

## Yazılım Mühendisliğinde Başarılı Deneyleri Nasıl Gerçekleştirebiliriz ?

Çağatay ÇATAL<sup>1</sup>, Banu DİRİ<sup>2</sup>

<sup>1</sup> TÜBİTAK, Marmara Araştırma Merkezi, Bilişim Teknolojileri Enstitüsü, Kocaeli

<sup>2</sup> Yıldız Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü, İstanbul

cagatay.catal@bte.mam.gov.tr, banu@ce.yildiz.edu.tr

**Özet:** Yazılım Mühendisliği disiplini içerisinde, yaygın olarak kabul gören fikirlerin çoğunun kesinliği söz konusu değildir. Günümüzde; deneysel verilerle desteklenen nesnel bilgiler yerine, pazarlama stratejisi başarılı olan fikirlerin yaygınlık kazanmış olduğunu görmekteyiz. Her geçen gün yeni bir programlama paradigması, geliştirme aracı ya da yöntem ortaya çıkmakta ve bu yaklaşımların yararları öncekileriyle deneysel olarak karşılaştırılmadan yazılım sektöründeki popülerliğine bağlı olarak kabullenilmektedir. Yazılım Mühendisliğinin geçen 40 yıla rağmen yeterli olgunluğa erişememesinin en önemli nedenlerinden birisi de bu tür deneysel çalışmaların yeterince yapılmamasıdır. Araştırmacıların eğitimleri sırasında bu kapsamda bilgilerin sunulmaması; bu tür deneylerin ülkemizde gerçekleştirilmemesine, gerçekleştiği durumda ise yeniden üretilebilir ve başarılı deneylerin oluşmamasına neden olmaktadır. Bu çalışmada, Yazılım Mühendisliği için deneylerin önemi, ne tür bir süreçle gerçekleştirilmesi gerektiği ve başarılı deneysel Yazılım Mühendisliği çalışmalarının raporlanma şekli konusunda bilgiler sunulmaktadır. Ülkemizde Yazılım Mühendisliği alanında çalışma yapan araştırmacıların deneylere daha fazla ilgi göstermesi sayesinde, ulusal yazılım sektörümüzün hızlıca değişen süreç ve teknolojik eğilimlerden olumsuz yönde etkilenmemesi sağlanacaktır. Aksi halde Yazılım Mühendisliği uygulayıcıları; gittikçe artan yaklaşımlar, yöntemler, araçlar ve teknikler ortamında, nesnel deneysel veriler yerine moda uyararak seçimlerini gerçekleştirebilir ve ulusal bir yazılım krizi doğabilir.

**Anahtar Kelimeler:** Deneysel Yazılım Mühendisliği, Empirik Yazılım Mühendisliği, Deney.

### How Can We Perform Successful Experiments In Software Engineering?

**Abstract:** There is no certainty about most of the ideas which are commonly accepted in Software Engineering discipline. Nowadays, we see that the ideas which have successful marketing strategies are widespread instead of objective knowledge supported by experimental data. With each passing day, a new programming paradigm, development tool or method come into play and are adopted with respect to their popularity in software sector rather than comparing their benefits with previous ones. One of the most important reasons why Software Engineering could not reach enough maturity even 40 years is the lack of performing these experimental studies. Not providing knowledge to researchers during their education causes not to be performed such experiments in our country and even they are performed, it causes not to have a repeatable and successful experiment. In this study; importance of experiments for software engineering, a process to perform these experiments and the way how to report successful software engineering studies are presented. Thanks to the Software Engineering researchers who will be more interested in experiments in our country, our National software sector will not be affected negatively from rapidly changing process and technological trends. Otherwise, Software Engineering prac-

tioners may choose approaches, methods, tools and techniques according to the fashion instead of objective experimental data and a national software crisis may occur.

**Keywords:** Experimental Software Engineering, Empirical Software Engineering, Experiment.

## 1. Giriş

Yazılım Mühendisliği terimi, 1968 yılında Almanya, Garmisch’de düzenlenen NATO Yazılım Mühendisliği konferansı sayesinde popülerlik kazanmıştır. Her yıl düzenlenen Uluslararası Yazılım Mühendisliği Konferansı (International Conference on Software Engineering) ilk kez 1973 yılında ABD, Washington’da; ülkemizde ise (UYMS) ilk kez 2003 yılında İzmir’de düzenlenmiştir.

NATO konferanslarını Yazılım Mühendisliği için başlangıç noktası olarak kabul edersek, son 40 yıl içerisinde Yazılım Mühendisliği disiplininin önemli aşamalar kaydettiğini ancak hala yeterli olgunluğuna erişemediğini söyleyebiliriz. Örneğin, yazılım fonksiyonel büyüklük kestirimi problemini ele alalım. Bu noktada, yazılım uygulayıcısı olarak, hangi yöntemi tercih edeceğimiz çoğu zaman yaşadığımız coğrafyaya, o anki kestirim yöntemlerinin popülaritesine ve bilgi birikiminize bağlı olarak şekillenmektedir. Bu amaçla, uzun yıllar yazılım mühendisliği kitaplarında tek yöntem olarak tanıtılan IFPUG yöntemi yerine, son yıllarda Mark II yöntemi popülerlik kazanmış ve şu anda COSMIC isimli hesaplama yöntemi ön plana çıkmıştır. İngiltere’de Mark-II popüler iken, ABD’de IFPUG’un daha fazla tercih edildiği dönemler olmuştur. Bu örnekte de görüldüğü gibi Yazılım Mühendisliği içerisindeki bir çok fikir, nesnel veriler yerine fikrin çok fazla kişi tarafından kullanılıp kullanılmadığına bağlı olarak değer kazanmaktadır. Her geçen gün yeni programlama paradigmaları, mimari yaklaşımlar, yazılım geliştirme süreçleri önerilmekte ve bu fikirler destekleyicileri tarafından öznel (subjective) düşüncelerle aşırı derecede savunulmaktadır.

