# JAR Kütüphanelerini Bileşen Olarak Kullanan Bir CBS Alan Çerçevesi

**Ebru ÖZDOĞRU**[1]

[1] MST Grubu, Yazılım Mühendisliği Müdürlüğü Bölümü, ASELSAN A.Ş., Ankara

[1] e-posta: eozdogru@mst.aselsan.com.tr

## Özet

Bir Bileşen Yönelimli Yazılım Mühendisliği (BYYM) modelleme ortamına, çalıştırılabilir bileşenleri dışarıdan aktarma ve bu bileşenlerin birleştirilmesi yolu ile uygulama geliştirilme yeteneği eklendi. Bu amaçla, JAR kütüphanelerini bileşen olarak kullanmayı sağlayan bir arayüz katmanı geliştirildi. Ayrıca, Coğrafi Bilgi Sistemleri (CBS) alanı üzerinde Alan Mühendisliği süreci uygulandı. Böylelikle modelleme ortamından bir geliştirme çerçevesi olarak yararlanıldı. Arayüz katmanı, JAR kütüphanelerinin BYYM yaklaşımı ve Bileşen Yönelimli Yazılım Mühendisliği Modelleme Dili (COSEML)'ni destekleyen grafiksel bir editor olan COSECASE aracı içerisinden kullanılmasını sağlamaktadır. Sonuçta sistemler, soyut bileşenler ile tasarlanır ve sonrasında soyutlamalara karşılık gelen var olan bileşen kodları ile gerçekleştirilir. Bileşen kodları, COSECASE'te bu arayüz katmanı sayesinde kullanılabilir. Ek olarak, Alan Mühendisliği sürecinin, Alan Analizi, Alan Tasarımı ve Alan Uygulaması evreleri CBS alanına uygulandı. Alan Uygulaması evresinde geliştirilmiş bileşenler kullanılmak üzere COSECASE'e yüklendi. Gösterim amaçlı, basit bir CBS uygulaması, COSECASE'in arayüz katmanı kullanılarak tasarlandı ve kod üretildi.

## 1. Giriş

Bu çalışmada, mevcut yazılım bileşenlerini birleştirerek uygulama geliştirme üzerine çalışma yapılmıştır. Çalışmanın odak noktası, JAR kütüphaneleri biçiminde var olan bileşen kodlarını, bileşenlerin bağlanmaları ile bir araya getirilerek Bileşen Yönelimli Yazılım Mühendisliği sürecine uygun olarak uygulama geliştirme ve kodların kullanılması ile otomatik uygulama üretme yeteneklerinin kullanılmasını sağlayan bir arayüz katmanının geliştirilmesidir. Bunun yanı sıra, çalışma yeniden kullanılabilirlik, alan mühendisliği sürecinin kullanılmasının da önemini ortaya koymaktadır. Geliştirilen arayüz katmanı Bileşen Yönelimli Yazılım Mühendisliği sürecine uygun sistem geliştirmeyi destekleyen COSECASE aracı için gerçekleştirilmiştir [8]. Yeniden kullanılabilirliğin az çaba ve hızlı bir şekilde yazılım geliştirmedeki rolünün artması ile bileşenler ve özelleşmiş bileşenlerin özelleşmiş bir alan kullanımını destekleyen Alan Mühendisliği süreci popüler hale gelmiştir. Savunma sektöründe, coğrafi verinin çeşitli şekillerde sunulup işlendiği Coğrafi Bilgi Sistemleri (CBS) alanında sistem geliştirmek için sürekli tekrar eden çabaların fark edilmesi, CBS alanında Alan Mühendisliği sürecinin bir ara adım olarak uygulanmasının faydalı olacağı görüşüne gerekçe olmuştur. Bu ara çalışma, alan modellemesi ve bileşenlerin geliştirilmesi ile asıl çalışmaya işlevsel bir ortam oluşturmuştur.

