Sistemler ve Yazılım Dergisi 140 (2018) 74–90

içindekiler listesi ScienceDirect'te mevcuttur

Sistemler ve Yazılım Dergisi

dergi ana sayfası: www.elsevier.com/locate/jss

Nesne yönelimli tasarım ilkelerinin ölçülmesi: Odak grup temelli araştırmanın sonuçları

Johannes Bräuer [a] , Reinhold Plösch [a,*] , Matthias Saft [b] , Christian Körner [b]

[a] Johannes Kepler Üniversitesi, İşletme Bilişimi - Yazılım Mühendisliği Bölümü, Altenbergerstraße 69, Linz 4040, Avusturya
[b] Siemens AG, Kurumsal Teknoloji, Otto-Hahn-Ring 6, Münih 81739, Almanya

Makale info

Makale geçmişi:
24 Ekim 2017 tarihinde alınmıştır
16 Ocak 2018 tarihinde revize edildi
Kabul tarihi: 1 Mart 2018
Çevrimiçi olarak 3 Mart 2018 tarihinde yayınlanmıştır

Anahtar kelimeler:
En iyi tasarım uygulamaları
Tasarım kuralları
Tasarım ilkeleri Yazılım tasarım kalitesi Tasarım iyileştirme

özet

Nesne yönelimli tasarım ilkeleri, önemli tasarım bilgilerini içeren ve iyi tasarım kalitesine odaklanan yazılım yoğun sistemlerin geliştirilmesini destekleyen temel kavramlardır. Bu ilkeler, nesne yönelimli programlama alanındaki ilk adımların atılmasından ve bu programlama paradigmasının sürdürülebilir yazılımlar geliştirmek için en iyi uygulamaların tanınmasından sonra ortaya çıkmıştır. Tasarım ilkeleri yazılım geliştiricileri tarafından bilinmesine rağmen, takip edilecek somut kurallar olmadan bunları pratikte uygulamak zordur. Bu boşluğu fark ettik ve bir dizi tasarım ilkesi için sistematik olarak en iyi tasarım uygulamalarını türettik ve bu uygulamaların otomatik olarak ölçülmesi için araç desteği sağladık. Bu makalenin amacı, en iyi tasarım uygulamaları ile seçilen 10 tasarım ilkesi arasındaki ilişkiyi incelemektir. Bu, tasarım ilkelerinin temel tasarım yönlerinin kapsanıp kapsanmadığını kanıtlamalıdır. Toplamda altı odak grubu ve 31 katılımcı ile odak grup araştırması yaptık. Paralel olarak, her grup beş tasarım ilkesini tartıştı ve Delphi yöntemini kullanarak kapsamı değerlendirdi. Katılımcılar tarafından eklenen ek tasarım uygulamaları önerilerine rağmen, sonuçlar her bir en iyi tasarım uygulamasının tasarım ilkesine etkisini ortaya koyuyor ve tasarım ilkelerinin ana tasarım yönlerinin yaklaşımımızla kapsandığını ve bu nedenle somut tasarım iyileştirme eylemleri türetilmesinin mümkün olduğunu gösteriyor.

1. Giriş

İyi bir yazılım tasarımı diğerlerinden nasıl ayrılır? Bir cevap, birisinin bu konuda özen gösterdiği açıkça belli olmalı olabilir (bkz. Feathers in Martin, 2008). Nesne yönelimli programlama paradigması ve buna eşlik eden diller, yazılım sistemlerinin tasarımı konusunda yazılım geliştiricilere büyük sorumluluk yüklemiştir. Örneğin, soyutlamaları bölme, kapsülleme kullanma ve uygun arayüzler sağlama yöntemleri, büyük ölçüde programcı ve tasarımcının profesyonelliğine bağlıdır. Sonuç olarak, her yazılım geliştiricisi, tasarımı basit ve düzenli bir yazılım çözümü tasarlamaktan sorumludur. Kısacası, bir yazılım geliştiricisi, ortaya çıkan yazılım tasarımına özen göstermelidir.

Bu görevi yerine getirmek için gerekli araçlar, tasarım ilkeleridir, çünkü bunlar yazılım geliştiricinin nesne yönelimli programlama paradigmasından tam olarak yararlanacak bir yazılım sistemi geliştirmesine destek olur. Bu çalışmada, bağlanma veya uyum gibi genel tasarım ilkelerine odaklanmayıp, tek sorumluluk ilkesi gibi daha spesifik tasarım ilkelerine yoğunlaşıyoruz.

* İlgili yazar.
E-posta adresi: reinhold.ploesch@jku.at (R. Plösch).

https://doi.org/10.1016/j.jss.2018.03.002
0164-1212/

prensibi veya açık-kapalı prensibi. Genel olarak, bu tür tasarım prensipleri, tasarım kalitesi unsurlarını (örneğin, sürdürülebilirlik, taşınabilirlik veya işlevsel uygunluk) sağlamak ve tuzakların ve engellerin ortaya çıkmasını önlemek için kılavuzlar tanımlar (Dooley, 2011). Örneğin Martin (2003), yazılım tasarım yönlerini öğretmek ve yazılım sistemlerinin kalitesini değerlendirmek için de kullanılan bir dizi nesne yönelimli tasarım ilkesini geniş bir şekilde tartışmaktadır (Samarthyam ve ark., 2013). Bu tasarım ilkeleri kümesinin bir örneği, her soyutlamanın değiştirilme nedenleriyle ifade edilen tek bir sorumluluğa sahip olması gerektiğini belirten tek sorumluluk ilkesidir. Başka bir deyişle, bir sınıfın değiştirilme nedeni iki tane olduğunda ve bu değişiklikler birbiriyle ilişkili değilse, sınıfın çok fazla sorumluluk içerdiği muhtemeldir (Martin, 2003). Sonuç olarak, bu gibi durumlarda her sorumluluk tek bir sınıfa ayrılmalıdır. Bu kılavuza bağlı kalmak, kaynak kodun bakımını ve okunabilirliğini kolaylaştırır ve yan etkilerin olasılığını azaltır. Genel olarak, yazılımın tasarımını geliştirir.

Çeşitli tasarım ilkelerinin pratikteki önemini ortaya koymak için, 104 yazılım mühendisi ve mimarı ile bir anket yaptık (Plösch et al., 2016a). Bu ankette, yukarıda bahsedilen tek sorumluluk ilkesi ile ilgi alanlarının ayrılması ilkesi ve bilgi gizleme ilkesinin, anket katılımcıları tarafından bilinen en önemli üç ilke olduğunu tespit ettik. Al-

Mühendisler ve mimarlar bu ilkelerin farkında olsalar da, bir yazılım sistemi geliştirirken veya tasarlarken bu ilkeleri uygulamakta zorluklar yaşadıklarını belirttiler ve biz de bunu gözlemledik. Bunun nedenlerinden biri, tasarım ilkelerinin açıklamalarının hala çok belirsiz olması ve tasarımda kolayca anlaşılmaması ve doğru bir şekilde uygulanamamasıdır. Bu eksiklikten yola çıkarak, tasarım ilkelerini analiz ederek ilgili tasarım en iyi uygulamalarını (Riel (1996) göre tasarım sezgisel yöntemleri) sistematik olarak belirledik. Bu uygulamaların kaynak kodunda doğrudan ihlallerini belirlemek için, Plösch et al. (2016b) tarafından tanıtılan bir ölçüm aracı geliştirdik. Son olarak, bu uygulamaları ilkelere resmi bir şekilde atayan Quamoco tabanlı bir kalite modeli tanımladık (Plösch et al., 2016a).

Bu tasarım kalitesi modelinin geçerliliğini doğrulamak için, aşağıdan yukarıya bir strateji izledik ve tasarımda en iyi uygulamaları incelemeye başladık. Bu nedenle, 214 katılımcıyla geniş çaplı bir anket yaptık ve genel olarak tasarım kalitesi üzerinde farklı tasarım en iyi uygulamalarının önemi hakkında bulgular elde ettik (Bräuer et al., 2017b). Bu ankette tasarım ilkeleri göz ardı edildi.

Gerçekleştirilen anketlere dayanarak, bu tasarım bilgisi taşıyan kavramların (tasarım en iyi uygulamaları ve tasarım ilkeleri) önemi hakkında kanıtlar elde ettik ve bulgular beklentilerimizi karşıladı. Ancak, önemli ve geriye kalan soru, belirli bir tasarım ilkesi için önerilen tasarım en iyi uygulamalarının, ilkenin temel yönlerini kapsayıp kapsamadığı veya sadece bazı küçük tasarım yönlerine değinip değinmediğidir. Bu genel soruyu yanıtlamak için aşağıdaki araştırma sorularını türettik:

- RQ1: Tasarım ilkeleriyle ilgili en iyi tasarım uygulamaları ne kadar önemlidir?
- RQ2: Bir tasarım ilkesini işlevsel hale getirmek için ek tasarım en iyi uygulamaları var mı?
- RQ3: Bir tasarım ilkesinin tasarım bilgisi, ilgili tasarım en iyi uygulamalarıyla ne ölçüde kavranabilir?

RQ1, atanan ilkelerle ilgili en iyi uygulamaların önemine odaklanmaktadır. Bu, diğerlerine kıyasla tasarım ilkesine daha güçlü etki eden en iyi uygulamaları ortaya çıkarmalıdır. Bu ilişkiyi tartışırken, katılımcıların önerdiğimiz uygulama setinde ele alınmayan yönleri belirledikleri varsayılabilir. Bu nedenle RQ2 bu noktayı açıkça ele almakta ve bizim tarafımızdan dikkate alınmayan fikirleri toplamaktadır. Son olarak, önerilen tasarım en iyi uygulamalarının tasarım ilkelerini ne ölçüde kapsadığını anlamak gerekir. Her ilke için eksiksizlik değerlendirmesi açısından RQ3'ü yanıtlayarak, tasarım en iyi uygulamaları setimizin tasarım ilkelerini ölçebildiğini ve iyileştirme eylemleri türetmek için temel sağlayabildiğini iddia edip edemeyeceğine dair kanıt elde edeceğiz.

Üç araştırma sorusunun yanıtlarını ortaya çıkarmak için, yazılım tasarımı alanında 31 uzmanla odak grup araştırması gerçekleştirdik. Odak grup araştırması yaklaşımı, derinlemesine tartışmalar için bir çerçeve, yeni fikirlerin toplanması için bir ortam ve bir araştırma sorusu üzerinde işbirliği içinde çalışmak için teknikler sağladığı için seçildi. Sonuncu husus, odak grubu katılımcılarının birbirlerinden öğrenerek, özellikle tasarım ilkeleri konusunda konuya daha ayrıntılı bir bakış açısı kazanabilecekleri anlamına gelir. Bu, ilkelerin ortak bir şekilde anlaşılması sayesinde çalışmanın genel geçerliliğini artırmaya yardımcı olur – bu durum, örneğin bireysel olarak yapılan görüşmeler veya anketler kullanılarak elde edilmesi zor bir koşuldur.

Bu makalenin geri kalanı aşağıdaki şekilde yapılandırılmıştır. Bir sonraki bölümde tasarım ilkeleri ve araştırma yaklaşımı hakkında daha ayrıntılı bilgi verilmektedir. Bölüm 3 , odak grubunun tartışma sürecini ve sonunda birincil sonuçların özetini içeren bu araştırmanın tasarımını açıklamaktadır. Bölüm 4, araştırma sürecinin sonucunu çıkarır ve sonuçları tartışır. Son olarak, Bölüm 5, bu araştırmanın sınırlamalarını vurgulamaktadır. Son bölümde, sonuç ve gelecekteki çalışmalar için öneriler sunulmaktadır.

2. İlgili çalışmalar

Bu çalışma, tasarım en iyi uygulamalarını kullanarak tasarım ilkelerine uyumu doğrulayarak nesne yönelimli yazılım tasarım kalitesini ölçme araştırma alanını incelemektedir. Bu konuyu daha ayrıntılı incelemek için odak grup araştırma yöntemi seçilmiştir. Bu nedenle, araştırma alanı ve araştırma yöntemiyle ilgili çalışmalar tartışılmaktadır.

2.1. Tasarım ilkeleri

Nesne yönelimli yazılım tasarımını ölçmeye yönelik yaklaşımlar, Chidamber ve Kemerer'in nesne yönelimli tasarımın özelliklerini ölçen tanınmış bir metrikler dizisi önermesinden bu yana büyük ilgi görmüştür (Chidamber ve Kemerer, 1994). Bazı yaklaşımlar, belirli nesne yönelimli tasarım özelliklerinin karakteristiklerini anlamak için bu seti uyarladı ve genişletti (Subramanyam ve Krishnan, 2003; Srivastava ve Kumar, 2013) veya kohezyon, bağlanma ve miras ile ilgili daha genel tasarım ilkelerinin belirli yönlerini araştırdı (Briand ve ark., 1999, 2001). Ölçümlemenin yanı sıra, yeniden yapılandırma kararları belirlenebilir ve tasarım iyileştirmeleri gerçekleştirilebilir. Ancak, tek bir ölçüt kullanmanın ve bunları ayrı ayrı değerlendirmenin, sağlam tasarım iyileştirmeleri önermek için yeterli bilgi sağlamadığı kabul edilmiştir (Marinescu, 2004).

Çalışmalarımız, yazılım tasarımında tasarım ilkelerinin uygulanmasını anlamaya odaklanmaktadır. Bu bağlamda, tasarım ilkelerini Coad ve Yourdon (1991) tarafından önerilen veya Meyer (1997) tarafından açıklanan düşük bağlanma, yüksek uyum, orta düzeyde karmaşıklık ve uygun kapsülleme gibi kaba düzeyde ele almıyoruz, daha ince bir düzeyde ve tek sorumluluk ilkesi veya açık-kapalı ilkesi gibi belirli bir tasarım sorununu ele alan ilkelerle ele alıyoruz.

Bu sözde *ince taneli tasarım ilkelerinin* daha doğru olduğu ve yüksek kaliteli yazılım tasarımı oluşturmak için yararlı bir rehberlik sağladığı tartışılmıştır (Dooley, 2011; Sharma et al., 2015). Ayrıca, (nesne yönelimli) yazılım tasarımının yapısal bileşenlerini düzenlemek ve organize etmek, tasarım bilgisi konusunda ortak bir fikir birliği oluşturmak ve yeni başlayanların tuzaklardan ve engellerden kaçınmasına yardımcı olmak için kullanılırlar. İnce taneli tasarım ilkeleri tasarım unsurlarını ayrıntılı olarak ele alsa da, doğrudan ölçülebilecek kadar somut değildirler.

Tasarım ilkelerinin işlevselleştirilmesine değinen bir çalışma dışında, bu konuda daha geniş kapsamlı bir araştırma yapıldığını bilmiyoruz. Bu araştırma konusunu bir ölçüde ele alan çalışma, iyi arayüz tasarımıyla ilgili tasarım ilkelerine odaklanmaktadır (Abdeen ve ark., 2013). Daha ayrıntılı olarak, ekip arayüzlerin tasarımını analiz ederek arayüzlerin arayüz ayrıştırma ilkesine ve programa uyumunu değerlendirir, uygulama ilkesine değil. Bu nedenle, ilkelere uyum ölçütleri atamakta ve bu ölçütleri bir dizi açık kaynaklı projede ölçmektedir. Bu araştırmanın sonuçları, yazılım geliştiricilerin bu dar ilkeler kümesini takip ettiklerini göstermektedir (Abdeen et al., 2013), ancak çalışma, ölçülen ilkelere uymayan bir tasarımın nasıl iyileştirilebileceğine dair herhangi bir rehberlik sağlamamaktadır.

Tasarım ilkelerinin ölçülmesi ve değerlendirilmesine ilişkin diğer çalışmalar Samarthyam ve ark. (2013) tarafından sunulmuştur. Bu araştırmacılar, tasarım uzmanlarının deneyimlerine dayanarak gerçek dünyadaki projelerin tasarımını manuel olarak değerlendirmektedir. Bu değerlendirmeler sırasında, tasarım kalitesinin düşük olmasının nedeninin tasarım ilkelerinin ihlali olduğunu gözlemlemektedirler. Ancak, Samarthyam ve diğerleri (2013) ile Sharma ve diğerleri (2015) tarafından vurgulandığı gibi, tasarım sorunlarını tasarım ilkelerinin ihlalleriyle ilişkilendiren bir analiz çerçevesi toplulukta eksiktir.

Önceki çalışmalarımızda bu boşluğu ele almış ve tasarım ilkelerine en iyi tasarım uygulamalarını atayan bir model önermiştik (Plösch et al., 2016a). En iyi tasarım uygulamalarını, ölçülebilir ve karşılaştırılabilir bir şekilde tanımlayan bir yaklaşım olarak görüyoruz.

statik kod analiz aracı ile işlevselleştirilebilen ölçülebilir tasarım özellikleri (Plösch et al., 2016b). Soyut tasarım ilkelerini nicel yazılım özelliklerine bağlamak için tasarım en iyi uygulamaları, diğer adıyla tasarım sezgileri (Riel, 1996) kullanmanın değeri Churcher et al. (2007) tarafından vurgulanmaktadır. Bu grup, tasarım sezgilerinin ihlallerini görselleştirmeye odaklanmakla birlikte, bu ölçüm tekniğinin tasarım sorunlarını belirlemek ve tasarım kalitesini değerlendirmek için değerli bir araç olduğunu savunmaktadır (Churcher et al., 2007).

