Dokuz Eylül Üniversitesi-Mühendislik Fakültesi
Fen ve Mühendislik Dergisi
Cilt 19 Sayı 55.1 (Özel Sayı) Ocak 2017

Dokuz Eylul University-Faculty of Engineering
Journal of Science and Engineering
Volume 19 Issue 55.1. (Special Issue) January 2017

DOI: 10.21205/deufmd.2017195533

Sentez Tabanlı Yazılım Mimarisi Tasarım Yaklaşımının Essence Çerçevesiyle Modellenmesi

Görkem GİRAY*[1], Bedir TEKİNERDOĞAN[2], Eray Tüzün[3]

[1]Bağımsız Araştırmacı, İzmir
[2] Wageningen Üniversitesi, Bilişim Teknolojisi Bölümü, Wageningen
[3] HAVELSAN, Teknoloji ve Akademi Direktörlüğü, Ankara

(Alınış / Received: 07.06.2016, Kabul / Accepted: 12.08.2016,
Online Yayınlanma / Published Online: 09.01.2017)

Anahtar Kelimeler
Sentez tabanlı yazılım mimarisi tasarımı, Mimari tasarım, Essence çerçevesi

Özet: Yazılım mimarisi tasarımı, yazılım geliştirme sürecindeki çok önemli ve zor bir aşamadır. Paydaşların çelişen amaçlarını yönetmek ve ilgili gereksinimlerden mimari soyutlamalar türetmek önemlidir. Ayrıca mevcut çözüm alanı bilgisi üzerinde temellendirilmiş semantik açıdan zengin ara ürünler (artifact) üretmek dikkat edilmesi gereken bir husustur. Elde edilen mimari ara ürünler yazılım geliştirme sürecinin geri kalanına kılavuzluk eder ve planlamayı kolaylaştırır. Yakın dönemde yazılım geliştirme yöntemlerinin ve etkinliklerinin eşlenebileceği, yazılım mühendisliğine soyut ve genel bir bakış sunan Essence çerçevesi önerilmiştir. Bu çalışmada, sentez tabanlı yazılım mimarisi tasarımı etkinliklerinin Essence çerçevesiyle eşleştirilmesi hakkında bilgi verilmektedir. Böylece, yazılım mühendisliğinin soyut ve genel bir modeli kullanılarak yazılım mimarisi tasarım etkinliklerinin açıklanması amaçlanmıştır. Ayrıca, sentez tabanlı yazılım mimarisi tasarım etkinlikleri için durum tabanlı etkinlik izleme düzeneği önerilmiştir. Essence çerçevesi ve sentez tabanlı mimari tasarımı yaklaşımı hakkında elde edilen deneyim paylaşılmaktadır.

Modeling Synthesis-based Software Architecture Design Approach Using Essence Framework

Keywords
Synthesis-based software architecture design, Architectural design, Essence framework

Abstract: Software architecture design is a pivotal yet a difficult phase in software development process. It is important to manage conflicting goals of the stakeholders and derive architectural abstractions from the relevant requirements. Moreover, it is significant to produce semantically rich artifacts based on the existing solution domain knowledge. Resulting architectural artifacts guides the rest of the software development process and facilitates planning. Recently the Essence framework has been proposed to provide an abstract and general view of software engineering on which software development methods and activities can be mapped. In this work, a mapping of the synthesis-based software architecture design activities to the Essence

framework is presented. By doing so, these activities are explained using an abstract and general model of software engineering. Moreover, a state-based activity tracking mechanism for synthesis-based software architecture design activities is proposed. The lessons learnt about the Essence framework and the synthesis-based architecture design approach are reported.

*Sorumlu yazar: gorkemgiray@gmail.com

1. Giriş

Yazılım mimarisi tasarımı, yazılım geliştirme sürecinin önemli etkinliklerinden biridir. Mimari tasarım, kendisinden sonra gelen birçok etkinlik için yönlendirici bir etkiye sahiptir; örneğin projenin daha iyi planlanmasını ve yönetilmesini sağlamak, yazılım geliştiricileri gerçekleştirim için yönlendirmek gibi. Mimariyi tasarlarken gereksinimlerin uygun seviyede soyutlanması ve bu soyutlamalardan mimari açıdan önemli yönlerin ortaya çıkarılması önemlidir. Bunlar yapılırken paydaşların çelişen amaçları irdelenerek ortaya tutarlı bir amaçlar bütünü konulmalıdır.

Sentez-tabanlı yazılım mimarisi tasarımı [1], mimari açıdan önemli problemlerin irdelenmesini, bu problemlerin alt problemlere ayrılarak çözülmesini ve elde edilen çözümlerin bütünleştirilerek mimari tasarıma ulaşılmasını önermektedir. Bunun yanında, alan güdümlü modelleme [2] gibi yaklaşımlar alan bilgisinin tasarım süreci için önemini vurgulamaktadır; başka bir deyişle mimari tasarımda sadece gereksinimler değil aynı zamanda ilgili alandaki bilgi birikimi modellenerek oluşturulan çözüm uzayı da göz önüne alınır.

Essence çerçevesi [3], yazılım mühendisliği için ortak bir dil ve alan modeli sağlamayı amaçlamaktadır. Essence çerçevesi yazılım geliştirme yöntemlerini modellemek için biçimsel bir temel oluşturmaktadır. Böylece bir yöntemin daha iyi anlaşılması, öğrenilmesi, başka yöntemlerle karşılaştırılması ve birleştirilmesi hedeflenmektedir.

