5. Hafta
2. Grup
Başlangıç

10.6. Veriye Yönelik Tasarım

Veriye yönelik ya da diğer adıyla veri yapılarına yönelik tasarımda [5], veriyle ilgili işlemler fazla dikkate alınmadığı için veri akış diyagramı kullanılmasına gerek yoktur; onun yerine veri yapılarının ve yordamların tanımlamaları yapılır. Tasarım, veri akışından çok bilgi yapısına dayandırılır. Bu yöntem daha çok, giriş ve çıkışın dosyalar şeklinde olduğu iş dünyası uygulamalarında, veri yapılarının yoğun bir şekilde kullanıldığı işletim sistemi geliştirmede, endüstriyel üretime ilişkin veri yapılarının üretildiği bilgisayar destekli tasarım ve üretim sistemlerinde kullanılır.

Tasarım sırasında genellikle aşağıdaki işler gerçekleştirilir:

- Veri yapılarının özellikleri değerlendirilir.
- Verilerin saklama, arama ve erişim yöntemleri, veri yapılarına ait temel gösterim şekilleriyle tanımlanır.
- Veri yapıları yazılımın denetim sıradüzeni içindeki yerine göre uyarlanır. Gerekirse yazılım sıradüzeni veri yapıları dikkate alınarak yalınlaştırılır.
- Bundan sonraki yazılımın genel tasarımı yordamsal tanımlama şeklinde yapılır.

Veri yapılarına yönelik sistem geliştirme için çözümlemenin nasıl yapıldığına daha önce değinmiştik. Bu yöntemde çözümleme ile tasarım hemen hemen birlikte yapılır ve çabuk bir şekilde gerçekleştirime geçilir.

10.7. Tasarım Kalıpları

Yazılım mühendisliğinde en yaygın ve etkin yöntemlerden biri olan nesneye yönelik çözümleme ve tasarım yeni bir konunun doğmasına neden olmuştur: *Tasarım kalıpları* ya da diğer çevirilerle *tasarım örüntüleri* veya *tasarım desenleri*. Tasarım kalıpları, elde edilen deneyimlerin, çıkarılan derslerin ve en iyi sonuçların belgelendirilmesiyle oluşmuş modern mimarilerle problem çözme disiplinleridir. Aslında, teknolojiye odaklanmak yerine, güvenilir ve sağlıklı bir mimariye dayalı tasarım yapılarak belgelendirilmesi için bir kültür oluşturma ilkesine dayanır.

10.7.1. Kalıp Tanımı ve Özellikleri

Tasarım kalıpları çözümleyicinin, tasarımcının ve kodlayıcının aynı dili konuşmalarını sağlayarak iletişimi kolaylaştırır. Bir tasarım kalıbı belirli bir bağlam içindeki bir problemi ve onun çözümünü kapsayan bir kuraldır. Genelde yazılım geliştirme sırasında karşılaşılan tekrarlanan problemlere tekrar kullanılabilir çözümler üretmeyi amaçlar. Kalıpların algoritmalar gibi belirli kuralları vardır. Ancak, algoritmalar ve bunlara bağlı olarak veri yapıları genellikle sıralama, arama gibi küçük çaplı bilgi işleme problemlerini çözmede kullanılırlar. Kalıplar ise daha geniş etkileri olan ve mimariye yönelik konularla ilgilenir ve bu düzeydeki problemlere çözüm getirir. Öte yandan, çatılar (frameworks) genellikle yürütülebilir kod şeklinde olup bir ya da birden fazla kalıp çözümlerinin gerçekleştirilmesinden oluşur. Kalıpların ortak özellikleri arasında şunlar vardır:

- Deneyimlerle ortaya çıkmışlardır.
- Daha büyük problemleri çözmek için beraberce kullanılabilirler.
- Çeşitli soyutlama düzeylerinde bulunurlar.
- Yapısal bir biçimde yazılırlar.
- Tekrar kullanılabilir öğelerdir.
- En iyi uygulamalar ile tasarımlar arasında bir tür iletişim sağlarlar.

