Gerçek Zamanlı Gömülü Sistemlerde Yeniden Kullanılabilir ve Yapılandırılabilir Yazılımların Kaliteye Etkisi: Radar Projeleri Destek Kütüphaneleri

Deniz AKDUR[1,2], Çağatay ÖZDEMİR[1,3]

[1]ASELSAN A.Ş. , REHİS-TTD / Görev Yazılımları Müdürlüğü, Ankara
[2]Enformatik Enstitüsü Bilişim Sistemleri, [3]Elektrik Elektronik Mühendisliği, ODTÜ, Ankara

{denizakdur, cagatayo}@aselsan.com.tr

Özetçe. Günümüzde hızla gelişen ve değişen gömülü yazılım istekleri az zamanda çok iş ortaya çıkarma gereği doğurduğundan, "yeniden kullanılabilir" kütüphaneler tasarlayıp bunları en iyi şekilde yapılandırıp kullanmak yazılım geliştirme ve idamesinde büyük kolaylıklar sağlamaktadır. Kara, deniz ve hava platformlarında değişik gerek ve isteklere dinamik bir takvim içinde hızlı cevap verebilmek amacıyla oluşturulan bu "yeniden kullanılabilir" birimler, yapılandırılabilme ve taşınabilme özellikleri ile radar yazılım dünyasında hem maliyeti azaltmakta hem de daha kaliteli ürünler ortaya çıkarmaya yardımcı olmaktadır.

Bu bildiride ASELSAN Radar ve Elektronik Harp İstihbarat Sistemleri (REHİS) Görev Yazılımları Müdürlüğü (GYM) Radar Ekibinde kullanılan Destek Kütüphanelerinin neden, nasıl, nerelerde kullanıldığı ile yeniden kullanılabilirlik ve yapılandırılabilirlik özelliklerinin maliyet, zaman ve kalite yönünden kazandırdıkları aktarılacaktır.

Anahtar Kelimeler. Yeniden Kullanım, yapılandırılabilirlik, taşınabilirlik, güvenilirlik, kütüphane

1 Giriş

Yeniden Kullanım ortak özelliklere sahip sistemlerde yeniden kullanılacak varlıkların birçok kez kullanabilmesine imkân vererek geçmişte gerçekleştirilmiş olan yazılım geliştirme etkinliklerinde elde edilen varlıkların veya bilginin yeni sistemler geliştirmede kullanılmasını sağlar [1]. Yeniden kullanım yazılım geliştirme zamanını azalttığı gibi maliyeti düşürür [2], yeni sistemlerin tasarlanmasını kolaylaştırır [3].

Yazılım mühendisliğindeki tüm güncel yaklaşımlar, yeniden kullanımın artırılmasını hedeflemektedir [4]. Ancak, yazılımcıların programlarında kullandıkları her "kopyala ve yapıştır" eylemi yeniden kullanım kapsamında sayılmamaktadır [5]. Yeniden kullanım tarihine baktığımızda, ilk olarak fonksiyonlar bazında karşımıza çıkan en basit yeniden kullanım tekniği göze çarpmaktadır. Bu tip yeniden kullanımda mühendislerin bilgisine bağımlılık fazla olduğundan çok verimlilik

sağlanamamaktadır [2]. Bir sonraki yeniden kullanıma yönelik geleneksel yaklaşım, saf kodun yeniden kullanımı veya kütüphane gibi daha düzenli yapılar aracılığı ile yeniden kullanımdır. Kütüphane, uygulamaların bazı fonksiyonlara sahip olabilmek için belirli işleri yapan sınıf ve metot topluluğu olarak tanımlanabilir [6]. Ancak kütüphanelerin yapılandırılabilirlikleri ya yoktur ya da çok düşüktür; bu yüzden kütüphane kullanarak geliştirilen uygulamalar bu kütüphanelere çok bağımlı kalabilmektedir [2][7]. Bu noktada, kütüphanelerin istenilen şekilde yapılandırılabilmeleri hem bu bağımlılığı ortadan kaldıracak hem de yeniden kullanımın maliyet, zaman ve kalite üstüne etkisini artırmaya yardımcı olacaktır.