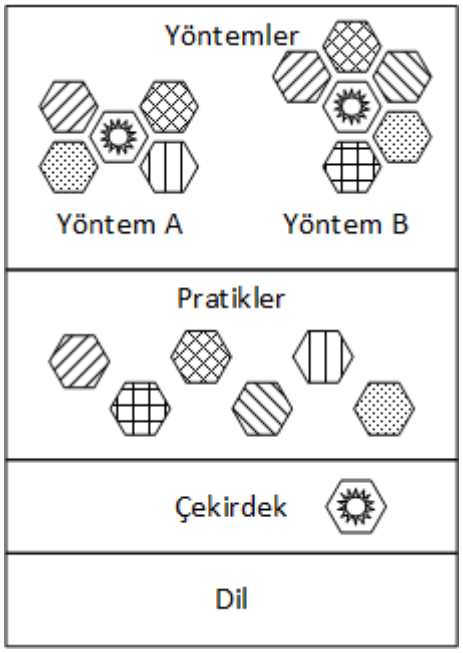
Bu makalede, sentez-tabanlı yazılım mimarisi tasarımı yaklaşımının Essence çerçevesiyle oluşturulmuş bir modeli sunulmuştur. Bu model sayesinde bu mimari tasarım yaklaşımının yazılım geliştirme süreci içindeki yeri Essence çerçevesi bakış açısıyla somutlaştırılmıştır. Essence çerçevesinin sunduğu kavramlar ve bu kavramları genişletme olanaklarıyla mimari tasarım sürecinin izlenebilmesi için bir yöntem önerilmiştir. Bu yöntem, bir yazılım geliştirme projesindeki önemli boyutların (alfalar ve alt alfalar) durum bazlı izlenmesi ve bu durumların değişip değişmediğinin kontrol listeleriyle anlaşılması üzerine temellendirilmiştir. Alt alfalar, alt alfaların durumları ve bu durumların kontrol listeleri her proje için özelleştirilebilir; böylece her projenin farklı koşulları doğrultusunda yeniden düzenlenebilir.

2. ve 3. bölümlerde sırasıyla Essence çerçevesi ve sentez tabanlı yazılım mimarisi tasarımı hakkında özet bilgi verilmiştir. 4. bölümde sentez-tabanlı yazılım mimarisi tasarımı, Essence çerçevesi kullanılarak modellenmiş ve elde edilen modelin tasarım sürecinin ilerlemesinin takip edilmesinde nasıl kullanılabileceği gösterilmiştir. 5. bölümde ilgili çalışmalar özetlenmiştir. Makale, elde edilen sonuçların özetiyle sonlanmaktadır.

2. Essence Çerçevesi

Essence çerçevesi, SEMAT (Software Engineering Method and Theory) girişimi tarafından yazılım geliştirmek için kullanılan yöntemlerin standartlaştırılması amacıyla oluşturulmuştur. Şekil 1'de gösterildiği gibi çerçeve dört temel bileşenden oluşmaktadır [4]:

- *Dil:* Çekirdeği, pratikleri ve yöntemleri tanımlamak için kullanılan dil ögelerini ve bu ögeler arasındaki ilişkileri içerir [3]. Dil ögelerine örnek olarak alfa, alfa durumu, iş ürünü, etkinlik uzayı, etkinlik ve yetkinlik; bu ögeler arasındaki bir ilişkiye de bir etkinliğin bazı yetkinlikleri gerektirmesi verilebilir [3].
- *Çekirdek:* Yazılım geliştirme sürecinde bulunan, pratiklerden ve yöntemlerden bağımsız olarak bu süreçteki önemli kavramları içerir [3]. Bu kavramları ifade etmek için alfa kavramı kullanılır [3]. Alfalar, (1) yazılım geliştirme sürecinin temelinde bulunan ortak kavramları temsil eder; (2) yazılım geliştirme sürecinin nasıl ve ne kadar sağlıklı ilerlediğinin izlenmesini ve değerlendirilmesini sağlar; (3) yazılım mühendisliğindeki yöntemleri ve pratikleri tanımlamak için bir kavramlar temeli sağlar [3].
- *Pratikler:* Bir pratik, yazılım geliştirme sürecinin bir yönünün nasıl ele alınacağını tanımlar ve net bir amaç ve bu amaca ulaşabilmek için çeşitli yönlendirmeler içerir. Pratik için bir örnek olarak gereksinimlerin belirtimi için kullanıcı hikayelerinin kullanımı gösterilebilir [5].
- *Yöntemler:* Bir yöntem ise çekirdek ve bunun etrafında konumlandırılan pratiklerden oluşur [3].



Şekil 1. Yöntem mimarisi [4]

2.1. Çekirdek

Çekirdek, yazılım mühendisliğinin özündeki kavramları pratiklerden bağımsız olarak yalın bir biçimde temsil eder [3]. Çekirdek, Essence mimarisinin dil katmanındaki dil ögeleri kullanılarak tanımlanmıştır. Çekirdekteki kavramlar üç ilgi alanında sınıflandırılmıştır:

1. *Müşteri:* Üretilecek yazılım sisteminin kullanımı ile ilgilidir.
2. *Çözüm:* Yazılım sisteminin belirtimi ve geliştirimi ile ilgilidir.
3. *Çaba:* Takım, yapılacak işler ve iş yapma biçimleri ile ilgilidir.

Bu üç temel ilgi alanı, üç tür kavram içermektedir:

1. *Alfalar:* Üzerinde çalışılacak temel kavramların temsilleri. Alfa kelimesi, Essence belirtiminde "Abstract-Level Progress Health Attribute" ifadesinin kısaltması olan "alpha" kelimesine karşılık gelmektedir.
2. *Etkinlik uzayları:* Yapılacak temel şeylerin temsilleri.

3. *Yetkinlikler:* Yazılım mühendisliği işi için gereksinim duyulan temel yetkinliklerin temsilleri.

Bu çalışmada, yer sınırlaması nedeniyle alfalar ve etkinlik uzayları kapsama dahil edilmiştir.