Özetlemek gerekirse, tasarım kokuları metrikleri aracılığıyla soyutlama, bağlanma, uyum veya miras gibi daha genel tasarım ilkelerini ölçmek için yaklaşımlar öneren çok sayıda yaklaşım ve çalışma bulunmaktadır. Ancak, tek sorumluluk ilkesi veya açık kapalı ilkesi gibi daha spesifik tasarım ilkelerini işlevsel hale getirmek için çok az çalışma bulunmaktadır. Mevcut yaklaşımlar, ölçüm için ya metrik tabanlı ya da koku tabanlı yaklaşımlara odaklanırken, bizim yaklaşımımız kaynak kodun tasarımını nasıl iyileştirebileceğine dair spesifik ipuçları sağlayan tasarım en iyi uygulamalarına dayanmaktadır. Önceki çalışmamızda (Bräuer et al., 2017b), tasarım ilkelerini ölçme yönünü dikkate almadan, genel olarak tasarım en iyi uygulamalarımızın önemini göstermiştik. Bu çalışmanın odak noktası, tasarım ilkeleri bağlamında tasarım en iyi uygulamalarımızın önemini ortaya çıkarmak ve tasarım ilkelerinin tasarım en iyi uygulamalarımızla ne ölçüde ölçülebileceğine dair içgörüler elde etmektir.

2.2. Odak grup araştırması

Yukarıda belirtilen araştırma sorularını ele almak için odak grup araştırma yaklaşımını seçtik. Odak grupları, belirli bir araştırma alanında katılımcıların kişisel algılarını elde etmek için esnekliğe sahip, özenle planlanmış tartışmalardır (Kontio ve ark., 2008). Araştırma sorusu belirli bir araştırma alanına odaklandığından ve odak grubu üyeleri görüşlerini ifade edebildiklerinden, bu araştırma yaklaşımını uyguluyoruz. Yazılım tasarım kalitesinin tartışılması karmaşık bir konu olduğundan ve bağlama bağlı olduğundan, katılımcıların bireysel görüşlerini yakalayabilmek için esneklik gereklidir.

Odak grup tartışmaları kullanılarak tasarım ilkeleriyle ilgili bir araştırma bulunmamakla birlikte, odak grup araştırması yazılım mühendisliği araştırma topluluğu içinde kabul gören bir ampirik araştırma yaklaşımıdır (Kontio ve ark., 2008). Bu yaklaşımın güçlü yanlarından biri, ortamın etkileşimli yapısı ve katılımcıların farklı geçmişleri sayesinde yeni içgörüler elde edilebilmesidir (Kontio ve ark., 2008). Bu, araştırmacıların önceden düşünülmemiş olabilecek yeni fikirler elde etmelerine yardımcı olur. Ayrıca, odak grupları, modellerin veya kavramların nasıl sunulduğuna ilişkin geri bildirim almak için çok uygundur (Edmunds, 2000). Örneğin, bu yaklaşım, gömülü sistem mühendisliği için bir süreç çerçevesini doğrulamak amacıyla başarıyla kullanılmıştır (Charalampidou et al., 2014).

3. Odak grup araştırma tasarımı

Giriş bölümünde bahsedilen araştırma sorularını yanıtlamak için, katılımcıların her bir tasarım ilkesinin kavramını ve özelliklerini anlamaları gerekir. Ayrıca, açıklığa kavuşturulması gereken bazı hususlar hakkında sorularları olabilir; aksi takdirde tasarım ilkeleri yanlış yorumlanabilir. Tasarım ilkesini sağlam bir şekilde anladıklarında, en iyi tasarım uygulamalarının bir tasarım ilkesiyle ilişkili olup olmadığını değerlendirmek mümkün olur. Bu koşullar doğrultusunda, odak grup araştırması yapmaya karar verdik.

Odak grubunun dinamikleri ve nispeten küçük örneklem büyüklüğü araştırma sonuçlarını etkileyebilir (Kontio ve ark., 2008), ancak araştırma yöntemiyle ilgili bazı kararlarımız bu riskleri açıkça azaltmaya yöneliktir. Araştırma yönteminin tüm tasarımı ve veri toplama yöntemi olarak Delphi ile birleştirilmesi, Kontio ve ark. (2008) tarafından verilen kılavuzla uyumludur.

3.1. Araştırma planlaması

Odak grup yöntemi, yeni kavramlar veya modeller hakkında geri bildirim toplamak ve fikir üretmek için uygundur (Kontio ve ark., 2008). Yazılım mühendisliğinde tanımlanmış bir dizi tasarım ilkesini tartışmak için bu yaklaşımı kullanmaya karar verdik. Bu ilkelerin seçimi, pratik önemi temel alınarak yapılmıştır. Bu nedenle, önceden bir anket gerçekleştirerek, belirlenen 31 nesne yönelimli tasarım ilkesinden en önemli on ilkeyi belirledik (Plösch ve ark., 2016a). Sonuç olarak, bu çalışmadaki araştırma konuları *tek sorumluluk ilkesi (SRP), bilgi gizleme (IHI), kendini tekrar etme ilkesi (DRY), açık kapalı ilkesi (OCP), asiklik bağımlılık ilkesi (ADP), arayüz ayrımı (ISP), miras yerine bileşim ilkesini tercih etme (FCOI), komut sorgu ayrımı ilkesi (CQS), ortak kapatma ilkesi (CCP)* ve *uygulama değil arayüze göre programlama (PINI)dır.* Aşağıda, odak grup araştırması boyunca kullanılan her ilkenin tanımı verilmiştir.

- **Tek sorumluluk ilkesi:** Bir sınıfın değiştirilmesi için iki neden varsa ve bu değişiklikler birbiriyle ilişkili değilse, sınıfın işlevselliği iki ayrı sınıfa bölünmelidir (Martin, 2003). Bu ilkeye uyulduğunda, her sınıf yalnızca bir sorumluluğu üstlenir ve uzantılar her sınıfta ayrı ayrı ele alınır.
- **Bilgi gizleme ilkesi:** Bir sınıf, uygulamasını karakterize eden tasarım kararlarını ifşa edemez.
- **Kendinizi tekrarlamayın:** "Her bilgi parçası, bir yazılım sistemi içinde tek, açık ve yetkili bir temsile sahip olmalıdır" (Hunt ve Thomas, 1999, s. 27), yani ne yinelenen veri yapıları veya kaynak kodu ne de anlamsız kaynak kodu belgeleri olmalıdır.
- **Açık kapalı ilke:** Kaynak kodu, mevcut kaynak kodunda minimum değişiklikle yeni işlevler eklenebilecek şekilde yazılmalı ve tasarım bu şekilde uygulanmalıdır.
- **Acyclic bağımlılık ilkesi:** Paketler arasındaki bağımlılık yapısı, yönlendirilmiş bir döngüsüz grafik olmalıdır; bu, paket düzeyinde bağımlılık yapısında döngüler olmaması gerektiği anlamına gelir (Martin, 2003).
- **Arayüz ayrıştırma ilkesi:** "Sınıflar, kullanmadıkları [bir arayüzün] yöntemlerine bağımlı hale getirilmemelidir" (Martin, 2003, s. 137). Tek bir büyük arayüz yerine, her bir arayüzün belirli bir sınıf grubuna hizmet ettiği bir dizi küçük arayüz tercih edilir.
- **Kalıtım yerine bileşim tercih edin:** İşlevselliği yeniden kullanmak için sınıf kalıtımı yerine nesne bileşimi kullanılmalıdır.
- **Komut sorgu ayrımı:** Bir yöntem ya bir nesneyi değiştirmeli (komut) ya da bir nesnenin verilerini döndürmelidir (sorgu). Bu iki kavram birbirine karıştırılmamalıdır.
- **Ortak kapatma ilkesi:** Bir paket içindeki sınıflar, aynı tür değişikliklere karşı birlikte kapatılmalıdır. Bu nedenle, bir paketi etkileyen bir değişiklik, o paketteki tüm sınıfları etkiler (Martin, 1996).
- **Uygulama değil, arayüze göre programlayın:** Somut sınıflar yerine arayüzler veya soyut sınıflar kullanılmalıdır.

Bu tasarım ilkelerinin her birine, Ek A'daki uygulamalar listesinden en az bir tasarım en iyi uygulaması (diğer adıyla tasarım kuralı) atadık. Çoğu durumda, ele alınan tasarım yönlerinin zenginliğine bağlı olarak, bir tasarım ilkesine birden fazla tasarım en iyi uygulaması bağlanmıştır. Ayrıca, bazı ilkeler benzer tasarım amaçlarını paylaşır ve bu da uygulamaların birden fazla atanmasına neden olur. Bu araştırmayı yürütmeden önce, iyi nesne yönelimli tasarım için önemlerini anlamak amacıyla, anket temelli bir yaklaşım kullanarak (Bräuer et al., 2017b) her bir tasarım en iyi uygulamasını sistematik olarak değerlendirdik. Sonuç olarak, önemlerinin farkına vardık.

tasarım en iyi uygulamalarının önem düzeyleri; bu, bizim beklediğimiz bir şeydi. Bu anketi kullanarak nesne yönelimli tasarım için tasarım en iyi uygulamalarının önemini iyi bir şekilde anlamış olsak da, tasarım en iyi uygulamaları ile tasarım ilkeleri arasındaki ilişki hala yeterince anlaşılmamıştır - bu çalışma, bu boşluğu doldurmaya çalışmaktadır.

Açıklığa kavuşturmak gerekirse, bu tasarım en iyi uygulamalarının geçerliliği sadece isimlerine ve açıklamalarına dayanmamaktadır, çünkü bu uygulamaların uygunsuzluklarını belirleyen MUSE adlı bir ölçüm aracı da geliştirdik (Plösch et al., 2016b). Böylece, araç tasarım en iyi uygulamalarının mantığını uygular

uygulamalarının mantığını uygular ve Java, C# veya C++ ile yazılmış kaynak kodundaki ihlalleri nicel olarak ölçer. Önceki çalışmalarda, MUSE farklı araştırma faaliyetleri ve endüstriyel ortakların gerçek dünya projelerinde uygulanmıştır
(Plösch et al., 2016a,b).

Sözlü olarak yürütülen odak grup tartışmaları, fikir liderleri veya grup davranışları tarafından domine edilebilir (Kontio ve ark., 2008). Bu nedenle, ilk şart olarak, ekip üyelerinin anonim olarak katkı sağlamalarının istendiği anonim tartışmaların yürütülmesini belirledik. Ayrıca, tasarım ilkeleri karmaşıktır ve kısa bir oturumda kavranması zordur. Bu nedenle, ikinci gereklilik olarak, katılımcıların tasarım ilkelerini bağımsız olarak inceleyebilecekleri bir yol olması gerektiğini belirledik. Son olarak, seyahat masrafları geri ödenemediği için katılımcıların yüz yüze görüşemeyecekleri bir kısıtlama vardı. Bu nedenle, üçüncü gereklilik tartışmaların uzaktan yürütülmesine odaklandı. Tüm bu ihtiyaçları karşılamak için, Turney ve Pocknee (2005) tarafından tanımlandığı ve önerildiği gibi sanal (çevrimiçi) odak

3.2. Odak grubu tasarımı

Odak gruplarını tasarlarken ve gerekli çabayı göz önünde bulundururken, her grupta on tasarım ilkesinin tamamını tartışmak mümkün olmadı. Sonuç olarak, tasarım ilkeleri önceden tanımlanmış gereksinimlere göre iki gruba ayrıldı. İlk olarak, atanan tasarım en iyi uygulamalarının sayısını ve dolayısıyla bunları değerlendirme çabasını odak grupları arasında dengeli tutmaya çalıştık. Örneğin, atanan uygulamaları çok olan IHI ve OCP ayrı tutuldu. İkinci olarak, her grup ADP veya CCP olmak üzere iki paketle ilgili tasarım ilkesinden birini içermeliydi. Üçüncü olarak, FCOI ve PINI'yi ayırdık, çünkü bu iki tasarım ilkesi soyutlama sorunlarını ele alıyor. Son olarak, SRP, IHI, FCOI, CQS ve CCP'yi içeren ilk grubu ve OCP, DRY, ADP, ISP ve PINI'yi içeren ikinci grubu oluşturduk.

Tipik bir odak grup araştırma yaklaşımı, dört ila sekiz katılımcıdan oluşan dört ila altı odak grubundan oluşur (Kontio ve ark., 2008). Daha küçük gruplar katılımcıların daha yüksek düzeyde katılımını sağladığından (Kontio ve ark., 2008), beş katılımcıdan oluşan gruplar oluşturmaya karar verdik. Ayrıca, en az üç grubun bir dizi tasarım ilkesini tartışmasını ve her tasarım ilkesi için 15 görüş elde etmeyi hedefledik. Sonuç olarak, altı odak grubu için en az 30 katılımcıya ihtiyaç vardı.

3.2.1. Katılımcıların seçimi

Katılımcıları seçmeden önce, gerekli minimum yazılım mühendisliği becerilerini belirledik. Bu nedenle, katılımcılar üç nesne yönelimli programlama dilinden herhangi birinde iyi veya üst düzey deneyime sahip olmalıdır

dilinden birinde iyi veya en üst düzeyde deneyime sahip olmalıdır. Bu gereklilik,
katılımcıların tasarım ilkesinin amacını kavrayamama riskini azaltır
. Bu deneyimin değerlendirilmesi için, katılımcılar beş seviyeli bir ölçekte kendi kendilerini değerlendirmek zorundaydılar ve bu ölçekte en yüksek iki değer iyi ve en iyi idi.

Çalışmamıza katılımcıları üç şekilde elde ettik. İlk olarak, önceki anketimizi dolduran kişileri davet ettik.

tasarım en iyi uygulamalarının önemi. Bu ankette, katılımcılar anketin özetini almak için abone oldular ve bu belgeyi dağıtarak onları bir odak grubuna katılmaya davet ettik. Bu edinim, yeniden katılan üyelerin bilgi düzeyindeki ilerlemenin geçerliliğine bir tehdit oluşturmasına rağmen, tüm odak grubu üyelerine tasarım en iyi uygulamaları hakkında aynı (hatta daha fazla) bilgi sağlayarak bu tehdidi kontrol altına aldık. İkinci olarak, yazılım geliştirmeye yoğun olarak odaklanan yerel şirketlerden, yazılım geliştiricilerinin nesne yönelimli tasarım becerilerini geliştirmek için katılımı teşvik etmelerini istedik. Son olarak, bir endüstriyel araştırma ortağı, odak grup tartışmalarını kıdemli geliştirici eğitiminin bir parçası olarak dahil etti. Bu eğitimde tasarım ilkeleri merkezi bir rol oynadığından, bu katılımcılar odak grup araştırmamız için mükemmel adaylardır.

3.2.2. Katılımcıların segmentasyonu

Odak gruplarının segmentasyonu, grupların uygun bileşimini ele alır ve Morgan'a (1997) göre iki temel avantaj sunar: (1) segmentasyon, tüm araştırma projesine karşılaştırılabilir boyutlar oluşturulmasına olanak tanır ve (2) grup üyelerini birbirine daha benzer hale getirerek tartışmaları kolaylaştırır. Bu iki avantajı göz önünde bulundurarak, farklı kısıtlamalara bağlı olarak altı odak grubu oluşturduk. Örneğin, bazı katılımcıların katılım süreleri sınırlıydı; bu nedenle, onları daha erken veya daha geç bir grup tartışmasına atadık. Ayrıca, ekiplerin homojenliğini desteklemek için aynı şirketten işe alınan katılımcıları bir araya getirdik. Kısaca özetlemek gerekirse,
Grupların temel özellikleri aşağıda gösterilmiştir:

- **FG-I:** Güçlü bir akademik geçmişe sahip, yazılım mühendisliği araştırma departmanlarında çalışan bir grup üye. Bu katılımcılar önceki anketimizi tamamlamışlardır. Bu nedenle, tüm katılımcıların yazılım mühendisliği niteliklerine ilişkin gereksinimlerimizi karşıladıklarından emin olabiliriz.
- **FG-II:** Bu grup, LinkedIn ve ResearchGate tartışma grupları aracılığıyla işe alınan gönüllülerden oluşmaktadır. Geçmiş kontrolleri ve önceki anketimize yaptıkları katkılar, yazılım mühendisliği yeterliliklerini kanıtlamıştır. Ancak, grup üyeleri farklı uygulama alanlarında çalışmaktadır.
- **FG-III ila V:** Endüstriyel araştırma ortağımız üç odak grubu oluşturdu. Daha spesifik olarak, bu katılımcılar araştırma ortağı tarafından sunulan şirket içi kilit geliştirici eğitimine kayıtlı kıdemli yazılım geliştiricileridir. Diğer katılımcılar için yapıldığı gibi mühendislik becerilerinin arka plan kontrolünü yapamadık, ancak araştırma ortağımız becerilerin gerekli seviyede olduğunu ve çoğu durumda bu seviyenin üzerinde olduğunu doğruladı. Ayrıca, eğitim programı tasarım ilkelerini öğretmeye ve bu ilkelere uyulması konusunda farkındalık yaratmaya odaklanmaktadır. Bu nedenle, eğitim alanlar kaynak kodlarında ilke avcılığı gibi ödevleri ve ilkeye dayalı iyileştirme faaliyetlerini tamamlamalıdır. Genel olarak, bu eğitim alanlar tasarım ilkelerini ve bu ilkelerin ihlallerini derinlemesine anladıkları ve kıdem seviyeleri nedeniyle mükemmel yazılım mühendisliği bilgisine sahip oldukları için bu araştırma için mükemmel adaylardır.
- **FG-VI:** Yerel bir endüstri ortağı bir odak grubu kurdu. Bu grup üyeleri endüstriyel otomasyon alanında çalışmakta ve nesne yönelimli teknolojiler kullanarak gömülü sistemler geliştirme konusunda beş yıldan fazla deneyime sahip kıdemli geliştiricilerdir.

