Lego Challenge ve Yazılım Mimarisi

Sedat Görmüş, PhD

Email : sedatgormus@gmail.com

: sedatgormus@ktu.edu.tr

İçerik

- Lego Challenge
- Mimari Adımları

Lego Challenge

- Bu projede 5 kişilik takımlara ayrılarak legodan bir kule yapacağız

- Takımlarda
  - 1 mühendis , lego parçalarına bu kişi tasarım sırasında dokunamaz. Mimardan direktifler alabilir. Mühendis konuşamaz.
  - 2 Mimar, gerçeklenecek modelin tasarımını yaparlar. Kendi aralarında konuşabilirler ve legolara dokunabilirler.
  - 2 Gözlemci, gözlemciler proje sırasında konuşamazlar. Sadece projenin tasarımı sırasındaki adımları not alabilirler.

Lego Challenge Gereksinimler

- 15 dakikalık dizayn aşamasından sonra size verilen parçalardan;
- En az miktarda parça kullanarak, en yüksek kuleyi en az lego kullanarak yapmanızı istiyorum.
- Kuleyi yapmak için 60 saniyeniz var ve kuleyi mühendis tek başına kimseden yardım almadan yapacak.
- Artırılan her parça size 10 puan, her cm yukseklik 10 puan kazandıracaktır.
- İşi hızlı bitirmeniz halinde her saniye size 3 puan kazandıracaktır.
- Parçaların en az yarısını kule inşaatında kullanmalısınız.
- Kule yapıldıktan sonra en az 60 saniye ayakta kalmalıdır.



Mimari dizaynı

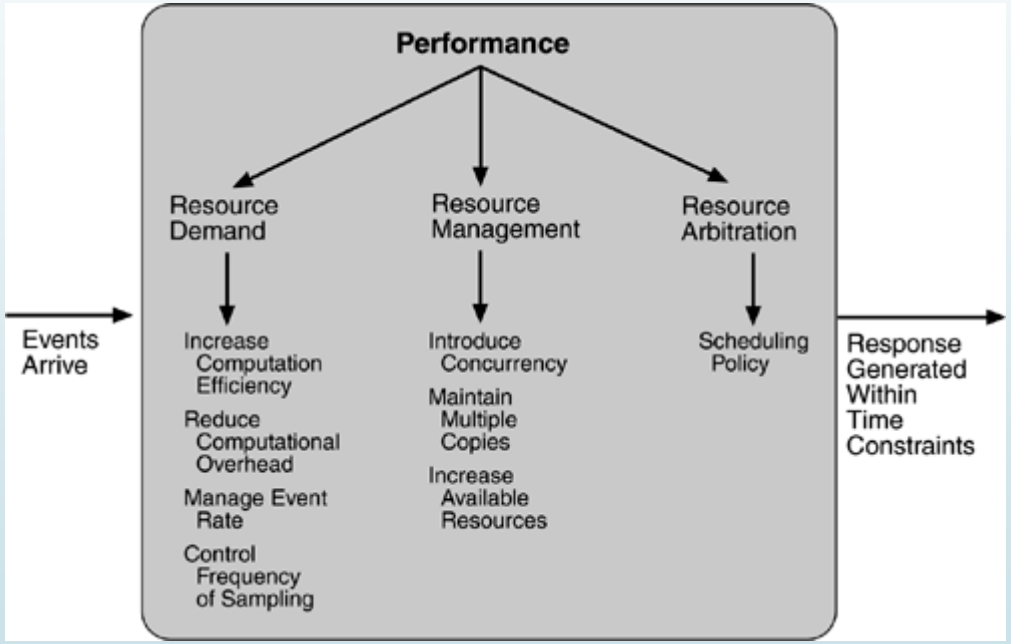
Teşekkürler

Kaynak Yönetimi

- Mimari kaynak yönetimiyle ilgili planları içermelidir.
- Kaynaklardan bazıları
  - Hafıza (özelikle sürücü dizaynı ve gömülü sistemler)
  - CPU
  - Disk
  - İletişim kaynakları
  - FPGA içindeki kapı miktarı
  - …

Performans

- Performans beklentileri gereksinimlerde tanımlanmalıdır.
- Mimari ise performans parametrelerinin birer tahminini içermelidir.
- Eğer bazı alanlarda performans sağlanmama riski varsa mimari bunu belirtmelidir. (Bu sayede geliştiriciler, bu alana daha fazla özen gösterebilirler)
- Ayrıca mimari her sınıf için zaman ve alan gereksinimlerini içerebilir.