2.1.1. Alfalar

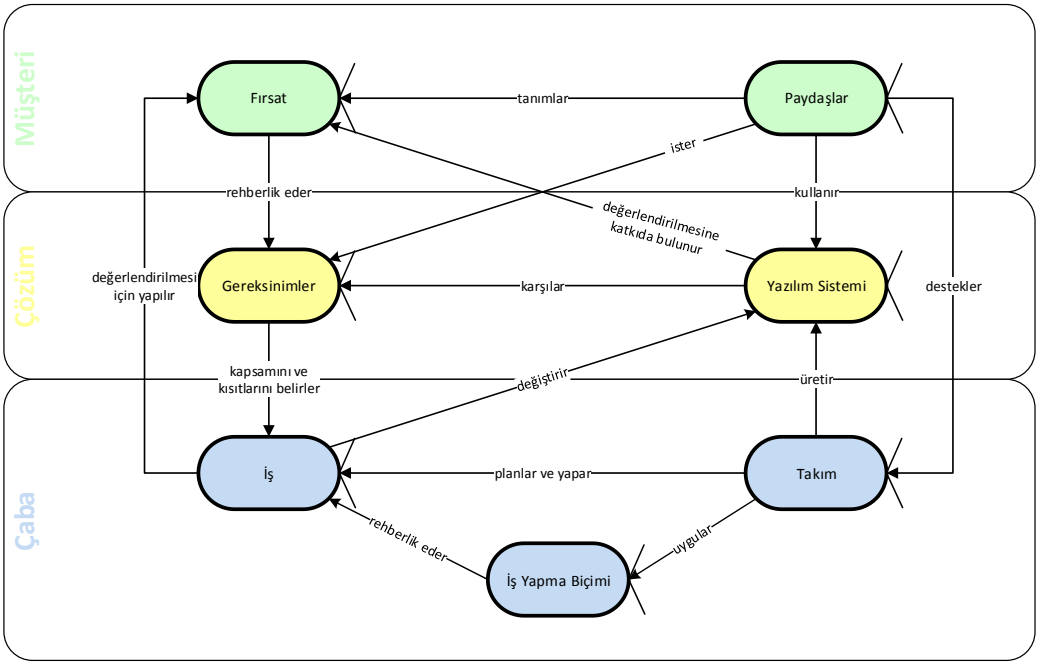
Çekirdek, yazılım geliştirme sürecinin temelinde bulunan kavramları (alfalar) ve bu alfalar arasındaki ilişkileri tanımlamaktadır. Bu alfaların ortak bir terminoloji oluşturarak takım içindeki iletişimi daha verimli kılmada, zorlukları çözümlemede ve yazılım geliştirme sürecinin verimini arttırma konusunda yardımcı olabileceği belirtilmektedir.

Çekirdekte bulunan alfalar ve bu kavramlar arasındaki ilişkiler Şekil 2'de gösterilmiştir [3]. Yedi temel alfa üç ilgi alanında (müşteri, çözüm ve çaba) gösterilmiştir.

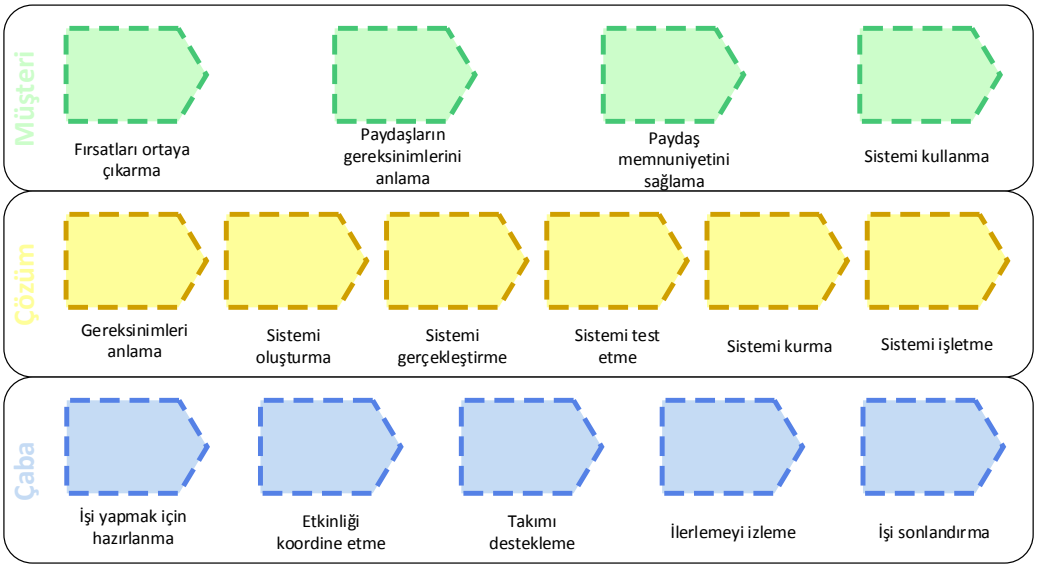
1. *Fırsat:* Yazılım sistemini geliştirmek ya da değiştirmek için oluşan durum, bir başka deyişle paydaşların iş gereksinimlerinin tüm takım tarafından anlaşılmış halini temsil eder. Fırsat, yazılım sisteminin geliştirilmesi için detaylı gereksinimlerin oluşturulmasına rehberlik eder.
2. *Paydaşlar:* Yazılım sistemini etkileyecek ya da yazılım sistemi tarafından etkilenecek kişileri, grupları, organizasyonları temsil eder. Paydaşlar, (1) fırsatı tanımlar; (2) detaylı gereksinimlerin belirlenmesinde temel bilgi kaynağı olarak davranır; (3) yazılım sistemini doğrudan kullanır ya da bu kullanımdan bir şekilde etkilenir; (4) takımı yazılım geliştirme sürecinde destekler.
3. *Gereksinimler:* Yazılım sisteminin yapacaklarını ve bunları hangi koşullar altında ve kalitede yapacağını tanımlar. Gereksinimler yapılacak işlerin kapsamını ve kısıtlarını belirler.
4. *Yazılım sistemi:* Yazılımdan, donanımdan ve veriden oluşan sistemdir. Yazılım sistemi (1) fırsatın değerlendirilmesine katkıda bulunur; (2) gereksinimleri karşılar.
5. *İş:* Bir sonucu elde etmek için yapılan zihinsel ya da fiziksel etkinliktir. İşler, (1) fırsatın tanımladığı duruma uygun olarak belirlenir; (2) yazılım sistemini değiştirir.
6. *Takım:* Bir yazılım sisteminin geliştirilmesinde ve bakımında görev alan kişilerin oluşturduğu gruptur. Bir ya da birden fazla takım (1) yazılım sistemini geliştirir ya da değiştirir; (2) işleri planlar ve yapar; (3) işleri yaparken çeşitli iş yapma biçimlerini kullanır.
7. *İş yapma biçimi:* Takımın gerçekleştirdiği etkinlikler ve kullandığı araçlardır. İş yapma biçimi, yapılan işlerin nasıl yapılacağına rehberlik eder.

