söz konusu sistemin doğrudan ve dolaylı tüm lehtarlarını (veya muhaliflerini) ve ayrıca davayı görüntüleyecek avukatlar, hakimler, davacılar, jüriler, bölge savcıları, hakemler, arabulucular vb. Uyuşmazlık durumunda yasal belge olarak SRS' i kullanırlar.

Gereksinim Belgesi Gereksinimleri

SRS için doğru format nedir? Birçok olası biçim vardır. Örneğin, IEEE 29148 standardı, bir gereksinim belirtimi 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, özelleştirebilir biçimlerde belgeler üretecektir. Ancak "doğru" format sponsorun, müşterinin, işverenin ve diğer paydaşların neye ihtiyacı olduğuna 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 kaydetmesi gerekir.

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 açıklayıcı davranıştan ayırma
- Modeller içinde tutarlı soyutlama düzeyleri ve modeller arasında ayrıntılandırma düzeyleri arasında uygunluk kullanma
- Spesifikasyon modellerinin bir parçası olarak işlevsel olmayan gereksinimleri modelleme – 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 Chapter-5' te tartışılacaktır.

Tercih Edilen Yazı Stili

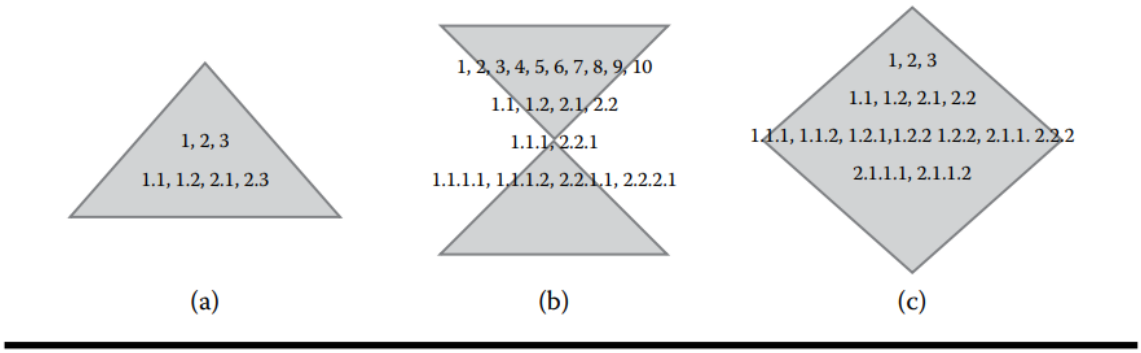
Mühendisler (her türden), zayıf iletişim becerileri, özellikle de yazma konusunda haksız bir itibar kazanmıştır. Her durumda, gereksinim belgelerinin çok iyi yazılması gerektiği artık açık olmalıdır.

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 önermiştir (Norden 2007).

Her durumda, gereksinim belgesine herhangi bir yazı gibi yaklaşın – tekrar tekrar yazmaya ve yeniden yazmaya hazır olun. Gereksinimler belgesinin birkaç diğer paydaş tarafından (ve iyi yazan paydaş olmayanlar tarafından) gözden geçirilmesini sağlayın.

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 özelliği, numaralandırma yapısının derinliğidir.



(a) Piramit, (b) Kum Saati ve (c) Elmas şekli --------
(Gereksinimler metin yapısı için yapılandırmalar.)

Numaralandırma yapısı derinliği, kaynak belgenin her düzeyindeki numaralı ifadeleri sayan bir ölçümdür. Örneğin, 1.0, 2.0, 3.0 vb. numaralandırılmış birinci düzey gereksinimlerin çok yüksek düzeyli (soyut) gereksinimler olması beklenir. 1.1, 1.2, 1.3 … 2.1, 2.2, 2.3 … 3.1, 3.2 vb. numaralı ikinci seviye gereksinimler, daha düşük ayrıntı düzeyindeki ikincil gereksinimlerdir. Daha da ayrıntılı gereksinimler, 1.1.1, 1.1.2 vb. olarak numaralandırılmış üçüncü düzeyde bulunacaktır. 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 düzenlenmiş bir SRS belgesi tutarlı bir ayrıntı düzeyine sahip olmalıdır ve her düzeyde gereksinimleri listelerseniz, sonuçta ortaya çıkan şekil, düzey 1'de birkaç numaralı ifade olması gerektiği için bir piramit gibi görünmelidir ve her alt düzey, üzerindeki seviyeden giderek daha fazla numaralı ifadeye sahip olmalıdır (Şekil 4.3a).

