Yazılım Geliştirmede Sanal Makinelerin Kullanımı

Selçuk Bozcan[1], Ahmet Erdinç Yılmaz[2]

Aselsan A.Ş. SST-MD-YMM, P.K. 1 06172, Yenimahalle, Ankara
[1]sbozcan@aselsan.com.tr
[2]aeyilmaz@aselsan.com.tr

Özet. Yazılım geliştirme ortamlarının hazırlanması ve korunması, genellikle detaylı yapılandırma gerektiren ve zaman alan bir süreçtir. Yazılım ekibine yeni katılımlar ve geliştirme ortamlarının kurulu olduğu mevcut fiziksel donanımlarda meydana gelen değişiklikler gibi nedenlerle geliştirme ortamlarının yeniden oluşturulması ihtiyacı yazılım geliştirme çalışmalarında sıkça karşılaşılan bir durumdur. Bununla birlikte farklı uygulamalara ait geliştirme ortamlarının, bazı durumlarda çeşitli uyumsuzluklardan dolayı tek bir fiziksel donanım üzerine kurulamıyor olması, uygulama geliştirme ortamlarının kurulumunu ve korunmasını güçleştirmektedir. Geliştirme, test ve bakım-idame süreçlerinde, geliştirme ortamının kolayca ulaşılabilir olması, ayrıca önem taşımaktadır. Bu makalede, bahsedilen sorunların çözümünde sanal makinelerin kullanımı ile ilgili yaklaşımlar ve tecrübeler anlatılmıştır.

Anahtar Kelimeler. Yazılım Geliştirme, Geliştirme Ortamı, Sanal Makine, Geliştirme Ortamının Korunması

1 Giriş

Günümüzde yazılım geliştirme, farklı disiplinlerden farklı ekiplerin bir arada çalışmasını gerektiren, birbirinden farklı geliştirme ve test ortamlarının aynı anda kullanılmasını zorunlu kılan, karmaşık bir süreç haline gelmiştir [5]. Buna ilave olarak, proje bakım ve idame süreçlerinin karmaşıklığı da yazılım geliştirme sürecinin karmaşıklığına paralel olarak artış sergilemektedir. Bu süreçlerin iyileştirilmesi yönünde çok sayıda çalışma bulmak mümkündür [1], [2], [3].

Yazılım geliştirme sürecinin karmaşıklığını arttıran en temel nedenlerden birisi, sistemlerde yer alan yazılım birimi sayısının artmasıdır. Bu yazılım birimlerinin her biri farklı geliştirme ortamlarına ihtiyaç duyabilmektedir. Bu da yazılım ekibi tarafından kullanılması gereken geliştirme ortamlarının sayısını artırmaktadır. Yazılım geliştirme ortamlarının sayısının artması, bu ortamların hazırlanması, yönetimi ve idamesi ile ilgili güçlükleri doğurmaktadır.

Aynı sistemde yer alan ilişkili yazılım birimlerinin farklı ekipler tarafından geliştiriliyor olması da yazılım geliştirme sürecinin karmaşıklığını önemli ölçüde artırmaktadır. Yazılım geliştirme ekipleri arasındaki entegrasyonu sağlamak amacıyla geliştirilen yazılımların belirli aralıklara paylaşılması gerekmektedir. Bunun için ise standart olarak benimsenen yöntem, diğer geliştiricinin bilgisayarı üzerine yazılımın kurulu-

munu gerçekleştirmektir. Yazılımların kurulumu, çoğu zaman hedef platform üzerinde bir takım özel ayarlamaların gerçekleştirilmesini, ilave kütüphanelerin ve yardımcı yazılımların kurulmasını gerektirmektedir. Bu işlemler de kurulum işlemini karmaşık ve zaman alıcı hale getirebilir. Bunun da ötesinde, geliştirilen yazılımlar hedef platforma özel olarak tasarlandıkları için, bazı durumlarda, özel donanım gereksinimleri, kurulu olan diğer yazılımlarla ve işletim sistemi ile olan uyumsuzlukları ve benzeri diğer nedenlerden dolayı, yazılımların her türlü platforma kurulumunu yapmak her zaman mümkün olmayabilmektedir. Diğer bir yöntem ise, geliştiricinin bilgisayarından ayrı, yeni bir bilgisayar üzerine kurulum yapmaktır. Bu yöntem ise hem ilave maliyet gereksinimleri, hem de fiziksel yerleşim problemlerinden dolayı her zaman uygulanabilir bir yöntem değildir.

