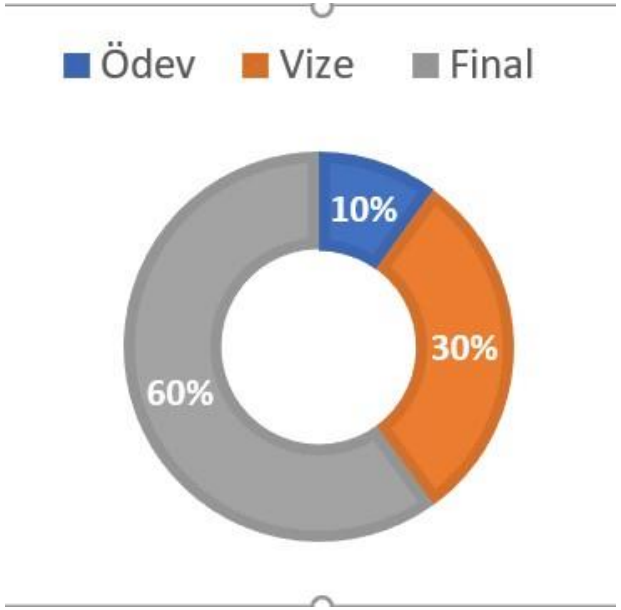
Yazılım Mühendisliği
1906003082015

Dr. Öğr. Üy. Önder EYECİOĞLU
Bilgisayar Mühendisliği

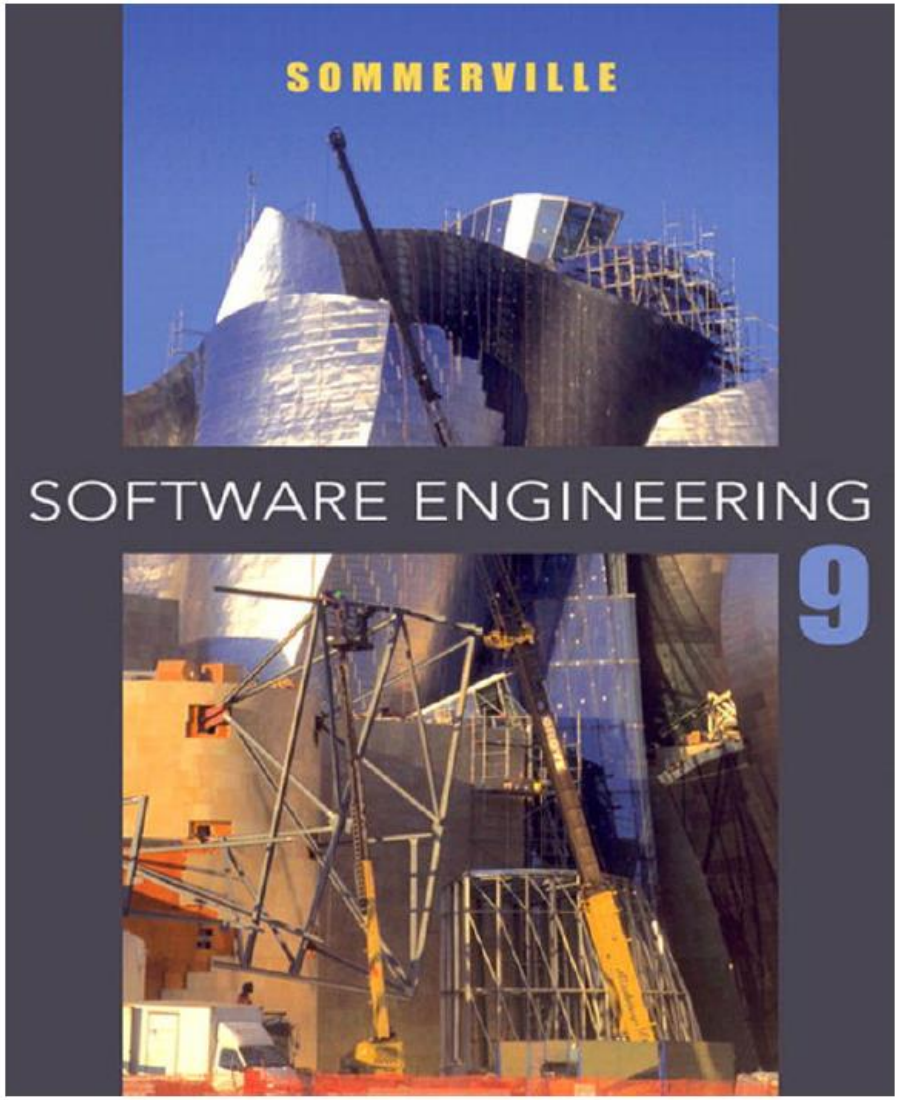
Giriş

Ders Günü ve Saati:
Salı: 09:15-13:00
Devam zorunluluğu %70



HAFTA	KONULAR
Hafta 1	Yazılım Mühendisliğine Giriş
Hafta 2	Yazılım Geliştirme Süreç Modelleri
Hafta 3	Yazılım Gereksinim Mühendisliği
Hafta 4	Yazılım Mimarisi
Hafta 5	Nesneye Yönelik Analiz ve Tasarım
Hafta 6	Laboratuar Çalışması: UML Modelleme Araçları
Hafta 7	Yazılım Test Teknikleri
Hafta 8	Ara Sınav
Hafta 9	Yazılım Kalite Yönetimi
Hafta 10	Yazılım Bakımı - Yeniden Kullanımı ve Konfigürasyon Yönetimi
Hafta 11	Yazılım Proje Yönetimi (Yazılım Ölçümü ve Yazılım Proje Maliyet Tahmin Yöntemleri)
Hafta 12	Yazılım Proje Yönetimi (Yazılım Risk Yönetimi)
Hafta 13	Çevik Yazılım Geliştirme Süreç Modelleri
Hafta 14	Yazılım Süreci İyileştirme, Yeterlilik Modeli (CMM)

Kaynaklar



11.
Yazılım teslim sonrası kalite

Giriş

BÖLÜM HEDEFLERİ

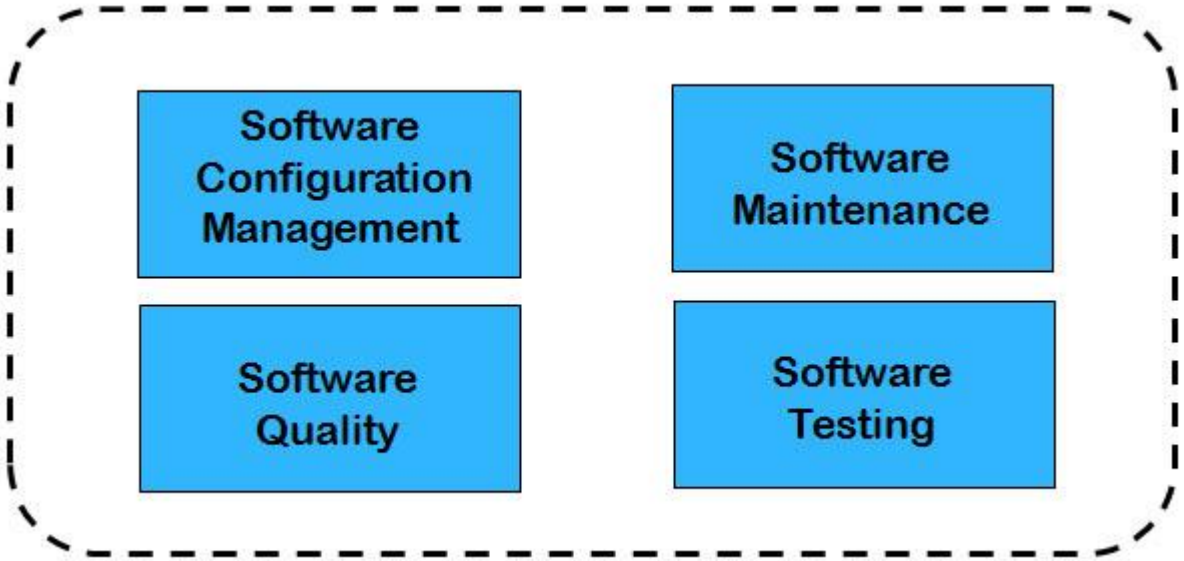
- Test spesifikasyonu belgesi içeriğini inceleme
- Test Planı içeriğini öğrenme
- Test senaryolarını hazırlayabilmek
- Yazılım bakımı (software maintenance) kavramını anlama,
- Yazılım mühendisliğinde konfigürasyon yönetiminin yeri ve önemini anlama,
- Yazılım değişim kontrolü ve versiyon kontrolü yollarını inceleme

