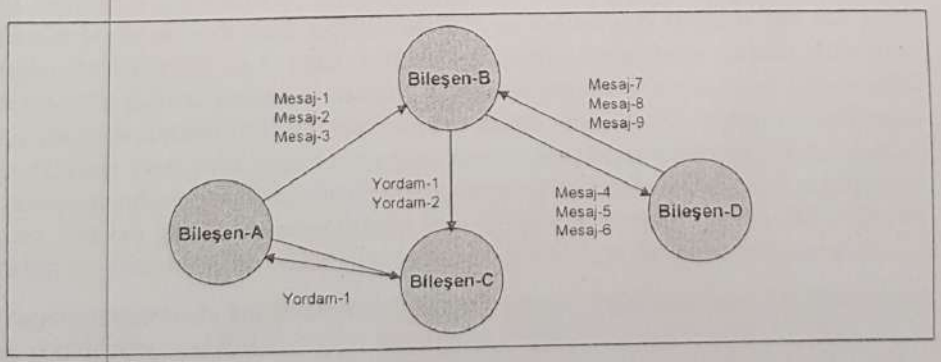
6.2.4. Arayüz Tasarımı

Modüler bir şekilde geliştirilen yazılımın çeşitli arayüzleri bulunur. Bunların bir kısmı içsel arayüzler bir kısmı da dışsal arayüzlerdir. İçsel arayüzler genellikle yazılımın kendi iç öğeleri, bileşenleri ve birimleri arasındadır. Yazılımın dış dünya ile arayüzü ise başka sistemlerle olabileceği gibi etkileşimli sistemler için kullanıcıyla da olabilir. Tasarım, arayüzün bu özelliğine göre değişiklik gösterir.

6.2.4.1. Bileşen Arayüz Tasarımı

Büyük yazılımlar birkaç ana öğeden, herbir öğe birkaç bileşenden ya da birimden oluşabilir. Bu bileşenler arasında mutlaka tanımlı bir arayüz vardır. Bileşenler ya da birimler birer yürütülebilir program olabilecekleri gibi, bir program grubu da olabilirler. Daha ince tanelikli yapılarda ise bileşenler ayrı birer görevcik (thread), hatta birer yordam grubu olabilirler. Bağımsız birer süreç halinde geliştirilen bileşenler Şekil-6.6 da gösterildiği gibi birbirleriyle işletim sisteminin sağladığı çeşitli alt düzey iletişim düzenekleriyle haberleşirler. Örneğin, Unix tabanlı işletim sistemleri için bağlantılar (socket), ileti kuyrukları (message queue), paylaşılır bellek parçaları (shared memory) ve semaforlar kullanılır. Bu düzenekler yardımıyla arayüzü oluşturan iletilerin veya uzaktan yordam çağrılarının (remote procedure call) gerçekleştirimi yapılır.



Şekil-6.6. Bileşen arayüzleri.

Bileşenler arası arayüz tasarlarken dikkat edilmesi gerekenleri şöyle özetleyebiliriz:

- Arayüz anlaşılır yapıya sahip ileti ya da yordamlardan oluşmalıdır.
- İleti tabanlı arayüzlerde başarım için ileti boyutu uygun şekilde ayarlanmalı, çok kısa ve çok uzun iletiler kullanılmamalıdır.

