Yazılım Yeniden Yapılamada Öz Çerçeve Yaklaşımı

Murat Paşa Uysal[1], Görkem Giray[2]

[1] Yönetim Bilişim Sistemleri Bölümü, Başkent Üniversitesi, Ankara, Türkiye
mpuysal@baskent.edu.tr

[2] Bağımsız araştırmacı, İzmir, Türkiye
gorkemgiray@gmail.com

Özet. Günümüzde yazılım ömür devri kısalmış, güncel yazılım yöntem ve teknikleriyle geliştirilen sistemler eski yazılım sistemleri (legacy system) arasında yer almaya başlamıştır. Dolayısıyla, Yazılım Yeniden Yapılamanın (Software Re-engineering) (YYY) önemi artarak devam etmektedir. Bu bağlamda YYY projelerinde kullanılacak yazılım süreç modelleri iş kuralları, teknoloji ve bilgi alanından bağımsız olabilmeli ve her türlü yazılım gereksinimine cevap verebilmelidir. Çevik yazılım geliştirme yöntem ve uygulamaları desteklenirken yazılım takımları kendi ihtiyaçları ve deneyimleri doğrultusunda YYY için gereken uygulama, araç ve teknikleri esnek biçimde kullanabilmelidir. Değişik büyüklük, yapı ve platformdaki yazılım sistemlerine kolayca uyarlanabilmeli, küçük çaplı projeden büyüğe doğru evirilebilen ve ölçeklenebilir nitelikte olmalıdır. Literatür incelendiğinde söz konusu problem sahalarına çözüm getirebilecek bir YYY çerçevesinin olmadığı gözlenmektedir. Bu amaçla çalışmamızda, YYY süreçlerini bütünleşik olarak ele alan ve Öz Çerçeve (Essence Framework) (ÖÇ) Standardına dayalı bir YYY modeli geliştirilmiştir. İlk izlenimlerimiz önerilen modelin yazılım mühendisliği alanına katkıda bulunabilecek nitelikte olduğu, endüstri uygulamaları ve deneysel veriyle desteklenmesi gerektiği yönündedir.

Anahtar Kelimeler: Yazılım yeniden yapılama, Öz Çerçeve, Yazılım gösterimi

Essence Framework Approach to Software Re-engineering

Abstract. Today, software life cycle is shortened and systems developed with current software technologies have already started to be regarded as legacy systems. Thus, this has increased the importance of Software Re-engineering (SRE). The methods, tools and techniques used in SRE have to be independent of technology, platform and business domain. While software teams can use the methods suit to their preferences, they should also be able to adopt agile approaches. Moreover, SRE should be easily adapted to software systems of different size, structure and platforms, and that scale from small to big projects. However, the literature

review cannot present comprehensive and high level solutions to the current problems of SRE practices. In this study, therefore, we propose a SRE model designed and developed according to the guidelines of Essence Framework. Our first impressions are as the proposed model has a potential for contributing to SRE domain, however, it should be supported by empirical evidences and industrial applications.

Keywords: Software Re-engineering, Essence Framework, Software representation

1 Giriş

Nesneye Yönelimli Programlamanın (NYP) endüstride en iyi uygulama (best practice) haline gelmesi ve 1990'lı yıllardan itibaren eski kurumsal yazılımların (legacy system) dönüştürülme ihtiyacıyla birlikte Yazılım Yeniden Yapılamanın (Software Re-engineering) (YYY) önemi gittikçe artmıştır [1]. YYY ile ilgili yöntem, araç ve teknikleri içeren çalışmalar incelendiğinde bunların genel olarak: (a) mevcut yazılımların işlevsel (functional) ve/veya işlevsel olmayan (non-functional) niteliklerinin geliştirilmesi [2] ile (b) eski yazılım sistemlerinin dönüştürülmesinde kullanılan yöntem ve araçlar üzerine odaklanıldığı gözlenmektedir [3, 4]. Ancak son yıllardaki bilgi, iletişim, veri ve yazılım teknolojilerindeki hızlı gelişmeler, bireysel ve kurumsal ihtiyaçları da etkilemiş, beraberinde köklü değişiklikleri gündeme getirmiştir. Örneğin yazılım ömür devri kısalmış, NYP yöntem ve teknikleriyle geliştirilen yazılım sistemleri gün geçtikçe eski yazılım sistemleri (legacy) arasında yer almaya başlamıştır [1, 5, 6].