13. HAFTA İÇERİĞİ

- Yazılım Teslim Sonrası Kalite Sağlama
  - Yazılım Bakımı
  - Yazılım Konfigürasyon Yönetimi
  - Bakım Maliyetlerinin Azaltılması

1. Yazılım Teslim Sonrası Kalite Sağlama

- Yazılım sisteminin geliştirilmesi, yazılım ürününün müşteriye teslimi ve kullanılmaya başlanması ile tamamlanmış olmaktadır. Kullanım süresinde de bakımı ve korunması, onarılması ve geliştirilmesi gerekmektedir.



1.1. Yazılım Bakımı

- Yazılımın bakımı ve onarımı (software maintenance); sonradan görülen hataların düzeltilmesi, yazılımın iyileştirilmesi-uyarlanması ve geliştirilmesi şeklindedir.
- Geniş bir programın kullanımı sırasında, sınama aşamasında bulunamamış olan çeşitli işlem, yetenek ve tasarım hataları ortaya çıkabilmektedir. Bu hatalar, işlem sırasında kilitlenme, kesilme, yavaş çalışma, anormal işleme vb. olaylarla kendini göstermektedir.

1.1. Yazılım Bakımı

- Yazılımın yeteneğini iyileştirmek ve kullanımını kolaylaştırmak üzere, programa bazı eklemeler gerekebilmektedir.
- Donanımın, işletim sisteminin ya da verinin değiştirilmesi hallerinde de yazılımın yeni ortama uydurulması yoluna gidilmektedir.
- Bu durumda, birkaç modül üzerinde değişiklik yapmak gerekmektedir.

1.1. Yazılım Bakımı

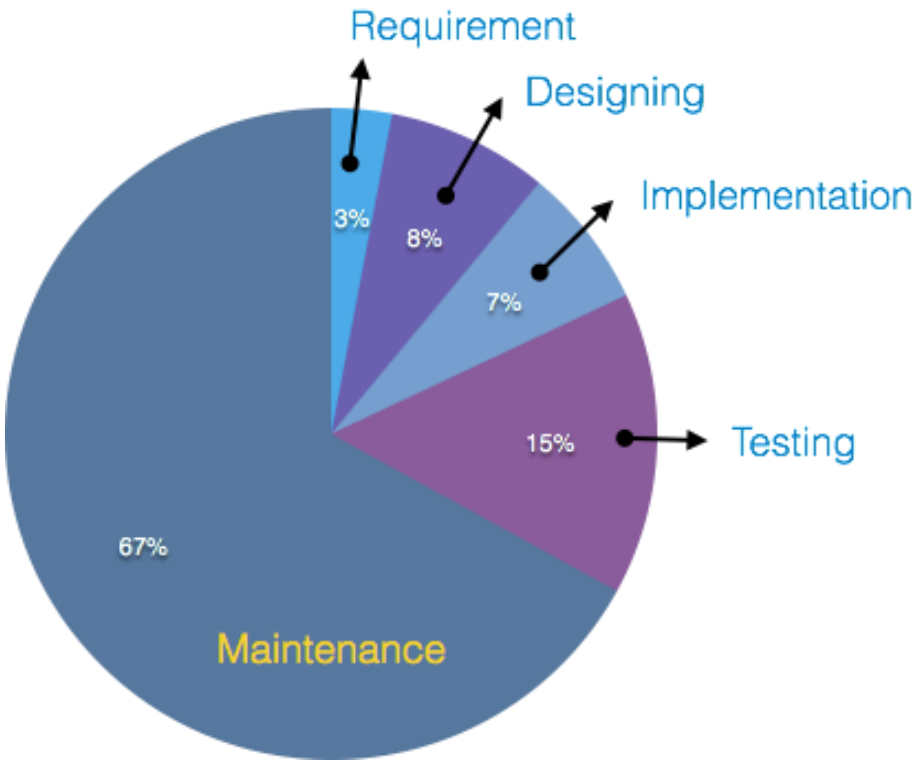
- Geniş bir programın kullanımı sırasında, sınama aşamasında bulunamamış olan çeşitli işlem, yetenek ve tasarım hataları ortaya çıkabilmektedir.
- Bu hatalar, işlem sırasında kilitlenme, kesilme, yavaş çalışma, anormal işleme vb. olaylarla kendini göstermektedir.
- Ayrıca, günün koşullarına göre yeterli bulunmayan bir yazılımın; gereksinim analizi, tasarım, tamamlama ve sınama basamakları halinde yeniden geliştirilmesi ve böylece onarılması da söz konusudur.

1.1. Yazılım Bakımı

- Geliştirilmesi tamamlanan ve müşteriye teslim edilen yazılım, kullanıcı tarafından, yazılım ortamının veya uygulama alanının değişmesi, yeni gereksinimlerin belirmesi, yazılımdaki mevcut kusur ve eksikliklerin giderilmesine yönelik değişim isteklerinden dolayı, sürekli değişim baskısı altında tutulmaktadır.
- Bu noktadan itibaren, yazılımın bakım sürecine ilişkin çeşitli mühendislik faaliyetlerinden söz edilmektedir.

1.1. Yazılım Bakımı

- Yazılım bakımı, yazılım faaliyetlerinin bütünsellik içinde, maliyet etkin olarak gereksinimleri karşılaması ve desteğin sağlanması olarak tanımlanabilir. Yazılım bakımına olan gereksinim, yazılım kullanıldığı sürece devam etmektedir.
- Bakım işleri yanlızca hataları düzeltmek olmayıp ürünün tesliminden sonra yapılması gereken çeşitli işleri kapsar.

