ASANSÖR DENETİM DİZGELERİNDE YAZILIM MİMARİSİ GELİŞTİRMEK

Selami KOÇLU
Elektrik Mühendisi

ÖZET

Bu tür dizgelerde yazılım sağlamlığını emniyete almak birinci önceliktir. Çok sayıda iç içe tekleşik dizge (embedded system) üst düzey emniyet ve bağımlılık şartlarını yerine getirmelidir. Tekleşik bilgisayar dizgeleri giderek daha karmaşık olur ve işlevikliği artarken, bunların yazılım tasarımı, inşası ve bakımı daha zorlaşmaktadır. Bizim yazılım sağlamlığını artırmada kullanacağımız yöntem; tam uygun yapıdır. Hemen belirtilmelidir ki tam uygun yapı özelliği, tanım ve inşa açısından oldukça zordur. Herkesçe bilinen donanım öğelerini azaltma yöntemi, yazılım güvenliği ve bağımlılığını başarma açısından yeterli değildir. Dizgenin yazılım mimarisi, tam uygun yapıyı yazılım dizgesine inşa etmede anahtardır. Bu çalışmada, tam uygun yapıyı bir asansör denetim dizgesi için, yazılım mimarisi önererek geliştirilecek ve mimarinin uygulama ve denenmesi hakkında tartışılacaktır.

GİRİŞ

Tekleşik bilgisayar dizge katı sınırlamaları-, örneğin: gerçek zaman koşulları, sınırlı donanım kaynakları, ve katı güvenlik ve bağımlılık standartlarında,-zorlu tasarım imkanları sunmaktadırlar. Bu tür dizgeler giderek karmaşık olurken yazılımlarda gerçekleştirilen işleviklikte giderek artmaktadır. Maalesef karmaşık yazılım dizgelerinin güvenlik ve bağımlılığını kesin emniyete almak henüz çözülememiş bir sorundur. Yazılım sağlamlığı, yazılımın güvenlik ve bağımlılık özelliklerinin başarılmasında çok yardımcı olur. Yazılım sağlamlığından kastedilen, istisnai ve aşırı zor çevre koşullarında görevini doğru olarak yapabilmesidir. [IEEE 90]

Yazılım sağlamlığını başarmadaki yaklaşımlardan biri 'tam uygun yapıdır'. TUY dizge, bir parçada meydana gelen bir arıza dizgenin işlevikliğini azaltır, dizgenin tamamının çökmesine yol açmaz. TUY u uygulayabilmek için tasarım esnasında her arıza moduna karşılık bir sayı atanır, ve her arıza için bir prosedür önerilir. Ayrıca tam uygun dağıtık yazılım dizgelerinde bütün donanım ve yazılım arızalarını saptamak mümkün olmayabilir, ve bunları da sayılaştırmak kısıtlı tasarım süresinde gerçekçi olmaz. Dizgeyi o şekilde kurmalıyız ki, arıza ortaya çıktığında, her olası arızayı belirtmek zorunluluğu ve önceden düzeltme eylemi olmadan, otomatik olarak kritik olmayan işlevler dizgeden çıkmalıdır.

Yazılım mimarisinin üst düzey soyutlaması, TUY u yazılım dizgesine yerleştirmede anahtardır. Yazılım mimarisi; “ yazılım öğeleri, yazılım öğelerinin dıştan görünen özellikleri ve onların arasındaki ilişkileri kapsayan yapı, veya dizge yapısı” olarak tanımlanır. Yeniden kullanılabilir yazılım öğeleri çoğalırsa, yazılım tasarımı bir

sentezden ziyade bir harç (composition) uygulaması olur. Bundan dolayı, bütün dizge yapısı ve harcın kumuna çakılına ayrılması, öğelerin beraber çalışmasını ve dizge işlevikliğini yönetir. TUY özelliğini ilerleten bir mimari biçim geliştirilebilseydi, ve tekleşik dizgelerin farklı alanlarına başarılı bir şekilde uygulanabilseydi, bu çeşit dizgelerde tasarım gayretleri öğelerin yeniden kullanımında manivela etkisi yaparken, emniyet ve bağımlılık ta korunmuş olurdu.