2.1.2. Etkinlik Uzayları

Etkinlik uzayları, yazılım mühendisliğinin etkinlik tabanlı bir görümünü sağlar ve alfaları tümler [3]. Etkinlik uzaylarındaki etkinlikler bir ya da birden fazla alfanın durumunu ilerletmek için yapılması gerekenleri tanımlar. Örneğin, *gereksinimleri anlama* etkinlik uzayındaki etkinlikler geliştirilecek sistemin ne yapacağı konusunda ortak bir anlayış oluşturulmasını hedefler. Bu etkinlik uzayındaki *gereksinimler* alfasının durumunu *başlangıç* noktasından *bağdaşık* durumuna getirmeyi hedefler. Bu ilerleme bir ya da daha fazla etkinliğin yapılmasıyla elde edilebilir. Essence çekirdeğindeki üç ilgi alanındaki etkinlik uzayları Şekil 3'te gösterilmiştir.



Şekil 2. Çekirdekte bulunan kavramlar ve bu kavramlar arasındaki ilişkiler [3]



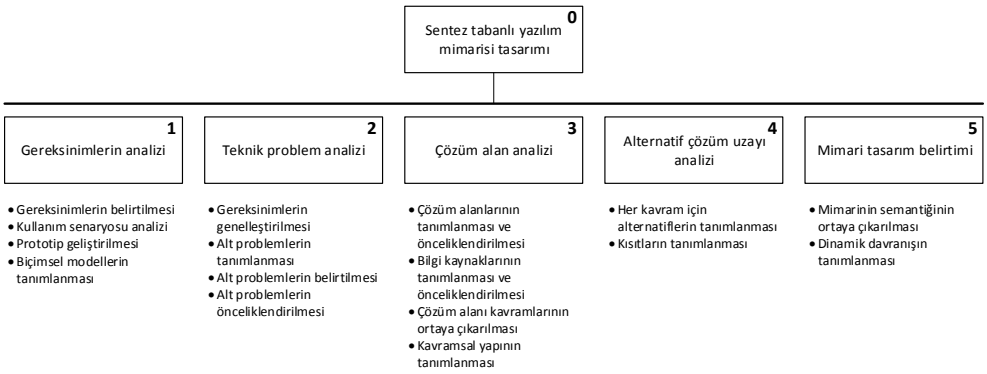
Şekil 3. Essence çekirdeğindeki etkinlik uzayları [3]

3. Sentez Tabanlı Yazılım Mimarisi Tasarımı

Sentez, geleneksel mühendislik alanlarında iyi bilinmektedir ve tasarım problemlerini çözmek için kullanılmaktadır [6]. Sentez sürecinde, bir ana problem birbirine gevşekçe bağlı alt problemlere bölünür; bu alt problemler çözülür ve bu alt çözümler ana çözümü oluşturulacak şekilde

bütünleştirilir [1]. Ana çözüm ve alt çözümler belirli amaçları belirli kısıtlar altında karşılamalıdır.

Sentez tabanlı yazılım mimarisi tasarımı, sentez sürecinin yazılım mimarisi tasarımına uygulanması sonucunda elde edilmiştir [1]. Tasarım yaklaşımı, yazılım mimarisi tasarımı için gereksinimlerin yanında alan bilgisinin de kullanılması gerektiği üzerine temellendirilmiştir. Tasarım yaklaşımının adımları Şekil 4'te gösterilmiştir.



Şekil 4. Sentez tabanlı yazılım mimarisi tasarım yaklaşımı [1][7]

Sentez tabanlı yazılım mimarisi tasarım yaklaşımı beş ana etkinlikten oluşmaktadır:

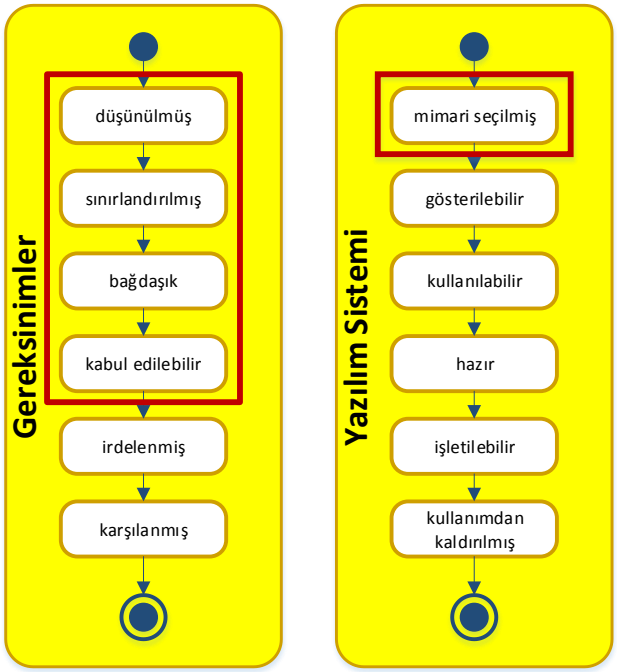
1. *Gereksinimlerin analizi*nde amaç paydaşların gereksinimlerini anlamaktır. Gereksinimler, kullanım durumları, kullanıcı hikayeleri gibi şablonlar kullanılarak belirtilebilir.
2. *Teknik problem analizi*nde amaç kullanıcı gereksinimlerini teknik problemlere dönüştürmektir. Gereksinimler uygun seviyede soyutlanarak yazılım mimarisi açısından önemli problemler tanımlanır. Bu problemler, gerekirse, alt problemlere bölünür ve önceliklendirilir.
3. *Çözüm alan analizi*nde amaç mimari tasarımda kullanılabilecek bir çözüm alan modeli oluşturmaktır. Her alt problem için ilgili alanlardaki kavramlar ortaya çıkarılır ve çözüm alan modelleri oluşturulur.
4. *Alternatif çözüm uzayı analizi*nin amacı mimari tasarım için kullanılabilecek farklı çözüm uzaylarını, bu uzaylardaki kavramları ortaya çıkarmaktır. Bu alternatifler arasında çeşitli ölçütlere ve kısıtlara göre seçimler yapılır.
5. *Mimari tasarım belirtimi*nin amacı, elde edilen mimari tasarımı yeterli bilgiyi içerecek şekilde belirtmektir. Bu belirtim, geliştirilecek yazılımın dinamik davranışını da kapsamaktadır.