3.3. Odak grup tartışması

Çevrimiçi odak grup tartışmaları düzenlemeye karar verdiğimizden, uygun araç ve veri toplama yöntemlerine ihtiyaç duyuldu. Araç açısından, üniversite öğrenim yönetim sistemlerine güvenilmesi önerilir (Turney ve Pocknee, 2005). Bu, şu sonuçları doğurur

araştırma sürecinin üniversite tarafından tanımlanan kalite, güvenlik ve gizlilik düzenlemelerinden yararlanması avantajına sahiptir (Turney ve Pocknee, 2005). Ayrıca, katılımcıların anonimliği ve gizliliği de sağlanabilir (Turney ve Pocknee, 2005). Bu araştırma için, JKU tarafından barındırılan Moodle[1] hizmeti kullanılmıştır. Bu hizmeti dersleri ve öğrenci ekiplerini yönetmek için zaten kullanıyorduk, bu nedenle herhangi bir eğitim gerekmedi.

Grup tartışmasından verileri toplamak için Delphi yöntemini seçtik. Literatürde, Delphi yöntemi bu amaçla önerilmektedir çünkü tartışma sürecini yapılandırır ve belgelenmesini destekler (Adler ve Ziglio, 1996). Linstone ve Turoff (1975), Delphi yönteminin daha genel bir tanımını sunmakta ve bu yöntemin, katılımcıların karmaşık b i r sorunu ele almalarını etkili bir şekilde sağlamak için grup iletişim sürecini yapılandırdığını belirtmektedir. Bu nedenle, sürecin uygulanması, bireysel katkılar hakkında geri bildirim, grup yargısı veya görüşünün değerlendirilmesi, bireylerin görüş ve yargılarını revize etme fırsatı ve bir dereceye kadar anonimlik sağlamalıdır (Linstone ve Turoff, 1975).

Odak grup tartışmaları sırasında geleneksel bir anket veya görüşmeler yapabilirdik, ancak Delphi yöntemini uygulamaya karar verdik. Bu yöntemin temel avantajı, katılımcıların ilk turda görüşlerini ifade edebilmeleri ve odak grubun diğer katılımcılarının görüşlerini dikkate aldıktan sonra bu görüşleri revize edebilmeleridir. Genel olarak, bu yöntem katılımcıları sorgulamak için daha güçlü bir yaklaşımdır ve katılımcıların, yazılım geliştirme görevleri ve çalıştıkları ortamın etkisiyle şekillenmiş olabilecek ilk görüşlerini yeniden düşünmelerine yardımcı olur. Bu yöntemin bir diğer önemli avantajı, uzmanlar arasında çatışmayı önlemesi (Okoli ve Pawlowski, 2004) ve anonim bir tartışma yürütme konusundaki ilk şartımızı desteklemesidir. Son olarak, Delphi yöntemi katılımcıların yüz yüze görüşmesini gerektirmez (Okoli ve Pawlowski, 2004). Bu, üçüncü gerekliliğimizle mükemmel bir şekilde uyumludur ve araştırmayı yerel uzmanlarla sınırlamaz.

Tartışma sürecine başlamadan önce, iki tasarım ilkesi setinden birini her bir odak grubuna atadık. Daha spesifik olarak, FG-I, FG-III ve FG-V odak grupları SRP, IHI, FCOI, CQS ve CCP tasarım ilkelerini analiz ederken, FG-II, FG-IV ve FG-VI odak grupları DRY, OCP, ADP, ISP ve PINI tasarım ilkelerini analiz etti. Buna göre, her tasarım ilkesi üzerinde en az 15 uzman çalıştı.

Okoli ve Pawlowski (2004) tarafından önerilen sürece dayanarak, odak gruplarımızın tartışmasını üç aşamaya ayırdık. Bu üç aşamanın genel bir özeti, ana faaliyetleri ve çıktıları vurgulayan Şekil 1'de gösterilmektedir. İlk aşama *beyin fırtınası aşaması olarak* adlandırıldı ve katılımcılara konu ve tasarım ilkeleri tanıtıldı. Bu adımın ardından, *açıklığa kavuşturma aşaması* geldi. Bu aşama, beyaz noktaların belirlenmesi ve belirli bir tasarım ilkesi bağlamında tasarım en iyi uygulamalarının öneminin açıklığa kavuşturulmasına odaklandı. Devam eden grup tartışmasının sonucu olarak, son aşama *olan eksiksizlik değerlendirme aşamasında,* katılımcılardan bir tasarım ilkesinin, kendisine atanan tasarım en iyi uygulamalarıyla ne kadar kapsandığını tahmin etmeleri istendi. Aşağıdaki alt bölümlerde, bu üç aşamayı daha ayrıntılı olarak ele alıyoruz.

3.3.1. Beyin fırtınası aşaması

Beyin fırtınası aşamasında, her bir tasarım ilkesinin açıklaması sunuldu. Bu açıklamalar aynı şekilde yapılandırılmıştı ve bir tanım, ele alınan tasarım sorunu, teknik ayrıntılar, etkilenen kalite unsurları ve ilkenin orijinal tanımını içeriyordu. Bu bilgiler, Moodle sistemimizi kullanarak sözde dersler aracılığıyla sunuldu. Ayrıca, aşağıdaki bağlantıdan indirilebilirdi her ilke için aynı bilgileri içeren bir tasarım ilkesi sayfası indirilebilirdi.

Bu aşamanın amacı, her grupta beş tasarım ilkesinin anlaşılmasını sağlamaktı. Bu nedenle, katılımcılar sağlanan materyali incelemek ve açık grup forumunda fikirlerini, sorularını veya yorumlarını paylaşmakla görevlendirildiler. Böylece aralarında etkileşim kurulması ve sağlanan bilgiler üzerinde eleştirel bir şekilde düşünmeleri amaçlandı. Bu aşamanın beyin fırtınası aşaması olarak adlandırılmasının nedeni de budur. Beyin fırtınası, katılımcıların spontan olarak ortaya koydukları fikirlerin bir listesini oluşturmak için kullanılan b i r grup yaratıcılık tekniğidir.

3.3.2. Açıklama aşaması

Tasarım ilkelerinin sağlam bir şekilde anlaşılmasına dayanan açıklığa kavuşturma aşaması, bir dizi tasarım en iyi uygulaması getirmiştir. Bu tasarım en iyi uygulamaları, tasarım ilkelerini ölçmek için kullanıldıkları için önemli bir rol oynamaktadır (Plösch et al., 2016a). Ancak, çeşitli tasarım en iyi uygulamalarının önemi ve ilkeler üzerindeki olumlu veya olumsuz etkileri konusunda net bir anlayış bulunmamaktadır. Sonuç olarak, bu aşama tasarım ilkeleri ile ilgili tasarım en iyi uygulamaları arasındaki ilişkiyi açıklığa kavuşturmaktadır.

Bu belirsizliği gidermek için Delphi yöntemi iki turda uygulandı. İlk turda, atanan tasarım en iyi uygulamalarını içeren beş tasarım ilkesini içeren bir anket dağıtıldı. Bu ankette katılımcılardan, ilgili tasarım ilkesiyle bağlantılı olarak tasarım en iyi uygulamasının önemini değerlendirmeleri istendi. Buna göre, tasarım en iyi uygulamalarını tasarım ilkesinin perspektifinden incelemek önemliydi. Bu husus katılımcılara açıkça iletildi. Değerlendirme, *çok yüksek (5)* ile *çok düşük (1)* arasında beş puanlık bir ölçekte, *alakasız (0)* seçeneği de dahil olmak üzere gerçekleştirildi. Ayrıca, anket katılımcılardan tasarım ilkesini ölçmek için alakalı olduğunu düşündükleri diğer tasarım en iyi uygulamalarını listelemelerini istedi.

İlk turda gelen önerileri topladık ve tekrarları sildik. Ayrıca, önerileri isimlendirme kurallarımıza ve açıklama formatımıza uydurmamız gerekiyordu. Bu aşamada, bunlar sadece öneri niteliğindeydi, ancak önem derecelendirmesi henüz yapılmamıştı. Sonuç olarak, yeni tasarım en iyi uygulamalarının da önerilenlerle aynı değerlendirme sürecinden geçmesi gerekiyordu. Bu ek adım ve ikinci anket ile uzmanlardan (a) yanıtlarını doğru yorumladığımızı doğrulamaları ve (b) önerilerin önemini değerlendirmeleri istendi. Schmidt'e (1997) göre, "bu adım olmadan, geçerli, konsolide bir liste oluşturulduğunu iddia etmek için hiçbir dayanak yoktur."

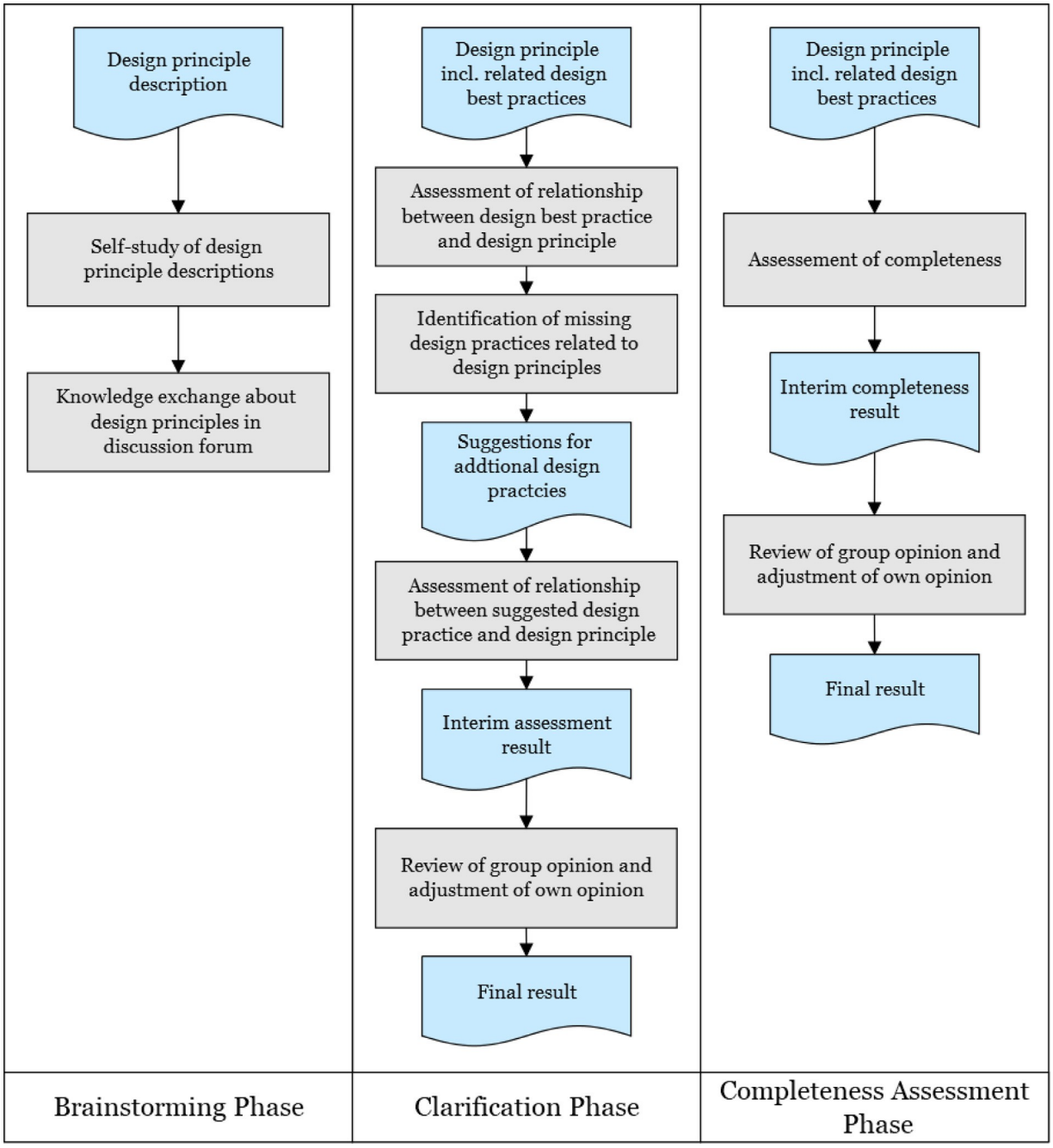
Bu ara adımın ardından, tüm tasarım en iyi uygulamaları katılımcılar tarafından ayrı ayrı değerlendirildi. Bu yanıtlar daha sonra ikinci tur için gerekli olan ara sonuçta birleştirildi. Daha ayrıntılı olarak, Delphi yöntemi, katılımcıların grup görüşüne göre görüşlerini revize edebileceklerini belirtir. Bu nedenle, tüm katılımcılara ilk turdan ve ara sonuçtan elde edilen anketler verildi. Ardından, anketi yeniden gözden geçirmeleri ve algılarını değiştirdiklerinde ilk önem derecelendirmelerini ayarlamaları istendi. Son olarak, revize edilip geri gönderilen anket, netleştirme aşamasının sonucunu temsil ediyordu.

3.3.3. Tamamlanma değerlendirme aşaması

Üçüncü aşama, tasarım ilkelerinin atanan en iyi tasarım uygulamalarıyla tamlığını ortaya çıkarmayı amaçladığından, araştırmanın karmaşıklığını en üst düzeye çıkardı. Daha ayrıntılı olarak, bir anket katılımcılara her tasarım ilkesi için iki tahmin sordu:

- (1) Önerdiğimiz tasarım en iyi uygulamaları seti ile bir tasarım ilkesinin kapsamı (puan olarak) ve

[1] https://moodle.org/ adresinden indirilebilirdi.

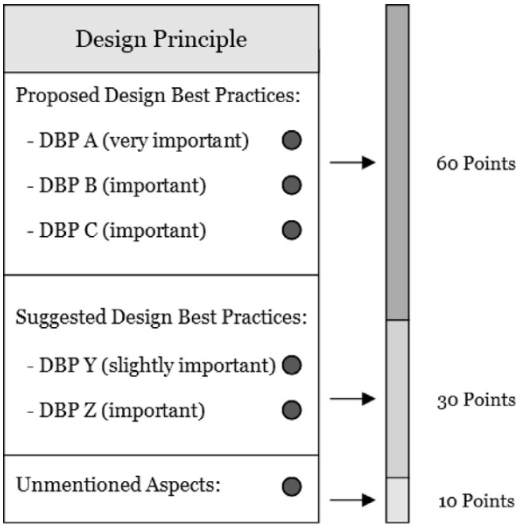


Şekil 1. Odak grup tartışmasının iş akışı.

- (2) Odak grubu tarafından önerilen tasarım en iyi uygulamalarının bir tasarım ilkesini kapsama alanı (puan olarak).

Bu görev için katılımcılar, tasarım en iyi uygulamalarının türetilen önemini bir gösterge olarak dikkate almak zorundaydılar. Bu görevi daha iyi açıklamak için Şekil 2'deki örneği inceleyin. Bu örnekte, bir tasarım ilkesinin bizim tarafımızdan belirlenen üç tasarım en iyi uygulaması ve çalışma katılımcıları tarafından önerilen iki tasarım en iyi uygulaması ile karakterize edildiği varsayılmaktadır. Ardından katılımcılar, ilkenin önerilen küme tarafından ne dereceye kadar kapsandığını ve önerilen küme tarafından ne kadar kapsandığını değerlendirmek zorundaydılar. Bahsedilmeyen yönler hala varsa, bu iki değerin toplamı mutlaka 100 puan olmak zorunda değildir. Şekil 2'deki örneğe göre, katılımcı DBP A , DBP B ve DBP C 'nin ilkeye 60/100 puan katkıda bulunduğunu, DBP Y ve DBP Z 'nin (iki tasarım en iyi uygulaması) ise çalışma katılımcıları tarafından sorulan) 30/100 puan alır ve hala bahsedilmeyen, 10/100 puanla değerlendirilen yönler vardır.

Yukarıdaki örnekte, bahsedilmeyen yönlerin fikri biraz açıklığa kavuşturulmalıdır. Bu araştırmayı tasarlarken ve amacını tartışırken, tasarım ilkelerinin bazen çok yönlü olduğu ve bu nedenle sayılabilir bir dizi tasarım en iyi uygulamasıyla tam olarak kapsanmasının zor olduğu tespit edilmiştir. Örneğin, OCP, sınıflar, modüller ve işlevler gibi yazılım varlıklarının genişletmeye açık, ancak değiştirmeye kapalı olması gerektiğini belirtir (Martin, 2003). Bu ilke çoğu yazılım varlığını ele aldığından, ilkeyi oluşturan tüm özellikleri tanımlamak imkansızdır. Bu durumu ele almak ve bütünlüğün çok katı bir şekilde değerlendirilmesini önlemek için, katılımcıların içgüdülerine dayanan bu değerlendirme tamponunu sağladık.



Şekil 2. Tasarım ilkelerinin, atanan tasarım en iyi uygulamalarıyla kapsamı.

