Yazılım Yeniden Yapılamaya Yönelik Model Güdümlü ve Kaliteye Yönelimli Süreç Modeli

Murat Paşa Uysal[1], A. Erhan Mergen[2]

[1] Bilgisayar Teknolojileri Bölümü, Ufuk Üniversitesi MYO, İncek, Gölbaşı, 06836, Ankara, Türkiye.

[2]Saunders College of Business, Decision Sciences, Rochester Institute of Technology, Rochester, N.Y. 14623-5608, USA.

Özet. Yazılım Yeniden Yapılama (software re-engineering) (YYY), yoğun kaynak ve zaman kullanımını gerektiren, gidiş-dönüşlü ve yinelemeli yazılım mühendisliği etkinliklerini içermektedir. Dolayısıyla, söz konusu süreçler otomatik hale getirilebilmeli, ortaya çıkan ürün, araç ve yöntemler yeniden yapılanmış yazılımla ilgili sonraki süreçlerde tekrar kullanılabilmelidir. Bu bağlamda, Model Güdümlü Mimari (MGM) ve Model Güdümlü Yazılım Geliştirme (MGYG), yazılımların otomatik geliştirilebilmesi ile kalite ve öngörülebilirliğini hedefleyen yaklaşımlardır. Ancak model, modelleme ve kalite kavramlarını YYY çalışma alanında bütünleşik olarak birlikte ele alan araştırmalar sınırlı düzeydedir. Bu amaçla çalışmamız, Tasarım Bilimi Araştırma Yöntemi (Design Science Research) (TBAY) doğrultusunda yürütülmüş, sistematik haritalama ile desteklenmiştir. Araştırmamızda "Model Güdümlü ve Kaliteye Yönelimli bir YYY Süreç Modeli" geliştirilmiştir. MGM, YYY ve ISO/IEC 2500n "Software Quality Requirements and Evaluation" (SQuaRE) yazılım kalite standartları bütünleşik olarak kullanılmış, TBAY kapsamında çalışmanın kuramsal temellerini oluşturmuştur. Geliştirilen modelde hesaplama bağımsız modeller mevcut yazılıma ait iş akışı vb. çizeneklerle, platform bağımsız modeller ise UML çizenekleri ve Soyut Söz Dizim Ağaçları (Abstract Syntax Tree) (SSDA) ile temsil edilmektedir. SSDA'ları aynı zamanda iyileştirilecek yazılımın anlambilim yapısının, model ve kod dönüşümlerinde kullanılabilmesini sağlamaktadır. Mevcut yazılıma ait kalite gereksinimleri MGYG doğrultusunda ve SQuaRE standardındaki metrik ve ölçütlerle belirlenmektedir.

Anahtar Kelimeler: Yazılım yeniden yapılama, model güdümlü mimari, yazılım kalitesi,

1 Giriş

Yazılım sistemlerinin ömür devri boyunca ortaya çıkan gereksinimlere yönelik güncelleme, bakım, iyileştirme vb. yazılım etkinliklerin birçoğu Yazılım Yeniden Yapılama (software re-engineering) (YYY) kapsamında ele alınabilecek niteliktedir.

Literatürdeki YYY çalışmaları incelendiğinde, temel amacın mevcut yazılım sistemlerinin kalitesinin iyileştirilmesi, işlevsel ve/veya işlevsel olmayan özelliklerinin geliştirilmesi olduğu görülmektedir [1]. Nesneye yönelimli sistemlere yönelik YYY ile ilgili çeşitli çalışmalar yapılmış, süreçlerin iyileştirilmesine yönelik yöntemler önerilmiştir [2]. Örneğin, bu kapsamda gerçekleştirilen önceki çalışmamızda, "6 Sigma", yazılım süreçleri ve yazılım kalite standartlarının bütünleştirildiği bir model geliştirilmiştir [3]. Ancak YYY, yoğun kaynak ve zaman kullanımını gerektiren, gidiş-dönüşlü (round-trip), yinelemeli ve artırımsal yazılım mühendisliği etkinliklerini içermektedir [4]. Dolayısıyla, bu süreçler otomatik hale getirilebilmeli, ortaya çıkan ürün ve araçlar yeniden yapılanmış olan yazılımla ilgili sonraki süreçlerde kullanılabilmelidir.