4. Sentez Tabanlı Yazılım Mimarisi Tasarımı Yaklaşımının Essence Çerçevesiyle Modellenmesi

Mevcut pratiklerin ve yöntemlerin Essence çerçevesiyle modellenmesi için bir süreç [8]'de önerilmiştir. Bu sürece göre yaklaşımın anlatıldığı kaynaklardan [1] kavramlar elde edilmiş, bu kavramlar Essence çerçevesinin kavramlarına göre sınıflandırılmış, kavramlar Essence diliyle ifade edilmiş, kavramların özellikleri tanımlanmış, birbiriyle bağlantılı kavramlar ilişkilendirilmiş ve ortaya çıkan sonuç gözden geçirilmiştir. Bu adımlar takip edilerek yinelemeli ve

artırımlı olarak modelleme işlemi tamamlanmıştır.

Sentez tabanlı yazılım mimarisi tasarım yaklaşımının Essence çerçevesine göre kapsamı Şekil 5'te gösterilmiştir. Buna göre bu tasarım yaklaşımı Essence çerçevesinde bulunan yedi alfanın ikisinin durumlarının değişmesi için etkinlikler içermektedir:



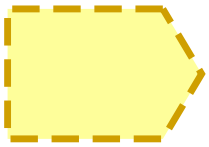
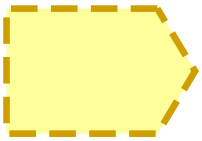
Şekil 5. Sentez tabanlı yazılım mimarisi tasarım yaklaşımının Essence çerçevesine göre kapsamı

1. *Gereksinimler:* Yeni bir yazılım sistemine gereksinim duyulduğu konusunda hemfikir olunduğunu gösteren *düşünülmüş* durumundan, gereksinimlerin paydaşlar tarafından kabul edilebilir bir sistemi tanımladığını gösteren *kabul edilebilir* durumuna geçiş
2. *Yazılım sistemi: Başlangıç* durumundan önemli teknik riskleri ve ilgili organizasyonel kısıtları irdeleyen bir mimarinin seçilmiş olduğu *mimari seçilmiş* durumuna geçiş

Şekil 5'te gösterilen kapsam, Essence çerçevesinin belirtimi [3] (alfaların durum açıklamaları ve bu durumlara gelindiğini kontrol etmek için kullanılması önerilen kontrol listeleri) ve sentez tabanlı yazılım mimarisi tasarımını anlatan metinler okunarak oluşturulmuştur. Bu eşleştirme işleminde yorum farklılıkları olması mümkündür.

Sentez tabanlı yazılım mimarisi tasarım yaklaşımındaki etkinliklerin Essence çerçevesindeki etkinlik uzayları ile ilişkisi Tablo 1'de gösterilmiştir. Aşağıdaki tabloda görüldüğü gibi mimari tasarım etkinliklerinin ait olduğu etkinlik uzaylarının tamamlanma ölçütlerinde belirtilen alfa durumları ile Şekil 5'teki kapsam tutarlıdır.

Tablo 1. Sentez tabanlı yazılım mimarisi tasarım etkinliklerinin Essence çerçevesinin etkinlik uzaylarıyla ilişkisi

Essence Çerçevesi		**Sentez Tabanlı Mimari Tasarım**
 Gereksinimleri anlama	*Başlangıç:* yok *Tamamlanma ölçütü:* Gereksinimler::bağdaşık	• Gereksinimlerin analizi • Teknik problem analizi • Çözüm alan analizi • Alternatif çözüm uzayı analizi
 Sistemi oluşturma	*Başlangıç:* Gereksinimler::bağdaşık *Tamamlanma ölçütü:* Gereksinimler::kabul edilebilir, Yazılım Sistemi::mimari seçilmiş	• Mimari tasarım belirtimi

Essence çerçevesindeki yedi alfa, bir yazılım geliştirme projesinin durumu hakkında üst seviyede bir genel bir bakış sunar. Projenin hakkında daha detaylı bilgi sahibi olabilmek için bu yedi alfanın durumlarında değişiklikler olmasını sağlayan ve daha detaylı durum bakışı sağlayabilecek alt alfalar tanımlanabilir. Sentez tabanlı yazılım mimarisi tasarımı yaparken sürecin ilerleme durumunu daha detaylı takip edebilmek amacıyla *gereksinim ögesi*, *problem ögesi*, *alan ögesi* ve *mimari öge* alt alfaları tanımlanmıştır.

Gereksinimler, *gereksinim ögeleri*nden oluşmaktadır. *Gereksinim ögeleri*, bir *paydaş* tarafından bir amacı gerçekleştirmek için istenen bir yetenek ya da koşul olarak tanımlanabilir. Her bir *gereksinim ögesi*nin durumu ayrı gözlemlenebilir. *Gereksinim ögeleri*nin tümünün durumu ise *gereksinimler* alfasının durumunu belirler.

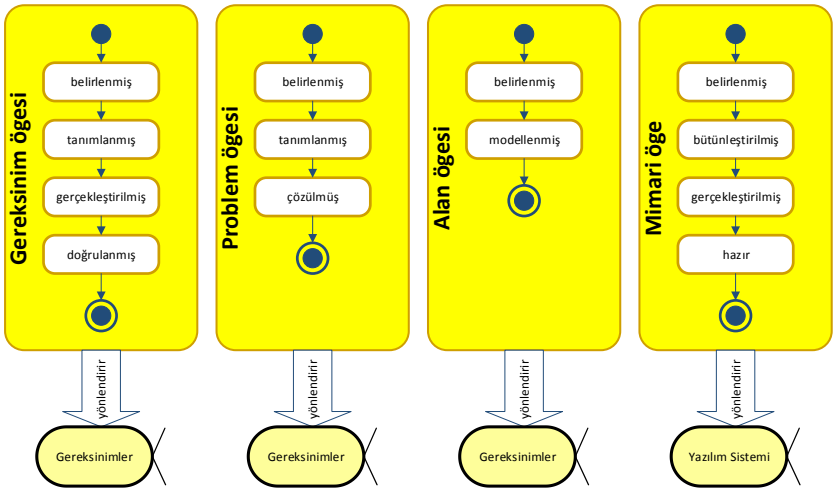
Problem ögesi, yazılım mimarisi tasarımında göz önünde bulundurulması gereken önemli bir yönü [9] temsil etmektedir. *Problem ögeleri*, temel olarak *gereksinim ögeleri*ndeki bilgilerden, yazılım mimarının deneyiminden ve *paydaş*lardan gelen girdilerden oluşturulmaktadır.

