Gömülü Sistemler için Yazılım Mimari Çerçevesi

Bedir Tekinerdogan[1], Şafak Şeker[2], Gökhan Kahraman[3], Metin Tekkalmaz[4], Özgü Özköse Erdoğan[5]

[1]Bilkent University, Department of Computer Engineering
Bilkent 06800 Ankara, Turkey
[2,3,4,5] Aselsan,
PK. 1, 06172 Yenimahalle, Ankara, Türkiye

[1]bedir@cs.bilkent.edu.tr
[2,3,4,5] {sseker|gokhank|tkalmaz|ozkose}@aselsan.com.tr

Özet. Tasarım kararları verilirken farklı paydaşlar için mimari görünümlerin oluşturulması yazılım mimari tasarımında kullanılan yaygın pratiklerden biridir. Bir mimari çerçeve, önerilen mimari bakış açılarını organize eder ve yapısını ortaya koyar. Literatürde farklı mimari çerçeveler sunulmuştur. Ancak bunlar öncelikli olarak geleneksel masaüstü tabanlı alanlara ve bazıları da dağıtık geliştirme platformlarına odaklanmıştır. Bu bildiride Aselsan'da yürütülen çoklu ürün hattı mühendisliği projesi kapsamında geliştirilen gömülü sistemlerden hareketle bir yazılım mimari çerçevesi tanımlanmaktadır. Gömülü sistemlerin ve ilgili metamodellerin literatürde incelenmesinin ardından Aselsan'daki gömülü yazılımlar için bir metamodel oluşturulmuş ve bu metamodel temel alınarak çerçeve tanımlanmıştır. Çerçeve, her biri metamodel'in belirli bir perspektifini temel alacak ve farklı paydaşların ihtiyaçlarını adresleyecek şekilde dokuz bakış açısını içermektedir. Bu bildiride bakış açılarının seçimi ve Aselsan'daki kullanımı ile ilgili yaklaşım da sunulmaktadır.

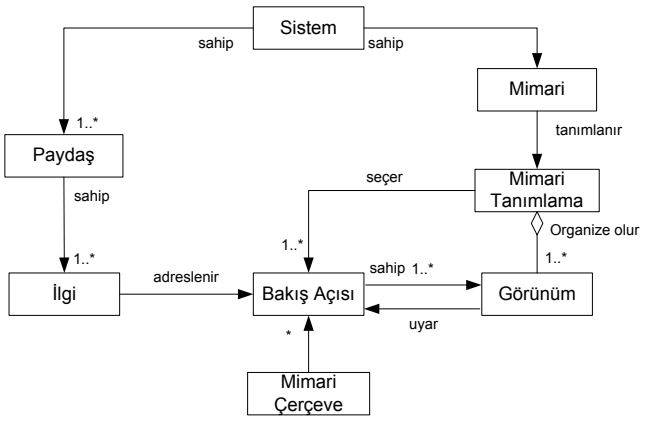
Anahtar Kelimeler. Yazılım Mimari Modelleme**,** Mimari Görünümler, Mimari Kalite Gerekleri

1 Giriş

Mimari girdiler paydaşların ihtiyaçlarını tanımlar ve mimariyi şekillendirir. Bir paydaş bir sistem üzerinde ilgisi (concern) olan kişi, ekip ya da organizasyon olarak tanımlanabilir. Her paydaş'ın ilgisi mimarın verdiği kararlara etki eder [11]. Mimariyi tanımlarken paydaşların ilgilerine göre sistemin farklı mimari görünümler ile modellenmesi ve dokümante edilmesi yaygın bir uygulamadır [3][6][10]. Bir mimari görünüm,belirli bir ilgiyi desteklemek için sistemi oluşturan elemanların bir alt setinin

ve aralarındaki ilişkilerin gösterimidir. Birden fazla görünümün kullanılması,ilgilerin ayrıştırılmasını sağlar ve dolayısıyla yazılım mimarisinin farklı paydaşlar için modellenmesi, anlaşılması, iletişimi ve analizi için destek olur.

Şekil 1'de, IEEE tarafından önerilen mimari tanımlama standardı temel alınarak mimari görünümler için kavramsal model gösterilmiştir. Şekil 1'de gösterildiği gibi bir sistemin kendisiyle ilgisi olan bir ya da daha fazla paydaşı vardır. İlgiler bakış açılarıyla adreslenir ve görünümler bakış açılarından elde edilir. Her sistemin bir mimari tanımlama ile elde edilen belli bir mimarisi vardır. Mimari tanımlama bakış açılarıyla uyumlu bir grup görünümü içerir.



