DOOB: RAHAT Ürün Olarak Komuta Kontrol Yazılımı ve Geliştirme Deneyimleri

Murat Şahin, Betül Bostancı, Tuba Yağlı, ve Turgay Yılmaz

Havelsan A.Ş.
Komuta Kontrol ve Savaş Sistemleri
Ankara, Türkiye
{muratsahin,bbostanci,tkizik,tyilmaz}@havelsan.com.tr

Özet. Komuta kontrol bilgi sistemleri (KKBS), askeri organizasyonlar tarafından her türlü planlama, görevlendirme ve harekât faaliyetlerinin bütünleştirilmesi ve gerçekleştirilmesinin kolaylaştırılması amacıyla kullanılmaktadır. Her ne kadar farklı askeri organizasyonlar kendi KKBS'leri için farklı gereksinim kümeleri ortaya koysa da, tüm organizasyonlar için ortaklaştırılabilecek bir takım gereksinimler tespit edilebilmektedir. Bu doğrultuda, HAVELSAN tarafından NATO destekli JC3IEDM veri modeli baz alınarak bir "komuta kontrol ortak gereksinim kümesi" belirlenmiş, bu gereksinimler doğrultusunda RAHAT (Rafta Hazır Ticari Ürün) ürün olarak piyasaya sürülebilecek bir KKBS geliştirilmiştir (DOOB – Defence Out Of Box). Bu makalede, DOOB ürünü kapsamında gerçekleştirilen yazılım geliştirme faaliyetleri, ürüne ait sistem mimarisi, faydalanılan uygulama geliştirme çatıları ile yazılım desenleri anlatılmakta, ayrıca sistemin gerçeklemesine dair kazanılmış olan deneyimler ve karşılaşılan sorunlar paylaşılmaktadır.

Anahtar Kelimeler: komuta kontrol uygulamaları, rahat ürün geliştirme, komuta kontrol, tasarım desenleri, apache wicket, spring

1 Giriş

Komuta kontrol (KK) kavramı genel olarak komutan veya diğer karar vericilerin harekatları yürütebilmek için gerekli her türlü kaynağın düzenlenmesi ve organize edilmesi olarak tanımlanmaktadır. Bu bağlamda Komuta Kontrol Bilgi Sistemleri (KKBS) tüm komuta kademeleri (milli, stratejik, operasyonel ve taktik) için tüm askeri harekatların planlanması, yürütülmesi ve bu harekatlar kapsamında yapılacak görevlendirme faaliyetlerinin eşgüdüm içinde gerçekleştirilebilmesi ve otomasyonunda kullanılmaktadır.

Her askeri organizasyonun istihbarat, hava savunma, levazım gibi farklı ilgi alanları olduğundan, bu ilgi alanlarına göre özelleşmiş KKBS'ler beraberinde farklı ihtiyaçlar getirmektedir. Askeri ilgi alanları ve kullanılan KKBS'lerin farklı ihtiyaç ve özellikleri bulunmasına rağmen, tüm ilgi alanı kullanıcılarının kullandığı KKBS'den beklediği temel fonksiyonaliteler bulunmaktadır. Entegre olmuş bir CBS (Coğrafi Bilgi Sistemi) bulundurması, temel askeri mesajlaşma standartlarını desteklemesi, muharebe alanı elemanlarını görüntülemesi, detay

bilgilerinin gösterilmesi ve semboloji standardı desteği gibi ihtiyaçlar temel KKBS fonksiyonlarına örnek olarak gösterilebilir.