Bu makale şu şekilde organize edilmiştir: 2. bölüm, yazılımların yeniden kullanılabilirliği ve bileşenler, bileşen-tabanlı yaklaşım, alan mühendisliği ve BYYM Modelleme Dili olan COSEML ve COSECASE aracı hakkında gerekli ön bilgi vermektedir [8]. 3. bölüm, CBS alanında uygulanan Alan Mühendisliği sürecini anlatmaktadır. 4. bölümde, COSECASE için geliştirilmiş olan arayüz

katmanı sunulmakta ve bu arayüz katmanının uygulama mühendisliği sürecinde kullanımı açıklanmaktadır.

## 2. Yazılımlarda Yeniden Kullanılabilirlik

Yeniden kullanılabilirlik, yazılım bileşenlerinin yeni uygulamalar geliştirilirken kullanılması ile gerçekleştirilir [4, 7]. Bu yaklaşım, geliştirme zamanını, maliyetini ve iş gücünü azaltmaktadır ve daha kaliteli yazılım geliştirilmesini sağlamaktadır [5]. Bir yazılım bileşeni, iyi tanımlanmış bir mimari içinde işlevselliği olan standart, yeniden kullanılabilir ve önceden geliştirilmiş, değiştirilebilir birimdir [7, 9]. Programcılar yazılım bileşenlerini farklı uygulamaları yaratmak için bağlayarak bir araya getirirler [6, 11, 12, 10]. Bileşenler, nesne yönelimli yaklaşımdaki sınıflar gibi verileri dış dünyadan saklarlar, bir protokole uyum sağlarlar ve birbirleriyle ara yüzlerini kullanarak iletişim kurarlar.

Var olan yazılım bileşenlerinin planlı bir şekilde entegre edilmesi ile yazılım geliştirme sürecine Bileşen Yönelimli Geliştirme (BYG), Bileşen Yönelimli Yazılım Mühendisliği (BYYM) denir [13]. BYYM sürecinde iki yaklaşım vardır. Yukarıdan aşağıya (top-down) yaklaşımda, müşteri gereksinimleri dikkate alınır. Sistem, detay seviyesi var olan bileşenler ile geliştirme yapılmasına uygun oluncaya kadar küçük parçalara bölümlenir. Diğer yandan, aşağıdan yukarıya (bottom-up) yaklaşımda, var olan bileşenler kullanıcı gereksinimlerini karşılamak üzere daha büyük parçalar oluşturulmak için birleştirilir [14].

