GİRİŞ----

Biz 5. bölümde zaten NASA ARM aracılığıyla otomatikleştirilmiş gereksinim analizini ve ilgili gereksinim doğrulama ve doğrulama araçlarını tartışmıştık. Kelime işlemciler, veritabanı yöneticileri, elektronik tablolar, içerik analizörleri, kavram haritalama programları, otomatikleştirilmiş gereksinim denetleyicileri ve benzerleri de gereksinim mühendisinin ilgilendiği araçlardır.

Tablo 8.1 de gördüğümüz gibi gereksinim yönetimi için kullanılan geleneksel ofis araçlarının yeteneklerini, zengin özelliklere sahip bir gereksinim yönetimi aracıyla karşılaştırır. Ancak en ünlü gereksinim araçları, yüksek düzeyde entegre işlevsellik sağlayan büyük, ticari veya açık kaynaklı paketlerdir. Bu gereksinim yönetimi araçlarının başlıca işlevi, gereksinimler için kaynaklar, gereksinimler, kullanım senaryoları, senaryolar ve gereksinim mühendisliği yaşam döngüsü boyunca, kullanıcı öyküleri gibi tüm "tipik" gereksinim mühendislik nesnelerini temsil etmek ve düzenlemektir. Kullanıcı tanımlı varlıklar için de genellikle destek sağlanır. Bu büyük, ticari gereksinim mühendisliği araçları için diğer tipik işlevler şunları içerir:

- Çoklu kullanıcı desteği ve sürüm kontrolü
- Çevrimiçi işbirliği desteği
- Özelleştirilebilir kullanıcı arayüzleri
- Standart şablonlar için yerleşik destek (IEEE 29148 gibi)
- Doğrulama ve doğrulama araçları
- Programlanabilir bir arayüz aracılığıyla özelleştirilebilir işlevsellik
- İzlenebilirlik desteği
- Kullanıcı tanımlı sözlük desteği (Heindl et al. 2006)
- Yazılım ve Sistemler için Gereksinim Mühendisliği

Tablo 8.1 Gereksinimler Deposu Metrik Yeteneklerinden Uyarlanan Çekiç ve ark. (1998)

	Kelime İşlemci	E-tablo	İlişkisel Veritabanı	Gereksinim Yönetim Aracı
Belge Boyutu	EVET	HAYIR	HAYIR	Önceden biçimlendirilmiş durumda değil
Zaman İçinde Dinamik Değişiklikler	Karmaşık değişiklik izleme etkinken mümkün	HAYIR	HAYIR	EVET
Yayın Boyutu	EVET	EVET	EVET	EVET
Gereksinim Genişletme Profili	HAYIR	HAYIR	EVET	EVET
Gereksinim Doğrulama	HAYIR	HAYIR	Mümkün	Mümkün
Gereksinim Oynaklığı	EVET	EVET	EVET	EVET
Test kapsamı	EVET	Karmaşık denklem mantığıyla mümkün	EVET	EVET
Test Aralığı		Karmaşık denklem mantığıyla mümkün	EVET	EVET
Test Türleri	EVET	EVET	EVET	EVET

Doğrulama ve doğrulama özellikleri, herhangi bir otomatikleştirilmiş gereksinim mühendisliği aracının önemli bir bileşenidir. Gerçekten de, daha karmaşık ticari gereksinim mühendisliği araçları, diğer gereksinimleri kontrol etme, izleme ve arşivleme özellikleri sağlar. Bunlar Tablo 8.2'de gösterilmiştir.

Bu özellikler özellikle önemlidir çünkü zamanla artefaktların doğru bir şekilde izlenmesini sağlarlar. İzlenebilirlik, SRS belgesinin önemli bir özelliğidir ve daha fazla tartışmayı gerektirir.

Tablo 8.2 Otomatikleştirilmiş Gereksinimler Mühendislik Aracı Özellikleri (Heindl 2006)

Araç Özelliği	Tanım
Gereksinimler için iş akışının tanımı	Gereksinimler için bir iş akışı (durumlar, roller, durum geçişleri) yapılandırılabilir.
İzlerin çift yönlülüğünün otomatik olarak oluşturulması	Kullanıcı, yapı A ve yapı B arasında bir iz oluşturduğunda, otomatik olarak B'den A'ya geriye doğru bir iz oluşturur.
Kullanıcıya özel izleme türlerinin tanımı	Yetkili bir kullanıcı, izleme türleri tanımlayabilir ve adlar atayabilir.
Şüpheli izler	Bir gereksinim değiştiğinde araç, izlemeleri kontrol etmek ve güncellemek için bu gereksinimle ilgili tüm izleri otomatik olarak vurgular.
Uzun süreli arşivleme işlevi	Araçtaki tüm veriler, gerektiğinde araç olmadan erişilebilen bir formatta arşivlenebilir.

İzlenebilirlik Desteği

İzlenebilirlik, gereksinimler arasındaki ilişkilerle ilgilidir. Kaynakları, ve çok sayıda diğer eserler, Üç tür izlenebilirlik tanımlıyoruz kaynak izlenebilirliği, gereksinimleri teklif eden paydaşlara bağlayan bu gereksinimler gereksinim izlenebilirliği bağımlı arasındaki bağlantılar gereksinimler; ve tasarım izlenebilirliği, gereksinimlerden tasarıma bağlantılar sağlar.

İzlenebilirliğin başka türleri de vardır, örneğin, gereksinimlerden kaynağa kod, kaynak kodundan test senaryolarına ve test senaryolarından gereksinimlere kadar. Bu tür bir gidiş-dönüş izlenebilirliği, özellikle kritik görev sistemlerinde önemlidir.