Çalışmamız, kendini kendiliğinden düzenleyen tekleşik dizgelerin bir parçası sayılabilir. Dizgede bir arıza ortaya çıktığında, TUY u başarabilmek için kendini kendiliğinden düzenleme (KKD=automatic reconfiguration) uygulanır. Dizge yazılım mimarisi, öğeler arasındaki arayüzleri ve geçerli öğe düzenlemelerini tanımlar. Bu sayede, uygun biçimde inşa edilmiş mimari, bir arıza ortaya çıktığında en yüksek işlevikliği sağlamak için, öğeleri yeniden yerleştirirken yeniden düzenleme yöneticisine yardımcı olur. Yapı denetlenebilir ortamda işlevikliği yaymaktadır. Bu doğrultuda atacağımız ilk adım; belirtilen özellikleri bir örnek dizgede gerçekleştirmektir.

Makalenin geri kalan kısmı, bir asansör denetim dizgesinde anılan mimari yapı için bir öneridir. Bir asansör dizgesi yeteri kadar karmaşık bir dağıtık tekleşik dizgedir. Bu dizgede TUY için bir mimari yapı geliştirerek, tekleşik dizgelere kısıtlamalar bağlamında mimari açıdan bakış atmak, asıl amacımızdır.

ASANSÖR DİZGE MODELİ

Asansör karmaşık dağıtık denetim dizgesidir. Katı emniyet gereksinimleri vardır: insanları kapılar arasında sıkıştıramaz, güvensiz hızlarda seyahat edemez, ve insanları kabinde mahsur tutamaz gibi. Biz asansör modeli geliştirirken algılayıcılar, aktuatörler, ve alınan ve verilen değerleri kullanacağız. Bu dizge modelinde asansörün bazı çok karmaşık özelliklerini kullanmayacağız; yangın yanıt modu, bakım modu, alt-uç ve üst-uç modu gibi, ama ilginç olacak kadar yeterli özellikleri kullanılacaktır.

Asansör tek kabinli olup, raylar üzerinde hareket etmektedir ve bir grup kapı bulunmaktadır. Kabin de tek kapı ve tabii kapı motoru vardır, sürücü kabini iki hızda hareket ettirmektedir (yavaş ve hızlı), ve güvenlik için acil durdurma freni kullanılmaktadır. Kullandığımız notasyonda zincirli ayraçlar ({ }) arasındaki değerler algılayıcı ve aktüatör dizilerinin standart yanıtı, ayraçlar ( ) arasında ki değerler ise algılayıcı ve aktüatörlerin çıktılarıdır. Örnek KaTta algılayıcısı bir algılayıcılar dizisi olup k:(asansörün servis yaptığı kat sayısı), d:(asansörün hareket doğrultusu: aşağı, yukarı, dur), dizinin her bir öğe değeri (v), ya doğru veya yanlış (true, false) atanır. Asansör bir k katına yaklaştığında, ya aşağı ya da yukarı olabilir, Öyle bir şekilde KaTta algılayıcısı bulunur ki, kabinin aşağıdan mı yoksa yukarıdan mı geldiğini saptayabilir. Kabin ilgili kat seviyesine tam uygun hale geldiğinde KaTta{geçerli kat, stop} algılayıcısı true olur.

DİZGEDE BULUNAN ALGILAYICILAR:

KaTta{k,d}(v) : Kat yaklaşım algılayıcısı, k=kat, d={yukarı, aşağı, dur}, v={true, false}.

KabinÇağrı{k}(v) : Kabin çağrı butonları, k=kat, v={true, false}, hepsi kabinde bulunur.

KapıKapalı(v) : Kapı kapatma anahtarı, v={true, false}, kapı tam kapandığında true üretilir.

KapıAçık(v) : Kapı açma anahtarı, v={true, false}, kapı tam açıldığında true olur.

KapıTers(v) : Kapı ters algılayıcısı, v={true, false}, kapıda engelleme olduğunda true üretilir.

DışÇağrı{k,d}(b) : Dış çağrı butonları, k=kat, d={yukarı, aşağı}, b={basılı, serbest}, apartmanda her katta bulunur.

KuyuSınır{d}(v) : Kabinin kuyudaki emniyeti için kullanılan sınır anahtarları, d={yukarı, aşağı}, v={true, false}, en alt ve en üst katta kabin sınır seviyeleri geçtiğinde true üretilir.