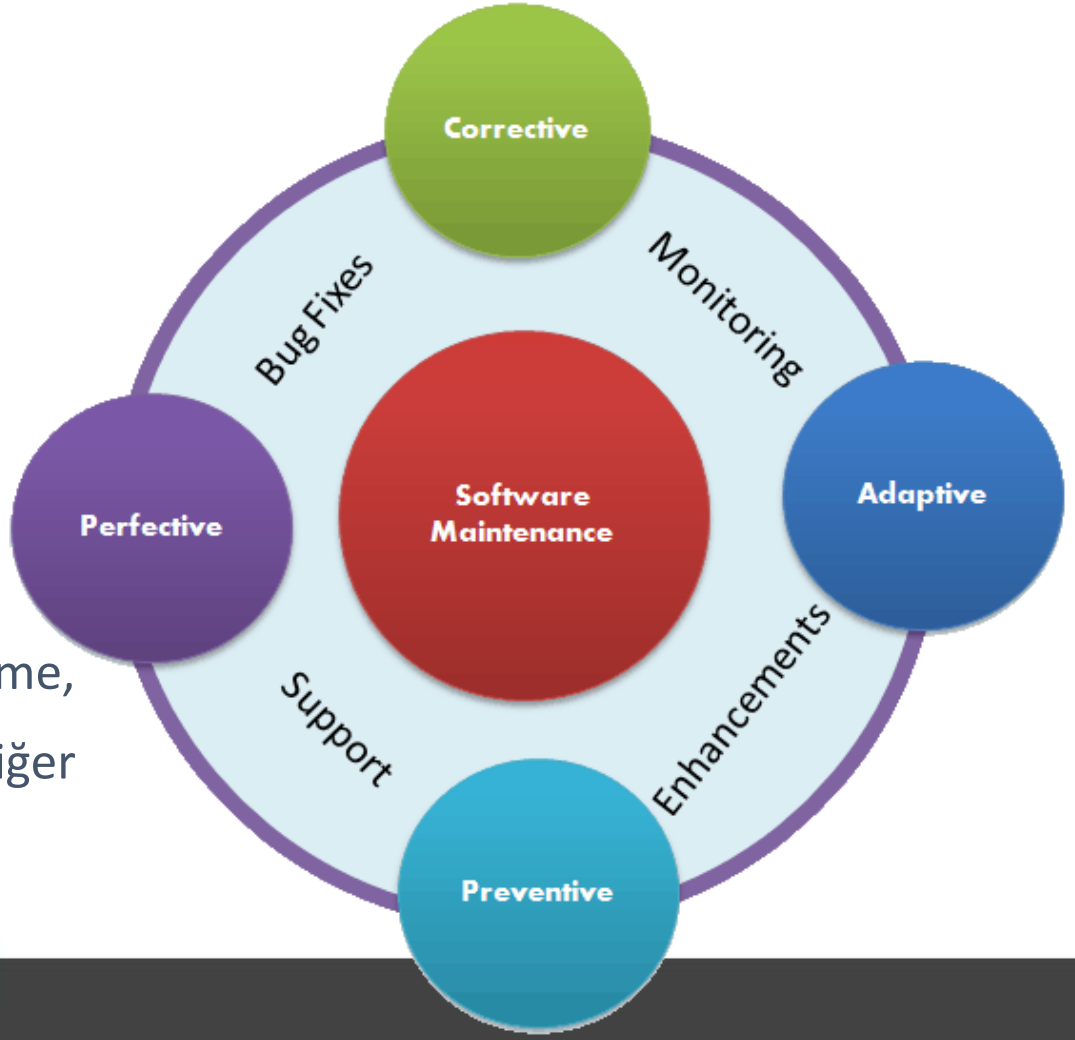


1.1. Yazılım Bakımı

Bakım Türleri

- Düzeltici bakım
- Uyarlayıcı bakım
- İyileştirici bakım
- Önleyici bakım

Yazılımın bakımı konusundaki işlerin %21'inin hata düzeltme, %25'inin iyileştirme, %50'sinin uyarlama ve %4'ünün diğer durumlarda olduğu bildirilmektedir.



1.1. Yazılım Bakımı

Düzeltici (Corrective) Bakım

- Tespit edilen hataların giderilmesi işlemidir.
- Bir kısım yazılım kusurları kullanım sırasında ortaya çıkar. Bu hatalar giderilebilmesi için geliştiriciye bildirilmesi gerekir.
- Kusurun kaynağını araştırıp hatayı ortadan kaldırmaya yönelik çalışmalara düzeltici bakım adı verilir
- Kodlama hatalarını düzeltmek genelde az maliyetlidir. Tasarımdan kaynaklı hataların giderilmesi ise bazı sistem bileşenlerinin baştan yazılmasını vb. gerektirebilir ve nispeten yüksek maliyetlidir.
- Bu tür hatalar giderilerek yeni sürümler ortaya çıkartılır

1.1. Yazılım Bakımı

Uyarlayıcı (Adaptif) bakım

- Yazılımın yeni bir çalışma ortamına uyarlanmasıdır.
- Son yıllardaki gelişmelere bakılarak bir yazılımın ortalama ömrünün 10 yıl olduğu tesbit edilmiştir. Fakat sahip olduğu donanım birimlerinin yürürlükte kalma süresinin 1 yada 2 yıl olmasından dolayı yazılımın yeni şartlara ayak uydurulması germektedir.
- Bu bir donanım platformu değişikliği olabileceği gibi (32 bitten 64 bite geçiş gibi) farklı bir işletim sistemine uyarlama şeklinde de olabilir (kodun Windows'dan Linux'a taşınması gibi).
- Veritabanı sistemi değişikliği de bu türden bir bakım olarak görülebilir (MS SQL Server bağımlı kodların Oracle'a uyarlanması gibi).

1.1. Yazılım Bakımı

İyileştirici (Perfective) Bakım

- Sisteme yeni işlev ve özelliklerin eklenmesi, performansın arttırılması gibi bakım çalışmalarıdır.
- Örneğin yazmış olduğunuz bir kod parçası ilgili işlevi 10 sn yapıyor . Bu kod parçasını geliştirerek yapılan işlevi 5 sn düşürülüyorsa bu olaya iyileştirici bakım denir.
- Genellikle bir sisteme çalışmaya başladıktan sonra yeni bir işlev eklemek, aynı işlevin henüz geliştirme sürecindeyken eklenmesine göre çok daha maliyetlidir.

1.1. Yazılım Bakımı

Önleyici (preventive) Bakım

- Yazılımın gelecekte uygulanabilecek değişikliklere daha iyi adapte olması için bakıla bilirliği ve güvenile bilirliği artırabilmek için alınabilecek tedbir niteliğindeki işlemlere önleyici bakım denir
- Örneğin sık sık değişim yapılan modülün tasarımını daha esnek hale getirmek

1.1. Yazılım Bakımı

Bakım Aşamaları

Bakım evresinde bulunan bir yazılım için bir bakım işi ortaya çıktığında yazılım geliştiricisi tarafından standart bir süreç izlenmelidir. Bakım aşamasını aşağıdaki gibi özetleyebiliriz.

- İsterler çözümlenmesi
- Tasarım
- Gerçekleştirim
- Test
- Teslim

1.2. Yazılım Konfigürasyon Yönetimi

- Yazılım mühendisliğinin ürünleri, programlar, belgeler ve veri yapılarıdır. Bu ürünlere ait bütün maddelere topluca, yazılım konfigürasyonu denir. Yazılım Konfigürasyon Maddesi (Software Configuration Item), ise, YKY işlemlerinin uygulandığı yazılım modülüdür.
- Konfigürasyon yönetimi (KY), geliştirilen bir sistem üzerinde meydana gelen değişikliğin kontrol edilmesi ve belgelenmesi sürecidir. Bu, genel değişim yönetimi yaklaşımının bir parçasıdır. Bir yazılım ortamında Konfigürasyon yönetimi mutlak bir zorunluluktur. Konfigürasyon yönetiminde; kim, ne, ne zaman ve neden gibi sorular sorularak değişiklikler kontrol edilmeye çalışılır. Etkili konfigürasyon yönetimi, bir proje için aşağıdaki temel yararları sağlar:

1.2. Yazılım Konfigürasyon Yönetimi

- Karışıklıkları azaltır ve düzeni kurar.
- Ürün bütünlüğünü korumak için gerekli faaliyetleri düzenler.
- Yaşam döngüsü maliyetlerini azaltır.
- İstikrarlı bir çalışma ortamı sağlar.
- Gereksinimleri ile uyumlu bir proje geliştirilmesini sağlar.
- Standartlara uygunluğu artırır.