Öte yandan, her düzeyde gereksinimleri sayılan gereksinim belgeleri kum saati şekline benzer (Şekil 4.3b) genellikle çok miktarda tanıtıcı ve idari bilgi içeren belgelerdir. Son olarak, bir piramit tarafından temsil edilen elmas şeklindeki belgeler ve ardından daha düşük seviyelerde azalan ifade sayıları (Şekil 4.3c) tutarsız bir ayrıntı gösterimi düzeyine işaret eder (Hammer ve ark. 1998).

Bölüm 5'te sunulan 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 (değildir)

[isim tamlaması] gerekliliğin ana konusunu tanımlarken,
olacak (veya olmamalıdır) ifadesi, gerekli veya yasaklanmış bir davranışı belirtir ve [fiil deyimi] gereksinimin eylemlerini gösterir.

Bagaj taşıma sistemi için şu iki gereksinimi göz önünde bulundurun:

■ Sistem, etiketlenmemiş bagajı reddedecektir.

■Sistem hasarlı bagajı reddetmeyecektir.

Şartların geçerli olduğunu varsayıyoruz “bagaj” ,”reddetmek”,”etiketli” ve “ hasar” kelimesi srs belgesinde bi yerde tanımlanmıştır.

Gereksinimlerin standart olmayan bir 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 da nadir görülen bir durum değildir, bu nedenle iki örnek gereksinim şöyle olacaktır:

■Sistem, etiketlenmemiş bagajı reddedecektir.

■Sistem hasarlı bagajı reddetmeyecektir

burada gereksinimlerin bu formülasyonunda yanlış bir şey yok. 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]

Önceki iki gereksinime dönersek,ve bazı tanımlayıcı numaralar icat edersek, aşağıda iki örneğimiz var:

1.2.1 Sistem, etiketlenmemiş bagajı reddedecektir.

1.2.2 sistem hasarlı bagajı reddetmeyecektir.

aslında, standart gereksinim formunu aşağıdakiler ile zenginleştirerek, mümkün olduğunda işlevsel gereksinimler için performansa ölçülebilir kısıtlamalar koyulabilir :

• 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 türden ortalama 1.000'den fazla parçayı 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.

Şekil 4.4 Bagaj taşıma sistemi için örnek gereksinimlerin öznitelik metrikleri.

Ölçülebilir bir hedef içeren bir gereksinime örnek olarak bagaj taşıma sistemine dönelim. 1.2.1 numaralı gereksinim için, 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. Bununla birlikte, bu varyasyonların daha önce tanıtılan formata indirgenebilir olduğu gösterilebilir.

Gereksinimlerin kullanımı :

“Shall” sözcüğü gereksinim belirtimlerinde sıklıkla kullanılan bir komut sözcüğü veya buyruktur. Sık kullanılan diğer komut sözcükleri ve deyimleri arasında “olmalı”, “zorunlu”, “olacak”, “sorumlu” ve daha fazlası yer alır. Wilson ve ark. (1997), Tablo 4.2'de özetlenen bu gereksinimlerdeki ince farklılıkları tanımlamaktadır.

gereksinimlerin gerekli olduğu durumlarda gereklilik olarak "olacak" ifadesinin kullanılması ve isteğe bağlı gereksinimler için olması gerektiği gibi zayıf gereksinimlerin kullanımının atlanması 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.

Olacak mı , olmayacak 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.

tablo 4.2 Gereksinim Sözcükleri ve Deyimleri (Wilson ve ark. 1996)

Gereksinim	En Yaygın Kullanım
uygulanabilir	Belirtilen gereksinimlere ek olarak , referans olarak, standartlar veya diğer belgeleri dahil etmek
…İçin gereklidir	pasif bir sesle yazılmış olan Spesifikasyonlarda bir gereksinim
gerekli	Performans oluşturmak için gereksinimler veya kısıtlamalar
-den sorumlu	Gereksinim belgelerinde sistemler için yazılmış mimariler önceden tanımlanmış
Ecek - acak	İşlevsel bir yeteneğin sağlanmasını dikte(prensip) etmek
Meli-malı	Gereksinim belirtim ifadelerinde bir zorunluluk olarak sıklıkla kullanılmaz
Azim	Geliştirme ortamının belirtilen yeteneğe sağlayacağı şeylerden alıntı yapmak