Genişletilebilirlik.

- Mimari, ürünün ne şekilde genişletilebileceğini belirtmelidir.
- Genişlemeye örnek olarak kullanıcı sayısı, Networkteki düğüm sayısı, Veri tabanındaki girdi sayısı verilebilir.
- Eğer sistem büyümeye müsait değilse, ve genişlemesi beklenmiyorsa, mimari bunu açık şekilde ifade etmelidir.



Lokalizasyon (Yerelleştirme)

- Eğer yapılan proje farklı coğrafyalarda çalışacaksa, birden farklı dili içermelidir.
- Bunun için mimari farklı ortamları desteklemek için gerekli mekanizmaları tanımlamalıdır.
- Arayüz dili, hata mesajları, log mesajları, v.s. Birden fazla dilde tanımlanmalıdır.
- **Soru: Farklı dilleri desteklemek için uygun bir mimari tasarlanmanız istenirse, nasıl bir yöntem aklınıza gelir?**

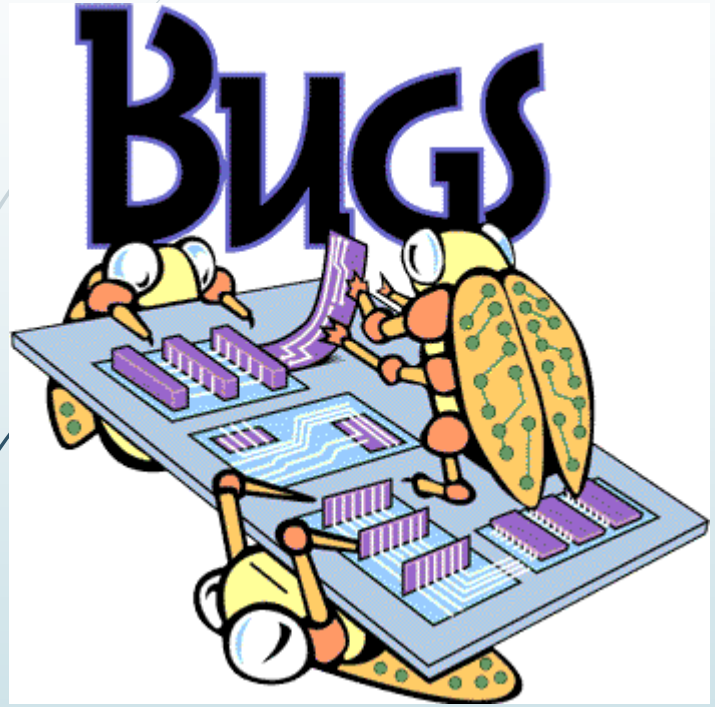


Hata İşlemleri



- Hata ayıklama ve işleme, bir çok proje için en büyük problemlerden birini teşkil eder.
- Dolayısıyla, hata ayıklama ve yakalama için yazılacak rutinlerin nasıl dizayn edileceği mimarinin bir parçası olmalıdır. (**Hata yakalama alışıla gelmiş olarak kodun içinde yapılan bir uygulamadır**)
- Hata ayıklama için mimari dizaynı sırasında sorulacak bazı sorular:
  - Hata işlemleri, düzelticimi yoksa hatayı bulmak için mi yapılıyor?
  - Eğer düzeltici ise, hatadan sağlıklı bir şekilde kurtulabilir miyiz?
  - Eğer hata yakalama için yapılıyorsa, program hatayı log dosyasına yazıp, bir şey olmamış gibi devam edebilir yada çıkabilir.
  - Her iki durumda kullanıcıya bir uyarı göndermelidir.
  - Program hata mesajlarını sistemde nasıl yönlendirmelidir (propagate)?