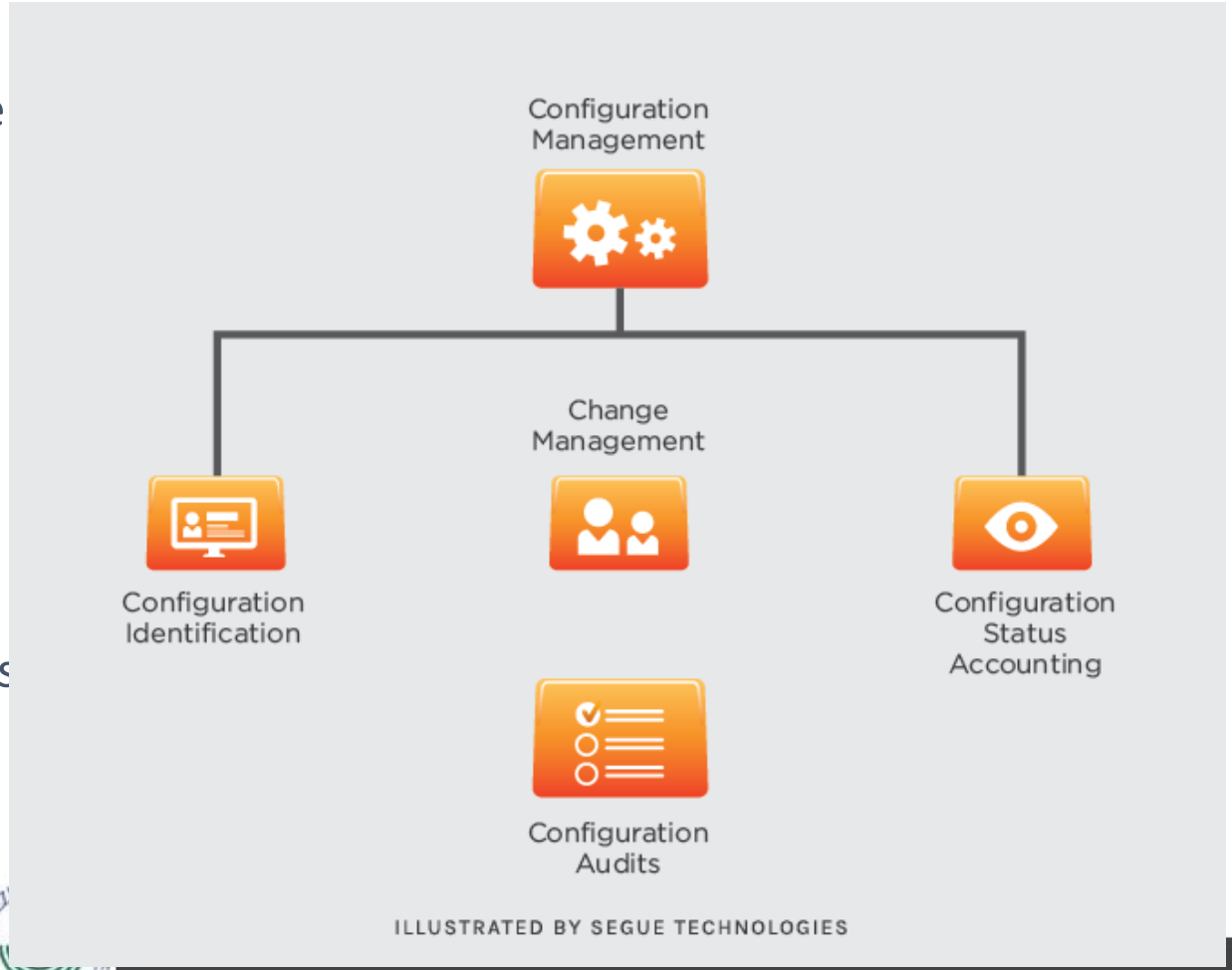
1.2. Yazılım Konfigürasyon Yönetimi

- Konfigürasyon yönetimi, değişim kontrol programının önemli bir unsuru olmakla birlikte çok yönlü bir disiplindir. Etkili Konfigürasyon yönetimi programının kurulması için Konfigürasyon yönetimi fonksiyonlarının ve genel Konfigürasyon yönetimi sürecinin anlaşılması gerekir.

1.2. Yazılım Konfigürasyon Yönetimi

Konfigürasyon yönetimi faaliyetleri temel olarak KB'lerin konfigürasyon temellerinin tanımlanması ve kontrol altında tutulması amacıyla yürütülmektedir. Konfigürasyon yönetimi faaliyetleri;

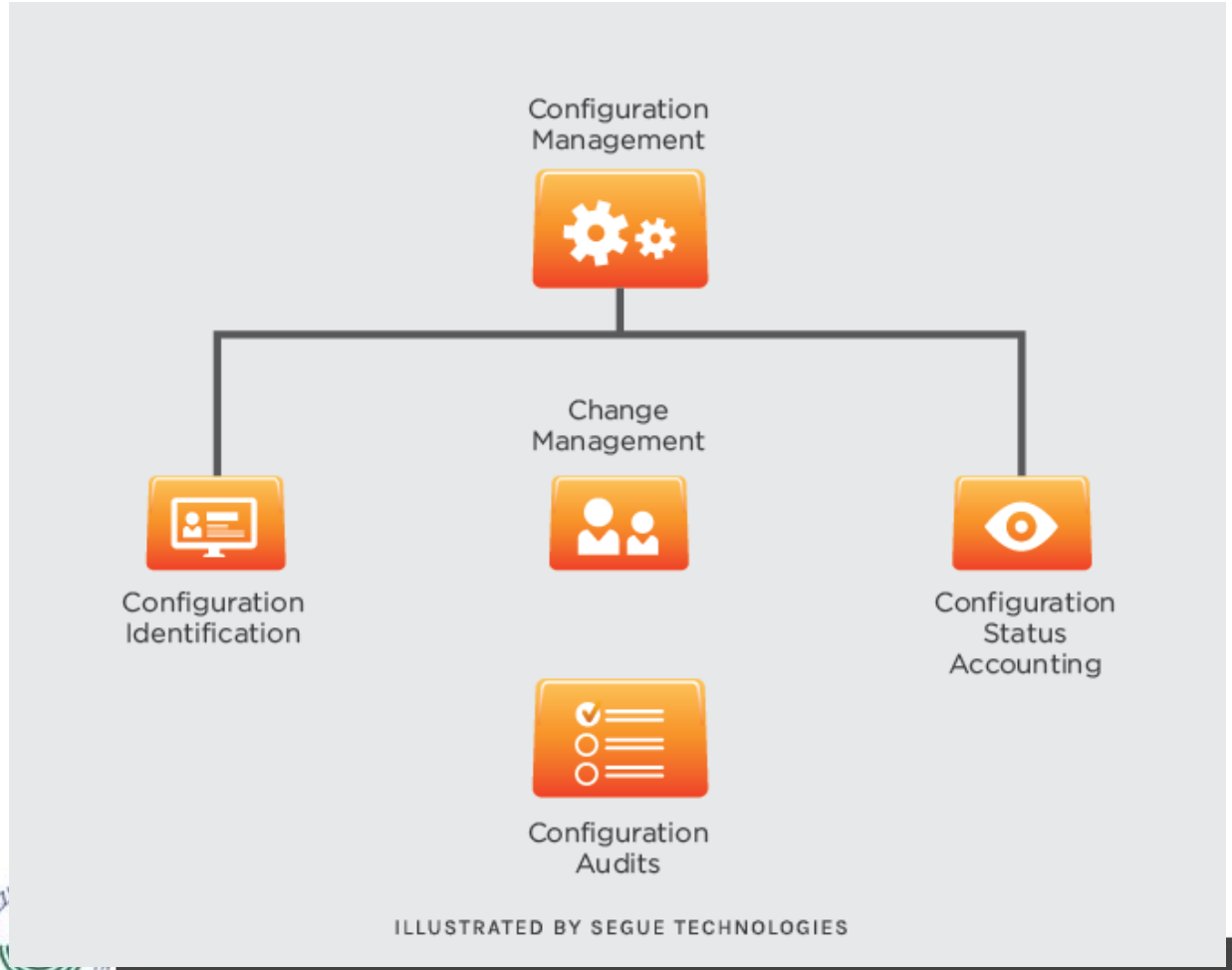
- Konfigürasyon Tanımlama (Configuration Identification)
- Konfigürasyon Kontrol (Configuration Control)
- Konfigürasyon Tetkikleri (Configuration Audit)
- Konfigürasyon Durum Takibi (Configuration Status Accounting)



1.2. Yazılım Konfigürasyon Yönetimi

Konfigürasyon Tanımlama (Configuration Identification)