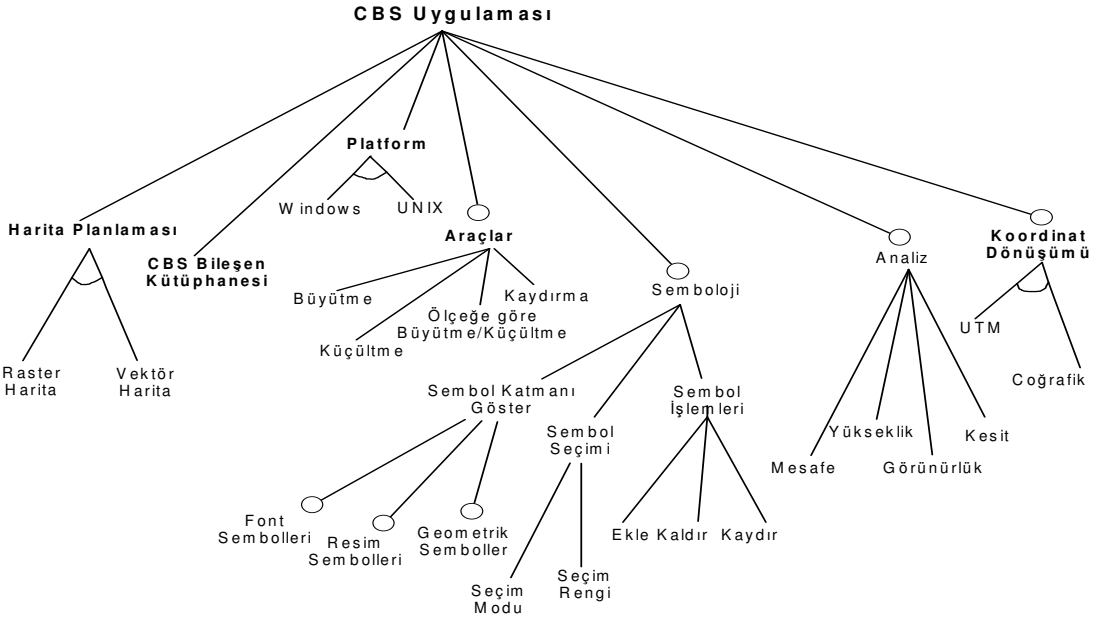
COSE Modelleme Dili (COSE Modeling Language - COSEML), Bileşen Yönelimli Yazılım Mühendisliğine uygun sistem geliştirmek için, tasarlanmıştır ve uygulanabilmesi için COSECASE isminde bir araç geliştirilmiştir [8]. COSEML ile sistem, yukarıdan aşağıya yaklaşımla iki seviyede tasarlanır; soyut ve uygulama. COSECASE'de bileşen simgesi var olan bileşenleri simgeler. Her bir bileşen bileşenin özelliklerini, metotlarını ve olaylarını sunan arayüzlere sahiptir. Arayüz bir bileşenin bağlantı noktasıdır ve Bileşenin servislerini sunar. COSECASE kullanılarak bileşen ve arayüz şekilleri çizim paneli üzerine yerleştirilerek sistem tasarlanır.

Yeniden kullanılabilirlik şirketler içinde kurulan bir organizasyon ile yazılım geliştirme kavramında tek projeden çok projeye genişletilir. Sistemler arasında benzer fonksiyonelliği içeren fonksiyon setleri *alan* olarak adlandırılır. Bir alan içerisinde sistemler pek çok gereksinimi paylaşırlar. *Alan Mühendisliği*, yazılım mühendisliğinin benzer yetenekler ve gereksinimler içeren yazılım sistem aileleri için genel çözümlerde yoğunlaşan parçasıdır [15]. Alan mühendisliği üç alt işlemden oluşmaktadır: Alan *Analizi*, Alan *Tasarımı* ve Ala*n Gerçekleştirmesi*. Alan mühendisliği sonucunda elde edilen ürünler yazılım birimlerinden yazılım ürünleri geliştiren *Uygulama Mühendisliği* sürecinde kullanılır. *Alan Analizi* aktiviteleri, alan seçimi ve alan içeriği, özellikleri ve çözümlerini içerir ve çıktısı, yeniden kullanılabilir bir alan modelidir [16, 21, 17, 18]. *Alan Tasarımı* aktiviteleri, alandaki sistemler için bileşenlerin yeniden kullanılabilirliğini destekleyen genel bir mimariyi modeller [21]. *Alan Gerçekleştirmesi* aktivitelerinde, alan için yeniden kullanılabilir bileşenler gerçekleştirilir [19, 20, 21].

