PROMOL ile Yazılım Proje Yönetimi Modellenmesi

Software Project Management Modeling with PROMOL

Kadir Alpaslan Demir
Bilgisayar Bilimleri Bölümü
Naval Postgraduate School,USA
kdemir@nps.edu

Abdülkerim Ergüner
Bilişim Bilimleri Bölümü
Naval Postgraduate School,USA
aerguner@nps.edu

Özet

Yeni yapılan anket sonuçlarına göre, yazılım projeleri hala önemli ölçüde başarısız olmaktadır. Projeler teknik hususlardan ziyade idari hususlardan dolayı başarısızlıkla sonuçlanmaktadır. Dolayısıyla, yazılım proje yönetim kalitesinin arttırılması, projelerin başarıyla sonuçlanma ihtimalini artıracaktır. Kaliteyi arttırma yöntemlerinden bir tanesi, proje yönetim araçlarının etkin bir şekilde kullanılmasıdır. Bu bildirimizde, formal ve görsel bir yazılım proje yönetim modelleme dili olan PROMOL'u tanıtacağız. Görsel, formal ve birden çok faktörü aynı anda ele alması ile PROMOL kendini diğer proje yönetim araçları arasında farkettirmektedir. Ayrıca, modelleme dilinin kullanımını göstermek amacıyla basit bir örnek de bildiride sunulmuştur.

Abstract

According to the recent surveys, software projects are still failing significantly. The projects are failing due to management related issues rather than technical issues. Therefore, improving the quality of software project management increases the odds of success. One of the ways to increase the quality is the effective use of project management tools. In this paper, we introduce PROMOL, a formal and visual software project management modeling lanaguage. PROMOL distinguishes itself from other project management tools with being visual, formal and able to address various aspects of project management at the same time. We also provide a simple example to illustrate the use of modeling language.

1. Giriş

Yazılım projelerinin başarı seviyeleri diğer endüstrilerdeki projelerle karşılaştırıldığında hala oldukça düşük bir düzeydedir. Çeşitli zamanlarda yapılan bilimsel anketlerde, yazılım projelerinin başarı ile tamamlanma yüzdelerine ait değişik ve zaman zaman çelişkili rakamlar bildirilmiştir [1,2,3]. Ancak en iyimser rakamlar bile bu konuda kat edilmesi gereken çok mesafe olduğunu göstermektedir. En güncel rakamlara göre [3] yazılım projeleri %26-%34 oranında başarısızlıkla sonuçlanmaktadır.

Çeşitli araştırmalarla, yazılım projelerinin başarı veya başarısızlıkla sonuçlanmasına etki eden faktörler bildirilmiştir [2,3,4,5]. Bu faktörlerin teknik faktörler olmaktan ziyade yazılım projelerinde karşılaşılan idari güçlüklere ait faktörler olduğu belirtilmiştir [6,7,8]. Weinberg'in anektodal anlatımı ile yazılım proje başarısızlıklarının üç sebebi "insanlar, insanlar, insanlar"'dır.[8] Dolayısıyla, yazılım projelerinin yönetilmesinde kalitenin arttırılması, projelerin başarı ile sonuçlanmasına olanak veren en önemli hususlardan biri olarak karşımıza çıkmaktadır. Son yıllarda hızla artan ve çeşitlendirilen yazılım mühendisliği teknik hususlarına ait çalışmalara rağmen, projelerin başarılarında gözle görülür bir iyileşme görülmemiştir. Bu da, yazılım proje yönetimi alanındaki araştırmaların önemini ortaya çıkarmaktadır. Yazılımların ve yazılım projelerinin yönetimlerinin kendine has karakteristikleri, Brooks'un yazılım mühendisliği alanında artık bir klasik haline gelen kitabında ve bir makalesinde anlatılmaktadır [9].

Jones, başarılı ve başarısız yazılım projelerinde kullanılan yönetim araçlarını incelemiştir [10]. Başarılı projeler çeşitli yazılım proje yönetim araçlarını etkinlikle kullanırken, başarısız projeler genellikle bu araçları kullanmamaktadırlar. Dolasıyla yazılım projelerinin yönetiminin kalitesinde yönetim araçlarının etkisi oldukça fazladır.

Bu bildirimizde, yazılım projelerinin yönetiminin görsel ve formal olarak olarak modellemesine olanak veren PROMOL modelleme dili tanıtılacaktır. PROMOL'un en önemli özelliklerinden biri proje yönetimini ilgilendiren önemli bir çok hususun ilişkilerini aynı model üzerinde gösterebilmesidir. Bu özelliği, PROMOL'u diğer yazılım proje yönetim araçlarından ayıran en önemli kabiliyetlerden biridir. Literatür taramamıza göre proje yönetiminin bu şekilde modellenmesine olanak sağlayan ilk araçlardan biridir. Ayrıca, görselliği, basitliği ve birden çok faktörü aynı model üzerinde birleştirebilmesi ile proje yöneticileri tarafından kolayca kullanılabilir olmakla beraber, formal özelliği ile yazılım projelerinin yönetiminin ve kalitesinin araştırmacılar tarafından matematiksel

olarak incelenebilmesine, başarı ve başarısızlık faktörlerinin araştırılabilmesine olanak sağlamaktadır. PROMOL her türlü proje geliştirme yaşam döngüsü ile kullanılabilir. Bildirimizde, modelleme dilinin kullanılmasını tanıtmak amacıyla PROMOL ile modellediğimiz projelerden birinin küçük bir bölümünü basitçe anlatılacaktır.