Yazılım geliştirme sürecinde yaşanan diğer bir güçlük de geliştirme ortamının tutarlılığını ve güvenirliğini tüm geliştirme süreci boyunca sağlanmasıdır. Yazılım geliştiricinin makinesi üzerine kurulu olan yazılım geliştirme ortamı, zamanla çeşitli sebeplerle yüklenen diğer uygulamalar ve yapılan ayarlamalar nedeniyle farklılaşabilir. Bu durum genellikle kirlenme olarak adlandırılmaktadır. Kirlenmiş bir yazılım geliştirme ortamı, kaynağı tespit edilmesi güç hataların oluşmasına yol açabilmektedir.

Yazılım geliştirme ortamının, sadece geliştirme sürecinde değil, bundan daha sonra gelen idame ve bakım sürecinde de erişilebilir olması kritik önem taşımaktadır. Yazılım geliştirme ortamlarının projenin idame ve bakım dönemi boyunca saklanması, özellikle farklı geliştirme ortamlarına ihtiyaç duyan karmaşık sistemlerde, özel olarak ele alınması gereken, yönetimi zor bir süreçtir.

Bahsedilen bu sorunların çözümü için geliştirme ortamının proje yaşam döngüsü içinde her zaman korunması ve hızlı bir şeklide ulaşılabilir halde olması gerekmektedir. Her bir projede yazılım geliştirmesinde ve paylaşılmasında ayrı bir donanım kullanarak koruma sağlamak her ne kadar bir alternatif olsa da, ciddi kaynak israfı nedeni ile etkin bir yöntem olarak görülmemektedir [5].

DSS (Deniz Savunma Sistemleri) Yazılım Ekibi bünyesinde yukarıda bahsedilen sorunlarla başa çıkmak için sanal makineler üzerinde yazılım geliştirme yöntemi benimsenmiştir. Yürütülen farklı projeler için sanal makineler oluşturulup, proje yaşam döngüsü boyunca gerçekleştirilen tüm geliştirme ve idame faaliyetleri projeye özgü bu sanal makineler kullanılarak gerçekleştirilmektedir. Bu sayede geliştirme ortamlarının kurulum ve yapılandırma maliyetlerinden önemli ölçüde kazanımlar sağlanmıştır.

Makalenin geri kalanı şu şekilde düzenlenmiştir: İkinci bölümde sanal makinelerin yazılım geliştirme faaliyetlerinde nasıl kullanılabileceği açıklanmıştır. Üçüncü bölümde DSS Yazılım Ekibi bünyesinde sanal makinelerin yazılım geliştirme faaliyetlerinde nasıl kullanıldığı önerilen yaklaşım olarak anlatılmıştır. Dördüncü bölümde sanal makine kullanımının zorlukları belirtilmiş ve beşinci ve son bölümde sonuç ve değerlendirme yer almaktadır.

2 Sanal Makineler Kullanarak Yazılım Geliştirme

Yazılım geliştirme sürecinde sanal makinelerin kullanımı, giriş bölümünde bahsedilen zorlukların çözümüne önemli katkıda bulunmaktadır. Klasik yöntemdekinin tersine, her bir projenin ayrı bir sanal makine üzerinde geliştirilmesi ile hem geliştirme ortamı hem de geliştiricin ortamı birçok bakımdan korunmuş olacaktır.

Geliştirme aşamasında projenin bağımlı olduğu kütüphaneler, yazılım çerçeveleri (framework) ve diğer tüm yardımcı yazılımlar yeni bir sanal makine ortamına kurularak, geliştirme ortamının diğer tüm etkenlerden soyutlanması sağlanmaktadır. Bu şekilde üçüncü parti yazılımlar arasındaki olabilecek çakışmalar engellenip zaman ve kaynak kaybına sebep olan hataların çoğu daha oluşmadan engellenmektedir [7].