## 3. Coğrafi Bilgi Sistemleri (CBS) Alan Çerçevesi

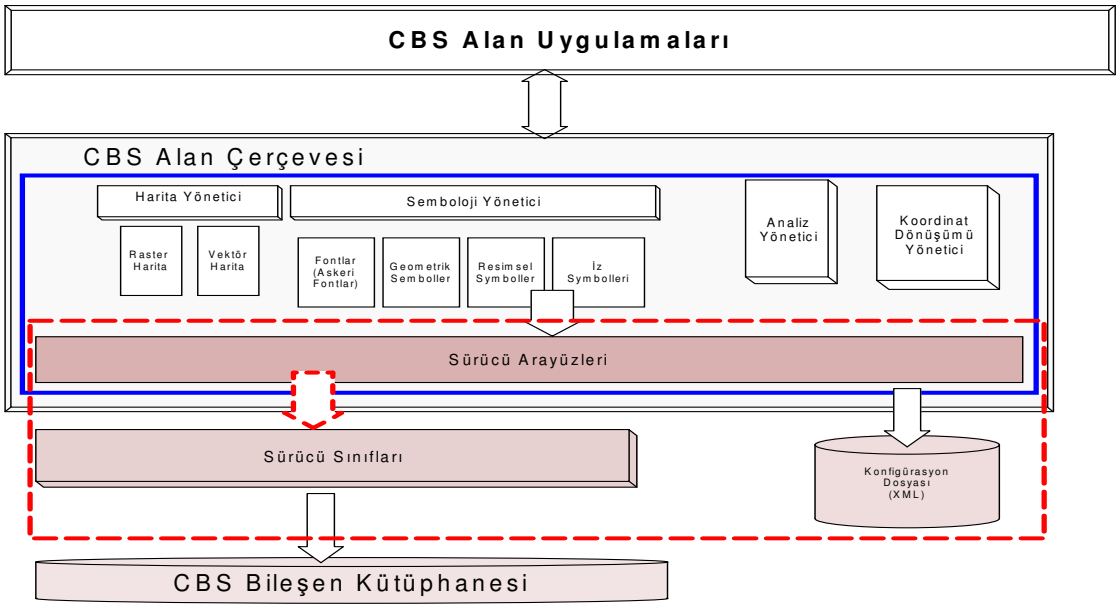
Alan Mühendisliği süreci, coğrafi verilerin işlendiği Coğrafi Bilgi Sistemleri (CBS) alanına uygulandı. Öncelikle, CBS alanındaki projelerden gereksinimler ve var olan alan bilgisi toparlandı.

CBS alan çerçevesi, bir organizasyonda projelerin içerdiği CBS uygulamalara hizmet edecektir. Çerçevenin verdiği servisler bu uygulamaların gereksinimlerini karşılayacağından CBS uygulamaları incelenerek gereksinimler belirlenmiştir. Alan analizi yapılarak, bir alan modelleme metodolojisi olan Feature Oriented Domain Analysis (FODA) alana uygulandı ve alan modeli oluşturuldu [18]. FODA'da her bir özellik alandaki bir gereksinime karşılık gelmektedir. FODA'da düz çizgiler zorunlu gereksinimleri, ucu çemberli çizgiler seçmeli gereksinimleri, diğer çizgiler de alandaki uygulamalar için alternatif gereksinimleri sembolize etmektedir. Şekil 1, CBS alanının alan analizi sonucunda modellenen gereksinimleri gösterilmektedir:




**Şekil 1.** CBS Alanı için Özellik Modeli

Alan tasarımı fazında, CBS çerçevesinin mimari tasarımı, alan analizi modelinden faydalanılarak gerçekleştirildi. Alan analizi fazında, ortaya çıkan gereksinimlerden biri piyasadan temin edilen bir CBS kütüphanesinin çerçeve tarafından kullanılabilmesi ve çerçevenin yeniden derlenmesine gerek kalmadan değiştirilebilmesi idi. CBS alan çerçevesi bu gereksinim göz önüne alınarak tasarlandı. Bunun için bir "*driver*" sınıfı ve XML tabanlı bir konfigürasyon dosyasının kullanımı öngörüldü. "*driver*" sınıfı CBS çerçevesi tarafından sunulan bir arayüzü CBS kütüphanesini kullanarak gerçekleştirmektedir. Konfigürasyon dosyasından ise bu sınıfın ismi, çerçeve içerisindeki bileşenler tarafından okunarak CBS kütüphanesinin kullanımı sağlanmaktadır. Çerçeve içindeki bileşenler her zaman "driver" arayüzleri metotlarını kullanmaktadır. Böylece, kullanılan CBS kütüphanesinin değiştirilmesi CBS çerçevesini etkilememektedir. Şekil 2'de CBS Alan çerçevesi tasarımı gösterilmektedir:




**Şekil 2.** CBS Alan Çerçevesi tasarımı

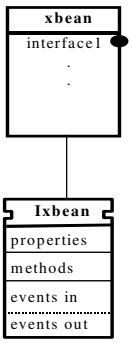
Alan gerçekleştirmesi fazında, bileşenler Java Bean olarak gerçekleştirildi ve çerçeveye koyuldu. Bu bileşenleri oluşturmak için, öncelikle projelerde var olan kodlar bir araya getirilmeye çalışıldı. Gereksinimin evdeki projelerden karşılanamadığı durumlarda bileşenler baştan yazıldı. Yazılan bu Java Bean'ler JAR arşiv kütüphaneleri olarak paketlendi.

Elde edilen bileşenler Uygulama Mühendisliği sürecinde uygulama geliştirmek için kullanıldı. Uygulama geliştirmek için bileşenler COSECASE'e aktarıldı. Bu işlemi sağlayabilmek için, COSECASE aracına bir arayüz katmanı geliştirildi. Bir sonraki bölümde bu arayüz katmanı anlatılmaktadır.

## 4. JAR Kütüphanelerini Bileşen Olarak Kullanan Arayüz Katmanı

Mevcut çalıştırılabilir bileşenlerin, Bileşen Yönelimli Yazılım Mühendisliği sürecinde sistem tasarımı sırasında kullanılabilmesi, otomatik uygulama geliştirme açısından önem içermektedir. Mevcut çalıştırılabilir bileşenlerin COSECASE'te sistem tasarımı yapılırken kullanılması, COSECASE'i daha fonksiyonel yapmaktadır. Bu bileşenlerin COSECASE'e aktarılması; bileşenlerin özelliklerinin, metotlarının ve olaylarının çalışma alanına taşınması gerekmektedir. Bu işlemleri gerçekleştirmek için, Java Bean'ler içeren JAR kütüphanelerini araca aktarmayı sağlayan bir arayüz katmanı geliştirilmiştir. Bu arayüz katmanı kullanılarak, JAR kütüphaneleri COSECASE'te bileşen olarak kullanılabilmektedir. Bu çalıştırılabilir bileşenler, BYYM (COSE) yaklaşımı ile sistem tasarlamada kullanılır. Bileşenlerin bağlantıları araçta yarı-otomatik olarak yapılarak, araçtan yarı-otomatik uygulama kodu üretilmesi sağlanabilmektedir. Çalışmada, işlevsel ortam olarak önceki bölümde Alan Mühendisliği süreci kapsamında gösterim amaçlı hazırlanan bileşenler kullanılmış ve COSECASE'e aktarılarak sistem tasarlanmıştır. COSECASE'e JAR kütüphanelerini aktarmak için bir menü seçeneği hazırlanmıştır. Buradan araca aktarılan kütüphaneler içindeki bean-info yapıları okunarak çalıştırılabilir bileşen ikonları araç çubuğu üzerine yerleştirilir. Araç çubuğu üzerindeki bu simgeler simgelemektedir. Simgeler seçilip çizim paneli üzerinde

yerleştirilerek COSECASE'te çalıştırılabilir bileşenler ile sistem tasarlanır. Çizim paneli üzerine yerleştirilen çalıştırılabilir bileşenler COSECASE'in bileşen sembollerine uygun olarak gösterilirler. Bean-info yapısından alınan bileşen özellikleri, metotları, olaylar (events in/out) otomatik olarak bileşen sembolleri içinde gösterilirler. Çalıştırılabilir bileşenler bir bileşen ve bir de ona ait arayüz (interface) ile gösterilirler. Bileşenlere ait okunan özellik, metot ve olaylar arayüz simgesi üzerinde gösterilir. Şekil 3'te bu yapı gösterilmiştir.