Şekil 1. Mimari Görünümler için Kavramsal Model (IEEE Standard)

Bir mimari çerçeve önerilen mimari bakış açılarını organize eder ve yapılandırır. ISO standardının tanımı şu şekildedir [7]:

"Mimarileri tanımlamak için belirli bir alanda ya da paydaş topluluğu arasında oluşturulan kurallar, prensipler ve pratikler"

Mimari çerçeveler, ilk zamanlarda, sınırlı sayıda ve belli bakış açılarının kümesi olarak önerilmiştir. Farklı sistemler için farklı ilgilerin adreslenmesi ihtiyacı nedeniyle, bakış açılarının sabitlenmemesi gerektiğinin ve bunun yerine çoklu bakış açılarının kullanılabileceğinin farkına varılarak bu yöndeki eğilim artmıştır. Örnek olarak güncel bir mimari çerçeve olan "Görünümler ve Ötesi" (Views and Beyond) yaklaşımı [3] mevcut bakış açılarını uyumlandırma ve yeni bakış açılarını ekleme konusunda mekanizmalar sağlamaktadır.

Mimari çerçeve, fikirlerin ve bakış açılarının modellenmesinde ve mimarilerin dokümante edilmesinde önemli rol oynamaktadır. Ancak, mimari modelleme konusundaki mevcut yaklaşımlar geleneksel masaüstü ve bazen dağıtık geliştirme platformlarına odaklanıyor gibi görünmektedir. Gömülü sistemler sıklıkla veya açık olarak ele alınmamaktadır.

Bu bildiride Aselsan'da Çoklu Ürün Hattı Mühendisliği projesi kapsamında geliştirilen Gömülü Sistemler için Yazılım Mimari Çerçevesi tanımlanmaktadır.

Gömülü sistemler ve ilgili metamodeller ile ilgili yapılan literatür araştırmasının ardından oluşturulan metamodel kullanılarak çerçeve tanımlanmıştır. Çerçeve, her biri metamodelin belirli bir perspektifi temel alınarak ve farklı paydaş ilgilerini karşılayan dokuz bakış açısından oluşturulmuştur. Bu çalışmada bu bakış açılarının seçimi ve Aselsan'daki kullanımı açıklanmaktadır.

Bildirinin kalanı şu şekilde organize edilmiştir: Bölüm 2'de çalışma için teorik altyapı ve motivasyon sunulmaktadır. Mimari bakış açılarının modellenmesi için kavramların tanımlandığı metamodel Bölüm 3'te anlatılmaktadır. Bölüm 4 gömülü sistemler için tanımlanan bakış açılarını sunmaktadır. Bölüm 5'te bakış açılarının kullanımı incelenmektedir. Son olarak Bölüm 6 çalışma sonuçlarını paylaşmaktadır.

2 Teorik Altyapı ve Motivasyon

Bu çalışmada gömülü sistemlerin gereksinimlerini ifade etmek amacıyla bazı bakış açıları tanımlanmaktadır. Bunun yanında gömülü sistemlerin diğer yazılım ağırlıklı sistemler ile ortak özellikleri de bulunmaktadır, bu tip genel ilgilerin ifade edilmesi için ise güncel bir yaklaşım olan "Görünümler ve Ötesi" yaklaşımından [3] faydalanılmaktadır. "Görünümler ve Ötesi" yaklaşımında bakış açılarının tanımlanması için görünüm tipi (view category) ve stil (style) kavramları kullanılmaktadır. Bu yaklaşıma göre üç tip görünüm tipi belirlenebilir: Sistemin temel gerçekleme birimlerinin tanımlanması için Modül Görünümü (modul), sistemin çalışma birimlerinin tanımlanması için Birim ve Bağlantılar Görünümü (components and connectors) ve yazılım ile yazılımın geliştirme ve çalışma ortamları ile ilişkisinin tanımlanması için Konuşlandırma Görünümü (Deployment). "Görünümler ve Ötesi" yaklaşımı halihazırda belli mimari stiller tanımlamakla birlikte farklı ihtiyaçlar için yeni stiller de tanımlanabilir.

Son dönemde mimari görünümlerin tanımlanması için genel yaklaşımların yanı sıra gömülü sistemlere özel yaklaşımlar da göze çarpmaktadır. Arias et al. [1] hedeflerine göre yaklaşımları iki gruba ayırmışlardır.