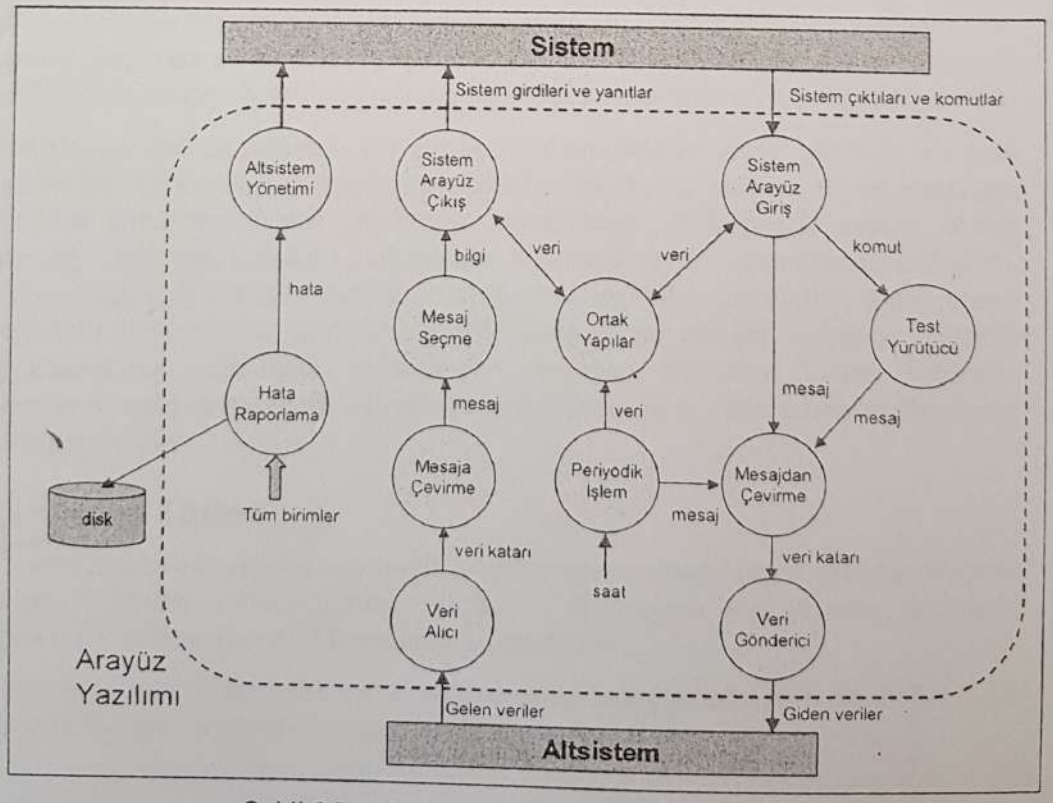
- Büyük miktarda veri aktarımı için ileti yerine ortak bir veri deposu ve aktarılacak verinin adresi kullanılmalıdır.
- Arayüzler belirli bir veri tipine bağımlı olmamalıdır.

6.2.4.2. Kullanıcı Arayüz Yazılımı Tasarımı

Bilgisayar sistemlerinin hemen hemen hepsi insanların denetiminde çalışırlar. Bu nedenle de kullanımı kolay, etkili ve açık bir arayüze sahip olmalıdırlar. Sistemler genellikle standart bir bilgisayarın arayüzü ile denetlenirler. Yani, bir klavye ile girdi sağlanır ve ekranda çıktı görülür. Bunlar dışında, basılı çıktılar, çeşitli göstergeler çıktı için kullanılabileceği gibi çeşitli tuş takımları, işaretçi ve ayar düğmeleri de girdi için kullanılabilir. Önemli olan, uygulama alanında en uygun kullanım olanağını sağlayacak yöntemi bulmak, uygulamak ve en iyi etkileşimi sağlamaktır. Bir insan mühendisliği konusu olan kullanıcı arayüzü tasarımına Ayrıt 6.8 de değinilmektedir.

6.2.4.3. Sistem-Altsistem Arayüz Yazılımı Tasarımı

Çoğu sistem birkaç altsistemi tümleştirerek daha büyük sistemler elde etmek üzere tasarlanır. Tümleştirme için altsistem arayüz yazılımları kullanılır. Bu yazılımlar denetledikleri altsistemlerin gerektirdiği iletişim protokollerini destekleyerek veri alışverişi sağlarlar. Tipik bir arayüz yazılımı iç mimarisi Şekil-6.7 de gösterilmektedir.



Şekil-6.7. Altsistem arayüz yazılımı mimarisi.