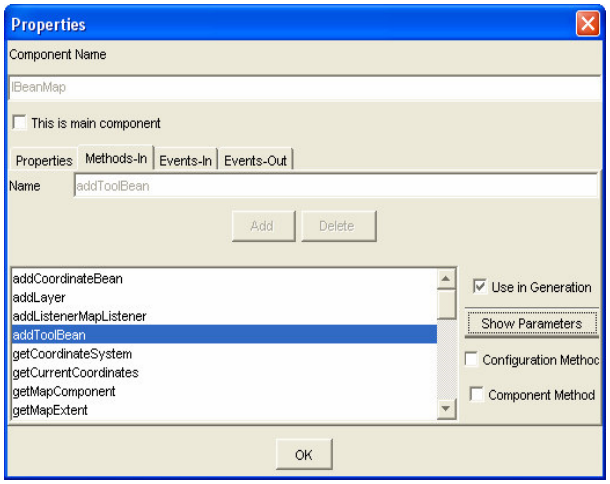



**Şekil 3.** COSECASE'te JAR kütüphanelerinden aktarılan bileşenlerin gösterimi

Java Bean bean-info yapısı bileşene ait özellikler, metotlar, metotların parametreleri, simge ve bean tarafından tetiklenen olaylara ait bilgi içermektedir. JAR kütüphanesi yüklenirken, bean-info yapısı çalışma zamanında dinamik olarak okunarak yüklenir.

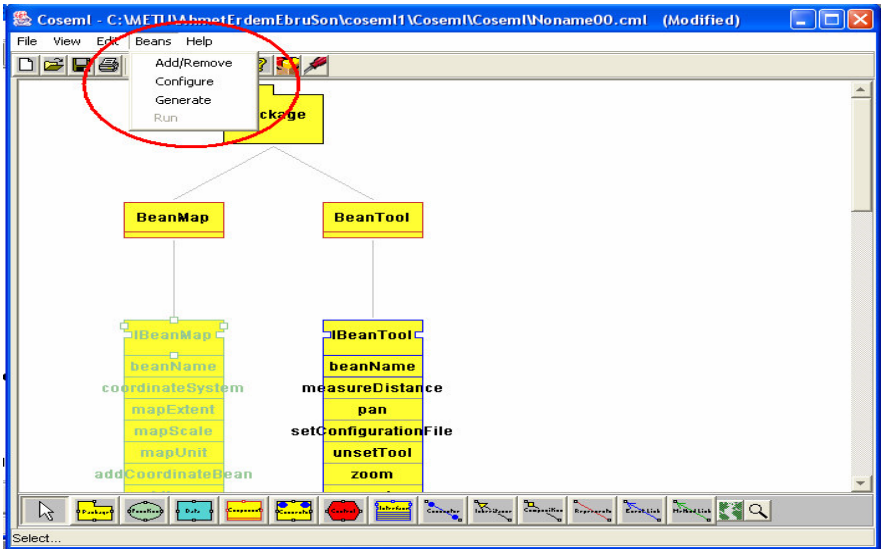
COSEML ve COSECASE'te bileşen iki tip olaya (event) içermektedir: Bileşenin aldığı olaylar (events in) ve bileşenin tetiklediği olaylar (events out). Bean-info yapısında yalnızca bileşen tarafından tetiklenen olaylara ait bilgi bulunmaktadır. Bileşenin kabul ettiği olayları bulmak için arayüz katmanı bir bileşenin gerçekleştirdiği tüm dinleyici (listener) tiplerini araştırır ve bu tiplerin metotlarını bulur: Dinleyicilerin "java.util.EventListener" arayüzünü gerçekleştirdiği varsayılmaktadır. Arayüz katmanında bir bileşenin tetiklediği olayları bulmak için, bean-info yapısı kullanılmamıştır. Arayüz katmanında bu olayların bulunması için Java'nın isimlendirme uyumundan faydalanılmıştır. Java'da bir bileşenin tetiklediği olayları dinlemek isteyen bileşen dinleyici olarak (listener) bileşene kendini tanıtır. XYZ dinleyicisini bileşene tanıtmak için addXYZListener(XYZ) metodu kullanılmaktadır. Arayüz katmanı, bileşendeki tüm add<ListenerType>Listener metotlarını araştırır ve dinleyici tiplerinin metotlarını tetiklenen olaylar olarak kaydeder.