Birçok farklı askeri ilgi alanına yönelik olarak uzun yıllardır kara, deniz ve hava kuvvetleri için KKBS uygulamaları geliştirmekte olan bir kurum olarak HAVELSAN, yukarıda anlatılan ortak isterleri analiz etmiş ve kullanıcıya ilk etapta bu ortak isterleri sunup, kullanıcı isteğine göre farklı ilgi alanlarına göre genişletilebilecek bir ürün sağlama planını ortaya koymuştur. Bu doğrultuda, şirket içi Ar-Ge (Araştırma-Geliştirme) ve Ür-Ge (Ürün-Geliştirme) faaliyetleri kapsamında RAHAT (Rafta Hazır Ticari) ürün olarak piyasaya sürülebilecek, "DOOB – Defence Out Of Box" ismi verilen, bir KKBS geliştirilmiştir.

Bu makalede, DOOB ürünü kapsamında gerçekleştirilen yazılım geliştirme faaliyetleri, ürüne ait yazılım mimarisi, ortaya konan bileşenler, faydalanılan uygulama geliştirme çatıları ile yazılım desenleri anlatılmakta, ayrıca sistemin gerçeklemesine dair kazanılmış olan deneyimler paylaşılmaktadır.

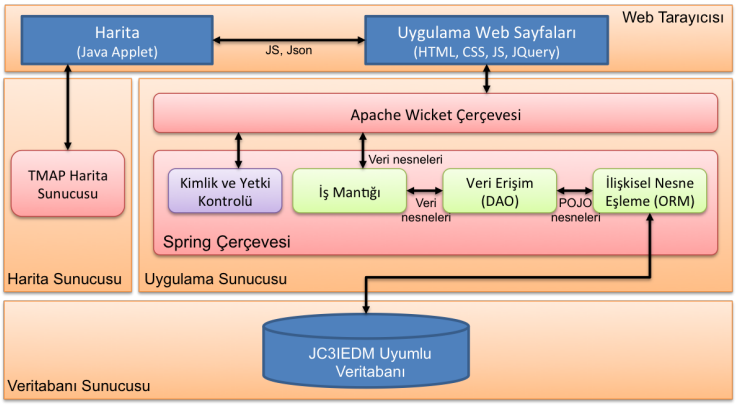
2 Yazılım Geliştirme Süreci

DOOB ürünü, savunma sanayi sektöründeki diğer birçok projeden farklı olarak, herhangi bir müşteri ile sözleşme yapılmadan, RAHAT ürün olarak geliştirilmeye başlanmıştır. Bu durum proje yönetim sürecinin alışılmış süreçlerden farklı olarak yürütülmesini gerektirmiştir.

Ürün geliştirme için Yalın Girişim (İng. The Lean Startup) [5] modeli benimsenmiştir. Bu model, özet olarak, en az bütçe ile ihtiyaçları tam olarak netleşmemiş bir ürünün geliştirilmesi olarak tanımlanmaktadır. Yalın Girişim modelinde, öncelikli olarak (potansiyel) müşterinin ihtiyacını karşılayacak en yalın ve sade ürünü yaparak (İng. Minimum Viable Product, MVP), en kısa sürede müşteriden geri dönüş alınması hedeflenmektedir. Bu açıdan DOOB'un geliştirme koşulları Yalın Girişim modeli ile örtüşmektedir. Zira, DOOB projesi herhangi bir müşteri ihtiyacı olmadan başlamış, ihtiyaçlar şirketin geçmiş tecrübeleri kullanılarak belirlenmiştir. Ayrıca en kısa sürede ürün ortaya konup, potansiyel müşterilere bu ürün tanıtılarak geri dönüşleri alınması hedeflenmiştir.

DOOB ürünü isterleri HAVELSAN'ın geçmiş KKBS uygulamaları tecrübeleriyle, birçok farklı askeri ilgi alanının ortak isterleri analiz edilerek belirlenmiş, ve bu ihtiyaçlar DOOB için MVP olarak tanımlanmıştır. Belirlenen bu ihtiyaçlarla ortaya çıkan ürünün, ilk etapta erken müşterilere (İng. early adaptors) sunulması, daha sonra ise kullanıcı isteklerine göre genişletilmesi hedeflenmiştir. Bu bakış açısı doğrultusunda DOOB uygulaması bir RAHAT ürün ailesinin ilk üyesi olarak geliştirilmiştir. Orta ve uzun vadede, DOOB ürününün kullanıcı isterlerine göre özelleştirilerek geniş bir ürün ailesi elde edilmesi amaçlanmıştır.