SürücüHız(h,d) : Ana sürücü hız algılayıcısı, h={hızlı, yavaş, dur}, d={yukarı, aşağı, dur}

DİZGEDE BULUNAN AKTÜATÖRLER:

KapıMotor(m) : kapıdaki motor, m={açık, kapalı, dur}

Sürücü(h,d) : 2-hızlı ana asansör sürücüsü, h={hızlı, yavaş, dur}, d{yukarı, aşağı, dur}.

KabinİkazI{d}(f) : Kabin ikazları, d={yukarı, aşağı}, f={on, off}, kabin çerçevesinde bulunan aşağı/yukarı ışıkları o anki kat hakkında bilgi verir.

KabinIşık{k}(f) : Kabin çağrı buton ışıkları, k=kat, f={on, off}, bir kat seçildiğinde gösteren kabin çağrı ışıkları.

KabinDurumu(k) : Kabindeki pozisyon göstericisi, k=kat.

DışIşık{k,d}(f) : Asansöre dışarıdan bineceklerin kullandığı çağrı buton ışıkları, k=kat, d={yukarı, aşağı}, k={on, off}, yolcuların istediği katta yanan çağrı butonları.

AcilFren(b) :Acil durdurma freni b={on, off}

Bunlar modern bir asansörü tam olarak betimleyemese de, oldukça karmaşık asansör davranışlarını rahatlıkla açıklayabilir. Bir sonraki bölümde asansör denetim dizgesi için, TUY u bir özellik olarak yerleştirdiğimiz yazılım mimarisi, yukarıdaki değişkenlerle açıklanmaya çalışılacaktır.

ÖNERİLEN YAZILIM MİMARİSİ

İç öğelerde arızalar ortaya çıktığında tam uygun yapı, dizge sürekliliğini önemser. Bu yüzden yazılım öğelerinin arasında, özellikle kritik işlevlerde, çok az bağımlılık olmalıdır. Bu, her yazılım öğesinin görevlerini yapabilmeleri için, donanım algılayıcıları ve aktüatörlerine doğrudan bağlanması anlamına gelir. Dizgedeki her görevin dizgedeki öteki öğelerle bağlantılı olmadan, sadece bir yazılım öğesi ihtiyacı olmasına da pek benzemez. Mükemmel ayarlanmış dağıtık dizgede, görevlerin tamamlanması için öğelerin uyumlu beraberliği gerekir. Bundan dolayı TUY u gerçekleştirmek için öğeleri olduğunca gevşek bağlamalı, ve öğeleri yapılabilirse veya işlev özellikle kritikse, özerk (otonom) yapmalıdır.

Asansör denetim dizge mimarisindeki temel düşünce; dizge işlevikliğini, kritik ve kritik olmayan olmak üzere ikiye ayırmaktır. Asansörün çalışmadığı dizge çöküşünden önceki çalışır hal, yani en alt düzey işleviklik esas alınır. Öğelerin görevlerini kendiliğinden yapması (otonom) ve çok geniş hata payları olursa, o zaman ek kritik olmayan öğeler ve işlevler, çekirdeğe (çekirdekten kastedilen yazılım çekirdeği) eklenebilir. Bu yeni ekler çekirdek kümesinin kısıtlamalarını kaldırmadığı sürece, yani çalışmazlarsa, anılan öğeler dizgeden çıkarıldıklarında asansörün en alt seviyede çalışmasına izin verirler.

Bu düşünceyi asansör modeline uygulayalım: bir asansör en basit halde sadece kuyuda katlar arasında aşağı ve yukarı hareket eder, kapıları açar ve kapar, yolcuların güvenliğini sağlar. Asansör kuyuda yavaş bir hızla da hareket edebilir, katlarda durarak yolcuların inip binmesine izin verir, her katta kapıları açıp kapatabilir, bütün yolcu istemlerini gözardı ederek ve yolcuya geri bilgi vermeyerek, her ne kadar verimsiz bir çalışma süreci olsa da, hala bir asansör olarak kabul edilebilir. (gerçek asansör dizgelerinde yolcu istemlerinin yerine gelmediği çalışma modu olarak düşünülebilir). Dizgedeki öbür bütün işlevler: yolcu istemlerinin yerine getirir, yolcuya geri bilgi aktarır, ve sadece gerekli katlarda durur, ve
dizgenin verimliliğini arttırır.