Arayüz yazılımları tümleştirilen altsistemin özelliğine göre çok çeşitli yapıda olabilirler. Şekil-6.7 de gösterilen yazılımın modüllerinin herbiri ayrı ayrı yürütülebilir şekilde, ayrık süreçler halinde geliştirilebileceği gibi, zaman kısıtlamaları elverdiği takdirde, paralel çalışan görevciklerden oluşan, bir tek süreç halinde de geliştirilebilir. Altsistemden gelen veriler Veri Alıcı modül ile arayüz donanımından okunur. Bir veri katarı şeklinde olan bu ham veri, Arayüz İsterleri Belirtimi'nde ya da Teknik Anlaşma'da anlatılan yönteme göre ve sınır kontrolü yapılarak tüm alanları anlaşılabilir iletiler haline dönüştürülür. İletinin adına göre bir dallanma yapılarak içindeki veriler işlenmek üzere alınır ve bilgi haline getirilir. Bu bilgiler ana sistemin kullandığı arayüze sistem girdisi ya da bir komutun yanıtı olarak gönderilir, aynı zamanda sistem içinde bulunan ortak veri yapıları da tazelenir.

Ana sistemden gelen çıktılar ya da komutlar arayüz giriş biriminde işlenerek ortak yapıların da yardımıyla iletiler oluşturulur. Bu iletiler protokole göre veri katarına çevrilerek arayüz donanımına yazılır. Ana sistemden gelen komutlara göre bir test çevrimi başlatmak ya da durum bilgisi almak mümkündür.

Bazı altsistemler düzenli aralıklarla haberleşmek isteyebilirler. Bu iletileri üretmek, zaman aşımlarını denetlemek ve atık toplamak üzere bilgisayar saatini kullanan bir periyodik işlem birimi kullanılabilir. Bu işlevler yanında, altsistem arayüz yazılımı tüm birimlerden gelen hata iletilerini toplayıp raporlama ve sonradan çözümleme amaçlı olarak kaydetme yeteneğine sahip olmalıdır. Her altsistem, ana sisteme kendi durumu hakkında sürekli rapor (heart-beat) vermelidir.

6.3. Tasarım Yöntemleri

İyi bir tasarım için belirli bir yöntemi seçip kurallarıyla uygulamak gereklidir. Hangi tasarım yöntemi veya aracı seçilirse seçilsin iyi kullanıldığı takdirde pek çok yarar sağlar. Ancak, eksik ya da hatalı kullanım geliştirilecek yazılımın da hatalı olmasına neden olur. Çünkü, kodlama, tasarıma dayanarak yapılır. Projede resmi tasarım tekniklerinin benimsenmesi bazı etmenlerden dolayı engelleniyor olabilir. Bunlar arasında, proje sürelerinin yeterli olmaması, tasarım araçlarına yeterince yatırım yapılamaması, gerekli eğitimlerin alınamaması, üst yönetimden yeterli desteğin sağlanamaması, yazılım geliştirme personelinin isteksizliği gibi nedenler sayılabilir. Yine de belirli bir yöntem seçilerek kurallarına göre uygulanması için çaba harcanmalıdır.

Yazılım tasarımında kullanılabilecek pek çok yöntem bulunmaktadır. Bu yöntemleri şu şekilde listeleyebiliriz:

- Böl ve yönet (divide-and-conquer)
- Tümevarım (bottom-up)
- Tümdengelim (top-down)
- Aşamalı ayrıntılandırma (stepwise refinement)
- Buluşsal yöntemler (heuristic methods)
- Deneme yanılma yaklaşımı (iterative approach)

- Artımlı yaklaşım (incremental approach)
- İşleve yönelik tasarım (function-oriented design)
- Yapısal tasarım (structural design)
- Veri akışına yönelik tasarım (dataflow-oriented design)
- Nesneye yönelik tasarım (object-oriented design)
- Veriye yönelik tasarım (data-oriented design)

Tasarım oldukça geniş bir konu olduğu için, biz burada en yaygın olarak kabul edilen yöntemlerden birkaçının özelliklerine değinmek istiyoruz. Bunlar, *yapısal tasarım* (structural design), *veri akışına yönelik tasarım* (data-flow-oriented design), *nesneye yönelik* (object-oriented) *tasarım* ve *veriye yönelik* (data-oriented) *tasarım*dır.

