Oyun Kuramı Kullanarak Yazılım Takımlarının Üretkenliğini Artırmak İçin Geliştirilen Bir Yazılım Süreç Mühendisliği Yaklaşımı

Murat Yılmaz[1,2] Rory V. O'Connor[1,2]

[1]Bilgisayar Bilimleri Bölümü, Dublin Şehir Üniversitesi, İrlanda
[2]Lero, İrlanda Yazılım Mühendisliği Araştırmaları Merkezi, İrlanda

[1]e-posta: murat.yilmaz@computing.dcu.ie [2] e-posta: roconnor@computing.dcu.ie

Özetçe

Yazılım geliştirme süreçleri bilgi ve insan tabanlı aktivitelerdir. Sosyal açından değerlendirildiğinde yazılım geliştiren bireylerin sosyal etkileşimi ve işbirliğine yatkınlığı takım ve organizasyon üretkenliğinin artırılmasını destekleyecek en önemli unsurlar arasındadır. Bu sebeple, özellikle son zamanlarda, yazılım mühendisliği konusunda yapılan birçok çalışma yazılım geliştirme faaliyetlerinde insan faktörünün kritik rolünü vurgulamaktadır. Ekonomi ve sosyoloji bilimlerinde aktif olarak araştırılmakta olan sosyal ağ yapısı ve buna bağlı olarak incelenen davranış analizi kavramları yazılım mühendisliği araştırmalarında henüz yaygın bir uygulama alanı bulamamıştır. Özellikle sosyal ve ekonomik karar ve kişilik analizlerinde sıkça kullanılan bu kavramların yazılım mühendisliğine uygulanması yazılım firmalarının üretkenliğini, dolayısı ile yatırım karlılıklarının artırılmasını mümkün kılacaktır. Bu makalede, ekonomi alanında birçok uygulaması bulunan oyun kuramının bir alt dalı olan mekanizma tasarımı kullanılarak geliştirmekte olduğumuz bir süreç mühendisliği yaklaşımı anlatılmaktadır. Bu yaklaşım kullanılarak, yazılım geliştiren organizasyonların yapısının iyileştirilmesi ve buna bağlı olarak üretkenliklerinin artırılması hedeflenmektedir.

1. Giriş

Yazılım süreçlerinin iyileştirilmesi yöntemleri yazılım ürününün kalitesinin artırılmasını sağlayacak olan tüm aktiviteleri kapsamalıdır [1]. Bu aktiviteler yazılım geliştirilmesi süreçlerine etki eden insan faktörleri ve sosyal etkileşim sorunları gibi etkenler göz önüne alınarak yeniden değerlendirilmelidir [1,2]. Yazılım geliştirme süreçleri genellikle bir takım çalışması gerektirdiğinden sosyal bir aktivite olarak kabul edilebilir [3]. Yazılım takımı bireyleri çalışma koşullarının ve genel kişilik özelliklerinin bir sonucu olarak bireysellikleri, rasyonellikleri ve karşılıklı dayanışma gibi karakteristikleri ile tanımlanırlar [4]. Bu anlayış ile birlikte yazılım geliştirme süreçlerini etkileyen faktörlerin başında; bireylerin etkileşimi, sosyal ilişkilerin karmaşıklığı ve iletişim problemleri gelmektedir. Dolayısıyla, sosyal faktörlerin incelenmesi ve yazılımın sosyal yönünün araştırılması yazılım mühendisliği araştırmalarında önemli bir yer teşkil etmektedir.

Yazılım süreç mühendisliğinin temel hedeflerinden birisi yüksek kaliteli yazılım ürünleri elde etmek için proje planına ve bütçesine uygun bir yol haritası çıkarmaktır [5]. Bu yol haritası, son ürün veya sonuçlar dışında bu neticelere ulaşmak için tanımlanan ve yönetilen alt süreçlerin tanımlamalarından oluşur. Tipik bir yazılım geliştirme süreci, bütçe ve plan kısıtlarına uygun olarak tanımlanmış proje uygulaması sırasında oluşabilecek problemlerin çözümlerine yönelik aşamaları içerir. Bu bağlamda bir yazılım süreci, yazılım geliştirme sırasında yönetim, üretim ve bakım gibi yapısal sosyal aktivitelerin koordinasyonu ve bireylerin bu aktivitelerdeki rol ve yetenekleriyle tanımlanmalıdır.