Sanal makine kullanımı sadece kullanılacak yazılım ortamın yalıtımı için değil ayrıca donanım ile de ilgili ihtiyaçlara da cevap vermektedir. Çalışacağı hedef bilgisayarın bellek, sabit disk, işlemci gücü, ağ bağdaştırıcısı ve seri kanal gibi donanım özelliklerinin de sanal makine yönetim aracı ile kolaylıkla ayarlanabilmesi sağlanmaktadır. Bu özellik ile geliştirmenin hedef ortamın fiziksel olarak da eş bir ortamda geliştirilmesi için ayrı donanım masraflarının düşmesini sağlayacaktır. Yapılan araştırmalarda geliştirme sürecinde sanal makine kullanımının donanım maliyetlerini %48 ile %80 arasında azalttığı bildirilmiştir [5]. Hem donanımsal hem de yazılımsal olarak hedef çalışma ortamının eşleniğinde çalışılması, hedef ortamda müşterinin karşılaşabileceği sorunların geliştirme ortamında yakalanıp önlem alınması için fayda sağlamaktadır.

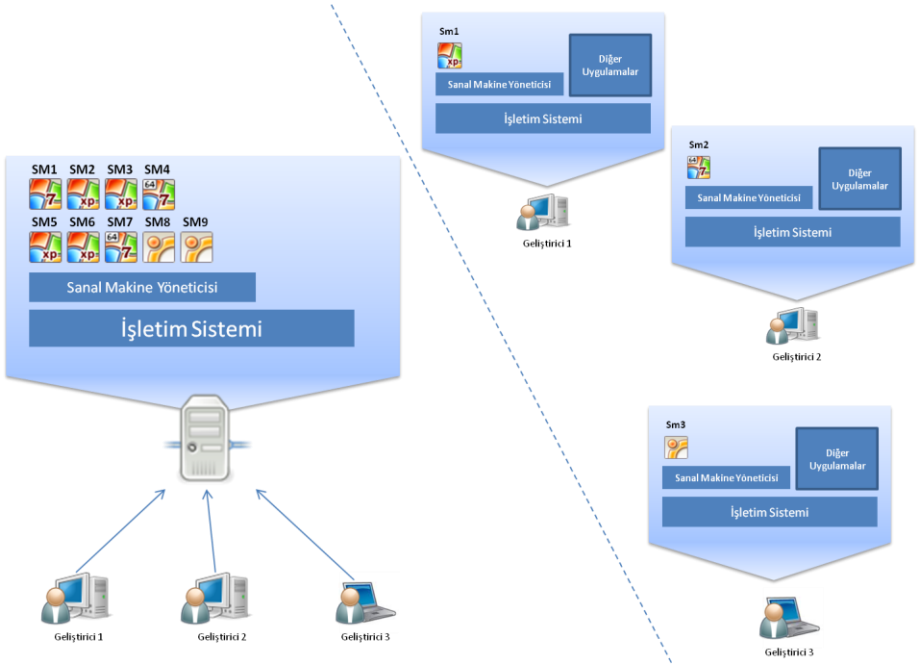
Geliştirme aşamasında yapılan yapılandırma hataları ve yazılım geliştirmesi sırasında yapılan testler sonucu ile geliştirme bilgisayarının çökmesi gibi sorunlardan korunmak için de sanal makine çözümü kullanılmaktadır. Kimi durumlarda bu gibi testlerin çok sayıda tekrarlanması gerekip hedef bilgisayarın ortamının bozulmasına periyodik olarak sebep olmaktadır. Sanal makine kullanımında ise yapılacak test öncesi sanal makine dosyası kopyalanarak, geliştirme ortamının zarar verici test öncesi haline getirilmesi kolaylıkla yapılabilmektedir.

Sanal makine kullanımı geliştirme yapılan gerçek bilgisayar ortamının da korunmasını sağlamaktadır. Farklı projelerin aynı bilgisayar üzerinde geliştirilmesi, bağımlı yazılım bileşenlerin çakışması gibi problemlerin yaşanmasına sebep olmaktadır. Sanal makine ile yapılan geliştirme ise geliştiricinin kişisel bilgisayarının kirlenmesini önemli ölçüde azaltacaktır. Bu önlem zamanla artan, kişisel bilgisayarların tekrar kurulması ihtiyacı için ayrılan kaynakların tasarrufunda da önemli rol oynayacaktır.

