6. Hafta
2. Grup
Boşluğu

10.8.6. **Kullanıcı Arayüz Geliştirme Süreci**

Tüm sistem yazılımı belirli bir disiplinin uygulandığı geliştirme süreci ile gerçekleştirilir. Kullanıcı arayüz yazılımı da ana sistem yazılımının temel öğelerinden biridir.

Kendi içinde bir süreç uygulanarak geliştirilir. Bu süreç yine standart yazılım geliştirme sürecine benzer ve dört aşamadan oluşur.

8.6.1. Çözümleme

Bilgisayar tabanlı bir sistem genellikle elle yapılan işleri otomatik hale getirmek üzere tasarlanır ve geliştirilir. Otomasyonun ne derece olacağı, işletmenlerin hangi düzeyde devreye gireceği, sistemin girdi ve çıktıları sistem çözümlemesi sırasında belirlenir. Bu çözümleme sırasında, sistemin kullanıcı arayüz isterleri de belirlenir. Çözümleyici bu amaçla sistemin amaçlarını ve temel işlevlerini tanımlar. Her amaca karşılık düşen eylemleri ve bunların kullanıcı arayüzünden nasıl başlatılacaklarını belirtir. Eylemler sırasında sistemin içinde bulunması gereken durumları tanımlar. Kullanılacak denetim aygıtlarını ve düzenekleri, bu düzeneklerin sistem durumunu nasıl etkileyeceğini açıkça ortaya koyar. Kullanıcıya sistem durumu hakkındaki bilgilerin nasıl sunulacağını, uyarıların nasıl yapılacağını belirtir.

Bu çözümleme sonunda sistem belirtiminin arayüz tanımlaması ortaya çıkar. Arayüz şekillerinin karmaşıklığı aynı zamanda sistemin karmaşıklık derecesini de ortaya koyar. Etkileşim sırasında kullanılan eylemlerin, komutların ve sistem durumlarının sayısı ne kadar yüksekse kullanıcının hatırda tutması gereken bilgi miktarı da o kadar yüksek olacaktır. Bu nedenle sistemin genel tasarımı sırasında arayüzün en azından ön tasarımına da yer verilmelidir.

8.6.2. Tasarım

Sistemin arayüz tasarımı sırasında kullanıcıya ne tür arayüzler sağlanacağı, giriş ve çıkışların hangi aygıtlarla, ne şekilde yapılacağı, hata ve uyarı iletilerinin nasıl verileceği belirlenir. Özel bir donanım ile arayüz gerçekleştirilecekse, donanım tasarımı aşamasında gerekli çalışma yapılmalı, uygulama alanının özelliklerine göre en uygun giriş/çıkış aygıtları seçilmelidir. Yalnızca ekran, klavye ve işaretçi aygıtından oluşan standart giriş/çıkış ile denetlenen bir grafiksel kullanıcı arayüzü bulunacaksa bunun gerekli tasarımı yazılım geliştirme aşamalarında yapılmalıdır. Grafiksel kullanıcı arayüzünü gerçekleştirmeden önce bir ilkörneğini (prototip veya ön ürün) yapmak ve bunu kullanıcı ile gözden geçirmek her zaman büyük yarar sağlar.

Kullanıcıya dostça ve standart bir görünüm sağlamak için tüm uygulamaların aynı standart şablon üzerine oturtulması sağlanmalıdır. Birçok pencere yönetim sistemi ile çalışan uygulamalarda, menü çubuğu, araç çubuğu, gövde ve durum çubuğu standart olarak bulunmakta, bunlara ait ayarlar pencere sistemi tarafından yapılmaktadır.

8.6.3. Gerçekleştirim

Arayüz tasarımı çeşitli çizim araçlarıyla yapılabileceği gibi gerçekleştirimde kullanılacak geliştirme aracıyla da yapılabilir. Bu araçlar, önceden geliştirilmiş çeşitli grafiksel şablonlar, tuşlar, menüler ve daha pek çok grafik nesneler sağlayarak çok hızlı geliştirme olanağı sunmaktadırlar. Ne yazık ki, yaygın olarak kullanılan pek çok

gereç belirli bir pencere yönetim sistemini desteklediğinden taşınabilirlik özelliği son derece zayıftır. Bu araçlar aynı zamanda otomatik olarak kod üretirler. Ya aracın sağladığı arayüz üzerinden ya da üretilen kod içine elle yazarak yordamların gerçekleştirecekleri eylemler kodlanır. Bir arayüz yazılımının pencere kısmını gelişmiş bir araç ile birkaç saatte tamamlayıp çalıştırmak mümkündür. Ancak bu arayüzü gerçek sistemle tümleştirmek, tuşlara basıldığında veya menülerden seçimler yapıldığında bir takım işlemler ve denetimler yaptırmak, hatalara karşı dayanıklılık sağlamak için önemli miktarda kod yazılması gereklidir.

10.8.6.4. Test

Arayüz testlerini ilkörnek arayüz testlerinden ayırmak gereklidir. Arayüz ile beraber yapılan işlevsel testler aslında tüm sistemin testi demektir. Zira, verilen komutlar, eylemler aslında sisteme bir giriştir, sergilenen çıktılar da sistemin yanıtı veya çıkışıdır. Bu giriş ve çıkışların doğruluğu kullanıcı için bir kabul testi sayılır.