Kurumların hızlı değişen iş modelleri ve süreçleri, büyük veri ve onun yönetimindeki sorunlar, mobil teknolojiler, bulut bilişim, yeni donanım gereksinimleri, YYY projelerinde karşımıza çıkabilen diğer önemli problem sahalarıdır [7]. YYY'da gidiş-dönüşlü (round-trip), artırımsal, yinelemeli ve yoğun kaynak kullanımını gerektiren yazılım süreçleri bulunmaktadır [8]. Dolayısıyla, bir YYY projesinde benimsenecek yazılım süreç modeli, alan/teknoloji/yöntem bağımsız bir yapıda [9] ve aşağıda belirtilen gereksinimlere de cevap verebilecek özelliklere sahip olmalıdır:

a. Yazılım mühendisliğinin ne (what) ve nasıl yapılmalı (how) konusu kapsamında YYY süreç yönetimi, teknikleri ve araçları; yeniden yapılanacak yazılımın türü, yapısı ve teknolojisinden bağımsız ve ortak bir çerçevede ele alınabilmelidir.
b. Uygulamacılar (practitioner), YYY süreci boyunca her aşama ve durumu tanımlayabilmeli ve bunlara dayalı olarak proje sürecini etkili biçimde takip edebilmelidir.
c. YYY projesi boyunca kullanılan yazılım geliştirme yöntemi ,yazılım takımının ihtiyaçları, deneyimi ve istekleri doğrultusunda, istenilen uygulama (practice), araç ve tekniklerle esnek biçimde değiştirilebilmelidir.
d. Çevik yazılım yöntem ve uygulamalar desteklenebilmeli, YYY projesi süresince kullanılan teknik ve araçların gelişen durumlara göre değiştirilmesi, güncellenmesi veya yeniden oluşturulmasına olanak verecek yapıda olmalıdır.

e. Kullanılan YYY yöntemi, değişik büyüklükte her türlü yazılım sistemine uyarlanabilmeli, küçük çaplı projeden büyüğe doğru kolayca evirilebilen ve ölçeklenebilen niteliklere sahip olmalıdır.

Problem sahaları ve YYY ile ilgili literatür incelendiğinde; SEMAT (Software Engineering Method and Theory) girişimi ve organizasyonu tarafından yazılım geliştirme bilgi alanı için geliştirilen Öz Çerçeve (Essence Framework) (ÖÇ) Standardının söz konusu problemlere çözüm getirebileceği düşünülmektedir [10]. ÖÇ'de yazılım geliştirme yöntemleri ve uygulamaları, çekirdek (kernel) adı verilen yedi bileşenden (Abstract-Level Progress Health Attribute) (Alpha) oluşmakta ve bunlar müşteri (customer), çözüm (solution) ve çaba (endeavor) adlı ilgi sahalarında yer almaktadır. ÖÇ'de temel amaç, yazılım takımları için kendi bilgi, beceri ve deneyimlerine uygun çevik yazılım geliştirme yöntemlerini oluşturabileceği ve uygulayabileceği yöntem, teknik, model ve araçları sağlamaktır. Ayrıca, ÖÇ'e özgü alan bağımlı görsel ve metin tipi olmak üzere iki farklı dil kullanılarak yazılım geliştirme süreçleri formal biçimde tasarlanabilmektedir. Bu bağlamda çalışmamızın yazılım mühendisliği araştırma alanına olan katkıları aşağıdaki gibidir:

a. Literatürdeki ÖÇ uygulama kütüphanesine yönelik olarak YYY uygulamasının ÖÇ rehberliğinde tanımlanması ve gösterimi (essentialize),
b. YYY süreçlerini bütünleşik olarak ele alacak, üst seviyede, biçimsel bir YYY modelinin önerilmesidir.

Bildirinin sonraki bölümlerini çalışmanın kuramsal temellerini oluşturan YYY ve ÖÇ bilgi alanları, araştırma yöntemi, sonuç ve önerileri içeren başlıklar oluşturmaktadır.