Günümüzde yazılım mühendisliği içerisindeki fikirler de ürünler gibi, maalesef pazarlama stratejilerine bağlı olarak yaygınlık kazanmaktadır. Oysa ki tüm mühendislik disiplinlerinde öznel düşünceler yerine, deneylere bağlı olarak fikirler oluşmalı ve nesnel (objective) deneysel verilerle yaygınlık kazanmalıdır. Yazılım üzerinde deneyleri gerçekleştirerek teoriler ortaya koyan Yazılım Mühendisliği dalına **Deneysel Yazılım Mühendisliği** adı verilmektedir. Yazılım Mühendisliğinin bu dalında yeterli çalışma yapılmaması nedeniyle, bu disiplin halen istenen olgunluk düzeyinde değildir. Yazılımın yeniden kullanılabilirliği konusunda yapılan çalışmalara bakarsanız; 1960’larda alt rutinlerin, 1970’lerde modüllerin, 1980’lerde nesnelere, 1990’larda bileşenlerin ve 2000’lerde yazılım ürün hatlarının bu kapsamda kullanıldığını görebilirsiniz. Ancak bu yaklaşımlarla yazılımın yeniden kullanılabilirliğinin artıp artmadığı ve üretkenliğin ne oranda etkilendiğini ortaya koyacak deneysel çalışmalar yapılmamıştır.

Fonksiyonel programlamadan nesneye yönelik ve daha sonra bileşen tabanlı programlamaya geçilmesiyle birlikte geliştiricilerin üretkenliklerinin ve programların kalitesinin arttığı düşünülmektedir [5]. Bu yaklaşımların geliştirilmesinin üzerinden 30 yılı aşkın bir süre geçmesine rağmen, sistematik olarak bu faydalar analiz edilmemiştir. Bu nedenle yıllar sonra, C++ dilinin programcı üretkenliği konusunda olumsuz etkilerine ilişkin güçlü kanıtlar içeren makaleler ortaya çıkmıştır [4]. Yaklaşımları üretirken ortaya konan yaratıcılık ve istek, bu yaklaşımları kıyaslama noktasında ortadan kalkmakta ya da zayıflamaktadır. Frederick Brooks [1] Yazılım Mühendisliği’nde her derde deva olan gümüş kurşunun (silver bullet)

bulunmadığını 20 yıl öncesindeki makalesinde ifade etmiş olmasına rağmen, Yazılım Mühendisleri bu arayışı sürdürmektedir. Son yıllarda popüler olan servis yönelimli mimari, model güdümlü mimari ve ürün hattı mimarisi gibi yaklaşımların birbirlerine göre avantajları ve dezavantajları değerlendirilmeden bir yaklaşımı tüm problemlere çözüm bulacak bir gümüş kurşun olarak görmek mühendislik hatasıdır. Yöntemleri, araçları ve yaklaşımları içerisinde buldukları koşullardan, proje ortamlarından, çözülecek problemlerden ve kalite beklentilerinden bağımsız düşünmek istenen sonuca geliştiricileri ulaştıramaz.

Yazılım Mühendisliği içerisindeki çok az sayıda düşünce, deneysel verilerle ortaya konulmuştur. 1968 yılında ortaya konan ünlü Yazılım Krizi [8] bile deneylerle doğrulanan bir olgu olmaktan ziyade topluluk içerisindekilerin öznel izlenimleriydi. Bir mühendislik dalı içerisinde bir krizden bahsediyorsak diğer mühendislik dallarındaki kusur oranları ile kıyaslamalar yapılmalıdır [5]. Kusur oranları diğer dallara göre güvenilirliğin çok alt seviyede olduğunu gösterirse, bu durumda bir krizden bahsedilebilir. Ancak 1968 yılında böyle bir çalışma yapılmamış ve yazılım krizinin patlak verdiği öznel olarak ifade edilmiştir. Bu kriz ifadesine karşın, Maibaum'un çalışmasında [6] Turski ve Hoare adlı araştırmacılar yazılım krizi olmadığına dair iddialarda bulunmuştur [5].

Zelkowitz [11], yazılım dergilerindeki makalelerden sadece %30'unun deneysel geçerliliğinin bulunduğunu ve sadece %10'unun formal bir yöntem izlediğini ifade etmiştir [5]. Tichy [9], deneysel geçiş ihtiyacı bulunan 400 yazılım mühendisliği araştırma makalesinden %40'ının hiç deneysel bilgi içermediğini raporlamış ve diğer disiplinlerde bu oranın %15 olduğunu ifade etmiştir [5].

ABD'de Ulusal Bilim Vakfı'nın 1998 yılında düzenlediği yazılım çalıştayında, farklı strate-

jiler gündeme gelmiş ve bu stratejilere bağlı olarak belirlenen hedeflerden birisi de aşağıda ifade edilen deneysel yazılım mühendisliği üzerinedir. Bu çalıştayda, "başarılı projelerin empirik olarak incelenmesi ile yazılım oluşturmadaki yararlı prensiplerin çıkarılması ve araştırma literatüründe ya da başka yerlerde geliştirilen tasarım prensiplerinin geçerlenmesi; Yazılım Mühendisliği sürecini anlayışımız açısından yeni yaklaşımların deneylerle incelenmesi" hedefi gündeme gelmiştir [5]. Ülkemiz açısından da, deneysel çalışmalarla ne tür araçların, tekniklerin ve yaklaşımların yazılım projelerimizde başarıyı arttırdığının incelenmesi kritik öneme sahiptir.

Bildirinin ikinci bölümünde Yazılım Mühendisliği içerisindeki bilimsel yöntemler, üçüncü bölümünde neden yeterince deney yapılmadığı, dördüncü bölümünde deneylerin neden tekrarlanması gerektiği, beşinci bölümünde deneyler için bir süreç, altıncı bölümünde sonuç ve gelecek çalışmalar sunulmakta, yedinci bölümünde ise referanslar verilmektedir.

## 2. Bilim ve Yazılım Mühendisliği

"Araştırma (research), insanoğlunun gönüllü ve bilinçli olarak bir sorun hakkında tartışılmaz bilgi arayışı kapsamında gerçekleştirdiği aktivitedir" [5]. Araştırmaları iki grupta değerlendirebiliriz [5]:

- *Bilimsel araştırmalar (scientific research)*: Kapsam, *teknolojik araştırmalara* göre daha geniş ve tek bir problemin çözülmesi ile araştırma sonlanmaz.
- *Teknolojik araştırmalar*: Amaç daha belirlenmiş olup, problemin uygun şekilde çözümü sağlandığında aktiviteler sonlandırılabilir.