Örneğin, standart DO-178C, Airborne'da Yazılımla İlgili Hususlar Sistem ve Ekipman Belgelendirmesi , ABD'deki federal havacılık düzenleme kurumları tarafından kullanılan, Kanada ve başka yerlerde, eser izlenebilirliği için kurallar bulunur. DO-178C, tüm düşük seviyeli gereksinimler arasında izlenebilirlik gerektirir ve ebeveyn üst düzey gereksinimleri. Kaynaklar arasında bağlantılar da zorunludur kod öğeleri, gereksinimleri ve test durumları. Tüm yazılım bileşenleri olmalıdır bir gereksinime bağlı, yani, tüm öğelerin gerekli bir amacı olmalıdır (yani, altın kaplama yok). Daha sonra bunların sağlanması için sertifikasyon faaliyetleri yürütülür. kurallara uyulur (RTCA 2011). Ama amaçlarımız için, we are only concerned with traceability artifacts found in the requirements specification document (or ancillary documents).

Yazılım ve Sistemler için Gereksinim Mühendisliği

SRS belgesi içinde izlenebilirlik arasındaki karşılıklı ilişkiye odaklanır gereksinimler ve kaynakları paydaşlar, standartlar, yönetmelikler vb.

Bu amaçlar için, izlenebilir eserler şunları içerir:

- Gereklilik
- Bir gereksinimle ilişkili paydaşlar
- Gerekliliği zorunlu kılan standart, düzenleme veya yasa
- Gerekçe (gereksinim için)
- Anahtar kelimeler (aramak için)

Bunların ve diğer artefaktların çeşitli kombinasyonları, birçok izlenebilirlik matrisi formatına yol açar. bunlardan birkaçı birazdan sunulacak ayrıca gereksinimler arasındaki bağlantıları koruyan izlenebilirlik matrisleri oluşturmayı önermek elemanlar ve analiz modeli elemanları.

Kapsamlı bir gereksinim tartışması izleme yaklaşımları ve uygulamaları Lee ve ark. (2011).

Gereksinimler Bağlantı İzlenebilirlik Matrisi

Bir gereksinimin başka bir gereksinime "kullanımlar" veya "başvuru" ilişkisi olarak açık bir atıfta bulunması alışılmadık bir durum değildir.

Aşağıdaki gereksinimi göz önünde bulundurun:

- Sistem, ısıtma sisteminin kontrolünü sağlayacak ve zaman çizelgeleme fonksiyonuna göre çizelgelenebilir olacaktır.

hangi bir "başvuru" ilişkisini gösterir. “Kullanımlar” ilişkisi şu şekilde gösterilmektedir:

aşağıdaki gereksinim:

Sistem, ısıtma sisteminin uygun şekilde kontrol edilmesini sağlamalıdır.

gereklilik 3.2.2'de gösterilen zaman çizelgesi ve güvenlik özellikleri ile

gereksinim 4.1'de açıklanmıştır. "Kullanımlar" ve "belirtilenler" arasındaki temel ayrım, "kullanımlar"ın bir daha güçlü, doğrudan bağlantı.

İzlenebilirliğin birincil eseri, gereksinim izlenebilirlik matrisidir. Gereksinim izlenebilirlik matrisi, SRS belgesinde tablo şeklinde görünebilir, bağımsız bir izlenebilirlik belgesinde,

veya dahili olarak temsil edilen bir modifikasyon aracı, görüntüleyin ve yazdırın.

Hangi eserlerin dahil edileceğine bağlı olarak birçok izlenebilirlik matrisi varyasyonu vardır.

Bir matris formu, gereksinimler arasındaki karşılıklı ilişkiyi gösterir. Burada, gereksinim izlenebilirlik matrisindeki girişler aşağıdaki gibi tanımlanır:

Rij = R eğer gereksinim i, gereksinim j'ye atıfta bulunuyorsa ("belirtilen" anlamına gelir)

bilgi amaçlı).

Gereksinim Mühendisliği için Araç Desteği

Rij = U, eğer gereksinim i, gereksinim j'yi kullanırsa (doğrudan “bağlıdır” anlamına gelir).

Rij= aksi halde boş. Bir gereksinim, başka bir gereksinimi hem kullandığında hem de referans gösterdiğinde, giriş "U", "R"nin yerini alır. Öz referanslar dahil edilmediğinden, köşegeni

matris her zaman boş girişler içerir. Ayrıca gereksinimlerde dairesel referanslar beklemeyiz, böylece Rij = U veya Rij = R ise Rji'nin olmasını bekleriz. boşluk. Aslında, gereksinim mühendisliği aracında otomatik doğrulama özelliği bu tür dairesel referansları işaretlemelidir. Tipik bir izlenebilirlik gereksinimleri matrisinin formatı Tablo 8.3'te gösterilmektedir.

Örnek girişleri olan varsayımsal bir sistem için kısmi izlenebilirlik matrisi

Tablo 8.4'te gösterilmiştir.

R genellikle seyrek olduğundan, yalnızca bu satırları ve sütunları listelemek uygundur

ki boş değil. Örneğin, bulunan akıllı ev için SRS belgesi için ekte, gereksinimlerden açıkça türetilen izlenebilirlik matrisi şu şekildedir: Tablo 8.5'te gösterilmiştir.