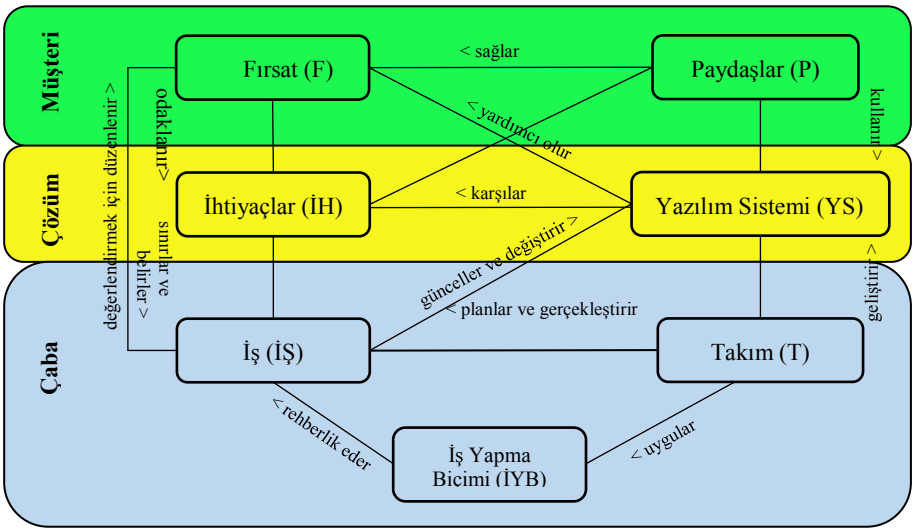
2 Yazılım Yeniden Yapılama

YYY, (a) eski yazılım sisteminin işlevsel olan (functional) ve işlevsel olmayan (non-functional) niteliklerini geliştirme ile (b) yazılıma yeni işlevler kazandırmak amacıyla gerçekleştirilen bir yazılım sürecidir. Sistem mühendisliği kapsamında yeniden yapılama (re-engineering) süreci ise hedeflenen sistemi, mevcut sistemle aynı ya da daha üst mantıksal ve yapısal düzeyde yeniden oluşturma ve böylece bu sistemi sürdürülebilir hale getirme olarak tanımlanabilir [2]. YYY projesi üç aşamadan oluşmaktadır: (a) tersine mühendislik (reverse engineering); (b) yeniden yapılandırma (restructuring) ve (c) ileriye mühendisliktir (forward engineering). Tersine mühendislikte mevcut sistem ve onu oluşturan bileşenler belirlenmekte, aralarındaki ilişkiler incelenmektedir. Bu amaçla mevcut sistemin aynı biçimde ya da daha üst soyutlama düzeyindeki gösterimleri gerçekleştirilmektedir. Yeniden yapılandırmada aşamasında mevcut sistemin işlevleri değiştirilmemekte ve yazılım aynı soyutlama düzeyindeki bir gösterim biçiminden başka bir gösterim biçimine dönüştürülmektedir. İleriye mühendislik aşamasında ise üst düzey soyutlama düzeyinde yeniden gösterimi yapılan sistem tasarım, geliştirme ve test süreçlerinden geçirilmektedir. Öte yanda program dönüştürme (program transfor-

mation) ve program gösterimi (program representation) YYY süreçlerinde gerçekleştirilen temel etkinlikler arasındadır. Program gösteriminde soyut söz dizim ağaçları (abstract syntax tree), ayrıştırma ağaçları (parse tree), çizge (graph) vb tekniklerden bir ya da birkaçı kullanılabilmektedir. Program dönüştürme etkinliği gereksinim ve çeşitli ölçütlerine bağlı olarak (soyutlama düzeyleri, hedef mimari, yazılım dili vb.) program göçü (migration), program çevirisi (translation) vb. yazılım etkinliklerini içerebilmektedir [3, 4].

3 Öz Çerçeve (Essence Framework)

Yazılım geliştirme süreçlerinin formal biçimde tanımlanması, bunların karşılaştırılması ve yazılım takımlarına uyarlanması için geliştirilen ÖÇ'nin temelini Çekirdek (Kernel) adı verilen temel yapı oluşturmaktadır [10]. Çekirdek'in ana bileşenleri olan Alfa'lar (Alphas), Etkinlik Uzayları (Activity Spaces) ve Yetkinlikler (Competencies), Müşteri (Customer), Çözüm (Solution) ve Çaba (Endeavor) olarak ifade edilen üç ayrı ilgi alanı içerisinde yer almaktadır (Şekil 1). Alfa'lar, yazılım geliştirme projelerinde ele alınması gereken önemli bileşenleri temsil etmektedirler. Bunlar aynı zamanda yazılım geliştirme projesinin ilerleme ve gelişimini farklı seviyelerde gösteren durumları içermektedir (state). Etkinlik Uzayları, YM'nin etkinlik tabanlı boyutunu simgelemekte ve yazılım geliştirme faaliyetlerini temel ilgi alanlarına (müşteri, çözüm ve çaba) göre gruplamaktadır. Yazılım geliştirme etkinliklerinin yerine getirilebilmesi için edinilmesi gerekli olan bilgi, beceri ve yetenekler ise Yeterlilikler adı verilen bileşenle gösterilmektedir.



Şekil 1. Öz Çerçeve'nin Alpha Bileşenleri ([10]'dan uyarlanmıştır).

ÖÇ yaklaşımının ana amaçları şunlardır:

a. Yazılım geliştirme yöntemleri ve uygulamalarının formal biçimde ortak bir temelde bütünleştirilmesi ve gösterimi,