Yazılım Mühendisliği alanı içerisinde dört araştırma yöntemi mevcuttur [3],[10]:

- **Bilimsel yöntem:** Gözlemlenen dünyaya göre modeller kurulur. Örnek olarak bir denizaltı simülasyon modeli verilebilir [10].
- **Mühendislik yöntemi:** Mevcut çözümler optimize edilerek yeniden değerlendirilir.
- **Empirik yöntem:** Bir model önerildikten sonra empirik çalışmalarla değerlendirilir. Anket, vaka çalışması ve deneyler empirik yöntemlerdendir [10].
- **Analitik yöntem:** Formal teoriler empirik gözlemlerimiz dikkate alınarak değerlendirilir.

Bu bildirinin konusu, Yazılım Mühendisliğindeki deneyler olduğu için sadece Empirik Yöntemler üzerinde durulacaktır. Empirik çalışmalarda 2 farklı yaklaşım söz konusu olabilir [10]:

- **Kantitatif:** Bu tür çalışmalar, çeşitli değişkenler arasında sayısal bir ilişki bulmaya odaklanmıştır.
- **Kalitatif:** İlişkilerin nedenlerini aramak üzere yapılan çalışmalardır.

Genellikle kalitatif çalışmalarla hipotezlerin oluşturulması sağlanırken, kantitatif çalışmalarla hipotezlerin doğruluğu sınanarak gerçeklikle ilişkisi belirlenir [5]. Çoğu araştırmacı, kalitatif ve kantitatif yaklaşımları nesnel ve öznel olma durumu ile ilişkilendirir ancak böyle bir durum söz konusu değildir. Kantitatif ya da kalitatif yaklaşımların her birisi duruma göre nesnel ya da öznel özellik taşıyabilir. Örneğin; isterleri 1-10 arasında anlaşılma düzeyine göre değerlendirirsek bu öznel kantitatif bir çalışmadır [5]. Bir test aracının bir grup içerisinde tercih edilme nedeninin incelenmesi öznel kalitatif çalışma iken, modüllerin birbirini çağırmasını gösteren ağacın incelenmesi nesnel kalitatif bir çalışmadır.

### 3. Yetersiz Deneyler

Yazılım Mühendisliği alanında yurt içi ve yurt dışında çalışma yapan araştırmacıların deneysel araştırmalara gösterdiği ilgi gereken düzeyin çok altındadır. Üniversitelerimizdeki akademisyenlerimiz, sahip oldukları öğrenci potansiyelini kullanarak bu deneyleri kolaylıkla gerçekleştirebilir. Bu aşamadan sonra, üniversite-sanayi işbirliğine geçilerek aynı deney şirketlerin kendi ortamlarında yeniden gerçekleştirilebilir. Yazılım Mühendisliği içerisinde, deney yapma konusunda birçok yanlış inanç mevcuttur. Bu düşüncelerin temeline inildiğinde; araştırmacıların eğitim sürecinde bu kapsamda yeterince bilgiyi ve motivasyonu almadığını görmekteyiz.

Yazılım Mühendisliği içerisindeki yetersiz deneylerin gerekçeleri aşağıdaki şekilde sıralanabilir [5]:

1. Yazılım mühendislerinin eğitimi sırasında, bilimsel yöntemlerin bu disiplindeki önemi vurgulanmadığı için bu tür yöntemlerin doğa bilimlerine uygun olduğunu düşünülmemektedir.
2. Yazılım mühendisleri, istatistik konusundaki yetersiz bilgileri nedeniyle deney verilerinin analizini kolaylıkla nasıl gerçekleştireceklerini bilememektedirler. İstatistik dersinin okutulması sırasında, yazılım mühendisliğindeki deneylerle ilişki kurulmadığı için bu bilgiler soyut düzeyde kalmakta ve yıllar sonra uygulamada önemli sorunlar doğurmaktadır.
3. Yazılım mühendisliğindeki deneysel tasarım ve analiz kitapları yetersizdir. Farklı terminolojilerin kullanıldığı diğer kaynaklardan (kimyasal deneyler ya da ilaç endüstrisi) bu bilgileri anlamak kolay olmadığı için, bu tarz kitapların sayısı artırılmalıdır. 5 ve 10 numaralı referanslar, bu kapsamda akademisyenlere yarar sağlayacaktır.

4. Farklı bir araştırmacı tarafından önerilen bir yaklaşımın geçerliliğini sınavacak empirik çalışmalar yürütmenin, çok fazla yayımlanabilir bir özellikte olmadığı düşünülmektedir. Yazılım Mühendisliğinin mevcut durumunda, araştırmacıların çok büyük kısmı yeni yaklaşımlar ve fikirler üretmektedir. Birçok disiplinde bu durum 2 farklı grup tarafından ele alınır: Teorisyenler ve deneyciler. Teorilerin üretilmesi teorisyenlerin sorumluluğunda iken deneyciler bu fikirleri test ederler. Üniversitelerimizde deneycilerin sayısını arttırmak üzere yüksek lisans / doktora tezleri gerçekleştirilerek Yazılım Mühendisliği disiplini içerisindeki bu problem aşılmaya çalışılmalıdır.

5. Yazılım geliştirmeyi etkileyen çok fazla faktör olduğu için, bu faktörleri kontrol ederek deneyleri gerçekleştirmek zor bir süreçtir.

6. Yazılım sektöründeki ürünlerin geliştirme maliyetleri çok yüksek olduğu için, farklı yaklaşımlar güvenilir şekilde uygulanıyor olabilir. Yeni bir araç/teknik/yöntemle projenin başarısızlıkla sonuçlanması beklenirken, proje bütçeleri sayesinde yeni çalışanlar projeye eklenerek bu olumsuzluğun önüne geçilebilir.