Günümüzde hızla gelişen ve değişen gömülü yazılım istekleri az zamanda çok iş çıkarma gereği doğurduğundan, yazılım ürünlerinin istenen fonksiyonları yüksek kalitede sağlaması ve planlanan maliyet-takvim içinde bitirilmesi gittikçe daha zor bir iş haline gelmektedir [8]. Bu durum yazılım mühendislerini daha önce geliştirilmiş ve hatasızlığı önceki kullanımlarla artırılmış yapıtaşlarının yeniden kullanılabilmesine yöneltmektedir [9]. Yapılan çalışmalar sonucu yeniden kullanım ve yapılandırılabilirlik düşünülerek geliştirilen kütüphanelerin hem donanımlar arası taşınabilirliği (portability) hem de yazılım güvenilirliğini (reliability) artırdığı görülmektedir [3] [4] [7]. Bu nedenle, hızla değişen istek ve gereklere yine hızlı bir şekilde cevap verebilme amacı ile GYM Radar Ekibi yeniden kullanılabilir yapıları mümkün olduğunca kullanmaya çalışmaktadır. Değişik platform tiplerini içeren radar projelerinin gereksinimlerine göre geliştirilen gerçek zamanlı gömülü yazılımlar, çalışma zamanında farklı dış birimlerden (Flash, USB, FTP) donanımlara yazılım dağıtan ve farklı modüllerin proje özelinde istenen özelliklere göre yapılandırılıp sıralı açılmalarını yöneten bir kütüphaneye ihtiyaç duymuş; bu gerekler Radar Açılış Kütüphanesi ile sağlanmıştır. Bunun yanı sıra, farklı donanımlarda koşan radar gömülü yazılımlarında çalışma zamanında, seçilen protokolle sunucu/istemci mimarisi kullanılarak, seçilen hedef ortama sıralı kayıt alınmasını sağlayan bir kütüphaneye ihtiyaç duyulmuş; bu gerekler ise Radar Kayıt Kütüphanesi ile sağlanmıştır. Bu bildiride, üstte bahsi geçen Radar Destek Kütüphanelerinin yeniden kullanılabilir ve yapılandırılabilir olma özellikleriyle; yazılım kalitesine, farklı donanımlara kolay taşınabilirlik ve güvenilirlik açılarından katkıları detaylandırılacaktır.

2 Radar Yapıtaşları Kapsamında Geliştirilen Kütüphaneler

Yazılım geliştirme yöntemlerindeki iyileştirmeler elde edilen bilgi ve tecrübe birikimine bağlıdır. Radar Projelerinde yazılım yapıtaşı yaratma fikri de eski projelerimizde aynı işlevi sağlayan ama her proje özelinde tekrar gerçeklenen, alan bilgisi gerektiren varlıkların daha verimli kullanılması amacıyla ortaya çıkmıştır. Proje sayısı az, geliştirme süresi uzun, kaynak sıkıntısı olmayan zamanlarda bu tip bir yaklaşıma gitme gereği görülmemiş; ancak farklı platformlarda değişik gerek ve isteklere hızlı cevap vermek bir zorunluluk haline geldiğinde yeniden kullanılabilir yapıtaşları yaratma fikri değerlendirilmeye başlanmıştır. Bu kapsamda, radar

projelerindeki yeniden kullanım bilinçsiz (accidental/ad hoc reuse) olarak ortaya çıkmış, ancak sonrasında taşınabilirlik odaklı (adaptive reuse) devam etmiştir [10].

Yeniden kullanım ile şu hedeflere ulaşılmak istenmiştir:

• **Geliştirme Sürelerini ve Maliyetlerini Azaltmak:** Bu hedefe, ortak varlık havuzu yaratıp bu havuzda yer alan yeniden kullanılabilir kütüphanelerin farklı ürünlerde kullanılmasıyla ulaşılacaktır.

Varlık havuzunu geliştirmek için ilk aşamada bir giriş yatırımı yapılması gerekecektir. Ancak varlık havuzunda yer alan yeniden kullanılabilir kütüphaneler kullanılarak hazırlanan yeni ürünlerin ortaya çıkarılma hızları belirgin bir şekilde artacaktır. Böylelikle toplam maliyet önceki yaklaşımlara göre daha düşük olacak; daha az zamanda ve daha az maliyetle aynı büyüklükte işler yapılabilecektir.

• **Yazılım Güvenilirliğini Artırmak:** Bu artış, ortak varlık havuzundaki kütüphanelerin öncelikle bağımsız olarak test edilmiş olması, ardından pek çok üründe test edilmesi ve hataların giderilmesi ile sağlanacaktır.

Bu kapsamlı kalite güvencesi ile hataların tespit ve düzeltilme oranının yükseleceği, tüm ürünlerin kalitesinin güvenilirlik yönünden artacağı değerlendirilmiştir.