Sanal makine kullanımının bir diğer faydası da hem geliştirme ortamının hem de kişisel bilgisayarların güvenliğinin korunmasıdır. Kurulan kütüphanelerin, yardımcı yazılımların sayısı arttıkça güvenlik açığına sebep olacak zararlı yazılımların kontrolü de zorlaşmaktadır. Her ne kadar kullanılan güvenlik yazılımları bu açıkları engellemek için yer alsa da, şüphe duyulan kurulumların, testlerin ağa bağlı olmayan ve tam kontrollü sanal makineler de gerçeklenmesi güvenlik açıkları için daha baştan bir engel oluşturacaktır [4].

Hata! Başvuru kaynağı bulunamadı.' de sanal makinelerin kullanımında yaygın olarak kullanılan iki temel yaklaşım bulunmaktadır. İlki tüm sanal makinelerin ayrı bir sunucu da yer alıp, geliştiricilerin bunlara uzaktan erişim sağlamasıdır. Bu kulla-

nımda, geliştiricilerin sanal makinelerini yönetmesi için internet tarayıcısı üzerinden ulaşabilmesini sağlayan yardımcı yazılımlar kullanılmaktadır. Bu şekilde yeni makine ihtiyacı ya da kendi geliştirme makinesini açma, kapama gibi işlemleri için sunucuya uzaktan bağlanabilmektedir. Bu yaklaşımın en büyük avantajı, geliştirmenin uzaktan bağlantı yapılan herhangi bir bilgisayar üzerinden yapılabilmesidir. Bu sayede geliştirici, kendi kişisel bilgisayarının kullanılamaz olduğu durumda geliştirmeye başka bir bilgisayardan devam edebilmektedir.. En büyük dezavantaj ise sanal makine sunucusundaki bir hata tüm sanal makinelerin çalışma durumunu etkilemektedir.



Şekil 1. Sanal Makine Kullanım Yaklaşımları

Bir diğer yaklaşım ise tüm geliştiricilerin sanal makineleri kendi bilgisayarları üzerinde çalıştırmasıdır. Bu durumda kendi geliştirme makineleri ana bir sunucu üzerinde olmadığı için çalışma durumu sadece geliştiricin kendi kişisel bilgisayarının durumuna bağlıdır, bir geliştiricinin kişisel bilgisayarındaki hata diğer bir makineyi etkilemez. Fakat bu kullanım yönteminde geliştirme artık geliştiricinin bilgisayarına da bağımlı olmaktadır. Bu sebeple uzaktan bağlanıp geliştirme yeteneği daha sınırlıdır.

Hangi yöntemin kullanılacağı proje ve ekip içi ihtiyaçlara göre belirlenmelidir. Örneğin işyeri dışında gösterim, test gibi durumlarda ikinci yöntem çok daha uygundur, işyeri içinde, birden fazla projede, farklı makinede çalışma gibi durumlarda ise ilk yöntem daha çok fayda sağlayacaktır.

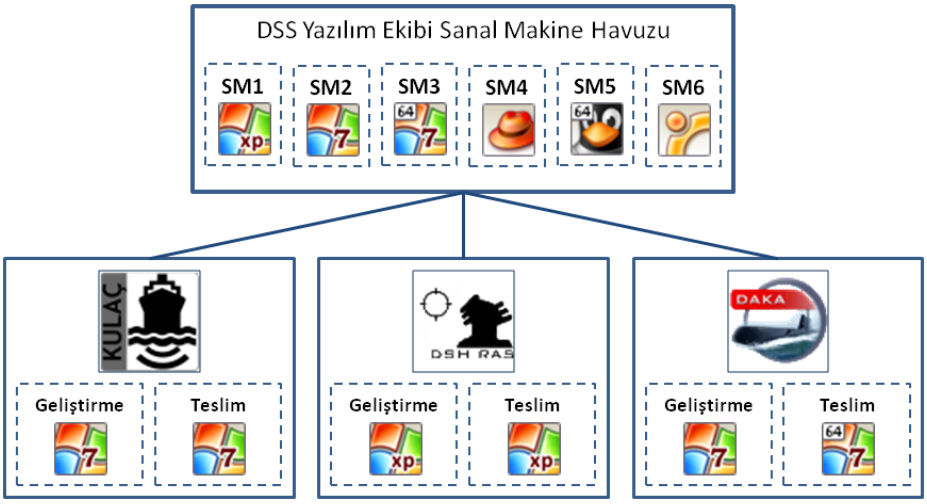
3 Önerilen Yaklaşım

Bu makalede yazılım geliştirmede sanal makinelerin kullanımı ile ilgili olarak önerilen yaklaşım iki farklı aşamada incelenebilir. İlk aşama, hızlıca sanal makineler oluşturabilmek amacıyla, yazılım ekibi genelinde sanal makine şablonlarını içeren bir havuz oluşturmayı hedeflemektedir. İkinci aşama ise bir yazılım projesi özelinde sanal makinelerin geliştirme, test ve yazılım entegrasyonu gibi safhalarda etkin olarak kullanılmasını hedeflemektedir.