2. Literatür Taraması

Proje Yönetim Enstitüsü (PMI-Project Management Institute) dünyada birçok araştırmacı, proje yöneticisi ve proje yönetiminin çeşitli kademelerinde çalışan insanların oluşturduğu profesyonel bir organizasyondur. Amaçlarından bir tanesi proje yönetimi konusunda kaynak oluşturacak bir proje yönetim literatürünün oluşturulmasıdır. PMI'nın çeşitli revizyonlardan sonra en son 2004 yılında yayınladığı proje yönetim klavuz kitabı (Project Management Body of Knowledge-PMBOK) [11] proje yönetimi konusundaki en temel kaynaktır. Klavuzda projelerde çeşitli amaçlar için kullanılabilecek degişik proje yönetim araçlarından en çok bilinenleri tanıtılmaktadır.

Bütünleşik Kapasite Olgunluk Modeli (Capability Maturity Model Integration-CMMI) [12] Software Engineering Institute (SEI) tarafından uzun süren araştırmaların bir ürünü olarak geliştirilmiştir. CMMI, organizasyonları kapasitelerine göre 5 seviyeye ayırır. Üst seviyelere ulaşmak için projelerde CMMI dökümanında tanımlanan genel ve özel hedeflerin gerçekleştirilmesi gerekir. CMMI dökümanında proje yönetimi ile ilgili birden çok alan ayrıntılı olarak anlatılmış olup, ilişik bölümlerde proje yönetiminde kullanılması tavsiye edilen çeşitli prensipler, metodlar ve araçlar belirtilmiştir.

Yazılım Mühendisliği Bilgi Klavuzu (Software Engineering Body of Knowledge-SWEBOK) [13] IEEE tarafından 2004 yılında derlenmiş olup, yazılım mühendisliği alanında taban oluşturabilecek ilk klavuzdur. 10 adet bilgi alanı (knowledge area-KA) tanımlanmıştır. Bu bilgi alanlarından bir tanesi de yazılım mühendisliği yönetimidir. Klavuzun bu bölümünde yazılım projeleri yönetiminde kullanılabilecek çeşitli araçlardan bahsedilmektedir.

Yukarıda bahsedilen üç döküman yazılım mühendisliği ve proje yönetimindeki temel dökümanlardandır. Hernekadar bu dökümanların temel amaçları çeşitli araçları tavsiye etmek olmasa da, günümüzde yazılım proje yönetim alanının geldiği seviyeyi göstermeleri açısından burada bahsedilmesinin önemi vardır. Gerek yazılım mühendisliği gerek de proje yönetimi alanlarında yaptığımız literatür taramasında özellikle yazılım projeleri için önerilmiş herhangi bir proje yönetimi modelleme dili ile karşılaşmadık.

Pinto, herhangi bir projeye kaynak aktarmadan ve hatta daha proje başlamadan önce projelerin ticari, teknik, finansal, çevresel ve diğer yönlerinin modellenmesinin önemini vurgulamaktadır.[14] Jaafari, projeler için çok basit ve yüksek seviye bir proje modelini sunmaktadır. Ayrıca, bu modeller için ideal gereksinimleri de listelemektedir. Jaafari'ye göre projelerin karmaşık bir çok yönünü istenen seviyede modelleyebilmek için kat edilmesi gereken çok yol vardır [15].

Yazılım projelerinin yönetiminde kullanılan çeşitli metod ve araçlardan en çok tanınanlarından bazıları PERT, CPM, GANTT, work breakdown structure (WBS), karar ağacı analizi (decision tree analysis), ağ diyagramları (network diagrams)'dır. Bu proje yönetim araçlarının ve metodlarının çoğu uzun yıllar evvel tanıtılmış olup hala etkinlikle kullanılmaktadırlar. Yazılım projeleri yönetimi alanında yeni metod ve araçların geliştirilmesine yönelik yakın dönemlere ait çalışmalar oldukça kısıtlı olup genellikle yazılım projelerinde maliyet ve emek tahmini [16] ve risk yönetimi üzerinde yoğunlaşmıştır.

3. PROMOL: Görsel ve Formal bir Proje Yönetim Modelleme Dili

Proje yönetimi çok yönlü ve kompleks bir süreçtir. Bunun üzerine Brooks'un tespit ettiği yazılım karakteristikleri [9] de eklenince yazılım projelerinin yönetimi üstesinden gelinmesi gereken güç bir süreç olmaktadır. Projenin yönetiminin modellenmesi kompleksliğinin azaltılmasına, dolayısıyla bu sürecin başarı ile sonuçlanmasına yardımcı olacaktır. PROMOL görsel ve formal bir proje yönetim modelleme dilidir. Yazılım projeleri üzerindeki çalısmalarımızın sonucu olmasına rağmen diğer proje tiplerinde de etkin olarak kullanılabilir.

PROMOL'un özünde proje yönetimi ile ilgili iki önemli kavram bulunmaktadır. Bunlar aktivite (activity) ve varlık (entity) kavramlarıdır. PROMOL'de aktivite ve varlıkların tanımları aşağıda verilmiştir:

Bir aktivite, tanımlanmış bir süreç, fonksiyon, veya iş olup belirli bir süre ile kısıtlıdır. Bir varlık, kendi başına ayrıca varolan herhangi bir şey olup, maddi olarak varolmak zorunda değildir.

