Gömülü Yazılım Projelerinde Gözlemlenen Modelleme Yaklaşımları Kalıp ve Kültürleri:

Tanımlamaya Doğru İlk Adımlar

Deniz AKDUR[1], Onur DEMİRÖRS[2]

[1]ASELSAN A.Ş., Ankara
[2]Bilgisayar Mühendisliği Bölümü, İYTE, İzmir

denizakdur@aselsan.com.tr
onurdemirors@iyte.edu.tr

Özet. Tasarım, geliştirme ve sınanması diğer yazılım sistemlerine göre daha karmaşık olan yazılım-yoğun gömülü sistemlerde, artan karmaşıklıkla başa çıkmak için kullanılan en etkin yöntemlerden biri yazılım modellemesidir. Gömülü yazılım endüstrisinde kullanılan diyagramların öz niteliklerinin (örneğin, amaç, modelleme katılığı, kullanılan medya tipi, modelleme paydaşının profili, hedef sektör, vb.) farklılaşması, modelleme yaklaşımlarının da hem sektörler hem de sistemler özelinde değişiklik göstermesine neden olmaktadır. Endüstrideki en son modelleme kullanımlarını ortaya çıkartarak modelleme sırasında kullanılan diyagramların öz niteliklerini ve birbirleriyle olan ilişkilerini karakterize eden önceki çalışmalarımız ışığında bir karakterizasyon modeli oluşturulmuştur. İki aşamada oluşturulan bu model, gömülü yazılım geliştirme projelerinde gözlemlenen modelleme yaklaşımı kalıp ve kültürlerini ortaya çıkarmakla kalmamış, modelleme paydaşına etkin bir modelleme yaklaşımı için öneriler sunarak, yapılan çoklu vaka çalışmaları ile de doğrulanmıştır. Bu bildiri ile, henüz bir benzeri olmayan bu modeli ortaya çıkartırken izlenen ilk aşama süreçlerinin anlatılması ve Türkçe karşılığı olmayan modelleme ile ilgili terimlerin kullanılmasıyla ulusal anlamda Türkçe'ye katkıda bulunulması hedeflenmiştir.

Anahtar Kelimeler: gömülü yazılım, modelleme, kabataslak çizim, model-tabanlı, model-güdümlü, modelleme kalıp ve kültürleri, modelleme karakteristikleri, diyagram öz nitelikleri

Modeling Patterns and Cultures of Embedded Software Development Projects:

Towards Preliminary Characterization Model

Abstract. Due to their multiple constraints across different dimensions of performance and quality, the analysis, design, implementation and testing of software-intensive embedded systems are not trivial, which makes their development more challenging. To cope with these growing complexities, modeling is a widely used approach in this industry. The modeling approaches in embedded software vary since the characteristics of diagram development and usage (e.g., purpose, modeling rigor, medium type used, modeling stakeholder profile, target sector, etc.) differ among systems as well as among sectors. In the light of our previous studies, which figures out the current state-of-practice of modeling to investigate the relations between the characteristics of software modeling and also the significant parameters, this study proposes a characterization model in two iterations. This model, which is validated by multiple case studies, not only identifies and defines modeling patterns and cultures of the modeling stakeholder in embedded software development projects, but also gives recommendations for commonsense modeling practices. In this article, the processes to create

the preliminary version of this model is presented besides enriching Turkish Software Engineering Literature with the modeling terminologies.

Keywords: embedded software, modeling, sketch, model-based, model-driven, modeling patterns and cultures, modeling characteristics

1 Giriş

Farklı performans ve kalite kriterlerine sahip olan gömülü yazılımların tasarım, geliştirme ve test süreçleri diğer sistemlere göre daha karmaşık; donanım ve yazılım alt parçalarının tek bir sistemde toplanmasıyla daha zorludur [1]. Gömülü yazılım endüstrisinde yaygın olarak kullanılan modelleme yaklaşımları, bu zorluklar ve artan karmaşıklıkla başa çıkmak için bir araç olarak görülmektedir. Farklı uygulama alanlarında neden, nasıl ve hangi sıklıkla kullanıldığı değişen modelleme yaklaşımları gibi, kullanılan diyagramların öz nitelikleri de değişebilmektedir. Uç bir örnek olarak, bir modelleme paydaşı kâğıt üstünde kabataslak diyagram çizip sadece fikir alışverişi yapmak isteyebilir. Bu durumda amaç, tanımlı bir spesifikasyon yerine, hızlı bir iletişim olduğundan bu diyagram ya atılır ya da yazılan kodla uyumlu olmadığından bir süre sonra tutarsız hale gelebilir. Diğer bir uç örnekte ise, yazılım modellemesi programlama diline dönüştüğünden, programcılar tüm yazılım geliştirme yaşam döngüsü (YGYD) çıktılarını (örneğin, kod, doküman, test simülatörü) bu modeller aracılığıyla oluşturabilir. Bu durumda ise bu diyagramların kullanım ömürleri ve arşivlenebilirliği daha fazladır. Aynı şirkette farklı bölümdeki paydaşlar bile modelleme yaklaşımlarını farklı amaç ve YGYD evrelerinde kullanabilirler [2].

Gömülü yazılım endüstrisindeki modelleme yaklaşımlarının son durumunu anlamak için 2015 yılında gerçekleştirdiğimiz çalışmamız, kimlerin ne amaçla, nasıl modelleme yaptığını belirlemiştir [3, 4]. Dünyadaki en son kullanım ve eğilimi ortaya çıkarmakla kalmayan ve ulusal anlamda bu alanda ilk olan diğer çalışmamız [5], gömülü yazılım profesyonellerinin farklı ihtiyaçlar için farklı modelleme yaklaşımları kullandıklarını göstermiştir (kabataslak çizimden otomatik kod oluşturabilmek için formal modellere kadar). Tüm bu yaklaşımların hiç biri en iyi ya da en kötü olarak değerlendirilemez; modellemenin amacına göre verimli ve maliyet-etkin olabilir. Elde ettiğimiz sonuçlar gömülü yazılım geliştirmede bir modelleme katılığı (formalitesi) olduğunu, bunun da kullanılan diyagramların öz niteliklerine, başka bir deyişle modelleme karakteristiklerine bağlı olarak değiştiğini göstermiştir.

Ne zaman, nasıl ve ne kadar modelleme katılığı ile (örneğin, kabataslak çizim yaparken bir modelleme diline bağlı kalmadan ya da otomatik kod oluşturma yaparken model güdümlü mühendislik (MGM) kullanarak) modelleme yapmak yazılım geliştirme ekipleri için sıklıkla sorulan sorular arasındadır. Bu zorlukları adreslemek ve çözüm önermek için modelleme karakteristiklerini (örneğin, modelleme katılığı, amaç, paydaş, YGYD evresi, fayda, maliyet vs) ve bunların birbirleriyle olan kritik ilişkilerini ortaya çıkarmak önemlidir. Bu soruna olası çözüm yaklaşımlarından biri, yazılım iyileştirme süreçlerinde de kullanılan (örneğin, yazılım alt-kültürleri [6]), "modelleme kalıp ve kültürlerini" ortaya çıkartarak tanımlayan bir karakterizasyon modelidir.