6.4. Veri Akışına Yönelik Tasarım

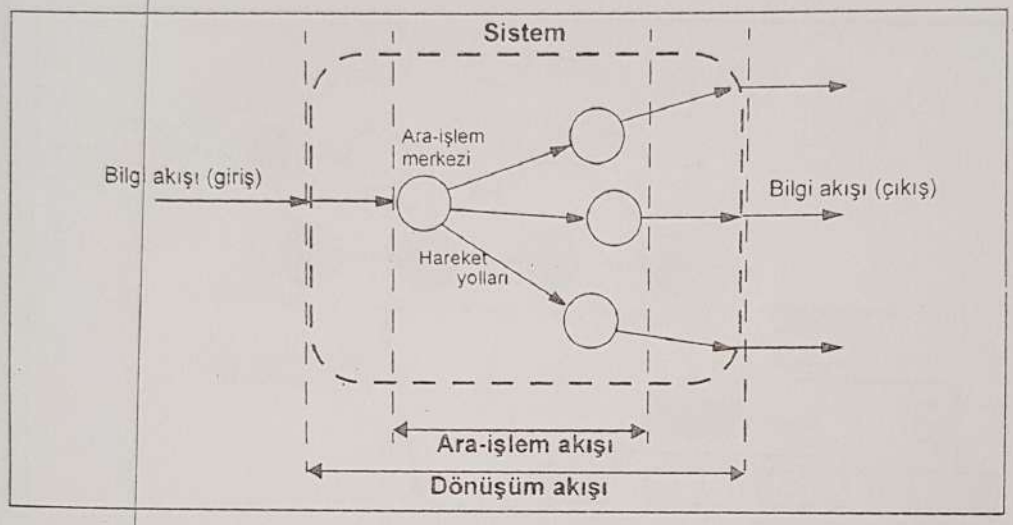
Yazılım isterleri çözümlemesinin bir parçasının bilginin çözümlenmesi olduğuna değinmiştik. Bilgisayar tabanlı bir sistemde, bilgi, belirli bir şekilde sisteme girer, çeşitli aşamalarda değişikliğe uğrar ve sistemden çıkar. Yazılım tasarımında bu bilgi akışı dikkate alınarak *veri akış diyagramları* (data flow diagram) kullanılır. Bu diyagramlara daha önce yapısal çözümleme kısmında değinmiştik. Veri akışına yönelik (data flow-oriented) tasarım yönteminde [5], verilerin değişim şekilleri program yapısına uyarlanır. Bu yönteme yapısal çözümlemeye benzediği için kimi zaman *yapısal tasarım* (structured design) adı da verilir. Bu yöntem modüler yaklaşım, yukarıdan aşağı tasarım modeli ve yapısal programlama ile birleştirilerek oluşturulmuştur. Her ikisi de aynı özellikleri taşıdığından yapısal tasarıma ayrıca değinmeyeceğiz.

Veri akışına yönelik yöntemin çok çeşitli uygulama alanları vardır. Aslında, her türlü yazılım bir veri akış diyagramıyla gösterilebilir. Ancak, veri akışına yönelik yaklaşım, özellikle sıradüzensel veri yapılarının bulunmadığı ve bilgilerin ardışık olarak işlendiği yazılımlar için daha kullanışlıdır. Karmaşık sayısal çözümleme yazılımları, mühendislikle ilgili çeşitli yazılımlar ve kontrol sistemleri yazılımları örnek olarak verilebilir. Bu yöntemin biraz daha genişletilerek gerçek zamanlı ve kesme kontrollü uygulamalarda kullanılması sağlanmıştır. veritabanı sistemleri, uzman sistemler, nesneye yönelik arayüzlerin bulunduğu sistemlerde diğer yöntemlerin kullanılması daha uygun olur.

6.4.1. Akış Türleri

Yazılım isterlerini veri akışına yönelik olarak tasarıma aktarabilmek için bilgi akışının program yapılarına dönüştürülmesi gereklidir. Bu amaçla bilgi akışının, sınırların, işleme şeklinin ve yapıların tanımlanması gereklidir.