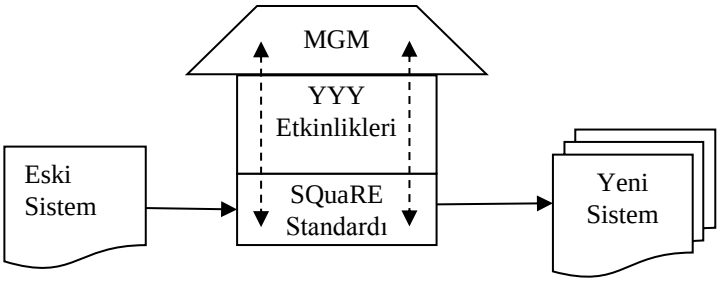
Yazılım sistemlerin anlaşılırlığı ve sürecin kolaylaştırılması amacıyla çeşitli modelleme yöntem ve tekniklerine yazılım süreçlerinin farklı aşamalarında başvurulmaktadır [5]. Bu bağlamda, Model Güdümlü Yazılım Geliştirme (MGYG) ve Model Güdümlü Mimari (MGM) yazılım sistemlerinin otomatik olarak geliştirilebilmesini, kalite, etkililik ve öngörülebilirliğini farklı modelleme ve soyutlama düzeylerinde hedeflemektedir [6, 10]. Ancak, YYY, MGYG ve kalite kavramları arasında kavramsal ve işlevsel ilişkiler bulunmakla birlikte bu çalışma alanlarını birlikte ele alan çalışmalar sınırlı düzeydedir. Söz konusu bilgi alanlarını bütünleşik kullanarak YYY sürecinin iyileştirilmesini hedefleyen kapsamlı çalışmalar ise az sayıdadır [1]. Dolayısıyla, söz konusu araştırma boşluğunu doldurmaya yönelik yapılacak çalışmaların, YYY ile ilgili kuramsal alana katkıda bulunabileceği gibi yazılım endüstrisinde konuyla ilgili problem sahalarına da ışık tutulabileceği düşünülmektedir.

1.1 Problem

Bir YYY süreci, farklı soyutlama düzeylerinde bulunan bir dizi model dönüşümleri ile bu modellerin program kodlarına çevrildiği yazılım dönüşümlerini içermektedir. Yeninden yapılandırılacak yazılımın sahip olması istenildiği özellikler ile kalite ölçütleri dikkate alındığında, çalışmamıza konu olan "Model Güdümlü ve Kaliteye Yönelimli YYY" (MGKY_YYY)'ya yönelik araştırma problemi formal olarak aşağıdaki gibi ifade edilebilir:

> Yeniden yapılanacak bir yazılım sistemi ***S*;** bu sistemin sahip olması istenildiği P_1, P_2, …P_n özellikleri ve ilgili kalite ölçütleri $K\ddot{O}_1$, $K\ddot{O}_2$, …$K\ddot{O}_n$ verildiğinde; model ve yazılım dönüşümlerini içeren öyle bir Yeniden Yapılama Dönüşüm ***YYD*** dizisi bulunuz ki MD_1, MD_2, …MD_n model dönüşümleri ile YD_1 YD_2, …YD_n yazılım dönüşümlerini içersin ve yeniden yapılanan yazılım sistemi $\boldsymbol{S'} = \boldsymbol{YYD}$ ((*MDn-*$_1$ (…MD_1)) X (*YDn-*$_1$ (…YD_1)), P_1,(*S'*), P_2,(*S'*), … P_n,(*S'*) özellikleri ile $K\ddot{O}_1$,(*S'*), $K\ddot{O}_2$,(*S'*), … $K\ddot{O}_n$,(*S'*) kalite ölçütlerini karşılasın.

Bu amaçla çalışmamızda, araştırma probleminin çözümüne yönelik olarak, Tasarım Bilimi Araştırma Yöntemi (Design Science Research) (TBAY) [7] [8] doğrultusunda geliştirilen ve Şekil 1'de ana bileşenleri verilen bir YYY süreç modeli sunulmaktadır.