Literatürde modelleme yaklaşımlarını sınıflandıran sadece iki çalışma vardır (Bölüm 2.1). Literatürdeki bu boşluğu dolduran ve gömülü yazılım geliştirme projelerindeki modelleme yaklaşım kalıp ve kültürlerini ortaya çıkaran çalışmamız iki aşamadan oluşmaktadır. İlk adım olarak, önceki sonuçların ışığında [3, 4, 7], modelleme sırasında kullanılan diyagramların öz nitelikleri ve bunların birbirleriyle olan ilişkileri karakterize edilmiş [8] ve karakterizasyon modelinin ilk versiyonu ortaya çıkarılmıştır. Nicel verilere (anket sonuçlarının analizine) dayanan bu çıkarım, ikinci aşamada daha derin ve nitel verilerle (yarı-yapılandırılmış mülakatlar aracılığıyla) güçlendirilerek genişletilmiştir. Bu süreç sonunda gizli kalmış modelleme yaklaşım kalıpları (*ing. "hidden patterns"*) da ortaya çıkarılıp tanımlanarak [9], karakterizasyon modelinin temeli atılmıştır. Bu modelle, gömülü yazılım geliştirme projelerinde kullanılan modelleme yaklaşım kalıpları ve kültürleri ortaya çıkarılmakla kalmamış, modelleme paydaşına etkin bir modelleme yaklaşımı için öneriler de verilmiştir. Bu bildiride, anket sonuçlarının analizi ve bu karakterizasyon modelinin oluşturulmasının ilk aşamasını (*ing. "preliminary"*, ön hazırlık) oluşturan süreçler anlatılacaktır. Ayrıca, anlatım sırasında, Türkçe karşılığı olmayan modelleme ile ilgili terimlerle ulusal anlamda Türkçe'ye katkıda bulunulması da hedeflenmiştir.

Bu bildirinin devamı şu şekilde yapılandırılmıştır: Yazılım modellemesi yaklaşımlarını sınıflandıran önceki çalışmalar, terminoloji ve kavramlar ile önceki çalışmalarımız hakkında kısa bir bilgilendirme 2. bölümde sunulmuştur. 3. bölümde araştırma yöntemimiz verilmiştir. Karakterizasyon modelinin oluşturulmasının ilk aşaması, dolayısıyla bu bildirinin asıl amacı 4. bölümde sunulmuştur. Geçerliliğe tehditleri veren 5. bölüm sonrasında sonuç ve planlanan çalışmalar ile bildiri sonlandırılmaktadır.

2 Genel Bilgiler

2.1 Yazılım Modelleme Kalıpları ile İlgili Önceki Çalışmalar

Literatürde "kalıp" (*ing. "pattern"*) için bir çok tanım vardır. Bir olay ya da problemin tanımlanmasına yardımcı olan, tutarlı ve tekrarlanabilir karakteristik(ler) [10] olarak kullanılan bu terim, Türkçe'de örüntü ya da desen olarak da tanımlanabilmektedir. Yazılım Mühendisliği literatüründe de bulunan bu kavram, tekrarlanan tasarım zorluklarına karşı kanıtlanmış çözümleri içeren yazılım tasarım kalıpları (*ing. "software design patterns"*) olarak da karşımıza çıkmaktadır [11]. Bu çalışmada kullanılan "modelleme yaklaşım kalıbı", bir modelleme paydaşının modellemeyi neden, nasıl ve hangi yaklaşımla kullandığını ortaya çıkarmaya yarayan ve belirli modelleme karakteristiklerini (örneğin, amaç, medya tipi, modelleme dili, YGYD evresi gibi diyagram öz nitelikleri) içeren bir terim olarak tanımlanmıştır.

Literatürde modelleme yaklaşımlarını inceleyen iki çalışma bulunmaktadır. Kleppe, Warmer ve Bast, modelleme karakteristiklerinden sadece birine odaklanarak (modelleme katılığı, *ing. "modeling rigor"*) modelleme yaklaşımlarını olgunluk seviyelerine göre sınıflandırmıştır [12]. Kleppe ve diğ. göre, yazılım geliştirme projelerinde modelleme katılığını baz alan altı adet (0'dan 5'e) Modelleme Olgunluk Seviyesi (MOS) bulunmaktadır. Ancak, önceki çalışmalarımız, modelleme sürecindeki farklı modelleme karakteristiği ve proje ihtiyaçlarının, modelleme paydaşını bir üst seviyeye çıkmaya zorlamasının doğru olmadığını göstermiş; farklı modelleme yaklaşımlarının kendi içinde maliyet-etkin çözümler yaratabilirken, sadece tek bir karakteristiğe odaklanarak (örneğin, modelleme katılığı) diğer öz niteliklerin göz ardı edilmesinin organizasyonları ve paydaşları yanlış yönlendirebileceği değerlendirilmiştir. Aslında, modelleme karakteristiklerinin çeşitliliği farklı amaçlar, işler ve görevlerle ilişkilidir.

İkinci çalışmada Petre, modelleme paydaşlarından sadece yazılım geliştiricilere odaklanarak UML (Birleşik Modelleme Dili, *ing. "Unified Modeling Language"*) kullanımı kategorilerini araştırmıştır [13]. UML kullanımının pratikte ne anlama geldiğine dair çok farklı algılar ve yöntemler olduğunu savunan Petre, UML kullanım çeşitlerini beşe ayırmıştır. Ancak, gömülü yazılım geliştirme sırasında, değişik görevlerde bulunan paydaşlar (yazılım geliştiricisinden test mühendisine, sistem mühendisinden proje yöneticine kadar) UML dışında değişik modelleme dilleri de kullanabilmektedir (kabataslak çizimden, alana özgü dillere (AÖD, *ing. "Domain Specific Language" (DSL)*) [3, 4]. Dahası, MGM ve model güdümlü geliştirmede (MGG) AÖD'lerin UML'e göre çok daha verimli olduğu iddia edilmektedir [14].

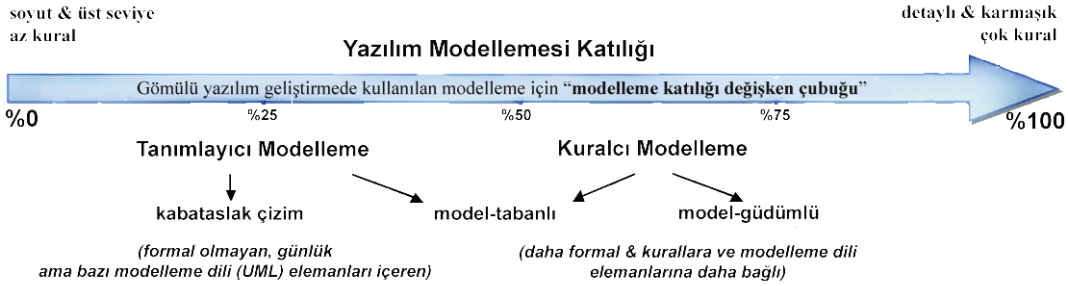
Bu çalışmaların hiç biri gömülü yazılım geliştirme ile doğrudan ilişkili değildir.