Hata işlemleri



- Hata mesajları çıktı formatları ve hata ayıklama yöntemleri ve kod stilleri ve yönelimleri belirlendi mi?
  - Karma karışık bir hata ayıklama yaklaşımı, kodu ve hata ayıklama sürecini içinden çıkılmaz hale getirir. (Güzel ve standart kod yaklaşımı yazılımın her parçası için geçerlidir).
- Hatalar nasıl işlenecek?
  - Mimari yazılımın hangi durumlarda hata vereceğini ve bu hataların nasıl yakalanacağını belirlemelidir.
  - Örneğin, verilen hatalar hata ayıklama sınıfına mı yönlendirilecek?
  - Yada yazılımda oluşan hata hatayı oluşturan ilk komutun olduğu noktaya kadar yönlendirilecek mi?
- Sınıfların gelen verilerin geçerliliğini belirlemek için sorumluluğu nedir?
  - Her sınıf kendi verisini kontrol etmeli midir yada bir sistem sınıfı mı bu işlemi yapacaktır?
  - Sınıflar gelen verinin temiz olduğunu varsayabilirler mi?
  - Hata ayıklama sistemini siz mi dizayn edeceksiniz yada programlama dilinin hata ayıklama sistemini mi kullanacaksınız?

Dayanıklılık(Fault Tolerance)



- Dayanıklılık(fault tolerance) sistemin kararlılığını arttırmak için kullanılan teknikler bütünü olarak görülebilir.
- Mimari projeden beklenen dayanıklılık şeklini ve hedefini belirlemelidir.
- Örneğin, bir sistem bir sayının karekökünü farklı dayanıklılık ilkeriyle şu şekilde hesaplayabilir;
  - System bir hata algıladığında, hata olmadan hesap yapılan en son noktaya geri dönebilir.
  - Sistem hata algıladığında, yedekte tutlan bir kod parçasını koşabilir.
  - Sistem birden farklı kare kök alma sınıfını içinde barındırır ve sistem he sınıfın çıktısını diğerleriyle karşılaştırır. Ve sonuç bir oylama yöntemiyle belirlenir.

Mimari Fizibilite



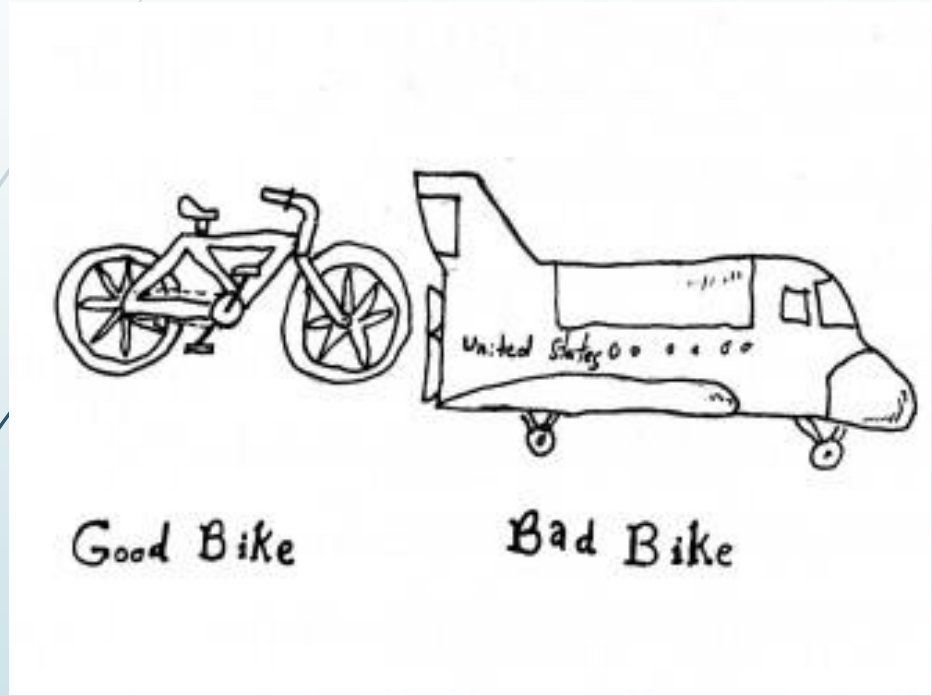
- Mimariyi dizayn eden grup, sistemim performans hedeflerini yakalayıp yakalamayacağından emin olmalıdır.
- Bu kaygılar;
  - Kaynak gereksinimleri
  - Performas hedefleri
  - Ve geliştirme ortamı tarafından destek yönleriyle incelenmelidir,
- Design yapan grup sistemin teknik olarak mümkün ve fizibıl olduğunu göstermelidir.
- Her fizibilite risk faktörü dizayn yapan kişi(ler) tarafından belirtilmeli, ve bu risklerin nasıl engellenebileceği belirtilmelidir.