Bu türden bir seyrek izlenebilirlik matrisi, düşük düzeyde açık bir bağlantı olduğunu gösterir.

gereksinimleri arasında. Gereksinim belgesinde düşük düzeyde bağlantı

arzu edilir—bağlantılı gereksinimler ne kadar fazlaysa, bir gereksinimde o kadar fazla değişiklik olur

başkalarına yaymak. Aslında, açık gereksinimler bağlantıları,

Tablo 8.4 Hayali Bir Sistem Spesifikasyonu için Kısmi İzlenebilirlik Matrisi

Gereksinim Kimliği	1	1.1	1.1.1	1.1.2	1.2	1.2.1	1.2.2	2	2.1	2.1.1	3
1									R	R	
1.1			U								
1.1.1	R							R			
1.1.2											
1.2						R				U	
1.2.1							R				
1.2.2										R	
2	U								R		
2.1				U							
2.1.1											R
3					R						

Tablo 8.5 Akıllı Ev için İzlenebilirlik Matrisi Ekte Gösterilen Gereksinimler Spesifikasyonu

Gereksinim Kimliği	5.11	9.11
9.1.1	R	
9.10.7		R

Endişelerin ayrılması ilkesi, yazılım mühendisliğinin temel bir ilkesidir. Bu nedenle, bağlantı gereksinimleri yalnızca kesinlikle gerekli olduğunda bağlayın.

Genel olarak, her gereksinimin birden fazla test senaryosu tarafından test edilmesini isteriz. Aynı zamanda, her bir test senaryosunun birden fazla gereksinimi yerine getirmesini isteriz. "Test süresi" metrikleri, test planını karakterize etmek ve yetersiz veya aşırı testi belirlemek için kullanılır:

-Her testin gereksinimleri.

-Her gereksinimin testleri.

Bu metrikler için uygun istatistikleri belirlemek için araştırmalar halen devam etmektedir. Ancak en azından tutarsızlıkları ve tek biçimli olmayan test kapsamını aramak için bu ölçümleri kullanabilirsiniz. Elbette, testin kapsamlılığı ile testin süresi ve maliyeti arasında her zaman bir denge vardır. Ancak test bu kitabın konusu değil.

205 * Gereksinim Mühendisliği için Araç Desteği.

Gereksinimler Kaynak İzlenebilirlik Matrisi.

Yine bir başka tür izlenebilirlik matrisi, gereksinimleri kaynaklarına bağlar.

Doğrudan kullanıcılardan gelenlerin yanı sıra, birçok gereksinim hükümet düzenlemelerinden ve standartlardan türetilmiştir.

Gereksinimleri bu kaynaklara bağlamak, kaynaklar değiştiğinde çok yardımcı olabilir.

Tablo 8.6, böyle bir izlenebilirlik matrisi için tipik formatı göstermektedir.

Bu tür izlenebilirlik matrisi, özellikle işlevsel olmayan gereksinimlerin izlenmesi için kullanışlıdır.

Proje yaşam döngüsü boyunca işlevsel olmayan gereksinimlerin izlenememesi önemli bir sorun olabilir (Kassab ve Ormandjieva 2014).

Gereklilikler Paydaş İzlenebilirlik Matrisi.

Başka bir izlenebilirlik matrisi türü, paydaşları sundukları gereksinimlere bağlar; bunun bir örneği Tablo 8.7'de gösterilmektedir.

Tablo 8.7 ayrıca, gereksinim görüşmesini ve takas analizini büyük ölçüde kolaylaştıran bir sıralama şeması içermektedir.

Önceki bölümlerde yalnızca üç tür izlenebilirlik matrisi gösterilmiştir, ancak bunların herhangi bir varyasyonu veya kombinasyonu oluşturulabilir.

Tablo 8.6 Gereksinimleri ve Kaynaklarını Gösteren İzlenebilirlik Matrisi

Gereksinim Kimliği	Federal Düzenleme #1	Federal Düzenleme #2	Belirtmek Düzenleme #1	Belirtmek Düzenleme #2	Uluslararası Standart #1
3.1.1.3	X				
3.1.2.9	X	X			
3.2.1.8			X	X	
3.2.2.5			X		
3.2.2.6					X
3.3.1		X			
3.3.2		X			
3.4.1		X			
3.4.3		X			
3.4.4		X			
3.6.5.1				X	
3.6.6.4					X

Tablo 8.7 Gereksinimleri, Sıralamalarını ve Paydaş Kaynaklarını Gösteren İzlenebilirlik Matrisi.

Gereksinim	Rütbe 1 (en düşük önem) – 5(en yüksek önem)	Paydaş Kaynak D – Doktor, H – Hemşire, A – İdari Destek Personeli, HA – Hasta, R – Regülatör
3.1.1.1	5	R
3.1.1.2	4	R
3.1.2.1	5	D,H,İ
3.1.2.2	3	D,H,İ

bahsedilen diğer izlenebilirlik yapılarını dahil edin. Ticari gereksinim mühendisliği araçları, kullanıcının her türlü izlenebilirlik matrisini özelleştirmesine olanak tanır.