Yazılım mühendisliği ekonomisi, yazılım mühendisliğinin önemli bir alt dalı olarak kabul edilir. Bu dal yazılım üretimi için fiyat tahmini, fiyatlandırma yöntemi belirleme ve risk düşürme konuları ile sıkça ilgilenmiştir. Günümüzde, iktisadi bir perspektiften bakıldığında yazılım geliştirme süreci sadece maliyet düşürme amaçlı yapılan bir iş olmaktan çıkıp bir katma değer yaratma işi halini almıştır. Yazılım süreç mühendisliği konularında yapılan araştırma ve geliştirme çalışmalarındaki yeni bulgular, yazılım mühendisliği ekonomisi alanında yapılan çalışmaların (özellikle değer yaratma ve karar verme konularındaki) yetersizliğine ve azlığına işaret etmektedir. Diğer bir taraftan farklı araştırma disiplinlerinden elde edilen bilgilerin yazılım geliştirme alanında kullanılmasının faydalı sonuçlar doğuracağı açıktır. Ekonomi bilimi sınırlı kaynakların verilen kararlar doğrultusunda nasıl daha verimli kullanılabileceğini araştırırken, benzer bir şekilde, yazılım geliştirme süreçlerinde bireyler ve takımlar yazılım ürününün gelişimini verdikleri kararlar ile belirlerler. Bu sebeple, takımların ve bireylerin uyumlarını, karar verme hızlarını ve verimlerini artırmak yazılım projesinin ekonomik yaşayabilirliği ve yatırım gereksinimlerini gözetmek için önemlidir.

Bu makalenin devamı şu şekilde oluşturulmuştur. Bir sonraki bölümde, oyun kuramı hakkında genel bilgi sunulacaktır. İkinci olarak mekanizma tasarımı kavramı ve bu kavramın yazılım uygulamalarındaki yeri üzerinde durulacaktır. Daha sonra yazılım ekosistemi kavramı anlatılacaktır. Üçüncü bölümde, yazılım mühendisliğinde oyun kuramı uygulamaları, dördüncü bölümde oyun kuramı kullanılarak yazılım takımlarının modellenmesi üzerine yaptığımız yaklaşım tanıtılacaktır. Beşinci kısımda ise araştırmamız sırasında vardığımız bazı öncül sonuçlar sunularak, yapılması planlanan çalışmalarımız ile ilgili genel bilgi verilecektir.

2. Oyun Kuramı

1930'ların sonunda ortaya çıkan oyun kuramı uygulamalı matematiğin, bireyler veya diğer çeşitli yapılar (örn: takımlar veya kurumlar) arasındaki gözlenebilir etkileşimi incelemek üzere geliştirilmiş analitik metotlara ve karar geliştirme araçlarına verilen genel ismidir. Bu kuram başta ekonomi, sosyoloji ve uluslararası ilişkiler olmak üzere sosyal bilimlerin ve daha sonraları fen bilimlerinin çeşitli alanlarına kullanım

sahası bulmuştur. Son on beş yıl içinde ise, oyun kuramı ekonomik analizlerin temel dili haline gelmiş, şirketlerin veya çalışanlarının davranışlarını incelemek için sıkça kullanılmaya başlanmıştır.

Oyun kuramı bireyler, kurumlar veya uluslar birbirleri ile etkileşince ortaya çıkabilecek sonuçları matematiksel bir oyun biçiminde inceler. Oyun kuramı genellikle oyuncular (bireyler, takımlar veya katılımcılar), etkileşimleri için belirlenmiş kurallar, oyuncuların yolgüdümsel kimliklere (strateji profilleri) (örn: davranışları veya kararları), ve bu kararları sonucunda elde edebilecekleri sonuçlarla ifade edilir. Oyun kuramında genel bir kabul, oyuncuların rasyonel olduğudur. Bu rasyonellik oyuncuların oyun kurallarını takip ettiklerini ve kazanmaya çalıştıklarını gösterir.

Oyun kuramı kullanılarak, tanımlanmış kurallar doğrultusunda etkileşen bireylerin davranış farklılıkları; koalisyon, birleşme ve ayrılma davranışları ve bu davranışların sebepleri ortaya çıkartılabilir. Son on yılda oyun kuramı ekonomi bilimi yanında diğer bilim dallarında özellikle dilbilimleri, biyoloji, psikoloji ve bilgisayar bilimleri alanlarında kullanım alanı bulmuştur. Oyun kuramı ile ilgili çalışmalar iki ana dal ile sınıflandırılabilirler: (1) Rekabetçi oyun kuramı, (2) İşbirlikçi oyun kuramı. Rekabetçi oyun kuramı, seçimleri birbirilerinin çıkarlarını etkileyen belirli bir miktar oyuncunun belli durumlar dâhilinde etkileşmesine dayanır. Bu oyuncular genellikle bireysel çıkarlarını düşünmeleri durumunda daha karlı çıkmaktadırlar. Daha sonra Nash dengesi diye anılacak olan, John Nash'in rasyonel oyuncuların nasıl oynamaları gerektiği ile ilgili çözüm önerisi, fiziksel bir sistemde denge durumunun oyuncuların yordam değiştirerek bir kar sağlayamayacaklarını anlayana kadar devam edeceğini öne sürer. Daha sonra oyunun kuralları doğrultusunda tüm oyuncular durumdan sağlayacakları kazançlarını en yükseğe çekmeye çalışacaklardır.