Mimari Fizibilite



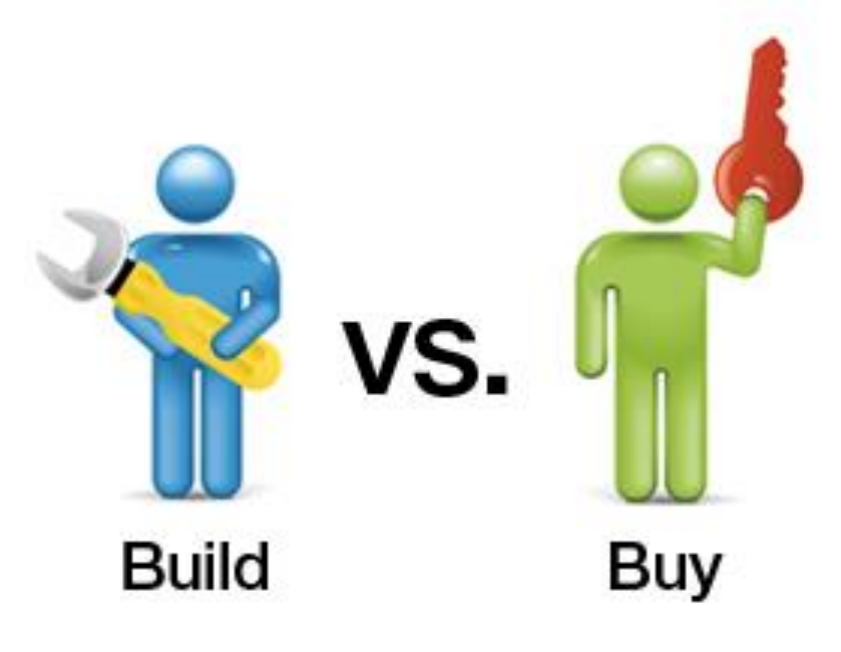
- Mimariyi dizayn eden grup, sistemim performans hedeflerini yakalayıp yakalamayacağından emin olmalıdır.
- Bu kaygılar;
  - Kaynak gereksinimleri
  - Performas hedefleri
  - Ve geliştirme ortamı tarafından destek yönleriyle incelenmelidir,
- Design yapan grup sistemin teknik olarak mümkün ve fizibıl olduğunu göstermelidir.
- Her fizibilite risk faktörü dizayn yapan kişi(ler) tarafından belirtilmeli, ve bu risklerin nasıl engellenebileceği belirtilmelidir.

Overengineering(Aşırı dizayn)



- Kararlılık(robustness) sistemin bir hata oluşmasına rağmen çalışmaya devam etmesinidir
- Çoğu zaman sistem mimarisi sistemin gereksinimlerinin önerdiği kararlılıktan fazlasını belirtir.
- Bunun nedeni bilinçli olabileceği gibi, sistem dizaynından da kaynaklanabilir.
- Eğer bir sistem kendi içinde kararlı alt sistemlerden oluşursa, toplu sistem kararlılık gereksinimden daha kararlı olabilir.
- Yada tam tersi bir durum ortaya çıkabilir.