Mimarimizde minimum işlevi sağlayan öğeler; kapı denetim, sürücü denetim ve güvenlik öğeleridir. (Şekil 1) Bunların hepsi doğru işlem için algılayıcılarla doğrudan bağlantıdadır, ve birbirleri ile iletişimde bulunmazlar. Kapı denetimi; Sürücü hareketli iken ve Kabin durduğunda kapıyı açıp kapamak için mutlaka bilgi alır, bu sayede ilgili algılayıcılara erişilmiş olur. Sürücü Denetimi; kapılar tamamen kapatıldığında ve katta durduğunda mutlaka bilgi sahibi olmalıdır. Ayrıca kuyu sınır algılayıcıları iç güvenlik denetimleri için erişilebilir olmalıdır. Emniyet nesnesi, sürücü ve kapı güvensiz bir hareket yaptığında veya yaparsa algılayabilmelidir. Örnek olarak hareket halinde kapıların açılması, katlar arasında kapıların açılması, kuyu alt ve üst sınırları geçildiği halde hareketin devam etmesi verilebilir.

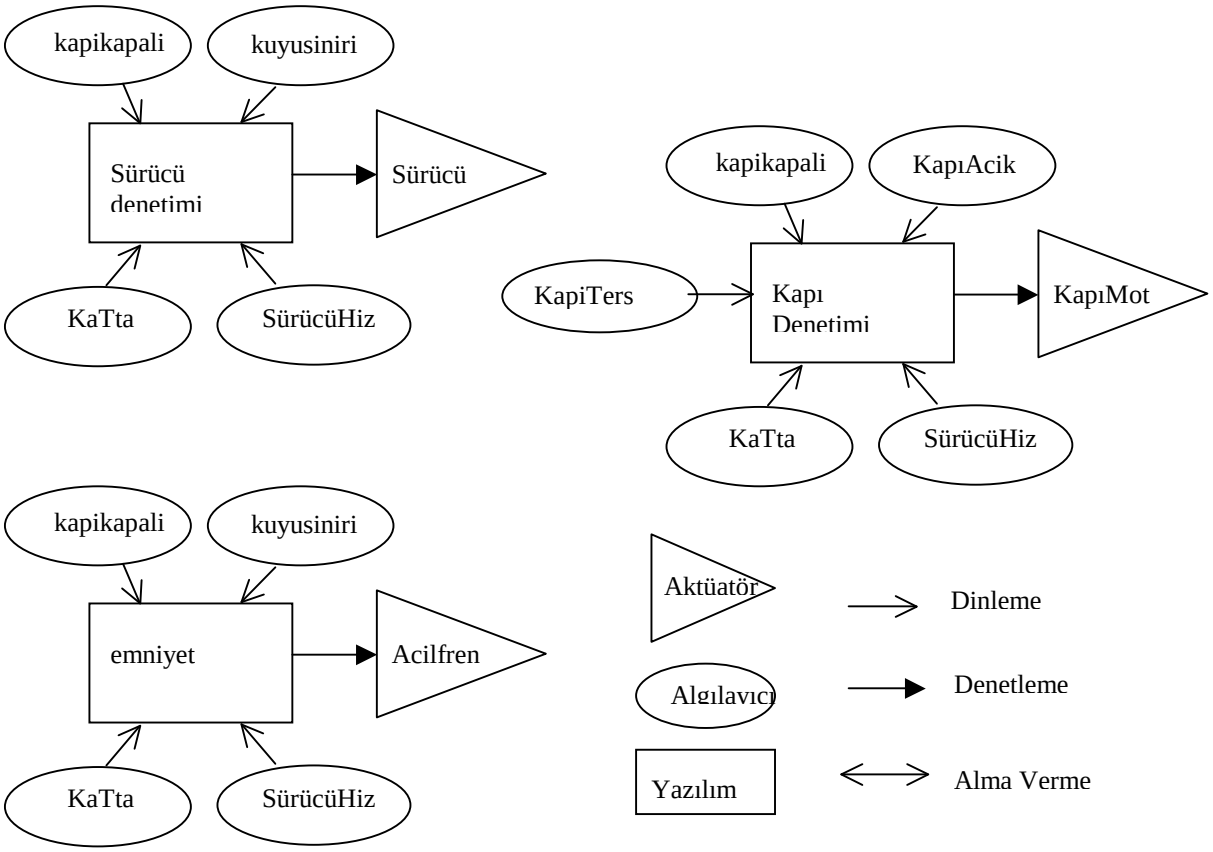
Yazılım mimarisinde bunlar mantık algılayıcıları olarak adlandırılırlar. Uygulama ya aynı algılayıcılara çok sayıda Girdi/Çıktı kanalı ile, ya da her yazılım öğesine çok sayıda farklı algılayıcı ile gerçeklenir. Buradaki tercih maliyet/güveninirlik oranı tarafından saptanır ve yazılım mimarisini etkilemez. Yazılım öğeleri algılayıcılar ile aralarında aynı arayüzlerikullanırlar ve bu arayüzü fiziki düzen etkilemez. Kritik öğeler için algılayıcıların yanıtları, tek noktada ortaya çıkan bir arızanın dizgede felakete yol