Yine de her çözüm, algoritma, pratik uygulama veya deneyim mutlaka bir kalıp oluşturmaz. Gerçekten bir kalıp olabilmesi için hiç değilse üç farklı sistemde tekrarlanan bir probleme çözüm olması gereklidir. İyi bir kalıbın özellikleri arasında da şunları sayabiliriz:

- Yalnızca bir takım ilke ve stratejilerden oluşmaz, belirli bir probleme belirli bir çözüm sağlar.
- Çözümleri kayıt altına alarak herhangi bir şüpheye yer bırakmaz.
- Kalıp yalnızca modülleri değil, daha alt düzeydeki yapı ve düzenekleri de tanımlar.
- İnsanlara kullanım kolaylığı ve yarar sağlar.
- Gerçek dünyadaki bir probleme uyarlanabilecek şekilde kullanışlıdır.
- Kullanılmıştır ve bu şekilde belgelendirilmiştir.

Bir kalıp çeşitli öğelerden oluşmaktadır. Genel olarak, bir kalıpta bulunması gerekli öğeler şunlardır:

- **İsim**
  Kalıbı tanımlamak üzere temsil ettiği bilgiyi ve yapıyı anlatan anlamlı bir isim kullanılmalıdır. İyi seçilmiş isimlerle bir kalıp sözlüğü oluşturularak geniş bir uygulama alanının kapsanması sağlanabilir.
- **Problem**
  Problem, kalıbın bağlam içindeki kullanım amacını belirleyen bir açıklamadır.
- **Bağlam**
  Problemin oluştuğu ve çözümün arzu edildiği ön koşullardır. Aynı zamanda kalıbın uygulanabilirliğini gösterir.
- **Kuvvetler**
  Amaçlanan hedefle veya birbirleriyle etkileşen ya da çelişen her türlü kuvveti ve kısıtlamaları anlatır. Problemin karmaşıklık derecesini ve uygulamada oluşabilecek istenmeyen durumları gösterir.
- **Çözüm**
  Arzu edilen getirinin nasıl elde edileceğini tanımlayan kurallardır.
- **Örnekler**
  Kullanıcının kalıbın kullanımını ve uygulanabilirliğini daha iyi anlamasına yardım eden görsel ve metinsel anlatımlardır.

- **Sonuç bağlamı**
  Kalıbın uygulanmasından sonra oluşabilecek durumları, son koşulları ve yan etkilerini tanımlar.
- **Mantık açıklaması**
  Kalıp içindeki basamakların ve kuralların doğru olduğunun açıklanmasıdır.
- **İlgili kalıplar**
  Aynı bağlam ya da sistem içindeki kalıplarla bu kalıp arasındaki ilişkiler açıklanır.
- **Bilinen kullanımlar**
  Daha önce kullanıldığı bilinen ve halen var olan uygulamalarda bu kalıbın kullanım şekli anlatılır.

6.7.2. Karşı Kalıplar

Bir kalıp en iyi uygulamayı ya da pratiği gösterirken bir *karşı kalıp* (anti-pattern) çıkarılan bir dersi gösterir. Karşı kalıpların iki şekli vardır:

- Kötü sonuçlanan bir problem çözümünde kullanılan kötü bir yöntemi tanımlayanlar (kötü örnek)
- Kötü bir durumdan kurtulup iyi bir duruma geçmeyi tanımlayanlar

Karşı kalıplar, çeşitli basılı yayınlarla kullanıcıların bilgisine sunulur. Sürekli gelişen bir bilgi topluluğu olduğu için henüz belirli bir standart içinde değildir.