3.1 Sanal Makine Havuzunun Oluşturulması

Yazılım geliştirme sürecinde sanal makinelerin kullanımından sağlanacak temel faydalardan biri, ihtiyaç duyulduğunda çok hızlı bir şekilde istenilen yazılım platformunun oluşturulabilmesidir. Önceden hazırlanmış sanal makine şablonlarını barındıran bir havuz oluşturmanın, yeni bir sanal makine oluşturma işlemini önemli ölçüde hızlandıracağı değerlendirilmektedir. Buna göre, yazılım ekibi tarafından çözüm sunulan platformlara ait sanal makine şablonlarını içeren bir havuz oluşturulması önerilmektedir. **Hata! Başvuru kaynağı bulunamadı.**'de DSS Yazılım Ekibi tarafından oluşturulan sanal makine havuzu örnek olarak verilmiştir. Bu havuzda yer alan sanal makine şablonları, standart donanım özelliklerine sahip bir sanal makine üzerine varsayılan ayarları ile kurulmuş işletim sistemlerini içermektedir. Yazılım ekibi tarafından çözüm sunulan platformlar güncellendikçe, sanal makine havuzu da güncellenir. Ayrıca, havuzda yer alan sanal makine şablonları düzenli olarak gerçekleştirilen işletim sistemi güncellemeleri ile sürekli güncel tutulur.

DSS Yazılım Ekibi tarafından 48GB belleğe ve 12 işlemciye sahip Ubuntu 12.04 işletim sistemli bir sunucu makinesi, sanal makine sunucusu olarak kullanılmaktadır. Sanal makine yöneticisi olarak Oracle VM VirtualBox 4.2.16 tercih edilmiştir. Sanal makine yöneticisinin uzaktan kontrolü için ise açık kaynak kodlu phpVirtualBox 4.2.4 uygulaması kullanılmaktadır. Sunucu üzerindeki sanal makine havuzunda Windows XP, Windows 7, RedHat Linux ve Ubuntu işletim sistemlerinin kurulu olduğu sanal makine şablonları hazırlanmıştır. DSS Yazılım Ekibi tarafından yürütülen çeşitli projeler için hâlihazırda 40 adet sanal makine hazırlanmıştır.



Şekil 2. DSS Yazılım Ekibi Sanal Makine Havuzu

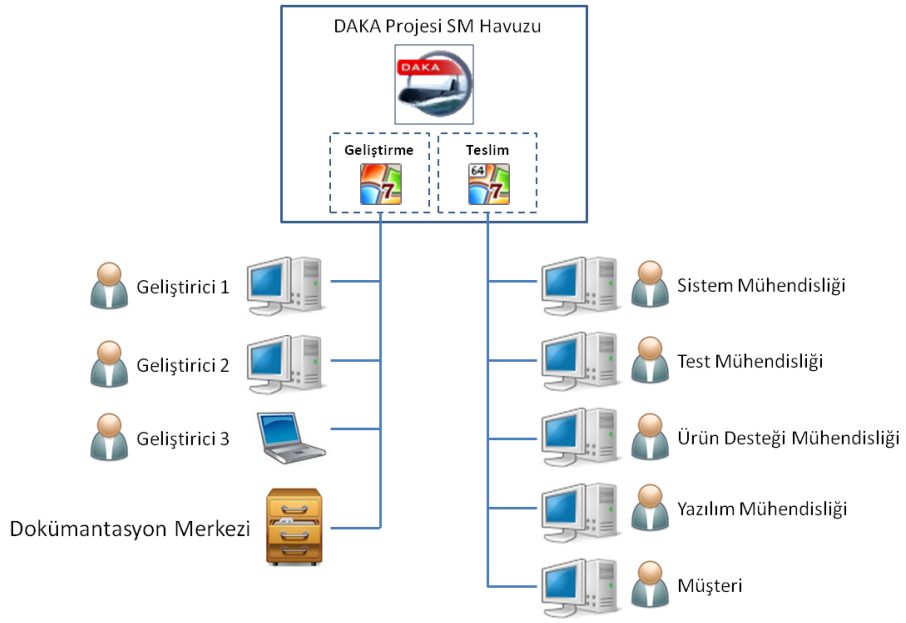
Yazılım ekibi tarafından yürütülen her bir proje için bu havuzdaki şablonlar kullanılarak projeye özgü olacak şekilde yeni bir sanal makine havuzu oluşturulur. Projeye özgü olarak oluşturulan bu havuzda iki farklı sanal makine şablonu yer almalıdır. Bunlardan ilki, proje kapsamındaki her türlü geliştirme faaliyetlerinde kullanılmak üzere bir Geliştirme makinesi şablonudur. İkincisi ise, projenin yazılım geliştirme ekibi dışındaki paydaşlarına çeşitli amaçlarla verilecek olan Teslim makinesi şablonudur. Proje süresince ihtiyaç duyulan bütün sanal makineler bu iki şablon kullanılarak üretilmelidir.