İlk grupta aşağıdakiler yer almaktadır:

- Sistemde eş zamanlı çalışan parçaları açık bir şekilde belirtmek amacıyla, işlevsel elemanları eş zamanlılık birimleri ile ilişkilendirerek sistemin eş zamanlılık mimarisini tanımlayan *Eş Zamanlılık Bakış Açısı* [16]
- Sistem elemanları arasında davranışsal konuları dilden bağımsız bir şekilde tanımlayan *Davranışsal Tanım* [3]

İkinci grupta aşağıdakiler yer almaktadır:

- Sistemin konuşlandırılacağı ortamın tanımlanması amacıyla ortaya konulan ve sistemin çalışma zamanı ortamına bağımlılıklarını da içeren Konuşlandırma Bakış Açısı [16]
- Çalışma platformunun parça ve bağlantılara tahsisinin içeren *Konuşlandırma Stili* [3] ve işlevlerin fiziksel kaynaklarla ve çalışma zamanı özellikleri ile ilişkilendiren *Çalışma Mimarisi*ne [6] ek olarak [1]'de verilen bakış açıları (bkz. Tablo 1.).

Tablo 1. [1]'den uyarlanan mevcut görünümler

Bakış açısı	İlgi alanı	Sistem Elemanları
Eş Zamanlılık [16]	- Görev yapıları ve işlevlerin görevlere dağılımı - İşlem (process) arası iletişim ve durum yönetimi - Senkronizasyon ve bütünlük - Açılış, kapanış, görev hataları	İşlem, İşlem grupları, izlekler(thread), işlemler arası haberleşme
Davranış Tanımı [3]	- Haberleşme tipleri - Çalışma sırası ile ilgili kısıtlar - Zamanlayıcı ile tetiklenen işler	Kullanım durumları, yapısal elemanlar, işlemler, durumlar, uygulamalar ve nesneler
Konuşlandırma [16]	- Donanım (özellikler ve miktar) - Hazır kullanılan yazılımlar ile ilgili ihtiyaçlar ve teknoloji uyumlulukları - Ağ gereksinimleri, fiziksel ve kapasite kısıtları	İşlemci ve istemci düğümleri, ağ bağlantıları, donanım birimleri ve işlemler
Konuşlandırma Stili [3]	- Yazılım elemanları ile donanım arasında tahsis, göç ve kopyalama ilişkileri - Donanım özellikleri, (Örn. bant genişliği, kaynak kullanımı)	Yazılım elemanları (işlemler) ve donanım (işlemci, bellek, disk, vb.)
Çalışma Mimarisi [6]	- Çalışma konfigürasyonu ve donanımsal birimlerle ilişki - Konfigürasyonun dinamik davranışı - Haberleşme protokolü - Çalışma zamanı varlıklarının tanımı	Işlemler, görevler, izlekler, istemciler, sunucular, tampon alanları, mesaj kuyrukları, ve sınıflar
Çalışma Bakış Açıları [1]	- Sistem çalışması	Parçalar, işlemler, izlekler, işlemler arası haberleşme

Genel olarak aşağıda verilen ihtiyaçlar gömülü sistemlere yönelik bakış açılarının oluşmasında etkili olmuştur, bu ihtiyaçlar farklı paydaşların gömülü sistemler için ilgilerini belirtmektedir.

- Sistemin anlaşılması: Gömülü sistemin daha iyi anlaşılmasına yönelik bakış açılarıdır. Çerçeve, farklı paydaşların farklı gereksinimlerine yönelik farklı bakış açılarını içererek sistemin farklı özelliklerinin anlaşılmasını sağlar.
- İletişim: Gömülü sistem bakış açıları sistem tasarımı ve gerçeklemesi ile ilgili teknik bilginin aktarımı açısından faydalıdırlar. Bu sayede tasarımcının geliştirici, yönetici gibi farklı paydaşlar ile iletişimini sağlar. Gömülü sistemlere yönelik

görünümler yazılımın platform ile nasıl bir etkileşim içerisinde olduğunu göstermeleri açısından da gereklidirler.