6.7.3. Kalıp Örnekleri

Kalıp örneklerini geliştirme yapan büyük deneyim sahibi firma, kurum ve kuruluşlarda bulabilmek mümkündür. Bu konuda halen yayınlanmakta olan birçok kaynak da vardır. Biz şimdi yaygın olarak kullanılan tasarım kalıplarından bazı örnekleri kısa birer özet halinde verelim:

- **Önyüz (façade)**
  Önyüz kalıbı, bir nesne grubuna arayüz olacak şekilde dışarıdaki nesnelerin erişebileceği bir tek nesne sağlayarak dışarıdakilerin grupla haberleşmesini kolaylaştırır. Genelde grup içinde yer alan nesneler arasındaki bağlılık kullanıcılar için karmaşıklık getirir. Oysa kullanıcı karmaşıklığının azaltılması istenir. Bu amaçla bir önyüz nesnesi kullanılarak grup ile kullanıcı arasında soyutlama sağlanır. Kullanıcıların bu nesnenin arkasındaki ayrıntıları bilmelerine gerek yoktur.
- **Teklik (singleton)**
  Bu kalıpta bir sınıfın yalnızca bir tek yaratımı (instance) vardır. Bu sınıfın yaratımlarını kullanan diğer nesnelerin tümü aynı yaratımı kullanırlar. Genel olarak özkaynağın merkezden yönetilmesi amacı vardır. Yönetilen özkaynak dışsal bir veritabanı bağlantısı olabileceği gibi, durağan bir değişkende tutulan

içsel bir sayaç da olabilir. Sınıfın tek yaratımı yürütme başladığı anda belleğe yüklenir. Bu tek yaratıma erişmek için sınıfın durağan bir işlemi bulunur.

- **Fabrika (factory)**
  Fabrika kalıbı, kendisine aktarılan veriye bağlı olarak olası sınıflardan birinin yaratımını döndürür. Genellikle, döndürülebilen tüm sınıfların ortak bir temel sınıfı ve ortak işlemleri vardır; fakat herbiri başka tür verileri işleyerek başka işler yaparlar. Herbirinin aynı işlemi bulunduğundan kullanıcı açısından hangi sınıfın döndürüldüğü fark etmez, ancak bu sınıf işlemlerinin gerçekleştirimleri farklıdır. Hangi sınıfın nasıl ve neye göre döndürüleceği tamamen fabrikaya bağlıdır. Kullanıcı sınıf, döndürülen sınıflardan tamamen farklıdır.
- **Bileşik (composite)**
  Bileşik kalıpla ağaç yapısında, birbirine benzer nesneleri özyineli (recursive) biçimde oluşturarak karmaşık nesneler yaratmak mümkündür. Hatta, tüm nesnelerin ortak bir süper sınıfı olduğu için ağaç yapısındaki nesneler üzerinde daha tutarlı şekilde işlem yapmak oldukça kolaylaşır. Bir belgeyi oluşturan karakterlerin, satırların, metin sütunlarının ve sayfaların gösterimi buna örnektir. Bunları içeren bir bileşik belge nesnesi süper sınıf olarak kullanılarak alttaki nesnelerin yönetimi sağlanır. Bu şekilde karmaşıklık alt parçalara bölünerek azaltılmış olur. Alttaki nesneler ise basit veya bileşik olabilirler.

10.8. Kullanıcı Arayüzü Tasarımı

Bilgisayar tabanlı sistemler insanların işlerini kolaylaştırmak üzere geliştirilirler. Ne kadar mükemmel tasarlanmış ve gerçekleştirilmiş olursa olsun, eğer bir sistem kullanıcısına zor anlar yaşatıyorsa tam başarılı sayılamaz. Bu nedenle, sistemin insanlarla olan arayüzünün çok etkin ve kullanışlı olarak, verimliliği artıracak şekilde, kullanıcı dostu olarak tasarlanması gereklidir. Bilgisayar tabanlı sistem arayüzünün önemi çok geniş uygulama alanları düşünüldüğünde daha iyi ortaya çıkar. Bir programlanabilir fırın veya çamaşır makinesi, bir cep telefonu, bir tıbbi cihaz, bir uçuş kontrol altsistemi gibi gömülü sistemlerin arayüzleri, bir veritabanı işletim sisteminin arayüzü, hele hele günümüzün en yaygın bilgisayar uygulamalarından biri olan İnternet tabanlı programların arayüzleri insan mühendisliğinin ne kadar önemli olduğunu çok güzel ortaya koyan örnekler arasındadır. Bir sistemin arayüzünü öğrenmek ve etkin bir şekilde kullanmak ne kadar kolay olursa o sistemin yetenek ve işlevlerinden yarar sağlamak da o kadar artar.