Kaynak: Wilson, W.M., Rosenberg, L.H., & Hyatt, L.E. 1997, Automated analysis of requirement specifications, in Proceedings of the 19th International Conference on Software Engineering, pp. 161–171

gereksinimler tamamen karşılamayabilir. Ancak, özellikle tehlikelerle ilgili gereklilikler için istisnalar olması mümkündür ve bazen arzu edilir. Aşağıdaki örneği düşünün.

■Sistem sadece yetkili kullanıcıların erişimine izin verecektir.

eşdeğer olarak yeniden yazılabilir

■ Sistem, yetkisiz kullanıcıların erişimine izin vermeyecektir.

Bu durumda, ilk biçimin anlaşılması daha kolaydır çünkü negatiflik

içermez. Ancak, gereksinimin karşılandığını kanıtlamak için test senaryoları tasarlama sorununu düşünün. İlk formda, test senaryoları yetkili ve yetkisiz kullanıcıların alt kümelerini içermelidir.

Ardından, hem yetkili hem de yetkisiz kullanıcılar için kapsamlı test mi, eşdeğerlik sınıfı testi mi yoksa ikili test vb. mi kullanacağımıza karar vermemiz gerekiyor.

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, gerekliliğin “yapmalı” formu yerine “yapmaz” formunu kullanmanın paydaşlar (özellikle avukatlar) üzerinde psikolojik bir etkisi olabilir. Bu örneği düşünün:

■ Sistem insanlara zarar vermeyecektir.

Bu gereksinim, diğer gereksinimden daha basit görünüyor

■sistemin amacı insanlara zarar vermemektir.

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(şartname,beyanneme) belgesindeki tüm gereksinimler kesin olmalıdır. Ölçülebilir miktarlar kullanılmalıdır ve sayısız, biraz, yaklaşık, çok büyük, küçük, mikroskobik vb. gibi belirsiz değiştiricilerden kaçınılmalıdır. Bu belirsiz kelimelerin ölçümlerle değiştirilmesi gerekiyor.

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

4.3.1 Sistem tarafından kaybedilen bagaj sayısı minimum olmalıdır.

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

4.3.1 Sistem tarafından kaybedilen bagaj sayısı, girilen bagajların %0,001'inden az olmalıdır.

Bu gereksinim artık ölçülebilir ve dolayısıyla daha kesindir.

Kesin olmayan kelimelere daha fazla örnek "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 bahsedilen sözcüklerin yerini alabilir. Uygun aralıklar sırasıyla "<10", "10 ila 20", "21 ila 50" ve "51 ila 99" olabilir.

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:

- Büyük ihtimalle
- 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 paydaşların müzakere ettiği olasılıkların gerekli olacağı 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 geri kalanın 251 veya daha fazla sayfa uzunluğunda olduğu bildirildi.

Yukarıda bahsedilen 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 standart, nispeten kısa gereksinim spesifikasyon (tanımlama,şartname) 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. Bu 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. 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 eder. ikincisi, kullanıcıların sistemin ne yapması gerektiğine dair en iyi anlayışını temsil eder. Davranışsal belirtimin ek bir çıkarım katmanı vardır ve bu nedenle, bir gereksinim belirtiminden bile daha az eksiksiz olması beklenebilir. Neyse ki, davranışsal belirtimi oluşturmaya yönelik bir yaklaşım vardır. Teknik, mümkün olduğu kadar çok yapıtın bir koleksiyonunu bir araya getirmeyi içerir. Bu sistemin amacını açıklayabilir. 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. Daha sonra, sürüm notlarına başvurulabilir. Sürüm notları, belirli bir sürümde hangi özelliklerin uygulandığını açıklamaya odaklanma eğiliminde olduklarından, test senaryoları geliştirme sürecinde genellikle sınırlı fayda sağlar. Desteklenen bu özelliklerin nasıl çalışması gerektiğine dair genellikle bir açıklama yoktur.