İşbirlikçi oyun kuramı, rekabetçi oyun kuramından farklı olarak oyuncuların iş birliği yapmasına olanak tanır. Bu durumda oyuncuların, çıkarları doğrultusunda istikrarlı işbirlikçi birleşimler oluşturmaları sağlanır. Bu yaklaşım, kooperatif çözüm önerileri oluşumuna, dolayısı ile kolektif çıktı oluşumuna izin verdiği için yazılım süreçlerinin, aktivitelerinin ve dolayısı ile ürünlerinin değerlendirilmesinde kullanılabilecektir. Yazılım takımlarının etkileşimlerinin analitik incelenmesi ve bu etkileşimlerin düzenlenerek bireylerin ve takımların üretkenliğin artırılması için gerek işbirlikçi gerekse rekabetçi oyun kuramı kavramlarının kullanılması mümkündür.

Yazılım geliştiren organizasyonlar tüm katılımcılarının (takımlarının ve bireylerinin) iş birliği ile oluşturulan bir sosyal yapı merkezinde çalışmalarını ister. Bu doğrultuda psikoloji ve sosyoloji bilimlerinde yapılan çeşitli araştırmalar bireylerin sosyal etkileşimleri ve daha iyi bütünleşmesi sayesinde takım uyuşmasını artırmayı hedefler. Dolayısı ile gruplar, aktiviteler, roller ve kişiler arasında uyumu sağlamak oyun kuramı gibi bir yaklaşım kullanılarak mümkün olabilecektir. Son yıllarda, oyun kuramının yazılım mühendisliği uygulamaları ile nasıl bütünleştirilebileceği konusunda çeşitli çalışmalar yapılmaya başlanmıştır. Deliller göstermektedir ki [6,7], yazılım geliştiren organizasyonlar yazılım yönetim aktivitelerinin yolgüdümsel (stratejik) doğasına uygun olarak hareket ederler. Proje sırasında alınan yolgüdümsel kararlara örnek olarak; çekince yüklenicilerin (stakeholders) belirlenmesi, bu bireylerin veya kurumların aralarındaki pazarlıkların irdelenmesi verilebilir. Sonuç olarak oyun kuramı yazılım yönetim ekiplerinin tanımladıkları birçok problemin çözümü için rahatlıkla kullanım alanı bulacaktır.

2.1. Mekanizma Tasarımı

Oyun kuramının bir alt araştırma dalı olan mekanizma tasarımı özellikle organizasyonlar düzeyinde alınan sosyal kararların firma çıktıları veya örgütsel sonuçlar üzerindeki olumlu etkisini artırma ilkesine dayanır. Bu yapı, mekanizma tasarımcısının veya sosyal planlayıcının organizasyonun sosyal yapısını tasarlaması ve böylelikle katılımcıların bireysel olarak teşvik edilmelerini hedefler. Dolayısı ile kişisel amaçlar organizasyon kapsamlı amaçlara dönüştürülür. Başka bir deyişle, mekanizma tasarımı bir organizasyonun sosyal kurgusunun değerlendirilmesi ile alakalıdır. Mekanizma tasarımı kuramına göre, bir organizasyon içerdiği sosyal örüntülere ve katılımcılarının verebileceği kararlar kümesine göre modellenebilir. Ayrıca, oyun kuramı kullanılarak organizasyon parametreleri ve beklenen neticelerin ilişkisi hakkında tahminler yürütülebilir.