10.8.1. İnsan-Bilgisayar Etkileşimi

İnsan ile bilgisayar arasındaki etkileşim genellikle görsel, işitsel ve dokunmatik olarak gerçekleşir. Ekrandaki bilgiler görsel iletişimi sağlarken, mikrofon ve hoparlör ile ses bilgileri alınır veya verilir. Bunlar yanında, girdi sağlamak için klavye, işaret aygıtı (mouse veya trackball), oyun çubuğu (joy-stick), çeşitli özel tuşlar ve giriş aygıtları kullanılır. Tüm bunlara *insan-makine arayüzü* (Human-Machine Interface - HMI) adı

verilmektedir (bazen Man-Machine Interface - MMI kısaltması da kullanılır). Etkileşimin daha çok bilgisayarla yapıldığı durumlarda bu isim *insan-bilgisayar arayüzü* (Human-Computer Interface - HCI) adını almaktadır. Bu şekilde, bir insan, bilgisayardan çeşitli bilgileri alabilir, kendi belleğinde saklayabilir, neden-sonuç ilişkilerini değerlendirebilir ve kendisi yeni bilgiler girebilir [3].

Günümüzde, insan-bilgisayar arasındaki arayüzü daha da iyi duruma getirebilmek için bilimsel ve teknik çalışmalar sürdürülmektedir. Örneğin, bir zamanlar siyah-beyaz olan ekranlar sonradan, 16 renk, 256 renk ve en sonunda da milyonlarca rengi destekleyerek gerçeğe en yakın görünüşü vermeye başlamışlardır. Aynı zamanda ekranların çözünürlüğü, canlılığı ve büyüklüğü artmış, insan gözünü yormayacak (en az 72 Hz yatay tarama standardı gibi) ve sağlığı tehdit etmeyecek (düşük radyasyon yayımı) hale gelmişlerdir.

Bilgisayar tarafından ekranda sergilenen bilgiler renk, şekil, büyüklük ve hareket bakımından göz ile algılanıp beyin tarafından değerlendirilerek insan tarafından algılanmaları sağlanmaktadır. Gittikçe daha rahat yani ergonomik yapıya ulaştırılan klavye ve işaretçi aygıtlarla bilgisayar ile iletişim kurmak, komutlar vermek, bilgileri seçmek daha da kolay olabilmektedir. Gelecekte sesli komutlarla da anlaşabilmek tabii ki mümkün olabilecektir. Tüm giriş/çıkış aygıtlarının iyi bir uyum içinde kullanılmasıyla insanlarla makine arasındaki büyük engeller yavaş yavaş aşılmaktadır. Belki de gün gelecek düşüncelerimizle iletişim kuracağız ve bilgisayarları aklımızın bir uzantısı olarak kullanacağız.

Tüm bilgisayarlı sistem kullanıcılarının aynı anlayış derecesine sahip olduklarını kabul etmek yanlış olur. Kimi kullanıcılar bilgisayarlara alışkın olduklarından hızlı bir şekilde öğrenip etkin olarak kullanabilirken, kimileri bir bilgisayarı ancak açıp yavaş bir şekilde kullanabilmektedir. Bir bilgisayar mühendisine çok uygun gelen bir arayüz, deneyimsiz bir işçi için hiç de uygun olmayabilir. Bu düşünceyle, kullanıcıları üç sınıfa ayırmak doğru olur:

- *Deneyimsiz:* Sistem hakkında hiç bilgisi olmayan ancak bir miktar bilgisayar kullanabilen kişilerdir.
- *Az deneyimli:* Sistem hakkında bir miktar bilgisi olan ancak az miktarda kullanım pratiğine sahip kişilerdir.
- *Çok deneyimli:* Hem sistemi hem de kullanımını iyi bilen, hatta daha hızlı kullanabilmek için kestirmeler arayan kişilerdir.