açmasını önler. Dizgedeki bu öğelerden biri çalışmayacak olursa, dizge güvenli olmaktan çıkmalı ve işlem mutlaka durmalıdır.

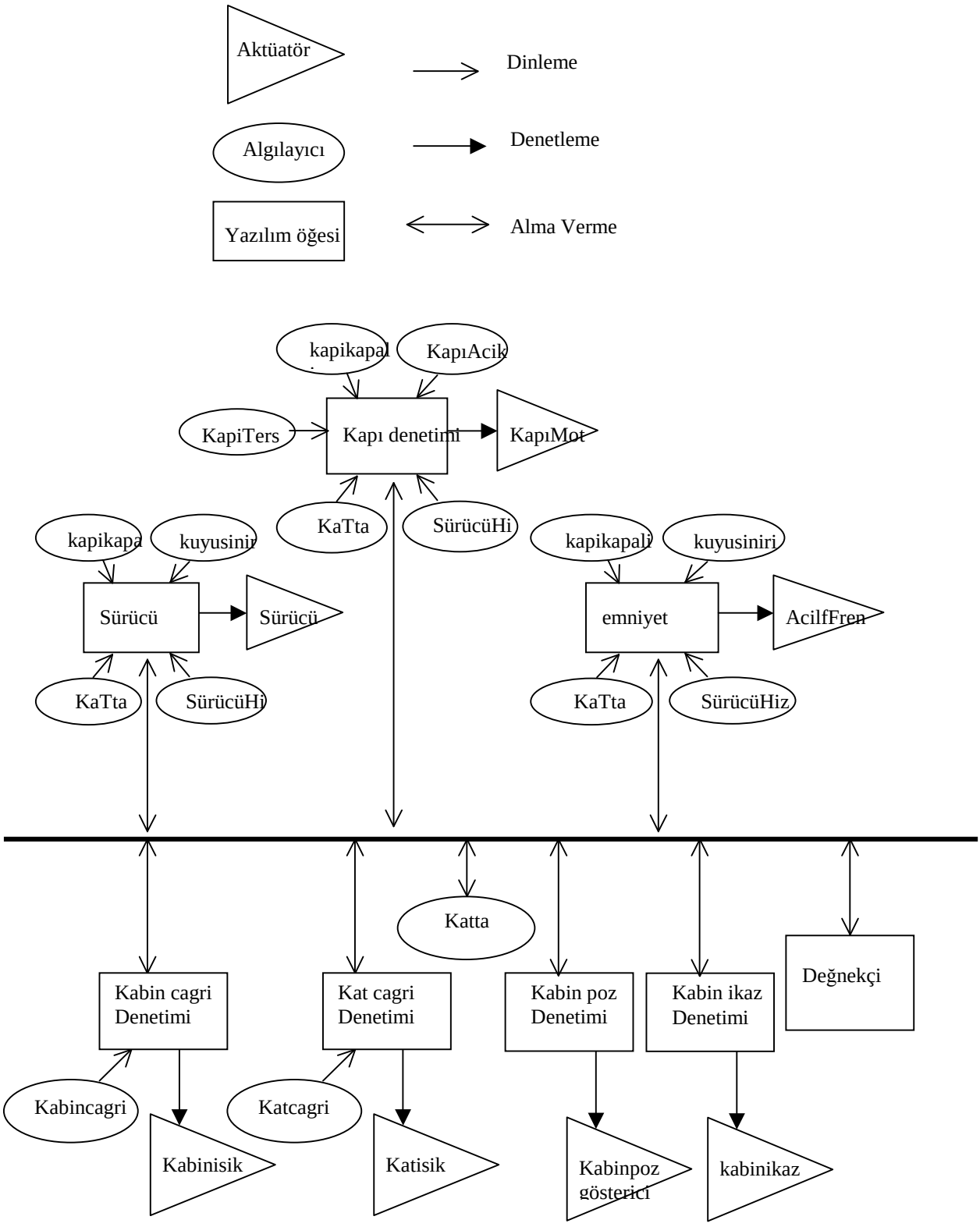
Yukarıda anlattığımız ana düzenlemeye, öğelerin iletişimi ve birlikte çalışabilmesi için, gerçek zamanlı ağ hattı da ilave edilebilir. Daha sonra da, kabin buton ve kat buton denetleyicileri eklenip yolculardan gelen kullanıcı girdileri yönetilebilir; kabin ikaz denetleyicisi ve kabin pozisyon gösterge denetleyicisi de kullanıcı geri beslemesi olarak kullanılır. Bir değnekçi (dispatcher) öğesi asansör kabin varışlarını etkin şekilde düzenler. (Şekil 2).