Tanımlama için kullanılacak veri akış diyagramlarında gösterim şekli olarak Şekil-6.8 de verildiği gibi iki tür bilgi akışı yer alır:



Şekil-6.8. Dönüşüm ve ara-işlem akışı.

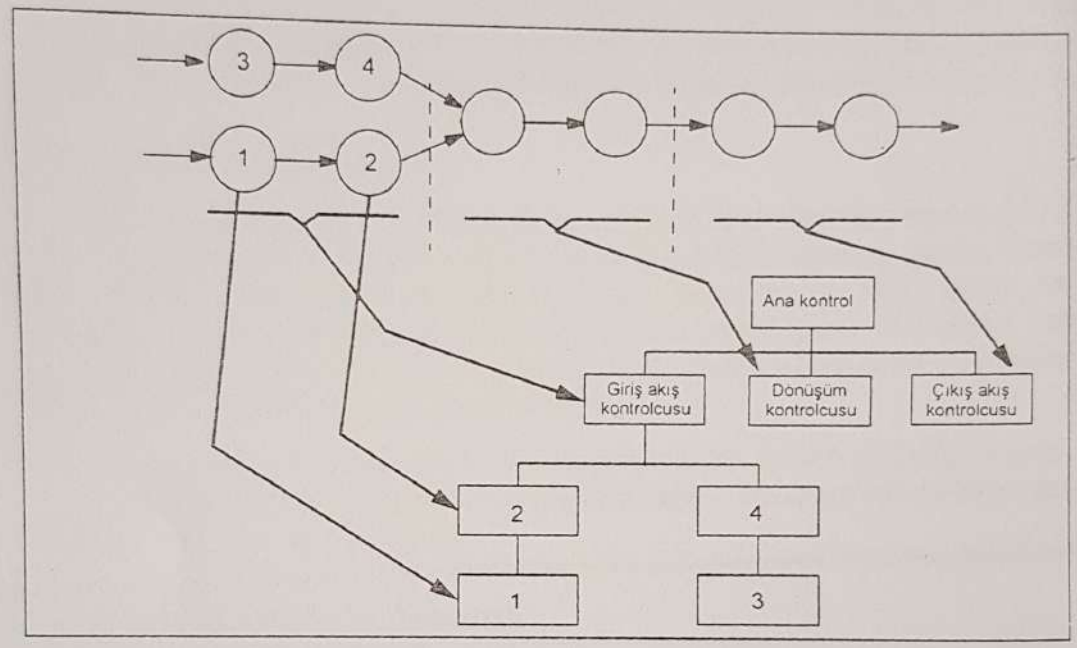
- *Dönüşüm* **(transform) akışı:**
  Her sisteme dış dünyadan bir giriş vardır. Girişler sistem içinde işlenir ve dış dünyaya çıkış şeklinde gönderilir. Giriş akışı, sistemin dönüşüm merkezinde işlenir ve çıkış haline dönüşür. Her giriş bu merkezde ardışık bir sıra ile işlenir ve bir dönüşüm akışını oluşturur.
- *Ara-işlem* **(transaction) akışı:**
  Giriş şeklinde gelen bir veri akışı, bir veri öğesine göre, çeşitli akış yollarından birine yönlendirilerek bir başka veri akışını tetikleyebilir. Bu şekilde bir ara-işlem akışı oluşur. Her akışta ara-işlem değerlendirilerek elde edilen değere göre bir hareket yolu seçilir.

6.4.2. Tasarım Aşamaları

Veri akışına yönelik tasarım, akış diyagramının incelenmesiyle başlar. Önce akış türleri belirlenir ve akışın sınırları çizilir. Dönüşüm ve ara-işlem merkezleri belirlenir. Sınırların yerine göre dönüşümler, yani daire ile gösterilen süreçler, birer modül olarak program yapısı ile örtüşür hale getirilirler. Tanımlama ve örtüşmenin yapılışı dönüşüm ve ara-işlem çözümlemeleri ile gerçekleşir. Süreçlerle yapıların hassas bir şekilde örtüşmesi, yürütme denetiminin yukarıdan aşağıya doğru süreçlere dağıtılmasıyla yapılır. Bu aşamada modülerlik özelliğinin korunmasına dikkat edilir.