Bununla birlikte, sürüm notları, kısmen uygulanan özellikler için bir uygulama beklendiği gibi yanıt vermediğinde ortaya çıkan çakışmayı çözmede özellikle önemlidir.

Hata raporları, genellikle kullanıcılar tarafından yazıldığından, yazılımın sorunlu alanlarını belirlediklerinden ve genellikle bir geliştiricinin belirli davranışa yönelik niyetini açıkladığından, davranışsal yanıtları çıkarmak için harika bir kaynak olabilir. Kusur raporları, en azından açık kaynaklı sistemler için , Bugzilla gibi açık depolarda kolayca 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, beklenmedik davranışları rasyonelleştirmeye 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. 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

En İyi Uygulamalar ve Öneriler Etkili gereksinim belirtimlerini 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 seviyeli donanım fonksiyonelliği ile üst seviye sistem ve yazılım fonksiyonelliğini aynı fonksiyonel seviyede birleştirmek

■ Zamanlama bilgilerinin ihmal edilmesi

■ Yazılım ve Sistemler için Gereksinim Mühendisliği.

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 olmalıdır

Bunların nesi var? Sorun şu ki, tamamen belirsizler ve

ölçülemez ve bu nedenle memnuniyetleri kanıtlanamaz.

Örneğin, "tamamen güvenilir" ne anlama geliyor?

Herhangi bir keyfi kişi, belirli bir sistem için güvenilirlik için farklı bir anlama sahip olacaktır. Modülerlik (yazılımda)

belirli bir anlam, ama nasıl ölçülebilir? "hızlı" ne demek olarak hızlı

tren? Bir mermiden daha hızlı? Bu gereksinim çok belirsizdir. Ve sonunda

“Hatalar %99'dan az olacak” bir dava reçetesidir. neyin %99'u? Ne üzerine

zaman aralığı?

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 kusurlu, çünkü bazı önemli detaylar olabilir.

eksik—varsa, bağlamın dışında neyin eksik olduğunu gerçekten bilemeyiz.

SRS belgesinin geri kalanı. Glintz'in öznitelik şablonunu kullanmak (Şekil 4.4) bir

bu soruna iyi bir çözüm.

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) ayrıca gereksinimleri yazarken aşağıdakilerden kaçınmayı önerir:

belge:

■ Saçma sapan ifadeler

■ “Gerekirse” gibi riskten korunma maddeleri

■ “ve”, “veya” ve “ama” gibi bağlaçlar

■ tek bir ifadede birden fazla gereksinime yol açabilir

İhtiyaç Belgesinin Yazılması ■ 113

■ “Belki” ve “bazen” gibi spekülatif terimler

■ “Kullanıcı dostu” ve “genel olarak” gibi muğlak terimler kullanılamayacağından

doğrulandı.

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) paylaşılan 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 maksatlı gereksinim mühendisliği, tehlike yaklaşımlarına 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 gereksinimleri mühendisliğinin kaldıraç etkisi gerçekleştirilmemiştir.

Araç kullanımı projeler ve mühendislik sahaları arasında tutarsızdır, bu nedenle yeniden kullanılabilirlik, değişiklik takibi ve kaçırılan veya aşırı gereksinimlerin belirlenmesinin gücü kaybolur. Bir tane olur

büyük bir organizasyonun işleri daha iyi yapmasını beklemek, ancak bu şirket için sorunun bir kısmı, şirketin ilk günlerinde mevcut hiçbir aracın olmaması ve eski mühendislerin ürün hattı mühendisliği potansiyelini ve ihtiyacı fark etmemiş olmalarıdır.

işlerin yapılma şeklini değiştirmek. 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 kullanabilir. Kurumsal ve projeli bir “metafor”

mühendislerin hepsinin “aynı sayfada” olması için oluşturulmalıdır. Katı ürün hattı mühendisliği uygulamaları o zaman mümkün olacaktır. Az önce açıklanan durum büyük şirketlere özgü değildir.

Yazılım ve Sistemler için Gereksinim Mühendisliği

Exercies

Hangi koşullar altında bir SRS'yi yalnızca gayri resmi teknikler kullanarak temsil etmek uygun olur?

Davranış spesifikasyonu, bir gereksinim belgesinin ne olmasını sağlayabilir?

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

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