DOOB ürün isterleri, iterasyon sürelerini ve sayısını doğrudan etkilediğinden, geliştirme sürecinin en önemli aşaması ortak KKBS isterlerinin belirlenmesi olmuştur. İsterlerin belirlenmesi aşamasında ise en önemli kaynağın, uygulama genelinde ve diğer sistemlerle entegrasyon kapsamında paylaşılacak veri olduğu değerlendirilerek, HAVELSAN'ın da geliştirme sürecine dahil olduğu, birçok ülke tarafından kullanılan ve NATO tarafından standart bir veri değişim modeli olarak kabul edilen JC3IEDM [4] modelinin ortak isterleri belirleme konusunda



Şekil 1. DOOB Yazılım Mimarisi

önemli bir kaynak olduğu kararına varılmıştır. Bu sebeple, DOOB ürününün isterleri JC3IEDM temel alınarak belirlenmiştir. Buna ek olarak HAVELSAN kapsamında gerçekleştirilen ve farklı ilgi alanlarına yönelik projeler incelenmiş ve tüm bu projelerin ortak noktaları analiz edilmiştir.

Yalın Girişim modeline paralel olacak şekilde, yazılım geliştirme süreci olarak Artırımsal Modelin uyarlanmış bir hali olan Döngüsel Model kullanılması tercih edilmiştir. Bu modele uygun şekilde her döngü içinde gerçeklenecek hedefler belirlenerek bu hedeflere uygun gereksinimler belirlenmekte, belirlenen gereksinimlere göre tasarım yapılmakta, yapılan tasarıma uygun olarak gerçekleme yapıldıktan sonra doğrulama sürecine girilmektedir. Her döngü sonucunda bir uygulamanın çalışan yeni bir sürümü çıkarılmakta ve potansiyel kullanıcılar ile paylaşılabilmektedir. Takip eden döngüler eski döngüde elde edilen sürüme yeni fonksiyonaliteler kazandırılarak devam etmektedir.

3 Yazılım Mimarisi

DOOB ürünü, güncel eğilimler ve talepler de dikkate alınarak, bir web uygulaması olarak tasarlanmıştır. Ortaya konan yazılıma ait mimari Şekil 1'de verilmiştir. Yazılımın mimarisine dair alınan kararlar, deneyimler karşılaşılan problemlerle beraber aşağıda belirtilmiştir.

- HAVELSAN'ın geçmiş projelerdeki deneyimleri dolayısıyla, kodun yeniden kullanımı ve ekibin etkinliği açısından programlama dili olarak *Java* tercih edilmiştir. Böylece geliştirme süreci daha hızlı başlamış, ilerlemiş, ve proje üyelerinin değişikliği durumlarındaki riskler minimize edilmiştir.
- Daha önce geliştirilen projelerdeki operasyonel veritabanı (VT) *JC3IEDM* veri modeli üzerinde geliştirildiğinden yine bu veri modeli tercih edilmiştir. Böylece bu veri modelini kullanan NATO kapsamındaki diğer ülkeler ile beraber çalışabilirlik kabiliyeti sağlanmıştır.
- Geliştirme sürelerinin artmasına sebep olan önemli nedenlerden birisi, kullanılan nesneler ile ilişkisel veritabanlarındaki gösterimleri arasındaki farklılıklardır. Bu soruna getirilen en kullanışlı çözümlerden biri *İlişkisel Nesne*