Dolayısıyla, insan bilgisayar arayüzü, hedef kullanıcının yaş, fiziksel yeterlilik, eğitim ve kültürel durum, dil, kişisel güdü ve amaçlarıyla konu hakkındaki bilgi ve becerisi, hatta teknolojik korkusu dikkate alınarak tasarlanmalıdır. Örneğin, ekran çıktısında bulunan bilgi miktarı karmaşıklığa yol açmayacak kadar az ve öz olmalı, kısaltmalar herkes tarafından anlaşılabilir olmalı, basılacak tuşların isimleri ve sırası kolay takip edilebilmeli, yanlış sırada kullanımları engellemek üzere giriş seçenekleri kısıtlanmalıdır (istenen bir bilgi girilmezse "Tamam" tuşunun geçerli hale gelmemesi gibi).

Arayüz herhangi bir anda sistemin içinde bulunduğu durum, yaptığı iş, beklediği veri girişi gibi bilgileri sergilemeli ve uyarılarda bulunabilmelidir. Kullanıcıların olduğu gibi bilgisayarların da bir kişiliği vardır. Bu nedenle ideal bir arayüz, kullanıcıların değişik kişiliklerini yansıtabilmelerini için ayarlanabilir olmalıdır.

6.8.2. İnsan-Bilgisayar Arayüz Tasarımı

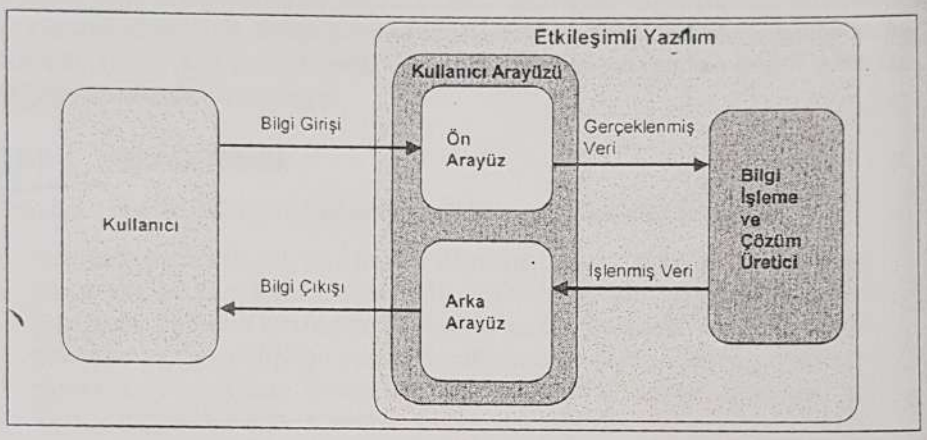
Grafiksel sergileme ortaya çıkmadan önce bilgisayarlarla etkileşim yalnızca klavye ile komut satırından yapılmaktaydı. Öğrenmesi oldukça zor ve hata yapma olasılığı çok yüksek olan komut satırı ile işlem yapmak artık geride kalmış sayılır. Ancak, Unix tabanlı işletim sistemleri gibi yaygın bazı sistemler hala komut satırı kullanmakta, büyük bir kullanıcı kitlesi de bu komutlarla daha etkin kullanım sağlayabilmektedir. Fakat uygulama yazılımları çoğunlukla grafiksel kullanıcı arayüzlerine sahip olacak şekilde geliştirilmektedir.

Günümüz standartlarında bir sistem için kullanıcıyla doğrudan etkileşimi sağlayan en önemli yazılım öğesi grafik tabanlı pencerelerdir. Bir pencere yönetim sistemi ile denetlenen çeşitli grafiksel birimler ile kullanıcının gözüne, anlayışına en uygun bilgi sergileme, en çabuk ve en doğru şekilde bilgi girme olanağı sağlanır. Standart bir pencere işleticisi çalıştırabilen, yeterli grafik özelliklere sahip bilgisayarlar üzerinde grafiksel arayüzler insanlara çok daha yakın olabilmektedir. Üzerinde resim veya simge bulunan küçük şekiller (ikonlar), renkli çubuklar, metin veya veri giriş alanları, bilgi sergileme alanları, seçme menüleri, tıklama veya seçme tuşları gibi grafik öğeleri ile bir arayüz tasarımı artık hem bilim hem de sanat içerir hale gelmiştir.