7. Yazılım mühendisleri arasında deney yapma bilinci yerleşik olmadığından çok fazla deney yapılmamaktadır.

8. Teknoloji çok hızlı değiştiğinden bu tür deneylerin gereksiz olduğu düşünülür. Ancak 80'lerin sonu 90'ların başında popüler olan nesneye yönelik programlamanın üretkenlik konusundaki etkileri uzun yıllar irdelenmemiş ve sonunda olumsuz yöndeki gelişmelerin raporlandığı deneyler ortaya çıkmıştır.

9. Deney yapmanın çok masraflı olduğu düşünülmektedir ancak anlamlı deneyler kü-

çük bütçelerle sağlanabilir. Bazı araştırmacılar ise Yazılım Mühendisliğinde yeterince deney yapıldığını düşünmektedir.

#### 4. Deneylerin Tekrarlanması

Gerçekleştirilen bir deneyin doğru şekilde raporlanması sayesinde deneylerin tekrarı söz konusu olabilir. Aksi halde; muğlak ifadeler ve deney ortam şartlarının açıklanmaması, deneylerin yeniden üretilebilirliğinin önündeki büyük engellerdir. Bu noktada; deneylerin tekrarlanabilirliği konusunda tarihteki bazı çarpıcı olayları sunmak yararlı olacaktır. 1962'de Schibuzawa isimli araştırmacı TRF isimli hormonu izole ettiğini ve aminoasit oluştuğunu iddia etmiştir [5]. Schibuzawa'nın bu çalışmaları, bilim çevrelerince eleştirilmiş ve farklı laboratuvarlarda TRF'nin aktif olmadığı ifade edilmiştir. Diğer laboratuvarlarda deneyleri tekrarlaması için davet edildiğinde ise bu davetlere katılmamıştır. Schibuzawa'nın iddialarının güvensiz olduğu ifade edilerek bilim çevrelerince görüşleri onaylanmamıştır. Bu araştırmacı, 1962 yılından sonra hiç makale yazmamış ve bu problemi çözdüğüne ilişkin iddialar zamanla ortadan kaybolmuştur. 10 yıl sonra, Guillermin ve Schally isimli araştırmacılar TRF izolasyonunu gerçekleştirerek (aminoasit oluşumu dışında) Nobel ödülünü kazanmıştır [5]. Bir yanda deneysel çalışmasını doğru şekilde bilim insanları ile paylaşmayan araştırmacının geldiği nokta ve diğer yanda da bu araştırmayı yeniden üretilebilir bir şekilde gerçekleştiren, önemli bir ödülün sahibi araştırmacılar. Bu olay gösteriyor ki deneylerden elde edilen sonuçlar kadar, doğru raporlanması, yeniden üretilmesi noktasında yeterli bilgilerin bilim insanlarıyla paylaşılması kritik öneme sahiptir. Benzer şekilde, 1989'da Utah Üniversitesi'nden Pons ve Fleischmann isimli fizikçiler soğuk füzyonu gerçekleştirdiklerini dünyaya duyurmuştu ancak bu deneyler başarılı şekilde tekrarlanmadığı için bilgileri geçerlilik kazanmadı [5]. Bilgileri çarpıtarak

sunmanın ötesinde bazı durumlarda araştırmacılar, ne tür bilgilerin raporlanması gerektiğini bilememektedir. Farklı bilim insanları deneyi yeniden tekrarladığında sunulan eksik bilgiler nedeniyle farklı sonuçlar elde ederse, deney doğru şekilde gerçekleştirilmiş olsa dahi bilim çevrelerince eleştiri alabilir. Deneylerin tekrarlanmasını 2 alt gruba ayırabiliriz [5].

1. *Dış tekrarlar (external)*: Farklı araştırmacılar tarafından gerçekleştirilen deneylerdir.

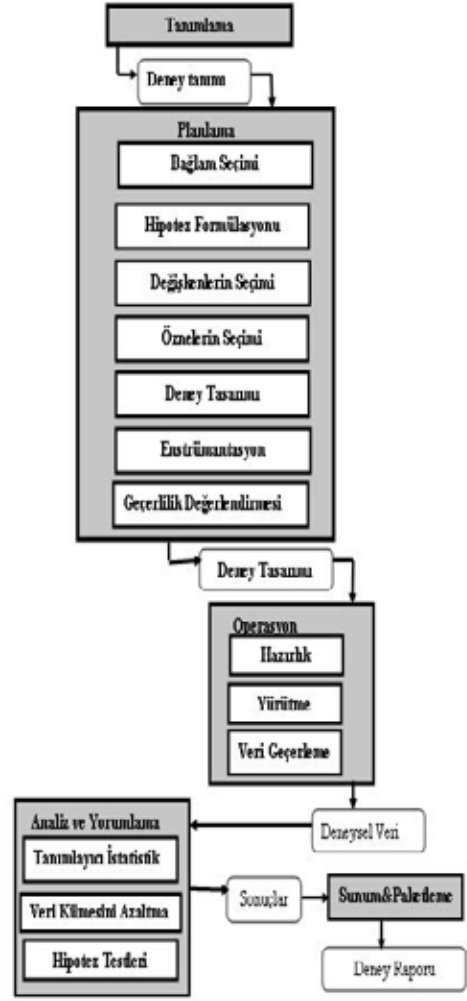
2. *İç tekrarlar (internal)*: Doğru çıkarımları yapmak üzere aynı deneyi araştırmacının birkaç defa yapmasıdır.

Yazılım Mühendisliğinde gerçekleştirilen deneylerin sonuçlarının önceki makalelerle kıyaslanması da kritik öneme sahip diğer bir husustur.

## 5. Deney Süreci

Yazılım Mühendisliğinde fikirlerin geçerliliğinin sağlanması için diğer mühendislik dallarında olduğu gibi 3 farklı grup tarafından deneyler gerçekleştirilmelidir [5]:

- İnnovatif fikir, ilk olarak bu fikri ortaya koyan **araştırmacı** tarafından laboratuvar ortamında deney tabi tutulmalıdır.
- Fikir, gerçek projeler üzerinde **“innovatif geliştirici”**, ya da “erken benimseyici” olarak ifade edebileceğimiz geliştiriciler tarafından test edilir. Fikir ilk kez gerçek bir proje üzerinde denendiği için, çeşitli riskler mevcuttur.
- Fikrin innovatif geliştiriciler tarafından yararı gözlemlenirse, **“rutin geliştiriciler”** ya da “pragmatik uygulayıcılar” olarak ifade edebileceğimiz 3. aşamada geliştiriciler tarafından projelerde uygulanması sağlanır.