Gereksinim Yönetimi araçları

Gereksinim araçları, bir projenin yaşam döngüsü boyunca gereksinim bilgilerinin yönetilmesinde son derece önemlidir. Örneğin, en yaygın kullanılan gereksinim yönetimi araçlarından biri olan IBM'in kapılarını (dinamik nesne yönelimli gereksinim sistemi) göz önünde bulundurun. KAPILAR aslında gereksinimleri yönetmek için bir nesne veritabanıdır. Diğer bir deyişle, her özellik bir özellik açıklaması, özellik grafiği ve kullanım durumu diyagramı içeren bir nesne olarak temsil edilir. Veritabanı ayrıca klasörler ve projeler oluşturularak düzenlenir ve tüm modül ve nesne düzeyi eylemlerinin geçmişi korunur. KAPILARI, UML modellerinin KAPILARIN içine gömülmesini ve izlenmesini sağlayan başka bir araçla bütünleştiren Analyst adlı bir uzantı vardır (Hull ve ark. 2011). Özellik öznitelikleri zorunlu, isteğe bağlı, tek bağdaştırıcı ve çoklu bağdaştırıcıyı içerir ve kullanıcı gereksinimlerin tamamlayıcı ve destekleyici niteliğini içeren bir özniteliğin gerekli mi yoksa hariç mi olduğunu belirtebilir. Diğer nitelikler kullanım durumu paketlerine ve ürün örneklerine dayanır (Eriksson ve ark. 2005). DOORS, tam izlenebilirlik ve eksik bağlantı analizi için bir projedeki tüm nesneler arasında bağlantı sunar. Araç, C benzeri bir programlama dili ile özelleştirilebilir. ISO 12207, ISO 6592 ve diğer standartlara (Volere 2013) uygun olarak gereksinimleri yapılandırmak için standart şablonlar mevcuttur. Doors'un en son sürümünde daha birçok gereksinim analizi ve yönetimi özelliği bulunmaktadır. DOORS Next Generation (NG) adı verilen Doors'un güncel sürümü (yazının yazıldığı tarih itibariyle) hakkında daha fazla bilgi için lütfen Ek F, DOORS Ng'ye başvurun.

Araç Değerlendirmesi

Birçok ticari ve açık kaynaklı araç vardır ve işletmeye almadan önce herhangi bir aracın özelliklerini dikkatlice değerlendirmek önemlidir. ISO / IEC TR 24766 (2009) Bilgi Teknolojisi—Sistemler ve Yazılım Mühendisliği— Gereksinimler için Kılavuz Mühendislik Aracı Yetenekleri, gereksinim mühendisliğinin yeteneklerini değerlendirmek için bir çerçevedir. Bu yetenekler aşağıdaki alanlarda düzenlenir: seçim, analiz, şartname, doğrulama ve doğrulama, yönetim ve diğer yetenekler.

ISO/IEC TR 24766 çerçevesi kullanılarak ticari araçlarla ilgili çeşitli çalışmalar yapılmıştır (örn., Carrillo de Gea ve ark. 2011, 2015; Daud ve diğ. 2014). Bu çalışmalar genel olarak alet pazarının hızla değiştiğini ve aletlerin giderek daha karmaşık ve kullanımı zorlaştığını ortaya koymuştur. Pahalı ticari araçların karmaşıklığı daha sonra ucuz araçların ortaya çıkması için fırsatlar yaratır, ancak karmaşık özellikler sunmaz. Ayrıca, bu çalışmalar, araçların çoğunda tutarlılık, doğruluk ve bütünlük gibi doğrulama işlevlerinin hala eksik olduğunu göstermiştir. Sud ve Arthur (2003) bir dizi gereksinim yönetimi aracını aşağıdaki boyutları kullanarak değerlendirdi:

- Gereksinimleri izlenebilirlik mekanizması
- ihtiyaç analizi mekanizması
- Güvenlik ve erişilebilirlik mekanizması
- Taşınabilirlik ve arka uç uyumluluk
- Yapılandırma yönetim anlayışı
- İletişimi ve işbirliği mekanizması

- Değişim yönetimi destek
- Çevrimiçi yayımlama desteklemek gibi bir kelime işlemci uyumluluk
- SRS belgeleri biçimi olarak Kullanılabilirlik özellikleri
- , bu değerlendirme boyutları, mühendisler tarafından kabul edilmeden önce çeşitli ticari ve açık kaynak gereksinimleri yönetim araçlarını karşılaştırmak için kullanılabilir. Örneğin, bu boyutlar kullanılarak basit bir kontrol listesi oluşturulabilir veya Likert ölçeğine dayalı bir değerlendirme yapılabilir

Açık kaynak gereksinimi mühendislik araçları

Birçoğu tam özellikli gereksinim yönetimi araçları olan yüz binlerce açık kaynak projesi vardır ve bunları satın almadan veya sıfırdan geliştirmeye çalışmadan önce araçları aramak için önce açık kaynak depolarına yönelmeniz önerilir. Gereksinim mühendisliği için yardımcı programlar veya kaynaklar da vardır.

Özgür düşünceler

Zihin haritalama araçları, kullanıcının şeyler ve fikirler arasındaki ilişkiyi betimleyen kavram haritaları oluşturmasına olanak tanır ve beyin fırtınası, görev analizi, merdivenleme, izlenebilirlik, JAD, odak grupları ve diğer birçok gereksinim mühendisliği faaliyeti için gereksinim mühendisi için yararlıdır. Kavram haritalama araçları, kavram haritalarının kolayca oluşturulmasını sağlar ve yukarıdaki tartışma açıklayıcı amaçlar için belirli bir araca dayanırken, herhangi bir kavram haritalama aracı gereksinim mühendisi tarafından kullanılabilir. FreeMind, Java ile yazılmış açık kaynaklı bir zihin haritalama aracıdır. (FreeMind 2017). Araç, kullanımı kolay bir grafik kullanıcı arayüzüne sahiptir. Kavramlar, birincil bir konsepte dayanan hiyerarşik bir yapıda eşlenir ve fikirleri bu konsepte tabi tutar. Kullanışlı simgeler, her fikir için öncelikleri, önemli kriterleri veya tehlikeleri işaretlemek için kullanılabilir. Zihin haritasının içeriğini zenginleştirmek için başka birçok simge mevcuttur. Gereksinim mühendisinin zihin haritalarının kullanımını göstermek için Phil'in akıllı ev kontrol sistemi kavramına bakalım. Bir akıllı evin temel kavramından başlayarak Phil, Şekil 8.1'de gösterildiği gibi bir üst düğüm (bir balonla temsil edilir) oluşturur. Ardından, Phil aracı sisteme (güvenlik) bir özellik eklemek ve bu özelliğe öncelik atamak için kullanır. Yedi seviyeye kadar sıralama mevcut olsa da, Phil basit bir üç seviyeli sistem seçti.