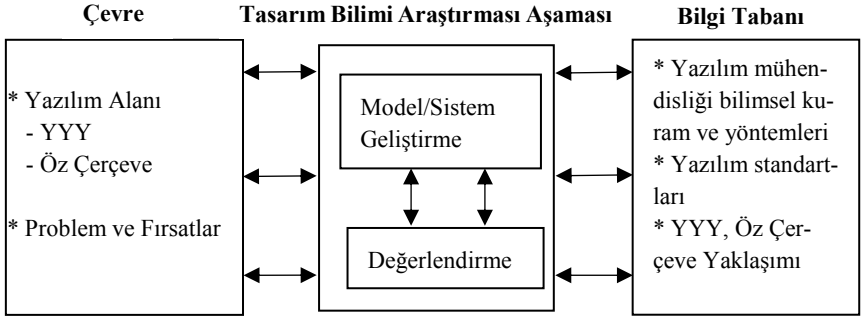
b. Çevik yaklaşım doğrultusunda yazılım takımlarının kendilerine özgü yazılım yöntemlerini dinamik ve esnek biçimde oluşturabilmesi ve kullanabilmesi,

c. Durumlara (state) bağlı olarak yazılım geliştirme süreçlerinin her aşamasının sağlıklı bir şekilde izlenebilmesi,

d. Yazılım geliştirme çalışma alanındaki araştırmalar ve endüstri uygulamaları arasında kuramsal ve uygulama boyutunda köprü vazifesi görmektir.

4 Yöntem

Bu araştırma, Tasarım Bilimi Araştırma Yöntemi (TBAY) (Design Science Research) [12, 13] çerçevesinde yürütülmüş, çalışmanın kuramsal temellerini YYY ve ÖÇ bilgi alanları oluşturmuştur. TBAY'de mühendislik, bilişim sistemleri ve yazılım alanındaki problem alanlarına yönelik, belirli işlev ve özelliklere sahip sistem ve modeller geliştirilir. Ancak, aynı zamanda bunların analizi, tasarımı ve geliştirilmesine yönelik bilimsel bilgi birikiminin oluşturulması da ana amaçtır. TBAY dayalı bir araştırma projesinde, gerçek hayat problemlerinden hareket edilmekte, araştırma yapar gibi araç, yöntem, model veya kuram geliştirilmekte, iyileştirilmekte ya da test edilmektedir. Bu kapsamda çalışmamızın ana çıktısı ÖÇ dayalı YYY modelidir. TBAY'de yinelemeli ve gidiş-dönüşlü araştırma etkinliklerin bulunduğu; (a) problem alanı, (b) tasarım bilimi araştırması ve (c) bilimsel bilgi tabanından oluşan üç ana bileşen bulunmaktadır (Şekil 2).



Şekil 2. TBAY Temel Bileşenleri, ([12]'den uyarlanmıştır).

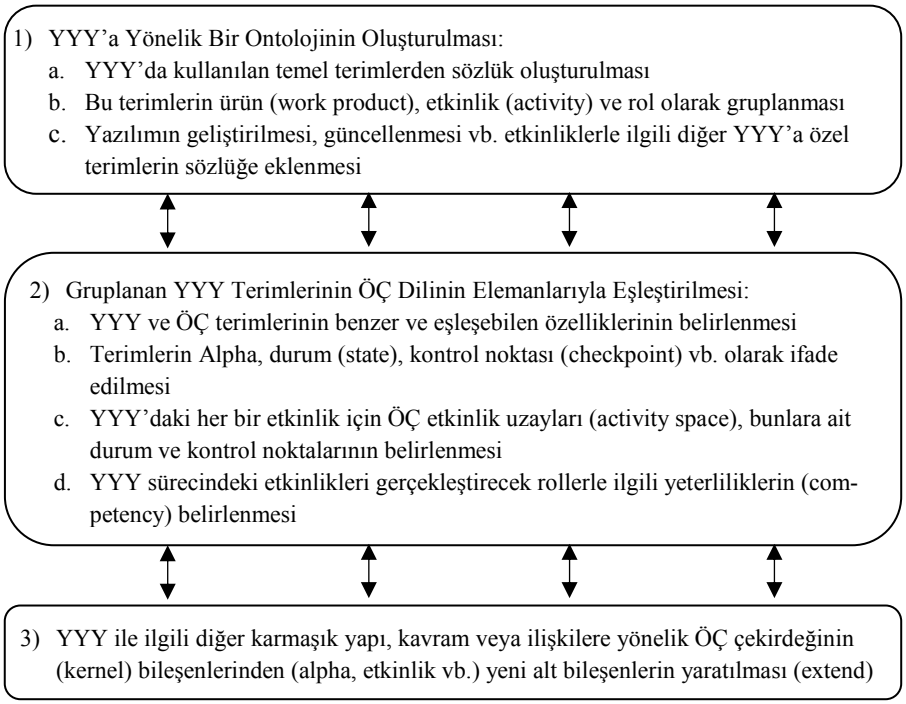
Çalışmamızda ÖÇ dayalı YYY modelinin geliştirilmesi iki aşamada olmuştur. Birinci aşamada YYY uygulamaları, bunlarla ilgili ürün ve kavramlar ÖÇ'deki Alpha'larla eşleştirilmiştir. İkinci aşamada eşleştirilen bileşenler EssWork Practice Workbench IDE kullanılarak ÖÇ'nin alan bağımlı diliyle gösterimi gerçekleştirilmiştir.