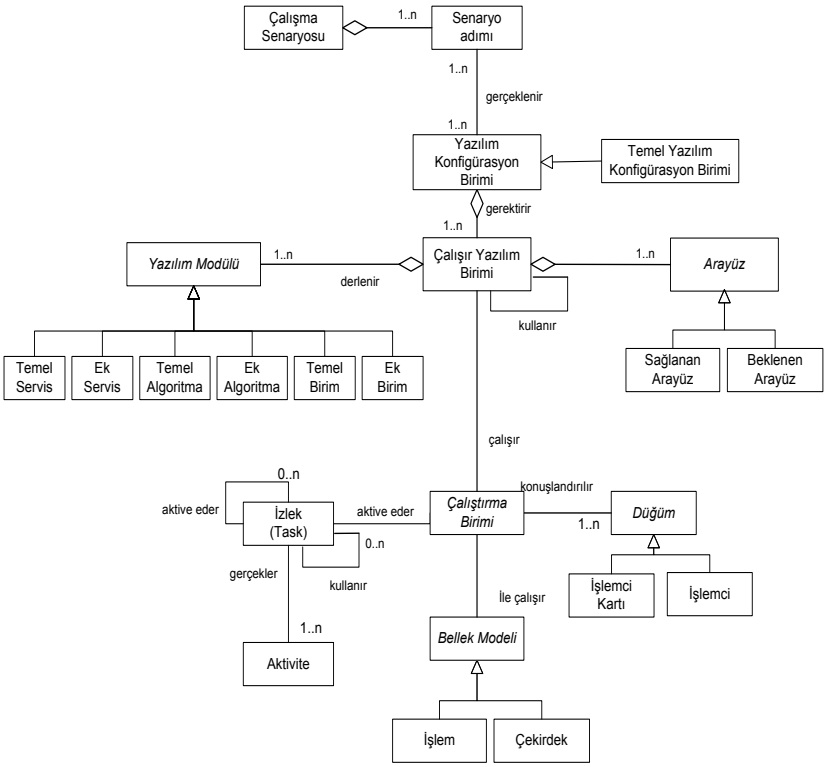
- Proje planlama: Görünümler farklı tasarım alternatiflerini tartışmak ve projeyi buna göre planlamak amacıyla da kullanılabilir.
- Gereksinim ile tasarım ve gerçekleme arası uyumluluk: İdealde mimari tanımlama güvenilirlik, güvenlik, performans gibi işlevsel olmayan gereksinimlerin nasıl karşılandığını da tanımlıyor olmalıdır. Gerçekleme ile hedeflenen gereksinimler arasında farklılıklar oluşması sıklıkla karşılaşılan bir durumdur. Mimari tanım bu farklılaşmaların belirlenmesi ve gerçeklemeyi uygun şekilde yönlendirmesi amacıyla kullanılabilir.

3 Gömülü Sistemler için Metamodel

Arias [1], koşar yazılım için bir metamodel tanımlamıştır. Bu metamodel, fonksiyonel bileşenleri koşar zaman elemanlarıyla ilişkilendirmekte ve çalışma zamanındaki aktiviteleri tanımlamaktadır. Arias, çalışma bileşenleri (executing components) arasındaki bağımlılıkları analiz eden bir metamodel kullanmıştır. Bu metamodelin performans yükü ile ilişkili paralelleştirme analizine odaklanan daha rafine edilmiş bir versiyonu Muhammad ve ark. [13] tarafından sunulmuştur. Şu anki literatürü temel alarak ve gömülü sistem tasarımı ihtiyaçlarına da odaklanarak gömülü sistemler için Şekil 2'de gösterilen bir metamodel önermekteyiz. Metamodelin kavramları aşağıda tariflenmiştir:

- Sistemin çalışması, bir kullanıcı etkileşimi veya çalışma senaryosunun koşması ile başlar.
- Çalışma senaryosu geliştirici ve son kullanıcı bakış açıları ile hazırlanmış görünümlerle [10] detaylı olarak tanımlanmış normal ve sıra dışı kullanımları tanımlar. Senaryolar kullanım durumu çizimleriyle tanımlanabilir. Bir senaryo belli bir işlevin nasıl yerine getirildiğini adım adım açıklar.
- Yazılım Konfigürasyon Birimi (YKB) (SCI-Software Configuration Item) terimi yazılım konfigürasyon yönetiminin yapısal bir parçasını ifade eder. Her bir YKB sistem seviyesinde tanımlı çalışma senaryolarının bir kısmını gerçekler. YKB birden fazla çalışır bileşen ve her bir çalışır bileşen birden fazla modülden oluşabilir.
- Bir yazılım bileşeni (çalışır bileşen), tariflenmiş arayüzler ve dış bağımlılıklar ile oluşmuş bir birimdir. Bileşenleri haberleştirebilmek için, bileşenlerin dış dünyadan beklediği ve dış dünyaya sunduğu arayüzler tanımlanmıştır.
- Gömülü sistemlerde yazılım bileşenleri çekirdek (kernel) veya işlem (process) durumlarında çalıştırılabilir. Çekirdek durumda çalışan bileşenler işletim sistemi ile aynı hafıza bölümünü paylaşırlar. Bununla birlikte, işlem durumunda çalışan bileşenler kendileri için ayrılmış bir korumalı hafıza bölümünde çalışırlar; kod, veri ve işletim sistemi kaynaklarını bu hafıza bölümünde tutarlar.
- Bir düğüm (processing node), üzerinde yazılım bileşenlerinin yüklendiği ve çalıştırıldığı bilgisayar veya donanımı tarifler. Düğüm terimi tek çekirdekli bir işlemci olabileceği gibi, çok çekirdekli bir işlemcinin her bir çekirdeğini de ifade edebilir.