Proje yönetiminde kullanılan kavramların hepsini bu çok geniş tanımlanmış iki kavramla kategorilendirmek mümkündür. Aktivitelere örnek olarak problem tanımlanması, yazılım dizayn aktiviteleri, yazılım mimari geliştirilmesi, personel temini ve görevlendirilmesi, test faaliyetleri, prototip geliştirme faaliyetleri, çeşitli proje toplantıları vs. verilebilir. Proje yemek ve partileri gibi personel için sosyal faaliyetler ile eğitim faaliyetleri de eğer projenin kalitesine etki edeceği düşünülüyorsa aktivite olarak kategorilendirilebilir. Varlıklara örnek olarak proje tanım ve problem dökümanı, proje planı, teknik dizayn, yazılım mimarisi, test kriterleri, test verileri, proje personeli, proje ile ilişkisi olan her türlü kişi ve

kurumlar vs. verilebilir. Eğer modellemeye etkileri olduğu düşünülüyorsa ve etkileri herhangi bir şekilde tahmin veya hesap edilebiliyorsa iletişim, takım çalışması, liderlik gibi kavramlar da varlık olarak değerlendirilip modellere eklenebilir. PROMOL modelleme dili bu iki kavram üzerinde tanımlanan matematiksel ilişkiler bütününden oluşmaktadır.

3.1. CREATE (YARAT) İlişkisi

Projelerdeki aktiviteler, aktivite sonucu olarak bir varlık üretiyorsa, "CREATE" (YARAT) ilişkisi kullanılır. İlişkinin matematiksel tanımı aşağıda verilmiştir:

$$e_x = a_y(\ )$$

Genellikle projelerde aktiviteler bir veya birden çok varlığı girdi olarak alıp başka bir varlığı çıktı olarak verirler. Ancak bazı durumlarda aktivitenin girdilerinin belirtilmesi proje yönetimi ve yöneticisi açısından önemli değilse, bu ilişkilerin belirtilmesinde de YARAT ilişkisi kullanılabilir.

Örnek olarak, bir varlık olarak degerlendirebileceğimiz proje personeli, bir işe alma faaliyeti sonucu ortaya çıkar. Bu ilişkiyi PROMOL'de aşağıdaki şekilde belirtebiliriz:

proje_personeli=işe_alma()

3.2. TRANSFORM (DÖNÜŞTÜR) İlişkisi

Projelerde birçok aktivitenin bir veya birden çok girdileri çıktılara dönüştürmesi, "TRANSFORM" (DÖNÜŞTÜR) ilişkisi ile modellenebilir. Bu ilişki PROMOL'de tanımlanmış en temel ilişkilerden biridir. Formal tanımı:

$$e_y = a_x(e_1, e_2, e_3, ..., e_n)$$

Örnek vermek gerekirse, bir varlık olan gereksinim dökümanı, proje ile çözümlenecek olan iş probleminin gereksinim incelemesi aktivitesine girdi olur. Bu ilişkiyi aşağıdaki şekilde modelleyebiliriz:

gereksinim_dökümanı=gereksinim_incelemesi(iş_problemi)

3.3. DIVIDE (BÖL) İlişkisi

Projelerde birçok aktivite ve varlığın daha alt parçalarına bölünerek ayrıntılı olarak modellenmesi ihtiyacı için PROMOL'de "DIVIDE" (BÖL) ilişkisi kullanılır. Bu ilişki ile proje yönetim modelleri hiyerarşik bir yapıya kavuşur. Böl ilişkisi formal olarak aşağıdaki şekilde tanımlanmıştır:

$$DIVIDE(a(\ )) = \{a_1(\ ), a_2(\ ), a_3(\ ), ..., a_m(\ )\}$$
$$DIVIDE(e) = \{e_1, e_2, e_3, ..., e_n\}$$

Mesela, bir yazılım projesinde test faaliyeti, modül ve entegrasyon test faaliyetleri olarak daha alt aktivitelerine bölünebilir:

DIVIDE(test())={modül_testleri(), entegrasyon_testi()}

Bir yazılım dizaynı, üst seviye ve alt seviye dizayn olarak ikiye bölünebilir:

DIVIDE(dizayn)={üst_seviye_dizayn, alt_seviye_dizayn}

3.4. AGGREGATE (BİRLEŞTİR) İlişkisi

Projelerde birden çok aktivite veya varlığı birleştirerek modellemek için "AGGREGATE" (BİRLEŞTİR) ilişkisi kullanılır. Böylece elimizdeki detaylı ve alt seviye modelleri çeşitli amaçlar için üst seviye ve daha basit modellere dönüştürme imkanı vardır. Birleştir ilişkisi formal olarak aşağıdaki şekilde tanımlanmıştır:

$$a(\ ) = AGGREGATE(a_1(\ ), a_2(\ ), a_3(\ ), ..., a_m(\ ))$$
$$e = AGGREGATE(e_1, e_2, e_3, ..., e_n)$$

3.5. NEXT (SONRAKİ) İlişkisi

Yazılım ve diğer alanlardaki projelerde, proje yöneticisinin veya yöneticilerinin en temel görevlerinden biri de aktivitelerin veya varlıkların arasındaki öncelik ve sıra ilişkilerini belirlemektir. Çeşitli aktivitelere başlanması için aktivite öncesinde diğer başka aktivetelerin bitirilmesine gerek olabilir. Aynı şekilde proje ile ilgili çeşitli varlıkların ortaya çıkması için diğer başka varlıkların daha evvelden elde edilmesine gerek olabilir. Mesela geliştirilen yazılımın entegrasyon testlerinin yapılabilmesi için tüm modellerin teste hazır halde bitirilmesi gerekebilir. İyi bir yazılım dizayni için çeşitli yazılım mimari alternatiflerinin incelenip o yazılım için uygun mimarinin seçilmesi ya da geliştirilmiş olmasına ihtiyaç vardır. Bu tür ilişkilerin modellenebilmesi için PROMOL'de "NEXT" (SONRAKİ) ilişkisi tanımlanmıştır. Sonraki ilişkisinin sembolü sağa oktur.