Delphi yaklaşımını izleyerek, bu aşamadaki değerlendirme yine iki turda gerçekleştirildi ve katılımcılar, değerlendirmelerini revize etmek için diğer katılımcıların yanıtlarını aldılar. Son olarak, düzeltilmiş anketler odak grubunun sonucunu temsil etti.

4. Sonuçlar

Beyin fırtınası aşamasında, katılımcılar tasarım ilkeleriyle ilgili sağlanan materyalleri incelediler ve bir forum tartışmasına katkıda bulundular. Bu tartışma, daha fazla analiz edilme amacı gütmeden bir anlayış oluşturmayı amaçlıyordu. Ana veri toplama işlemi ikinci ve üçüncü aşamalarda gerçekleştirildi.

4.1. Ek tasarım en iyi uygulamaları için öneriler

İkinci aşamada tasarım ilkeleri üzerindeki en iyi tasarım uygulamalarının öneminin değerlendirilmesi tanımlanmış bir ölçeğe dayalıyken, ek tasarım yönleri hakkında sorulan açık uçlu soru niteliksel olarak analiz edildi. Böylece, katılımcıların notları iki bağımsız araştırmacı tarafından incelenerek tasarım amaçları çıkarıldı ve diğer uygulamalarla benzerlikler belirlendi. Önerinin amacını yargılamadan, notları bir tasarım kuralı açıklamasına ve anlamlı bir isme dönüştürdük. Bu nedenle, önerilen açıklamayı tasarım en iyi uygulamalarını tanımlama modelimize uyarladık ve diğer açıklamalarda kullanılan terimleri değiştirdik. Ayrıca, yanlış yorumlamaları ortaya çıkarmak için orijinal not, türetilen açıklama ve isim öneren kişi tarafından doğrulandı. Toplamda, 34 ek tasarım en iyi uygulaması önerildi ve bunlardan ikisi kavramsal olarak havuzumuzdaki uygulamalarla örtüşüyordu.

Odak gruplarının ayrı ayrı çalıştığını ve yeni fikirlerin diğer ekiplerle paylaşılmadığını anlamak önemlidir. Sonuç olarak, her tartışma Tablo 1 ve 2'de gösterildiği gibi 34 önerinin sadece b i r alt kümesini içeriyordu. Önerilen her bir tasarım en iyi uygulaması açıklaması Ek B'de listelenmiştir. Ayrıca, Tablo 2, FG-V'deki hiçbir üyenin ek bir tasarım en iyi uygulaması öneri s i s u n m a d ığını göstermektedir. Bununla birlikte, bu iki tablo, eksik tasarım en iyi uygulamalarını belirlemeyi amaçlayan RQ2'nin cevabını özetlemektedir.

Bu yeni tasarım en iyi uygulamaları fikirleri için, bunların statik kod ölçüm aracıyla otomatikleştirilebilmesini şart koşmadık. Örneğin, *UseMeaningfulVariableNames* tasarım en iyi uygulaması, değişkenlerin adlandırma kurallarıyla ilgilidir ve bir değişkenin anlamlı bir ada sahip olmasını gerektirir. Statik kod analizör, değişken adının uzunluğunu ve notasyonunu doğrulayabilir, ancak altta yatan semantiği bilmeden anlamlılığını kavramak zordur. Bu nedenle, tüm önerileri otomatik bir kural olarak uygulamak için statik kod analizi yetersizdir.

Tablo 1
IHI, SRP, FCOI, CCP ve CQS için önerilen tasarım uygulamaları.

DP	Odak Grubu	En İyi Tasarım Uygulaması
IHI	FG-I	Değiştirilemez Nesneleri Kullan Sınıfın İç Yapısını Açığa Çıkarma Yöntem N'de Uygulama Ayrıntılarını Açığa Çıkarmaktan Kaçın.
	FG-III	Gerekirse Sınıfları Halka Açık Hale Getir Katmanların Birbiriyle İlişkisini Kontrol Et
	FG-V	Koleksiyon Dizilerini Geri Dönmekten Kaçın
SRP	FG-I	Durum Bilgilerinin Yinelenmesini Önleme Yöntem Kullanımını Dinamik Olarak Kontrol Etme
	FG-III	Dengesiz Kalıtım ve Yetki Devri Hiyerarşisini Önlemek. İlgisiz Alanları Önlemek
	FG-V	Statik Yöntemlerden Kaçının Uyumsuz Arayüzlerden Kaçının
CCP	FG-I	Tutarlı Adlandırma Kullanın
	FG-III	Alt Paketlerde Paylaşılan Sınıfları Önlemek Birden Çok Pakette Basit Bağımlılıkları Önlemek
CQS	FG-I	Komutlardan Konteyner Nesnelerini Geri Dönmekten Kaçınmak
	FG-III	Mümkün Olduğunda Değiştirilebilir Alanlardan Kaçının Sahte Nesne Durumlarından Kaçının
FCOI	FG-I	Büyük Nesnelerden Kaçın

Tablo 2
DRY, ADP, OCP ve ISP için önerilen tasarım uygulamaları.

DP	Odak Grubu	En İyi Tasarım Uygulaması
DRY	FG-II	Anlamlı Değişken Adları Kullan Özel Yöntem Kullanımını Kontrol Et Yardımcı Sınıfların Gruplandırılmasını Kontrol Et
	FG-VI	Farklı Yapıtlarda Aynı Bilgiden Kaçın Ölü Koddan Kaçın
ADP	FG-II	Uyumsuz Paket Uygulamalarından Kaçın Sıkı Katmanlama Kullan Arayüz Uygulamalarını Bir Arada Tut
	FG-VI	Uygulama Paketlerine Referans Vermekten Kaçının Özellik Enjeksiyonundan Kaçının
ISP	FG-II	İlgisiz Yöntemleri Kontrol Et
	FG-VI	Arayüz Kalıtımından Kaçın
OCP	FG-II	Dış Paket Bağımlılıkları İçin Arayüz Kullan

4.2. Tasarım en iyi uygulamalarının göreceli önemi

İkinci aşamanın sonunda, tüm tasarım en iyi uygulamalarının atanan tasarım ilkesine göre göreceli önemi belirlenmiştir. Bu değerlendirme için katılımcılar beş puanlık b i r ölçekte ilk görüşlerini belirtmek zorundaydılar ve ikinci turda görüşlerini revize edebildiler. Bu, diğer görüşleri dikkate alma ve ilk görüşün grup anlayışından önemli ölçüde farklı olması durumunda ilk görüşü eleştirel bir şekilde değerlendirme fırsatı sağladı. Delphi yöntemi bu değerlendirme turunu tanımlasa da, katılımcılar sadece birkaç durumda ilk değerlendirmelerini değiştirdiler.

Bu aşamanın sonucu, her bir uygulamanın bir ilkeye göre göreceli önemi hakkında 15 görüşten oluşan bir listedir. Önce sıralı ölçek değerlerini sayısal temsiline aktardıktan sonra aritmetik ortalamayı hesaplayarak tüm bireysel yargıları tek bir temsili grup görüşünde topladık. Bu ortalama daha sonra beş puanlık ölçeğe (*çok yüksek (5), yüksek (4), orta (3), düşük (2) ve çok düşük (1))* eşleştirildi, çünkü bu yeterliydi.

Tablo 3
IHI, SRP, FCOI, CCP ve CQS için en iyi tasarım uygulamaları.

DP	En iyi tasarım uygulaması	Önem
IHI	AvoidPublicFields	çok
	DontReturnMutableCollectionsOrArrays	yüksek
	AvoidUncheckedParametersOfSetters	çok
	UseInterfacesAsReturnType AvoidProtectedFields	yüksek
	AvoidSettersForHeavilyUsedFields	yüksek
	AvoidManySetters	yüksek orta
	Çok Sayıda Alıcıdan Kaçının	orta düşük
		çok düşük
SRP	Yapışkan Olmayan Uygulamalardan Kaçının Sınıfların	çok
	Uygun Olmayan İşlevselliğini Kontrol Edin	yüksek
		yüksek
FCOI	Kullanılmayan üst tipleri kontrol et Bileşim kullan,	çok
	miras kullanma	yüksek
		çok
		yüksek
CCP	Yapışkan Olmayan Paketlerden Kaçının Güçlü	çok
	Bağlantılı Paketlerden Kaçının Soyut Paket Diğer	yüksek
	Paketlere Bağımlı Olmamalıdır.	çok
		yüksek
		yüksek
CQS	Sorgu Yöntemlerinde Komutlardan Kaçının	çok
	Komutlardan İlgisiz Verileri Döndürmeyin Komuttan	yüksek
	Veri Döndürmekten Kaçının	çok
		yüksek
		düşük

Tablo 4
DRY, ADP, OCP, ISP ve PINI için en iyi tasarım uygulamaları.

DP	En iyi tasarım uygulaması	Önem
DRY	Yinelemeleri önleme	çok yüksek
	Belge Arayüzleri	çok yüksek
	BelgeGenelSınıfları	yüksek
	Benzer Soyutlamalardan Kaçın	orta
	BelgePublicMethods	orta
	Aynı Tasarım Öğeleri İçin Benzer İsimlerden Kaçın	orta
	Farklı Tasarım Öğeleri İçin Benzer İsimlerden Kaçın.	orta
	Kodda Uzun Yorumlardan Kaçının	düşük
ADP	Paket döngülerinden kaçının	çok yüksek
OCP	Setter'ların Kontrol Edilmemiş Parametrelerinden Kaçın	orta
	AvoidPublicStaticFields	orta
	Değiştirilebilir Koleksiyonlar veya Diziler Döndürmeyin	orta
	UseInterfacesAsReturnType	orta
	AvoidPublicFields	orta
	Çalışma Zamanı Tür Tanımlamasından Kaçın	orta
	Soyutlamaları Kullan	düşük
	Yoğun Kullanılan Alanlar İçin Ayarlayıcıları Önle	çok düşük
	Korunan Alanlardan Kaçın	çok düşük
ISP	Sınıfların Uygun Olmayan İşlevlerini Kontrol Et	yüksek
PINI	Mümkünse Arayüzleri Kullan	yüksek
	Sınıf için Arayüz Sağla	yüksek

ve bu değer temsilini iletmek daha kolaydır. Ayrıca, odak gruplarının hiçbir katılımcısı tarafından kullanılmayan *"ilgili değil" (0)* yanıt seçeneğini de sunduk.
odak gruplarının hiçbir katılımcısı tarafından kullanılmadı. Haritalama için, varsayılan değerin etrafında +/−0,5 puan arasında her sıralı ölçek değerinin aralığını tanımladık; örneğin, aritmetik ortalama
3,3 olan değer *orta düzeyde (3)* olarak değerlendirildi. Önerilen tasarım en iyi uygulamalarının aritmetik ortalamasını hesaplamadık, çünkü bunlar tüm gruplar arasında dağıtılmamış, sadece her grup içinde tartışılmıştı. Tablo 3 ve 4, 10 tasarım ilkesinin tümü için tasarım en iyi uygulamalarını ve grupların görüşlerini göstermekte ve RQ1'e yanıt vermek üzere bulguları özetlemektedir.

OCP hariç, araştırma diğer dokuz ilkenin en az bir tasarım en iyi uygulamasına sahip olduğunu ve bunun yüksek veya çok yüksek öneme sahip olduğunu ortaya koymaktadır. Diğer bir deyişle, çok yüksek veya yüksek olarak değerlendirilen tasarım en iyi uygulamaları, ilkenin tasarım yönünü karşılamaktadır. Bu, uygulamaların ilgili tasarım ilkelerinin belirli kısımlarını ifade ettiğine dair ilk gerekçeyi sunmaktadır. Ancak, önerilen ve tavsiye edilen uygulamaların gücünü anlamak için katılımcılara tahminlerini sorduk.

Tablo 5
IHI, SRP, FCOI, CCP ve CQS'nin eksiksizlik değerlendirmesi.

DP		FG-I		FG-III		FG-V	
IHI	(1)	70,0	10,0	56	16,2	74,0	7,4
	(2)	26,3	9,6	11,0	2	12,0	4,0
	(3)	3,7	4,1	33,0	15,4	14,0	3,7
SRP	(1)	65,0	16,6	56,0	23,3	54,0	4,9
	(2)	17,5	8,3	15,0	4,5	32,0	4,0
	(3)	17,5	14,8	29	22,0	14,0	4,9
FCOI	(1)	87,5	10,3	73,0	16	88,0	4
	(2)	6,5	8	–	–	–	–
	(3)	6,0	10,4	27,0	16,0	12,0	4,0
CCP	(1)	82,5	4,3	52,0	16	82,0	4,0
	(2)	8	4,9	18,0	5,1	–	–
	(3)	9,5	7,3	30,0	20,2	18,0	4,0
CQS	(1)	85,0	5	54,0	12	92,0	9,8
	(2)	12,5	2,5	30,0	8,9	–	–
	(3)	2,5	2,5	16,0	13,6	8	9,8

Tablo 6
DRY, ADP, OCP, ISP ve PINI'nin eksiksizlik değerlendirmesi.

DP		FG-II		FG-IV		FG-VI	
DRY	(1)	64,0	4,9	77,6	2,2	50,4	14,8
	(2)	26,0	3,7	–	–	32,6	11,5
	(3)	10,0	7,1	22,4	2,2	17,0	12,4
ADP	(1)	66,0	15,9	78,6	8,7	54,0	6,9
	(2)	34,0	15,9	–	–	24,0	9,6
	(3)	0	0	21,4	8,7	22,0	9,4
OCP	(1)	73,0	4,0	80,6	16,4	68,0	14,6
	(2)	18,0	6,8	–	–	–	–
	(3)	9,0	6,6	19,4	16,4	32,0	14,6
ISP	(1)	45,0	4,5	80,8	11	44,0	14,9
	(2)	46,0	10,2	–	–	29,0	16,3
	(3)	9	9,2	19,2	11,0	27,0	11,7
PINI	(1)	89,0	12,0	80,8	10,3	72,0	25,7
	(2)	–	–	–	–	–	–
	(3)	11,0	12,0	19,2	10,3	28	25,7

4.3. En iyi tasarım uygulamalarıyla elde edilen eksiksizlik

Tasarım ilkelerinin, atanan tasarım en iyi uygulamalarıyla ne kadar eksiksiz olduğunu anlamak için, katılımcılar 0 ile 100 arasında değişen bir kardinal ölçeği üç parçaya bölmek zorunda kaldılar. Bu parçalar (1) önerdiğimiz tasarım en iyi uygulamaları setiyle elde edilen eksiksizlik, (2) önerilen tasarım en iyi uygulamaları seti ile elde edilen eksiksizlik ve (3) hiçbir uygulama tarafından ele alınmayan tasarım ilkesinin bahsedilmeyen yönleri. Bölüm 3.3 ve Şekil 2, bu görevi bir örnekle açıklamaktadır.

İkinci aşamada olduğu gibi, katılımcılar ilk olarak eksiksizlik konusunda görüşlerini belirtmek zorundaydılar. Bu görevin zorluğunun farkında olduğumuzdan, ilk tahmin için içgüdüsel olarak hareket etmelerini teşvik ettik. Grup üyeleri görev kağıdını geri verdikten sonra, görüşleri bir ara grup sonucu olarak özetledik. Bu sonuç, grup görüşünü eleştirel bir şekilde değerlendirmeleri gereken katılımcılara bireysel olarak dağıtıldı. Bu değerlendirme sürecinde bir katılımcının görüşü değişirse, üç bölümün dağılımını değiştirerek ilk değerlendirmesini değiştirebilirdi.

Son olarak, revize edilen tüm değerlendirmeleri tek bir nihai grup görüşünde özetledik. Tablo 5 ve 6, iki tasarım ilkesi grubu ayrımına göre altı odak grubunun nihai grup görüşlerini göstermektedir. Her tasarım ilkesinin ilk satırı (1), önerdiğimiz tasarım en iyi uygulamaları setinin sağladığı eksiksizliği, ikinci satırı (2) ise eksiklikleri göstermektedir.

2. satır, önerilen tasarım en iyi uygulamaları seti ile elde edilen ek eksiksizlik, 3. satır ise tasarım ilkesinin bahsedilmeyen yönlerini temsil etmektedir. Bir odak grubu belirli bir ilke için herhangi bir öneride bulunmamışsa, bu bölümün değerlendirmesi yapılamamış ve 2. satırda boş bir hücre oluşmuştur.

Tablo 5'e göre, tasarım ilkesi CQS, FG-V tarafından 92,0 puanla en yüksek eksiksizlik değerlendirmesini almıştır. Buna karşılık, FG-III, önerdiğimiz tasarım en iyi uygulamaları setine göre tasarım ilkesi CCP'yi 52,0 puanla en düşük eksiksizlik değeriyle değerlendirmiştir. FG-III'ün CCP için önerdiği uygulamaların kapsama alanı (18,0) dikkate alındığında, bu özel tasarım ilkesi 70,0 puan almaktadır. Tablo 6'da, en yüksek tamlık değerlendirmesi FG-II tarafından 89,0 puanla tasarım ilkesi PINI tarafından elde edilmektedir. Buna karşılık, tasarım ilkesi ISP aynı odak grubu tarafından sadece 45,0 puan almaktadır. Bu nispeten düşük sayının nedeni, önerilen tasarım en iyi uygulamalarının 46,0 puanla yüksek değerlendirilmesi olabilir.