Sosyoekonomik mekanizma tasarımı bir organizasyonu eniyileyerek çalıştırma yordamıdır. Diğer bir deyişle, bu yordam organizasyonu oluşturan bireylerden alınan birçok girdi kullanılarak belli kurallar aracılığı ile istenen çıktı kümesinin oluşturulması işidir. Bu doğrultuda amaç yazılım takımlarını, kişilerden alınan belli bilgiler doğrultusunda çıkarılan yolgüdümsel kimliklere (stratejik profillere) göre oluşturmaktır. Ayrıca, organizasyon tabanlı üretimin organizasyonun yapısal kalitesi ile ilgileşim (korelasyon) içinde olduğu [8] fikrinden yola çıkarak yazılım geliştiren organizasyonların yapısal iyileştirilmesinin önemi daha iyi anlaşılabilir.

Yazılım şirketleri bilgi işçilerinden oluşan entelektüel sermaye tabanlı bir organizasyon türü olduğundan, tüm yazılım geliştirme basamaklarında takımlar ve bireyler arası işbirliğinin yanında güçlü bir iletişim yapısının kurulması şarttır. Klasik görüşün bir parçası olan merkeziyetçilik ilkesi yerini organizasyon tabanına sorumluluğun dağıtılması ilkesine bırakmıştır. Bu değişim süreci sayesinde organizasyon birimleri kendini geliştirme ve dolayısı ile daha kolay evrilme yetenekleri kazanmaya başlamışlardır.

Mekanizma tasarımı teorisi ve onun yazılım geliştiren organizasyonlar üzerindeki uygulaması şirketlerin sosyal yapısının takım oluşturma yöntemleri ve dolayısıyla üretkenliği üzerindeki etkisini anlamamıza yardımcı olacaktır. Ayrıca bu sayede organizasyonun bilgi iletim yapısının düzenlenmesi ve iletişim kanallarının ve hızının şekillendirilmesinde de faydalı olması beklenmektedir.

Sosyoekonomik davranış, sosyal ve ekonomik faktörlerin kişiler veya organizasyonlar üzerindeki etkilerini inceleyen bir araştırma alanıdır. Dolayısı ile yazılım geliştiren şirketler için tasarlanacak bir sosyoekonomik mekanizma, yazılım geliştirirken kullanılan ekonomik ve sosyal aktivitelerin bireylerin etkileşimine katkısı üzerine kurulmalıdır. Bu

etkileşim modeli kişilerin seçimleri, davranışları ve tercihleri doğrultusunda belirlenen model ve sosyal yapıya uygun olarak geliştirilen kurallar ile kurgulanabilir. Bu sayede bireylerin sosyal ve ekonomik istekleri ve güdülenmeleri şirketin verimliliğini geliştirmek hedefine yönelik olarak şekillenir. Böyle bir mekanizma bireylerin seçimleri ile yazılım geliştiren organizasyonların sosyal görünüşü arasında bir köprü oluşturacaktır. Bu bağ sayesinde yazılım takımlarının yapı ve etkileşim modelleri daha anlaşılır bir hal kazanacaktır. Bu çaba ile yazılım geliştirme süreçlerinin verimliliği artırılarak şirketlerin sosyal ve ekonomik çıktılarının en yükseğe çıkarması amaçlamaktadır.

2.2. Yazılım Ekosistemi

Günümüzde yazılım geliştiren organizasyonların, iş ve teknik kapasitelerini birlikte artırdıkları ve dolayısı ile beraber evrimleştikleri gerçeği daha belirgin bir hal almıştır. Bu sebepledir ki, yazılım iş geliştirme görüşünde, "ürün geliştir ve sat" mantığı, yerini bileşen tipi (mashup) ürün geliştirme kurgusuna bırakmıştır. Bu kurguda; yazılım ürün hattı, tek bir firmadan ziyade aralarında kar ve kazanç ilişkisi bulunan birçok irili ufaklı firmanın etkileşmesinden oluşmaktadır. Bu ürün hattı şekilsel olarak biyolojik bir ekosistem [9] modeline benzer. Moore [10] yazılım ekosistemini yazılım ürününün üzerindeki bir ekonomik katman olarak tanımlamıştır. Mitleton-Kelly [11] sosyal organizasyon yapısını şirketlerin, müşterilerinin ve rakiplerinin ticari, teknik ve örgütsel ilişkilerden oluşan karmaşık ve katmanlaşmış bir yapı üzerinde beraber evrilen sosyal bir ekosisteme benzetmiştir. Bu tanımdan hareketle, endüstriyel bir yazılım ekosistemi birçok bilgi değişim ağ yapısı ve bu yapıların etkileşimi doğrultusunda kurgulanmalıdır. Bu kurgu birçok ticari yapının piyasa paylaşımı ilkesine bağlı olarak kolektif bir çıktı uğruna beraber çalışması ve bu varlıkların piyasadaki rolleri üzerinden incelenmelidir. Bu ticari ilişkiler parasal paylaşımlardan değil, proje ve yazılım çıktıları üzerinden bilgi ve tecrübe değişimine dayanmalıdır. Bu bağlamda yazılım ekosisteminin varlığının kabulü yazılım geliştiren organizasyonlardaki bireylerin ve takımların sosyal yapılarının (bağlanabilirlikleri, uyumları, etkileşimleri) ve bunların sonuçlarının araştırılması gereğini gündeme getirmektedir.