$$a_1(\ ) \rightarrow a_2(\ )$$
$$e_1 \rightarrow e_2$$

Mesela, yazılım projelerinde dizayn faaliyetlerine geçmeden önce etkin bir gereksinim mühendisliği faaliyeti icra edilmelidir. Bu ilişkiyi aşağıdaki şekilde modelleyebiliriz:

gereksinim_incelemesi()→yazılım_dizaynı()

3.6. PREVIOUS (ÖNCEKİ) İlişkisi

"PREVIOUS" (ÖNCEKİ) ilişkisi sonraki ilişkisine çok benzerdir. Tek farkı sonraki faaliyetler yerine önceki faaliyetler belirtilir. İlişkinin formal sembolü sola ok olup aşağıdaki şekilde tanımlanmıştır:

$$a_2(\ ) \leftarrow a_1(\ )$$
$$e_2 \leftarrow e_1$$

3.7. REQUIRE (GEREK) İlişkisi

Yazılım projelerinde çeşitli faaliyetlerin icrası için diğer bazı faaliyetlere ya da varlıklara gerek vardır. Bu ilişki PROMOL'de "REQUIRE" (GEREK) ile modellenir.

$$REQUIRE(a()) = \{a_1(), a_2(), a_3(), ..., a_m(), e_1, e_2, e_3, ..., e_n\}$$

$$REQUIRE(e) = \{a_1(), a_2(), a_3(), ..., a_m(), e_1, e_2, e_3, ..., e_n\}$$

3.8. DECISION (KARAR) İlişkisi

Proje yöneticilerinin önemli görevlerinden biri de projelerde çeşitli aktivitelerin sonucunda oluşan varlıkların durumuna göre hangi aktivitelerin icra edileceğine dair kararlar vermektir. Bu karar verme faaliyeti "DECISION" (KARAR) ilişkisi ile modellenir. Karar ilişkisinde girdilerden bir tanesi karara kıstas teşkil edecek kriter ya da kriterlerdir. Bu kriter ya da kriterler bütünü de aslında bir varlıktır. Diğer girdiler ise kararın oluşmasına etki eden varlıklardır. Karar ilişkisinin çıktısı ise karar ve alınan o kararın devamında işleme konulacak aktivitelerin oluşturduğu bir kümedir.

$$DECISION(e_1, e_2, e_3, ..., e_m) = \begin{Bmatrix} \{decision_1, a_1(\ )\}, \\ \{decision_2, a_2(\ )\}, \\ \{decision_3, a_3(\ )\}, \\ ..., \\ \{decision_n, a_n(\ )\} \end{Bmatrix}$$

3.9. EXIST (VAR) İlişkisi

PROMOL'de proje yönetiminin matematiksel modellerinin araştırmacılar tarafından formal olarak sorgulanması için "EXIST" (VAR) ilişkisi tanımlanmıştır. Var ilişkisinin girdisi herhangi bir aktivite, varlık ya da modelin bir parçası olabilir. Var ilişkisi girdide sorgulananın modelde olup olmadığını kontrol eder. Yapısı aşağıdaki şekildedir:

$$EXIST(a_1(\ )) = true / false$$
$$EXIST(e_1) = true / false$$
$$EXIST(a_1(\ ) \rightarrow a_2(\ )) = true / false$$

Mesela gereksinimleri proje başlangıcında tam olarak bilinmeyen yazılım projelerinde yazılımın çeşitli yönlerini ortaya çıkaran prototiplerin geliştirilmesi projenin başarıya ulaşmasındaki önemli etkenlerden biridir. PROMOL ile geliştirilmiş formal bir modelde bu önemli husus aşağıdaki şekilde sorgulanabilir:

EXIST(prototip)=True / False

Var ilişkisinin PROMOL'de tanımlanmasının en önemli sebebi, projeler üzerinde araştırma yapan uzmanların projelerin çeşitli özelliklerini formal olarak inceleyebilmesine olanak sağlamaktır.

3.10. Hazır Tanımlanmış Kavramlar

PROMOL modelleme dilinde hazır olarak tanımlanmış kavramlar mevcuttur. Bunlardan önemlileri START (BAŞLA) ve END (BİTİR) tanımlarıdır. Bu iki özel tanım büyük harflerle yazılır ve bir proje modelinin başlangıcını ve bitimini simgeler. Küçük harflerle (start-başla, end-bitir) yazıldıkları zaman ise bu iki hazır tanımlanmış kavram böl ilişkisi ile detaylandırılan bir aktivite veya varlığın kapsamını belirtir.

3.11. Grafik Tanımları

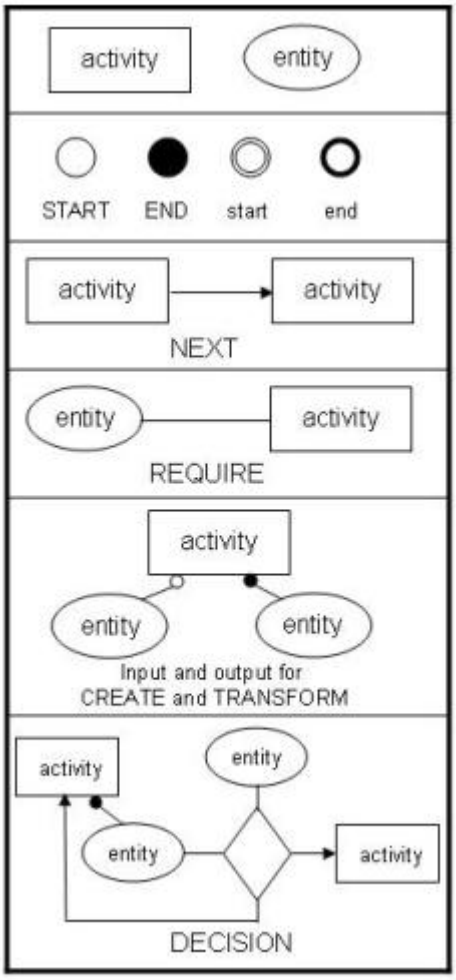
PROMOL'de tanımlanan kavramların ve ilişkilerin sayısının az olmasına özen gösterilmiştir. Miller kanununa göre kısa süreli hafıza 7 ± 2 adet bilgi parçası alır. Özel durumları da içerecek şekilde daha birçok kavramın ve ilişkinin tanımlanabilecek olmasına rağmen, önemli olan modelleme dilinin basit, kullanılabilir ve optimal olduğu gibi gerçek hayattaki birçok projede uygulanabilir olmasıdır. Dolayısıyla PROMOL, proje yönetiminin özünde olan en temel kavramlar ve ilişkilerin tanıtılmasıyla sınırlandırılmıştır.