- Yazılım Konfigürasyon Birimleri içerdikleri bileşen sayısına göre bir veya daha fazla düğüm üzerinde konumlandırılmaktadır.
- Bir izlek (thread) bir işletim sistemi tarafından çalışması planlanabilen en küçük yazılım birimidir. Bir izlek küçük boyutlu bir işlemdir.
- Bir işlem, ana izleği koşturarak çalışmaya başlar; fakat daha sonra yeni izlekler oluşturabilir. Bir izlek, bir işlem kapsamında ardışık olarak tanımlı bazı yazlım kodlarını çalıştırır ve aynı kapsamda bazı işletim sistemi kaynaklarını kullanır.
- Aktiviteler izlekler tarafından yerine getirilmesi gereken görevleri tanımlar. Aktiviteler veri ulaşım, kod kullanım, platform kullanım gibi farklılaştırılabilir.



Şekil 2. Gömülü sistemlerin bakış açılarının türetildiği metamodel

4 Gömülü Sistemler için Bakış Açıları

Önceki bölümde tariflenen metamodel temel alınarak oluşturulan gömülü sistem bakış açıları Tablo 2 de gösterilmektedir. Mimari çerçeve dokuz adet ilişkili bakış açısından oluşur. Bu bakış açıları hem Aselsan hem de gömülü sistemlerin genel ihtiyaçlarını

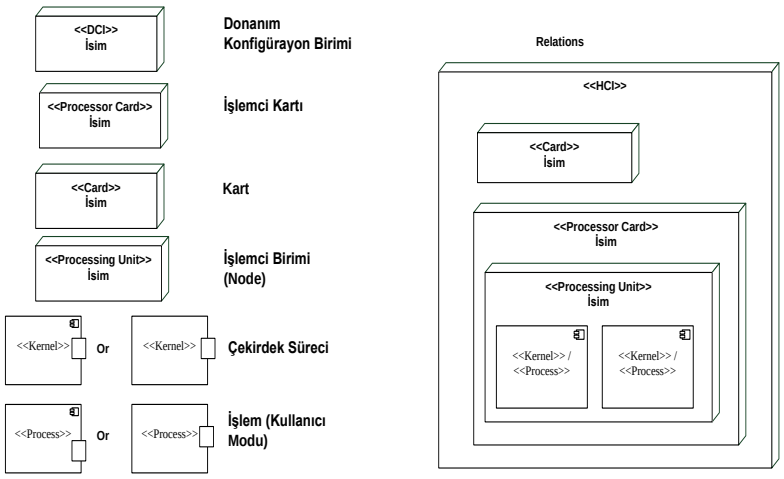
karşılayacak şekilde belirlenmiştir. Bakış açışları standart bir şablon yapısı ile tariflenmektedir. Yer sınırlamasından dolayı bütün bakış açıları yerine örnek bir bakış açısı detaylı olarak verilmiştir. "Bileşen Düğüm Tahsisi" (Component to Node Allocation) bakış açısı, bileşenler ile işleme birimleri arasındaki eşleşmeyi tanımlamaktadır. Bu bakış açısı Tablo 3'te gösterilmektedir. Bu bakış açısı kapsamında, gömülü yazılımlarda kullanılan bileşen tipleri de tariflenmektedir.

Tablo 2. Gömülü Sistemler Bakış Açıları Kümesi.