6.4.2.1. Dönüşüm Çözümlemesi

Dönüşüm çözümlemesi, program yapısını oluşturmak üzere, isterler çözümlemesi sırasında yapılan işin tekrar gözden geçirilmesiyle başlar. Şekil-6.9 da gösterilmekte olan dönüşüm çözümlemesi ana basamaklarını şöyle sıralayabiliriz:

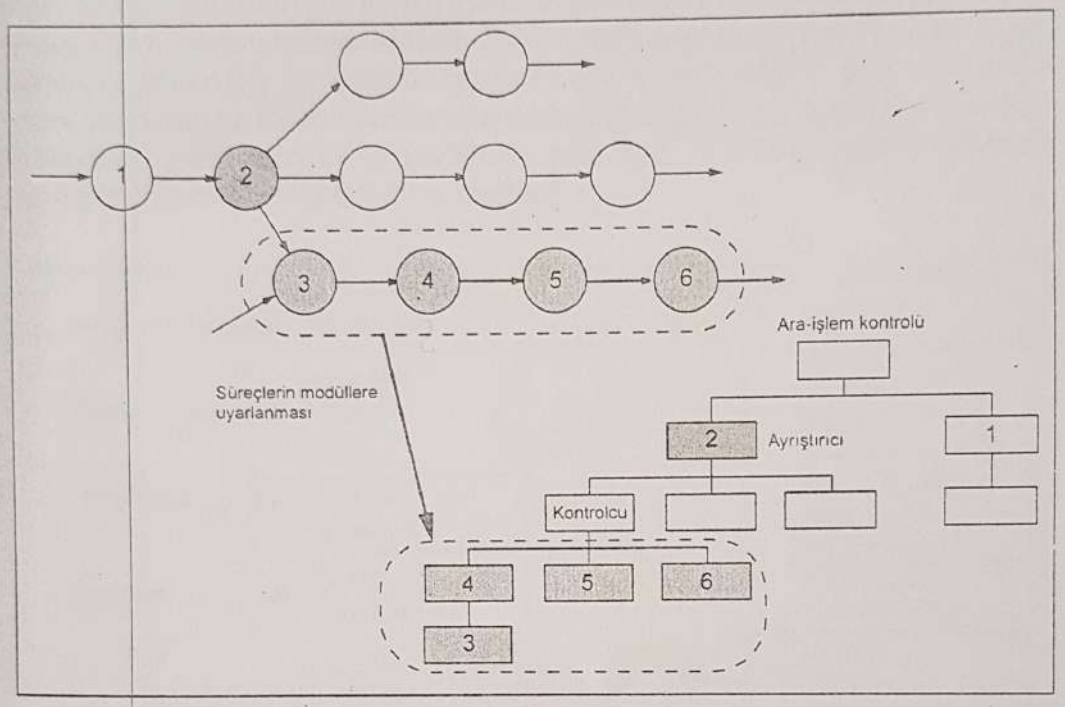


Şekil-6.9. Dönüşüm çözümlemesinde yapıya uyarlama.

- Yazılım İsterleri Belirtimi'nde anlatılan sistem özellikleri ile Düzey 0 veri akış diyagramı kullanılarak bilgi akışları, yapı ve arayüzler incelenir.
- Veri akış diyagramı daha ayrıntılı hale getirilerek Düzey 1 ve Düzey 2 veri akış diyagramları hazırlanır. Bunlar içinde dönüşüm akış özellikleri olanlar aranır. Bu maksatla, girişten çıkışa doğru giden akışlar ele alınır, ara-işlem dönüşümü olup olmadığına bakılır.
- Bilginin yorumlanarak girişe dönüştürüldüğü ve işlenen verilerin çıkış bilgisi haline dönüştüğü yerler akış sınırları olarak çizilir. Bu sınırlar arasında kalan süreçler dönüşüm merkezini oluşturur.
- Program yapısı, yürütme denetiminin yukarıdan aşağıya dağıtımının bir gösterimidir. Bu yapının oluşturulması amacıyla ayrıştırma yapılır, giriş, karar verme, bilgi işleme, çıkış ve diğer bazı toplu işlerin yapıldığı modüller belirlenir. Önce, giriş, dönüşüm ve çıkış akışları üzerinde bulunan süreçler modüllere dağıtılır, sonra da kalan süreçler en uygun modüllere bölüştürülür.
- Genel tasarım ilkeleri göz önüne alınarak program yapısı iyileştirilir. Bir kısım modüller birleştirilir ya da daha küçük başka parçalara ayrılır. Sonuçta iyileştirilmiş bir program yapısı elde edilir.