Matematiğe dayalı modelleme dilleri formalliği sağlarken, yazılım projelerinde çalışan birçok yazılım geliştiricileri tarafından çeşitli modelleme dilleri kullanılarak geliştirilen modellerin kısa sürede karmaşık bir hal almasından dolayı tercih edilmemektedir. Grafiksel modeller ise daha kolayca kabul görmektedir. Yazılım mühendisliği alanında artık önemli bir konuma gelmiş ve yaygınlaşmış UML modelleme dili, grafiğe dayalı modelleme dillerinin başarısına güzel bir örnektir. PROMOL'deki bazı matematiksel kavramların ve ilişkilerin karşılığı olan grafik model tanımları Şekil 1'de verilmiştir.

Aktivite, bir dikdörtgen şeklinin içerisine ya aktivitenin adı ya da tanımlama simgesi yazılarak belirtilir. Bir varlık ise gene aynı şekilde bir elips ile

belirtilmiştir. Gerek ilişkisi aktiviteler ve varlıklar arasına çizilen düz bir çizgi ile gösterilir. Sonraki ve önceki ilişkileri formal simgelerine benzeyerek sağa ve sola ok çizgilerle kullanılırlar. Aktitivelere girdi olarak modellenen varlıklar, girdi oldukları aktiviteye çizginin sonunda boş ufak bir daire ile birleştirilip, çıktı oldukları zaman ise dairenin içi dolu olur. BAŞLA ve BİTİR özel tanımlarının değişik durumlardaki şekilsel gösterimi şekil 1'de belirtilmiştir.

Çeşitli seviyelerdeki modellerde bazı aktivitelerin veya varlıkların artık daha fazla bölünemeyeceğini belirtmek için bu aktivite veya varlıkların altı çizilir. Bu da artık o aktivite veya varlığın hiyerarşide en alt seviyeye ulaştığını belirtir.



Şekil 1. PROMOL'deki Kavramların ve İlişkilerin Şekil Gösterimleri

4. Modellerin Geliştirilmesi

PROMOL ile bir yazılım projesinin yönetim modelinin geliştirilmesi için genel olarak izlenmesi gereken adımlar aşağıda verilmiştir:

1.Projedeki temel aktiviteler tespit edilir.

2.Proje ile ilgili kişiler, aktivitelerin girdi ve çıktıları, kontratta ayrıca belirtilmiş proje ile teslim edilmesi gereken varlıklar (yazılım kodu, kullanma klavuzları, servisler vs.) ve diğer tüm modellenmesi gereken varlıklar tespit edilir.

3. BAŞLA (START) ve BITIR (END) tanımları kullanılarak üst seviye bir model oluşturulur.

4.Bu üst seviye modeldeki aktivite ve varlıklar PROMOL'de tanımlanmış ilişkiler kullanılarak detaylandırılır.

5.Üst seviye modeller proje yöneticileri tarafından böl ve birleştir ilişkileri ile istenen seviyeye kadar ayrıntılandırılır.

5. Proje Yönetim Modeli Örneği

PROMOL'un kullanılabilirliğini ve yeterliliğini anlamak ve incelemek maksadıyla çeşitli projelerin yönetimleri modellenmiştir. Bunların arasında F-16 savaş uçağının yazılım ve elektronik sistemleri revizyon projesinin yönetimi de vardır [17]. Fakat burada birçok okuyucuya hitap etmesi için basit bir proje modelinin bir parçası örnek olarak verilmiştir.

Bir video kiralama şirketi çeşitli depo işlemlerini bir yazılım ile otomatik hale getirmek istemektedir. Bu projeyi şirketin kendi bilgi işlem departmanında çalışan personel üstlenmiştir. Proje yöneticisi şelale yaşam döngüsü metoduna benzer bir yaklaşım ile projeyi ele almıştır. Bu örneğin seçilmesinin nedeni, bu projenin birçok yazılım projesindeki temel aktiviteleri ve varlıkları içermesidir.

Bir önceki bölümde anlatılan modellerin geliştirilmesindeki adımlar takip edilerek proje PROMOL ile modellenmiştir. Öncelikle projenin en üst seviyesi modellenmiş olup, şekil 2'de formal model, şekil 3'te ise grafik modeli verilmiştir. Şekil 2'nin ilk kısmında proje yönetimi için tespit edilen üst seviye aktiviteler ve varlıklar bulunmaktadır. Şekil 2'nin devamında ise projenin üst seviyesini oluşturan aktiviteler ve varlıkların arasındaki ilişkiler listelenmiştir. Daha sonra bu üst seviye aktivitelerden gereksinim analizi ($a_4()$) aktivitesi böl (divide) ilişkisi ile detaylandırılmıştır. Gereksinim analizi için gerekli (require) olan gruplar proje geliştirme takımı (project team) ve sistem kullanıcılarıdır (system users). Şekil 4'de gereksinim analizi aktivitesinin formal modeli, şekil 5'te ise grafik modeli verilmiştir. Bu seviyede aktiviteler tarafından yaratılan (create) ve dönüştürülen (transform) varlıkların ilişkisi ayrıntılı olarak gözükmektedir. Şekillerde verilen modellerin takibini kolaylaştırmak maksadıyla aktivite ve varlık adlarıyla beraber simgeleri (parantez içindekiler) birlikte sunulmuştur.