3.2 Proje Genelinde Sanal Makinelerin Kullanımı

Projeye özgü sanal makine havuzunda yer alan Geliştirme ve Teslim sanal makine şablonları, genel sanal makine havuzunda yer alan şablonların üzerinde projenin ihtiyacına yönelik gerekli donanımsal ve yazılımsal ayarlamalar gerçekleştirilerek oluşturulur. Buna göre, Geliştirme sanal makine şablonu oluşturulurken ilk olarak projenin geliştirme ihtiyaçlarına uygun olacak şekilde bir sanal makine şablonu genel havuzdan seçilir. Seçilen bu şablon üzerinde işlemci, bellek, ağ bağdaştırıcısı ve diğer gerekli görülen diğer donanımsal ayarlamalar yapılır. Bahsedilen bu donanımsal ayarlamalar yapıldıktan sonra projenin geliştirme ortamını oluşturmak amacıyla aşağıda listelenen yazılımsal ayarlamalar gerçekleştirilir:

- İşletim sistemi yamaları ve ayarlamaları
- Bütünleşik geliştirme ortam(lar)ı kurulumu
- İhtiyaç duyulan yazılım geliştirme çerçeveleri ve kütüphanelerinin kurulumu
- Veritabanı, birim test aracı, performans analiz aracı gibi yazılım geliştirmede kullanılan diğer yazılımların kurulumu

- Yazılım geliştirmede kullanılabilecek diğer her türlü yazılımsal kaynağın kurulumu



Şekil 3. Proje İçi Sanal Makine Yönetimi

Yukarıda verilen donanımsal ve yazılımsal ayarlamalar yapıldıktan sonra oluşturulan Geliştirme sanal makinesi şablonu kullanılarak **Hata! Başvuru kaynağı bulunamadı.**'de belirtildiği gibi yazılım ekibinde yer alan her geliştirici için ayrı bir sanal makine oluşturulur. Böylece geliştiriciler tarafından kullanılan geliştirme ortamlarının tutarlılığı garanti edilmiş olunur. Ayrıca geliştirme ekibine yeni katılımlarda geliştirme ortamının baştan kurulması ile kaybedilecek zaman önlenmiş olunur.

Geliştirme ortamının proje idame süresi boyunca korunabilmesi için proje için özel olarak hazırlanan Geliştirme sanal makinesi şablonunun dokümantasyon merkezine aktarımı önerilmektedir.

Projeye özgü olarak oluşturulan sanal makine havuzunda, Geliştirme sanal makinesi şablonuna ilave olarak Teslim sanal makinesi şablonu da yer alır. Teslim sanal makinesi şablonu, Geliştirme sanal makinesi şablonundan farklı olarak projenin çalışma zamanı ihtiyaçları göz önünde bulundurularak hazırlanır. Buna göre örneğin, Teslim şablonunda bütünleşik geliştirme ortamı gibi geliştirme safhasında kullanılan araçlar yer almaz. Yazılımın çalışan sürümünün kurulumu, hazırlanan bu Teslim şablonu üzerine gerçekleştirildikten sonra projenin paydaşlarına çeşitli amaçlarla verilebilir. Teslim sanal makinelerinin çeşitli paydaşlar tarafından öngörülen kullanımı aşağıda özetlenmektedir:

- Yazılım Mühendisliği: Projede birden fazla YKB (Yazılım Konfigürasyon Birimi)'nin bulunduğu ve her bir YKB'nin farklı bir yazılım ekibi tarafından

geliştirildiği durumlarda sanal makineler, diğer yazılım ekiplerince yazılım entegrasyonu ve test amaçlı olarak kullanılabilir.