Satın alma vs. Programlama



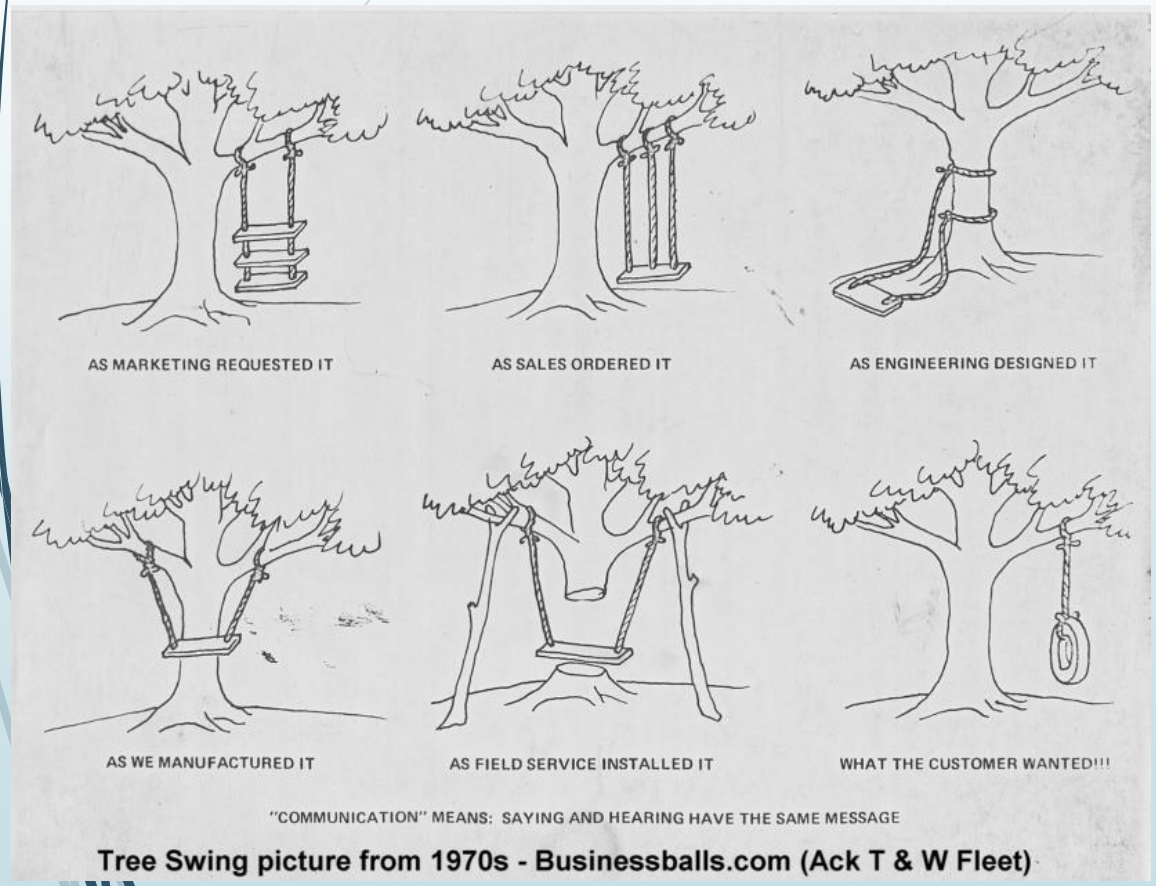
- Proje parçalarının tamamının aynı grup tarafından eksiksiz kodlanması, bazı projeler için pahalı olabilir.
- Bir çok başarılı proje açık ve free kaynak kodlu projelerden türemiştir.
- Bu yüzden projenin parçalarını mimaride satın alınalacak ve kodlanacak olmak üzere ayıtmak da mimari dizaynın bir parçası olmalıdır.
- Eğer proje hazır komponentleri kullanıyorsa, bu komponentlerin API ve arayüzleri mimari dizayna dahil edilmelidir.