Alan ögesi, *problem ögeleri*nin yönlendirmesiyle ortaya çıkarılan çözüm alanındaki alan kavramlarını temsil etmektedir. *Alan ögeleri* arasındaki ilişkiler kurularak bir çözüm uzayı oluşturulur. Bir *problem ögesi* için çözüm oluşturabilecek birden fazla çözüm uzayı oluşturulabilir.

Mimari öge, bir *problem ögesi*nin çözümünü temsil etmektedir. Bu çözüm, *gereksinim ögeleri*, *problem ögeleri* ve *alan ögeleri*nden elde edilen bilgiler doğrultusunda oluşturulur. *Mimari ögeler* bir araya gelerek yazılım mimari tasarımını oluşturmaktadır.

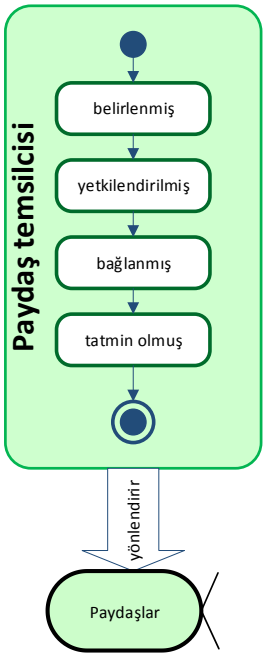
Mimari tasarım sürecinde yapılacakları belirlemek ve ilerlemeyi izlemek için kullanılabilecek alt alfalar ve bu alt

alfaların durumları Şekil 6'da gösterilmiştir. Alt alfalar, Essence çerçevesine göre yazılım geliştirme projesinin geneli için önemli olan yedi alfanın durum değişikliklerini yönlendirir. Şekil 6'da gösterildiği gibi, sentez tabanlı yazılım mimarisi tasarım etkinlikleri sadece *gereksinimler* ve *yazılım sistemi* alfalarının durumlarında değişikliğe yol açar.



Şekil 6. Sentez tabanlı yazılım mimarisi tasarımı ile ilgili önerilen alt alfalar

Sentez tabanlı yazılım mimarisi tasarım etkinliklerinin durumunu etkilemediği *paydaşlar* alfasının bir alt alfası olarak *paydaş temsilcisi* tanımlanmıştır. *Paydaş temsilcisi*nin durumları Şekil 7'de gösterilmiştir. Bu çalışma kapsamında, *paydaş temsilcisi* alt alfasının tanımlanmasının amacı, sentez tabanlı mimari tasarım etkinliklerinin Essence çerçevesi bakış açısıyla, yazılım geliştirme sürecinde nasıl bir yere oturduğunu daha geniş bir kapsamda göstermek ve mimari tasarım yaklaşımı etkinliklerinin başka etkinliklerle nasıl bütünleştirilebileceğini göstermektir.



Şekil 7. Paydaş temsilcisi alt alfasının durumları

Bir alfanın ya da alt alfanın herhangi bir durumda olup olmadığı kararı kontrol listeleri kullanılarak verilir. *Gereksinim ögesi*nin iki durumu için örnek bir kontrol listesi Tablo 2'de gösterilmiştir. Tablodaki kontrol listesine göre bir *gereksinim ögesi* kısaca tanımlanmış, kayıt altına alınmış, kaynağı ve değeri netleştirilmiş olduğu zaman *belirlenmiş* durumunda olmaktadır. Bu kontrol listeleri yazılım geliştirme takımı tarafından birçok koşul göz önüne alınarak oluşturulabilir. Önemli (ya da tüm) *gereksinim ögeleri*, *tanımlanmış* durumuna gelmesi *gereksinimler* alfasının durumunun *sınırlandırılmış* olmasını sağlamaktadır.

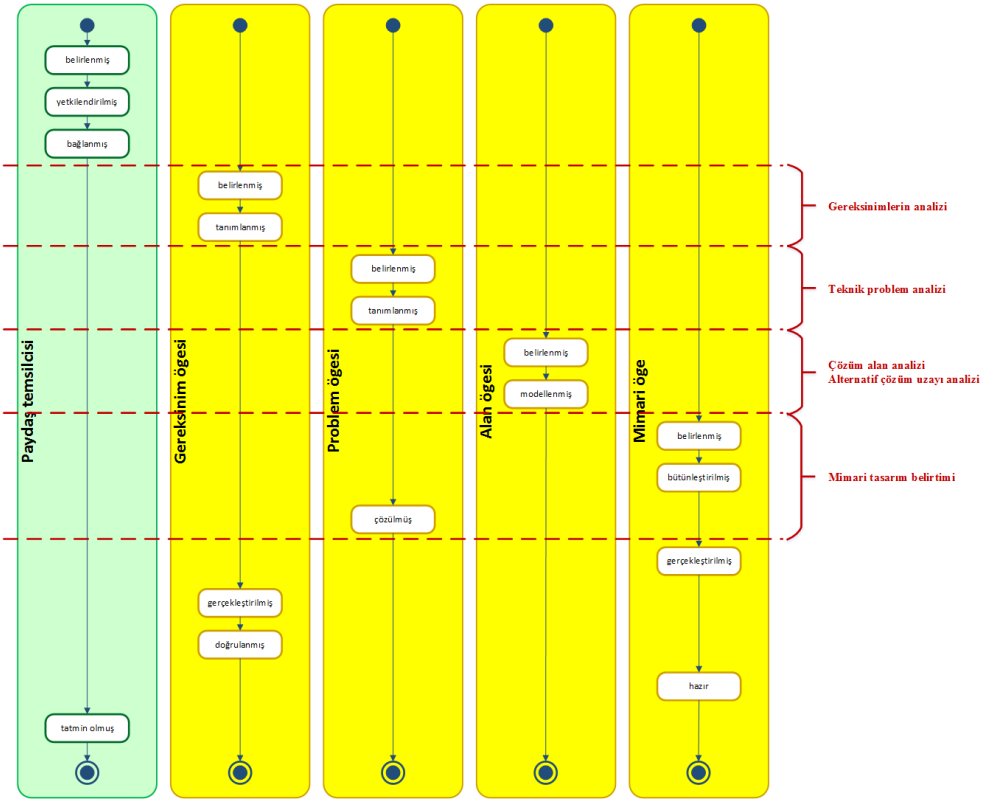
Tablo 2. Gereksinim ögesi alt alfasının durumlarının kontrol listesi (kısmi) [3]

Durum	Kontrol Listesi
Belirlenmiş	• Gereksinim ögesi kısaca tanımlanmış.
	• Gereksinim ögesi kayıt altına alınmış.
	• Gereksinim ögesinin kaynağı net.
	• Gereksinim ögesinin gerçekleştirilmesinin değeri net.
Tanımlanmış	• Gereksinim ögesinin gerekli ve yapılabilir olduğu gerekçelendirilmiş.
	• Gereksinim ögesinin belirtim tekniği net.
	• Gereksinim ögesi net, tutarlı, kısa ve öz tanımlanmış.
	• Gereksinim ögesi test edilebilir şekilde tanımlanmış.
	• Gereksinim ögesi önceliklendirilmiş.
	• Gereksinim ögesi gerçekleştirim için hazır.
	• Gereksinim ögesinin gerçekleştiriminin etkisi anlaşılmış.