COSECASE'te çizilen bileşenlere ait özelliklerin gösterildiği panelin arayüz katmanı ile de kullanılabilmesi sağlanmıştır. Kod üretiminin yapılabilmesi için otomatik olarak getirilen metotlar bu panel üzerinden seçilebilmektedir. Bir önceki bölümde tasarımı yapılan alan çerçevesine uygun uygulamalara ait kod üretiminin gerçekleştirilebilmesi için ise konfigürasyon dosyasının bileşenlerde ayarının yapılabilmesi için bir seçenek yerleştirilmiştir. Şekil 4'da bu panel gösterilmektedir. Metotun aldığı parametrenin girilmesi açılan bir diğer panel yardımı ile sağlanmaktadır.




**Şekil 4.** COSECASE'te özellikler paneli

İşaretlenen bileşenlere ve metotlara ait kodlar, arayüz katmanı tarafından bir Java uygulama dosyasına yazdırılmaktadır. Arayüz katmanı tarafından hazırlanan bir bat dosyası, menüden girilen bileşen kütüphanesine ait konfigürasyon ayarlarının kullanılmasını ve uygulamanın araç dışından çalıştırılmasını sağlamaktadır. Araç içerisinden uygulamanın çalıştırılmasını sağlayan bir seçenek de bulunmaktadır. Şekil 5, bir önceki bölümde hazırlana JAR kütüphanelerinin kullanılması ile tasarlanmış bir örnek çalışmayı göstermektedir:




**Şekil 5.** COSECASE'te arayüz katmanını kullanarak tasarlanan bir örnek uygulama

## 5. Sonuç

Çalışma henüz sektörde gerçek anlamda denenmemiştir. Ancak, üç küçük örnek uygulama geliştirilerek yapılan karşılaştırılmalar sonucunda, uygulamaların arayüz katmanının kullanılarak geliştirilmesinin, elle geliştirmeye oranla 2 kez daha hızlı olduğu gözlemlenmiştir. Geliştirilmiş araç seti ile yeni CBS uygulamaları bu alan mühendisliği çerçevesinden hızlıca geliştirilebilir.

Çalışmada COSECASE'e aktarılan bileşenler için bir bileşen ve bir arayüz katmanı gösterilmektedir. Ancak, COSEML'de bir bileşen birden fazla arayüz içerebilmektedir. Arayüz katmanına, bileşene ait arayüzü bazı kurallara göre ayırarak birkaç arayüz sağlayabilen akıllı bir mekanizma geliştirilebilir. Bu çalışmada yarı-otomatik kod üretimi yaptırılmaktadır. Tasarımda kullanılan bileşenler arasındaki ilişkileri otomatik olarak bulan akıllı bir mekanizma otomatik kod üretimi açısından iyi olacaktır.

## Kaynakça


[1]. Ilka Philippow, Detlef Streitferdt, Matthias Riebisch, "Design Pattern Recovery in Architectures for Supporting Product Line Development and Application", *Modelling Variability for Object-Oriented Product Lines* edited by M. Riebisch, J. O. Coplien, D, Streitferdt (Eds.). BookOnDemand Publ. Co., Norderstedt, pp 42-57, 2003.

[2]. D.L. Parnas, P.C. Clements, and D.M. Weiss, "Enhancing Reusability with Information Hiding", *IEEE Tutorial Software Reusability* edited by Peter Freeman, IEEE Computer Society Press, IEEE Catalog #EH0256-8, ISBN 0-8186-0750-5, pp. 83-90, 1987.

[3]. Bertrand Meyer, "Eifell: Reusability and Reliability", *IEEE Tutorial Software Reuse: Emerging Technology* edited by Will Tracz, IEEE Computer Society Press, IEEE Catalog #EH0278-2, ISBN 0-8186-0846-3, pp. 216-228, 1990.