Şekil 1: Deney Süreci

Yazılım Mühendisliği içerisinde bir yaklaşımın ya da yöntemin bu aşamalardan geçmeden, doğrudan gerçek bir proje üzerinde uygulanmaya çalışılması, fikri üreticiler ve rutin geliştiriciler açısından motivasyon düşüklüğüne ve zaman kaybına neden olacaktır. Bir ilacın ilk olarak test tüpünde, sonrasında gerçek bir organizmada ve devamında gönüllüler üzerinde test edilmeden doğrudan hasta üzerinde tatbik edilmesi hastalığın seyrini durdurılmaktan öte öldürücü etkiler göstereceği gibi, Yazılım

Mühendisliği içerisinde ortaya çıkan ve çok yaratıcı olduğu düşünülen fikirlerin laboratuvar ortamlarında ve sonrasında inovatif geliştiriciler tarafından kullanılmadan rutin geliştiriciler tarafından tatbik edilmesi, üzerinde çalışılan projenin başarısızlıkla sonlanmasına neden olur. Başarılı deneyler için uygulanabilecek bir deney süreci Şekil 1’de gösterilmektedir [5]. Bu süreç; Tanımlama, Planlama, Operasyon, Analiz&Yorumlama, Sunum&Paketleme alt süreçlerinden oluşmaktadır. Her alt süreçten çıkan ürünler, şekilde ayrıca gösterilmektedir.

### 5.1. Tanımlama

Tanımlama alt süreci; deneyin amaçlarını ortaya koymak üzere, deneyin planlanmasından ve yürütülmesinden önce önemli noktaların raporlandığı bir süreçtir. Deneyin neden gerçekleştirildiğini açıkça ifade eder. Bu tanımlama yapılırken; çalışmada kullanılan nesnelere, amaç, kalite odağı, perspektif ve bağlam bilgileri açıklanmalıdır [5]. *Çalışmanın nesnelere*; ürün, süreç, model, metrik ve teori olabilir. *Amaçlar*; izlemek, değerlendirmek, kestirmek, kontrol etmek ve değiştirmek olabilir. *Kalite odağı*; etkinlik, maliyet, güvenilirlik, bakım yapılabilirlik olabilir. *Perspektif*; geliştirici, proje yöneticisi, kullanıcı, müşteri, araştırmacı olabilir. *Bağlamda*; öznelere (deneyde görev alan insanlar) ve nesnelere ifade edilir [5]. Örneğin; yazılım kusur kestirimi problemi için Yapay Sinir Ağları (YSA) ve Bayes Ağlarının (BA) 2 ayrı teknik olarak kullanıldığı deneyi düşünelim. Böyle bir deney için aşağıda gerekli bilgileri sunulmalıdır:

**Çalışmanın nesnesi (object of study):** Çalışılan nesnelere Yapay Sinir Ağları ve Bayes ağlarıdır. **Amaç (purpose):** Amaç, yazılım kusur kestirimi problemi için 2 farklı yöntemin performansını değerlendirmektir. **Kalite odağı (quality focus):** Kalite odağı, kestirim performanslarının güvenilirliğidir (reliability). **Perspektif:** Perspektif, araştırmacının bakış açısidir. **Bağlam (context):** Deney NASA’nın

veri kümeleri üzerinde gerçekleştirilecektir. Bağlam bilgisi sayesinde, deneyin geçerlilik şartları ortaya konulmuş olacaktır.

### 5.2 Planlama

Planlama deneyin nasıl yürütüleceğinin açıklandığı alt süreçtir. Bu süreç içerisinde 7 alt süreç yer almaktadır.

#### 5.2.1 Bağlam Seçimi (Context Selection)

Deneyde görev alanların, deney sonuçlarının geliştirilmesinde önemli etkisi vardır. Öğrenci projeleri üzerinde denenilen bir yöntemin, sektördeki profesyoneller üzerinde etkisi farklı olabilmektedir. Deney planlama sürecinin ilk adımında, bu seçim gerçekleştirilmelidir. 4 boyutta deneylerin bağlamı ortaya konabilir [5]:

- *Çevrim dışı veya Çevrimiçi:* Öğrenci projeleri çevrim dışı, endüstriyel projeler çevrimiçi olarak adlandırılmaktadır.
- *Öğrenci veya Profesyonel*
- *Basit veya Gerçek problemler*
- *Spesifik veya Genel*

#### 5.2.2 Hipotez Formülasyonu

Hipotez, bir fikrin formalize edilmiş halidir. Bu aşamada; sıfır (null) hipotezi ve alternatif hipotezler yazılmalıdır [5]. Sıfır hipotezi, gözlemlerimizin tesadüfi olduğunu, herhangi bir örüntü taşımadığını ifade ederken; alternatif hipotez sıfır hipotezinin reddi durumunda geçerli olan hipotezdir. Örneğin; bir araştırmacı BA’ların kusur kestiriminde kullanılması durumunda elde edilen performansın, YSA’lardan daha yüksek olduğunu düşünüyor. Bu deney için yazılacak sıfır hipotez,  $H_0: P_{YSA} \geq P_{BA}$  iken alternatif hipotez  $H_1: P_{YSA} < P_{BA}$  şeklindedir. Hipotezlerin testi sırasında hatalar olabilmektedir. Tip1 hatası,  $H_0$  doğru iken  $H_0$  in red edilme olasılığıdır. Tip2 hatası ise,  $H_0$  hatalı iken  $H_0$  in reddedilmeme olasılığıdır. İstatistiksel testin gücünü (power) hesaplama formülü Eşitlik 3’te verilmektedir. Deneyi

yapan kişi, güç değerinin yüksek olduğu testi seçmelidir [5].