1. Zorunlu

2. İsteğe bağlı

3. Elde edilen sıralama özelliğine ve güncellenmiş zihin haritasına sahip olmak güzel Şekil 8.2'de gösterilmiştir. Phil daha sonra bu özelliğe ayrıntılar ekler. Örneğin, akıllı evin güvenlik sisteminin evdeki mevcut güvenlik sistemini kullanmasını ve onunla çalışmasını istiyor.

İstediği bir diğer özellik ise HVAC (ısıtma, havalandırma ve klima) sisteminin akıllı ev sistemi ile bütünleşmesidir. Ek olarak, Phil'in evde narin bitkileri, evcil hayvanları ve koleksiyonları olduğu için, sıcaklığın asla 100 ° F'yi aşmaması önemlidir, bu nedenle bu tehlike bir bomba simgesiyle işaretlenmiştir. Gözden geçirilmiş zihin haritası Şekil 8.3'te gösterilmiştir. Phil, özelliklerin beyin fırtınası, müzik yönetimi, çim bakımı ve telefon yanıtlama sistemi ekleyerek devam ediyor. Bazı özellikler uygun sembolle işaretlenmiş önemli detaylarını içerir (Şekil 8.4). Sistemin görsel anlamda

gerçekleştiğini gören Phil, daha fazla özellik düşünüyor ve uygun önceliklendirme kararlarını verebiliyor. Phil, Şekil 8.5'te görüldüğü gibi zihin haritasına buna göre ekler. Şekil 8.5'te gösterilen zihin haritası tamamlanmamıştır. Ancak bu zihin haritası, gereksinim mühendisleriyle elikitasyon tartışmaları sırasında kullanılabilir ve zaman içinde kolayca geliştirilebilir. FreeMind, zihin haritasını metindeki hiyerarşik yapıya dönüştürmenize izin veren güzel bir özelliğe sahiptir. Böylece, diyagramın tamamını seçip kopyalayıp bir Sözcük işlemcisine yapıştırırsanız, iç içe geçmiş bir metin listesi elde edersiniz. Daha sonra gereksinim belirtimi belgesinde kullanılmak üzere hiyerarşik numaralı bir liste elde etmek için sayı ekleyebilirsiniz. FreeMind, fikirleri yazmanıza ve bunları çok hızlı bir şekilde yeniden düzenlemenize olanak tanır ve gereksinim belirtimi belgesini başlatmak için harika bir araçtır. Birçok taşınabilir aygıtta çalışan FreeMindPDA adlı bir sürüm de vardır

Tabii ki, Freemind'e benzer işlevselliğe sahip başka açık kaynaklı ve ticari zihin haritalama araçları da var. Bu araçlardan herhangi biri, gereksinim belirtiminin yazılmasında ve gereksinimlerin açıklanmasında çok değerli olabilir.

FitNesse

FitNesse, etkileşimli test senaryoları oluşturmak için bir framework sağlayan açık kaynaklı, wiki tabanlı bir yazılım işbirliği aracıdır. FitNesse başlangıçta bir test aracı olarak geliştirilmiş olsa da, bizim ilgi alanımız onu örneğin çevik bir ortamda entegre gereksinimler/test spesifikasyonları için kullanmaktır.

FitNesse'nin nasıl çalıştığını kısaca açıklamadan önce, bir FitNesse test senaryosunun neye benzediğini incelemek uygun olacaktır. Test durumu, esasen, test edilecek işlevin (veya yöntemin) adından, girdi parametreleri kümesinden ve karşılık gelen beklenen sonuç parametrelerinden ve ardından tipik test durumlarından oluşan bir tablodur. (bkz. Tablo 8.8).

FitNesse tablosu yürütülebilir bir tablodur; gerçek kod geliştirildiğinde, bir düğmeye tıklandıktan sonra ilgili işlevin sonuçları hesaplanır.

Şimdi SRS'yi kullanarak akıllı ev sistemi için durumu gösterelim.
ek. Bu SRS'de, Bölüm 9.2.2 şunları belirtir:
9.2.2 Sistem, kullanıcı tarafından tanımlanan herhangi bir sürede kahve makinesini başlatmalıdır. Su mevcut olduğundan kahve çekirdeği seviyeleri yeterlidir ve üniteye güç verilir.

Table 8.8 Typical FitNesse Requirement/Test Case Format

Name of Function							
Input 1	*Input 2*	...	*Input n*	*Expected Result 1*	*Expected Result 2*	...	*Expected Result m*

Ön koşulların karşılandığını varsayarsak (kahve çekirdekleri ve su mevcuttur), kahve yapmak için makul bir program Tablo 8.9'da verildiği gibi olabilir.Tablo 8.9'da gösterilen program, ev sahibinin uyumayı sevdiğini kabul eder. Hafta sonları geç. Bu tabloyu farklı şekillerde doldurabiliriz. Artık arzu edilen işlevselliğin dinamik zamanlama için olduğu doğrudur, böylece örneğin, bir tatil haftasında ev sahibi her gün geç saatlere kadar uyuyabilir. Ancak buradaki tablo, uygulama planlamasının bir senaryosunu belirtmek için kullanılır. Aslında, FitNesse çerçevesi aracılığıyla düzgün bir şekilde yapılandırılırsa, Tablo 8.9, bitmiş sistem için yürütülebilir bir test durumu oluştururFazla ayrıntıya girmeden, FitNesse'nin kabaca nasıl çalıştığı aşağıda açıklanmıştır. FitNesse, Fit framework'ünün üzerine inşa edilmiş bir wiki GUI'dir. Fit, test tablosuna karşılık gelen bir Java veya C# sınıfı) fikstür kodunu çalıştırır. Örneğin, Tablo 8.9'un en üst satırında Activate_Coffee_Pot çağrılacak gerçek sınıfı belirtir. FitNesse test başarısız olursa, geçerse veya bir istisnası atılırsa, beklenen sonuç hücrelerini sırasıyla kırmızı, yeşil veya sarı renklendirir (Gandhi ve ark. 2005).