Günümüzdeki bilgisayar işletim sistemlerinden bir kısmı tamamen pencere sistemine dayalı olarak çalışmakta iken bir kısmı da ek bir pencere yönetim sisteminin (Linux ile X-Window Sistemi, vb.) çalıştırılmasını gerektirir. Pencere yönetim sistemleri özel olarak geliştirilebileceği gibi yaygın standartlardan biri de seçilebilir. Ancak dikkat edilmesi gereken nokta, seçilen standarda uygun olarak yazılım geliştirme olanağı sağlayan *yazılım geliştirme paketleri*nin (Software Development Kit) kullanılmasıdır. Zira, kendi içlerinde standart olsalar da birbirleri arasında uyum olmayan pek çok pencere yönetim sistemi bulunmaktadır.

Herhangi bir etkileşimli yazılım, Şekil-6.13 te görüldüğü gibi iki öğeden oluşur. Burada, kullanıcı, yazılıma girdi sağlayan ve çıktıları değerlendiren işletmendir. Bilgi işleme ve çözüm üretici, kendisine ulaştırılan gerçeklenmiş bilgilerin işlendiği ve eldeki problemin çözülmesi için gereken bilgilerin üretildiği öğedir.

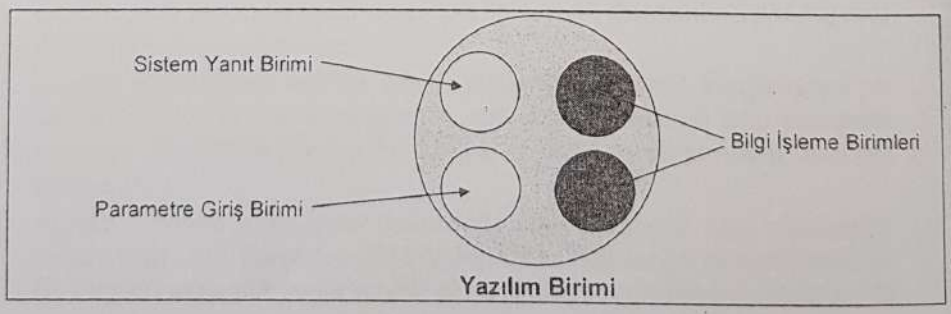
Kullanıcı arayüzü, etkileşimli yazılım sistemlerinde, kullanıcı ile bilgi alışverişini sağlayan öğedir. Arayüz, kendi içinde, ön ve arka arayüz olmak üzere ikiye ayrılır. Ön arayüz, kullanıcının yazılıma bilgi girmesini sağlayan kısımdır. Örneğin, gün, ay, yıl olarak girilen tarih bilgisini sayısal hale getirerek çözüm üreticisine ulaştırır. Arka arayüz, çözüm üreticisi tarafından üretilen işlenmiş veriyi kullanıcının görmek ya da almak istediği biçime çeviren kısımdır. Sistemin ürettiği kayan nokta tipindeki zaman bilgisi arka arayüz tarafından saat, dakika ve saniye gösterimine çevrilerek sunulur.



Şekil-6.13. Etkileşimli bir yazılımın öğeleri.