3. Yazılım Mühendisliğinde Oyun Kuramı Uygulamaları

Oyun kuramı uygulamaları yazılım mühendisliğindeki kökenini "Teori W" [12] ve "Spiral yazılım geliştirme" [13] modellerinde bulur. Bir proje yönetimi modeli olan Teori W, ürün geliştirirken alınan kararların incelenmesi temeline dayalıdır. Bu kurama göre, proje yöneticisinin temel amacı çekince yüklenicileri olabildiğince kazan-kazan ilkesinde birleştirmektir. Diğer bir deyişle, yazılım ürünü geliştirme süreçlerinde emek ve yatırım sahibi kişiler arasında oluşabilecek potansiyel çıkar çatışmalarının öngörülmesi ve çözülmesi temel amaçtır.

Yazılım mühendisliği alanında, oyun kuramının sınırlı sayıda uygulamaları bulunmaktadır. Lagesse [14] yazılım projelerinde görev dağılımını sağlamak için işbirlikçi oyun kuramı tabanlı bir model önermiştir. Grechanik ve Perry [4] yazılım geliştirme süreçlerini rekabetçi oyun kuramı analizi kullanarak incelemiştir. Yaptıkları çalışmada yazılım geliştiren bireyler ile yazılım geliştiren organizasyonlar arasındaki çeşitli hedef (çıkar) çatışmaları ve bu konuda yapılmış olan çalışmaların azlığına dikkat çekmişlerdir. Cockburn [15] yazılım geliştirme süreçlerini proje kaynakları ve bireylerin komünikasyon ve koordinasyon yeteneklerine dayalı bir oyun şeklinde ele almıştır. Baskerville'e göre [16] yazılım geliştirme işi, hızlı ve koordineli kaynak tüketimine ve dolayısı ile yazılım geliştiren bireylerin ve takımların karar verme kabiliyetlerine bağlıdır. Gerçek seçimler teorisi gibi çeşitli iktisadi kavramların yazılım mühendisliği alanına uygulanabilirliğini gösteren ve dolayısı ile yazılım tasarımı kararlarını sermaye yatırım kararlarına benzeterek analiz eden bir çalışma Sullivan ve arkadaşları [17] tarafından yapılmıştır. Bu çalışma yazılım süreçlerinde alınan birçok kararın, iktisadi değerlendirme yaparak, önceden hazırlanan stratejik planlar doğrultusunda eniyileme edilebileceğini iddia etmektedir. Sazawal ve Sudan [18] yaptıkları çalışmada yazılım geliştirici ile müşteri arasında geçen bir yazılım tasarım evrilme oyunu geliştirmişlerdir. Bu oyun sonucunda araştırmacılar basit oyun kuramı yaklaşımının yazılım ile ilgili çeşitli durumların tanımında faydalı olacağına kanaat getirmişlerdir.

Sosyal açmazlar genellikle kolektif ve kişisel çıkarlar arasında çıkan çatışmalardan kaynaklanırlar. Bir başka deyişle, oyuncular takım tabanlı kazançtan ziyade kişisel çıkarlarına daha fazla hizmet eden bir yol veya yordam geliştirebilirler. Mahkûmlar açmazı bu tür çıkar çatışması durumlarını analiz etmek için düşünülmüş bir oyun kurgusudur. Hazzan ve Dubinsky [19] yazılım geliştirme süreçlerindeki sosyal durumlardan yola çıkarak bu süreçlerdeki çatışmaları mahkûmlar açmazını kullanarak açıklamaya çalışmışlardır. Benzer bir şekilde, Feijs [20] yazılım testlerindeki olası mahkûmlar açmazı durumlarına dikkat çekmeye çalışmıştır. Bu çalışmada bir programcı ile yazılım test personeli arasında oluşabilen bir mahkûmlar açmazı modeli incelenmiştir. Bütün bu çalışmalar göstermektedir ki oyun kuramı yazılım mühendisliği alanında uygulama yeri bulabilmektedir. Bu tür yaklaşımlar, yazılım mühendisliği takımlarının iş birliği yapması konusundaki sorunların tespit edilmesi için kullanılabilir. Bu analizin sonucunda, gerekli görülen önlemlerin alınması ve dolayısı ile takımların ve bireylerin daha fazla işbirliği yapmaya yöneltilmesi yazılım ürününün kalitesinin ve takım üretkenliğin arttırılması adına önemli bir etkendir.