• **Projelerde Düşük Risk Sağlamak:** Güvenilirliği yüksek olan kütüphaneler yeniden kullanıldıkları için, öngörülemeyen hatalar nedeniyle proje takvimlerinin planlandığından gecikmesi riski azalacaktır.

• **İnsan Kaynağının Verimli Kullanımını Sağlamak:** Yetişmiş insan gücü başka işlerde kullanılabileceğinden kaynak verimliliği sağlanacaktır.

Bunların ötesinde, bakım maliyetlerinin azalması, gelişim ve karmaşıklık sorunlarının indirgenmesi gibi daha pek çok motivasyon ekibimizi yazılım yapıtaşı yaratmaya yöneltmiştir.

Bu yapıtaşlarının yapılandırılabilme özelliğinin yeniden kullanım kazanımlarını daha da artıracağı ve geliştirilen uygulamaların bu kütüphanelere bağımlı kalmalarının da engelleneceği düşünülmüştür [11]. *(Bknz.2.3 Kütüphanelerin Yapılandırılması)*

Tüm bu motivasyon ve dikkat edilmesi gereken noktalar düşünülerek, her projede kullanılacak gereklerin ortak bir analizi yapıldıktan sonra yapıtaşları belirlenip bunların geliştirilmesine geçilmiştir. Bu bildiride de her projenin ihtiyaç duyabileceği Açılış ve Kayıt Destek Kütüphanelerinden bahsedilmiştir.

2.1 Açılış Kütüphanesi

Radar projelerinde işlemci kartları üstünde koşması gereken birçok yazılım vardır. Yazılımların güvenli yüklenip projeye özgü yapılandırılarak alt sistemlerinin çalışır duruma gelmesini yönetecek bir kütüphaneye ihtiyaç duyulmuş ve Açılış Kütüphanesi oluşturulmuştur. vxWorks Gerçek Zamanlı İşletim Sistemine (RTOS) uyumlu Açılış Kütüphanesi GNU C ile derlenmekte ve hedef platform olarak ASELSAN'da tasarlanmış işlemci kartlarını ve piyasada kabul görmüş kartları temel almaktadır.

Yazılımların çekileceği dış birimler projelere göre değişebildiği gibi geliştirme zamanında da nihai duruma göre farklılık gösterebilmektedir. Hangi yazılımın, hangi işlemcide, hangi sırada ve hangi parametrelerle açılacağı, geliştirme süresince değişe-

bilmekte ve bu yüzden yapılandırılabilme özelliğine sahip olması büyük önem taşımaktadır. Bu esnekliği sayesinde bu kütüphanenin kullanıldığı sistem, olası yeni platform ve kartlar için genişletilmeye uygundur.

Yazılımların güvenli şekilde yüklenmesi ve alt sistemlerin çalışır duruma gelmesi işlemlerini yöneten bu kütüphane, yüklenecek yazılımların adresini, kimlik bilgilerini, çalıştırılacak modül parametrelerini konfigürasyon dosyasından okuyarak bu yazılımları farklı dış birimlerden (Flash, USB, FTP) yükleyip konfigürasyon dosyasından okuduğu sıraya ve gecikme (delay) sürelerine göre başlatmaktadır.

Radar projelerinde değişik kartlar üzerinde birçok yazılım bulunmakta; gereksinimlerin hızla değişmesi dolayısıyla da bu yazılımların değişik yapılandırmalarla teslim edilmesi gerekmektedir. Örneğin bazı platformlar ağ (network) sistemi içermediğinden FTP bulunmamakta, yazılımların kalıcı bir belleğe yazılması gereği ortaya çıkmaktadır. Bunun için yazılım yeri geldiğinde Flash üstünden, yeri geldiğinde donanım üzerinde yer alan USB bellek üstünden yüklenmekte; alt sistemlerin çalıştırılmasında kullanılan parametreler ortamdan ortama değişiklik gösterebilmektedir. İşte bu sebeplerle Açılış Kütüphanesi kullanan projeler, yazılımların "açılış" gereğini yerine getirirken yeniden kullanımın güvenilirlik karakteristiği yönünden kaliteyi artırırken maliyeti düşürmüştür.