Table 8.9 FitNesse Requirements Specification/Test Case for Activate Coffee Pot Function in Smart Home System

Activate_Coffee_Pot		
Day	*Time*	*Output*
Monday	7:00	coffee_pot_on_signal=high
Monday	8:00	coffee_pot_on_signal=low
Tuesday	7:00	coffee_pot_on_signal=high
Tuesday	8:00	coffee_pot_on_signal=low
Wednesday	7:00	coffee_pot_on_signal=high
Wednesday	8:00	coffee_pot_on_signal=low
Thursday	7:00	coffee_pot_on_signal=high
Thursday	8:00	coffee_pot_on_signal=low
Friday	7:00	coffee_pot_on_signal=high
Friday	8:00	coffee_pot_on_signal=low
Saturday	9:00	coffee_pot_on_signal=high
Saturday	10:00	coffee_pot_on_signal=low
Sunday	10:00	coffee_pot_on_signal=high
Sunday	11:00	coffee_pot_on_signal=low

Fitnesse'a işlevsellik açısından benzeyen başka açık kaynaklı araçlar da var. Örneğin, Cucumber, Scrum ortamlarında (cucumber.io) yaygın olarak kullanılan bir araçtır.

Gereksinim Mühendisliği Aracı En İyi Uygulamaları Kullandığınız gereksinim mühendisliği araçları ne olursa olsun, aracı akıllıca kullanmak ve belirli en iyi uygulamaları takip etmek uygundur.

Cleland-Huang ve diğerleri, bu tür uygulamaların mükemmel bir kümesini sunmaktadır. (2007):

■ Bir amaç için iz. Yani, hangi bağlantıların gerçekten önemli olduğunu belirleyin; aksi takdirde, çok sayıda yabancı bağlantı oluşturulacaktır.

■ Uygun bir iz ayrıntı düzeyi tanımlayın. Örneğin, bağlantılar uygun paket, sınıf veya yöntem düzeyine yerleştirilmelidir.

■ Yerinde izlenebilirliği destekleyin. Öğeler kendi yerel ortamlarında bulundukları için bunlar arasında izlenebilirlik sağlayın.

■ İyi tanımlanmış bir proje sözlüğü kullanın. İlk keşif sırasında sözlüğü oluşturun paydaşlarla toplantılar ve mühendislik süreci boyunca tutarlı bir şekilde kullanın.

■ Kalite gereksinimlerini yazın. İzlenebilirlik için özellikle önemli olan IEEE 29148 gibi genel kabul görmüş en iyi uygulamaları uyguladığınızdan emin olun.

■ Anlamlı bir hiyerarşi oluşturun. Deneysel sonuçlar, hiyerarşik olarak düzenlenmiş gereksinimlerin akıllı bağlantı yazılımına daha duyarlı olduğunu gösteriyor.

▪ Etki alanı içi anlamsal boşluğu kapatın. Örneğin, aşırı yüklenmiş terminolojiden, yani iki farklı bağlamda tamamen farklı anlama gelen sözcüklerden kaçının.

▪ Zengin içerik oluşturun. Gereksinimleri ve alan bilgisini her gereksinim ile birleştirin.

▪ Son olarak, Gereksinim mühendisliği sürecini iyileştirmek için bir süreç iyileştirme planı kullandığınızdan emin olun. Disiplinli uygulamaların izlenmesi, araç kullanımından ve süreçlerin iyileştirilebileceği bir çerçeveden daha iyi sonuçlar alınmasına neden olabilir. Her proje planı, kullanılacak araçların tanımını ve bunların nasıl kullanılacağını içermelidir.

Çıkarma Destek Teknolojileri

Bu bölümü, daha önce tartışılan çeşitli gereksinimleri ortaya çıkarma süreçleri ve tekniklerini desteklemek için kullanılabilecek bazı teknolojilere değinerek kapatıyoruz.
Bu teknolojiler şunları içerir:

▪ Wikiler

▪ Mobil teknolojiler

▪ Sanal ortamlar

▪ İçerik analizi

FreeMind, gereksinim belirtim belgesini ortaya çıkarmak ve yazmak için bir araç olarak zaten tartışılmıştı.

Gereksinim Ortaya Çıkarma için Wiki'leri Kullanma

Wiki'ler, kullanıcıların bir web sitesine metin ve görüntüleri biçimlendirip gönderebildiği ortak bir teknolojidir. Erişim kontrolü, parola koruması ve semafor benzeri koruma mekanizmalarıyla sağlanır (yani, sitenin belirli bir sayfasına herhangi bir anda yalnızca bir kullanıcı yazabilir).