- Sistem Mühendisliği: Yazılım prototiplerinin değerlendirilmesi, yazılım gereksinimlerinin kontrol edilmesi ve sistem entegrasyon testlerine hazırlık amaçlı olarak kullanılabilir.
- Test Mühendisliği: Yazılım yeterlilik testlerinin gerçekleştirilmesi ve simülatör uygulamalarının hazırlanması amacıyla kullanılabilir. Gerekli altyapılar kurulup, yazılım yeterlilik testleri sanal makineler üzerinde otomatik olarak gerçekleştirilebilir.
- Ürün Desteği Mühendisliği: Projeye ait destek ve eğitim dokümanlarının hazırlanması ve müşteriye eğitimlerin verilmesi amaçlarıyla kullanılabilir. Bunlara ilaveten, müşteri tarafından bildirilen hatalar sanal makineler üzerinde ürün desteği mühendisleri tarafından tekrardan oluşturulup yazılım ekibine iletilebilir.
- Müşteri: Yazılım prototiplerinin değerlendirilmesi ve yazılım gereksinimlerinin kontrol edilmesi amacıyla hazırlanan sanal makineler müşteri tarafından kullanılabilir.

4 Karşılaşılan Zorluklar

Yazılım geliştirme sürecinde sanal makinelerin kullanımı her ne kadar bahsedilen sorunlara çözüm sunsa da, birtakım zorlukları da beraberinde getirmektedir [8]. Yazılım geliştirme ve sonrasında bakım ve idame sürecinde sanal makinelerin etkin olarak kullanılabilmesi, bu zorlukların giderilmesine bağlıdır. Bahsedilen zorlukların en önemlileri performans, özel donanım gereksinimleri ve lisanslama problemleri olarak sıralanabilir.

4.1 Performans

Sanal makineler temelde, Sanal Makine Yöneticisi tarafından sağlanan donanım soyutlaması üzerinde çalışan işletim sistemleridir. Donanım kaynaklarının Sanal Makine Yöneticisini çalıştıran işletim sistemi ile paylaşılması nedeniyle, sanal makinelerin performansı normal makinelere göre daha düşük olmaktadır. Bu performans sorunu, sanal makinelerin kişisel bilgisayar donanımları üzerinde değil, güçlü sunucular üzerinde çalıştırılması ile giderilebilir. Fakat bu durumda da sunucu üzerinde çalışan sanal makinelere uzak erişim yöntemi ile bağlanmak gerekmektedir. Mevcut standart uzak erişim protokolleri, özellikle grafik yoğun işlemlerde performans yönünden yetersiz kalabilmektedir. Uzak erişim yöntemi ile kullanılan sanal makineler için standart uzak erişim protokollerinin yetersiz kaldığı durumlarda yüksek performans vadeden özel protokollerin kullanımı değerlendirilmelidir. [8]

4.2 Özel Donanım Gereksinimleri

Sanal Makine Yöneticisi, üzerinde çalışan sanal makinelere standart aygıtlara sahip bir donanım soyutlaması sağlamaktadır. USB, Seri Kanal ve ağ bağdaştırıcıları gibi

standart donanımlar, bütün Sanal Makine Yöneticileri tarafından desteklense de, bunlar dışında kalan grafik kartları, PCI kartlar gibi donanımlar desteklenmemektedir. Bu durum çoğu zaman bir problem teşkil etmese de, özel donanıma yönelik yazılım geliştirilmesinin yapıldığı durumlarda problem olabilmektedir. Örnek olarak özel bir grafik kartı gerektiren yazılımları sanal makineler üzerinde geliştirmek mümkün olmayacaktır.