Sentez-tabanlı yazılım mimarisi tasarımı etkinliklerinin önerilen alt alfaların durumları üzerinde hangi değişikliklere yol açtığı Şekil 8'de gösterilmiştir. Tasarım etkinliklerinin, yazılım geliştirme süreciyle nasıl bütünleştiğini göstermek amacıyla mimari tasarım etkinliklerinin etkilemediği bazı durum değişiklikleri de (örneğin *paydaş temsilcisi* alt alfasının durum değişiklikleri) örnekte gösterilmiştir.

Paydaş temsilcileri, *paydaşlar* alfasının durumunu yönlendiren alt alfalardır. Gereksinim analizi yapılmaya başlanmadan önce tüm *paydaşlar*ın gruplanması ve her grubu temsil edecek *paydaş temsilcileri*nin *belirlenmiş*, *yetkilendirilmiş* ve *bağlanmış* olması gerekmektedir. Paydaşlardan alınan bilgiler doğrultusunda *gereksinim ögeleri* belirlenir ve tanımlanır. *Gereksinim ögeleri*nden *problem ögeleri* belirlenir ve tanımlanır. Tanımlanan *problem ögeleri*nden yola çıkılarak *alan ögeleri* belirlenir ve modellenir. Bu *alan ögeleri* tüm çözüm uzaylarındaki kavramları temsil eder. Gereksinimler, alan bilgisi ve çözüm alternatifleri doğrultusunda *mimari ögeler* belirlenir. Her *mimari öge* bir alt problemin çözümü gibi görülebilir. Bu *mimari ögeler* birbirleriyle bütünleştirilerek mimari tasarım oluşturulur. Bir *mimari öge*, *bütünleştirilmiş* durumuna geldikten sonra ilgili *problem ögesi çözülmüş* olur. Bu noktada *gereksinimler* alfası *kabul edilmiş* ve *yazılım sistemi* alfası *mimari seçilmiş* durumuna gelir. Bu, projenin daha genel, üst seviyede bir durumunu yansıtır. Essence çerçevesine göre, sentez tabanlı mimari tasarım yaklaşımı etkinlikleri bu durum değişikliklerine yol açar. Yazılım sisteminin geliştirilmesinin tamamlanması için birçok başka etkinliğin de gerçekleştirilmesi gerekir. Başka yaklaşımlar, yöntemler, pratikler, etkinlikler gerçekleştirilerek sonrasında *mimari öge* gerçekleştirilir, ilgili *gereksinim ögesi* doğrulanır ve *mimari öge* kullanıma *hazır* hale gelerek *paydaş temsilcisi*nin *tatmin olmuş* durumuna

gelmesini sağlar. Bu adımlar bir seferde ve sıralı olarak gerçekleşmez. Durumlar arasında ileri ve geri gidişler olur. Ayrıca belirli bir anda birden fazla alt alfa farklı durumlara sahip olabilir. Örneğin bir *problem ögesi* çözülmüşken bir başkası *belirlenmiş* durumunda olur. Burada önemli olan projenin ilerlemesini istenen detayda görebilmek, sonraki adımları net olarak anlayabilmek ve takımın projenin durumu hakkında hemfikir olmasını sağlamaktır.



Şekil 8. Sentez tabanlı yazılım mimarisi tasarımı etkinliklerinin alt alfaların durumlarının değişimine etkileri

5. İlgili Çalışmalar

Object Management Group (OMG) tarafından yayımlanan Essence belirtimi [3], Scrum, kullanıcı hikayeleri, Unified Process, şelale yazılım geliştirme modeli gibi yaklaşımların Essence çerçevesi kullanılarak modellenmesine dair örnekler sunmaktadır. Scrum'ın Essence çerçevesiyle oluşturulmuş modeli [10] çalışmasında sunulmuştur. Bu çalışmada ayrıca Scrum, XP ve DevOps yöntemlerinin Essence çerçevesi kullanılarak nasıl bir araya getirilebileceği örneklenmiştir [10]. [8] çalışmasında bir yazılım geliştirme yönteminin Essence çerçevesiyle modellenmesi için bir süreç önerilmiştir. Bu çalışmada Nexus yöntemi bir örnek olarak Essence çerçevesiyle modellenmiştir.

Software and Systems Process Engineering Meta-Model (SPEM), yazılım geliştirme yöntemlerini ve süreçlerini modellemek için bir süreç mühendisliği üst modeli sunmaktadır

[11]. Essence dili ve SPEM, yöntemleri ve süreçleri modellemek için bir dil sunmaktadır. Essence çerçevesi bunun yanında yazılım mühendisliğinin genel bir alan modelini sunmaktadır. Bu ortak ve genel model yöntemlerin ve süreçlerin anlaşılması, karşılaştırılması ve birleştirilmesi için bir zemin hazırlamaktadır. Bunun yanında Essence çerçevesi bir projenin ilerlemesini alfalar ve alt alfalar kullanılarak izlenmesinin önemini vurgulamaktadır.

Mimari tasarım için çeşitli yaklaşımlar önerilmiştir. Bu yaklaşımlar ara ürün güdümlü, kullanım senaryosu güdümlü, alan güdümlü ve desen güdümlü olarak sınıflandırılabilir [1]. Ara ürün güdümlü mimari tasarım yaklaşımlarına örnek olarak OMT [12] ve nesneye yönelik analiz ve tasarım [13] verilebilir. Kullanım senaryosu güdümlü mimari tasarım yaklaşımında, sistem ve aktörler arasındaki etkileşimleri sıralı olarak belgeleyen kullanım senaryoları [14] mimari tasarımın temel girdileridir. Alan güdümlü mimari tasarımda, alan modellerinden alınan soyutlamalar mimari tasarımda kullanılır. Alan modelleme için çeşitli yöntemler kullanılabilir [15][16]. Desen güdümlü yaklaşımlarda mimari soyutlamalar desenlerden elde edilir [17].