Eşleme (İNE, Object/Relational Mapping, ORM) çözümüdür. Bu yöntem sayesinde nesnenin model ve ilişkisel veri gösterimleri arasında iki yönlü dönüşüm mümkün olmaktadır. DOOB sisteminde, *Hibernate*[2] çerçevesinin kullanımı nesnelerin VT tablolarına eşlenmesini kolaylaştırdığı gibi verilerin kaydedilmesi, güncellenmesi ve silinmesi gibi işlemlerin *SQL* sorgularına ihtiyaç duymadan kolayca gerçekleştirilmesini sağlamıştır.
- Yazılımda, VT tablolarına karşılık *Plain Old Java Objects (POJO)* nesneleri kullanılmaktadır. *POJO*'lar, VT katmanı ile iletişim kurmakla görevli olan Veri Erişim Katmanı (VEK) tarafından kullanılmaktadır. VEK, iş süreçlerinin yönetildiği iş mantığı sınıflarına ilgili veri nesnelerini sağlamakla sorumludur.Veri erişimi ve yönetimi organize ve katmanlı bir şekilde yapıldığından, uygulama geliştirilirken hatalar en aza indirilmekte, problemler kolayca çözümlenebilmekte ve kod tekrarlarından kaçınılmaktadır.
- İş mantığı katmanında geliştirilecek sınıflardaki bağımlılıkları azaltmak ve geliştirme sürecini kolaylaştırmak için *Spring*[3] çerçevesinin kullanımı tercih edilmiştir. *Spring*'in önemli özelliklerinden olan ve kodun birbirine bağımlılığını önleyen *Dependency Injection (DI)* ve modülerliği destekleyen *Aspect Oriented Programming (AOP)* özelliklerinden gerektiğince faydalanılmıştır. Bunun yanında kullanıcı doğrulaması ve yetkilendirilmesi gereksinimlerinin karşılamasında da *Spring - Güvenlik* çerçevesi kullanılmıştır.
- Arayüz tasarımında *Apache Wicket*[1] çerçevesi kullanılarak sunum katmanı oluşturulmuştur. *Wicket* çerçevesi, arayüz geliştirilirken *HTML* (görsel tasarım) ve *Java* (gerçekleştirim) kısımlarının ayrılmasını sağlayarak daha okunabilir ve kolay anlaşılabilir arayüz sınıfları geliştirilmesine olanak sağlamıştır. *Wicket*; kolay tasarlanabilir ve geliştirilebilir olması sebebiyle arayüz geliştirilme sürecini kısaltmıştır. Ayrıca *CSS (Cascading Style Sheets)* dosyaları kullanılarak özelleştirilebilir görsel tasarımlar gerçekleştirilebilmektedir.
- Coğrafi Bilgi Sistemi (CBS) bileşeni olarak, HAVELSAN'ın mevcut bir ürünü *TMAP* kullanılmıştır. *TMAP*, *Java* ile geliştirilen bir ürün olduğundan, web uygulamasına *applet* olarak dahil edilmiştir. Bu durum, *applet* ve web uygulaması arasında, DOOB nesnelerinin *JSON* formatına dönüştürülerek *Javascript* çağrıları aracılığıyla çalışan bir mekanizma kurulmasını gerektirmiştir. Her ne kadar web tabanlı bir CBS ürünü kullanılması geliştirme süresini kısaltabilecek olsa da, HAVELSAN'ın mevcut bir ürününü kullanmanın maliyetsiz olması ve olası hatalarda çözümün daha kolay sağlanabilecek olması, *TMAP* uygulamasının tercih edilmesini sağlamıştır.

4 Tasarım Örüntüleri

Projenin geliştirme sürecinde bir çok tasarım örüntüsünden faydalanılmıştır. Kullanılan tasarım örüntüleri, kullanım alanları, sağladığı kazanımlar, pratikte karşılaşılan sorunlar ve çözümleri üç ana başlık altında aşağıdaki gibi incelenmiştir.