2.2 Yazılım Modellemesi ile İlgili Terminoloji ve Kavramlar

Literatürde yazılım modellemesi için çeşitli terminolojiler bulunmaktadır. Brambilla ve diğ. göre, MGG'de diyagramlar geliştirme sürecinin başlıca çıktısı olarak görülür ve kaynak kod genellikle otomatik olarak oluşturulur [15]. Bu bağlamda, MGM kodlama dışında (test ya da bakım gibi) diğer yazılım geliştirme süreçlerini de kapsadığından MGG'nin üst kümesidir. Diğer yandan, Model Tabanlı Mühendislikte (MTM) diyagramlar yine önemli rol oynarken, geliştirme sürecinin anahtar çıktısı değillerdir. Örneğin, bir tasarımcı kâğıt üstüne ya da bir araç kullanarak bir diyagram çizip bunu programcıya elle kodlaması için verdiğinde model-tabanlı bir yöntem izlemiş, ancak diyagram ile kod uyumluluğunun garantisini insan faktörüne (geliştiriciye/programcıya) bırakmıştır.

Gömülü yazılım geliştirme projelerinde gözlemlenen modelleme yaklaşımlarını daha iyi yansıttığı düşünüldüğünden, bu çalışmada genişletilmiş bir terminolojiye ihtiyaç duyulmuştur: "model tabanlı" ve "model güdümlü" terminolojisi [15] (yani, "kuralcı modelleme",

ing.“Prescriptive Modeling” [2]), “kabataslak çizim”, *ing. “sketch”* [16] kavramıyla (yani, “tanımlayıcı modelleme”, *ing. “Descriptive Modeling”* [2]) sentezlenerek zenginleştirilmiştir. Zira, Brambilla ve diğ. [15], endüstride sıkça kullanılmasına rağmen [3, 4], tanımlayıcı modelleme ve kabataslak çizimleri kategorize etmemiştir. Buna göre, yazılım modelleme yaklaşımları temel olarak ikiye (tanımlayıcı ve kuralcı), kullanım kalıpları da dörde (modelleme kullanmayanlar, kabataslak çizim, model tabanlı ve model güdümlü) ayrılmıştır (**Şekil. 1**).



Şekil. 1. Bu çalışmada kullanılan terminoloji

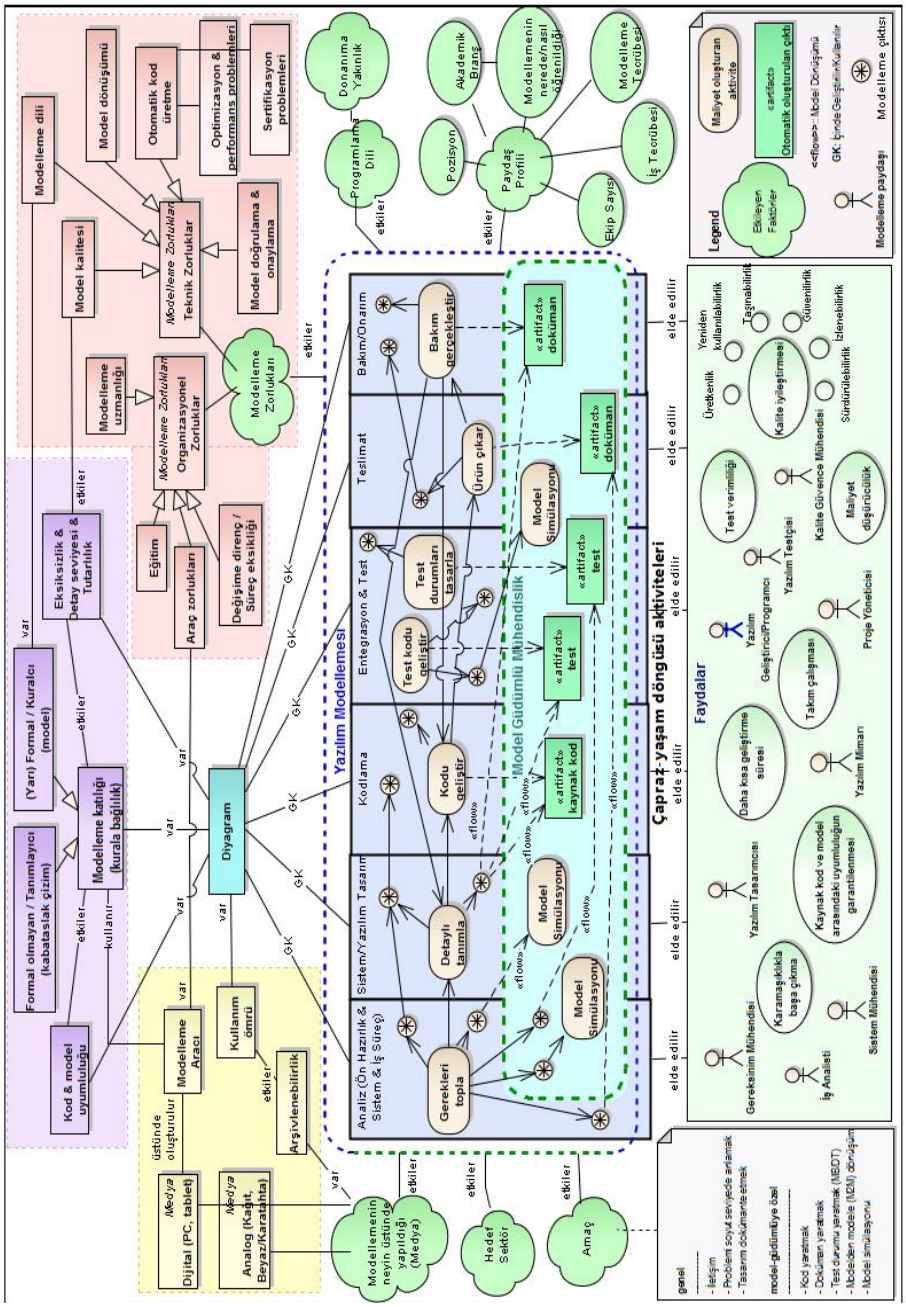
Tanımlayıcı modellemede nesneler, olaylar ve süreçler kategorilere ayrılır; kuralcı modellemede ise sistem bileşenlerinden neyin beklendiği ve nasıl geliştirileceği belirtilir [2]. Tanımlayıcı diyagramlar belli bir amaç (sistem) için gözlemler yaparak oluşturulur ve ne zaman ki bu amaç elde edilir, önemlerini kaybedebilirler. Dahası, bu amaç (sistem) değiştiğinde bu diyagramlar geçersiz olabilmekte ve güncellenmeye ihtiyaç duyulmaktadır. Oysaki kuralcı modellemede amaç (sistem) henüz yoktur ve diyagram o an varolan bilgilerden oluşturulur. Tanımlayıcı modellemede iletişim ve sistemi anlamaya yönelik amaçlar bulunurken, kuralcı modellemede doğrudan geliştirme süreçleri üstüne odaklanılmıştır [2]. Analiz ve tasarım modelleri arasında daha net görülen bu ayrımın modelleme amacına bağlı olarak modellemenin katılığını doğrudan değiştirdiği ve bunun da diyagram öz niteliklerine bağlı olduğu gözlemlenmiştir [8]. **Şekil. 1**’de görüldüğü gibi, modelleme yaklaşımları bu “*modelleme katılığı değişken çubuğu*” üstünde nereye düştüğüne bağlı olarak değişmektedir. Örneğin, soyut ve üst seviye bir modelleme yaklaşımına sahipseniz ve bu yaklaşım katı kurallara uymayı gerektirmiyorsa, çubuk üstünde başlangıca yakınsınız demektir (örneğin, kabataslak çizim yaparken katı UML kurallarına uymadan, günlük kullanım yapmak gibi). Öte yandan, daha detaylı ve karmaşık bir modellemeye, dolayısıyla daha kurallara bağlı bir yaklaşımı sahipseniz, modelleme katılığınız %100'e yakın olmaktadır (örneğin, MGG’de kod oluşturabilmek için sıkı UML kurallarına uymak gibi). Başka bir deyişle, modelleme yaklaşımınız, diyagram öz niteliklerinden (dolayısıyla modelleme karakteristiklerinden) biri olan modelleme katılığınız ile doğrudan ilişkilidir ve modelleme yaklaşımınızı etkilemektedir.