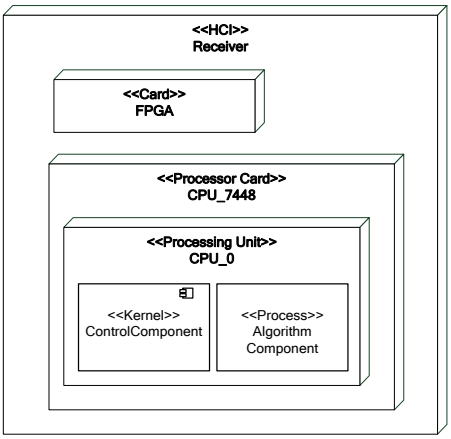
Bakış Açısı	**Tanımı**
Çalışma Senaryosu – YKB Tahsis (Execution Scenario to SCI Allocation)	Çalışma senaryolarını ve senaryoları oluşturan ana adımları ve bu adımların hangi YKB'de gerçeklendiğini tanımlar
YKB Tahsis (SCI Allocation)	Çalışır bileşenlerin YKB'lere ve modüllerin çalışır bileşenlere tahsisini tanımlar
Senaryo Adımı – Modül Tahsis (Scenario Step to component Allocation)	Senaryo adımlarının sıralanışını ve senaryo adımlarının modüllere tahsisini tanımlar
Bileşen – Düğüm Tahsisi (Component to Node Allocation)	Çalışır bileşenlerin düğümlere (node) tahsisini tanımlar
Bileşen Etkileşim (Component Interaction)	Çalışır bileşenlerin çalışma esnasındaki birbirleriyle olan ilişkisini tanımlar
Bileşen - İzlek Ayrıştırma (Component-Thread Decomposition)	İzleklerin ayrışımını çalışır bileşen tanımlar
Modül – İzlek Tahsis (Module to Thread Allocation)	Modüllerin izleklere tahsisini tanımlar
İzlek Etkileşim (Thread Interaction)	İzlekler arasındaki ilişkiyi tanımlar
Davranış (Behavior)	Tanımlanan bakış açılarındaki elemanların arasındaki davranış ve etkileşim tanımlanır

Tablo 3. "Bileşen – *Düğüm Tahsisi*" Bakış Açısı

Bölüm	**Tanım**
«Bakış Açısı İsmi»	Bileşen – Düğüm Tahsisi
«Genel Açıklama»	Bileşenlerin düğümlere dağılımı
«İlgiler»	İhtiyaç duyulan düğümler nelerdir? Düğümlerin ayrışımı nasıldır? Bileşenler düğümlere nasıl dağıtılır?
«Tipik Paydaşlar»	Sistem Mühendisleri
«Mimari Eleman »	Düğüm, Işlem, Çekirdek
«İlişkiler»	Dağılım
«Model tipleri ve gösterim»	



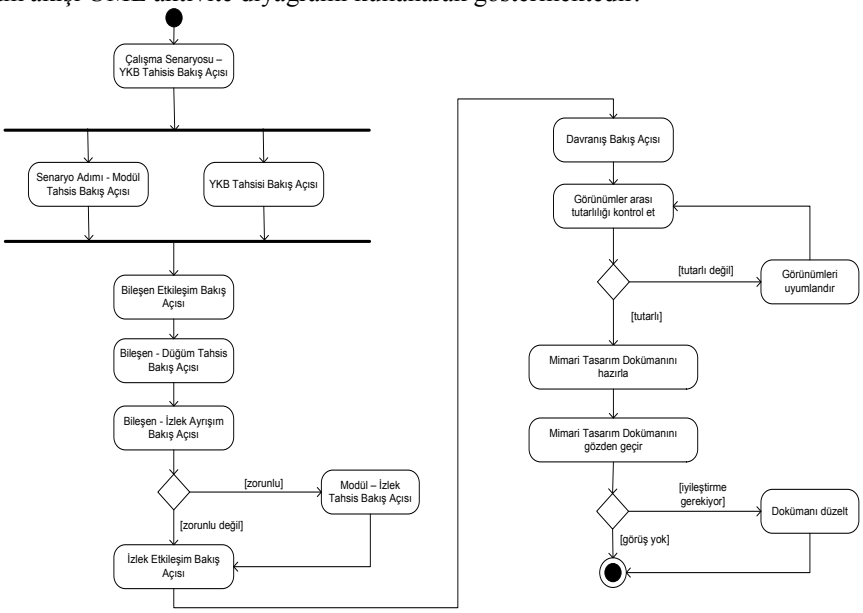
"Bileşen Düğüm Tahsisi" bakış açısı ile üretilmiş bir görünüm Şekil 3'de gösterilmektedir.