4.1 YYY Kavramlarının ÖÇ Bileşenleriyle Eşleştirilmesi

YYY ÖÇ ile eşleştirilmesi, Şekil 1'deki gibi ÖÇ bileşenlerinin kendi aralarındaki ilişkisel yapı, etkileşimler ve endüstrideki uygulamalardan hareket edilerek aşağıdaki genel adımlar izlenerek gerçekleştirilmektedir:

a. YYY sürecinde çıktı olarak ortaya konulan iş ürünleri (work product) ÖÇ'deki Alpha'larla,
b. YYY sürecinde gerçekleştirilen etkinlikler ÖÇ'deki Etkinlik Uzaylarıyla
c. YYY sürecinde üstlenilen roller ise ÖÇ'deki Yeterliliklerle eşleştirilmiştir.

Kavramsal eşleştirme yönteminin detaylandırılmış biçimi Şekil 3'te gösterilmiştir.



Şekil 3. YYY kavramlarının ÖÇ bileşenleriyle eşleştirilme yöntemi

Öncelikle YYY ile ilgili literatür incelenerek çalışmalarda yer alan ve sıklıkla tekrarlanan kavramlar belirlenmiş ve eşleştirme sürecine başlanmıştır. Ortak özelliklere sahip YYY kavramları eşleştirilmiş, diğerleri ÖÇ bileşenlerinden türetilerek (extend) yeni bileşen olarak tanımlanmıştır. Ancak, her iki bilgi alanında (ÖÇ ve YYY) ve "test and acceptance" ile "delivery" etkinlik uzaylarında yer alan bazı kavramlar dönüştürülememiştir (Tablo 2). Kavramsal eşleştirme sürecinin sonucu Tablo 1'de özetlenmiştir.

Tablo 1. ÖÇ ve YYY'da yer alan kavramların eşleştirilmesi

YYY Sürecindeki Bütün Anahtar Kavramlar	**Kavram Grubu**	**Ortak Özellikler Mevcut mu?**	**ÖÇ Karşılığı Olan Bileşen**
1. Problem definition	Etkinlik	Evet	Understand stakeholder needs
1.1 Existing system	İş Ürünü	Evet	Software system
2. Requirement Analysis	Etkinlik	Evet	Understand requirements
2.1. Target system	İş Ürünü	Evet	Software system
3. Implementation	Etkinlik	Evet	Implement system
3.1. Reverse engineering	Etkinlik	Hayır	Alpha'dan "extend" edilecek
3.1.1. Data structure diagrams	İş Ürünü	Evet	Work product
3.1.2. Abstract syntax trees	İş Ürünü	Evet	Work product
3.1.3. Parse trees	İş Ürünü	Evet	Work product
3.1.4. Graphs	İş Ürünü	Evet	Work product
3.1.5. Program structure Diagrams	İş Ürünü	Evet	Work product
3.1.6. Program representation	Alt etkinlik	Hayır	Alpha'dan "extend" edilecek
3.2 Re-structuring			
3.2.1. Program restructuring	Alt etkinlik	Hayır	Alpha'dan "extend" edilecek
3.2.2. Data re-structuring	Alt etkinlik	Hayır	Alpha'dan "extend" edilecek
3.3. *Forward engineering*	Etkinlik	Hayır	Alpha'dan "extend" edilecek
3.3.1. Code translation	Alt etkinlik	Hayır	Alpha'dan "extend" edilecek
3.3.2. Program transformation	Alt etkinlik	Hayır	Alpha'dan "extend" edilecek
3.2.3. Refactoring	Alt etkinlik	Hayır	Alpha'dan "extend" edilecek
3.2.4. Manual fine tuning	Alt etkinlik	Hayır	Alpha'dan "extend" edilecek
4. Test and acceptance	Etkinlik	Evet	Test system
5. Delivery	Etkinlik	Evet	Deploy system