4. Yazılım Takımlarının Modellenmesi

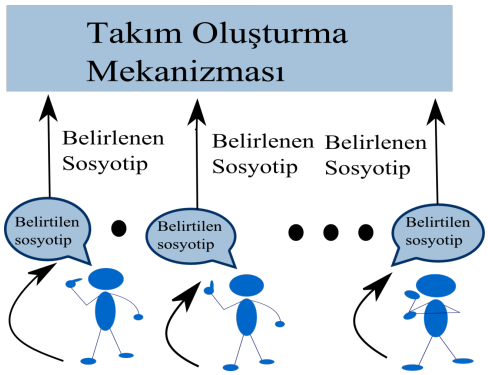
Yazılım geliştirme işinin bir bilgi değişim ekonomisi olduğunun kabulü ile birlikte, yazılım geliştirme süreçlerinde ve buna bağlı örgütsel araştırmalara yapılabilecek olan katkılardan birisi yazılım takımlarının ve yazılım geliştiren organizasyonların sosyoekonomik davranış biçimlerini anlamak ve modellemektir. Yazılım geliştiren organizasyonun sosyal yapısına bağlı olarak, bir yazılım takımı farklı işler üstlenmiş değişik kişilik özelliklerine sahip kişilerden oluşur.

Daha önceki çalışmalarımızda önerdiğimiz gibi [21], yazılım şirketlerinin verimliliğini artırmak için bireylerin ve takımların etkileşimlerini ve davranış şekillerini inceleyerek en uygun şirket yapısı veya takım biçimlenimi kurgulanabilir. Sonuç olarak, bir yazılım proje yönetim biriminin önemli

görevlerinden birisi, yazılım geliştiren organizasyonların örgütlenme yapısını geliştirerek daha fazla üretkenlik sağlanmaktır.

4.1. Yazılım Takımı Oluşturma Mekanizması

Yazılım geliştiren organizasyonların yönetim birimlerinin önemli görevlerinden biri en etkin takımları oluşturma işidir. Proje yöneticileri veya yazılım yönetim grupları takımların ve bireylerin refahlarını sağlama konusunda sosyal etkenlerin farkına varmalı, grup oluştururken bireylerin sadece teknik bilgi düzeylerini değil sosyal kişilik tiplerini de (sosyotip) dikkate almalıdır. Bu problem, yazılım geliştiren organizasyonların sosyoekonomik bir mekanizma olarak algılanması ile çözümlenebilir. Bu durumda sosyal planlayıcı (proje yöneticisi) takımları oluşturacak potansiyel bireylerin uyumluluk katsayısı ile en uygun (sosyal) takım biçimlenimini belirler. Takım oluşturma mekanizması şu şekilde işleyecektir; bireylere ilk olarak daha önce belirlenmiş kişilik tipleri kümesinden hangisine ait oldukları sorulacaktır, ikinci olarak bu kişilik tipleri ile geliştirdiğimiz oyun kuramsal kişilik analiz göstergesi birlikte kullanılarak yazılım takımının kişilik tipleri belirlenecektir. Kişilik analiz göstergesi yazılım takımında bulunan bireylerin kişilik tiplerini ve sosyal uyum olanaklarını belirleyen bir analiz aracıdır. Üçüncü olarak daha önce başarılı olarak belirlenmiş yazılım takımlarında gözlenmiş kişilik tipleri hakkında elde edilen bilgiler aracılığı ile hazırlanan örüntülerden faydalanarak potansiyel olarak üretkenlik katsayısı en yüksek olacak yazılım takımları tahmin edilmeye çalışılacaktır (bkz: şekil 1).



Şekil 1: Takım oluşturma modeli.

Yazılım geliştirme proje süreçlerinde kurgulanacak takım oluşturma oyununda; (***i***) belirli sayıda seçeneği bulunan bir oyuncu kümesini temsil etsin. Bu oyuncular s_i davranışını bir ***S*** kümesinden seçsin. Bu durumda S_{-i} ise ***i*** oyuncusunun seçmediği davranışları göstersin. Bu durumda $s_i \in S_{-i}$ diğer oyuncuların seçimlerine göre şekillenecektir. Bireylerin kazançları yazılım takımlarındaki diğer bireylerin karlarına dolayısı ile onların seçimlerine (davranış biçimlerine) bağlı olacaktır.