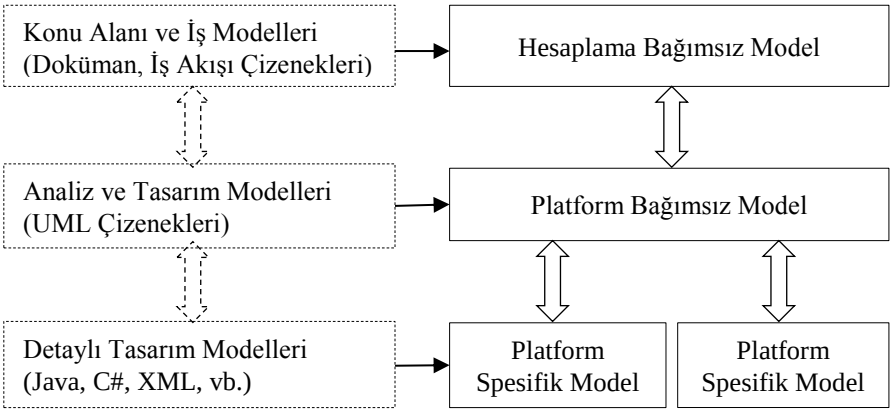
Şekil 1. Çalışmanın Ana Bileşenleri

Makalenin sonraki bölümleri, çalışmanın kuramsal temellerini, araştırma yöntemini, geliştirilen model ve sınırlıkları içermektedir.

2 Kuramsal Çerçeve

2.1 Yazılım Yeniden Yapılama

Yeniden Yapılamayı (re-engineering), yapılanacak bir sistemi, mevcut sistemle aynı ya da daha üst seviye bir soyutlama düzeyinde yeniden oluşturma ve yeni sistemi uygulamalarla devam ettirme olarak tanımlamak mümkündür [9]. Aynı kapsamda YYY sürecini ise (a) evrimleşebilen bir sistem oluşturmak, (b) mevcut yazılımın işlevlerini geliştirmek, (c) ona yeni işlevler katarak (d) kalitesini iyileştirmek amacıyla; (a) tersine mühendislik (reverse engineering), (b) yeniden düzenleme (restructuring, refactoring) ve (c) ileriye mühendislik (forward engineering) ile mevcut bir yazılımın iyileştirilmesi olarak tanımlamak mümkündür. Bu kapsamda YYY, genel olarak program dönüştürme (program transformation) ve program gösterimi (program representation) işlemlerinden oluştuğu söylenebilir. Program dönüştürmede, çeşitli gereksinim ve ölçütlere (yazılım dili, hedef mimari, soyutlama düzeyleri vb.) bağlı olarak, program çevirisi (translation), göçü (migration), optimizasyonu vb. yazılım etkinlikleri gerçekleşebilmektedir. Program gösteriminde ise, ayrıştırma ağaçları (parse tree), soyut söz dizim ağaçları (abstract syntax tree), çizgeler (graph) vb gösterim yöntemlerinden bir veya birkaçı kullanılabilmektedir [2].



Şekil 2. Model Güdümlü Mimari

2.2 Model Güdümlü Mimari

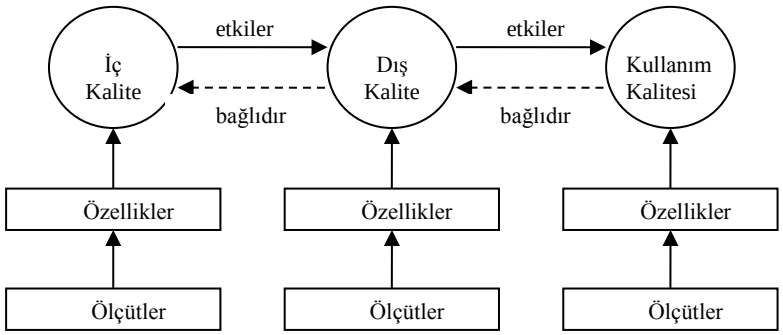
Modeller bir problem alanıyla ilgili karar alma ve ona yönelik bir çözüm geliştirmek amacıyla kullanılmaktadır. Modelleme ve MGM, MGYG olarak bilinen yazılım geliştirme yaklaşımının temelini oluşturmaktadır [10]. Dolayısıyla, modeller arasındaki ilişkiler, bir probleme yönelik çözümün yaratıldığı süreci kayıt altına alır ve karşılıklı bağımlılıkları içeren yapıyı da oluştururlar. Bu ilişkiler aynı zamanda sistem tasarlama ve geliştirme sürecinin herhangi bir noktasındaki değişikliklerin anlaşılabilmesini, etkilerinin öngörülebilmesine de sağlamaktadırlar. MGM'de yazılım geliştirme süreci, modeller arasında bir dizi dönüşüm olarak gerçekleşmekte, çeşitli katman ve dönüşüm işlemlerinden oluşan bir mimari çerçeve doğrultusunda evrilmektedir.