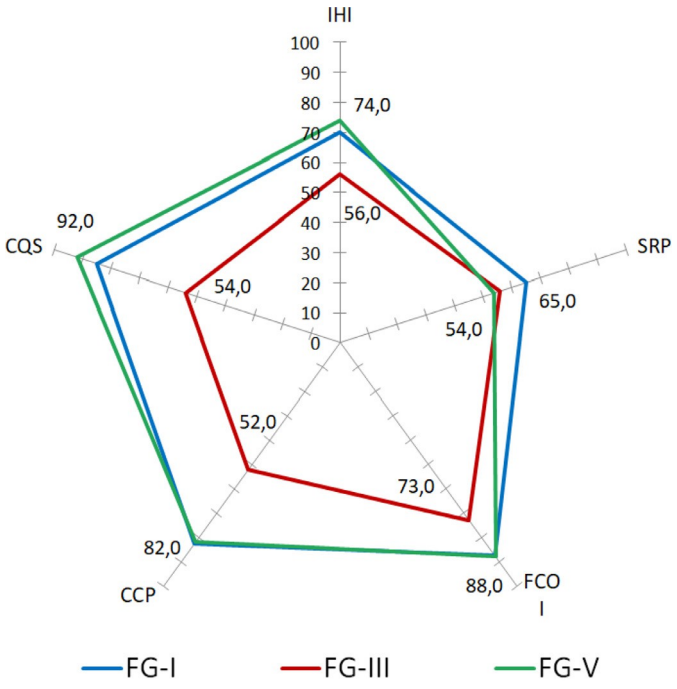
Tamlık değerlendirmelerini daha iyi karşılaştırmak için, Şekil 3(a) ve (b) Bu verileri net bir grafik kullanarak sunun. Her grafik, her bir tasarım ilkesi için bir tane olmak üzere beş boyuta sahiptir. Ayrıca, boyutlar 0 ile 100 puan arasında değişmektedir ve üç parçaya bölünmesi gereken kardinal ölçeği temsil etmektedir. Bu analiz için, önerdiğimiz tasarım en iyi uygulamaları setiyle elde edilen eksiksizliğe odaklanıyoruz. Bu nedenle, belirli bir tasarım ilkesi için üç odak grubunun değerlendirmelerini ilgili boyutta işaretledik. Örneğin, IHI için tasarım en iyi uygulamaları, Şekil 3 (a)'daki dikey boyutta gösterildiği gibi, bireysel odak grupları tarafından 56, 70 ve 74 puanla değerlendirilmiştir. Grafikteki üç şekil, her odak grubunun değerlerini birleştirerek çizilmiştir.

Şekil 3(a)'da yeşil ve mavi şekillerle gösterildiği gibi, iki odak grubu beş tasarım ilkesinin eksiksizliği konusunda neredeyse tamamen aynı görüşte. Her ikisi de CQS ve FCOI'nin eksiksizliğini sırasıyla yaklaşık 92 ve 88 puanla çok yüksek olarak değerlendiriyor. CCP ve IHI iyi bir tamlık düzeyine ulaşırken (sırasıyla yaklaşık 82 ve 74 puan), SRP yaklaşık 55 puanlık bir tamlık düzeyine ulaşmıştır. Kırmızı şekil ile temsil edilen üçüncü odak grubu, tamlık konusunda daha eleştireldir. Bu nedenle, dört ilke yaklaşık 54 puanlık bir tamlık düzeyine ulaşmış ve sadece FCOI için algıları 73 puana çıkmıştır.

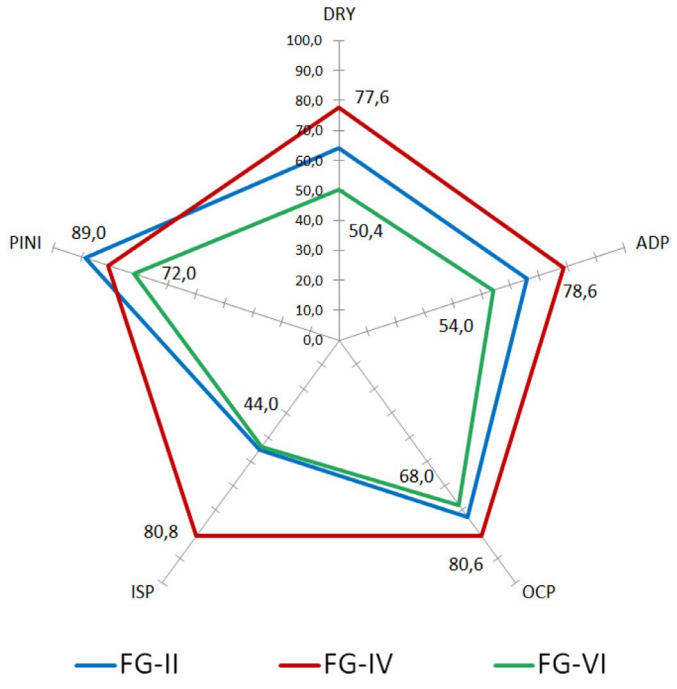
İlk tasarım ilkeleri grubunun sonucuna benzer şekilde, iki odak grubu diğer beş ilkenin eksiksizliği konusunda aynı görüşte. Şekil 3(b) bu iki grubu yeşil ve mavi şekillerle göstermektedir. Şekil 3 (b)'de yeşil grubun diğer gruba kıyasla daha eleştirel olduğu görülmektedir, ancak her iki odak grubu da PINI ve OCP ilkelerine yüksek bir bütünlük atfetmektedir. DRY ve ADP ilkeleri orta düzeyde bir bütünlük (yaklaşık 50 puan) alırken, ISP 44 puanla en düşük bütünlüğe sahiptir.

Kırmızı çizgiyle gösterilen odak grubu, beş ilkenin tamamının eksiksizliğini yaklaşık 80 puan olarak değerlendirmiştir. Bu grup, ek tasarım en iyi uygulamaları için önerilerde bulunma konusunda da mütevazı davranmıştır. Bu çekingenlik ve eksik yönlerin dikkate alınmaması nedeniyle, önerdiğimiz tasarım en iyi uygulamaları setini eksiksizliğe daha fazla katkı sağlayan bir şekilde değerlendirdiklerini varsayıyoruz. Bu etki, ISP ile ilgili olarak belirgindir. Bu ilke için, diğer gruplar önerilerde bulunmuş ve bu önerileri eksiksizliğe yaklaşık 40 puanlık bir katkı ile değerlendirmişlerdir, ancak FG-IV bu eksik yönleri dikkate almamıştır. Bu gözlemin sonuçları aşağıda tartışılmaktadır.

Şekil 4(a) ve (b), önerilen en iyi uygulamalar seti dahil olmak üzere belirli bir tasarım ilkesinin kapsamı hakkında daha fazla ayrıntı vermektedir. Daha spesifik olarak, her odak grubunun kesikli çizgisi, uygulamalar setimizle elde edilen kapsamın değerlendirmesini temsil eder (Şekil 3(a) ve (b)'de gösterilen bilgilerle aynı) ve karşılık gelen düz çizgi, yeni uygulamalar setini dikkate alır.



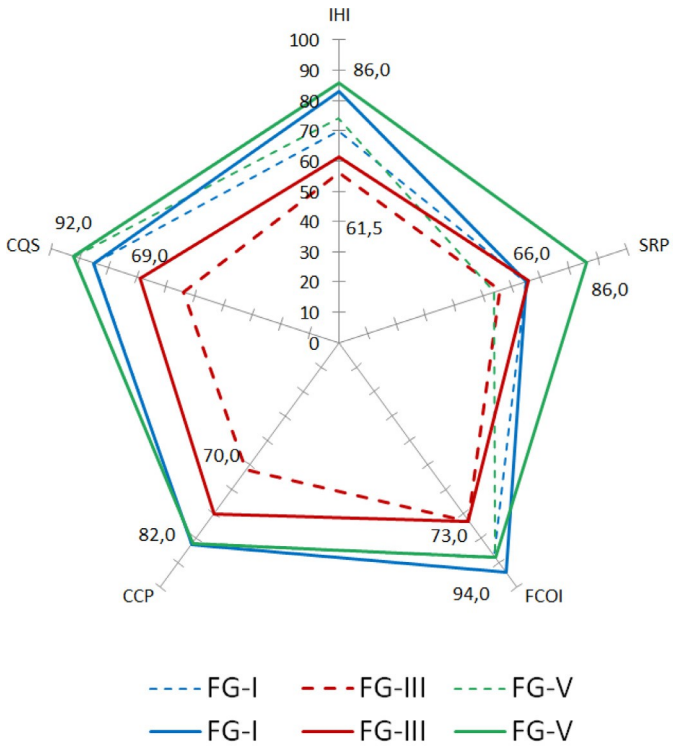
(a) Comparison principle set I



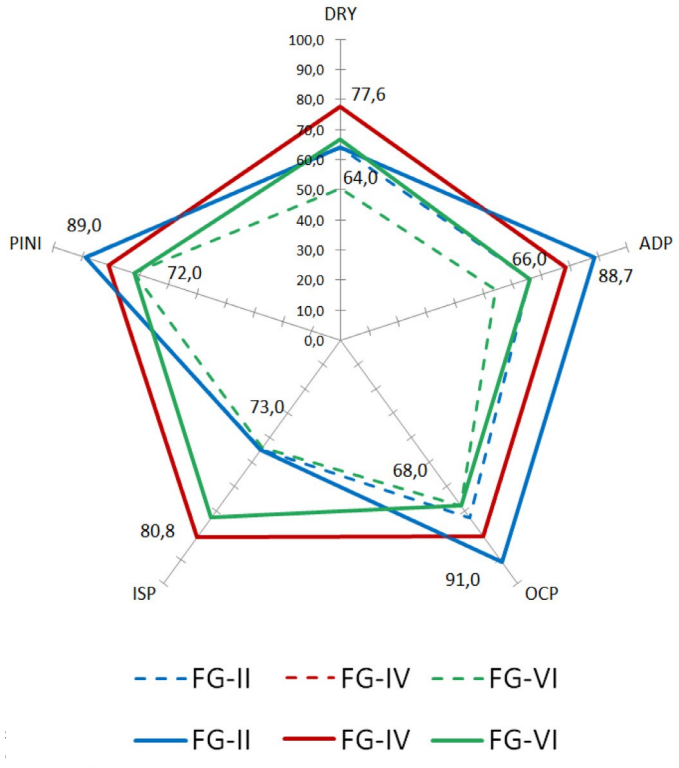
(b) Comparison principle set II

Şekil 3. Tasarım ilkesi kapsamının karşılaştırılması. (Bu şekil açıklamasındaki renklerin yorumlanması için okuyucu, bu makalenin web versiyonuna bakabilir.)

Şekil 4(a)'ya göre, IHI ve SRP, FG-V'nin değerlendirmesine göre 86 puanlık nihai kapsam ile en fazla artış elde edebilir. Genel olarak, FG-III, tasarım en iyi uygulamaları için birçok yeni fikir sunarak beş ilkenin tümü için en az 61,5 puan elde etmiştir. Şekil 4(b)'de FG-VI için de benzer bir tablo görülmektedir ve beş ilkenin tümü için kapsama alanı en az 64 puana yükselmiştir. Şekil 4(b)'de en yüksek kapsama alanına ulaşan ilkeler, FG-II'nin görüşüne göre yaklaşık 90 puanla PINI, ADP ve OCP'dir. Şekil 4(b)'de,



(a) Consideration of suggestions for principle set I



(b) Consideration of suggestions for principle set II

Tablo 7
Tasarım en iyi uygulamalarının ortalama eksiksizlik derecesi.

DP	Ort	Puanlarda Standart Sapma	DP	Ort	Puan Cinsinden Standart Sapma
IHI	66	14	DRY	65	14
SRP	58	17	ADP	74	16
FCOI	83	13	OCP	65	14
CCP	71	18	ISP	57	20
CQS	76	19	PINI	81	19

kırmızı kesikli çizgi eksik gibi görünüyor, ancak aslında kırmızı düz çizgi tarafından gizlenmiştir. Bu durum, FG-IV'ün yukarıda belirtildiği gibi ek en iyi uygulamalar önermemiş olmasından kaynaklanmaktadır.

Her bir tasarım ilkesinin eksiksizliği hakkında temsilci bir grup görüşü elde etmeye odaklanan RQ3'ü yanıtlamak için, standart sapma dahil olmak üzere değerlendirmelerin ortalamasını hesapladık. Tablo 7 bu sonucu göstermektedir. Bu tabloya göre, FCOI en yüksek eksiksizlik değerine ve en düşük standart sapmaya sahiptir. Diğer bir deyişle, gruplar tasarım en iyi uygulamalarının bu ilkenin ana tasarım kaygılarını ele aldığı konusunda hemfikirdir. Aksine, ISP en düşük tamlık değerine ve en yüksek standart sapmaya sahiptir. Bu nedenle, önerilen tasarım en iyi uygulaması (*CheckUnsuitableFunctionalityOfClasses)* ilkenin amacını kavramak için yeterli olup olmadığı konusunda hala bazı belirsizlikler bulunmaktadır.

5. Tartışma ve çıkarılan dersler

Bu bölümde, uygulanan araştırma yöntemi ve nihai sonuçlar tartışılmaktadır. Bu nedenle, bu bölüm tartışma süreciyle tanımlanan üç aşamaya ayrılmıştır. Üç aşamaya geçmeden önce, Delphi yöntemi ile birleştirilen odak grup araştırma yaklaşımı hakkında bazı genel açıklamalar yapacağız.

- *Yeterli zaman kaynağı planlayın:* Bu çevrimiçi odak grup araştırmasını yürütmek zaman alıcı bir işti. İlk olarak, gruplar zaman gecikmeli olarak başladı. Bu durum, grupları yönetme iş yükünü dağıttıysa da, toplam süre uzadı. Daha ayrıntılı olarak, ilk gruba 7 Şubat'ta başladık ve son yanıtı 28 Nisan 2017'de aldık. İkincisi, katılımcılar yavaş yanıt verdi. Başlangıçta dört haftalık bir odak grup tartışması planlamıştık ve ikinci ve üçüncü aşamaların on iş günü içinde tamamlanmasını öngörmüştük. Sonuç olarak, her grubun ortalama süresi yedi haftaya ulaştı.
- *Araç desteği ile tartışmayı yönetme:* Toplamda 31 uzman bu araştırmaya katıldı. Bunlar, farklı tarihlerde başlayan ve eşzamanlı olarak yürütülen altı gruba ayrıldı. Bu grupları uygun araç desteği olmadan yönetmek zor olurdu, çünkü grupların farklı yanıt davranışları vardı ve bu da çeşitli ilerleme düzeylerine yol açıyordu. Tartışmaları yolunda tutmak için, kurs ve içerik yönetim aracı Moo-dle'ı kullandık. Kişisel bildirimler için e-posta ile birlikte Moo-dle, grupları yönetmek için uygun bir araçtı. Bu noktada, mümkün olduğunda kişiselleştirilmiş iletişimin tercih edilmesi gerektiğini belirtmek önemlidir. Bunun nedeni, ödevler bireysel olarak verildiğinde daha hızlı yanıt verme davranışı gözlemlememizdir.
- *Potansiyel katılımcı kayıplarını göz önünde bulundurun:* Kontio ve ark. (2008) adlı çalışmada, yazarlar odak grup katılımcılarının tartışma sürecine devam etme olasılığının daha yüksek olduğunu, çünkü başlangıçta katkıda bulunma taahhüdünde bulunduklarını savunmaktadır. Ancak, bizim araştırmamızda, iki üyenin grup tartışmasından ayrıldığını ve artık ulaşılamadığını gözlemledik. Neyse ki, bu durum ilk odak gruplarının ilk aşamalarında meydana geldi, böylece boş yerler tartışmanın başlamasını bekleyen katılımcılarla dolduruldu. Başka bir deyişle, tartışmalardan ayrılan katılımcılar

henüz başlamamıştı, bu ayrılanların yerini aldı. Bağlantısı kesilen bir grup üyesinin yerini almadan önce, arka plan kontrolü ile yeni katılımcıların diğerlerinin özelliklerine uygun olduğu ve grubun homojenliğinin korunduğu doğrulandı. Sonuç olarak, araştırma tasarımını tehlikeye atmayacak şekilde ayrılanları ele almak için sağlam bir stratejiye sahip olmak önemlidir.

5.1. Beyin fırtınası aşaması

Tüm tartışma, tasarım ilkeleri konusunun tanıtılması ve kendi kendine çalışma materyallerinin sunulmasıyla başladı. Katılımcıları, sunulan materyaller üzerinde çalışmaya ve grup forumuna katılmaya teşvik ettik. Bazı katılımcılar görüşlerini ifade etmekte çekingen davransalar da, bu forumlardaki tartışmalar ilginçti. Yine de, tüm katılımcılar ilk görevi yerine getirerek iyi bir ortak anlayış düzeyine ulaştılar.

5.2. Açıklama aşaması

İkinci aşamadaki ilk görevin sonucu, birkaç grubun yeni tasarım en iyi uygulamaları için öneri sunmakta çekingen davrandığını göstermektedir. Bu konuda en belirgin grup, hiçbir öneri sunmayan FG-IV grubu olmuştur. Grupları her ilke için en az bir fikir sunmaya zorlamak araştırmayı çarpıtabileceğinden, bu konuda müdahale etmedik. Ayrıca, araştırma yöntemi bu konuya ilişkin bir öneri sunmamaktadır. Bu nedenle, tartışmaları grupların bireysel katkılarıyla sürdürdük.