$$P(\text{tip1hata}) = P(\text{red } H_0 \mid H_0 \text{ doğru}) \quad (1)$$

$$P(\text{tip2hata}) = P(\text{kabul } H_0 \mid H_0 \text{ hata}) \quad (2)$$

$$\text{Güç} = 1 - P(\text{tip2hata}) \quad (3)$$

### 5.2.3 Değişkenlerin Seçimi

Planlama aşamasında, *bağımsız* ve *bağımlı* değişkenler belirlenmelidir. Deneyde kontrol edebildiğimiz ve değiştirebildiğimiz değişkenler *bağımsız değişkenlerdir*. Bu değişkenlerin seçimi, problem alanının iyi bilinmesini gerektirir. Seçilen bu değişkenlere ilişkin, sınır değerler ve testin yapılacağı seviyeler de ayrıca belirlenmelidir [5]. Bağımlı değişken genellikle, deneyler için bir tane olup deney sırasında kontrol edilemez. Örneğin; yazılım kusur kestirimi problemi için bağımsız değişkenler; çevrimsel karmaşıklık, kod satır sayısı olabilirken, bağımlı değişken kusur bilgisidir (hatalı ya da hatasız). Bu kusur bilgisi, bir modülün bir önceki yazılım sürümünün testi sırasında hata üretip üretmemesine bağlı olarak kayıt altına alınmıştır.

### 5.2.4 Öznelerin Seçimi

Öznelerin (subjects) seçimini, bir popülasyondan örneklem (sample) alınması olarak ifade edebiliriz [5]. Örnekleme, olasılıklı örnekleme ve olasılıklı olmayan örnekleme şeklinde iki grupta değerlendirilebilir. Olasılıklı örneklemede her öznenin seçilme olasılığı bilinirken, olasılıklı olmayan örneklemede bilinemez. Olasılıklı örnekleme, 5 gruba ayrılır [5]:

- *Basit rasgele örnekleme*: Özneler bir popülasyondan rasgele seçilir.
- *Sistemik örnekleme*: İlk özne popülasyondan rasgele seçilir, sonrasında bu öznenin n adet sonra gelen özne seçilir ve örneklem dizisinin sonuna kadar bu şekilde özneler eklenir.

- *Tabakalı rasgele örnekleme*: Popülasyon alt tabakalara ayrılarak her tabakadan rasgele örnekleme seçilir. Örneğin; lisans, yüksek lisans ve doktora öğrencilerinden oluşan popülasyonu eğitim durumuna göre 3 tabakaya ayırabiliriz.

- *Uygunluk örnekleme*: “En yakın ve en uygun kişiler seçilir” [5].

- *Kota örnekleme*: Tabakalı örnekleme benzerdir ancak öznelerin seçimi rasgele değildir.

### 5.2.5 Deney Tasarımı

DeneySEL verilerin yorumlanması kadar bu deneylerin tasarlanması (design) da başarıda önemli rolü vardır. Sıfır hipotezini red edebilmek için hipoteze bakarak ne tür istatistiksel yöntemlerin uygulanması gerektiği belirlenmeli ve bu yöntemlerdeki varsayımlara bağlı olarak deney tasarımı ortaya konulmalıdır [5]. Genel tasarım kuralları; deneydeki özneleri ve nesnelere rasgele seçme, bir faktörün sonuç üzerinde etkisini azaltmak için özneleri bloklara ayırma (blocking) ve her bloğa eşit sayıda özne atayarak dengeleme (balancing) işlemini gerçekleştirme olarak sıralanabilir. Deneylerdeki bağımsız değişkenler *faktör* olarak da adlandırılmaktadır ve bir faktörün sayısal değerine *işlem* (treatment) adı verilir [5]. Sıkça kullanılan tasarım tipleri aşağıda açıklanmaktadır:

1-) *Tek faktörlü ve 2 işlemli*: Bu deneylerde, 2 işlem kıyaslanmaktadır. Örneğin; yeni bir test yöntemi *A* eski bir test yöntemiyle *B* kıyaslanacaksa, test yöntemi *faktör* olarak adlandırılırken *A* ve *B* test yöntemleri *işlem* olarak ifade edilir. Bağımsız değişken ise yöntemlerin tespit ettiği kusur sayısı olabilir. İki gruba ayrılır [5].

a) *Tamamen rasgele tasarım*: Özneler 2 farklı yöntemle rasgele atanırken özne

sayıları aynı tutulursa dengeli bir tasarım ortaya çıkar. Analiz için; t-test ya da Mann-Whitney kullanılır [5].

*b) Eşli kıyaslama tasarımı:* Her özne 2 işlemi aynı nesne üzerinde uygular, ancak uygulama sırası rasgele olarak belirlenir. İlk ve ikinci yöntemle uygulamaya başlayan özne sayısı eşit ise dengeli olduğu söylenir. Analiz için; eşli t-test, işaret testi, Wilcoxon kullanılabilir [5].

2-) *Tek faktörlü ve 2 işlemden fazlalı:* *A, B, C* test yöntemlerinin kıyaslanmasının istendiği bir deney örnek olarak verilebilir. Tamamen rasgele tasarım bu kez, 3 işlem ile gerçekleştirilirken analiz için ANOVA ya da Kruskal-Wallis kullanılabilir. Rasgele tam blok tasarımı adı verilen bu yöntem, eşli kıyaslama tasarımının 3 işleme uyarlanmış halidir ve analiz ANOVA ya da Kruskal-Wallis ile gerçekleştirilir.

3-) *İki faktörlü:* Faktör sayısının 2'ye çıkması ile birlikte hipotez sayısı da 1'den 3'e çıkmaktadır. *A* faktörü üzerinde *i* işleminin etkisi, *B* faktörü üzerine *j* işleminin etkisi ve bu 2 etki arasındaki etkileşimin etkisi olarak 3 hipotez sıralanabilir.

*2\*2 faktöriyel tasarımı:* 2 işlem, 2 faktör vardır. Örneğin; isterler belgesi *B* faktörü, tasarım yaklaşımı *A* faktörü olsun ve tasarım belgelerinin anlaşılabilirliği üzerine bir deney yapılmak istensin. İyi ve kötü şeklinde isterler belgesi için 2 işlem değeri; yapısal ve nesneye yönelik tasarım şeklinde 2 işlem değeri kullanılabilir. Özneler bu işlem kombinasyonlarına rasgele atanır. Analiz için; ANOVA kullanılabilir [5].