Ancak sosyal açmazların işaret ettiği gibi bireyler bir şekilde bencil davranmalarından ötürü daha karlı çıkacaklarını düşünebilirler ve dolayısı ile oluşturulan takımların verimliliği değişkenlik gösterecektir. Bu durumda yazılım ekibinde yer alan tüm bireyler çeşitli etkileşimlerden doğacak farklı türleri Θ kişilik biçimi kümesinin elemanlarından biri olarak θ_i ile gösterelim. Potansiyel olarak tüm bireylerin karakteristiklerini, bu bireylerin etkileşimlerini ve birleşimlerini gösteren fayda fonksiyonunu ise u_i (s_i, s_{-i}, θ_i) ile gösterelim. Bir Nash dengesi karar kurallarından oluşan bir eşitsizlikle gösterilirse;

$$E\left[u_i\left((s_i(.),(s_{-i}(.)\right)|\theta\right] \geq E\left[u_i\left((s_i'(.),(s_{-i}'(.)\right)|\theta\right]$$

Takım oluşmasını sağlayan bir karar kuralı $((s_i(\theta_i))$ ile gösterilirse ve eşitsizlikteki sosyal etkileşim kuralları; *(s(i)…s(n))* olarak ifade edilsin, en verimli takım oluştururken kurguladığımız en uyumlu takım $\bar{\theta}_i$ kişilik profilini kullanarak gerçekleştirdiğimizi kabul ederek, bu profil uygulandığında mekanizmanın çalıştığını ve en verimli takımları oluşturduğumuz durumu şu şekilde tanımlayabiliriz.

$$E\left[u_i\left((s_i(\bar{\theta}_i),(s_{-i}(\bar{\theta}_i)\right)\middle|\bar{\theta}\right] \geq E\left[u_i\left((s_i'(\bar{\theta}_i),(s_{-i}'(\bar{\theta}_i)\right)\middle|\bar{\theta}\right]$$

Özetle, önerdiğimiz yöntem yazılım projeleri ve özellikle görevleri için doğru takımları oluşturarak, yazılım mühendisliği araştırmalarında ele alınan başlıca sıkıntılarından biri olan üretkenlik tabanlı problemlere çözüm bulmayı hedeflemektedir. Yazılım takımlarının kişilik tiplerini, kişilik analizi göstergesi ve oyun kuramı kullanılarak geliştirdiğimiz mekanizma tasarımı sayesinde, birlikte daha verimli ve uyumlu çalışabilecek takımlar oluşturmayı hedeflemekteyiz.

5. Sonuçlar ve Gelecek Çalışmalar

Yazılım geliştirme süreçleri rasyonel bireylerin organizasyon amaçları doğrultusunda belirlenmiş hedeflere doğru çalışması koşulunu gerektirir. Yazılım geliştiren organizasyonların üretkenlikleri, çalıştırdıkları bireylerin ürün geliştirme konularında yeterli oranda isteklendirilmelerine bağlıdır. Oyun kuramı organizasyonları ve hedeflerini bireysel ve rasyonel kabul edilebilecek aktörlerin bakış açısı ile modellemeyi hedefler. Bu bakış açısı gerek örgütsel gerekse bireysel etkileşim ve buna bağlı sonuçları beraberinde getirir. Bu sebeple, yazılım geliştiren organizasyonlar bir yandan kaynak dağıtımı yapmak, bir yandan da verimli kaynak kullanımını desteklemek amacı ile takım yapılandırmalarını daha iyi çalışır hale getirme yöntemleri geliştirmek zorundadır.

Yazılım geliştirme, etkileşim halinde olan bireylerin entelektüel birikimleri ile müşterek olarak bir bilgi ve iletişim ağı üzerinde çalışması sonucunda ürün geliştirme işidir [21]. Bununla beraber, sosyal üretkenlik kavramı sosyal etkileşim kalitesinin artırılmasını yöntemi ile üretkenliğin ve verimin yükselmesini ifade eder. Ayrıca yazılım takımlarının sosyal yapısının ve refah düzeyinin geliştirilmesi ile takım işbirliği ve takım üretkenliğinin daha yüksek dereceye çıkarılması hedeflenmelidir.