2.3 Önceki Çalışmalarımız

Gömülü sistemlerde kullanılan modellemenin ve MGM’nin şu anki durumunu anlayabilmek ve kimlerin ne amaçla, nasıl modelleme yaptığını belirlemek için uluslararası bir anket hazırlandı. 27 değişik ülke ve farklı gömülü yazılım endüstrilerinden, farklı yazılım mühendisliği rollerindeki modelleme paydaşlarının farklı ihtiyaç ve özelliklere bağlı olarak farklı modelleme yaklaşımlarına sahip olduğu gözlemlendi [3, 4]. Kabataslak çizimden MGM’ye kadar, başka bir deyişle tanımlayıcı modellemeden kuralcı modellemeye kadar değişik modelleme yaklaşımlarını yansıtan bu anket çalışması, aslında tüm yaklaşımların (kabataslak, model-tabanlı, model-güdümlü) belli kriterlere (aslında, modelleme karakteristiklerine) göre amacına hizmet edip maliyet-etkin bir çözüm sağlayabileceğini gösterdi [3, 4].

Anketten elde edilen veriler ve daha önceki Eylem Araştırması (EA) projesinin [7] sonuçlarından sonra, bu modelleme karakteristiklerini ve birbirleriyle ilişkilerini daha iyi anlayabilmek amacıyla, öncelikle bir kavramsal model oluşturuldu [8]. Yarı-yapılandırılmış mülakatlar aracılığıyla, uzman görüşlerinin geri bildirimleri ile son halini alan bu kavramsal model, gömülü yazılım projelerinde kullanılan modellemenin daha iyi anlaşılabilmesi için kendi içinde beş ayrı kavramsal alana ayrıldı: (1) Modelleme katılığı ve sınıflandırılması, (2) Modelleme paydaşları için

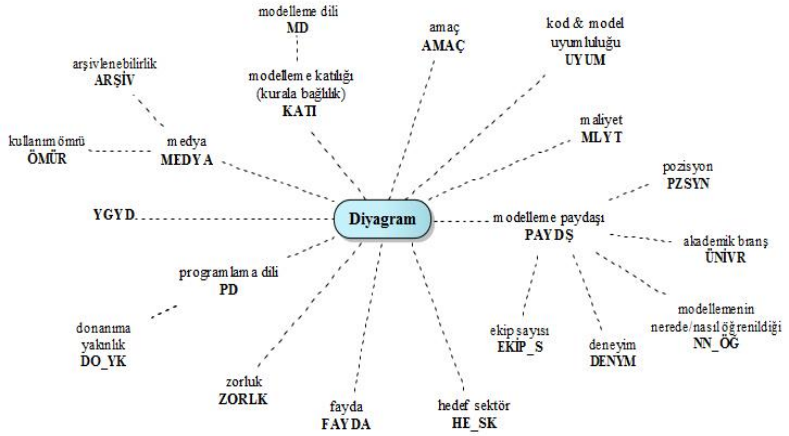
faydalar, (3) Modellemede kullanılan medya tipi ve etkileri, (4) Modelleme zorlukları, (5) Çapraz yaşam döngüsü aktiviteleri (Bknz. **Şekil. 2**)



Şekil. 2. Gömülü sistemlerde modelleme yaklaşımlarının kavramsal modeli

Bu kavramsal modelin bazı UML elemanları içeren (örneğin, seçici ve formal olmadan, bazı sınıf ve kullanım diyagram elemanları) tanımlayıcı bir diyagram olduğuna, başka bir deyişle, iletişim ve soyut seviyede bir problemi anlamak için oluşturulmuş kabataslak bir çizim olduğuna dikkat edilmelidir. Bu kavramsal alanların detaylı açıklamalarına ve kavramların birbirleriyle ilişkilerine [8]'den ulaşılabilir. Bu çalışmada, **Şekil. 2** bu kavramsal modelin Türkçeleştirilmesi hedefiyle sunulmuştur.

Oluşturulan kavramsal model yardımıyla, gömülü yazılım endüstrisinde kullanılan diyagramların öz nitelikleri, dolayısıyla modelleme karakteristikleri ortaya çıkarılmıştır (**Şekil. 3**). Temel olarak 11 öz nitelik ve bunları etkileyen bazı alt niteliklere sahip olan modelleme karakteristiklerine ve bunların birbirleriyle olan korelasyonlarına [8]'den ulaşılabilir.



Şekil. 3. Modelleme karakteristikleri (Diyagram öz nitelikleri)

3 Araştırma Yöntemi

Bu çalışmadaki araştırma yöntemi hem nicel hem de nitel metodlara dayanmaktadır:

- Gömülü yazılım endüstrisinde kullanılan modellemenin şu anki durumunu anlamak ve kimlerin ne amaçla, hangi sıklıkla kullandığını geniş bir yelpazede inceleyen anketimizde [3, 4], Hedef, Soru, Ölçüt (*ing. "Goal Qestion Metric", GQM*) [17] yaklaşımı kullanılmıştır.
- Önceki çalışmalarımızı referans alan [3, 4, 7], gömülü yazılım geliştirmede kullanılan diyagramların kavramsal modelininin oluşturulmasının ilk aşamasında, yazılım mühendisliğinde modelleme üstüne farklı görüş ve sınıflandırmalar temel alınmıştır [18, 19]. Nitel bir yaklaşım olan yarı-yapılandırılmış mülakatlar [20] sırasında uzman görüşleriyle güncellenen bu kavramsal model, gömülü yazılım endüstrisinde geliştirilen ve kullanılan diyagramların öz niteliklerini çıkarmak için bir girdi olarak kullanılmıştır [8].