6.4.2.2. Ara İşlem Çözümlemesi

Ara-işlem çözümlemesinde izlenen yol aşağı yukarı dönüşüm çözümlemesinde olduğu gibidir. Aralarındaki temel fark veri akış diyagramının Şekil-6.10 da gösterildiği şekilde program yapısına uydurulmasıdır. Ana basamaklar aşağıda olduğu gibidir:



Şekil-6.10. Ara-işlem çözümlemesinde yapıya uyarlama.

- Düzey 0 veri akış diyagramı, ya da diğer adıyla temel sistem modeli incelenir.
- Veri akış diyagramları gözden geçirilir ve gerekiyorsa düzeltme yapılır.
- Veri akış diyagramının neresinde dönüşüm, neresinde ara-işlem akışı olduğu araştırılır.
- Bir giriş ve birden fazla çıkış olan süreçler ara-işlem merkezi olarak belirlenir ve her çıkış, yani her eylem yolu için akış özellikleri tanımlanır. Giriş yolu, yani ara-işlem akışı ile çıkış yolları için sınırlar çizilir.
- Ara-işlem akışı, dağıtıcı (dispatcher) bir program yapısına uydurulur. Girişe göre yapılacak bir dallanmada kullanılacak modüller tanımlanmış olur.
- Genel tasarım ilkeleri göz önüne alınarak program yapısı iyileştirilir. Modüllerin bazıları birleştirilerek veya daha küçük başka parçalara ayrılarak program yapısı iyileştirilir.

6.4.2.3. Modüler Tasarım

Veri akışına yönelik tasarımla oluşturulan program yapısı üzerinde bazı düzenlemeler yapılarak etkin bir *modüler*, yani *birimsel tasarım* elde edilebilir. Sistem parçalara ayrıldıkça, her parçanın karmaşıklık derecesi azalacak, dolayısıyla da sistemin toplam karmaşıklığı düşecektir. Modül sayısının çok az olması yeterli soyutlama ve ayrıştırma sağlamaz; dolayısıyla da gerçekleştirimde yarar sağlamaz. Modül sayısının çok fazla olması da hem işgücünü hem de arayüzleri arttırır, tümleştirme güçlükleri yaratır.

Modüllerin sıradüzensel yapısının gösterildiği sistem yapısında oluşan derinlik veya genişlik çok fazlaysa azaltılmalıdır. Bunun için, denetim ve karar verme yapıları yukarıya çekilmeli, yalnızca çağrılarak kullanılan yapılar alt düzeylerde tutulmalıdır.

5.4.2.4. Tasarım Anlatımı

İyi bir mimari tasarımı için dönüşüm ya da ara-işlem çözümlemesi yanında iyi bir anlatım da gereklidir. Bu amaçla yazılım tasarımının belgelendirmesi yapılır. Genel olarak Tasarım Tanımlaması (Design Description) adını taşıyan belge içinde, her modül için yapılan işin metinsel anlatımı, arayüzün tanımı, yerel ve evrensel veri yapılarının tanımı, bellek, işlemci ve zamansallık gibi kısıtlamaların belirtimi bulunur. Karar verme düzenekleri ve giriş/çıkışlar belirtilir.

Her yazılım geliştirme işlemi gibi bu ön tasarımın da bir gözden geçirmesi yapılır, bulunan eksiklikler giderilir, gerekirse iyileştirme yapılır. Bundan sonra da takip eden aşama olan ayrıntılı tasarıma geçilir.

4. Hafta
2. Grup
Son