Bu araştırma temel hipotezini yazılım geliştirme faaliyetlerini sosyoekonomik aktiviteler olarak kabulüne dayandırır. Bu kabul sayesinde, yazılım geliştiren organizasyonların kurgusu bir sosyoekonomik mekanizma tasarımı problemi olarak incelenebilecektir [21]. Bu makalede yazılım takımlarının oyun kuramı kullanılarak örgütsel açıdan daha verimli kullanılmasını hedefleyen bir yaklaşımda bulunulmuş, kişilerden ve organizasyonlardan beklenen çıktıların

eşgüdümlü olarak en çoğa yükseltgenmesi için geliştirilmiş bir yaklaşım anlatılmıştır. Bir sonraki hedefimiz kurguladığımız modeli verimli takım biçimlenimleri göz önünde bulundurarak endüstriyel ortamda (orta ölçekli yazılım geliştiren organizasyonlarda) uygulayarak yazılım takımlarının daha verimli kılınması için kullanmaktır. Elde edilen bulgular ve kural kümeleri doğrultusunda, model daha verimli takım dokuları oluşturmak için düzenlenecektir.

6. Teşekkür

Bu çalışma İrlanda Bilimsel ve Teknolojik Araştırma Kurumunun, Lero – İrlanda Yazılım Mühendisliği Araştırma Merkezi'ne verdiği 03/CE2/I303-1 kodlu destek programı dahilinde desteklenmektedir.

7. Kaynakça

[1] R. Conradi and A. Fuggetta, "Improving Software Process Improvement," IEEE Software, vol. 19, no. 4, pp. 92–99, 2002.
[2] R. L. Glass, Facts and Fallacies of Software Engineering. Addison-Wesley Professional, 2002.
[3] Y. Dittrich, C. Floyd, and R. Klischewski, Social thinking-software practice. The MIT Press, 2002.
[4] M. Grechanik and D. E. Perry, "Analyzing Software Development as a Noncooperative Game," in IEE Seminar Digests, 2004, vol. 29.
[5] S. Zahran, Software Process Improvement: Practical Guidelines for Business Success. Addison Wesley, 1998.
[6] F. P. Deek, J. A. McHugh, and O. M. Eljabiri, Strategic software engineering: an interdisciplinary approach. CRC Press, 2005.
[7] T. Dingsøyr, T. Dybå, and N. B. Moe, Agile Software Development: Current Research and Future Directions, 1st ed. Springer, 2010.
[8] M. E. Conway, "How do committees invent," Datamation, vol. 14, no. 4, pp. 28–31, 1968.
[9] N. H. Madhavji, M. Lehman, D. Perry, and J. F. Ramil, Software evolution and feedback. Wiley Online Library, 2006.
[10] J. F. Moore, The Death of Competition: Leadership and Strategy in the Age of Business Ecosystems. Harper Paperbacks, 1997.
[11] E. Mitleton-Kelly, Complex systems and evolutionary perspectives on organisations. Emerald Group Publishing, 2003.
[12] B. Boehm and R. Ross, "Theory-W software project management principles and examples," Software Engineering, IEEE Transactions on, vol. 15, no. 7, pp. 902–916, 1989.
[13] B. Boehm, A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy, "Using the WinWin Spiral Model: A Case Study," Computer, vol. 31, no. 7, pp. 33–44, 1998.
[14] B. Lagesse, "A Game-Theoretical Model for Task Assignment in Project Management," in 2006 IEEE International Conference on Management of Innovation and Technology, Singapore, 2006, pp. 678–680.
[15] A. Cockburn, Agile software development: the cooperative game. Addison-Wesley, 2007.
[16] R. L. Baskerville, L. Levine, B. Ramesh, and J. Pries-Heje, "The High Speed Balancing Game: How Software Companies Cope with Internet Speed," Scandinavian Journal of Information Systems, vol. 16, no. 1, pp. 11-54, 2004.
[17] K. Sullivan and P. Chalasani, "Software Design Decisions as Real Options," 1997.
[18] V. Sazawal and N. Sudan, "Modeling Software Evolution with Game Theory," Trustworthy Software Development Processes, pp. 354–365.
[19] O. Hazzan and Y. Dubinsky, "Social Perspective of Software Development Methods: The Case of the Prisoner Dilemma and Extreme Programming," in Extreme Programming and Agile Processes in Software Engineering, Springer, 2005, pp. 74–81.
[20] L. Feijs, "Prisoner Dilemma in Software Testing," Computer Science Reports, vol. 1, pp. 65–80, 2001.
[21] M. Yilmaz, R. V. O'Connor, and J. Collins, "Improving Software Development Process through Economic Mechanism Design," in Systems, Software and Services Process Improvement, 2010, pp. 177–188.