Şekil 2'de görüldüğü gibi MGM sistem tasarımına üç farklı bakış açısıyla yaklaşmaktadır. İlk aşamada Hesaplama Bağımsız Modeller, teknolojiden bağımsız konu alanıyla ilgili sistemin nasıl gerçekleştirileceğini belirlemektedir. Platform Bağımsız Modeller, teknik detaylardan soyutlanmış ve yine platformdan bağımsız olarak sistemle ilgili bir grup servisi tanımlamaktadır. Son olarak Platform Spesifik Modeller, hedef platformu dikkate almakta ve sisteme yönelik teknik detayları eklemektedir. Bu üç farklı model, çeşitli model dönüşüm kuralları doğrultusunda birbirlerine dönüştürülmektedir. Yazılımın analiz, tasarım veya kodlama aşamasındaki herhangi bir değişiklik diğer aşamalara kolayca yansıtılabilmektedir.

2.3 Yazılım Kalitesi

Yazılım mühendisliğiyle ilgili kalite standartları genel olarak McCall [11], Boehm [12], ISO-9126 ve ISO/IEC 2501n [13] olarak sınıflamak mümkündür. Bu standart ve modellerin tanıtımı çalışmamızın kapsamı dışında olup MGKY_YYY süreç modelinin bileşenlerinden olan ISO/IEC 2501n (Software Quality Requirements and Evaluation (SQuaRE)) standardı temel alınmaktadır. Bu standart, yazılım kalite

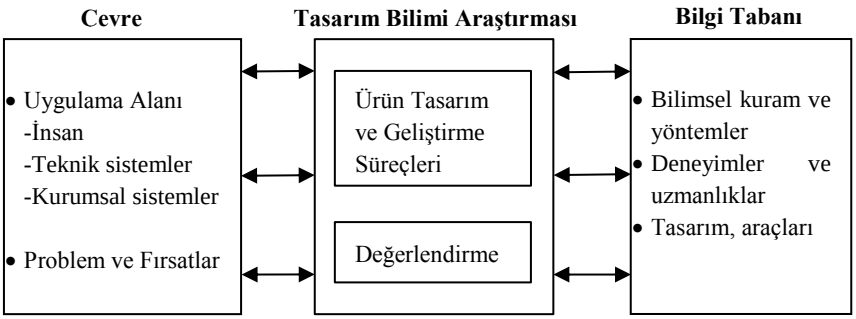
ihtiyaçlarının tanımlanması, ölçülmesi ve değerlendirilmesine yönelik kalite modelini, süreçleri ve ölçütleri belirlemektedir. Bu bağlamda yazılım kalitesi üç boyutta ele alınmaktadır: (a) *İç Kalite* (internal quality): Karmaşıklık, satır sayısı, Nesneye Yönelimli Programlama (NYP) metrikleri vb. ölçütler doğrultusunda yazılımın statik yapısına ait kaliteyi ifade etmektedir. (b) *Dış Kalite* (external quality): İlgili yazılımın işlevsel olarak test ortamında ihtiyaçları ne kadar karşıladığının ölçüsüdür. (c) *Kullanım Kalitesi* (quality in use), geliştirilen yazılımın gerçek kullanım ortamında kullanıcı ihtiyaçlarına ne kadar cevap verebildiğini ölçmektedir [13]. Bu standart, bir yazılım sisteminin kalitesini üç farklı bileşenle ortaya koymakla birlikte bu boyutlar aynı zamanda birbirlerine karşılıklı olarak bağımlıdırlar. Bir başka ifadeyle, yazılımın tasarım ve geliştirme aşamalarını ilgilendiren iç kalite özellikleri yazılımın dış kalitesini belirlemekte, dış kalite özellikleri ise söz konusu yazılımın kullanım kalitesini doğrudan etkilemektedir (Şekil 3).