4-) *İki faktörden fazlalı:* Bu tip tasarım faktöriyel tasarım olarak adlandırılmaktadır.  $2^k$  faktöriyel tasarım,  $2^k$  kesirli faktöriyel tasarım,  $2^k$  kesirli faktöriyel tasarımın bir yarım

kesirli faktöriyel tasarım gibi yöntemler vardır. Bu kapsamda, Montgomery'nin kitabı incelenebilir [7].

Bu gruplarda da görüldüğü üzere, başarılı bir test için doğru test tasarımının seçilmesi ve beraberinde uygun analiz yönteminin belirlenmesi gerekmektedir.

### 5.2.6 Enstrümantasyon

Deneylerin planlanması sırasında, deneyde kullanılacak enstrümanlar belirlenmelidir. Bu enstrümanlar; nesnelere, rehberler ve ölçüm enstrümanları olmak üzere 3 tiptir [5]. Nesnelere; program kodu, tasarım veya test belgesi olabilir. Rehberler sayesinde, katılımcıların deneye adaptasyonu kolaylaşmakta, üzerinde çalışılan teknikler hakkında kolayca fikir elde edilebilmektedir. Ölçüm enstrümanlarına örnek olarak, hazırlanan formlar ya da mülakat soruları verilebilir [5].

### 5.2.7 Geçerlilik Değerlendirmesi

Deney sonuçlarının genelleştirilmesi sırasında oluşabilecek tehlikelerin, planlamada belirlenmesi gerekmektedir. Cook ve Campbell [2], sonuçların geçerliliği noktasında tehlike yaratabilecek 4 tip tehlikeyi ortaya koymuştur [5]:

1. *Sonuç geçerliliği (conclusion validity):* İstatistiksel sonuç geçerliliği olarak da bilinmektedir. İstatistiksel testlerin bazı varsayımlarının ihlal edilmesi, ölçümlerin güvenilir olmaması, önceki bölümde verilen istatistiksel güç değerinin düşük olması, deney sırasında gürültülerin oluşması ve özel bir sonuca ulaşmayı düşünerek deneyi gerçekleştirme (avlama-fishing) sonuç geçerliliği için temel tehlikelerdendir.

2. *İç geçerlilik (internal validity):* Tekli grup, çoklu grup ve sosyal tehlikeler olarak 3 grupta değerlendirilebilir. Tekli gruplar için; deneyin gerçekleştirildiği günlerin özel bir anlamının olup olmadığı, zamanla

motivasyonun kaybedilmesi, testin tekrar tekrar yapılması nedeni ile elde edilen yetenek, enstrümantasyonların yetersizliği tehlike yaratabilir. Çoklu gruplarda yeteneğe bağlı olarak çalışılan teknik üzerinde farklı deneyimler elde edilebilir. Sosyal tehlikelere örnek olarak, iki grup için yeni tekniği kullanacak kişilerin daha fazla motive olma durumu verilebilir.

3. *Yapı geçerliliği (construct validity)*: Tasarım tehlikeleri ve sosyal tehlikeler olarak 2 grupta değerlendirebiliriz. Tasarım tehlikeleri için; iki yöntem kıyaslanırken birbirlerine göre iyi olma durumunun ne anlama geldiğinin açıkça tanımlanmaması, tek bir nesne ile deney yapılması durumunda oluşan tek yöndeki eğilim durumu örnek verilebilir. Sosyal tehlikeler için ise; deneyin sonucunun önceden tahmin edilmesi ile bir yönde eğilimin oluşması, değerlendirmeden korkan kişilerin kestirimde doğru sapmaları sunmaması örnekleri verilebilir.

4. *Dış geçerlilik (external validity)*: “Deneyimizin endüstriyel uygulamalar için genelleştirilebilmesini sınırlandıran durumlardır” [5]. İnceleme (inspection) deneylerinde programcılarla çalışıp testçileri göz ardı etmek, endüstride güncel araçlar kullanılırken testlerde eski araçları kullanmak, önemli bir olayın ardından mülakatları gerçekleştirmek dış geçerliliği etkileyen durumlardır [5].

### **5.3 Operasyon**

Deneylerde görev alan öznel deneyi ciddiye almıyorsa sonuçların geçerliliğinden söz edilemez [5]. Operasyon süreci; hazırlık, yürütme ve veri geçerliliği aşamalarından oluşmaktadır. Hazırlık aşamasında; katılımcıların bilgilendirilmesi sağlanmalı, katılımcılar yanıltılmamalı ve deneylerden önce tüm enstrümanlar hazır olmalıdır [5]. Veriler elle, otomatik araçlarla ya da mülakatlarla elde edilebilir. Veri geçerli-

liği için, aykırı değer analizi (outlier analysis) yapılabilir ve elde edilen verilerin kontrolü gerçekleştirilir.

### **5.4 Analiz ve Yorumlama**

Deney gerçekleştirildikten sonra, verilerin yorumlanması gerekmektedir. Bu aşamada; aşağıdaki alt süreçler gerçekleştirilir.

1. *Tanımlayıcı istatistik*: Veri kümesinin dağılımı incelenerek görselleştirilir. Verinin tipine (nominal, ordinal, aralık, oran) bağlı olarak verinin merkezi eğilimi (central tendency); mod, medyan, ortalama, geometrik ortalama gibi ölçümlerle hesaplanır. Merkezi eğilim, verinin orta kısmını (middle) göstermektedir. Verinin dağılımı (dispersion) ise yine veri tipine göre; sıklık, varyasyon aralığı, standart sapma veya varyasyon katsayısı gibi ölçümlerle hesaplanır [5]. Görselleştirme amaçlı da; serpm çizimi (scatter plot), kutu çizimi (box plot) ya da histogramlardan yararlanılabilir.

2. *Veri kümesini azaltma*: Aykırı değerler, yeniden görülme durumları yoksa veri kümesinden çıkartılmalıdır. Ayrıca, gereksiz bilgi taşıyan veriler varsa bunlar da temel bileşen analizi ve faktör analizi yöntemleriyle veri kümesinden çıkartılabilir [5].