Gerçek zamanlı ağın ilave edilmesi ile birlikte her ek öğenin doğrudan algılayıcılara erişmesi gerekmez. Öğeler arasında, denetim durumu kritik öğelerde işlevliği arttırmak için tavsiye olarak aktarılabilir. Tavsiye komutuna örnek olarak Değnekçi öğesi sürücü denetleyicisine bir sonraki kat olan altıncı katta yolcuların beklediğini orada durması gerektiğini bildirir. Eğer Kabin 2. katta ise, Sürücü Denetleyicisi ara katları atlayarak doğrudan 6. kata gider. Sürücü denetleyicisi 2. kattan hareketin güvenli olup olmadığına karar vermelidir. Eğer karar vermezse sadece Değnekçi hareket ettiremez, sadece durduracağı kat hakkında tavsiye alır. Değnekçi tavsiye komutları göndermeyi durdurursa, Sürücü denetleyicisi Değnekçinin çalışmadığını bildirir, ve varsayım olarak her katta durmayı uygulamaya koyar. Kritik olmayan öğelerden biri arızalanırsa, kritik öğelerden herhangi biri ile bağlantıda bulunmaması lazımdır veya ideal olarak, kritik olmayan öğelerden hiçbiri ile de bağlantıda olmamalıdır. Standart kabul; öğelerin “arıza sessizliği” olduğudur, Yani iletilerin gönderilmesi durur. Bu mimari kalıbın sağladığı en önemli fırsatlardan biri; her öğenin arızasını kendisinin, veya bir başka öğenin saptamasıdır. Eğer öğeler arıza sessizliğindeyse, zaman aşımı ile saptanabilir, ama bir öğe bozulduğunda yanlış ileti göndereceği için veya yanlış bilgi vereceğinden o zaman belirlenebilmeleri oldukça zordur.

Başka bir fırsatta, kritik çekirdek öğe koşullarını bozmayacak kritik olmayan öğelerin onaylanması ile ilgili olanlardır. Yaklaşımımız, öğeler arasında gönderilen bütün iletilerin tam tanımlanmış veri biçiminde, sağlam bir şekilde arayüz belirtmektir. Bunun yanında koşulları yerine getiren öğelerin arayüze yapışıklığını güvenceye almaktır.


kapikapali
kuyusiniri
Sürücü denetimi
Sürücü
KaTta
SürücüHiz
kapikapali
KapıAcik
KapiTers
Kapı Denetimi
KapıMot
KaTta
SürücüHiz
kapikapali
kuyusiniri
emniyet
Acilfren
KaTta
SürücüHiz
Aktüatör
Algılayıcı
Yazılım
Dinleme
Denetleme
Alma Verme

Sekil 1


Aktüatör
Dinleme
Algılayıcı
Denetleme
Yazılım öğesi
Alma Verme
kapikapal
KapıAcik
KapiTers
Kapı denetimi
KapıMot
KaTta
SürücüHi
kapikapa
kuyusinir
Sürücü
Sürücü
KaTta
SürücüHi
kapikapali
kuyusiniri
emniyet
AcilfFren
KaTta
SürücüHiz
Katta
Kabin cagri Denetimi
Kat cagri Denetimi
Kabin poz Denetimi
Kabin ikaz Denetimi
Değnekçi
Kabincagri
Katcagri
Kabinisik
Katisik
Kabinpoz gösterici
kabinikaz

Şekil 2