4.1 Yaratım Örüntüleri

Yegane (Singleton) Tasarım Örüntüsü. Bu örüntü, uygulama genelinde tek olması gereken günlük tutma (İng. logging) ve oturum (session) nesnelerinin

yaratılması ve ulaşılması için kullanılmıştır. Böylece, bu nesnelerin çok sayıda yaratılması engellenerek tekilliği garanti edilmiş, ve hafıza kullanımı açısında etkili bir çözüm sağlanmıştır. Ayrıca, günlük ve oturum için kullanılan global sınıflara ulaşım kolaylaştırılmıştır.
Sorunlar: Zaman içerisinde, kolay ulaşılabilir olması ve dikkatsiz kodlama sonucunda oturum nesnesi içinde gereksiz veriler tutulmaya başlanmış, bu durum oturum nesnesinin boyutunun büyümesine sebep olmuştur. Gözden geçirmeler sonucunda tespit edilen gereksiz veri kullanımları temizlenmiştir.

Soyut Fabrika (Abstract Factory) Tasarım Örüntüsü. Kullanılan *POJO* ve veri nesneleri derin bir hiyerarşik yapıda ve nesne tipi çeşitliliği de fazla olduğundan, POJO ve veri nesnelerin yaratılması işlemlerini kolaylaştırmak için Soyut Fabrika tasarım örüntüsü kullanılmıştır. Böylece, yaratım kodlarının iş mantığı sınıfları içinden ayrılıp soyutlanması ve yazılım bileşenleri arasındaki bağımlılığın azaltılması sağlanmıştır.
Sorunlar: Yaratılacak gerçek sınıflara karar verilmesi aşamasında kullanılan yapının (*if/else*) büyüklüğü kodu karmaşık hale getirdiği görülmüş, çözüm olarak da *Java* yansıma (İng. reflection) kütüphanesi kullanılarak sınıf tipine göre uygun gerçek sınıflar yaratılmasını sağlayacak bir altyapı hazırlanmıştır.

Yapıcı (Builder) Tasarım Örüntüsü. Askeri formatlı mesajların giriş ve görüntülenmesi için gerekli ekranların ve bu ekranlar içindeki panellerin sayısı çok fazla olduğundan, bunların otomatik olarak yaratılabilmesi için Yapıcı tasarım örüntüsü kullanılmıştır. Böylece, değişik panel tiplerinin ortak bir yapı kullanması sağlanmış, formatlı mesajlar için tanımlanmış *xsd* şemaları kullanılarak panellerin kendilerini yaratıp bağlı olduğu üst panele eklediği otomatik bir ekran oluşturma altyapısı kurgulanmıştır. Bu örüntü kullanılarak, yüzlerce ekran ve binlerce panelin elle gerçekleştirilmesi maliyetinden kurtulunmuştur.

4.2 Yapısal Örüntüler

Veri Erişim Nesnesi (Data Access Object) Tasarım Örüntüsü. POJO sınıflarının ve alt seviye VT sorgulama işlemlerinin, üst seviye iş mantığı sınıflarından ayrıştırılabilmesi için bu tasarım örüntüsü kullanılmıştır. Böylece, POJO nesneleri, veri erişim sınıfları, veri nesneleri ve iş mantığı sınıfları arasında bağımlılıklar azaltılmış ve düzenli bir hale getirilmiştir.
Sorunlar: POJO'lar, iş mantığı sınıflarından ayrıştırıldığından, iş mantığı içerisinde POJO'ların veri sınıflarına dönüştürülmesi ihtiyacı oluşmuş, bu amaçla uyumlayıcı (İng. adapter) sınıfları oluşturulmuştur.

Uyumlayıcı (Adapter) Tasarım Örüntüsü. POJO nesneleri ve iş mantığı veri yapısı arasındaki dönüştürme işlemleri için Uyumlayıcı tasarım örüntüsü kullanılmıştır. Böylece, veri erişim (İng. DAO) sınıfları ile iş mantığı sınıfları birbirinden soyutlanmıştır.