3. *Hipotez testi*: Bu testleri parametrik ve parametrik olmayan testler şeklinde 2 grupta ele alabiliriz. Parametrik testlerin kullanılabilmesi için, en azından aralık ölçeğinde parametreler ölçülebilmelidir. Parametrik testler daha az sayıda veriye ihtiyaç duyar. T-test, F-test, Eşli T-test, ANOVA parametrik testlerdendir. Chi-2, Binom testi, Mann-Whitney, Wilcoxon, işaret testi, Kruskal-Wallis parametrik olmayan test grubuna girmektedir. Hipotez testleri gerçekleştirildikten sonra hipotezlerin red ya da kabul durumu belirlenerek deney yorumlanmaktadır.

## 5.5 Sunum ve Paketleme

Deneyler gerçekleştirildikten sonra; bilgiler bildiri halinde konferanslarda sunulabilir veya şirketler için karar verme amaçlı olarak bir rapor düzenlenebilir [5]. Bu kapsamda bir makale aşağıdaki şekilde hazırlanabilir:

### 1. Giriş

#### 1.1. Problem Tanımı

- § Çalışmanın nesnesi
- § Amaç
- § Perspektif
- § Kalite odağı
- § Bağlam

#### 1.2. Tanımın Özeti

### 2. Planlama

- 2.1 Bağlam Seçimi
- 2.2 Hipotez Formülasyonu
- 2.3 Değişkenlerin Seçimi
- 2.4 Öznelerin Seçimi
- 2.5 Deneysel Tasarım
- 2.6 Enstrümantasyon
- 2.7 Geçerlilik doğrulaması

### 3. Operasyon

- 3.1 Hazırlık
- 3.2 Yürütme
- 3.3 Veri geçerliliği

### 4. Analiz ve Yorumlama

- 4.1 Tanımlayıcı istatistik
- 4.2 Veri azaltma
- 4.3 Hipotez testi

### 5. Tartışma ve Sonuçlar

### 6. Ek

## 6. Sonuç ve Gelecek Çalışmalar

Bu çalışmada, Yazılım Mühendisliği içerisindeki deneysel çalışmaların önemi vurgulanmış, yetersiz sayıda deney yapıldığı ifade edilerek deneylerde kullanılacak bir süreç ve deneyleri raporlarken kullanılacak format sunulmuştur. Her yıl farklı bir ülkede gerçekleştirilen “Ürün Odaklı Yazılım Süreci İyileştirme” (Product Focused Software Pro-

cess Improvement-PROFES) konferansında, bu yıl en iyi bildiri ödülünü kazanan çalışma, bu raporlama formatı ile hazırlanmıştır. Türkiye’deki araştırmacılar, deneylere daha fazla önem vererek yeni teoriler ve yaklaşımlar üretmenin yanı sıra varolan teorilerin ya da bilgilerin geçerliliğinin sağlanması üzerine de çalışma yapmalıdır. Bu çalışmada verilen deney raporlama formatı sayesinde, ülkemizde gerçekleştirilecek olan deneylerin sonuçlarının yeniden üretilebilirliği artacak ve uluslararası konferanslarda, dergilerde yayımlanması kolaylaşacaktır. Bu aşamadan sonra; gerçekleştireceğimiz tüm deneysel Yazılım Mühendisliği çalışmalarında bu süreci uygulamayı ve raporlama formatını kullanmayı planlıyoruz. Ayrıca; ülkemizde bu kapsamda gerçekleştirilecek çalışmaların hem özel sektör hem de üniversitemiz açısından önemli yarar sağlayacağı düşünülmektedir. Aksi halde firmalarının süreç iyileştirme çalışmaları, nesnel bilgiler yerine öznel düşüncelerle sonlanabilir; üniversitemizde gerçekleştirilen deneylerin açıklandığı bildiriler ya da makaleler ise uluslararası konferanslarda ve dergilerde yayımlanmayabilir.

## Teşekkür

Bu bildiri, TÜBİTAK tarafından desteklenen 107E213 numaralı Araştırma Projesinin ön hazırlık aşamasında elde edilen bilgiler kullanılarak hazırlanmıştır. Yayındaki hiçbir görüş, tespit ve kanaat, TÜBİTAK’ın resmi görüşü değildir.

## 7. Kaynaklar

- [1] Brooks, F. P., “No Silver Bullet: Essence and Accidents of Software Engineering”, *IEEE Computer*, Cilt:20, Sayı:4, 1987, 10-19.
- [2] Cook, T. D., Campbell, D. T., *Quasi-Experimentation – Design and Analysis Issues for Field Settings*, Houghton Mifflin Company, 1979.

- [3] Glass, R., “The Software Research Crisis”, *IEEE Software*, Cilt:11, Sayı:6, 1994, 42-47.
- [4] Hatton, L., “Does OO Really Match the Way We Think?”, *IEEE Software*, Cilt:15, Sayı:3, 1998, 46-54.
- [5] Juristo, N., ve Moreno, A.: *Basics of Software Engineering Experimentation*, Kluwer Academic Publishers, 2001.
- [6] Maibaum, T., “What We Teach Software Engineers in the University: Do We Take Engineering Seriously?”, Proceedings of the ESEC/FSE, 40-50, 1997, Zurich, Switzerland,.
- [7] Montgomery, D. C., *Design and Analysis of Experiments*, 4th Edition, John Wiley&Sons, 1997.
- [8] Naur, N., ve Raudell, B., *Software Engineering. Report on a Conference Sponsored by the NATO Science Committee*, 1968 Ekim 7-11; Garmish. Brussels: Scientific Affairs Division NATO, 1969.
- [9] Tichy, W. F., Lukowicz, P., Prechelt, L., ve Heinz, E. A., “Experimental Evaluation in Computer Science: A Quantitative Study”, *Journal of Systems and Software*, Cilt: 28, Sayı:1, 1995, 9-18.
- [10] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., Wesslén, A., *Experimentation in Software Engineering: an Introduction*, Kluwer Academic Publishers, Norwell, MA, 2000.
- [11] Zelkowitz, M. V., ve Wallace, D., “Experimental Models for Validating Computer Technology”, *IEEE Computer*, Cilt: 31, Sayı: 5, 1998, 23-31.