2.2 Kayıt Kütüphanesi

Yazılımların hatasız bir şekilde son kullanıcıya ulaştırılması projenin başarısını belirleyen en önemli etkenlerden biridir. Teslim edildikten sonra beklenmedik şekilde çalışan yazılımlardaki hataları düzeltmek çok maliyetlidir. Bu maliyetin azaltılabilmesi için yazılımlardaki olası hataların, yaşam döngülerinin en erken safhalarında bulunup düzeltilmesi gerekir [4]. Bu nedenle farklı donanımlardaki görevlerin çalışma sıralarının ve ürettikleri sonuçların hassas bir şekilde analiz edilip olası hataların testler esnasında kolay bir şekilde bulunabilmesi çok önemlidir. Ancak, çok işlemcili (multiprocessor) ve çokgörevli (multitasking) gömülü yazılımlarda, platform ve geliştirme ortamı kısıtlarından dolayı hata ayıklama işleminin hazır bir araç (COTS) yardımıyla gerçekleştirilmesi zor ve maliyetlidir. Bunun yerine gömülü yazılımlarda hata kaynağını belirlemek için fazladan kod yazıp log tutmak ve raporlanmış olan hatayı analiz etmek sıklıkla uygulanan bir yöntemdir. Ayrıca bir sistemin müşteriye teslim edildikten sonraki bakım çalışmaları (maintenance) için gerçek ortamdaki çalışma durumunun kayıt altına alınması ve gerektiğinde analiz edilebilmesi, bakım maliyetlerini azaltarak müşteri memnuniyetini de artırır. Radar projelerinde kullanılmak üzere yukarıdaki ihtiyaçları farklı platformlar için gideren ve bu esnada sistem kaynaklarını en az düzeyde tüketerek görev ve zaman kritik işlemlerin yapılmasına olanak sağlayan Radar Kayıt Kütüphanesi oluşturulmuştur.

Kayıt Kütüphanesi istemci ve sunucu yeteneklerini sunarak farklı işlemciler üzerindeki istemcilerden gelen logların bir kayıt sunucuya gönderilebilmesini sağlar. İstemcilerle sunucu arasındaki haberleşme protokolü paylaşmalı bellek mesaj kuyruğu ya da TCP/IP soketi olabilir. Ortak bellek alanlarına sahip işlemciler arasında paylaşmalı bellek mesaj kuyruğu protokolünün tercih edilmesi, ağ donanımı açısından kısıt taşıyan sistemlerde çalışılabilmesini sağlar. Ayrıca, istemcilerle sunucu arasında

TCP/IP soketi üzerinden haberleşmenin sağlanabilmesi ve bu protokolün genel geçerliliğinin fazla olması sayesinde bu kütüphane yeni platformlarda kolaylıkla kullanılabilir. Kütüphanenin tasarımındaki modülerlik ileride kullanılması gerekebilecek yeni protokollerin genel yapı bozulmadan gerçeklenmesine olanak sağlamaktadır. İstemcilerle sunucu arasında haberleşme protokolü konfigürasyon dosyası üzerinden seçilerek kütüphane farklı ortamlar için kolaylıkla yapılandırılabilir.

Kayıt sunucu, istemcilerden aldığı logları ara belleğinde biriktirir ve konfigürasyon dosyasından seçilen hedef platforma kaydedilmesini sağlar. Kaydın yapılacağı hedef platform olarak TCP/IP bağlantısı kurulabilen bir bilgisayar ya da kayıt sunucunun erişebileceği bir USB bellek seçilebilir. IP adresi ve port numarası konfigürasyon dosyasından yapılandırılabilen kayıt sunucu hedef platforma TCP/IP üzerinden kayıt yapılabilmektedir. Ancak, ağ donanımı açısından kısıt taşıyan sistemlerde hedef platform olarak USB bellek seçilerek kayıt işlevleri de yerine getirilebilir. Ayrıca, tasarımdaki modülerlik sayesinde, genel yapı değiştirilmeden farklı hedef platformlara kayıt yapılması ve bunun da konfigürasyon dosyası üzerinden seçilebilir olması sağlanır.

Kayıt Kütüphanesi hedef platformlara log kaydetmenin yanı sıra, üzerinde çalıştığı işlemciye ait seri kanal konsol ekranına ve Telnet servisinin sağladığı konsol ekranına da anlık log basılmasını sağlar. Böylelikle sistem durumu, bağlanılan herhangi bir konsol üzerinden gerçek zamanda takip edilebilmektedir. Ayrıca kütüphanenin sağladığı arayüz fonksiyonları kullanılarak loglar istenilen konsoldan takip edilebilir; hem hedef platforma log kaydetme hem de istenen konsola log basma seçeneklerinden biri ya da her ikisi seçilebilir. Bu esneklikler, yazılım durumlarının daha hızlı bir şekilde analiz edilebilmesine olanak sağlayarak entegrasyon maliyetini düşürmüştür.