Şekil 3. SQuaRE Standardında Yazılım Kalite Bileşenleri [13]

2.4 Tasarım Bilimi Araştırma Yöntemi

Fen ve Sosyal Bilimler, genel olarak doğa ve toplumda yer alan varlık, olgu ve kavramları tanımlamaya veya bunlar arasındaki ilişkileri açıklamaya çalışmaktadır. Bunlardan farklı olan Tasarım Biliminde, pratik amaçlara yönelik, belirli işlev ve özelliklere sahip araç, sistem ve modelleri, bunların analiz, tasarım, geliştirme ve değerlendirme aşamalarını da kapsayan ve bilimsel verilere dayalı bilgi birikimi oluşturulmaktadır [7]. Bilgi teknolojileri (BT) endüstrisinde tasarım ve geliştirme projelerinin temel amaç ve kaygısı, mevcut ve onaylanmış standartları, bilgiyi, rutin süreç ve modelleri kullanarak ürünleri maliyet etkin biçimde ve verimli geliştirmektir. TBAY'de ise bu ürünlerin daha iyi geliştirilmesini sağlayacak bilimsel bilgi birikimine katkıda bulunmak amaçlanmaktadır. TBAY dayalı bir projede, çevre ve gerçek hayat problemlerinden hareket edilerek araştırma yapar gibi BT araç, yöntem, model veya kuramları geliştirilir, iyileştirilir ya da ve sınanır [7, 8]. Şekil 4'te gösterildiği gibi TBAY'de içinde çeşitli yinelemeli etkinliklerin bulunduğu; (a) Çevre, (b) Tasarım Bilimi Araştırması ve (c) Bilgi Tabanı aşamalarından oluşan üç ana bileşen bulunmaktadır.



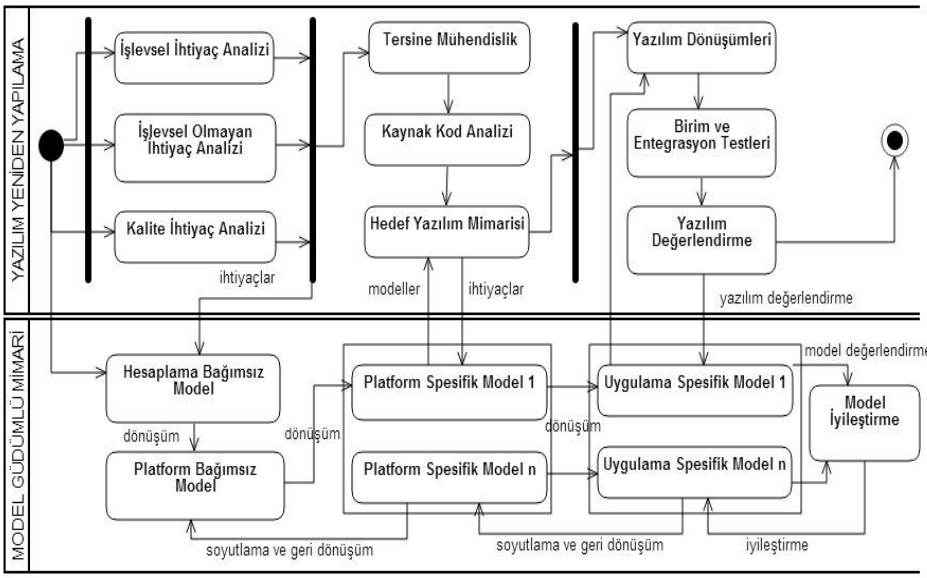
Şekil 4. TBAY Temel Bileşenleri, ([7]'den uyarlanmıştır).

3 Yöntem

Bu araştırma, TBAY [7,8] çerçevesinde yürütülmüş, sistematik haritalama (SH) yöntemiyle desteklenmiştir. Çalışmanın TBAY doğrultusunda kuramsal temellerini ve ana bileşenlerini MGM, YYY bilgi alanı ve ISO/IEC 2500n SQuaRE yazılım kalite standartları oluşturmuştur. Yazılım mühendisliği problem alanıyla olan ilişkiyi, endüstrideki uygulamalar ile SH sonucunda ortaya konulan bulgular belirlemiştir. SH ile taramaya başlamadan önce çalışmanın sistematikliği, yansızlığı, geçerlilik ve güvenirliğini sağlayacak bir protokol geliştirilmiştir. Bu protokol, araştırma problemiyle ilgili arama deyimlerini, yayınların seçim ölçütleri ile ulaşılan kaynaklara nasıl uygulanacağını, veri kayıt ve tasnif yöntemini, hangi veri ve bulguların alınacağı gibi konuları içermiştir [14]. Taramada, YYY ve MGM temelini oluşturan süreçler, modeller, aktörler, teknoloji, etkinlik vb. yapılar; YYY ile bu yapılar arasındaki ilişkileri ortaya koyan modeller; söz konusu YYY yapılarını hayata geçiren çeşitli kuram, yöntem, model ve uygulamalar incelenmiştir. Böylece, YYY ve MGM arasındaki olası dolaylı ve doğrudan ilişkiler araştırılarak MGKY_YYY'nin temelleri ve modelin süreç yapısı oluşturulmaya çalışılmış ve aşağıda detaylı olarak açıklanmıştır:

3.1 Model Güdümlü ve Kaliteye Yönelimli YYY Süreç Modeli

Literatürde önerilen YYY çözümlerinde modelleme etkinlikleri genellikle, yazılım geliştirme süreçleri içine gömülü ve sınırlı biçimde, modelleme ise yazılımı görselleştirmek, anlaşılırlığı sağlamak ve yazılım süreci desteklemek amacıyla kullanılmaktadır. Çalışmamızda önerilen MGKY_YYY modelinde geleneksel YYY faaliyetleri, model güdümlü olarak yürütülmekte, kalite gereksinimleri yine MGM doğrultusunda SQuaRE ölçütleriyle ifade edilmektedir. Şekil 5'te gösterildiği gibi önerilen modelde YYY ve MGM olmak üzere bütünleşik iki yazılım ve sistem tasarım /geliştirme süreç alanı bulunmaktadır. Modelleme ve YYY etkinlikleri bu iki alan arasında gidiş-dönüşlü olmakta, yazılım modülleri ve modeller farklı soyutlama düzeylerinde detaylandırılarak model dönüşümleri gerçekleştirilmektedir.



Şekil 5. Model Güdümlü ve Kaliteye Yönelimli YYY Süreç Modeli

3.1.1 İhtiyaç Analizi, Hesaplama ve Platform Bağımsız Model Aşaması

Bu aşama, YYY sürecinin ihtiyaç analizi, MGM kapsamında hesaplama ve platform-bağımsız modellerin oluşturulduğu, etkinliklerin bütünleştirilerek paralel yürütüldüğü aşamadır. En genel anlamda kaynak ve hedef yazılımlarının modellenmesi ve hedef yazılıma dönüştürülme süreci aşağıdaki gibi ifade edilebilir:

$$t : M_1(S_1) \mid_{F1} \rightarrow M_2(S_2) \mid_{F2} \quad (1)$$

t, YYY sürecindeki t_1, $t_2 \ldots t_n$ sıralı model dönüşümlerini, S_1 yeniden yapılandırılacak kaynak yazılımı, S_2 hedef yazılımı, M_1 kaynak yazılıma ait modelleri, M_2 hedef yazılıma ait modelleri, F_1 ile F_2 ise kaynak ve hedef yazılım model dönüşümlerinde kullanılan formal gösterim yöntemlerini simgelemektedir.

- Öncelikle ihtiyaç analizine, işlevsel olan ve olmayan gereksinimlerin belirtimi ile başlanılmaktadır. Doküman incelemesi, tersine mühendislik ve kaynak kod analizi ile iyileştirilecek yazılımın incelemesi yapılır. Mevcut yazılıma ait varsa doküman incelenmesi ya da tersine mühendislik ile iş akış çizenekleri Hesaplama Bağımsız Modeller olarak oluşturulmaktadır.
- MGM çerçevesinde, mevcut yazılım kaynak kodlarından platform bağımsız model olarak Soyut Söz Dizim Ağaçları (Abstract Syntax Tree) (SSDA) çıkarılmaktadır. Daha sonra bu modeller, modelden modelle dönüşüm kuralları, işlevsel ve kalite gereksinimleri de dikkate alınmak suretiyle hedef yazılımda aynı soyutlama düzeyinde yine SSDA'ları olarak gösterilmektedir. Böylece mevcut

yazılıma ait bütün gereksinimler ile anlam bilimsel yapının hedef yazılıma olduğu gibi aktarılabilmesi sağlanmaktadır:

$$M_s \rightarrow M_s{}^a \rightarrow M_t{}^a \rightarrow M_t \quad (2)$$

- M_s mevcut yazılıma ait modeller, $M_s{}^a$ yine mevcut yazılımın SSDA'ları olarak dönüştürülmüş platform bağımsız modelleri, $M_t{}^a$ ise gerekli düzenleme, kalite ve eklemeler yapılıp aynı soyutlama düzeyinde gerçekleştirilen hedef yazılımın SSDA'larını, son olarak M_t ise hedef yazılıma ait dönüştürülmüş modelleri simgelemektedir.
- UML (Unified Modelling Language) çizenekleri de kaynak ve hedef yazılımda platform bağımsız modelleri temsil etmektedirler. Bu çizeneklerde mevcut sistemle ilgili yazılım mimarisi, yapısal ve işlevsel hatalar belirlenerek düzeltilmektedir.
- SQuaRE çerçevesinde uygun metrik ve ölçütlerle hedef yazılımın kalite gereksinimleri ortaya konulmaktadır. Böylece, hedef yazılımın sahip olması istendiği P_1, P_2, …P_n özellikleri ve $K\ddot{O}_1$, $K\ddot{O}_2$, …$K\ddot{O}_n$ kalite ölçütleri MGM çerçevesinde platformdan bağımsız modellenmektedir.

3.1.2 Hedef Yazılım ve Platform Spesifik Model Aşaması

Bu aşamada, ihtiyaç analizinde belirlenmiş işlevsel/işlevsel olmayan ihtiyaçlar, yazılım kalite ihtiyaçları ve hedef programlama dili dikkate alınır, hedef yazılım mimarisi ve model dönüşüm kuralları belirlenir. Başka bir ifadeyle, farklı seviyelerdeki MD_1, MD_2, …MD_n model dönüşümlerini içerecek YD_1 YD_2, …YD_n yazılım dönüşümleri sırasıyla yinelemeli ve artırımsal olarak gerçekleştirilmektedir.

- Hedef programlama dili ve çalıştırma platformu dikkate alınır, bir önceki aşamadaki platform-bağımsız model olarak oluşturulan SSDA'ları, UML çizenekleri, ve NYP metrikleri, SQuaRE İç Kalite ölçütleri doğrultusunda hedef yazılım mimarisini belirlenmektedir. Aslında bunların MGYG'deki karşılığı düzenleme dönüşümleri (refactoring transformation) ya da modelden modele (model-to-model transformation) dönüşümlerdir. Model dönüşüm yöntemleri ve yazılım desenleri (pattern) bu dönüşümlerde yol gösterici ilkeleri ortaya koyarak model geliştirme ve iyileştirme sürecine de formalizm kazandırmaktadır.
- Daha sonra hedef programlama dili ve yazılım mimarisinden hareket edilerek iyileştirilmiş platform-bağımsız modeller (UML), platform-spesifik modellere dönüştürülürler. Diğer bir ifadeyle geliştirilen modeller hedef programla diliyle (Java, C#, XML vb.) gösterilmektedir. Bu yazılım dönüşümleri, elle veya otomatik, UML profili ve yazılım desenleri kullanılarak gerçekleştirilmektedir.
- SSDA'ları ile anlamsal bütünlük, NYP metrikleri ile yapısal bütünlük ve iç kalite, SQuaRE ölçütleri ile hedef yazılımın dış ve kullanım kalite ölçütleri karşılanmaktadır.

3.1.3 Uygulama Spesifik Model, Yazılım Dönüşümleri, Test ve Değerlendirme Aşaması

- YYY sürecinde, mevcut ve iyileştirilen yazılıma ait platform-spesifik modeller, hedef platforma yönelik olarak daha da detaylandırılmakta, MGM çerçevesinde uygulama-spesifik model olarak ifade edilen bilgisayar kodlarına dönüştürülmektedir (model-to-code transformation).
- Birim ve entegrasyon testleriyle yeniden yapılanan yazılım sistemi *S*', geçerleme ve doğrulama süreçlerinden geçirilmekte, P_1,(*S*'), P_2,(*S*'), …, P_n,(*S*') yazılım özellikleri ile $K\ddot{O}_1$,(*S*'), $K\ddot{O}_2$,(*S*'), …, $K\ddot{O}_n$,(*S*') kalite ölçütlerinin ne kadar karşıladığı bu aşamada değerlendirilmektedir.
- Değerlendirme sürecinde yapılan güncellemeler ve düzenlemeler, uygulama ve platform-spesifik modellere tekrar aktarılmakta, böylece model geri dönüşüm ve soyutlama ile platform ve hesaplama bağımsız modellere söz konusu değişiklikler yansıtılmaktadır.