Her bir uygulamanın göreceli önemini değerlendirmek için, öneriler herhangi bir etkiye sahip değildir, çünkü her uygulama ayrı ayrı değerlendirilmiştir. Diğer bir deyişle, bu turdaki değerlendirme görevinde katılımcılardan önerilen ve önerilen tasarım en iyi uygulamalarını sıralamaları değil, her bir uygulamanın göreceli önemini ayrı ayrı ifade etmeleri istenmiştir. Sonuç olarak, katılımcılar aynı koşullarda değerlendirmelerini yaptıkları için, önerdiğimiz tasarım en iyi uygulaması için temsili bir grup önemi belirlemek mümkündür.

Bu odak grup araştırmasından önce, tasarım en iyi uygulamalarının genel önemi hakkında bir anket yaptık, ancak bu anket temel ilişkiyi dikkate almadı (Bräuer et al., 2017b). Bu anketten elde edilen sonuç, tasarım en iyi uygulamalarının sıralamasını ve standart sapmaya dayalı kendi önem aralığını ortaya koydu. Bu önem aralığı, proje gereksinimlerine, kullanılan çerçevelere ve tasarım kalitesinin kritikliğine göre farklılık gösteren proje bağlamına bağlı olarak bir uygulamayı yükseltmek veya düşürmek için kullanılabilir. Bu odak grup araştırmasının sonucunu daha ayrıntılı analiz etmek için, anket sonucunu bu araştırmanın sonucuyla eşleştirdik. Bu nedenle, her bir uygulamayı, göreceli öneminin genel aralık içinde olup olmadığını değerlendirmek için kontrol ettik. Bu, önerilen dört tasarım en iyi uygulaması hariç tüm uygulamalar için geçerliydi.

İlk uygulama, düşükten yükseğe kadar bir önem aralığına sahip olan *DontReturnMutableCollectionsOrArrays*'dir; ancak, IHI bağlamında çok yüksek olarak değerlendirilmektedir. Aynı durum, FCOI ile ilgili olarak *CheckUnusedSupertypes* için de geçerlidir. Diğer iki istisna OCP bağlamında değerlendirilir. Yukarıda kısaca belirtildiği gibi, araştırma OCP için en azından yüksek öneme sahip bir tasarım en iyi uygulaması ortaya koymamıştır. Sonuç olarak, yüksek ila çok yüksek önem aralığına sahip olan *AvoidUncheckedParametersOfSetters* ve *AvoidPublicFields* adlı iki tasarım uygulaması, OCP bağlamında orta düzeyde bir değerlendirmeye indirgenmiştir.

Anketi odak grubu sonuçlarıyla ilke düzeyinde karşılaştırdığımızda, IHI ve DRY için küçük bir fark ortaya çıkmaktadır. Tablo 8 bu sapmayı göstermektedir. IHI bağlamında bu, *AvoidManyGetters* hariç uygulamaların anket sonucuna kıyasla eşit veya daha yüksek bir göreceli önem kazandığı anlamına gelmektedir. Örneğin,

Tablo 8
İlke düzeyinde sapma.

DP	Tasarım En İyi Uygulama	FG Sonucu	Anket Sonucu
IHI	AvoidPublicFields	vh	vh
	DontReturnMutableCollectionsOrArrays	vh	m
	Setter'ların Kontrol Edilmemiş Parametrelerinden Kaçın	h	h
	UseInterfacesAsReturnType	h	h
	Korunan Alanlardan Kaçın	m	l
	Yoğun Kullanılan Alanlar İçin Ayarlayıcıları Önle	m	l
	Birçok Ayarlayıcıdan Kaçın	l	l
	Birçok Alıcıdan Kaçın	vl	l
DRY	YinelemeleriÖnle	vh	vh
	Belge Arayüzleri	vh	h
	BelgeGenelSınıfları	h	h
	Benzer Soyutlamalardan Kaçın	m	h
	BelgeGenelYöntemleri	m	h
	Aynı Tasarım Elemanı İçin Benzer İsimlerden Kaçının.	m	h
	Farklı Tasarımlar İçin Benzer İsimlerden KaçınınE.	m	h
	Kodda Uzun Yorumlardan Kaçının	l	m

çok yüksek (vh), yüksek (h), orta (m), düşük (l), çok düşük (vl).

DontReturnMutableCollectionsOrArrays, anket orta derecede önemli olduğu sonucuna varmış olsa da çok önemli kabul edilmektedir. DRY için bu tablo biraz farklıdır, çünkü odak grup üyeleri daha belirgin bir eğilim göstermiştir. Bu nedenle, *Avoid-Duplicates, DocumentInterfaces* ve *DocumentPublicClasses'ı* anketten elde edilen benzer bir önem düzeyiyle değerlendirmişlerdir, ancak diğer uygulamalar orta veya düşük düzeye indirgenmiştir.

Özetlemek gerekirse, neredeyse tüm durumlarda odak grup katılımcıları aynı veya daha yüksek bir önem seviyesi vermiştir. Tasarım en iyi uygulamasının genel tasarım kalitesiyle olan ilişkisi yerine, daha spesifik bir tasarım ilkesiyle olan ilişkisi daha kolay anlaşılabilir olduğundan, bunu oldukça doğal buluyoruz. Tasarım en iyi uygulaması *AvoidMassiveCommentsInCode* bu kuralın bir istisnasıdır. Bu bağlamda DRY'nin temelinde yatan fikir, uygun adlandırma ve uygulama programcısı arayüzünün iyi bir şekilde belgelenmesi ile uygun bir kod yapısı seçmek ve yöntemlerin uygulanmasını yorumlamamaktır. Bu uygulama, odak grup katılımcıları için mantığa aykırı görünmektedir.

5.3. Tamlık değerlendirme aşaması

Odak grubunun eksiksizlik değerlendirmelerinin sonucu, özellikle her ilke setindeki iki grubun önerdiğimiz uygulama seti için neredeyse aynı değerlendirmeye varmış olmasının etkisi, tartışma için bir alan sağlar.

5.3.1. İlk ilke kümesindeki uç değerlerin tartışılması

İlk tasarım ilkeleri setinde, FG-I ve FG-V neredeyse aynı eksiksizlik değerlendirmesi sonucuna varmıştır. Eksiksizlik değerlendirmesini etkileyebilecek bir değişken, grupların her ilke için sunduğu öneri sayısıdır. Tablo 1 ve 2 bu bilgileri göstermektedir. Burada görüldüğü gibi, FG-I tüm ilkeler için ek uygulamalar önerirken, FG-V sadece IHI ve SRP için önerilerde bulunmuştur. Bu nedenle, FG-V'nin belirli uygulamalar sunmamış olmasına rağmen eksik yönleri dikkate alıp almadığı ilginçtir. Aslında, ortaya çıkarılmamış tasarım yönlerinin değerlendirilmesinde de görüldüğü gibi, eksik yönler dikkate alınmıştır. Bu gözlemin iyi örnekleri FCOI ve CCP'dir. Genel olarak, önerilerin sayısının iki grubun algılarını etkilemediği sonucuna vardık.

Bu bulguya dayanarak, FG-I ve FG-V'nin özellikleri bakımından benzer oldukları, ancak FG-III'ten başka bir açıdan farklı oldukları varsayılabilir. Sonuç olarak, her bir grup içindeki standart sapmalar analiz edildi. Bu analiz, FG-III'ün her bir ilke için en yüksek standart sapmaya sahip olduğunu göstermektedir. Başka bir deyişle, FG-III bu ilke kümesinde en heterojen gruptur. Daha spesifik olmak gerekirse, en düşük

Tablo 9
Önerilerin katkısı I.

FG	Tasarım En İyi Uygulama	Göreceli Önem	Tamamlanma Katkısı
FG-I (26.3)	Mümkünse Arayüzleri Kullanın	4.2	6
	Değiştirilemez Nesneleri Kullan	2	6
	Sınıfın İç Yapısını Açığa Çıkarma	4	–
	AvoidExposingImpl.DetailsInMethodN.	4.	–
FG-III (11.0)	GerekirseSınıflarıHalkaAçıkYap	4.6	5.5
	Katmanların Birbiriyle İlişkisini Kontrol Et	3	–
FG-V (12.0)	Koleksiyon Dizilerini Geri Dönmekten Kaçın	3.6	12
FG-I (17,5)	Durum Bilgilerinin Yinelenmesini Önleme	3.6	–
	YöntemKullanımınıDinamikOlarakKontrolEt	2.0	–
FG-III (15.0)	Dengesiz Kalıtımdan Kaçın.Hiyerarşi	3.6	–
	İlgisiz Alanlardan Kaçın	4.	5.
	KompozisyonKullan, KalıtımKullanma	5	5
FG-V (32.0)	Statik Yöntemlerden Kaçınma	4.4	16
	Yapışkan Olmayan Arayüzlerden Kaçın	3	16
FG-I (6,5)	Büyük Nesnelerden Kaçın	2	6
FG-I (8,0)	TutarlıAdlandırmaKullan	1	–
FG-III (18,0)	Alt Paketlerde Paylaşılan Sınıfları Önleme	4.	9
	Birden Fazla Paket Arasında Basit Bağımlılıklardan Kaçın.	2	9
FG-I (12,5)	AvoidReturningCont.Obj.FromComm.	1.4	–
FG-III (30,0)	Mümkün Olduğunda Değiştirilebilir Alanlardan Kaçın	4	15
	Sahte Nesne Durumlarından Kaçının	5	–

FG-III için standart sapma 12,0 puandır, FG-V'deki hiçbir ilkenin standart sapması ise 10 puanı geçmemektedir. Standart sapması 16,6 puan olan SRP hariç (FG-III'ün SRP için ortalama değerlendirmesi 23,3 puan sapmaktadır), aynı durum FG-I için de geçerlidir. Bu bulguya göre, FG-III'teki iki veya üç üyenin, eksiksizlik değerlendirmesine daha eleştirel b i r bakış açısıyla yaklaştığı açıktır. Sonuç olarak, FG-III'ün genel görüşü, kritik algıları azalttığında diğerlerine daha yakın hale gelecektir.

5.3.2. İkinci ilke setindeki uç değerlerin tartışılması

İkinci tasarım ilkeleri setinde, FG-II ve FG-VI değerlendirmelerinde birbirleriyle uyumludur, ancak FG-IV bir istisna oluşturmaktadır. Yukarıda kısaca bahsedildiği gibi, FG-IV katılımcılarının hiçbirinin yeni bir tasarım en iyi uygulaması için öneride bulunmaması, bu konuya derinlemesine girilmediğini desteklemektedir. Sonuç olarak, DRY, ADP, OCP, ISP ve PINI için temsil edici bir grup görüşü muhtemelen fazla iyimserdir ve pratikte kullanıldığında bir şekilde düşürülmelidir.

5.3.3. Önerilerin değerlendirilmesi

İkinci aşamada önerilen en iyi uygulamalar, ölçüm aracımız MUSE'yi geliştirmek için değerli bir katkıdır. Bunların otomatik olarak uygulanıp uygulanamayacağını ve kontrol edilip edilemeyeceğini değerlendirmek için, üç MUSE geliştiricisi her bir önerinin temelindeki fikri ayrı ayrı inceledi. Geliştiricilerden biri uygulanabilirlik konusunda farklı bir görüşe sahipse, ortak bir tartışma sonucunda öneri hakkında ortak bir anlayışa varıldı ve otomatikleştirilme potansiyeli konusunda nihai bir anlaşmaya varıldı.

MUSE'de uygulanabilir olma şartının yanı sıra, önerilerin ilkenin eksiksizliğine katkısı da değerlendirildi. Bunu yapmak için, grup değerlendirmesini her ilke ve her grup için öneri sayısına böldük. Örneğin, FG-I, IHI için dört önerisinin toplam 26,3 puan olduğunu sonucuna vardı. Diğer bir deyişle, bu değerlendirmeyi önerilen en iyi uygulamaların sayısına bölerek (26,3 bölü 4 öneri), bir otomatik öneri 6,6 puanlık bir katkı sağlıyor. Tablo 9 ve 10, önerileri göstermektedir.

son sütun tarafından uygulanabilecek öneriler. Bu nedenle, otomatikleştirilemeyen öneriler için hücre boş bırakılır; aksi takdirde, hücre tamlığa katkıyı temsil eder.

Her kuralın bir ilkenin eksiksizliğine katkısı hakkındaki bilgiler göz önüne alındığında, uygulanmaya değer adaylar türetilebilir. Örneğin, *AvoidInterfaceInheritance*, FG-VI'ya göre ISP için önemli bir tasarım yönünü temsil eder. Diğer örnekler arasında, SRP'yi ifade etmek için yüksek katkı sağlayan *AvoidStaticMethods* ve *AvoidNonCohesiveInterfaces* veya OCP'nin başka bir tasarım yönü olan *UseInterfaceForExternalPackageDependencies* sayılabilir.

Tablo 9 ve 10, uygulanabilir önerilerin göreceli önemini gösterirken, otomatikleştirilemeyen kuralların göreceli önemini de göstermektedir. Kalite yöneticisi bu bilgileri kullanarak, bu uygulamaları, özellikle göreceli önemi yüksek olanları kontrol etmenin alternatif yollarını bulabilir. Örneğin, *Avoid-PretendedObjectStates* önerisi CQS için çok önemli kabul edilir, ancak MUSE'de uygulanamaz. Dolayısıyla, kaynak kodunda bu kuralı kontrol etmek için başka teknikler düşünmek mantıklıdır. Diğer adaylar, sırasıyla değişkenlerin ve yöntemlerin temel semantiğini anlamayı gerektiren *UseMeaningfulVariable-Names* ve *CheckUnrelatedMethods*'dur.

5.3.4. Tasarım ilkelerini yorumlamadaki belirsizlik

Odak grup araştırması sırasında, bazı katılımcılar ilkelerin yorumlanması konusunda belirsizliklerini dile getirdiler. Bu konuyu anlamak için, her bir tasarım ilkesinin ortaya çıkan yönlerini yansıtan değerlendirmelerin ortalamasını inceledik. İlkeleri bu değere göre sıraladığımızda, Tablo 11'de gösterildiği gibi b i r sıralama elde ettik. Tabloya göre, OCP, SRP ve CCP, ortaya çıkmamış tasarım yönlerinin en yüksek derecesine sahiptir. Bu, tanımlarının hala yorumlamaya açık olduğunu ve anlaşılması zor olduğunu gösterebilir. Aksi takdirde, gruplar tasarımda en iyi uygulamalar için önerilerde bulunmuş veya ortaya çıkmamış yönleri düşük olarak değerlendirmiş olurlardı.

Sıralamanın karşı tarafında ise, tasarımın ele alınmamış yönleri konusunda en düşük puanları alan CQS, ADP ve FCOI yer almaktadır. Bu konular ya önerilerle ele alınmış ya da ilke önceden belirlenmiştir.

Tablo 10
Önerilerin katkısı II.

FG	Tasarım En İyi Uygulama	Göreceli Önem	Tamamlanma Katkısı
FG-II (26,0)	Anlamlı Değişken Adları Kullanma	4	–
	Özel Yöntem Kullanımını Kontrol Et	2	–
	Yardımcı Sınıfların Gruplandırılmasını Kontrol Et	3.4	–
FG-VI (32.6)	Farklı Yapıtlarda Aynı Bilgiden Kaçın	4.	–
	Ölü Kodu Önle	4.2	16,3
FG-II (34,0)	Yapışkan Olmayan Paket Uygulamalarından Kaçının	3	11
	Sıkı Katmanlama Kullan	3	–
	Arayüz Uygulamalarını Bir Arada Tut	3.2	11.3
FG-VI (24.0)	AvoidReferencingImplementationPkg.	3.7	12
	ÖzellikEnjeksiyonundanKaçının	2	–
FG-II (18.0)	UseInterfaceForExternalPackageDep.	4.4	18
FG-II (46.0)	İlgisiz Yöntemleri Kontrol Et	4.	–
FG-VI (29,0)	Arayüz Kalıtımından Kaçınma	3,5	29

Tablo 11
İlke sıralaması.

DP	Kapsanmayan yönlerin ortalaması
OCP	20,8
SRP	20,4
CCP	19,9
PINI	18,8
ISP	18,5
IHI	17,9
DRY	16
FCOI	15,6
ADP	15,4
CQS	9,3

uygun tasarım en iyi uygulamaları önceden belirlenebilecek kadar kesin olarak tanımlanmıştır. Hangi gerçek olursa olsun, bu ilkelerin daha açık olduğu ve tasarım en iyi uygulamaları tarafından takip edilebileceği sonucuna varıyoruz.

6. Sınırlamalar

Herhangi bir deneysel çalışmada, bazı faktörler bulguları etkiler ve geçerliliğe tehdit oluşturur. Daha ayrıntılı olarak, iç geçerliliğe yönelik tehditler, sonucu etkileyebilecek herhangi bir karıştırıcı değişkenle ilgilidir (Wohlin ve ark., 2012). Yerinde (yüz yüze) tartışmalar yerine çevrimiçi odak grup tartışmaları yapma kararı, bizim algımıza göre böyle bir tehdit oluşturmaktadır. Bu nedenle, metin yoluyla iletişim, beden dili veya yüz ifadeleri eksik olduğu ve metin yanlış anlaşılabileceği için daha az zengindir (Kontio ve ark., 2008). Anonimlik adına, tartışmalarda beden dilinin eksikliğini kabul ettik. Ayrıca, açık tartışma kısmı, katılımcıları aynı seviyeye getirmek üzerine odaklanan beyin fırtınası aşamasına indirgendi. Yanlış anlaşılma sorununu çözmek için, örneğin önerilen bir tasarım en iyi uygulaması gibi konuları açıklığa kavuşturmak için katılımcılara açıkça danıştık.