Kayıt Kütüphanesi işlevleri, farklı işlemcilerde farklı konfigürasyon dosyalarına göre Radar Açılış Kütüphanesi tarafından çalıştırılabilir. Açılış kütüphanesinin donanımsal kısıtlar nedeniyle yazılımları yükleyebildiği farklı dış birimlerden, Kayıt kütüphanesi de konfigürasyon dosyasını okuyarak kendini yapılandırabilmekte ve donanımsal kısıtları aşabilmektedir.

Radar projelerindeki gerek çeşitliliğine ve donanım kısıtlarına bağlı olarak hem sunucu bilgisayara hem de USB bellek üzerine kayıt alınma ihtimali vardır. Radar projeleri bu gereksinimi yapıtaşı çerçevesinde gerçekleştirip yazılım kalitesini artırmak için Kayıt kütüphanesini kendi gerekleri doğrultusunda yapılandırıp kullanmaktadır.

2.3 Kütüphanelerin Yapılandırılması

Literatürde kütüphanelerin yapılandırılabilirlikleri düşük olduğundan kütüphane kullanarak geliştirilen uygulamaların bu kütüphanelere bağımlı kalabilmeleri eleştirilmektedir [2][7]. Bu yüzden eğer kütüphane kullanımı esas alınacaksa, yapılandırılabilme önemli bir gerek olarak ortaya çıkmaktadır.

Yapılandırılabilir yazılım geliştirme geleneksel yazılım geliştirme tekniklerine kıyasla önemli avantajlara sahiptir. Bunlar arasında, hızlı geliştirme, yürütme

esnasında yazılımı değiştirebilme, evrimsel tasarıma uygunluk, yazılım idamesinin kolaylaşması sayılabilir [12].

Extensible Markup Language (Genişletilebilir İşaretleme Dili, XML), hem insanlar hem bilgi işlem sistemleri tarafından kolayca okunabilecek dokümanlar oluşturmaya yarayan, W3C tarafından tanımlanmış bir standarttır [13]. XML sağladığı esneklik, kolay okunabilir ve anlaşılabilirliği ile günümüzde birçok alan ve uygulamada yaygın olarak kullanılmaktadır [14]. Ayrıca, piyasada var olan açık kodlu ve birçok programlama dili ile kullanılabilen XML, yapılandırma için kullanılan en önemli tercihlerden biri olmaktadır [15]. Projelerimizde kullanacağımız yapılandırma yöntem ve tekniği için değişik alternatifler değerlendirildikten sonra, yapılandırılabilmenin sadece yazılımcılar tarafından değil, sistem mühendisliği ya da kullanıcılar tarafından da yapılabilmesine imkân sağlayacak, kolay anlaşılabilir XML seçilmiştir. Bu yöntem ile de kütüphaneler kolayca yapılandırılabilerek esneklik kazanmışlardır.

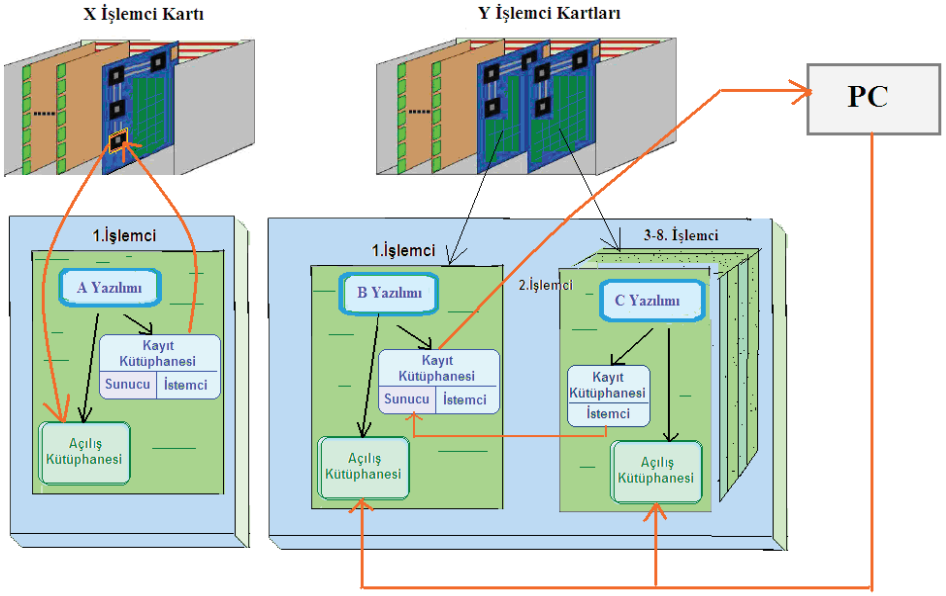
Açılış ve Kayıt Kütüphanelerinde kullanılan xml konfigürasyon dosyasının yapısı Tablo-1'de verilmiştir.