4 Çalışmanın Sınırlılıkları

Bu araştırma, bilişim ve bilgi sistemleri alanında son yıllarda kullanılmakta olan TBAY çerçevesinde yürütülmüştür. Bu yöntemde özellikle çalışmanın kuramsal temeli ile yeni bilgi olarak ilgili alana nasıl katkıda bulunduğu esas alınmakta, araştırmalarda geliştirilen sistem, model veya yöntem, araştırma teknikleriyle test edilerek geliştirilmektedir [7, 8]. Ancak, TBAY'nin önemli bir bileşeni test ve değerlendirme süreci olup bu araştırmada geliştirilen modelin, durum çalışması vb. yöntemlerle sahada sınanması mümkün olmamıştır. Dolayısıyla, çalışma sonuçlarının genellenebilirliği bu yönüyle sınırlı düzeydedir. Çalışmanın geçerliliğini tehdit edebilecek unsurlar SH ve uzman görüşlerine başvurularak giderilmeye çalışılmıştır. Araştırmamızdaki bir diğer sınırlılık ise geliştirilen modelin nesneye yönelimli yazılım sistemlerine yönelik olması ve bu bağlamda yapısal programlamayla geliştirilmiş sistemlerin YYY ihtiyaçlara cevap verebilecek nitelikte olmamasıdır.

5 Sonuç ve Öneriler

Araştırmamızın temel yaklaşımı, yoğun kaynak kullanımını gerektiren YYY süreç ve etkinliklerinin otomatik hale getirilmesi gerektiği, kalite ve diğer yazılım gereksinimlerinin MGYG doğrultusunda ele alınabileceği yönündedir. Bu amaçla çalışmamızda, TBAY doğrultunda geliştirilen "Model Güdümlü ve Kaliteye Yönelimli YYY Süreç Modeli" tanıtılmış, ana bileşenleri ve kuramsal temelleri ortaya konulmuştur. Nitel, deneysel/yarı deneysel araştırma yöntemleri ve yazılım mühendisliği teknikleriyle önerilen modelin yazılım kalitesi, etkililik ve verimlikle ilgili beklentileri ne kadar karşıladığı belirlenmelidir. Dolayısıyla makalemiz, ilgili modelin test ve değerlendirildiği ve TBAY doğrultusunda yürütülecek yeni çalışmaların önerisiyle son bulmaktadır.

Kaynaklar

1. Editorial: "A Retrospective View of Software Maintenance and Reengineering Research- A Selection of Papers From 2010 European Conference on Software Maintenance and Reengineering." Journal of Software Maintenance and Evolution, 2011, DOI: 10.1002/smr.548 (2011).
2. Tahvildari, L., Kontogiannis, K. & Mylopoulos J.: "Quality-Driven Software Reengineering." The Journal of Systems and Software, 66, 225-239 (2003).
3. Uysal, M.P. & Mergen, E.: A Quality-Oriented Approach To Software Reengineering. Proceedings of the Northeast Decision Sciences 2013 Annual Conference, Brooklyn, NY, USA, April 5-7, 2013, 971-979 (2013).
4. Wagner C.: Model-Driven Software Migration: A Methodology, Reengineering, Recovery and Modernization of Legacy System. Springer Vieweg (2014).
5. Swithinbank, P., Chessell, M., Gardner, T., Griffin, C., Man, J., Wylie, H. & Yusuf, L.: Patterns: Model-Driven Development Using IBM Rational Software Architect, Redbooks (2005).
6. Beydeda, S.,·Book M., Gruhn, V.: Model-Driven Software Development. Springer-Verlag Berlin Heidelberg (2004).
7. Hevner, A. & Chatterjee S.: Design Research In Information Systems, Integrated Series in Information Systems, 22, DOI 10.1007/978-1 (2010).
8. Vaishnavi, V.K. & Kuechler W.J.: "Design Science Research Methods and Patterns: Innovating İnformation And Communication Technology". Auerbach Publications, Taylor & Francis Group, NW, USA (2008).
9. Elliot J. Chikofsky and James H. C.: Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software, 7(1), 13-17 (1990).
10. Object Management Group. MDA Guide Version 1.0.1. Technical Report omg/2003-06-01, OMG (2003).
11. McCall, J. A., Richards, P. K., Walters, G. F.: Factors in Software Quality. Nat'l Tech. Information Service, Vol. 1, 2 and 3 (1977).
12. Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., Merritt, M.: Characteristics of Software Quality. North Holland, (1978)
13. ISO/IEC 2501n. Quality model division. Retrieved from "http://www.iso.org" on April 4, 2012.
14. Kitchenham, B., Charters, S.: Guidelines for Performing Systematic Literature Reviews in Software Engineering, Keele University and Durham University Technical Report EBSE 2007-001, 2007.