Tekrar Kullanma ve Mimari (kod) değiştirme Stratejisi



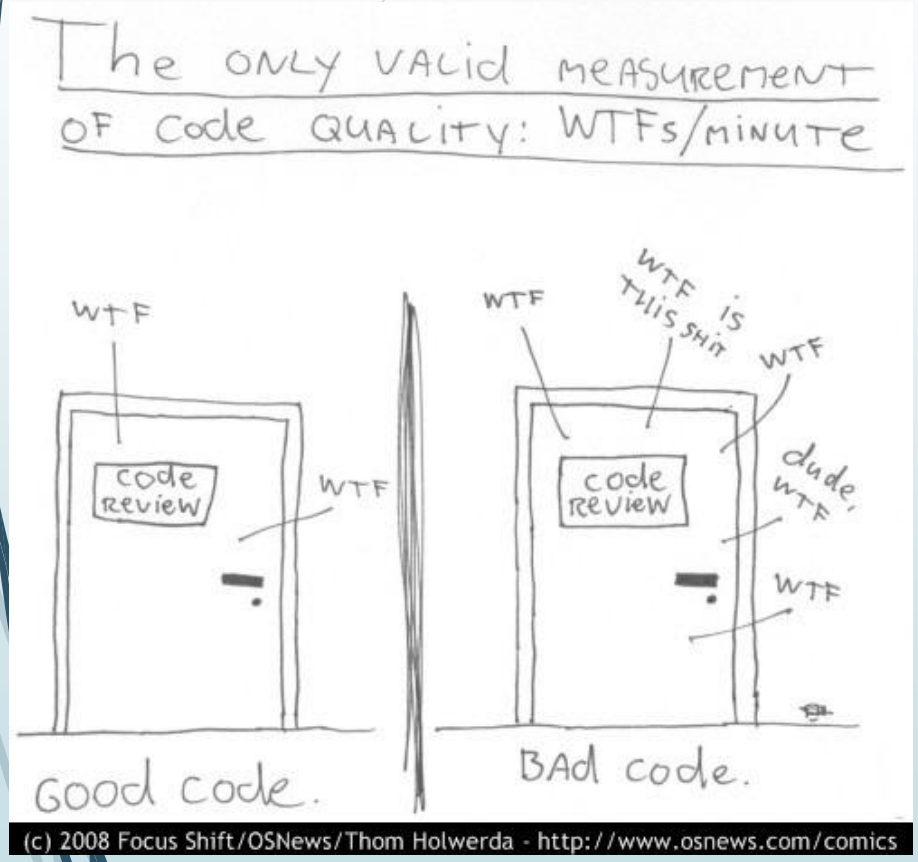
- Eğer projede daha önce yazılmış kodlar kullanılacaksa, bu kodların yeni projeden nasıl kullanılacağı mimari dizaynda belirtilmelidir.
- Daha önce belirttiğimiz gibi gereksinimler ve mimari dizayn proje sırasında değişecektir.
  - Bu durumda bir değiştirme stratejisi belirlenmelidir.
  - Ayrıca mimari olabildiğince esnek yapılarak değişikliklerin projeye olan etkisi azaltılmalıdır.
  - Mimari dizaynı yapan grup, olası değişiklikleri listelemeli, ve nasıl gerçekleşeceklerini ve sisteme etkilerini sıralamalıdır.