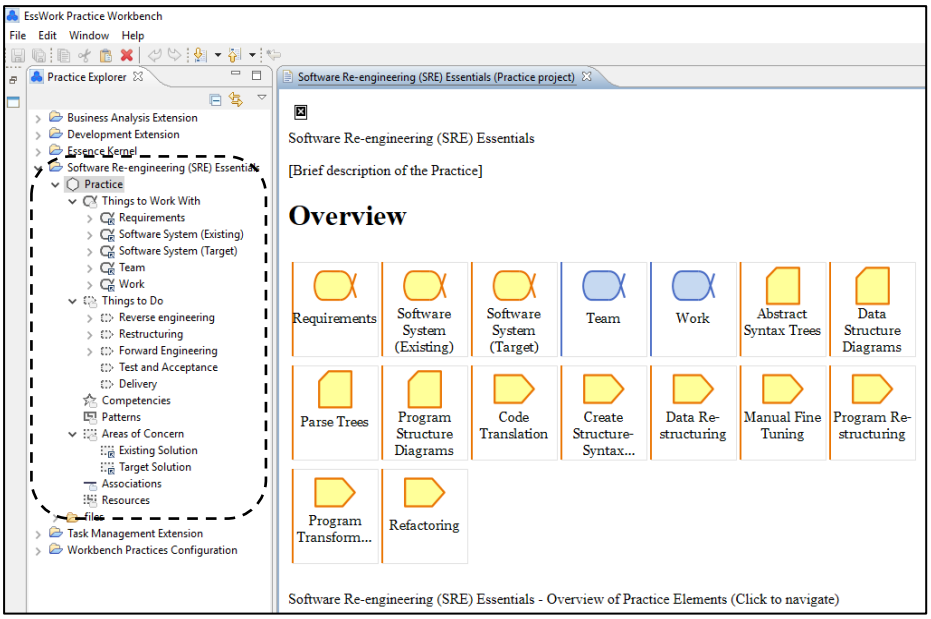
4.2 YYY Uygulamasının ÖÇ Ortamında Tanımlanması ve Gösterimi

Tablo 1'de eşleştirilen kavramsal yapıların, ÖÇ ortamında tasarımı yapılan YYY bileşenleri, etkinlikler ve ürünler Tablo 2'de gösterilmiştir.

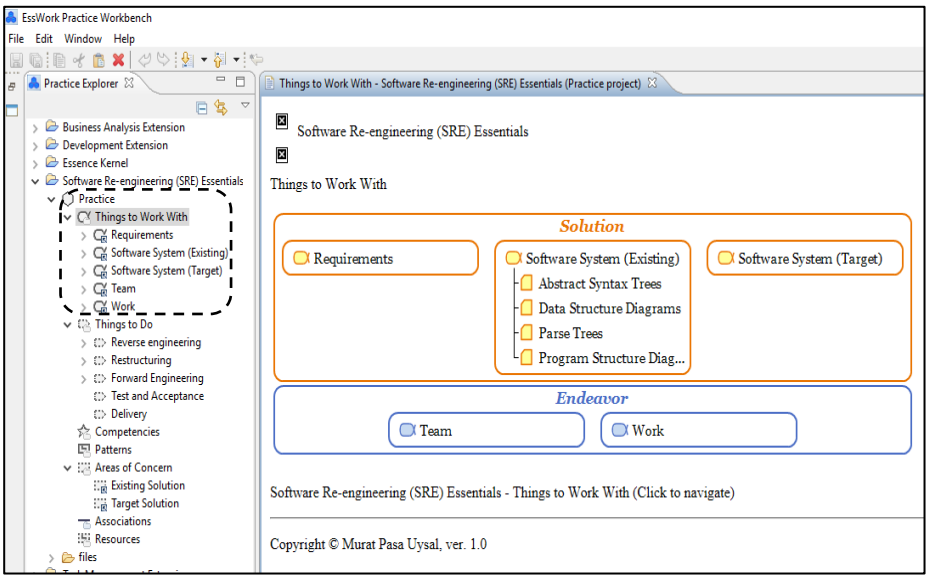
Tablo 2. YYY bileşenleri, etkinlikler ve ürünler

S. Nu	ÖÇ Bileşeni ve Türü	Yeniden Yapılama Uygulamasında Etkinlikler ve Ürünler
1	Alpha	"Requirements", "software system" (existing), "software system (target)", "work", "team"
2	Work Product	"Abstract syntax tree", "parse tree", "graph", "program structure diagram", "data structure diagram"
3	Activity Space ("reverse engineering")	"create structure and syntax diagrams"
4	Activity Space ("restructuring")	"data re-structuring", "program re-structuring"
5	Activity Space ("forward engineering")	"code translation", "program transformation", "refactoring", "manual fine tuning"
6	Activity Space (test and acceptance)	(dönüştürülememiş ve sonraki çalışmalara bırakılmıştır)
7	Activity Space (delivery)	(dönüştürülememiş ve sonraki çalışmalara bırakılmıştır)

Mevcut ve hedef yazılım sistemini (Alpha'ları) oluşturan ÖÇ bileşenleri "program structure diagram", "data structure diagram" vb. iş ürünleriyle eşleştirilmiştir. "reverse engineering", "restructuring", "forward engineering" ise ana etkinlik uzaylarını teşkil etmektedir. "data re-structuring" ve "program re-structuring" bu etkinlik uzayları altında yer alan ana etkinliklerdendir. Tablo 2'deki YYY bileşenleri daha sonra "EssWork Practice Workbench" IDE ortamına aktarılarak ÖÇ gösterimi gerçekleştirilmiştir (Şekil 4). Okuyucuya fikir vermesi amacıyla "Software Re-engineering (SRE) Essentials" adlı ÖÇ projesinin genel görünümü Şekil 4'te, bunun altında yer alan Alpha'lar ("Things to Work With") vb. bileşenler ise Şekil 5'te gösterilmiştir. Yer ve kapsam sınırlaması dolayısıyla tasarım, eşleştirme ve geliştirme ortamındaki işlemlerin ayrıntıları sonraki çalışmalara bırakılmıştır.