Activities

$a_1()$	Identify Stakeholders
$a_2()$	Scope Definition
$a_3()$	Problem Analysis
$a_4()$	Requirement Analysis
$a_5()$	Logical Design
$a_6()$	Physical Design
$a_7()$	Construction & Testing
$a_8()$	Implementation & Delivery

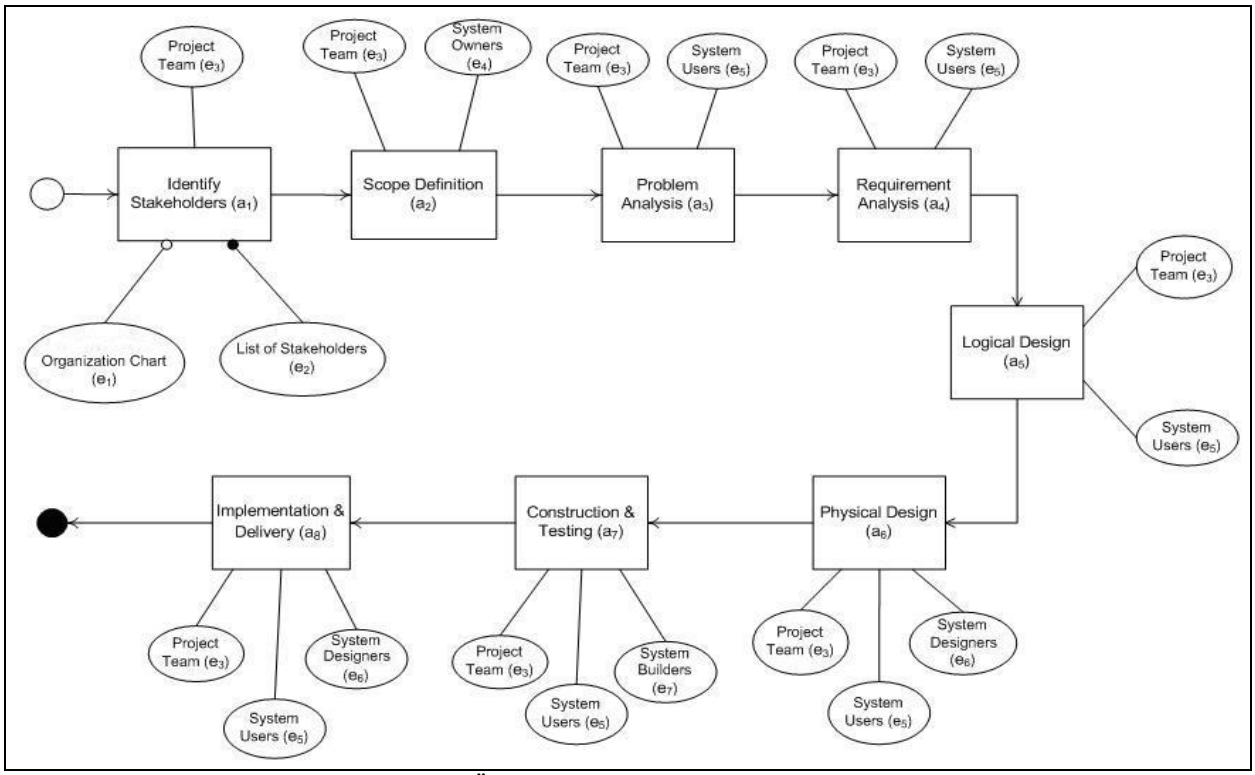
Entities

e_1	Organization Chart
e_2	List of Stakeholders
e_3	Project Team
e_4	System Owners
e_5	System Users
e_6	System Designers
e_7	System Builders

$$P_{ss} = PM\{START, a_1(), a_2(), a_3(), a_4(), a_5(), a_6(), a_7(), a_8(), e_1, e_2, e_3, e_4, e_5, e_6, e_7, END\}$$

$START \rightarrow a_1()$	$e_2 = a_1(e_1)$
$a_1() \rightarrow a_2()$	$REQUIRE(a_1()) = \{e_3\}$
$a_2() \rightarrow a_3()$	$REQUIRE(a_2()) = \{e_3, e_4\}$
$a_3() \rightarrow a_4()$	$REQUIRE(a_3()) = \{e_3, e_5\}$
$a_4() \rightarrow a_5()$	$REQUIRE(a_4()) = \{e_3, e_5\}$
$a_5() \rightarrow a_6()$	$REQUIRE(a_5()) = \{e_3, e_5\}$
$a_6() \rightarrow a_7()$	$REQUIRE(a_6()) = \{e_3, e_5, e_6\}$
$a_7() \rightarrow a_8()$	$REQUIRE(a_7()) = \{e_3, e_5, e_7\}$
$a_8() \rightarrow END$	$REQUIRE(a_8()) = \{e_3, e_5, e_6\}$

Şekil 2. En Üst Seviye Proje Yönetim Formal Modeli



Şekil 3. En Üst Seviye Proje Yönetim Grafik Modeli

Activities	
a_{13}	Identify Requirements
a_{14}	Prioritize Requirements
a_{15}	Update Project Plan
a_{16}	Present Project Plan