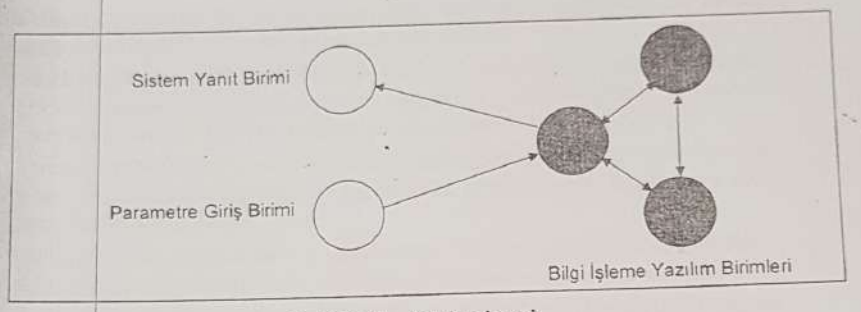
6.8.3. Arayüz Yazılım Mimarisi

Bir uygulama yazılımı tek bir yürütülebilir yazılım biriminden, yani programdan oluşabileceği gibi birden fazla birimden de oluşabilir. Pencere sistemi ile uyum sağlayan arayüz yazılımı ya ayrı bir birim halinde geliştirilir ya da aynı birimin içine gömülür. Tek birimden oluşan yazılımlarda, bilgi işleme birimleri ile arayüz birimleri aynı bağlam (context) içinde yer alırlar. Şekil-6.14 te gösterilen ve birleşik mimari olarak adlandırılan bu yapı, tek işlemciye sahip bilgisayarlar için uygundur:



Şekil-6.14. Birleşik mimari.

Birden fazla işlemcisi olan bilgisayarlarda veya tamamen dağıtık mimaride çalışabilmek üzere Şekil-6.15 te gösterilen ayrık mimari kullanılır. Bu mimarinin yararı, bilgi işleme ile arayüz birimlerini birbirlerinden ayırarak hem geliştirme hem de yürütme bağımsızlığı sağlanır. Özellikle yüksek başarım gerektiren uygulamalarda, bilgi işleme yeteneğinin arayüz ile kısıtlanmaması arzu edilir. Yeterli iletişim altyapısı bulunduğu takdirde, bilgi işleme ile arayüz yazılımları farklı işlemcilerde, hatta farklı özelliklere sahip bilgisayarlarda çalıştırılarak hem bağımsızlık, hem de verim artışı sağlanabilir.



Şekil-6.15. Ayrık mimari.

6.8.4. Arayüz Yazılım Birimleri

Arayüz donanımları cinsleri ne olursa olsun mutlaka ana sistemle tümleştirilmeleri gerekir. Tümleştirme işlemi için mutlaka birer arayüz donanımı ve onları etkinleştiren birer yazılım birimi kullanılmalıdır. Örneğin, ekranı uygun grafik özellikte sürmek için yeterli güce sahip bir grafik kartı kullanılmalıdır. Bu kartın da uygun bir sürücü yazılımı ile işletim sistemine tanıtılması gereklidir. Bir işaret aygıtı yine özel bir yazılımla bilgisayarın belirli bir kapısını (port) kullanarak bir sürücü ile sisteme tanıtılır. Özel tuş takımları için bir mikrokontrolcü ile seri kapı üzerinden haberleşen bir aygıt sürücüsü kullanılır. Aygıt sürücüler, verdikleri arayüzleri kullanan arayüz yazılım birimleri aracılığıyla uygulama yazılımlarının rahatça giriş/çıkış yapabilmelerini sağlarlar.

Kullanıcı arayüz yazılımlarının bir kısmı kullanıcı tarafından fark edilmez. Ancak en çok fark edileni şüphesiz ki ekran üzerinden çıkış, klavye ve işaretçi ile giriş olanağı sağlayan grafiksel arayüz yazılımlarıdır. Günümüzde pek çok işletim sistemi de grafiksel arayüz vermektedir. Bir kısmı da özel paket yazılımlar yardımıyla grafiksel arayüze sahip olurlar.

Grafiksel arayüz yazılımlarının tasarımına mutlaka özel dikkat gösterilmeli, uygulama alanının gereksinimleri dikkate alınarak en hızlı şekilde kullanıcı ile iletişim kurulması sağlanmalıdır. Örneğin, bir hava trafik kontrol merkezinin bilgisayarlarında radar vidyosu, bölge haritası gösterilmeli, yerdeki ve havadaki uçakların sürekli yenilenen konumları, kimlik ve çağrı adları, rota ve süratleri sergilenebilmelidir.

Sistemin kullanıcı arayüz yazılımları grafik özellikleri daha uygun olan ayrı bilgisayarlar üzerinde çalıştırılabileceği gibi, sistem tasarımına ve yazılım mimarisine bağlı olarak, ana sistemle beraber, aynı bilgisayar üzerinde de çalışabilirler.