Geçerliliğe yönelik bir diğer iç tehdit ise katılımcıların seçimi ve segmentasyonudur. Bu sorunu kontrol etmek için, önceki bir anketi kullanarak gönüllülerin yazılım mühendisliği becerilerini kontrol ettik. Daha spesifik olarak, nesne yönelimli programlama uzmanlıklarını iyi veya en iyi olarak değerlendiren katılımcılar odak grubuna katılmaya davet edildi. Bu katılımcıların yeniden katılımı, diğerlerine göre bilgi açısından daha ileride olmaları sorununa yol açabilir. Bu tehdidi azaltmak için, tasarım en iyi uygulamalarımızla ilk kez tanışan katılımcılara da aynı bilgileri sağladık. Araştırma ortağımız tarafından düzenlenen üç odak grubu, en üst düzey mühendislik yetkinliklerine sahip ve hırslı kıdemli yazılım mühendislerinden oluşuyordu.

yazılım tasarım bilgilerini pekiştirmek. Ayrıca, katılımcıların segmentasyonu, ekipler arasında homojenliği sağlamaya odaklanacak şekilde gerçekleştirildi. Bu nedenle, aynı şirketin üyeleri aynı grubun parçasıydı.

En kritik iç tehditler, sağlanan materyal ve tasarım ilkeleri ile tasarım uygulamaları arasında önerilen ilişkiler tarafından ortaya çıkmış olabilir. Her tasarım ilkesi, bilgileri sistematik olarak yapılandıran Moodle'da ayrı bir ders kullanılarak açıklanmıştır. Ancak, tasarım ilkelerinin anlaşılmasını doğrulamamış, bunun yerine katılımcılardan her ilkenin bir yönünü veya örneğini grup forumunda tartışmalarını istemiştir. Bu tartışmalara dayanarak, herkesin her ilkenin ana amacını anladığından emin olabilmiştir. Katılımcıları ilkeler ve uygulamalar arasındaki eşleşmelerle karşı karşıya getirmek, bir dereceye kadar ilgili bir ilişki olduğunu düşündürür. Sonuç olarak, katılımcılar bu ilişkiye karşı çıkma eğiliminde olmayacaktır. Ancak, b i r grubun bir ilke ve uygulama arasındaki önerilen ilişkiye karşı oy kullandığı ve böylece ödevle ilgili eleştirel düşüncelerini ortaya koyduğu örnekler gözlemledik.

Dış geçerliliğe yönelik tehditler, sonuçları genelleştirme yeteneği ile ilgilidir (Wohlin et al., 2012). Bu bağlamda, tasarım ilkeleri nesne yönelimli programlama dillerinden bağımsız olarak tartışılmış ve herhangi bir uygulama bağlamından kopuk olduğu için büyük bir tehdit görmüyoruz. Bununla birlikte, şu anda Java, C++ ve C# için bir dizi 67 tasarım en iyi uygulaması (bkz. Plösch et al. (2016b)) biliyoruz.

Java, C++ ve C# için kurallar içeren bir dizi en iyi tasarım uygulaması biliyoruz (bu kuralların bazıları,
Bu nedenle,
Bu bulgular diğer programlama dillerine de aktarılabilir, ancak bu işlem özel bir dikkat gerektirir; C++ için çoklu miras veya makrolar gibi ek tasarım özellikleri mevcuttur.

Katılımcı sayısı da bu çalışmada bir sınırlama oluşturabilir. Toplamda, 15 (+1) katılımcı tarafından iki grup halinde beş ilke ayrı ayrı tartışılmıştır. 15
bir ilkeye ilişkin 15 görüşün hala çok az olduğu iddia edilebilir, ancak
Konunun karmaşıklığı, örneğin çevrimiçi anket şeklinde geniş çaplı bir araştırma yaparak elde edilemeyecek tasarım ilkelerinin derinlemesine incelenmesini gerektirir. Tipik bir anket ortamında verilen kısa sürede tasarım ilkeleri ve en iyi tasarım uygulamaları üzerine eleştirel bir değerlendirme yapılamayacağı için bu mümkün değildir. Sonuç olarak, küçük bir grubun tek bir konu üzerinde odaklanmış bir tartışma yapmasının daha değerli içgörüler ortaya çıkardığı sonucuna varıyoruz.

7. Sonuç ve gelecekteki çalışmalar

Yazılım mühendisliğindeki tasarım ilkeleri, tasarımları iletmeleri nedeniyle yazılım geliştiricileri ve tasarımcıları için çok önemlidir.

bakım kolaylığı, işlevsel uygunluk ve taşınabilirlik gibi iç kalite özelliklerini (yetenekleri) sağlayan yazılımlar geliştirmek için gerekli bilgiler. Ancak, bunlar pratikte uygun şekilde uygulanamayacak kadar belirsizdir. Bu nedenle, takip edilebilecek kadar somut olan tasarım en iyi uygulamalarını kullanarak tasarım ilkelerini işlevsel hale getirmeyi amaçlıyoruz. Daha önce tasarım ilkeleri ile atanan en iyi uygulamalar arasındaki ilişkiyi yansıtan bir tasarım kalitesi modeli oluşturmuş olsak da (Plösch et al., 2016a), bu ilişkinin gücünün ne olduğunu veya belirli bir ilke için tasarım en iyi uygulamalarını unutmuş olup olmadığımızı bilmiyoruz. Bu araştırma soruları nedeniyle, bu çalışma gerçekleştirildi ve topluma önemli katkılar sağladı.

İlk olarak, odak grup araştırması, atanan tasarım ilkeleriyle ilgili olarak tasarım en iyi uygulamalarının önemine dair net bir tablo ortaya koydu. Bu tablo, yazılım tasarımı konusunda sağlam bir bilince sahip en az 15 kıdemli yazılım mühendisinin (beş kişilik beş grup ve altı kişilik bir grup – her biri beş tasarım ilkesini değerlendiriyor) görüşlerine dayanmaktadır. Bulgulara dayanarak, tasarım iyileştirme eylemleri önceden seçilebilir ve en yüksek iyileştirme etkisine sahip çabaya yatırım yapmak için önceliklendirilebilir. Bir proje ekibinin IHI ile uyumluluğunu artırmak istediğini varsayarsak, korumalı alanlardan kaçınmak, kamuya açık alanlardan kaçınmaktan daha önemlidir. Böylece, elde edilen bu bilgiler, iyileştirmelerin en etkili şekilde gerçekleştirilmesi için kılavuz olarak kullanılır.

Bu bulgular sadece tasarım iyileştirmeleri için değil, tasarım değerlendirmelerinin kalite görevi için de önemlidir. Örneğin, yukarıdaki örnekte, bir sınıftaki genel alanlar, korumalı alanlara kıyasla daha kritiktir. Aslında, ağırlık türetilen önem düzeyine göre belirlenir ve örneğin bilgi gizleme uyumluluğunu değerlendirmek için kullanılabilir. Plösch ve ark. (2016a) çalışmasında, bu amaca uygun ve türetilen ağırlıklara göre ayarlanabilen b i r tasarım kalitesi modelinin örneğini göstermekteyiz.

Bir diğer katkı ise, belirli bir tasarım ilkesini etkileyen eksik tasarım uygulamalarının belirlenmesi. Aslında, altı odak grubu tarafından 32 ek tasarım uygulaması önerildi. Önerilen tasarım uygulamalarının bazıları manuel önlemler olduğundan otomatikleştirilemezken, 18 uygulama bir dizi tasarım analizi çerçevesinde, özellikle MUSE araçlarımızda uygulanabilir. Gelecekteki çalışmalarımızın gündeminde, bu önerilerin uygulanması ve bunların bir tasarım ilkesi ile ilgililiğinin ortak değerlendirilmesi yer almaktadır. Bu değerlendirme için ipuçları, uygulamayı öneren ekibin grup görüşünden elde edilebilir.

Son olarak, elde edilen her bir tasarım ilkesinin eksiksizliğinin değerlendirilmesi, tasarımın çoğu yönünün tasarım en iyi uygulamaları tarafından kapsandığını göstermektedir. Diğer bir deyişle, hiçbir ilkenin merkezi bir unsuru eksik değildir. Ayrıca, yüksek derecede kural otomasyonu, tasarım ilkelerinin operasyonel hale getirilmesinin otomatik olarak gerçekleştirilebileceğini göstermektedir.

Gelecekteki araştırma yolları için, bu çalışmadan elde edilen bulgulara dayanarak endüstriyel bir ortamda tasarım kalitesi modelini incelemeyi planlıyoruz. Wieringa ve Moralı (2012) tarafından önerilen teknik eylem araştırması yaklaşımını kullanarak, tasarım kalitesi değerlendirmeleri için uygunluğu ve yararlılığı açısından tasarım kalitesi modeli değerlendirilecektir. Ayrıca, iyileştirme eylemlerini yönlendirme kalite görevini ele alan (karşılaştırma tabanlı) bir tasarım iyileştirme portföyü yaklaşımı yayınladık (Bräuer et al., 2017a). Bu yaklaşım, tasarım en iyi uygulamaları düzeyinde tanıtılmıştır. Artık, bu çalışmanın sonuçları portföy tekniğine dahil edilerek tasarım ilkeleri düzeyinde daha iyi öneriler sunulabilir. Bu tasarım iyileştirme kılavuzunun değerlendirilmesi beklemededir. Genel olarak, tasarım ilkelerinin daha iyi anlaşılması, yazılım mühendisliği disiplini için çok önemlidir, çünkü bunlar güçlü tasarım kılavuzlarıdır.

Stevenson ve Wood (2017) tarafından belirtilmiştir. Bu nedenle, gelecekteki çalışmalar bu konuya odaklanmalıdır.

Ek A. Tasarımda en iyi uygulamalar

- *AbstractPackagesShouldNotDependOnOtherPackages:* Çok sayıda soyut sınıf ve arayüz içeren bir paket, diğer paketlere bağımlı olmamalıdır.
- *AvoidCommandsInQueryMethods:* Sorgu yöntemi olarak tanımlanan bir genel yöntem, nesnenin durumunu değiştirmemeli ve yalnızca aynı sınıfın sorgu yöntemlerini çağırabilmelidir. Bir genel yöntem, adı get, has veya is gibi tanımlanmış bir önekle başladığında sorgu yöntemi olarak tanımlanır.
- *AvoidDuplicates:* Kaynak kodunda yinelenenler olmamalıdır.
- *AvoidManyGetters:* Getter yöntemleri ile toplam alan sayısı arasındaki oran belirli bir eşiği aşmamalıdır.
- *AvoidManySetters:* Setter yöntemleri ile toplam alan sayısı arasındaki oran belirli bir eşiği aşmamalıdır.
- *AvoidMassiveCommentsInCode:* Bir yöntemin kodunda çok fazla yorum satırı olmamalıdır. Yöntem belgeleri (API belgeleri) ve boş satırlar dikkate alınmaz.
- *AvoidNonCohesiveImplementations:* Bir sınıf, birbiriyle ilişkili olmayan yöntem kümelerine sahip olmamalıdır. İlişkili olmak, aynı alan kümesini kullanmaları/değiştirmeleri veya yöntem çağrılarıyla bağlantılı olmaları anlamına gelir.
- *AvoidNonCohesivePackages:* Bir paket mümkün olduğunca tutarlı olmalıdır, yani paketler bağımsız sınıf grupları içermemelidir.
- *AvoidPackageCycles:* Farklı paketlerdeki sınıfların ve arayüzlerin kullanımı, paket döngüsü oluşturmamalıdır.
- *AvoidProtectedFields:* Bir sınıfta korumalı alanlar bulunmamalıdır.
- *AvoidPublicFields:* Bir sınıfta genel alanlar bulunmamalıdır.
- *AvoidPublicStaticFields:* Bir sınıfta global değişkenler, yani genel statik alanlar bulunmamalıdır.
- *AvoidReturningDataFromCommands:* Komut yöntemi olarak tanımlanan genel bir yöntem, verilerin nesnenin iç durumuyla ilgili olup olmadığına bakılmaksızın herhangi bir veri döndürmemelidir.
- *AvoidRuntimeTypeIdentification:* Nesnelerin tür denetimleri, yani Java'da instanceof operatörünün veya C++'da typed operatörünün ve dynamic_cast operatörünün kullanımı kaçınılmalıdır.
- *AvoidSettersForHeavilyUsedFields:* Bir sınıfta setter olmamalıdır. yoğun olarak kullanılan özel bir alan için yöntemler. Bir alan, getter ve setter yöntemleri dahil olmak üzere beşten fazla yöntemde okunur veya yazılırsa yoğun olarak kullanılmış sayılır.
- *AvoidSimilarAbstractions:* Farklı türler benzer bir yapı veya davranışı temsil etmemelidir. Aynı türde ve benzer ada (kelime kökü) sahip alanlar belirli bir yüzdeyle örtüşüyorsa, iki sınıf benzer bir yapıya sahiptir. Aynı dönüş türüne ve parametre türlerine sahip yöntemler ile benzer ada (kelime kökü) sahip yöntemler belirli bir yüzdeyle örtüşüyorsa, iki sınıf benzer bir davranışa sahiptir.
- *AvoidSimilarNamesForDifferentDesignElements:* Farklı türdeki tasarım öğeleri benzer isimlere sahip olmamalıdır, yani bir paket adı bir sınıf adına benzer olmamalıdır.
- *Aynı Tasarım Öğeleri İçin Benzer İsimlerden Kaçının:* Aynı türden tasarım öğeleri (örneğin, (soyut) sınıflar, arayüzler veya paketler) benzer isimlere sahip olmamalıdır.
- *AvoidStronglyCoupledPackages:* Bir paket diğer paketlere aşırı derecede bağımlı olmamalıdır. Bu nedenle, diğer paketlerin türlerine bağımlı olan birçok sınıf içeren paketlerden kaçınılmalıdır.
- *AvoidUncheckedParametersOfSetters:* Bir alan, yalnızca önceden kontrol edilmiş bir yöntem parametresi tarafından ayarlanmalıdır. Bu, bir (set) yönteminin parametresi tarafından alanın ayarlanmasının her zaman (veya en azından sıklıkla) kontrollerle korunup korunmadığını kontrol ederek doğrulanabilir.

- *CheckUnsuitableFunctionalityOfClass:* Bir sınıfın yöntemleri, sadece (küçük) kümeler halinde değil, bir bütün olarak kullanılmalıdır. Müşteriler genellikle bir sınıfın sağladığı yöntemlerin sadece bir kısmını kullanıyorsa, bu sınıfın işlevselliği müşterilerin ihtiyaçlarına uygun değildir.
- *CheckUnusedSupertypes:* Bir sınıfın istemcileri (alt sınıflar değil!) yalnızca mevcut alt türün genel yöntemlerini kullanıyor ve üst türün yöntemlerini kullanmıyorsa, sınıf (alt tür) ile üst türü arasında gerçek bir "is-a" ilişkisi yoktur.
- *DocumentInterfaces:* Bir arayüzün, arayüz bildirimi ve her yöntem imzası için bir API belgeleri olmalıdır.
- *DocumentPublicClasses:* Bir genel sınıf ve genel yapı (C++'da) API belgeleri, yani belirtimi veya belirli varlığın tanımı üzerinde yorumlar olmalıdır.
- *DocumentPublicMethods:* Genel bir sınıftaki genel bir yöntem, API belgelerine, yani yöntem bildiriminin veya belirli varlığın tanımının üstünde yorumlara sahip olmalıdır.
- *DontReturnMutableCollectionsOrArrays:* Bir yöntem, bir dizi veya bir koleksiyon türünün örneğini döndürmemelidir. Dönüş değeri önceden değiştirilemez veya klonlanmışsa, yöntem bu kuralın dışında tutulur.
- *DontReturnUninvolvedDataFromCommands:* Nesnenin veya sınıfın durumunu değiştiren bir komut yöntemi, değişiklikle ilgili olmayan verileri döndüremez.
- *ProvideInterfaceForClass:* Genel bir sınıf, değişkenler ve parametreler için tür olarak kullanılan bir arayüz sağlamalıdır. Yalnızca statik üyelere erişim sağlayan sınıflar bu kuralın dışında tutulur.
- *UseAbstractions:* Bir paket, soyut ve somut türler arasındaki oranla ifade edilen yeterli sayıda soyut sınıf ve arayüz sağlamalıdır.
- *UseCompositionNotInheritance:* Bir sınıf, belirli bir üst sınıftan yalnızca genel üyelere eriştiğinde, miras yerine bileşim kullanmalıdır. Arayüzler ve soyut sınıflar bu kuralın dışında tutulur.
- *UseInterfacesAsReturnType:* Bir yöntemin dönüş türü temel veri türü değilse, sınıfın arabirimi veya soyut üst sınıfı olmalıdır.
- *UseInterfacesIfPossible:* Arayüz gerekli tüm yöntemleri sağladığında, değişken bildirimleri, parametre tanımları veya dönüş türleri için genel sınıf yerine arayüz kullanın.