Şekil 4. YYY uygulamasının genel görünümü ve ÖÇ bileşenleri



Şekil 5. YYY uygulamasında Alpha'lar, iş ürünleri ve ilgi alanları

5 Tartışma

Bu çalışmada, bir YYY projesine ait bileşenlerin; yazılımın türü, platform ve teknolojisinden bağımsız olarak ÖÇ tabanlı gösterimi ve modellemesi gerçekleştirilmiştir. ÖÇ yaklaşımının merkezinde bir yazılım sistemi yer almakta ve doğal olarak yazılım süreçleri, etkinlikleri ve bunlara ilişkin ürünler bu yazılıma göre evirilmektedir. Bundan farklı olarak YYY'da ise mevcut yazılım ve hedef yazılım olmak üzere çözüm ilgi alanında iki sistemin ele alınması gerekmektedir. Bu yönüyle araştırmamız iki farklı ilgi alanını içeren Giray vd. [14]'nin çalışmasıyla benzer nitelikler taşımaktadır. Ancak, onlardan farklı olarak çalışmamızda aynı anda iki yazılım sisteminin ele alınmasında bazı güçlüklerin olduğu gözlenmiştir. Dolayısıyla ÖÇ'nin kavramsal yapısında ve tasarım ortamında YYY'a yönelik yeni eklemelerin ve güncellemelerin yapılması gerektiği düşünülmektedir.

YYY'da mevcut yazılım; "legacy" olarak ifade edilen eski bir yazılım ise farklı yöntem, nesneye tabanlı geliştirilmiş bir yazılım ise daha farklı yazılım geliştirme yöntem ve tekniklerinin benimsenmesi gerekmektedir. Bu bağlamda ÖÇ'nin, yazılım geliştirme yönteminden bağımsız biçimde, kavramsal ve uygulama boyutunda üst seviyede yaklaşım sunduğu söylenebilir [10]. Bu durumun bildirinin giriş bölümünde belirtilen söz konusu soruna belirli ölçülerde çözüm getirdiği düşünülmektedir. ÖÇ hakkında gözlenen bir diğer önemli konu ise sahip olunan kuramsal alt yapı, araç ve teknikler aracılığıyla her türlü yazılım geliştirme süreçlerinin derinlemesine inceleme olanağını sunmasıdır. Bu bağlamda ÖÇ'nin, kuramsal, deneysel ve uygulamalı yazılım mühendisliği çalışmalarına önemli katkılar sağlayabileceği değerlendirilmektedir [11].

Çalışmada araştırmacıların ilgilendiği diğer bir konu ise yöntem mühendisliği kapsamında ÖÇ'nin YYY uygulamalarını ne ölçüde modelleyebileceği ya da temsil yeteneğinin ne ölçüde güçlü olduğudur [14]. Genel olarak sorun yaşanmamakla birlikte tasarımın bazı detaylarında güçlükler gözlenmiştir. Örneğin Practice Workbench kullanılırken YYY etkinliklerinin bitiş ölçütü olarak sadece çekirdekte (kernel) yer alan "software system" belirtilebilmiş, YYY'nın mevcut ve hedef yazılım bileşeni referans gösterilememiş, modelin geçerlemesi sırasında çeşitli hatalar alınmıştır. Bu durum Practice Workbench aracının bir eksikliği olarak değerlendirilmekte, aynı anda iki farklı yazılımın ele alınmasını sağlayacak yeni eklemelerin yapılması gerektiği gözlenmektedir.

5.1 Çalışmanın Sınırlılıkları

ÖÇ'e dayalı YYY modelinin zaman ve kapsam sınırlılıklarından dolayı eylem araştırması, durum çalışması vb. deneysel yöntemlerle sınanması mümkün olmamıştır. Bu bağlamda çalışma sonuçlarının genellenebilirliğinin sınırlı düzeyde olduğu söylenebilir. Araştırmanın iç geçerliliğini tehdit edebilecek faktörler ÖÇ literatürü incelenerek ve alan uzmanlarına başvurularak giderilmeye çalışılmıştır. ÖÇ'e dayalı tasarımda sınırlı kaldığı düşünülen diğer konular aşağıdaki gibidir:

a. Practice Workbench ortamından kaynaklanan nedenlerden dolayı hedef yazılımla ilgili Alpha'lara ait durum (state), kontrol noktası (checkpoint) vb. yazılım kontrol ölçütleri modellenememiştir.
b. Bir YYY projesindeki yazılım etkinliklerini gerçekleştirmekle sorumlu rollere ilişkin yeterlilikler (competency) belirlenememiştir.
c. "test and acceptance" ve "delivery" etkinlik uzaylarıyla ilgili ÖÇ tasarımları sonraki çalışmalara bırakılmıştır.