[4]. Clements Szyperski, "Component Software Beyond Object Oriented Programming", *Addison-Wesley*, 1998.

[5]. "Software Reuse, Major Issues Need To Be Resolved Before Benefits Can Be Achieved", *United States General Accounting Office*, January 1993.

[6]. Ruben Prieto Diaz, Gerald A. Jones, "Breathing New Life into Old Software", *IEEE Tutorial Software Reuse: Emerging Technology* edited by Will Tracz, IEEE Computer Society Press, IEEE Catalog #EH0278-2, ISBN 0-8186-0846-3, pp. 152-160, 1990.

[7]. Vedat Bayar, A Process Model for Component Oriented Software Development, *M.S. Thesis*, Middle East Technical University, November 2001.

[8]. Aydın Kara, A Graphical Editor for Component Oriented Modeling, *M.S. Thesis*, Middle East Technical University, April 2001.

[9]. Jim Q. Ning, "A Component Model Proposal", *International Workshop on Component-Based Software Engineering*, pp. 13-16, Los Angeles, CA, USA, 17-18 May 1999.

[10]. Kurt C. Wallnau, "On Software Components and Commercial ("COTS") Software", *International Workshop on Component-Based Software Engineering*, pp. 213-218, Los Angeles, CA, USA, 17-18 May 1999.

[11]. Gail E. Kaiser, David Garlan "Melding Software Systems from Reusable Building Blocks", *IEEE Tutorial Software Reuse: Emerging Technology* edited by Will Tracz, IEEE Computer Society Press, IEEE Catalog #EH0278-2, ISBN 0-8186-0846-3, pp. 267-274, 1990.

[12]. Geral Jones, "Methodology/Environment Support for Reusability", *IEEE Tutorial Software Reuse: Emerging Technology* edited by Will Tracz, IEEE Computer Society Press, IEEE Catalog #EH0278-2, ISBN 0-8186-0846-3, pp. 190-193, 1990.

[13]. Oh-Cheon Kwon, Seok-Jin Yoon and Gyu-Sang Shin, "Component-Based Development Environment: An Integrated Model of Object-Oriented Techniques and Other Technologies", *International Workshop on Component-Based Software Engineering*, pp. 47-53, Los Angeles, CA, USA, 17-18 May 1999.

[14]. D. Ansorge, K. Bergner, B. Deifel, N. Hawlitzky, C. Maier, B. Paech, A. Rausch, M. Sihling, V. Thurner, S. Vogel, "Managing Componentware Development –Software Reuse and the V-Modell Process", *Proceedings of the 11th International Conference on Advanced Information Systems Engineering*, pp 134-148, 14-18 June 1999.

[15]. K. Czarnecki and U. Eisenecker, "Generative Programming: Methods, Techniques, and Applications", *Addison-Wesley* 1999.

[16] Rubén Prieto-Díaz, "DOMAIN ANALYSIS: AN INTRODUCTION", *ACM SIGSOFT Software Engineering Notes*, pp. 47-54, April 1990.

[17] Rubén Prieto-Díaz, "Domain Analysis for Reusability", *Proceedings of COMPSAC'87*, pp. 23-29, 1987.

[18] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, A. Spencer Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study", *Technical Report, Software Engineering Institute*, November 1990.

[19] Rosario Girardi and Carla Gomes de Faria, "A Generic Ontology for the Specification of Domain Models", *Proceedings of 1st Workshop Component Engineering Methodology*, pp. 41-50, 24 September 2003, Erfurt, Germany.

[20] Thomas Cleenewerck, "Component-based DSL Development", *Proceedings of GPCE03 Conference*, Lecture Notes in Computer Science 2830, pp. 245–264, Springer- Verlag, 2003.

[21] Carnegie Mellon University Software Engineering Institute (SEI), "Domain Engineering", *http://www.sei.cmu.edu/domin_engineering/domain_emg.html*, March 2005.