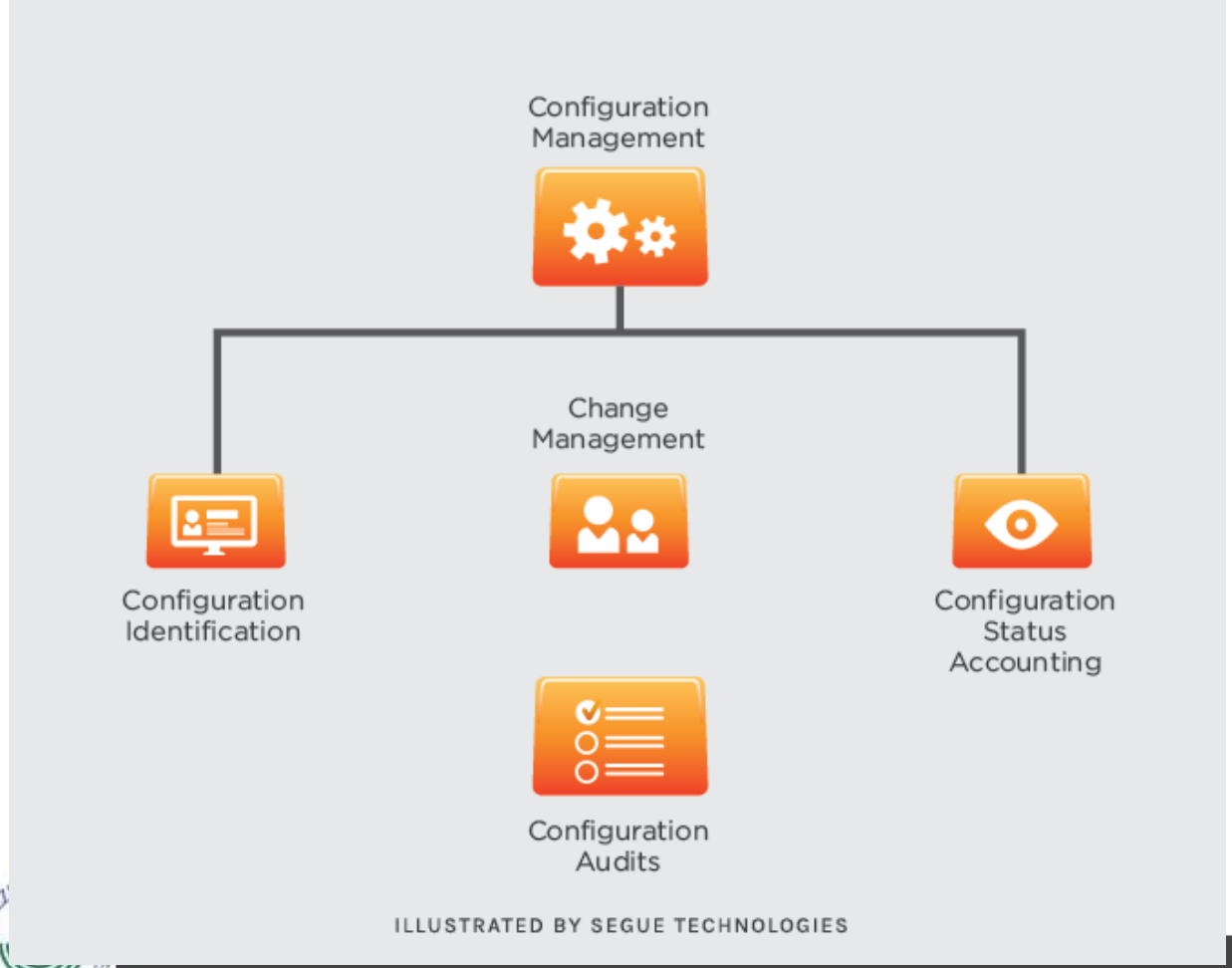
- Bu fonksiyon yapılandırılmaya ihtiyacı olan donanım, yazılım ve belgelerden oluşan kontrol edilmesi gereken bu öğeleri tanımlar.
- Bu ögeler; yazılım kodu, belge, tasarım, veri, çizim, derleyiciler, bağlayıcılar ve donanım bileşenlerinden oluşur.



1.2. Yazılım Konfigürasyon Yönetimi

Konfigürasyon Kontrol (Configuration Control)

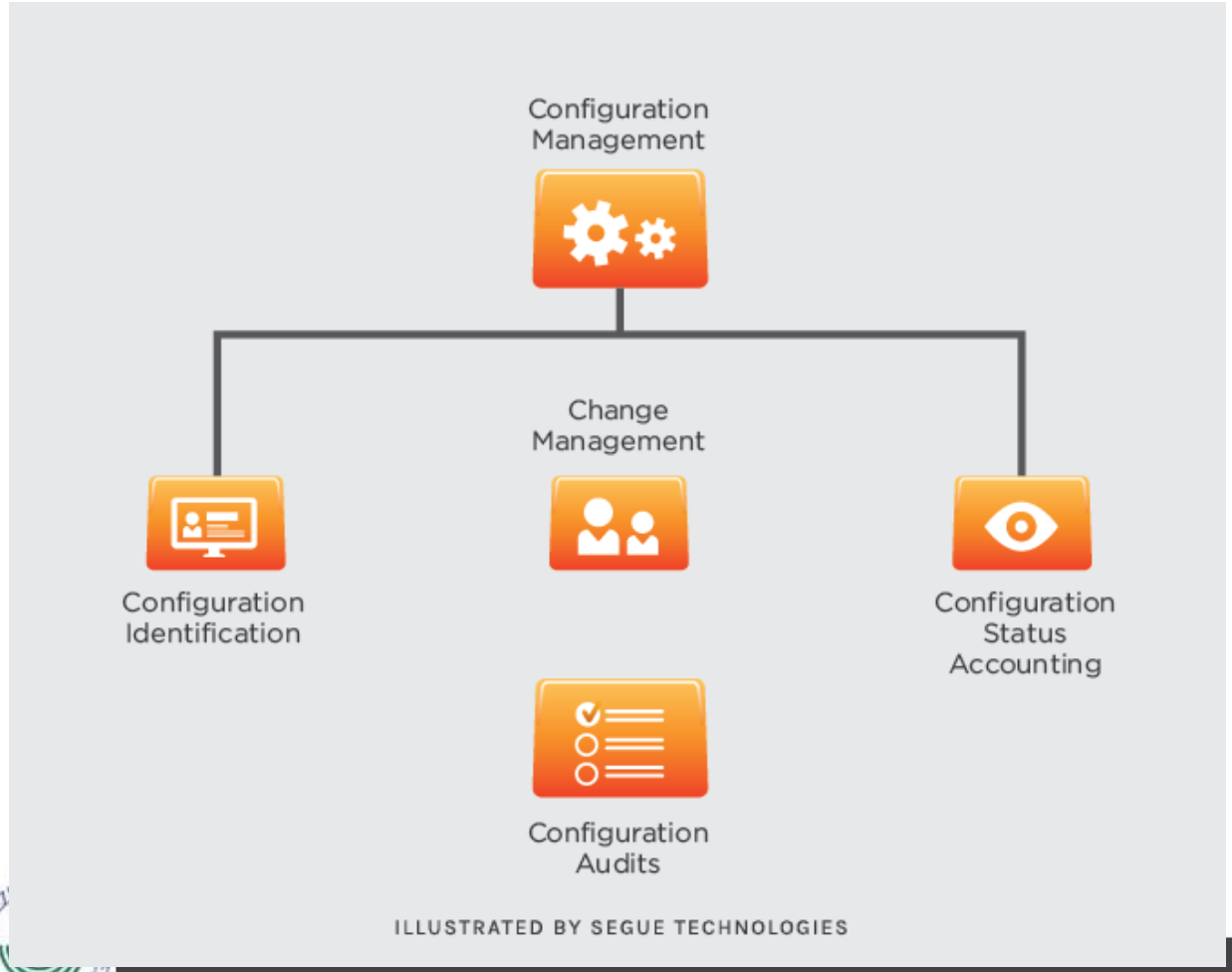
- Bu fonksiyon; değişiklik istemi, değişikliklerin değerlendirilmesi, yayınlanması ve izlenmesi ve son olarak değişikliklerin uygulanması için bazı prosedürleri belirler.