Şekil 3. "*Bileşen – Düğüm Tahsisi*" bakış açısı için örnek görünüm

5 Bakış Açılarının Kullanımı

Önceki bölümlerde yapısal bakış açıları ve dinamik davranış modelleme tanımlanmıştı. Bu bölümde mimariyi dokümante ederken bakış açılarının kullanımı için bazı önemli noktalar anlatılmaktadır. Şekil 4 bakış açılarının seçimi için gerekli tüm akışı UML aktivite diyagramı kullanarak göstermektedir.



Şekil 4. Gömülü sistemler bakış açılarının kullanımını gösteren aktivite diyagramı

1. Öncelikle kavramsal ve üst seviye modellerin tanımlanması, ardından daha alt seviye koşar-zaman modellerle devam edilmesi fikri süreçte temel alınmıştır. Bu amaçla ilk olarak "*Çalışma Senaryosu – YKB Tahsis*" bakış açısı kullanılarak her bir YKB için sistem seviyesinde tanımlı senaryoların hangilerinin gerçekleneceği ve bu senaryoların ana adımları belirlenir. Bu bakış açısı sistemin davranışsal modeli ile gerçekleme modeli arasında bağlantıyı sağlar.
2. Bundan sonra YKB'yi oluşturan çalışır bileşenler belirlenerek "*YKB Tahsis*" bakış açısı ile gösterilir. Gömülü sistemlerde çalışır bileşenler bir veya birden fazla izlek içerebilir, YKB Tahsis bakış açısında izlek detayına girilmeden çalışır bileşenler belirlenir.
3. Bu aşamada daha detay analiz isteniyor ise senaryo adımlarının çalışır bileşenlere dağılımı "*Senaryo Adımı – Bileşen Tahsis*" bakış açısı kullanılarak tanımlanabilir. Böylece her bir modül için atanan senaryo adımları belirlenerek gerçekleme ve sistem entegrasyon testleri için daha detay bilgi sağlanmış olur.
4. Daha sonra çalışır bileşenlerin birbirleri ile ve dış dünya ile olan arayüzleri "*Bileşen Etkileşim*" bakış açısı ile tanımlanabilir. Bu görünüm izlekler arası haberleşme arayüzlerinin görülmesini ve YKB dış arayüz gereklerinin hangi izlek tarafından sağlandığının görülmesini sağlar.
5. Bu adımlardan sonra "*Bileşen – Düğüm Tahsisi*" bakış açısı kullanılarak çalışır bileşenlerin hangi düğümlerde çalışacağı modellenir. Çalışır bileşenler ortak olarak tek bir düğüm üzerinde çalışabileceği gibi aynı kart üzerinde farklı düğümlerde veya farklı kartlarda çalıştırılabilirler; aynı çalışır bileşen birden fazla düğüm üzerinde de çalışabilir. "*Bileşen – Düğüm Tahsisi"* bakış açısı düğümlerdeki işlem gücü ihtiyacı yanında çalışır bileşenler arası iletişim altyapılarının tanımlanmasına da yardım eder.
6. Akışın son adımı izleklerin detayını göstermektir.
7. İlk olarak "Bileşen - *İzlek Ayrıştırma"* bakış açısı kullanılarak her bir çalışır bileşen içinde yer alan izlekler modellenir.
8. Daha sonra eğer gerekliyse modüllerin izleklere dağılımı gösterilebilir; bu bakış açısı hangi modüllerin izlek içerisinde yer alacağını gösterir, bir modül birden fazla izlek içinde yer alarak o izlek içeriğinde çalışabilir.
9. "*İzlek Etkileşim*" bakış açısı ile izlekler arasındaki arayüzler tanımlanır. Arayüzlerin tanımları ortam, protokol, veri tipi bilgileri içerebilir.

Tüm yapısal görünümler modellendikten sonra davranışsal modeller tanımlanabilir. Bunun ardından bütün görünümler arasındaki tutarlılık kontrol edilir. Bunun için bakış açılarındaki elemanlar ve söz dizimi incelenir. Eğer herhangi bir tutarsızlık tespit edilirse buna karşılık gelen adaptasyonlar tanımlanmalıdır. Bu işlem bir kaç iterasyon gerektirebilir.

Sürecin takip eden adımı daha önceki adımlarda tanımlanan aktivitelerle mimariyi dokümante etmektir. Görünümleri tanımlamanın yanı sıra ihtiyaç duyulan diğer bilgiler de eklenebilir. Mimari dokümanlar tamamlandıktan sonra diğer önemli paydaşlar tarafından gözden geçirilmelidir. Eğer iyileştirmeler gerekiyorsa doküman güncel-

lenebilir. Güncellenme ihtiyacı yoksa süreç tamamlanır ve sistemin geliştirilmesinde kullanılmak için doküman hazır duruma gelir.