Ek B. Önerilen en iyi uygulamalar

- *UseImmutableObjects:* Mümkün olduğunca değişmez nesneler kullanın, yani yapıcılar ve const/final üyelerinde alan başlatma.
- *DontExposeInternalStructureOfClass:* İç tasarım kararları, sınıf arayüzü (yöntem adlandırma, ayrıntı düzeyi, parametreler ve dönüş değerleri) aracılığıyla müşteriler tarafından tanınabilir olmamalıdır. Örneğin, bir istemci bir noktanın kutupsal koordinatlar mı yoksa Kartezyen koordinatlar mı olarak depolandığını bilmemelidir (örneğin, Kartezyen koordinatlar için "getX()" ve "getY()" yöntemini, kutupsal koordinatlar için ise "calculateRadius()" ve "calculatePolarAngle()" yöntemini kullanarak).
- *AvoidExposingImplementationDetailsInMethodNames:* Genel bir yöntemin adı, uygulama ayrıntılarını ortaya çıkarmamalıdır.
- *GerekirseSınıflarıGenelYap:* Bir bileşen veya paketin genel arayüzünün parçası olmayan bir sınıf veya veri yapısı özel olmalıdır.
- *CheckInterrelatednessofLayer:* Hangi katmanların diğer katmanlarla etkileşime girebileceği açıkça belirtilmelidir.
- *AvoidReturningCollectionsArrays:* Dizi koleksiyonları veya koleksiyon dizileri döndürmekten kaçının.
- *AvoidDuplicationOfStateInformation:* Bir uygulamanın bir parçasının durumu tek bir sınıfta temsil edilmeli ve diğer sınıflar tarafından yalnızca buradan alınmalı, istemci sınıflarının yerel alanlarında durum bilgisi kalıcı olarak çoğaltılmamalıdır.
- *CheckMethodUsageDynamically:* Çalışma zamanında, bir nesnenin farklı yaşam döngüsü aşamalarında kullanılan yöntem grupları olup olmadığını kontrol edin.
- *AvoidUnbalancedInheritanceAndDelegationHierarchies:* Karmaşık uygulama ayrıntılarını, delegasyon kullanılarak entegre edilen ayrı sınıflara ayırın. Ayrıca, delege edilen sınıflarda karmaşık miras yapıları kullanmaktan kaçının.
- *AvoidUnrelatedFields:* Bir sınıf, ilişkili olmayan semantiğe sahip alanlara sahip olmamalıdır.
- *AvoidStaticMethods:* Bir sınıf, tekil desenin uygulanması gibi özel durumlar dışında statik yöntemler tanımlamamalıdır.
- *Tutarlı Olmayan Arayüzlerden Kaçının:* Bir arayüz, birbiriyle ilişkili olmayan arayüz yöntemleri kümelerine sahip olmamalıdır. İlişkili olmak, bir arayüzün varsayılan yöntemler sağlamadan uygulamaları tarafından tamamen uygulandığı anlamına gelir.
- *AvoidLargeObjects:* Miras, çok sayıda alana sahip nesnelerle sonuçlandığında (miras ağacındaki tüm sınıflar tarafından toplanan) mirastan kaçının.
- *UseCoherentNaming:* Bir pakette birbirine ait sınıf grupları tutarlı bir şekilde adlandırılmalıdır (yani, sınıf adının sabit bir kısmıyla bu gruba ait olduklarını belirtmelidir).
- *AvoidSharedClassesInSubPackages:* Farklı paketlerdeki sınıflar tarafından kullanılan ve genel API olmayan paylaşılan bir sınıf, onu kullanan sınıflarla (istemciler) ilgili bir üst pakette bulunmalıdır.
- *AvoidSimplyDependenciesAcrossMultiplePackages:* Yalnızca başka bir sınıfa bağımlı olan bir sınıf, bağımlı sınıfın alt paketinde bulunmalıdır.
- *AvoidReturningContainerObjectsFromCommands:* Bir nesnenin durumu bir komutla değiştirilirse, döndürülen veriler mümkün olan en ince ayrıntı düzeyinde olmalıdır (örneğin, bir listedeki yalnızca bir öğe değiştirilirse, listenin tamamı döndürülmemelidir).
- *AvoidMutableFieldsWhenPossible:* Bir alan asla değiştirilmeyecekse, özel ve nihai olmalıdır.
- Nesne *Durumlarını Taklit Etmekten Kaçının:* Özelliklerini kullanan bir nesnenin durumu asla taklit edilmemelidir (örneğin, bir sorgunun döndürülen türü gerçek nesne durumunu temsil etmelidir).
- *UseMeaningfulVariableNames:* Değişken adları anlamlı olmalıdır.
- *CheckPrivateMethodUsage:* Özel bir yöntem sınıf dışında yeniden kullanılamaz ve yinelemelere neden olabilir.
- *CheckGroupingOfUtilityClasses:* Yardımcı sınıflar, ideal olarak aynı pakette gruplandırılmalıdır.
- *AvoidSameInformationInDifferentArtifacts:* Tasarım sürecinde alınan kararlar, diğer artefaktlarda ve diğer düzeylerde tekrar tekrar açıklanmamalıdır; örneğin, mimaride açıklanan bir bileşenin bölünme nedeni, bileşenin tasarımında tekrar edilmemelidir.
- *AvoidDeadCode:* Bir yöntemde kullanılmayan kod kaldırılmalıdır.
- *AvoidNonCohesivePackageImplementations:* Bir paket mümkün olduğunca tutarlı olmalıdır (yani paketler bağımsız sınıf grupları içermemelidir).
- *UseStrictLayering:* Kütüphaneleri katmanlara ayırarak, bu kütüphanelerin kesiştiği noktaları tanımlayabilirsiniz. Düşük seviyeli katmanlar, üst seviyeli katmanların işlevlerini kullanmamalıdır.
- *KeepInterfaceImplementationsTogether:* Aynı arayüzü paylaşan sınıfları aynı pakette gruplandırın.
- *AvoidReferencingImplementationPackages:* Somut (uygulama) paketleri arasındaki bağımlılıklardan kaçının, ancak soyut paketleri kullanın.

paketleri kullanın. Soyut paketler içinde, yalnızca arayüzleri ve üst sınıfları yerleştirmek izin verilir.

- *AvoidPropertyInjection:* Bir sınıf özelliği, sınıf dışınd 'e enjekte edilmemelidir.
- *UseInterfaceForExternalPackageDependencies:* Paket dışındaki sınıflar tarafından erişimi için arayüzler kullanın. Aynı paket veya kitaplıkta bulunacak sınıfları için soyut sınıflar kullanın.
- *CheckUnrelatedMethods:* Bir yöntem sınıfıyla ilişkili değilse, başka bir sınıfta olmalıdır.
- *AvoidInterfaceInheritance:* Arayüzlerle miras mekanizmasının aşırı kullanımından kaçının, çünkü bu, spesifik olmayan ve geniş arayüzlere yol açabilir.

Referanslar

Abdeen, H., Sahraoui, H., Shata, O., 2013. Arayüzleri nasıl tasarlıyoruz ve nasıl erişiyoruz. 29. IEEE Uluslararası Yazılım Bakımı Konferansı (ICSM). IEEE, Eindhoven, Hollanda, s. 80–89. doi:10.1109/ICSM.2013.19.

Adler, M., Ziglio, E., 1996. Gazing Into the Oracle: The Delphi Method and Its Ap plication to Social Policy and Public Health. Jessica Kingsley Publishers, Londra, İngiltere.

Briand, L.C., Daly, J.W., Wust, J.K., 1999. Nesne yönelimli sistemlerde ölçümleri birleştirmek için birleşik bir çerçeve. IEEE Trans. Softw. Eng. 25 (1), 91–121. doi:10.1109/32.748920.

Briand, L.C., Wüst, J., Lounis, H., 2001. Nesne yönelimli tasarımlarda kalite faktörlerini araştırmak için tekrarlanan vaka çalışmaları. Empirical Softw. Eng. 6 (1), 11–58. doi:10.1023/A:1009815306478.

Bräuer, J., Plösch, R., Saft, M., Körner, C., 2017. Nesne yönelimli tasarımın kalitesini iyileştirme: portföy ve ölçüm tabanlı bir yaklaşım. İçinde: 27. Uluslararası Yazılım Ölçümü Çalıştayı ve 12. Uluslararası Yazılım Süreci ve Ürün Ölçümü Konferansı ( ) Bildirileri. ACM, Göteborg, İsveç, s. 244–254. doi:10.1145/3143434.3143454.

Bräuer, J., Plösch, R., Saft, M., Körner, C., 2017. Nesne yönelimli tasarım en iyi uygulamalarının önemi üzerine bir araştırma. 43. Euromicro Yazılım Mühendisliği ve İleri Uygulamalar Konferansı (SEAA). IEEE, Viyana, Avusturya, s. 27–34. doi:10.1109/SEAA.2017.14.

Charalampidou, S., Ampatzoglou, A., Avgeriou, P., 2014. Gömülü sistem mühendisliği için bir süreç çerçevesi. 40. Euromicro Yazılım Mühendisliği ve İleri Uygulamalar Konferansı (SEAA). Verona, İtalya, s. 137–140. doi:10. 1109/SEAA.2014.58.

Chidamber, S.R., Kemerer, C.F., 1994. Nesne yönelimli tasarım için bir metrikler dizisi. IEEE Trans. Softw. Eng. 20 (6), 476–493. doi:10.1109/32.295895.

Churcher, N., Frater, S., Huynh, C.P., Irwin, W., 2007. OO tasarım sezgilerini destekleme. İçinde: 18. Avustralya Yazılım Mühendisliği Konferansı (ASWEC). IEEE, Melbourne, Avustralya, s. 101–110. doi:10.1109/ASWEC.2007.47.

Coad, P., Yourdon, E., 1991. Nesneye Yönelik Tasarım. Prentice Hall, Londra, Birleşik Krallık.

Dooley, J., 2011. Nesneye yönelik tasarım ilkeleri. İçinde: Yazılım Geliştirme ve Profesyonel Uygulama. Apress, Berkeley, CA, ABD, s. 115–136. doi:10.1007/ 978- 1-4302- 3802- 7_10.

Edmunds, H., 2000. Odak Grup Araştırması El Kitabı, 1 McGraw-Hill, Lincolnwood, Ill.; St. Albans.

Hunt, A., Thomas, D., 1999. Pragmatik Programcı: Çıraktan Ustaya. Addison-Wesley, Boston, MA, ABD.

Kontio, J., Bragge, J., Lehtola, L., 2008. Yazılım mühendisliğinde ampirik bir araç olarak odak grup yöntemi. Shull, F., Singer, J., Sjøberg, D.I.K. (Eds.), İleri Düzey Ampirik Yazılım Mühendisliği Kılavuzu. Springer London, s. 93–116. doi:10. 1007/978- 1- 84800-044- 5_4.

Linstone, H.A., Turoff, M., 1975. Delphi Yöntemi: Teknikler ve Uygulamalar. Addison-Wesley Pub. Co., İleri Düzey Kitap Programı.

Marinescu, R., 2004. Algılama stratejileri: tasarım kusurlarını algılamak için metrik tabanlı kurallar. İçinde: 20. IEEE Uluslararası Yazılım Bakımı Konferansı (ISCM). IEEE, Chicago, IL, ABD, s. 350–359. doi:10.1109/ICSM.2004.1357820.

Martin, R.C., 1996. Granülerlik. C++ Raporu 8 (10), 57–62.

Martin, R.C., 2003. Çevik Yazılım Geliştirme: İlkeler, Kalıplar ve Uygulamalar. Pearson Education, Upper Saddle River, NJ, ABD.

Martin, R.C., 2008. Temiz Kod: Çevik Yazılım Ustalığı El Kitabı, 1 Prentice Hall, Upper Saddle River, NJ.

Meyer, B., 1997. Nesneye Yönelik Yazılım Yapımı, 2. baskı Prentice Hall PTR, Upper Saddle River, N.J.

Morgan, D.L., 1997. Nitel Araştırma Olarak Odak Grupları, 16. SAGE Yayınları. Okoli, C., Pawlowski, S.D., 2004. Bir araştırma aracı olarak Delphi yöntemi: bir örnek, tasarım hususları ve uygulamaları. Inf. Manage. 42 (1), 15–29. doi:10.1016/ j.im.2003.11.002.

Plösch, R., Bröuer, J., Körner, C., Saft, M., 2016a. Nesne yönelimli tasarım ilkelerine dayalı olarak yazılım kalitesinin ölçülmesi, değerlendirilmesi ve iyileştirilmesi. Open Comput. Sci. 6 (1), 187–207. doi:10.1515/comp-2016-0016.

Plösch, R., Bräuer, J., Körner, C., Saft, M., 2016b. MUSE - Nesne yönelimli tasarımı ölçmek için çerçeve. J. Object Technol. 15 (4), 2:1–29. doi:10.5381/jot.2016. 15.4.a2.

Riel, A.J., 1996. Nesneye Yönelik Tasarım Sezgisel Yöntemleri, 1. Addison-Wesley, Reading, MA, ABD.

Samarthyam, G., Suryanarayana, G., Sharma, T., Gupta, S., 2013. MIDAS: endüstriyel yazılımlar için tasarım kalitesi değerlendirme yöntemi. 35. Uluslararası Yazılım Mühendisliği Konferansı (ICSE). IEEE, San Francisco, CA, ABD, s. 911–920. doi:10.1109/ICSE.2013.6606640.

Schmidt, R.C., 1997. Parametrik olmayan istatistiksel teknikler kullanarak Delphi anketlerini yönetme. Decis. Sci. 28 (3), 763–774. doi:10.1111/j.1540-5915.1997.tb01330.x.

Sharma, T., Samarthyam, G., Suryanarayana, G., 2015. Tasarım ilkelerinin uygulamada kullanılması. 8. Hindistan Yazılım Mühendisliği Konferansı (ISEC). ACM, Bangalore, Hindistan, s. 200–201. doi:10.1145/2723742.2723764.

Srivastava, S., Kumar, R., 2013. CK-OO paketi kullanarak yazılım kalitesini ölçmek için dolaylı yöntem. 2013 Uluslararası Akıllı Sistemler ve Sinyal İşleme Konferansı (ISSP). IEEE, s. 47–51. doi:10.1109/ISSP.2013.6526872.

Stevenson, J., Wood, M., 2017. Nesne yönelimli yazılım tasarım kalitesini tanıma: uygulayıcı temelli anket araştırması. Softw. Qual. J. 1–45. doi:10.1007/ s11219-017-9364- 8.

Subramanyam, R., Krishnan, M.S., 2003. Nesne yönelimli tasarım karmaşıklığı için CK metriklerinin ampirik analizi: yazılım hataları için çıkarımlar. IEEE Trans. Softw. Eng. 29 (4), 297–310. doi:10.1109/TSE.2003.1191795.

Turney, L., Pocknee, C., 2005. Sanal odak grupları: araştırmada yeni ufuklar. Int. J. Qual. Methods 4 (2), 32–43. doi:10.1177/160940690500400203.

Wieringa, R., Moralı, A., 2012. Bilgi sistemleri tasarım biliminde bir doğrulama yöntemi olarak teknik eylem araştırması. İçinde: 7. Uluslararası Bilgi Sistemlerinde Tasarım Bilimi Araştırmaları Konferansı: Teori ve Uygulamada Gelişmeler (DESRIST). Springer-Verlag, Berlin, Heidelberg, Las Vegas, NV, ABD, s. 220–238. doi:10.1007/978- 3- 642-29863-9_17.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2012. Yazılım Mühendisliğinde Deneyler. Springer Berlin Heidelberg doi:10.1007/ 978- 3- 642-29044- 2.

Johannes Bräuer, Johannes Kepler Üniversitesi Linz'de İşletme Bilişimi - Yazılım Mühendisliği Bölümü'nde doktora derecesini almıştır. Tezi, temel tasarım ilkelerine dayalı yazılım tasarımının ölçülmesi ve değerlendirilmesine odaklanmaktadır. Ayrıca, araştırma alanları yazılım kodu kalitesi ve teknik borç değerlendirme yaklaşımlarıdır.

Reinhold Pläsch, Johannes Kepler Üniversitesi Linz İşletme Bilişimi - Yazılım Mühendisliği Bölümü'nde yazılım mühendisliği doçenti olarak görev yapmaktadır. Temel kod kalitesinden gömülü ve güvenlik açısından kritik sistemlerin kalitesine kadar kaynak kod kalitesiyle ilgilenmektedir. Ayrıca tasarım ilkelerine dayalı olarak nesne yönelimli tasarım kalitesini otomatik olarak ölçmekle de ilgilenmektedir.

Matthias Saft, Siemens Corporate Technology'de yazılım geliştirme ile ilgili konularda çalışmaktadır. Odak noktası kod ve tasarım kalitesi, bunların ölçümü, görselleştirilmesi ve iyileştirilmesidir. Buna uygun bir mimari temel zorunludur ve aynı şekilde dikkate alınmaktadır. Ayrıca, büyük ölçekli yalın ve çevik geliştirme metodolojileri ve bunların endüstriyel bağlamda uygulanmasıyla da ilgilenmektedir.

Christian Körner, Münih'teki Siemens Corporate Technology'de *Kıdemli Anahtar Mühendis* olarak görev yapmaktadır. Mesleki ilgi alanları, *Geliştirme Verimliliği* için teknik ve yönetim yöntemleri üzerinedir. Son yıllarda projeler, geliştirme organizasyonları için artefakt tabanlı değerlendirme yöntemleri geliştirmek ve uygulamak ile yazılım (tasarım) kalitesinin otomatik olarak değerlendirilmesine odaklanmıştır. Projeler, küçük proje müdahalelerinden uluslararası ortaklarla yapılan büyük araştırma işbirliklerine kadar çeşitlilik göstermektedir.