4 Karakterizasyon Modelinin Oluşturulmasına Doğru

Modelleme karakteristikleri arasındaki ilişkileri belirledikten sonra karakterizasyon modelinin ilk (*ing. "preliminary"*) versiyonunun oluşturulması süreci başlatıldı. Bu süreçte, eğer ~80 öznitelik/özellik *(ing. "attributes/features")* içeren anket verisinde [21], 11 temel modelleme karakteristiğini (**Şekil. 3**) gruplarsak çok sayıda kombinasyon oluşacaktı. Gereksiz kombinasyonları ortadan kaldırıp doğru alt kümeleri seçebilmek amacıyla, başlangıç noktası olarak en önemli ve kritik olan öznitelik/özellikleri belirleyip veri setindeki özellik sayısı azaltılmaya çalışıldı. Bu şekilde, olası bağımlı (*ing. "dependent"*) öznitelik/özellikler ortadan kaldırılarak, tüm kombinasyonların denenmesi yerine, en kritik olanlar denenmeye başlandı [22] (örneğin, modelleme yaklaşımlarındaki kullanımı etkileyen en önemli faktörlerden biri olan modelleme amacının, modelleme yaparken kullanılan medya tipi ile güçlü bir korelasyonu olması [8] ya da modelleme katılığının doğrudan modelleme dili ile ilişkisi olması (Bknz **Şekil. 3**), bu üç modelleme karakteristiğini kritik hale getirdi).

Bu süreçte, anket verisindeki öznitelik/özellik sayısının fazla olması dışındaki bir diğer önemli sorun ise, bu modelleme karakteristiklerinden bazılarının birden çok anket verisi kombinasyonu içermesinden dolayı bunları daha anlamlı hale getirebilmekteki zorluktu. Çünkü bu sorular 5-ölçekli *(ing. "5-point Likert-scale")* ya da çok-cevaplı *(ing. "multiple-response")* seçenekler içerebilmekteydi. Dolayısıyla, var olan veriden, türetilmiş *(ing. "derived")* bir öznitelik/özellik çıkartarak daha az kombinasyonla daha anlamlı gruplar elde edebilmek önemli bir gerekti.

Anket verimizde modelleme yaparken kullanılan dört medya tipi (PC, kağıt, kara/beyaz tahta, tablet/akıllı telefon) 5-ölçekli cevaplarla ("asla (%0)"dan "her zaman (%100)"e kadar) adreslenmekteydi [3, 4]. Bu da bizi "medya tipi grupları" adı altında mantıksal olarak gruplandırılmış bir türetilmiş nitelik çıkartmaya zorladı. Örneğin, eğer anket verisi "*Katılımcı modelleme yaparken PC ve tablet/akıllı telefonu hiç kullanmıyor*" ise (yani PC ve tablet/akıllı telefon kullanımı "asla (%0)" ise), bu "sadece analog medya kullanımı" anlamına gelmekteydi. Benzer şekilde, diğer 5-ölçekli cevaplar dikkate alınarak medya tipine bağlı olarak (analog ya da dijital), diğer 2 türetilmiş nitelik de belirlendi: "analog medya kullanımı dijital medya kullanımından daha fazla ya da eşit" ve "dijital medya kullanımı analog medya kullanımından daha fazla" olan modelleme kalıpları. Böylelikle, bu süreç sonunda, (5x4=20 yerine) üç farklı tipte kullanım içeren "medya tipi grupları" öz niteliği türetilmiş oldu [22].

Buna benzer bir teknik, "modelleme amacı grupları" için de uygulandı [22]. Yazılım modellemesi karakteristiklerinden, modelleme amaçları "model-güdümlüye özgü" bir amaç içerip içermemesine bağlı olarak gruplanabilmekteydi [8]. **Şekil. 2**'den de görüleceği gibi, model-güdümlüye özgü olmayan "genel" amaçlar, "İletişim", "Bir problemi soyut seviyede anlamak" ya da "Analiz & tasarımı dokümante etmek" iken; model-güdümlüye özgü amaçlar "Kod yaratmak", "Test durumu yaratmak (MB/DT)", "Doküman yaratmak", "Model simülasyonu" ve "Modelden modele çevrim" olarak sınıflandırılabilmektedir. Kuralcı ve tanımlayıcı modelleme arasındaki farklar dikkate alındığında [8], YGYD evrelerinden kodlama ve test ile doğrudan ilişkili olduğundan, model-güdümlüye özgü amaçlar arasında "Kod yaratmak" ve "Test durumu yaratmak (MB/DT)", diğer model-güdümlüye özgü amaçlardan ayrışabilmiş ve böylelikle "model-güdümlüye özgü" amaçlar da iki alt gruba ayrılabilmiştir. Diğer taraftan, model-güdümlüye özgü olmayan amaçlar, "Analiz & tasarımı dokümante etmek" amacını içerip içermemesine göre ikiye ayrılabilir. Buradaki çıkış noktası, bu amacın diğer modelleme karakteristiklerini (örneğin, kullanılan medya tipi ve dolayısıyla arşivlenebilirliği) doğrudan ya da dolaylı etkileyebilen ayırt edici bir öz nitelik olmasıdır [8]. Bu süreç sonunda, "modelleme amacı grupları" da dört gruba ayrılabilmiştir: "Kod oluşturmak ya da MBT içeren model-güdümlüye özgü amaçlar grubu" (*ing. "ModelDrivenPurpose WithCodeGenOrMBT"*), "Kod oluşturmak ya da MBT içermeyen model-güdümlüye özgü amaçlar grubu" (*ing. "ModelDrivenPurpose WithoutCodeGenOrMBT"*), "Model güdümlüye özgü olmayıp analiz & tasarımı dokümante etme amacı içeren gruplar" (*ing. "NoModelDrivenPurpose WithDocumentingDesign"*) ve "Model güdümlüye özgü olmayıp analiz & tasarımı dokümante etme amacı içermeyen gruplar" (*ing. "NoModelDrivenPurpose WithoutDocumentingDesign"*).

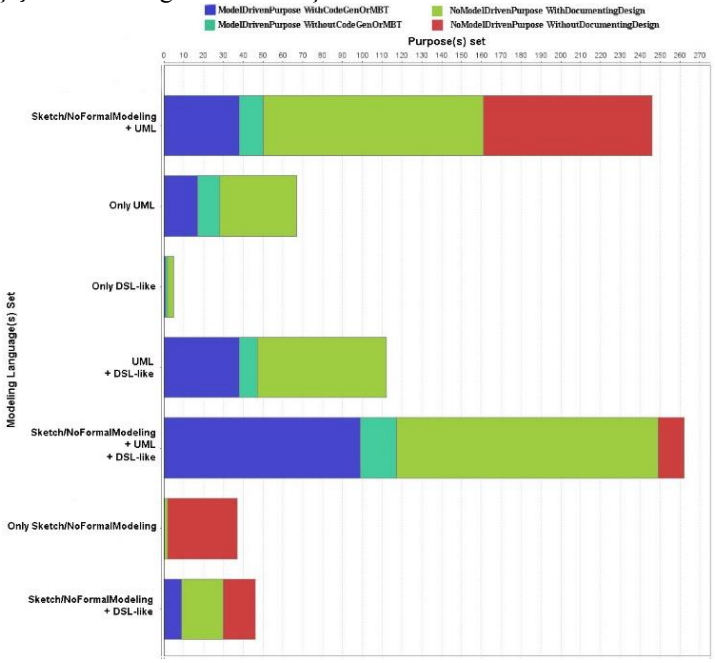
"Modelleme dili grupları" için de aynı süreç uygulanarak yeni türetilmiş özellikler çıkarılmıştır. Anket verisinde çok-cevaplı seçenekler içermesinin yanısıra "boş metin alanı" (*ing. "free-text area"*) da içeren modelleme dilinin analizi diğerlerine göre çok daha fazla kombinasyon çıkaracağından uygun alt kümeleri seçebilmek kritikti. Bu yanıt incelenirken, katılımcıların modelleme yaparken temel olarak "kabataslak çizim", "UML" ve "Her hangi bir AÖD-gibi modelleme dili (*ing. "DSL-like"*) (örneğin, [kendi tasarımı ya da hazır kullanılan] bir AÖD, UML profili, SysML, SoaML, BPML, MATLAB, AUTOSAR, AADL gibi)" kombinasyonlarını kullandıkları gözlemlenmişti [3, 4]. Dolayısıyla, "modelleme dili grupları" da bu kombinasyonları içeren yedi alt gruba (2^3-1 = 7) ayrıştırılarak kümelenmeleri tespit etmek kolaylaştırılmıştır.