Tablo 1. Açılış ve Kayıt Kütüphanesi Yapılandırma Dosyaları

```
<ACILIS>
  <Kart No="1">
    <CPU No="0">
      …
      <Yukle>Yonetim.out</Yukle>
      <Yukle>VeriIsleme.out</Yukle>
      <YukleYeri>3</YukleYeri><!--flash/usb/ftp-->
      <Modul>
        <Calistir>kayit</Calistir>
        <GorevIsmi>Kayit</GorevIsmi>
        <GorevOnceligi>150</GorevOnceligi>
        <GorevSecenegi>0</GorevSecenegi>
        <YiginBoyu>40000</YiginBoyu>
        <Arg1>1</Arg1
        <Arg2>Kayit.xml</Arg3
        …
        <Arg10>0</Arg10>
        <Gecikme>2000</Gecikme>
      </Modul>
      <Modul>
        …
      </Modul>
    </CPU No="0">
    <CPU No="1">
      …
    </CPU No="1">
</ACILIS>
```

```
<KAYIT>
  <SUNUCU>
    <SunucuTipi>1</SunucuTipi><!--tcp/usb-->
    <TCPLogIP>15.20.5.67</TCPLogIP>
    <TCPLogPort>3960</TCPLogPort>
    …
  </SUNUCU>
  <ISTEMCI>
    …
  </ISTEMCI>
<!-- Sunucu&Istemci arasındaki protokol -->
  <PROTO>2</PROTO><!--msgQ/soket-->
  <SOKET>
    <SoketIP>15.20.7.80</SoketIP>
    <SoketPort>5040</SoketPort>
  </SOKET>
</KAYIT>
```

Şekil-1'de değişik işlemci kartlarında koşan yazılımların değişik konfigürasyonlarla Açılış ve Kayıt Kütüphanelerini yapılandırması gösterilmiştir. Örneğin, ağ donanımı açısından kısıtlı bulunan X işlemci kartı üzerindeki Açılış Kütüphanesi, A Yazılımının ve Kayıt Kütüphanesinin USB bellekten yüklenip açılmasını, Kayıt Kütüphanesi de logların USB'ye yazdırılmasını sağlayacak şekilde yapılandırılmıştır. Y işlemci kartlarında ise Açılış Kütüphanesi sistemdeki yazılımları FTP'den çekip çalışacakları işlemcilerde sıralı bir şekilde açılmalarını sağlamıştır. Kayıt Kütüphanesi kullanımında ise 1. işlemcide kayıt sunucu, diğer işlemcilerde sadece istemci işlevi yapılandırılarak bu istemcilerin loglarını 1. işlemcideki sunucuya göndermeleri sağlanmıştır. Bu kayıt sunucu da loglarını TCP/IP bağlantısı kurulabilen bir bilgisayara gönderir.



Şekil. 1. Kütüphanelerin Değişik Konfigürasyonlarla Yapılandırılması

3 Kazanımlar

Yeniden kullanılabilir ve yapılandırılabilir yapıtaşları oluşturarak hedeflenen kazanımlar, ilk projede etkisini göstermeye başlamış; proje sayısı ve çeşitliliği arttıkça bu kararın hem maliyete hem de kaliteye olan etkisi açıkça gözlemlenmiştir. Bu kütüphanelerin kullanıldığı ilk proje, pilot proje olarak varlık havuzunu geliştirirken o proje özelinde bir giriş yatırımı gerektirmiş; ancak sonraki projeler varlık havuzunda yer alan yeniden kullanılabilir kütüphaneleri kullandıklarından toplam maliyet önceki yaklaşımlara göre daha düşük, ürün ortaya çıkarma hızı daha yüksek olmuştur.

Yeni bir işlemci kartına geçildiğinde, var olan kütüphanenin sadece yeni işlemci kartına özel kısımları için kod geliştirme yapılırken, zaten halihazırda çalışan kısımları için işçilik harcanmaz. Tablo-2'de görüldüğü gibi işlemci-özel kod yüzdesi (İşlemci Tipine Özel Kod Satırı / Toplam Kod Satırı) Açılış Kütüphanesinde %10'un altındayken, bu değer Kayıt Kütüphanesinde %5'in altındadır. Bu da kullanılan yapıtaşlarının yeniden kullanılabilirlik yüzdesinin yüksek ve yeni bir işlemci kartı geldiğinde eski yapının kolaylıkla taşınabilir olduğunu göstermektedir [10].