6 Sonuç ve Öneriler

Günümüzde teknolojideki baş döndürücü gelişmeler yazılım mühendisliği alanını etkilerken her geçen gün yeni ve önemli değişiklikleri de gündeme getirmektedir. Yazılım ömür devri kısalmış, kurum ve müşterilerin yazılım sistemlerinden beklentileri nitel ve nicel olarak artmıştır. Buna ek olarak maliyet etkinlik ve finansal açıdan konuya yaklaşıldığında ise işletmeler sahip oldukları yazılımları mümkün olduğu kadar ellerinde tutmayı istedikleri, bu arada her türlü işlevsel ve teknolojik ihtiyaçlarının karşılanmasını bekledikleri gözlenmektedir. Bu durum, YYY'nın önemini vurgularken onun endüstrideki uygulama sayısı ve çeşitliliğini de artırmıştır. YYY'da kullanılan yazılım geliştirme yöntemleri alan, teknoloji ve uygulamadan bağımsız niteliklere sahip olması zorunlu olmuştur. Çevik yaklaşımın YYY süreçlerinde kullanım gerekliliği, yazılım takımlarının kendi gereksinimlerine göre yöntem seçebilme ihtiyaçları, YYY süreçlerinin üst düzey soyutlama düzeyinde temsil edilebilmesini zorunlu kılmaktadır.

Söz konusu probleme yönelik olarak çalışmamızda TBAY ilke ve uygulamaları doğrultusunda bir YYY modeli geliştirilmiştir. Bu amaçla ÖÇ Standardının kavramsal alt yapısı ve alan bağımlı grafik dili kullanılmıştır. Yer, kapsam ve geliştirme ortamından kaynaklanan sınırlılıklardan dolayı bazı konular çalışma dışında tutulurken bir bölümü ise gelecek araştırmalara bırakılmıştır. İlk izlenimlerimiz geliştirilen modelin yazılım mühendisliği alanına katkıda bulabilecek nitelikte olduğu, ancak, endüstri ve deneysel uygulamalarla desteklenmesi gerektiği yönündedir. Bildirimiz; (a) EssWork Practice Workbench ortamının, YYY ihtiyaçlarına cevap verebilecek nitelikte güncellenmesi ile (b) çalışma bulguları ve sınırlılıklarını dikkat alan ÖÇ ve YYY'a yönelik yeni araştırmaların yapılması önerileriyle son bulmaktadır.

Kaynakça

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, DOI: 10.1002/smr.548, 2011.
2. Tahvildari, L., Kontogiannis, K. & Mylopoulos J. "Quality-driven software reengineering", The Journal of Systems and Software, 66, s.225-239, 2003.
3. Seacord R.C., Plakosh D., Lewis G.A. "Modernizing legacy systems: software technologies, engineering processes, and business practices". Addison-Wesley, USA, 2003
4. Birchall C. "Re-engineering legacy software". Manning Publications, 2016.
5. Valenti, S. "Successful software reengineering". IGI Global, USA, 2002.

6. Rada, R. Reengineering Software: How to reuse programming to build new, state-of-the-art software, Glenlake Publishing Co., 2005.
7. Furda A., Fidge C., Barros A., Zimmermann O. "Reengineering data-centric information systems for the cloud-a method and architectural patterns promoting multitenancy", Software Architecture for Big Data and the Cloud, DOI: 10.1016/B978-0-12-805467-3.00013-2.
8. Uysal, M.P, Mergen E.A. Yazılım yeniden yapılamaya yönelik model güdümlü ve kaliteye yönelimli süreç modeli, 9. Ulusal Yazılım Mühendisliği Sempozyumu, 2015.
9. Uysal, M.P. ve 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, s.971-979.
10. OMG (Object Management Group), "Essence-Kernel and language for software engineering methods, *Document ID: SMSC/15-12-02*, 2015.
11. Uysal MP, Giray G. "Yazılım mühendisliği araştırmalarında Öz Çerçeve (Essence Framework) yaklaşımı". 11. Ulusal Yazılım Mühendisliği Sempozyumu, 18-20 Eylül 2017, Alanya, Türkiye.
12. Hevner, A. & Chatterjee S. Design Research in information systems, Integrated Series in Information Systems, 22, DOI 10.1007/978-1, 2010.
13. Vaishnavi, V.K. & Kuechler W.J. Design Science Research methods and patterns: innovating ınformation and communication technology, USA, Auerbach Publications, Taylor & Francis Group, 2008.
14. Eray Tüzün E., Giray G., Tekinerdogan B, Macit Y. "Modeling software product line engineering with essence framework". Bilişim Teknolojileri Dergisi, 11(1), 2018