Entities	
e_{13}	Improvement Objectives
e_{14}	Draft Requirements
e_{15}	Prioritized Requirements
e_{16}	Project Plan

$$DIVIDE(a_4()) = \{start, a_{13}(), a_{14}(), a_{15}(), a_{16}(), e_3, e_4, e_5, e_{13}, e_{14}, e_{15}, e_{16}, end\}$$

$$start_{a_4} \rightarrow a_{13}()$$

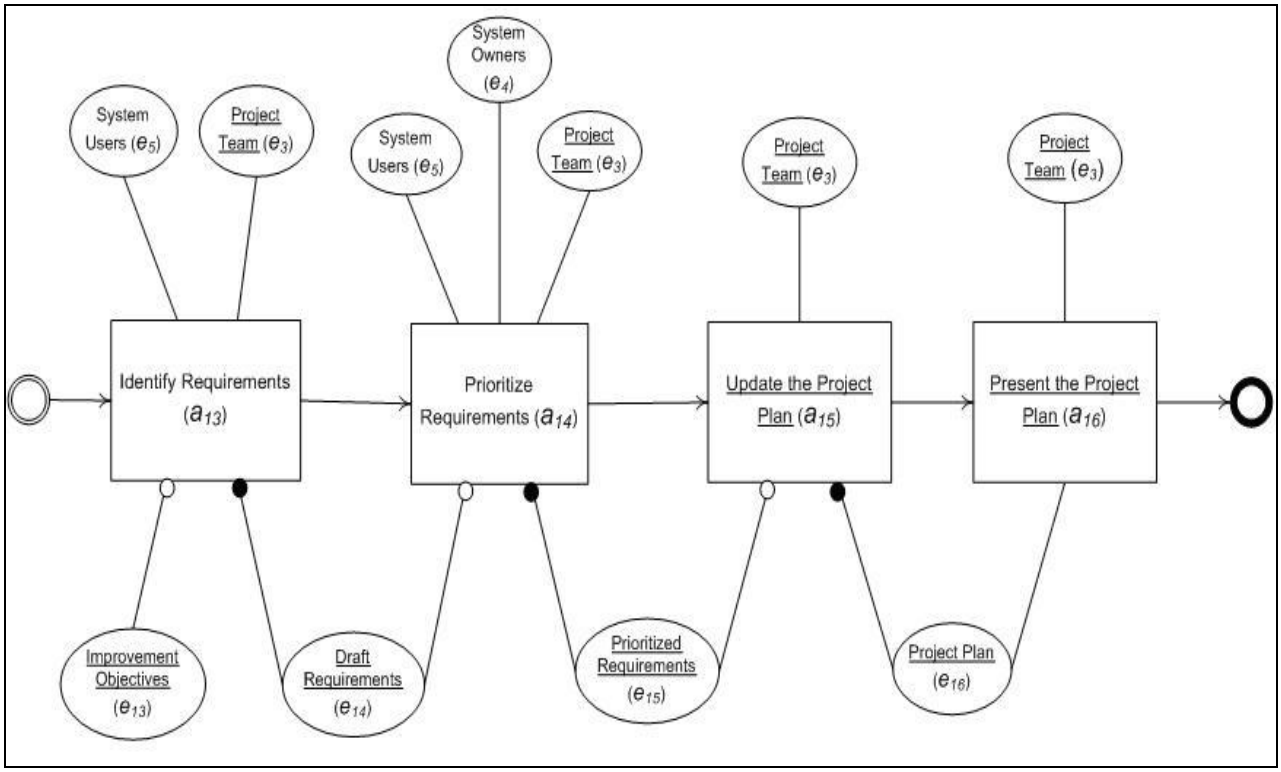
$$a_{13}() \rightarrow a_{14}() \qquad e_{14} = a_{13}(e_{13}) \qquad REQUIRE(a_{13}()) = \{e_3, e_5\}$$

$$a_{14}() \rightarrow a_{15}() \qquad e_{15} = a_{14}(e_{14}) \qquad REQUIRE(a_{14}()) = \{e_3, e_4, e_5\}$$

$$a_{15}() \rightarrow a_{16}() \qquad e_{16} = a_{15}(e_{15}) \qquad REQUIRE(a_{15}()) = \{e_3\}$$

$$a_{16}() \rightarrow end_{a_4} \qquad REQUIRE(a_{16}()) = \{e_3, e_{16}\}$$

Şekil 4. Gereksinim Analizi Aktivitesinin Detaylandırılmış Formal Modeli



Şekil 5. Gereksinim Analizi Aktivitesinin Detaylandırılmış Grafik Modeli

Söz konusu projenin genel olarak şelale yaşam döngüsü yaklaşımıyla geliştirilmesi özellikle grafik modellerin çok basit bir hal almasına yol açmıştır. Ancak, bu kadar basit bir modelde dahi proje yönetimi için önemli olan kavramlar açık ve öz bir şekilde ifade edilebilmiştir.

PROMOL modelleme dilinin en önemli özelliklerinden biri proje yönetimi ile ilgili önemli yönlerin birçoğunu aynı model üzerinde gösterebilmesidir. Şekil 3 ve 5'teki grafik modellerin hızlı bir şekilde gözden geçirilmesi bile projenin o seviyesi için hangi grup ve şahısların hangi aktivite ve varlıklarla ilişkili olduğunun, aktiviteler arasındaki öncelik ilişkilerinin, aktiviteler sonrası ortaya çıkan varlıklar ve bu varlıkların diğer aktivitelerle ilişkilerinin kolayca anlaşılabilmesine olanak sağlamaktadır.