Wiki'ler kullanılabilir. işbirliği için, örneğin grup çalışmasını kolaylaştırmak için, kart girişi (kart sıralama için), şablon tamamlamaları, anketler ve gereksinim belgesini düzenlemek için. Ayrıca, wiki tabanlı gereksinimler doğrudan yayınlama araçlarına ve doğrulama araçlarına aktarılabilir. Birkaç araştırmacı, çeşitli gereksinim etkinlikleri için wiki tabanlı gereksinim tekniklerini başarıyla kullandı paydaş işbirliği (Ferreira ve da Silva 2008), gereksinimler müzakere (Yang ve diğerleri 2008) ve dağıtılmış açıklama (Liang ve diğerleri 2009) dahil.

Ayrıca, wiki'ler etkileşimli oluşturmak için kullanılabilir. test senaryolarını otomatikleştirmeye yardımcı olabilecek belgeler (gereksinim belirtimlerinin tüm gereksinimler için kabul kriterleri içermesi gerektiğini unutmayın). Örneğin, FitNesse, bir Fit çerçevesi üzerine inşa edilmiş ücretsiz, wiki tabanlı bir yazılım işbirliği GUI'sidir. FitNesse, gömülü, etkileşimli test senaryoları ile Web tabanlı gereksinim belgeleri oluşturmak için bir framework sağlar (FitNesse 2007). wiki'ler, test senaryolarını otomatikleştirmeye yardımcı olabilecek etkileşimli belgeler oluşturmak için kullanılabilir (gereksinim belirtimlerinin tüm gereksinimler için kabul kriterleri içermesi gerektiğini hatırlayın). Örneğin, FitNesse, bir Fit frameworkü üzerine inşa edilmiş ücretsiz, wiki tabanlı bir yazılım işbirliği GUI'sidir. FitNesse, gömülü, etkileşimli test senaryoları ile Web tabanlı gereksinim belgeleri oluşturmak için bir framework sağlar (FitNesse 2007). wiki'ler, test senaryolarını otomatikleştirmeye yardımcı olabilecek etkileşimli belgeler oluşturmak için kullanılabilir.

Mobil Teknolojiler

Gereksinim bilgilerini yerinde yakalamak için cep telefonları ve kişisel dijital asistanlar gibi çeşitli mobil teknolojiler kullanılabilir. Örneğin, doktorlar hastalarla çalışırken, yürüttükleri faaliyetlerle ilgili bilgileri doğrudan gereksinim mühendisine iletebilir, bu mühendis yerinde olmak zorunda değildir. Mobil cihazları kullanmak, fikirlerin ve keşiflerin anında kaydedilmesine olanak sağladıklarından özellikle yararlıdır. Böyle bir yaklaşım, beyin fırtınası, senaryo oluşturma, anketler ve diğer birçok standart gereksinim belirleme tekniklerini, müşteri kolayca erişilebilir olmadığında bile (örneğin, açık deniz yazılım geliştirme durumlarında olduğu gibi) destekleyebilir.

Örneğin, Lutz ve diğerleri (2012), kullanıcıların akıllı telefonlar ve tabletler gibi Android özellikli cihazlar aracılığıyla rol oynamasına ve bunu yaparken, paylaşılan bir üzerinde görüntülenen bir temsil modeliyle etkileşime girmesine izin veren CREWSpace adlı bir uygulama geliştirdi. Yazılım, rol yapma durumlarını takip ederek, nesne yönelimli yazılım sistemlerinde kullanılan bir beyin fırtınası aracı olan CRC kartlarını oluşturmayı başardı. CRC kartları daha sonra gereksinimler belgesine dahil edilebilir. yazılım, nesne yönelimli yazılım sistemlerinde kullanılan bir beyin fırtınası aracı olan CRC kartları oluşturmayı başardı. CRC kartları daha sonra gereksinimler belgesine dahil edilebilir. yazılım, nesne yönelimli yazılım sistemlerinde kullanılan bir beyin fırtınası aracı olan CRC kartları oluşturmayı başardı. CRC kartları daha sonra gereksinimler belgesine dahil edilebilir.

SANAL ORTAMLAR

Mimari tasarımcılar, yıllardır ticari bina ve ev inşaatı için gereksinimleri ortaya çıkarmak için sanal izlenecek yollardan yararlandı. Sanal dünya ortamları, gelişmiş grafikler ve dokunsal basınç sensörleri, kuvvet geri besleme çevre birimleri ve üç boyutlu görüntüleme cihazları gibi çeşitli özel cihazlar kullanan daha karmaşık simülasyonlardır. Sanal dünya ortamları, özellikle yeni veya karmaşık uygulama ortamları için gereksinimlerin test edilmesi, doğrulanması ve kabul edilmesinde gerçekçilik sağlamak için kullanılabilir. Russell ve Creighton (2011),
sanal ortamları kullanmanın şunları yapabileceğini öne sürüyor:

- **Mevcut eksiklikleri netleştirin**
- **Potansiyel faydalara dikkat çekin**
- **Öncesi ve sonrası koşullarını gösterin**
- **Çeşitli aktörlerin ve rollerin izlenimlerini kişiselleştirin**
- **Değerlemenin etkili bileşenlerinin daha paylaşılan bir takdirini ve alternatif zaman çizelgelerinin hazır keşfini yaratın.**

Örneğin, havayolu bagaj taşıma sisteminde, Havalimanının ilgili bölümlerinin üç boyutlu bir yerleşim planı oluşturulabilir ve paydaşlar dijital sahneyi geçmek ve daha önce iletilen
gereksinimleri doğrulamak için karakterleri (avatarları) kullanabilir. Sanal. karmaşık sistemler için ortamlar oluşturmak pahalı olabilir. Ancak
yeni ve görev açısından kritik sistemlerde, sanal simülasyonu kullanmanın ince gereksinimleri keşfetmenin faydaları maliyete değebilir.