6 Sonuçlar

Yazılım mimarisini modellerken farklı ilgileri ifade edebilmek için çoklu görünümlerin kullanılmasına ihtiyaç duyulmaktadır. Mevcut mimari görünüm yaklaşımları gömülü sistemler alanında uygulanmak için yeterli olmamaktadır. Bu bildiride Aselsan'da gömülü sistemleri modellemek için rehber olarak kullanılan bir mimari çerçeve anlatılmıştır. Görünümlerin sağlam bir altyapıdan türetilebilmesi için ilk olarak gömülü sistemlerde kullanılan önemli kavramları gösterecek şekilde bir meta-model tanımlanmıştır. Farklı perspektifler temel alınarak ve ISO/IEC 42010 standardının mimariyi tariflemek için gösterdiği yol takip edilerek dokuz farklı bakış açısı ortaya çıkarılmış ve tanımlanmıştır. Buna ek olarak bakış açılarının kullanımı da anlatılmıştır. Bu bildiride bahsedilen bakış açılarının tümü çeşitli gömülü sistemlerin alan mimarisini modellemek için Aselsan'da halen kullanılmaya devam edilmektedir.

7 Teşekkür

Bu süreçte birlikte çalıştığımız iş arkadaşlarımız Selma Dökmen, Rabia Esra Giray ve Erhan Örümlü'ye katkılarından dolayı teşekkür ederiz.

Kaynaklar

1. Arias, T. B., Avgeriou, P., and America, P.: Analyzing the Actual Execution of a Large Software-Intensive System for Determining Dependencies. In Proceedings of the 2008 15th Working Conference on Reverse Engineering, pp. 49-58. WCRE. IEEE Computer Society, Washington, DC(2008)
2. P. Clements, L. Northrop. Software Product Lines: Practices and Patterns. Boston, MA:Addison-Wesley, 2002.
3. P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, J. Stafford. Documenting Software Architectures: Views and Beyond. Second Edition. Addison-Wesley, 2010.
4. E. M. Dashofy , A. van der Hoek , R.N. Taylor. A comprehensive approach for the development of modular software architecture description languages, ACM Transactions on Software Engineering and Methodology (TOSEM), v.14 n.2, p.199-245, April 2005
5. Eclipse Modeling Framework Web Site, http://www.eclipse.org/emf/, accessed on April 2011.
6. C. Hofmeister, R. Nord, and D. Soni. Applied Software Architecture. Addison-Wesley, NJ, USA.
7. ISO/IEC 10746-2:1996] International Organization for Standardization & International Electrotechnical Commission. Information Technology - Open Distributed Processing - Reference Model: Foundations (ISO/IEC 10746-2). 1996.

8. ISO/IEC 42010:2007] Recommended practice for architectural description of software-intensive systems (ISO/IEC 42010), 2011.
9. Kolb R, John I, Knodel J, Muthig D, Haury U, Meier G. Experiences with product line development of embedded systems at Testo AG. Proceedings of the 10th International Software Product Line Conference, Baltimore, U.S.A.,
10. P. Kruchten. The 4+1 View Model of Architecture. IEEE Software, 12(6):42–50, 1995.
11. A.J. Lattanze. Architecting Software Intensive Systems: A Practitioner's Guide, Auerbach Publications, 2009.
12. N. Medvidovic and R. N. Taylor. A classification and comparison framework for software architecture description languages, IEEE Trans. Software Eng., vol. 26, no. 1, pp. 70–93, 2000.
13. N. Muhammad, N. Boucké, and Y. Berbers, "Parallelism viewpoint: An architecture viewpoint to model parallelism behaviour of parallelism-intensive software systems," Jun-2010. [Online]. Available: https://lirias.kuleuven.be/handle/123456789/272344. [Accessed: 05-Jun-2012].
14. K. Pohl, G. Böckle, F. van der Linden. Software Product Line Engineering – Foundations, Principles, and Techniques, Springer, 2005.
15. M. Rosenmüller and N. Siegmund. Automating the Configuration of Multi Software Product Lines. In Proceedings of the International Workshop on Variability Modelling of Software-intensive Systems (VaMoS). Linz, Austria, Jan. 2010.
16. N. Rozanski and E. Woods, Software Systems Architecture: working with stakeholders using viewpoints and perspectives: Addison Wesley 2005.
17. J.A. Zachman. A Framework for Information Systems Architecture. IBM Systems Journal, Vol. 26. No 3, pp. 276-292, 1987.