Mevcut anket verisinden türetilmiş öz nitelik çıkartılmasının önemi ve gereğini daha iyi gösterebilmek için, bu çalışmada, "modelleme dilleri" ve "modelleme dili grupları" örnek olarak seçilmiştir (Süreç sırasında izlenen yöntem, kullanılan araçlar ve detaylı sonuçlar için [22] kullanılabilir. Bu bildiride, bütünlük oluşturulması için [22]'deki original İngilizce şekiller kullanılmıştır).

Hatırlanacağı gibi, modelleme amaçları dört gruba ayrılmıştı. Modelleme amacı grupları ile modelleme dilleri arasındaki ilişkiyi gösteren **Şekil. 4** incelendiğinde, çok sayıda modelleme dili kombinasyonu olduğu ve gruplandırma yapmanın ne kadar zor olacağı anlaşılabilmektedir.

Şekil. 4. Yığılmış çubuk (*ing. bar stacked*) – "Modelleme amacı grupları" ve "modelleme dilleri"

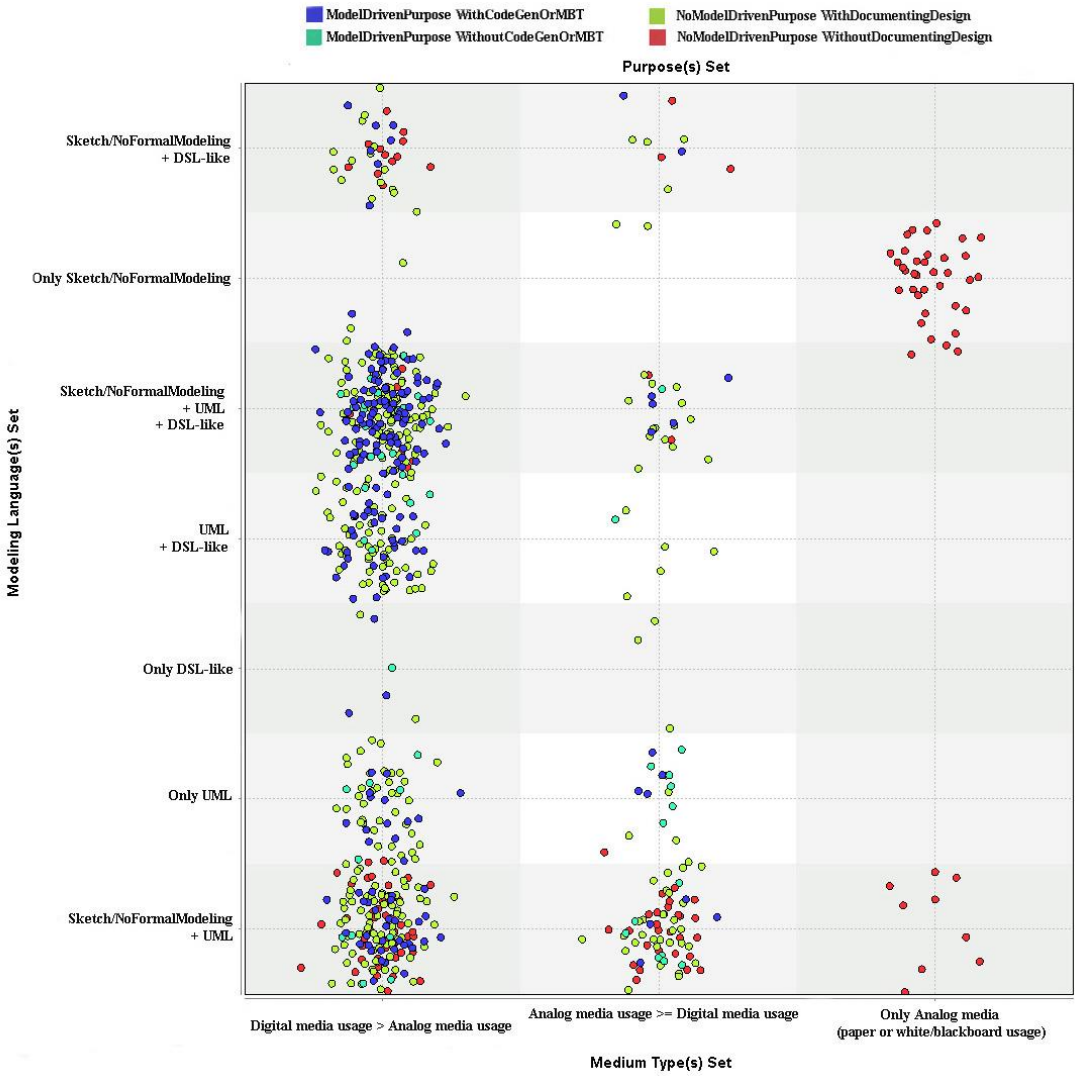
Oysa ki, aynı grafiğin y-ekseninde, türetilmiş olan "modelleme dili grupları" öz niteliği kullanılırsa, sonuç **Şekil. 5**'deki gibi daha anlaşılır ve anlamlandırılabilir bir hal almaktadır.



Şekil. 5. Yığılmış çubuk (*ing. bar stacked*) – "Modelleme amacı grupları" ve "modelleme dili grupları"

Yukarıda bahsi geçen üç türetilmiş öz niteliğin ("modelleme amacı grupları", "modelleme dili grupları" ve "medya tipi grupları"), birbiriyle olan ilişkileri dağılım (*ing. "scatter"*) grafiği çizildiğinde, anket verilerinden çıkan modelleme yaklaşım kalıpları **Şekil. 6**'da daha net görülmüştür. **Şekil. 6**'daki veriler üstünde daha detaylı analiz yapıldığında, aslında model-tabanlı kullanıcılardan, amaçları "Model güdümlüye özgü olmayıp analiz & tasarımı dokümante etmek amacı içeren" grupların (yani *NoModelDrivenPurpose WithDocumentingDesign* grubunun – ki anket verilerinin %38,6'sını içermektedir) kendi arasında da YGYD evrelerinde "Kodlama" ve/veya "Test" aşamasını içerip içermemesine bağlı olarak alt-gruplara ayrılabileceği gözlemlenmiştir [22]. Buradaki kümelenme de aslında modelleme paydaşının "tanımlayıcı" veya "kuralcı" modellemeye ne kadar yakın olduğuyla ilgilidir.