6. Sonuçlar ve Gelecek Çalışmalar

Sentez tabanlı yazılım mimarisi tasarım yaklaşımı, teknik problem analizi, çözüm alan analizi ve alternatif çözüm uzayı analizi etkinliklerini belirgin şekilde ifade etmektedir. Bu etkinliklerin yazılım geliştirme sürecinin tamamı içinde nasıl bir yere sahip olduğunu anlamak için Essence çerçevesi kullanılmıştır. Modelleme sonucunda sentez tabanlı yazılım mimarisi tasarım yaklaşımının, soyut ve genel bir yazılım geliştirme modeli sunan Essence çerçevesine yerleştirilebildiği görülmüştür. Diğer taraftan Essence çerçevesinin de mimari tasarım etkinliklerini modellemek için uygun olabileceği sonucuna varılmıştır. Bunun yanında, modelleme işlemi yapanların yorum farklarından dolayı sonuç olarak elde edilen modellerde farklılıklar olabilmektedir.

Essence çekirdeğinin sunduğu alfalar ve alt alfa tanımlayabilme olanağı kullanılarak sentez tabanlı yazılım mimarisi tasarımı etkinliklerinin ilerlemesini takip edebilmek için bir düzenek önerilmiştir. Bu düzenek, Essence çerçevesindeki kavramlar ve Essence dilinin ve çekirdeğinin izin verdiği genişletme olanakları kullanılarak oluşturulmuştur. Her yazılım geliştirme projesi için o projenin koşullarına uygun alt alfalar, durumlar ve kontrol listeleri tanımlanarak bu düzenek özelleştirilebilir. Essence çerçevesi kullanılarak modellenmiş başka yöntemler, pratikler ve etkinlikler birleştirilerek projede kullanılabilir.

Özet olarak, Essence çerçevesi yazılım geliştirme etkinliklerinin anlaşılması, karşılaştırılması ve birleştirilmesi için bir temel oluşturmaktadır. Ayrıca projelerin takip edilmesi için durum tabanlı bir izleme düzeneği ortaya koymaktadır. Bu izleme düzeneği aynı zamanda proje takımının projenin bir anında bir sonraki adım için neler yapması gerektiğini de somut biçimde gösterebilmektedir. Böylece takımın tamamı projede nerede olduğu ve bir sonraki aşamaya nasıl gidebileceği konusunda ortak bir anlayışa sahip olabilmektedir.

Gelecek çalışmalar kapsamında sentez tabanlı yazılım mimarisi tasarımı yaklaşımının bir projede kullanılması planlanmaktadır. Bu projede ilerlemenin takip edilmesi için bu çalışmada önerilen düzenek kullanılacaktır. Bu düzeneğin kullanımını kolaylaştırmak için çeşitli

araçlar kullanılabilir. Bu araçlara örnek olarak alfaların ve alt alfaların durumlarının kontrol listelerinin takımla birlikte oluşturulması için oynanabilecek bir oyun, bu kontrol listelerinin takım içinde görünürlüğünün arttırılması için kartların kullanımı verilebilir.

Kaynakça

[1] Tekinerdogan B. 2000. Synthesis-Based Software Architecture Design. Twente Üniversitesi, Bilgisayar Bilimleri Bölümü, Doktora Tezi, 226s, Twente.

[2] Evans E. 2003. Domain-Driven Design: Tackling Complexity in the Heart of Software, 1st edition, Addison-Wesley Professional.

[3] Object Management Group. 2015. Essence - Kernel and Language for Software Engineering Methods, Version 1.1.

[4] Péraire C. 2013. A Step Forward in Software Engineering Education: Introducing the SEMAT Essence Framework, Keynote Address - LACREST 2013, Medellin.

[5] Elvesæter B, Benguria G, Ilieva. C. 2013. A comparison of the Essence 1.0 and SPEM 2.0 specifications for software engineering methods. The Third Workshop on Process-Based Approaches for Model-Driven Engineering (PMDE 2013).

[6] Maher ML. 1990. Process Models for Design Synthesis, *AI-Magazine*, s.49-58.

[7] Tekinerdogan B, Aksit M. 2006. Integrating the Concept of Synthesis in the Software Architecture Design Process, *Transactions of the SDPS,* Cilt 10(1), s.45-56.

[8] Giray G, Tüzün E, Tekinerdogan B, Macit Y. 2016. Systematic approach for mapping software development methods to the essence framework. The 5th International Workshop on Theory-Oriented Software Engineering (TOSE '16), 26-32.

[9] Tekinerdogan B, Aksit M. 1999. Deriving design aspects from conceptual models, Object-Oriented Technology, ECOOP '98 Workshop Reader, 410-414.

[10] Park JS, Mcmahon PE, Myburgh B. 2016. Scrum Powered by Essence, *ACM SIGSOFT Software Engineering Notes*, Cilt 41, No 1, s.1–8.

[11] Object Management Group. 2008. Software & Systems Process Engineering Meta-Model Specification, Version 2.

[12] Rumbaugh J, Blaha M, Premerlani W, Eddy F, Lorensen W. Object-Oriented Modeling and Design, Prentice-Hall, 1991.

[13] Booch G. 1991. Object-Oriented Analysis and Design, with Applications, Redwood City, CA: The Benjamin/Cummins Publishing Company.

[14] Jacobson I, Booch G, Rumbaugh J. 1999. The Unified Software Development Process, Addison-Wesley.

[15] Arrango G. 1994. Domain Analysis Methods. Schaeffer W, Prieto-Diaz R, Matsumoto M. ed. 1994. Software Engineering Reusability, Ellis Horwood, New York.

[16] Wartik S, Prieto-Díaz R. 1992. Criteria for Comparing Domain Analysis Approaches, *International Journal of Software Engineering and Knowledge Engineering*, Cilt 2, No. 3, s. 403-431.

[17] Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M. 1996. Pattern-Oriented Software Architecture: A System of Patterns, John Wiley & Sons.