1.2. Yazılım Konfigürasyon Yönetimi

Konfigürasyon Tetkikleri (Configuration Audit)

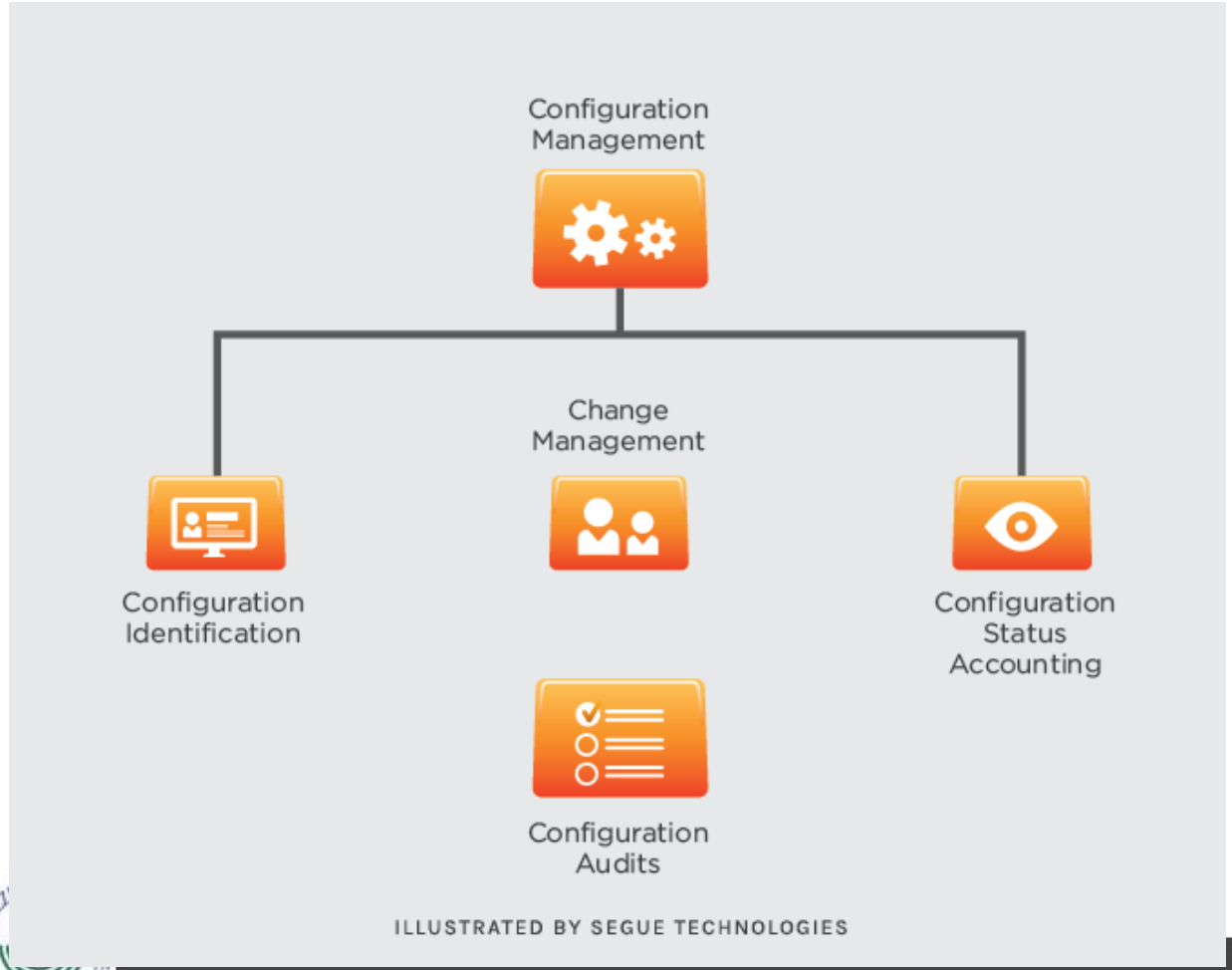
- Etkili bir konfigürasyon yönetiminde, yapılandırmanın düzenli olarak değerlendirilmesi gerekir. Bu fonksiyon ile fiziksel ve fonksiyonel yapılandırmalar, belgelenmiş yapılandırmalar ile karşılaştırılır. Denetleme fonksiyonu, yazılım ürününün gereksinimlere, standartlara ve sözleşmeye uygun olarak hazırlandığını doğrular.



1.2. Yazılım Konfigürasyon Yönetimi

Konfigürasyon Durum Takibi (Configuration Status Accounting)

- Konfigürasyon yönetiminin belgeleme fonksiyonudur.Bu fonksiyonun amacı; yapılan değişiklikleri resmi kayıtlar altında tutmak ve yapılandırma durumunu düzenli olarak raporlamaktır..



1.2. Yazılım Konfigürasyon Yönetimi

Yazılım Konfigürasyon Yönetim Prosesi

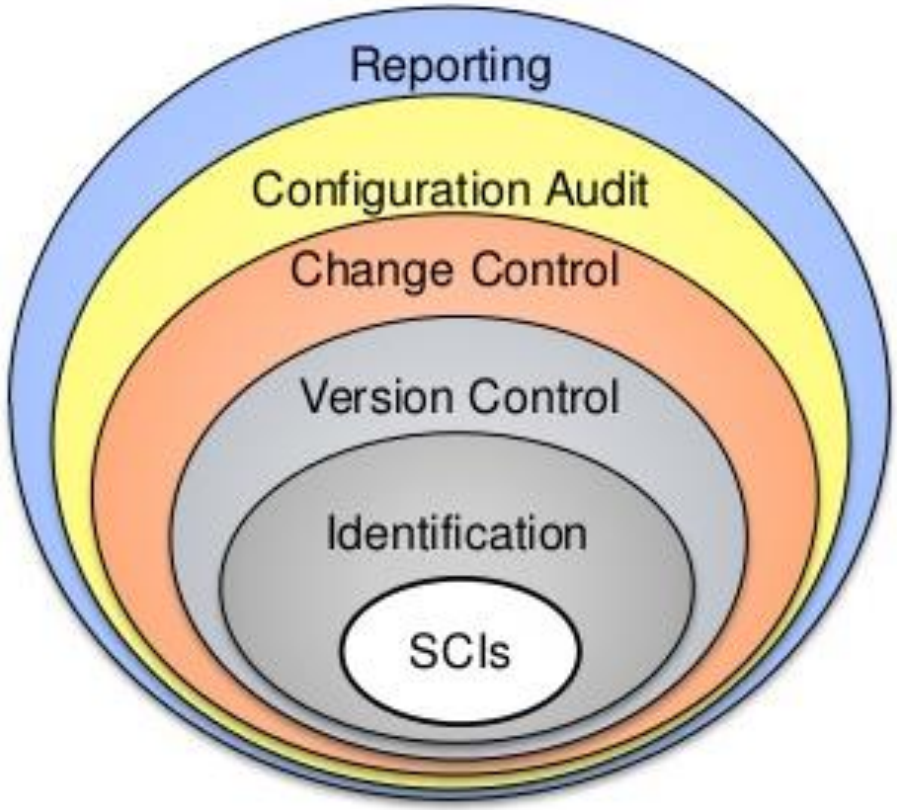
Yazılım Konfigürasyon Yönetimi prosesi dört ana amacı ödev olarak ele alır.

- Ortak olarak kullanılan tüm öğeler yazılım konfigürasyonunda tanımlanır , belirlenir.
- Bu öğelerden biri veya birçoğunun üzerinde yapılacak olan değişiklikleri yönetmeyi hedefler.
- Yazılım uygulamalarının değişik versiyonlarının oluşturulmasını kolaylaştırmayı amaçlar.
- Zamanında kalite güvencesi ile yazılımların konfigürasyonlarının bakımlarını sağlar.

1.2. Yazılım Konfigürasyon Yönetimi

Yazılım Konfigürasyon Yönetim Prosesi

Yazılım Konfigürasyon Yönetimi görevleri açısından tanımlayabilmek için beş adet tanım kavramı (Belirleme , versiyon denetimi , değişiklik kontrolü, konfigürasyon denetleme ve raporlama / izleme) ile takip eden katmanlı bir sema ile açıklamamız faydalı olur.