Önceki çalışmalarımız ışığında görselleştiren bu kümelenmelerle, modelleme yaklaşım kalıpları sınıflandırması için kritik öneme sahip modelleme karakteristikleri elde edilmiş olmuştur: *"modelleme amacı", "kullanılan medya tipi", "arşivlenebilirlik", "modelleme dili (kullanılıyorsa)"* ve "*YGYD evresi"* [22].



Şekil. 6. Dağılım (*ing. scatter*) grafiği – "Modelleme amacı", "modelleme dili" ve "medya tipi" grupları

Bu çok kaynaklı araştırma sürecinin sonunda **Tablo 1**'de sunulduğu gibi dokuz modelleme yaklaşımı kalıbı belirlenmiştir ("model-tabanlı" ve "kabataslak" kategorisinin, anket hazırlanırken daha iyi bir terminolojiye sahip olamadığımızdan dolayı aynı grupta olduğu fark edilebilir [3, 4]).

Tablo 1. Anket sonuçlarının analizinden sonra çıkarılan modelleme yaklaşımı kalıpları

Ana Yaklaşım	**Modelleme Yaklaşımı Kalıpları**				**Anketteki oran (%)**	
model-güdümlü	3.3	AÖD-gibi* ile	Modelleme amacı "Kod yaratma" ve/veya "Test durumu yaratma (MB/DT) içerir	AÖD-gibi kullanımı içerir.	16,9	29,5
	3.2	AÖD-gibi olmadan		Her hangi bir AÖD-gibi kullanımı içermez.	6,5	
	3.1	Sınırlı	Modelleme sadece "Doküman yaratma", "Model simülasyonu" veya "Modelden modele çevrim" amacıyla kullanılır.		6	
model-tabanlı	2.2	Kuralcı	Modelleme, YGYD evrelerinden "Kodlama" ve/veya "Test" de kullanılır.		24,9	59,5
	2.1	Tanımlayıcı	Modelleme "Kodlama" ya da "Test" YGYD evrelerinde kullanılmaz.		13,7	
kabataslak	1.3	Arşivlenir	Modelleme amacı analiz veya tasarımı "dokümante etmek"tir. Bu süreçte, analog medya kullanımı dijital medya kullanımından daha fazla ya da aynıdır.		3,6	
	1.2	Seçici	Modelleme günlük & informal bir şekilde ve bazı formal modelleme dili elemanlarını (genellikle UML elemanları) seçici olarak kullanarak gerçekleştirilir. Dolayısıyla, modelleme dili grubu kabataslak çizim ve her hangi bir formal modelleme dili içerir (örneğin, UML ve/veya AÖD-gibi).		13,1	
	1.1	Tasarsız/Plansız (*ing. ad-hoc*)	Modelleme amacı sadece "İletişim" ve "Problemi soyut seviyede anlama" içermektedir. Sadece kağıt & kalem kullanarak ve serbest formatta (her hangi bir modelleme dili elamanı içermeden) yapılır. Kullanılan medya tipi sadece analog'tur (kağıt veya tahta)		4,1	
kullanmayanlar	0	Modelleme yok	Her hangi bir modelleme yapılmaz.		11	

* "AÖD-gibi ile" tanımlaması, modelleme paydaşının kullandığı modelleme dilleri grubu içinde AÖD tabanlı dillerden en az birinin olduğunu göstermektedir (örneğin, [kendi tasarımı ya da hazır kullanılan] bir AÖD, UML profili, SysML, SoaML, BPML, MATLAB, AUTOSAR, AADL gibi)

Anket sonuçlarının analizinden (nicel verilerden) elde edilen **Tablo 1**'deki bu modelleme kalıplarının güçlendirilip doğrulanması amacıyla yarı-yapılandırılmış mülakatlar ile sadece anket verisinden elde edilemeyecek gizli kalmış modelleme yaklaşımı kalıpları da bulunarak [9], karakterizasyon modelinin temeli atılmıştır [22].

5 Geçerliliğe Tehditler

Bu bölümde, çalışmamıza girdi olarak kullandığımız anket çalışmamız, endüstri uzmanlarıyla yaptığımız mülakatlar ve karakterizasyon modelinin ilk versiyonundaki kritik öz nitelik belirleme süreci sırasında karşımıza çıkıp çalışmamıza sınır teşkil edebilecek olası geçerliliğe tehditlerin nasıl azaltıldığı ve ortadan kaldırıldığı verilmektedir.

Bu çalışmada, birçok kaynaktan veri toplanmış; yanlı cevap alınmaması adına anket anonim olarak yapılmıştır. Ölçüm yöntemi, birçok çalışmada kullanılan, her soru için istatistiki çıkarımlar yapmaya dayandığından, sonuçların modelleme yaklaşım teknikleri, kullanım ve görüşlerini yansıttığına ve yapısal geçerliliği sağladığına inanılmaktadır.

İngilizce olarak hazırlanan ankette kullanılan kelimelerin herkes tarafından anlaşılır olması kaliteli veri toplamak adına önemli olduğundan, ankette uygulanan pilot çalışma ile içsel geçerlilik güçlendirilmiştir. Uzmanlardan gelen geri bildirim ve önerilerle bazı model elemanlarının eklenip çıkarılması ya da birleştirilmesiyle model güncellenerek içsel geçerliliğe katkıda bulunulmuştur.

Dış geçerliliğe tehditlerden birisi hem anket hem de mülakatlardaki katılımcıların demografik dağılımında yatmaktadır. Değişik gömülü yazılım endüstrilerinden, farklı yazılım mühendisliği rolü ve tecrübelerindeki modelleme paydaşlarının farklı modelleme yaklaşımlarının incelenmesi hedeflenmiş; yazarların bireysel gönderilen davetlerinin yanı sıra çevrimiçi sosyal ve profesyonel ağlar (örneğin, Linkedin ve Twitter), forumlar, yazılım mühendisliği ve üniversitelerin ilişkili email gruplarında duyurular yapılmıştır. Mülakatlar sırasında geniş bir çerçevede geri bildirimler alınmış, uzmanlar farklı profillerden seçilmiştir. Yapılan ölçümlerin güvenirliği de gözden geçirme ve pilot çalışmalarla desteklenmiştir.

6 Sonuç ve Gelecek Çalışmalar

Bu çalışmada, gömülü yazılım geliştirme projelerinde gözlemlenen modelleme yaklaşımı kalıplarının ortaya çıkarılmasına çalışılmış, modelleme paydaşına etkin bir modelleme yaklaşımı için öneriler de sunan bir karakterizasyon modelinin oluşturulması için ilk süreçler anlatılmıştır. Gömülü yazılım endüstrisinde kullanılan diyagramların öz niteliklerini ve birbirleriyle olan ilişkilerini karakterize eden önceki çalışmalarımızda kullanılan, Türkçe karşılığı olmayan terimlerin kullanılmasıyla da ulusal anlamda katkıda bulunulmuştur.