Tablo 2. Kütüphaneler kullanılan İşlemci-Özel kod yüzdeleri

İşlemci-Özel kod yüzdesi	**İşlemci Tipi**			
	1	2	3	4
Açılış Kütüphanesi	% 9.4	% 6.5	% 8.1	% 8.1
Kayıt Kütüphanesi	% 2.6	% 2.7	% 4.5	% 3.9

Ortak varlık havuzundan kullanılan Açılış ve Kayıt Kütüphaneleri hem bağımsız müşteri testleriyle hem de kullanıldıkları projelerden dönen hata raporları sonucunda yapılan düzeltmelerle güvenilirliklerini artırmışlardır. Bu kalite güvencesi, yazılım süreç ve proje teslimat tarihlerinde düşük risk ile çalışmamıza olanak sağlamıştır. Tablo-3'te Açılış ve Kayıt Kütüphanesinin şirket içi Değişiklik Yönetimi Sistemi'nden alınan yıl bazında raporlanmış hatalara ayrılan işçiliğinin ve raporlanan hata sayısının bugüne yaklaştıkça düştüğü, bunun da güvenilirlik karakteristiği açısından kaliteyi olumlu yönde etkilediği görülmektedir.

Tablo 3. Kütüphanelerin raporlanan hataları için harcanan işçilik süreleri – hata sayıları

Harcanan İşçilik (dak) - Raporlanan Hata Sayısı	**2011**	**2012**	**2013**
Açılış Kütüphanesi	2040 - 13	1820 - 6	45 - 1
Kayıt Kütüphanesi	3280 - 18	665 - 11	85 - 4

Yapıtaşı kullanımından önce her proje sahibi bu ortak yapıları projesi için kendi yazılımı içinde kodlamakta; test maliyetleri de dâhil olmak üzere yazılım yaşam döngüsündeki her aktivite için zaman harcamaktaydı. Oysa yapıtaşı kullanımı, yetişmiş insan gücünün başka işlerde kullanılmasını imkân vererek kaynak verimliliğini sağlamıştır. Tablo-4'te yapıtaşı kütüphane kullanan ve kullanmayan projelerin açılış ve kayıt işlevlerini gerçeklemek için harcadığı işçilik bilgileri verilmiştir. Kütüphane kullanımında bu işlevlerin projeye kazandırılması maliyeti bu yapıların konfigürasyonun yapılması için geçen süre olup 1 adam-saat'ten daha az olduğu gözlemlenmiştir. Böylelikle kütüphane kullanan projelerde yaklaşık 300 katlık bir maliyet ve zaman kazancı sağlanmıştır.

Tablo 4. Kütüphane kullanan/kullanmayan projelerde kütüphane işlevleri için harcanan işçilikler

Harcanan İşçilik (adam-saat)	**Kütüphane Kullanmayan Projeler**		**Kütüphane Kullanan Projeler**		
	A	B	C	D	E
Açılış işlevlerinin gerçeklenmesi	~230	~140	~1	<1	<1
Kayıt işlevlerinin gerçeklenmesi	~110	~76	~1	<1	<1

Bölüm 2.3'de anlatıldığı üzere, kullandığımız yapıtaşı kütüphanelerin en önemli özelliği yapılandırılabilirlikleridir. Eğer bu özellik göz ardı edilseydi, bu durumda her proje kütüphane parametrelerini değiştirmek istediğinde kaynak kodu değiştirip yeniden derlemeliydi. Ancak xml konfigürasyon dosyası sayesinde bu derleme zorunluluğu olmadan kütüphane parametreleri istenilen şekilde değiştirilebilmektedir. Bu sayede entegrasyon çalışmaları esnasında farklı konfigürasyonlara hızlı bir şekilde geçiş yapılabilmekte ve sistem farklı koşullara hemen adapte edilebilmektedir.

Kayıt Kütüphanesi, gerçek zamanlı çalışma esnasında sistemdeki görev ve zaman kritik işlerin yapılmasını engellemeden sunucu/istemci işlevlerini yerine getirmelidir. Yardımcı bir işlev olarak kullanılan kayıt kütüphanesi kendi içindeki bir sorundan dolayı sistemin çalışmasını engellememelidir. Bu yüzden kendi hata durumlarını da aynı işlemci üzerindeki diğer yazılımlara işletim sisteminin hata kodu arayüzünü kullanarak bildirmektedir. Böylelikle kütüphanenin hata toleransı karakteristiği artırılarak diğer yazılımların hata yönetimi yapabilmeleri sağlanmıştır.