Cephe (Façade) Tasarım Örüntüsü. *Web Applet* olarak geliştirilen CBS uygulaması ile DOOB ekran sınıfları arasındaki *Javascript* çağrılarının tek bir sınıfta toplanması için Cephe tasarım örüntüsünden faydalanılmıştır. Böylece,

DOOB ekran sınıflarından CBS uygulamasına yapılacak çağrılar tek bir sınıf üzerinden yapılarak, kullanılan CBS altyapısı DOOB tarafından soyutlanmıştır.

4.3 Davranış Örüntüleri

Ziyaretçi (Visitor) Tasarım Örüntüsü. Veri yapısı olarak kullanılan model (JC3IEDM) karmaşık ve derin hiyerarşik yapılar içerdiğinden; üretilen POJO sınıfları da aynı derin hiyerarşi ve karmaşıklığa sahip olmuştur. *Hibernate* ile yapılan sorgulama işlemlerinde, sorgu sonucu elde edilen ata sınıf tipine sahip POJO nesnelerinin uygun şekilde kullanılabilmesi için sınıf tipi (*instanceOf*) kontrolü ihtiyacı ortaya çıkmıştır. Buna ek olarak *Hibernate* çerçevesinin sağladığı geç yükleme (İng. lazy-loading) kullanıldığı durumlarda, *Hibernate* vekil (İng. proxy) nesneler döndüğünden, çoğu zaman sınıf tipi kontrolü dahi yapılamamaktadır. Bu problemlerden kurtulabilmek için Ziyaretçi örüntüsü kullanılmıştır. Böylece, geç yükleme sonucunda dönen vekil nesnelerden gerçek POJO nesnelerine ulaşım gereği ortadan kaldırılabilmiş, ayrıca sınıf tipi kontrollerinin ortadan kalkması sonucu kolay okunabilir temiz bir kod elde edilmiştir.
Sorunlar: Ziyeretçi metodu içindeki gerçekleştirimin kapsamının iyi belirlenmesi ve doğru tasarlanmaması sonucunda modüller arası gereksiz bağımlılıkların ortaya çıktığı gözlemlenmiştir.

5 Sonuç

Bu makalede, HAVELSAN tarafından geliştirilen, farklı ilgi alanlarina sahip askeri organizasyonlar tarafından kullanılabilecek, temel KKBS işlevlerine sahip bir RAHAT ürün olan DOOB yazılımı anlatılmıştır. DOOB, Yalın Girişim ve MVP stratejisiyle geliştirilmiş, yazılım geliştirme süreci olarak Artırımsal Model'in uyarlanmış bir hali olan Döngüsel Model uygulanmıştır. Yazılım, Java programlama dili ile, Apache Wicket, Spring ve Hibernate çerçeveleri kullanılarak gerçekleştirilmiştir. Yazılım geliştirme esnasında yaratım, yapısal, davranış deseni olarak birçok desen uygulanmıştır. Halihazırda DOOB ürününün 4. versiyonu için çalışmalar devam etmekte, proje başarılı şekilde sürdürülmektedir.

Teşekkür

Bu makale HAVELSAN A.Ş. tarafından şirket içi Ar-Ge projesi olarak yürütülen DOOB projesi kapsamında yapılan çalışmaların sonucu olarak üretilmiştir. Yazarlar, DOOB projesinin tüm geçmiş ve şu anki çalışanlarına değerli katkıları dolayısıyla teşekkürlerini sunmaktadır.

Kaynaklar

1. Apache wicket. `https://wicket.apache.org`, accessed: 2015-05-01
2. Hibernate orm. `http://hibernate.org/orm/`, accessed: 2015-05-01
3. Spring. `http://spring.io`, accessed: 2015-05-01
4. Multilateral Interoperability Programme, Greeding, Germany: The Joint C3 Information Exhange Data Model (JC3IEDM), ver.3.1.4 edn. (February 2012)
5. Ries, E.: The lean startup : how today's entrepreneurs use continuous innovation to create radically successful businesses. Crown Business, New York (2011)