Kavramsal model ve kullanılan terminolojinin sanayi ortamında yayılması ve modelleme karakteristikleri arasındaki ilişkilerin daha iyi anlaşılması, farklı modelleme yaklaşımı kalıplarını kendi ihtiyaçlarına göre uyumlandırmak adına önemlidir. Zira bu çıkarımlar, modelleme karakteristikleri arasındaki ilişkileri kavrayarak maliyet-etkin modelleme çözümleri gerçekleştirmeye yardımcı olacaktır. Ulusal anlamda da katkısı olacağı düşünülen bu çalışma ile, endüstrideki sorunlar öz nitelik ilişkileri ile adreslenebilecek; akademik dünya ile modelleme paydaşları arasında olası işbirlikleri artırılacaktır (Örneğin, gömülü yazılım uzmanları kendi modelleme yaklaşımlarını daha iyi anlayabilecek, aynı profildeki paydaşların modelleme yaparken kullanıdıkları pratikleri öğrenebilecek; araştırmacılar endüstri ve paydaş bazında modellemedeki zorluklara ayrı ayrı odaklanabilecek; eğitimciler ise farklı paydaş profilleri için yazılım modelleme eğitimlerini geliştirebilecektir).

Nicel veriler üstüne yoğunlaşan bu çalışmadaki modelleme kalıpları, bir sonraki nitel ağırlıklı çalışmamızda, gizli kalmış modelleme yaklaşımı kalıplarını da bularak genişletilmiş ve 12 adet modelleme yaklaşım kalıbı tanımlanmıştır [9]. Bu süreçteki mülakatlarda, katılımcıların belli modelleme karakteristiklerine göre (örneğin, akademik branş, modellemeyi nerede öğrendiği, yazılım mühendisliği rolü ya da bu rolde yaptığı işe göre) modellemeye karşı bazı direnç ve ön yargılarının olduğu ortaya çıkarılmıştır. Bazı gömülü yazılım uzmanlarının yazılım modellemesinin sadece bir araç vasıtasıyla veya formal UML diyagramları kullanarak yapılabileceğini düşündüğü (oysa ki yazılım modellemesinin UML ile sınırlı olmayıp AÖD kullanımı ve kabataslak çizimlerin de çok yaygın olduğu); bazılarının ise modelleme, hatta bunun ötesinde MGM yaptığının farkında olmayıp, bu mülakatlar sırasında farkına vardığı gözlemlenmiştir [9]. Karakterizasyon modeli içinde yer alacak tüm modelleme yaklaşımı kalıplarını ortaya çıkaran bu mülakatlar sonrasında, modelleme paydaşını gerekli ve yeterli maliyet-etkin süreç & araç iyileştirmeleri konusunda yönlendirebilmek adına altı adet modelleme yaklaşımı kültürü tanımlanmıştır [22]. Anket verisinden oluşturulan benzer özelliklere sahip paydaşların modelleme pratiklerini içeren veritabanının karakterizasyon modeline eklenmesiyle de, dünyada bu alanda oluşturulmuş ilk model literature kazandırılmıştır [22]. Farklı gömülü yazılım sektörlerindeki farklı modelleme paydaşlarının bulunduğu çoklu vaka çalışmaları ile de sınanıp doğrulanan bu model, organizasyonlar için de modelleme yaklaşımlarında en iyi standartlaşma yaklaşımına karar vermeye yardımcı olmuştur [22].

Bundan sonraki çalışmalarımızda, değişik modelleme yaklaşım kalıplarının benimsenmesi sırasında ortaya çıkan/çıkabilecek, paydaş profili, farkındalık ve organizasyonel direnç gibi, hem teknik hem de sosyal faktörlerin etkileri konusunda çalışma yapılması planlanmaktadır.

Teşekkür. Bu çalışmalara katılan tüm gömülü yazılım profesyonellerine teşekkür ederiz.

Kaynaklar

1 C. Walls, *Embedded Software*: Elsevier Inc., 2012.

2 R. Heldal, P. Pelliccione, U. Eliasson, J. Lantz, J. Derehag, and J. Whittle, "Descriptive vs prescriptive models in industry," in *ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, France, 2016.

3 D. Akdur, V. Garousi, and O. Demirörs, "A survey on modeling and model-driven engineering practices in the embedded software industry," *Journal of Systems Architecture: Embedded Software Design (JSA),* Submitted, In review, 2018.

4 D. Akdur, V. Garousi, and O. Demirörs, "MDE in embedded software industry, Technical Report," *METU II-TR-2015-55, https://dx.doi.org/10.6084/m9.figshare.4262990,* 2015, Last accessed: Nov. 27, 2016.

5 D. Akdur, V. Garousi, and O. Demirörs, "Gömülü Sistem Mühendisliğinde Kullanılan Yazılım Modellemesi ve Model Güdümlü Teknikler Anketi: Türkiye Sonuçları," presented at the 9th Turkish National Software Engineering Symposium (In Turkish: Ulusal Yazılım Mühendisliği Sempozyumu (UYMS)), İzmir, Turkey, 2015.

6 G. M. Weinberg, *Quality software management (Vol. 1): systems thinking*: Dorset House Publishing, 1992.

7 D. Akdur and V. Garousi, "Model-Driven Engineering in Support of Development, Test and Maintenance of Communication Middleware: An Industrial Case-Study," in *International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, 2015.

8 D. Akdur, O. Demirörs, and V. Garousi, "Characterizing the development and usage of diagrams in embedded software systems," in *43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Vienna, Austria, 2017.

9 D. Akdur, O. Demirörs, and B. Say, "Towards Modeling Patterns for Embedded Software Industry: Feedback from the Field," in *44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Prag, Czech Republic, 2018.

10 "pattern," ed: Cambridge Dictionary, 2017.

11 B. P. Douglass, *Real-Time Design Patterns : robust scalable architecture for Real-time systems*. Boston, MA: Addison-Wesley, 2003.

12 A. G. Kleppe, J. Warmer, and W. Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*: Addison-Wesley Longman Publishing Co., Inc., 2003.

13 M. Petre, "UML in practice," in *35th International Conference on Software Engineering (ICSE)*, 2013, pp. 722-731.

14 J. Greenfield, K. Short, S. Cook, and S. Kent, *Software Factories - Assembling Application with Patterns, Models, Frameworks and Tools*: Wiley Publishing, 2004.

15 M. Brambilla, J. Cabot, and M. Wimmer, "Model-driven software engineering in practice," *Synthesis Lectures on Software Engineering,* vol. 1, 2012.

16 (2016). *Career Award Talk - Bran Selic*. Available: https://www.youtube.com/watch?v=9qPbGksB3d4

17 V. C. Basili, G.; Rombach, D.H., "The Goal Question Metric Approach," in *Encyclopedia of Software Engineering*, ed: Wiley, 1994.

18 J. Cabot. (2009). *Relationship between MDA,MDD and MDE*. Available: http://modeling-languages.com/relationship-between-mdamdd-and-mde/

19 S. Baltes and S. Diehl, "Sketches and diagrams in practice," presented at the Proceedings of ACM SIGSOFT International Symposium on Foundations of Software Engineering, China, 2014.

20 P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*: Wiley Publishing, 2012.

21 D. Akdur, V. Garousi, and O. Demirörs, "MDE in embedded SW industry-Raw survey data," *https://dx.doi.org/10.6084/m9.figshare.4262972,* 2015, Last accessed: Nov. 27, 2016.

22 D. Akdur, "Modeling Patterns and Cultures of Embedded Software Development Projects," *Thesis, Doctor of Philosophy (PhD), Information Systems, Middle East Technical University (METU), www.researchgate.net/publication/322701453_Modeling_Patterns_and_Cultures_of_Embedded_Software_Development_Projects,* Feb. 1, 2018.