Açılış ve Kayıt Kütüphaneleri aynı kart ve işletim sistemini kullanan radar dışındaki projelerde de kullanılabildiğinden, GYM içinde başka ekipler de bu kütüphanelerin müşterisi olmuştur. Bu dış müşteriler, ekipler arası iletişimi ve paylaşımı artırdığı gibi olası yeni yetenekler eklenirken bu yapıtaşlarının yol haritasının belirlenmesinde katkıda bulunacaktır.

4 Sonuç

Hızla karmaşıklaşarak gelişen, rekabet koşullarının sürekli değiştiği gömülü yazılım geliştirme sektöründe başarılı olmak için sadece en iyi mühendisleri işe almak veya en iyi yazılım geliştirme araçlarını kullanmak yeterli olmamaktadır. Kısa zamanda hızlı ve kaliteli iş çıkarmak için yeniden kullanım konsepti ile kullanılan yapıtaşı kütüphaneler hem taşınabilirlik hem de güvenilirlik karakteristiği yönünden yazılım kalitesinin artmasına yardımcı olmaktadır. Gömülü yazılımları çalışma zamanında farklı dış birimlerden donanımlara dağıtıp sıralı açılışı yapılandırabilme ve seçilen protokolle istenen hedef ortama sıralı kayıt alabilme işlevlerini yapıtaşı kütüphaneler kullanarak gerçekleştiren GYM Radar Ekibi, projelerindeki yeniden kullanılabilir ve yapılandırılabilir varlıkları artırmıştır. Yeni tip işlemci kartlarına daha kolay taşınabilir ve önceki proje deneyimleri sayesinde güvenilirliği artmış

yapıtaşı kütüphaneler, projelerdeki yazılım geliştirme süre ve maliyetlerini azaltarak, nitelikli insan kaynağının daha verimli kullanılıp proje risklerini azaltmıştır.

5 Teşekkür

Yazarlar, Radar Destek Kütüphanelerinin analizi, tasarımı, müşteri testlerinde bulunan GYM Radar Ekibi üyelerine ve bu kütüphaneleri projelerinde kullanan GYM'deki yazılım sorumlularına teşekkür eder.

Kaynakça

1. Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.
2. Karlsson, E. Software Reuse -A Hollistic Approach, John Wiley & Sons, 1995
3. Pressman, R. S., Software Engineering:A Practitioner's Approach, McGraw-Hill, 2010.
4. Sommerville, I. Software Engineering, Addison Wesley, 2010
5. IEC/PAS 62814:2012, *Dependability of software products containing reusable components – Guidance for functionality and tests*
6. Rotem, Arnon "Frameworks vs. Libraries",http://archive.is/HLcYJ, (Orjinal: http://www.ddj.com/blog/architectblog/archives/2006/07/frameworks_vs_l.html) Erişim Tarihi: 17.03.2014
7. Frakes, W.B & Kang, K (2005) *Software Reuse Research: Status and Future, IEEE Transactıons On Software Engıneerıng,* 2005
8. Leveson, N.G. & Weiss, K.A. *"Making Embedded Software Reuse Practical and Safe",* http://sunnyday.mit.edu/papers/fse04.pdf, Erişim Tarihi: 28.03.2014, 8 sayfa
9. Estublier, J. & Vega, G. *"Reuse and Variability in Large Software Applications",* ACM SIGSOFT Software Engineering Notes, 30(5): 316 – 325, 2005
10. Belli, F., "*Dependability and Software Reuse – Coupling Them by an Industrial Standard",* 2013 Seventh International Conference on Software Security and Reliability Companion.
11. Tillmann, N. and De Halleux, J., *"White-Box Approaches for Improved Testing and Analysis of Configurable Software Systems",* Proceedings of the 2nd International Conference on Tests and Proofs, 2008
12. Stewart, D. B. and Arora, G., *Dynamically Reconfigurable Embedded Systems – Does it Make Sense?,* Proc of Real Time Application Workshop, 1997.
13. Extensible Markup Language (XML). http://www.w3.org/XML/.
14. Storrs,J & McArdle, G. Fusion Engineering and Design 83 (2008) 429–433
15. Learning XML http://oreilly.com/catalog/learnxml2/chapter/ch02.pdf