İÇERIK ANALIZI

İçerik analizi, sosyal bilimlerde yapılandırılmamış bilgiyi yapılandırmak ve anlam bulmak için kullanılan bir tekniktir. Yani, yazı veya yazılı konuşma yapıtlarını analiz ederek, paydaş için önemli olan önemli şeyler elde edilebilir.

Bu yazıları analiz ederken amaç, tekrarlayan temaları belirlemektir(Krippendorff 2004).
İçerik analizi, gereksinim mühendisleri tarafından grup toplantılarından alınan dökümler gibi yazılı eserler içindeki anlamı keşfetmek için kullanılabilir. Yapılandırılmamış görüşmeler, anket verileri veya e-postalar (metne dönüştürülebilen herhangi bir metin veya yapı).

Bu gizli anlamlar, paydaşların doğrudan ifade edemeyecekleri ince gereksinimleri ortaya çıkarabilir. Aslında, bu gizli gereksinimler genellikle ürün teslim edildiğinde en sorunlu ve kullanıcı memnuniyetine yol açması muhtemeldir.
İçerik analizi, etiketlenerek (renkli vurgulayıcılarla) manuel olarak yapılabilir çeşitli yazılarda tekrarlanan aynı kelimeler ve benzer ifadeler . Örneğin, Şekil 8.6, bazı metinlerin örnek içerik analizini içerir.

Metin,bir müşteriyle akıllı bir ev hakkında görüşen bir gereksinim mühendisinin defterinden bir alıntıdır.
Metni okudukça, yinelenen temaları fark etmeye başlıyoruz ve yaptığımız gibi bu yüzden onları farklı renkli bir işaretleyici ile vurgularız. Örneğin, “ben” kelimesi defalarca geçiyor ve her ne sebeple olursa olsun bu ismin önemli bir temayı ima ettiğine karar veriyor ve tek renkle vurguluyoruz. Daha sonra akıllı evin veya "ev"den birkaç kez bahsedilir—bunları önceki temadan farklı bir renkle vurgularız. Zaman kavramından da sıklıkla bahsedilir ("uzun zaman periyotları", "zaman" ve "zaman periyotları") ve bu nedenle bunları tutarlı bir renkle vurgularız. Ve bunun gibi.
Bu süreci otomatikleştirmek için içerik analizi için ücretsiz ve ücretli araçlar kullanılabilir.

GEREKSINIM METRIKLERI

Metrikler, gereksinim mühendisliği süreçlerini ve uygulamalarını izlemek ve iyileştirmek için gereksinim mühendisliği yaşam döngüsünün çeşitli aşamalarında hesaplanabilir veya toplanabilir. En kullanışlı gereksinim yönetimi araçları, metriklerin toplanmasını tamamen otomatik hale getirir veya kısmen yardımcı olur. Costello ve Liu (1995), araçlar tarafından izlenecek aşağıdaki gereksinim metrikleri türlerini açıklar:

- **Uçuculuk**
- **İzlenebilirlik**
- **Tamlık**
- **Kusur yoğunluğu**
- **Hata yoğunluğu**
- **Arayüz tutarlılığı**
- **Sorun raporu ve eylem öğesi sorunları**

Gereksinim değişkenliği metriği, zaman içinde gereksinimlerde yapılan silme, ekleme, değişikliklerin sayısını izler. Açıkçası, bu sayılar araçlar tarafından izlenir. Genellikle proje ömrü döngüsünde daha sonra ortaya çıkan sorunların, nispeten yüksek oynaklığa sahip gereksinimlere göre izlenebilir olması söz konusudur.
İzlenebilirlik metrikleri, gereksinim seviyeleri arasındaki bağlantı istatistiklerini içerir. ARM tarafından toplananlar ve test kapsamı ve yayılma istatistikleri. Test senaryosu kapsamı metriği, birçok gereksinimin test senaryosu olduğunu gösterir ve test süresi ölçüsü, bir test senaryosunun kaç gereksinimi kapsadığını gösterir. Gereksinimler, mimari ve tasarım öğeleri arasında iz bağlantı metrikleri de oluşturulabilir.

IEEE 29148 standart niteliklerin belirttiği gibi, yüksek düzeyde dahili ve harici izlenebilirlik arzu edilir. Tamlık metrikleri, bu tür yer tutucu terimlerin sayısıyla ilgilidir. TBD kullanıldığı için ve Şekil 4.3'te gösterildiği gibi gereksinim ayrıştırma seviyeleriyle ilgili olabilir.
Hata yoğunluğu metrikleri, bir gereksinim denetimi sırasında ortaya çıkarılan gereksinim hatalarının sayısını gösterir. Bu metrikler, ürün ve süreç oynaklığını tahmin etmede ve diğer metrikleri yorumlamada faydalıdır.

Gereksinim hata yoğunluğu, başlangıçta test yürütme veya son test analizi sırasında algılanan gereksinim hatalarının sayısını gösterir. Gereksinim hataları, bir dizi farklı metrik elde etmek için kritikliğe göre sınıflandırılabilir. Bu metrikler testin etkililiğinin belirlenmesinde kullanılır. Hata yoğunluğu metrikleri, ürün/süreç değişkenliğini ve kalitesini tahmin etmede kullanılır ve yorumlamada önemlidir.
diğer metriklerin anlamı (Costello ve Liu 1995).
Son olarak, problem raporu ve eylem öğeleri, projeyi izlemek ve kontrol etmek için kullanılabilen türetilmiş metriklerdir. Örneğin, gereksinim başına kusurlar, gereksinimler dalgalanması (gereksinimlerdeki değişiklikler) ve gereksinim kusuru biriktirme listesi.
Gereksinim yönetimi araçları bu metriklerin çoğunu oluşturabilir.