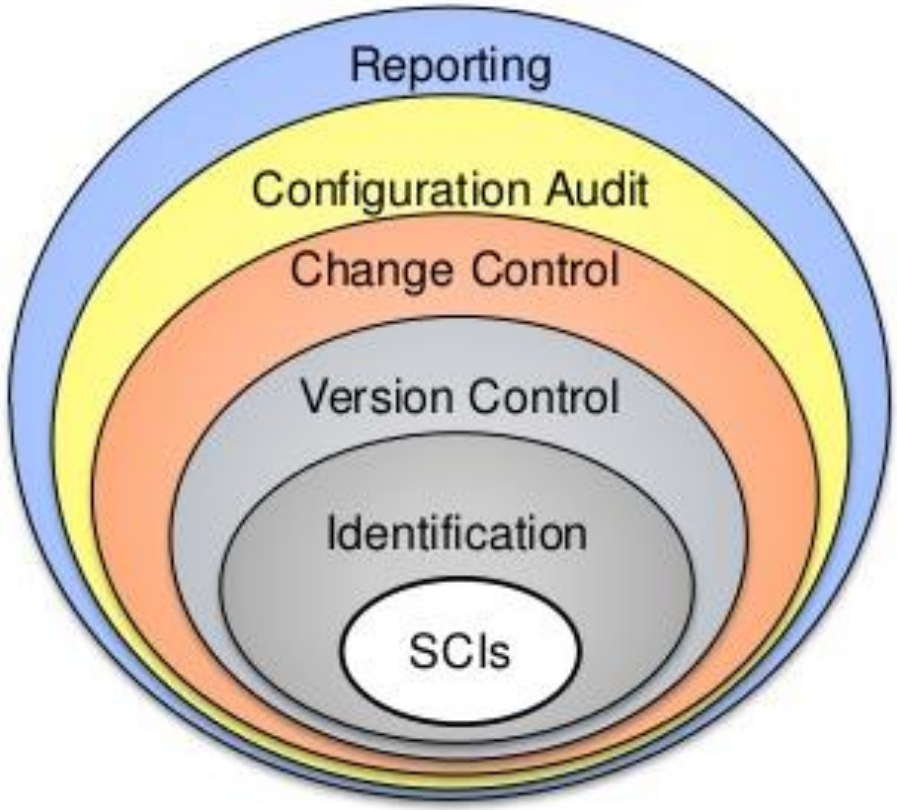


1.2. Yazılım Konfigürasyon Yönetimi

Yazılım Konfigürasyon Yönetim Prosesi

Yazılım Konfigürasyon Yönetimi prosesleri ortak merkezlidirler. Yazılım konfigürasyon öğeleri içten dışarıya doğru bir ilişki ile birbirlerine bağlıdırlar.

Bu öğelerden birisi katmanlar arasında hareket edince yazılım konfigürasyon yönetimi kavramları katmanlar arasında bir disiplin, uygulama prensibi olarak karsımıza çıkar.

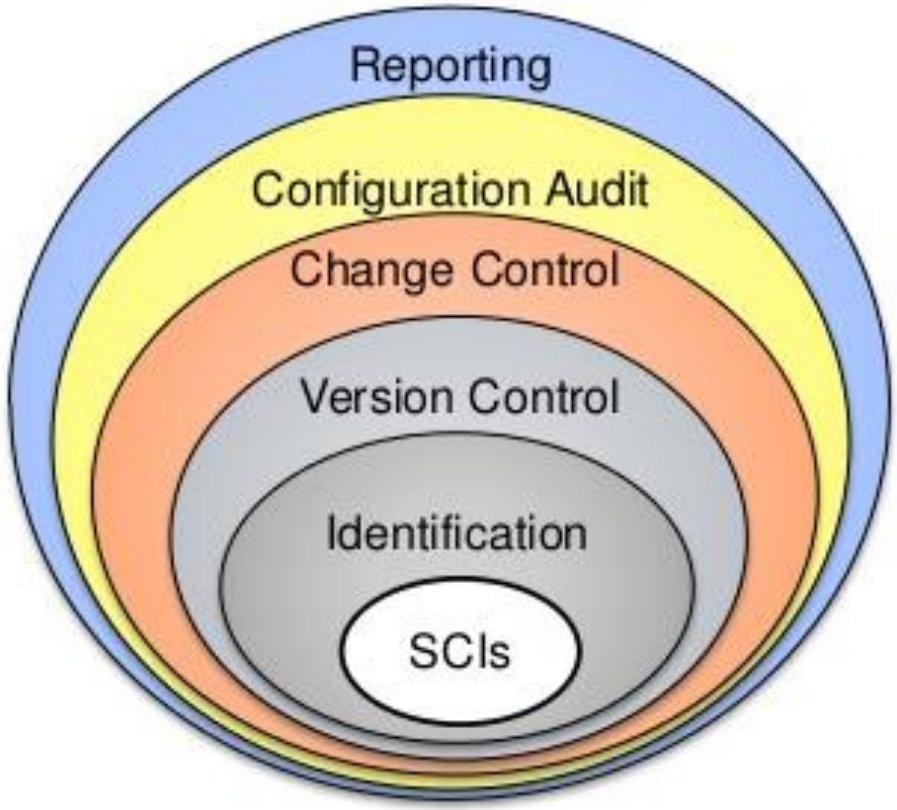


1.2. Yazılım Konfigürasyon Yönetimi

Yazılım Konfigürasyon Yönetim Prosesi

Her bir yazılım konfigürasyon öğesi yazılımın belirli bir versiyonuna belirleyici olan bir rakamsal versiyon numarası ile atanır. «Software V.1.2.3»

Böylece birçok öğe ile farklı yazılım versiyonlarını aynı yazılıma ait birer farklı versiyon kümesi olarak görebiliriz.



1.2. Yazılım Konfigürasyon Yönetimi

Yazılım Sistemlerindeki Nesnelerin Belirlenmesi

Yazılım konfigürasyon öğelerini yönetmek ve kontrol etmek için nesneye yönelik yaklaşım kullanılabilir. İki çeşit nesne tipi belirlemesi yapılır. Bunu temel nesneler ve küme nesneler olarak isimlendirilir.

- Temel nesne , yazılım muhendisi tarafından analiz , tasarım , kodlama ve test sırasında olusturulan birim bilgiden elde edilir.
- Küme nesne ise bu temel nesnelerin toplanması veya diğer küme nesnelerin bir araya gelmesi ile oluşur.