İşlevsel testler yanında arayüzün gürbüzlüğü (robustness) mutlaka sınanmalıdır. Kullanıcının hiçbir zaman basmayacağı varsayılan tuş kombinasyonları hiç olmadık yerde ortaya çıkıp tüm yazılımı çökertebilir. Örneğin, klavye üzerine konmuş bir kitap uygunsuz bir tuş grubunu çalıştırıp arayüzün kilitlenmesine neden olabilir. Sistemi bilmeyen bir kullanıcı uygunsuz bir menü seçeneğini kullanarak sistemin durumunu değiştirebilir. İşte bu gibi durumların denenmesi, mantıklı senaryoların dışında rastgele testler yapılması, sistemi çökertmeye yönelik girişimlerde bulunulması aslında sistemin sağlamlığının sınanması için çok önemlidir.

10.9. Gerçek Zamanlı Sistem Tasarımı

Gerçek zamanlı sistemler, uygulama alanının ve isterlerin tanımlanması, tasarım, geliştirme, test, geçerleme ve çalıştırma aşamalarının herbirinde zaman unsurunun tüm yönleriyle dikkate alınması nedeniyle sıradan geliştirmeden ayrı bir yere sahiptirler. Uygulama yazılımları kadar, kullanılan dilin, altyapı sistemi veya ara katmanın, işletim sisteminin ve donanımın uygun seçilmesi zorunludur. Kaynaklardan [5], [7] ve [10] bu konuda önemli temel bilgiler içermektedir.

10.9.1. Yapısal Özellikler

Gerçek zamanlı sistemlerin yapısal özellikleri, donanım, işletim sistemi, ara katman ve uygulama yazılımlarına ilişkin çeşitli teknik özellikleri kapsar. Donanım yeterli işlemci gücünü ve belleği sağlayabilecek yetenekte, arıza yapma olasılığı çok düşük ve yedekli olmalıdır. İşletim sistemi, süreçlerin donanımla etkileşim içinde olduğu noktalarda hiçbir kesmenin kaçmasına izin vermemeli, öncelikli ve durdurulabilir (pre-emptive) iş sıralaması yapabilmeli, eşzamanlı çalışma, paylaşılır verilere erişimde karşılıklı dışlama gibi özellikler taşımalıdır. Altyapı sistemi, verilen zaman kısıtı içinde güvenli veri iletişimini sağlayabilen, hataya dayanıklı bir ara katmana ve çeşitli işlevlere sahip olmalıdır. Programlama dili, tasarımın kolay ve doğru bir şekilde koda

dönüştürülebilmesini sağlayabilmeli, istenen hassasiyette zaman değerlerinin belirtilebilmesine olanak tanıyabilmeli, kuvvetli tip kontrolü ile yürütme anı güvenliğini sağlamalıdır. Bunlara ek olarak, sistem gereksinimlerine göre aşağıdaki özelliklerin de bulunması arzu edilir:

9.1.1. Mimari

Sistemin doğasına ve iletişim gereksinimlerine bağlı olarak, sıkı bağlı veya gevşek bağlı bilgisayar sistemleri kullanılabilmektedir. Her iki durumda da, çok miktarda giriş/çıkış işlemi ile farklı düğümler üzerinde bilgi işleme gerektiren gerçek zamanlı sistemleri tasarlayıp geliştirmek oldukça güçtür. Bir kısım uygulamalar merkezi sistemler gerektirirken bazıları için gevşek bağlı sistemler daha uygun olmaktadır.

9.1.2. Özkaynak Gereksinimleri

Gerçek zamanlı bir sistem tasarlanırken özkaynak gereksinimlerinin önceden iyi kestirilmesi gereklidir. Aksi halde, olağan dışı küçük bir durumda dahi, belirlenen zaman kısıtlamalarını karşılamak mümkün olmayabilir. Verilen zaman kısıtlamaları içinde gerekli işlemlerin bitirilmesinin garanti edilebilmesi için bilgi işleme hızının ve bilgi miktarının birlikte değerlendirilmesi gerekir. Bu değerlendirme çeşitli çözümsel ya da istatistiksel bilgilere dayanan ön kestirimlerle yapılabileceği gibi, çalışma sırasında dinamik olarak da hesaplanabilir. Bu şekilde tahsis edilen özkaynakların yetersiz kalması durumunda ilgili hata kotarma düzenekleri devreye girmelidir. İsterler belirlenirken çeşitli ekyüklerin (overhead) de göz önüne alınması gereklidir.

Özkaynakların istenildiği anda elde edilebilmesi (availability) gerçek zamanlılığın en önemli gereğidir. Dinamik iş sıralaması kullanıldığında periyodik olmayan ya da yeni yaratılan görevlerin özkaynak paylaşımının da o anki gereksinime göre yapılması zorunluluğu vardır. Bu şekilde bir görevin bir özkaynağa gerek duyması durumunda, önceliğine uygun olarak ona o özkaynak verilebilir.

Özkaynakların mutlaka atanabilmesini sağlayabilmek için gerçek zamanlı işletim sistemleri çeşitli garanti düzenekleri kullanırlar. Böylelikle daha az kritik olan görevler onlardan daha kritik olan görevlerin çalışmalarını aksatmamış olurlar.

Çoğu bilgisayar sisteminde özkaynaklar tamamen işletim sistemi tarafından yönetilirler. Değişim hızına bağlı (rate-monotone) iş sıralaması yöntemi buna bir örnektir. Halen kullanılan birçok sistemde ise uygulama yazılımı bu yönetimden sorumludur. Dinamik gerçek zamanlı sistemlerde ise tasarıma bağlı olarak, yönetim, işletim sistemi denetiminde uygulamaya verilebilir. Uygulama yazılımı çeşitli anlamsal (semantic) düzenekler kullanarak gereksinimleri ve kısıtlamaları belirtir; bunların uygulanması işletim sisteminin sorumluluğundadır. Bir başka yöntemde de işletim sisteminin sağladığı çeşitli özellikler ve yetenekler uygulama yazılımı tarafından denetlenirler.