4.3 Lisanslama

Yazılım geliştirme sürecinde sanal makinelerin kullanımı, yazılımların lisanslanması için farklı yaklaşımların uygulanmasını gerektirebilir. Örneğin bazı yazılımlar, kullanıldıkları sistemde yer alan işlemci başına lisanslanmaktadır. Sanal makinenin üzerinde çalıştığı sunucunun çok sayıda işlemcisi olmasına rağmen, sanal makine ise sadece bir sanal işlemciye sahip olabilir. Bu gibi durumlarda halihazırda kullanılan lisanslama yaklaşımının değiştirilmesi gerekebilir.

5 Sonuç

Yazılım geliştirme ve sonrasında bakım ve idame süreçlerinde sanal makinelerin kullanımının üretkenlik, maliyet ve etkinlik adında önemli katkıları olduğu değerlendirilmektedir. Yazılım geliştirme ortamlarının sanal makineler üzerinde oluşturulması, geliştirme ortamının tutarlılığını garanti etmekle birlikte, geliştirme ortamlarının oluşturulmasını ve korunmasını kolaylaştırmaktadır. Yazılım geliştirme süreci boyunca yazılımın değerlendirilmesi, test ve entegrasyon gibi çeşitli amaçlarla proje paydaşları arasında yazılımın paylaşılması, sanal makineler ile hızlı ve etkin bir şekilde sağlanabilmektedir. Bu sayede tüm proje ekibinin etkinliği ve üretkenliği önemli ölçüde artırılabilmektedir.

Yazılım geliştirme sürecinde sanal makinelerin kullanımının bir diğer önemli faydası, donanım maliyetlerinin önemli ölçüde azaltılabilmesidir. Normal şartlar altında, donanımsal ve yazılımsal uyumsuzluklar gibi çeşitli nedenlerden dolayı ayrı makineler üzerine kurulması gereken ortamlar, sanal makinelerin kullanımı ile tek bir fiziksel makine üzerinde yer alabilmektedir.

İlerleyen dönemlerde yazılım geliştirme sürecinin her safhasında sanal makine kullanımının daha da önem kazanacağı ve yaygınlaşacağı öngörülmektedir. Bu nedenle, sanal makinelerin yazılım geliştirme ve bakım ve idame süreçlerinin her safhasına nasıl daha etkin kullanılacağını yönelik araştırmaların gerçekleştirilmesinin süreçlerin iyileştirilmesi adına büyük fayda sağlayacağı değerlendirilmektedir. Yapılacak araştırmaların kurumsal olarak desteklenmesi araştırmaların etkinliğini arttıracaktır.

Kaynaklar

1. Piri, A., "Challenges of Globally Distributed Software Development – Analysis of Problems Related to Social Processes and Group Relations," Global Software Engineering,

2008. ICGSE 2008. IEEE International Conference on , vol., no., pp.264,268, 17-20 Aug. 2008
2. Gomes, V.; Marczak, S., "Problems? We All Know We Have Them. Do We Have Solutions Too? A Literature Review on Problems and Their Solutions in Global Software Development," Global Software Engineering (ICGSE), 2012 IEEE Seventh International Conference on , vol., no., pp.154,158, 27-30 Aug. 2012
3. Loka, R.R., "Software development: what is the problem?," Computer , vol.40, no.2, pp.112,111, Feb. 2007
4. Peter M. C.,Brian D N., "When Virtual Is Better Than Real" 8th Workshop on Hot Topics in Operating Systems , pp. 133,138, 2001
5. VMware Inc., "Accelerate Software Development,Testing and Deployment with the VMware Virtualization Platform". Retrieved from http://www.vmware.com/pdf/dev_test.pdf
6. Robert P. G., "Survey of Virtual Machine Research", Journal Computer vol. 7, Issue 9, pp. 34,35, Sep. 1974
7. Glen K., Megan D., Wellie C., Martin H., Edmon B., John P. C.,H., 2006. "The Developer's Guide To Virtual Machines". Retrieved from http://assets.devx.com/ebook/20359.pdf
8. Marilyn B., "VMware® Horizon View™ 5.2 Reviewer's Guide". Retrieved from http://www.vmware.com/files/pdf/view/VMware-View-Evaluators-Guide.pdf
9. Daniel D.(Feb. 11, 2009). "How Virtualization Improves Software Development". Retrieved from "http://www.cio.com/article/480420/How_Virtualization_Improves_Software_Development"