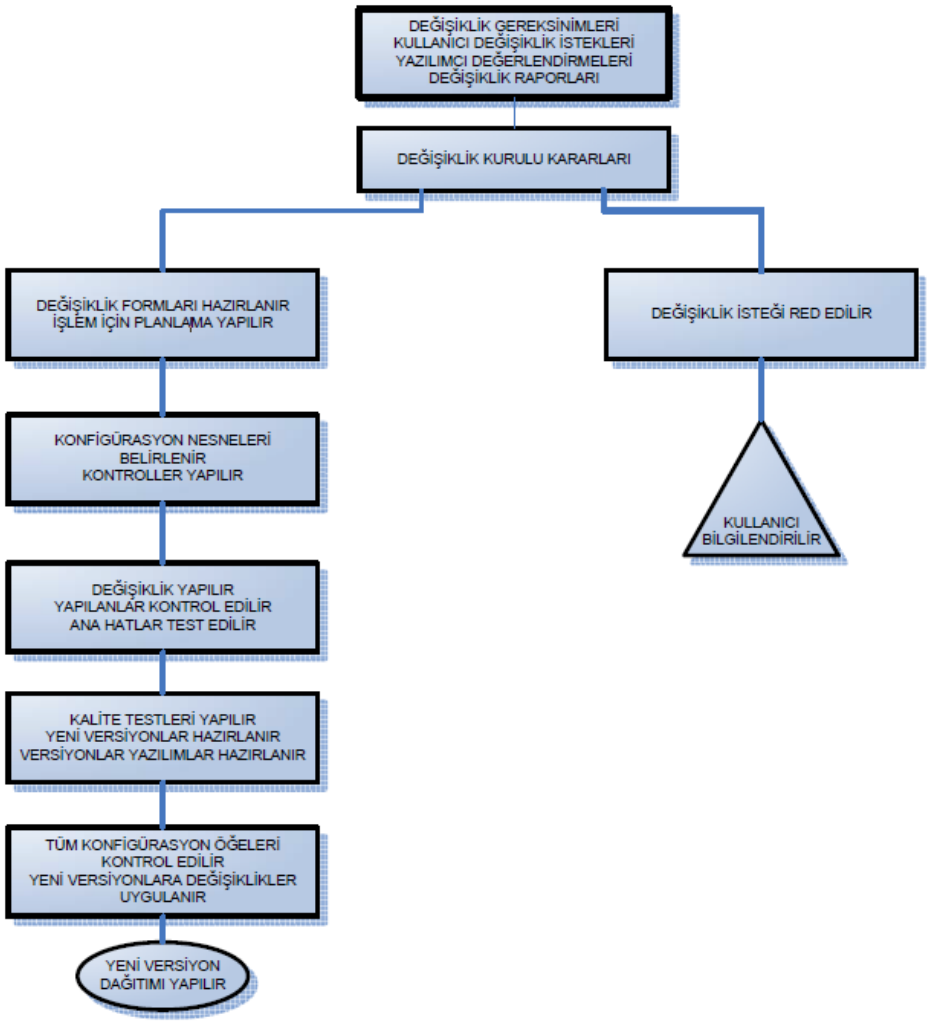
1.2. Yazılım Konfigürasyon Yönetimi

Versiyon Denetimi

Yazılım gelistirme sürecinde birçok aracın ve yazılımın yönetilebilmesi amacı ile üretilen ürünlere ait kodların bir disiplin altında takip edilebilmesi çalışmasıdır. Süreç içinde yasayan bir varlık olan yazılıma ait gelen değişiklik istekleri değerlendirme kurullarından geçtikçe, oluşan sorunlar tespit edilip düzeltildikçe yasayan yazılıma ekler, çıkartmalar , uyarlamalar ve düzenlemeler yapılır. Bunu düzenleyen yapıya versiyon denetimi diyebiliriz.

1.2. Yazılım Konfigürasyon Yönetimi

Değisiklik Yönetim ve Kontrolü



1.2. Yazılım Konfigürasyon Yönetimi

Konfigürasyon Denetimi

Bundan önceki adımlarda yazılımcının yapmış oldukları gelen isteklerin biraz belirsizlik ve kaos ortamlarından kurtulup , düzen içinde çalışmalarını yapmasını sağlar.

Bununla beraber etkili bir denetiminde kesinleşen değişiklik isteklerinde uygulaması gerektirdiği çok açıktır.

Bu isteklerin uygun bir biçimde nasıl uyarlanabileceğinin yanıtı ise 1) resmi teknik değerlendirmelerin 2) yazılım konfigürasyon denetimlerinin tam yapılmasındadır.

1.2. Yazılım Konfigürasyon Yönetimi

Raporlama

Nelerin yapıldığına , kimin yaptığına , ne zaman yapıldığına ve nelerin etkilendiğine verdiğimiz yanıtların tamamı bir sürecin raporlama düzeyini tanımlar. Konfigürasyon durum raporlaması bu bilgi akısının yapılmasını bir belgeleme düzenine sokmasını sağlar. Bu yapmış olduğumuz değişiklik isteğinden bunların versiyonlanarak yapılmasına , test edilmesine, onaylanmasına ortamlara taşınmasına kadar olan tüm süreçte belgelenmesine ve dolayısı ile raporlanmasına yardımcı olur. Bu raporlardan elde edilecek sonuçların sisteme geri beslenmesi ile de sistemin kendini iyileştirmesi , yapılan faaliyetlerin hem mühendislik hemde maliyet açısından proje incelenmesine alınmasına neden olur.

1.2. Yazılım Konfigürasyon Yönetimi

Standartlar

- YKY sürecinde herhangi bir işlemin yada ürünün değerlendirilmesi için bir standart gerekmektedir. Örneğin YKY planı standardı, YKY planının ne içermesi gerektiğini tanımlar YKY ile ilgili standartlara örnek,
- IEEE 828-2012 - IEEE Standard for Configuration Management in Systems and Software Engineering
- IEEE 1042-1987 - IEEE Guide to Software Configuration Management
- ISO 9004 Quality Management & Quality System Elements Part 6

IEEE 828-2012 - IEEE Standard for Configuration Management in Systems and Software Engineering

Standart Ayrıntıları

- Bu standart, sistem ve yazılım mühendisliğindeki Konfigürasyon Yönetimi (CM) süreçleri için minimum gereksinimleri belirler. Bu standardın uygulanması herhangi bir form, sınıf veya yazılım veya sistem türü için geçerlidir. Standardın bu revizyonu, yapılandırma öğelerinin tanımlanması ve alınması, değişikliklerin kontrol edilmesi, konfigürasyon öğelerinin durumunu raporlamanın yanı sıra yazılım oluşturma ve sürüm mühendisliği dahil olmak üzere CM'yi açıklamak için önceki sürümü genişletir. Selefi yalnızca bir yazılım konfigürasyon yönetimi planının içeriğini tanımladı. Bu standart, hangi CM faaliyetlerinin gerçekleştirileceğini, yaşam döngüsünde ne zaman olacağını ve hangi planlama ve kaynakların gerekli olduğunu ele almaktadır. Ayrıca bir CM Planı için içerik alanlarını da açıklar. Standart, ISO / IEC / IEEE 12207: 2008 ve ISO / IEC / IEEE 15288: 2008'i destekler ve ISO / IEC / IEEE Std 24765'deki terminolojiye ve IEEE Std 15939TM'nin bilgi maddesi gereksinimlerine bağlıdır.

Board Approval	2012-02-06
History	Published Date:2012-03-16

IEEE 1042-1987 - IEEE Guide to Software Configuration Management

Standart Ayrıntıları

- Yapılandırma yönetimi (CM) disiplinlerinin yazılım mühendisliği projelerinin yönetimine uygulanması anlatılmaktadır. Yazılım konfigürasyon yönetimi (SCM) aktivitelerini planlayanlar için bu kılavuz, göz önünde bulundurulması gereken çeşitli faktörler hakkında fikir vermektedir. SCM disiplinlerini uygulayan kullanıcılar için öneriler ve ayrıntılı plan örnekleri verilmiştir. Farklı türdeki bilgisayar programı geliştirme ve bakım faaliyetlerinin yönetimini planlamak için Yazılım Geliştirme Yönetim Planları için IEEE Standardı olan ANSI / IEEE Std 828-1983'ün nasıl kullanılabileceğinin yorumlanması.

Status	Withdrawn
Board Approval	1987-09-10
History	Published Date:1988-09-12 Reaffirmed:1993-12-02 Withdrawn Date:2000-03-06

1.3. Bakım Maliyetlerinin Azaltılması

- Bakım ve onarım giderini ve en aza indirmek için, yazılım ürününün "bakım ve onarıma elverişli" nitelikte oluşturulması gerekmektedir (maintainability).
- Bu nitelik; yazılımın kolay anlaşılabilir, düzeltilebilir, uyarlanabilir ve geliştirilebilir özelliklerde tasarlanmış olmasıyla sağlanabilmektedir. Bunun için de;
- Yetenekli ve deneyimli yazılım mühendisleri görevlendirmek
- Anlaşılabilir bir sistem yapısı ve kolay işletilebilir bir sistem tasarlamak
- Standart programlama dilleri, işletim sistemleri kullanmak ve belgeleri standart biçimde düzenlemek
- Test programlarından yararlanmak
- Tasarım aşamasında, hata bulma ve düzeltme kolaylıkları sağlamak gerekmektedir.

1.3. Bakım Maliyetlerinin Azaltılması

Yazılımın bakım ve onarıma elverişliği; yazılımın diğer kalite faktörlerinden olan,

- Sınama kolaylığı
- Basitlik
- Değiştirilebilirlik
- Taşınabilirlik
- Güvenirlik
- Esneklik özeliklerinin bir bileşkesi olarak ortaya çıkmaktadır.