Genel Mimari Kalitesi



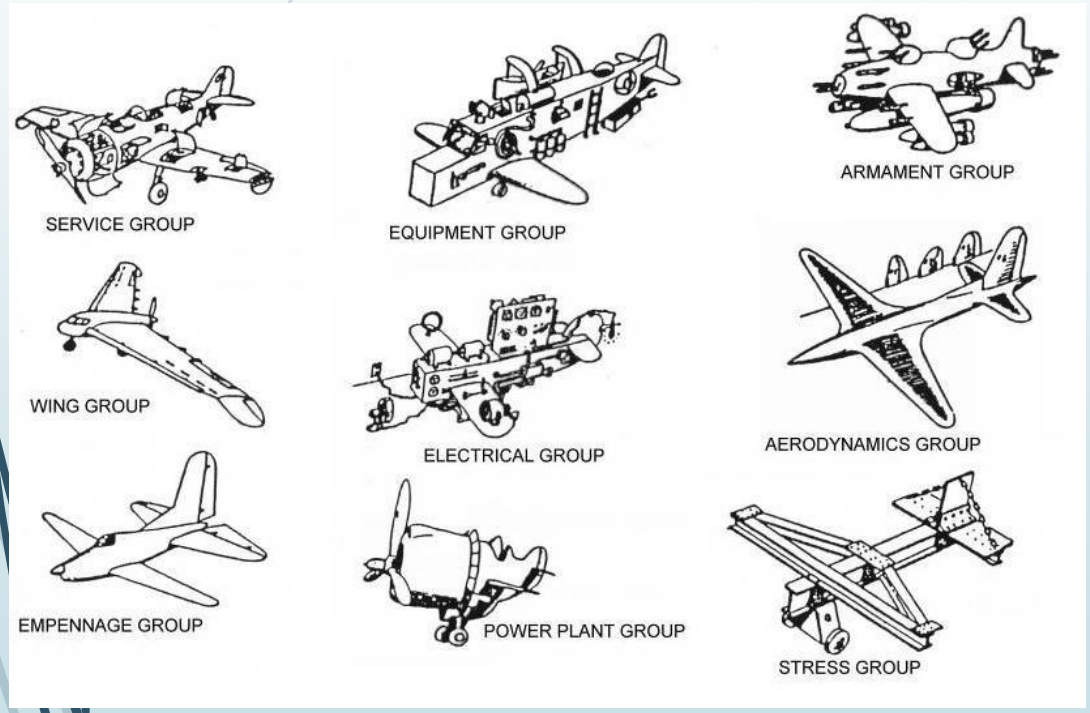
- Genel anlamda, iyi tasarlanmış bir mimariyi sınıfların ayrıntılı bir biçimde ifade edilmesinin yanısıra
- Sıınıflarda saklanan bilgilerin ve çeşitli dizayn alternatiflerinin nasıl seçildiğinin tartışılması karakterize eder.
- Çoğu kaynağa göre büyük sistemlerin ana problemi konseptin ve mimari anlamda bütünlüğün sağlanamamasıdır.
- Mimariye baktığınızda hemen ilk bakışta problemi güzel ve akıcı bir şekilde çözdüğünü hissetmelisiniz (**6. his**)

Genel Mimari Kalitesi



- Mimari hedefleri açıkça ifade edilmelidir.
- Hedefi performans olan bir sistemle, hedefi moduler ve değiştirilebilir olan sistemlerin mimari gereksinimi doğal olarak farklı olacaktır.
- Mimari ana kararları içeren ifadeler içermelidir.
- İyi bir mimari makine ve programlama dilinden büyük ölçüde bağımsızdır. (**Eğer proje belli bir hedef makine ve dile göre dizayn edilmiyorsa**)

Genel Mimari Kalitesi



- Mimari sınıfları ve yapıları ne fazla ve ne de az tanımlamalıdır.
- Mimarinin hiçbir parçası gerektiğinden fazla detaylandırılmamalıdır.
- Dizayn takımı, mimariyi bir bütün olarak ele almalı ve her parçasına aynı özeni göstermelidirler.
- Mimari her gereksinimi yeteri ölçüde abartmadan karşılamalıdır.
- Mimari risk alanlarını ifade etmeli ve neden riskli olduklarını belirtmelidir.
- Ve son olarak mimari anlaşılması zor ve karmaşık şeyleri içermemelidir.

Teşekkürler