6. Sonuçlar ve Tartışma

Yazılım projelerinin yönetiminin modellenebilmesine olanak veren bir modelleme dilinin olmasının çok çeşitli faydaları vardır. Bu bildirinin ilk yazarının 2007 yılında dünya çapındaki yazılım yönetici ve geliştiricilerinin katıldıkları son yazılım projelerinde karşılaştıkları güçlüklere ait bir anketin (toplam 78 yazılım projesine ait) sonucuna göre yazılım projelerinin yarısı proje kapsam yönetimi ve gereksinim yönetimi alanlarında problem yaşamaktadırlar. Bu empirik gözlem Pinto'nun [14] projeye başlanmadan önce projenin mümkün olan tüm yönlerinin modellenmesinin önemini vurgulamasını açıklamaktadır. PROMOL modelleme dili proje yönetiminin hızlı bir şekilde birçok yönüyle modellenmesine olanak vermesi ile yazılım projelerinin kapsam yönetimi konusundaki yaşadığı sıkıntıları hafifletecektir. PROMOL'un sağladığı avantajlar aşağıda kısaca sıralanmıştır.

- PROMOL proje planlamasına yardımcı olur. Proje planlaması yazılım projeleri yöneticilerinin temel sorumlulukları arasındadır. PROMOL ile proje yöneticileri hızlı bir şekilde projeleri modelleyip çeşitli çözüm alternatiflerini değişik kısıt ve imkanlar altında inceleyebileceklerdir.
- PROMOL ile geliştirilen proje yönetim modelleri aktivitelerin girdi ve çıktılarının takibine olanak sağlar. Dolayısıyla kontratta yeralan proje ile teslim edilecek döküman, servis, yazılım ve diğer varlıkların yönetilmesine olanak sağlar.
- Proje ile ilgili grup ve şahısların çeşitli aktivite ve varlıklarla olan ilişkilerini modellerde açık bir şekilde tanımlamak mümkündür. Bu da proje ile ilişkili grup ve şahısların koordinasyonlarına ve yönetimine yardımcı olur.
- Projelerin degişmez gerçeklerinden bir tanesi proje planlarının proje icrasi esnasında değişmesidir. Dolayısıyla bir proje yönetim modelleme dilinin yazılım projelerinin bu kalıtsal özelliğini dikkate alması önemlidir. PROMOL ile üst seviye modeller proje baslangıcında geliştirilebilirken, aktivitelerin ve varlıkların detaylı modellerinin geliştirilmesi sonraki zamanlara bırakılabilir. Bu detaylandırma işlemi modelin diğer kısımlarında değişikliğe yol açmaz. Dolayısıyla PROMOL modelleri dinamiktir ve projelerdeki değişikliklere hızlı bir şekilde adapte olur.
- Projelerin planlarının ve çeşitli yönlerinin ilgili şahıs ve gruplara yanlış anlaşılmaya mahal vermeyecek şekilde iletişimi önemlidir. PROMOL formal bir modelleme dilidir. Dolayısıyla iletişimde netlik ve açıklık sağlar. PROMOL'un formal özelliği proje modellerinin herkes tarafından aynı şekilde anlaşılmasına olanak verir. Ayrıca formal modeller üzerinde araştırmacılar tarafından kolaylıkla incelemeler yapılabilir. Proje yöneticileri tarafından ise öğrenilen dersler dökümanları kolayca hazırlanıp dersler formal bir şekilde tanımlanabilir.
- PROMOL'un görsel özelliği proje yöneticileri tarafından kolayca öğrenilebilmesine ve kullanılabilmesine olanak sağlar. PROMOL'un basitliği modellerin kullanılabilirliğini artırır. Aynı zamanda proje yönetimindeki birçok önemli kavramın aynı anda görsel olarak algılanabilmesine olanak sağlaması da modelleme dilinin üstün özellikleri arasındadır.
- PROMOL ile proje yönetim şablonları hazırlanabilir. Özellikle çok sayıda projenin geliştirildiği büyük yazılım şirketlerinde ya da organizasyonlarında projelerin bir standarda bağlı kalarak belirli bir kalite seviyesinde ve ortak prensiplerle hazırlanması önem arzeder. PROMOL bu ihtiyaca da kolayca cevap verir.

PROMOL geliştirilmeye açık bir modelleme dilidir. Modelleme diline ek yeni ilişkiler tanımlanabilir. Ancak, burada dikkat edilmesi gereken bir husus vardır. Modelleme dilinde özel durumlar için çok fazla yeni ilişkinin tanımlanması PROMOL'un optimalliğini bozacak ve yarar sağlamak yerine kullanılabilirliğini azaltacaktır.

PROMOL modelleme dili günümüzdeki birçok proje yönetim aracında olmayan özelliklere sahip olup yazılım projeleri yönetimi için önerilmiş ilk çok yönlü modelleme dilidir. Modellediğimiz projelerden edindiğimiz tecrübeler, PROMOL'ün büyük ve kapsamlı yazılım projelerindeki ihtiyaçlara rahatlıkla cevap verebileceğini göstermiştir [17]. Ayrıca değişik yaşam döngüleriyle de kullanılabilir.

Yazılım projelerinde proje maliyet ve süre tahmini, kaynak ve risk yönetimi gibi faaliyetler proje yöneticilerinin en önemli sorumlulukları arasındadır. Halihazırda modelleme dili bu faaliyetlere çözüm

sunamamakta, ancak yaptığımız incelemelere göre bu önemli faaliyetlere ait yaklaşımların kolayca entegre edilebilir olduğudur. Bildirinin yazarları bu konular üzerinde yoğunlaşmış olup mütakip bildirilerde araştırma sonuçlarını sunacaklardır.

PROMOL modelleme dili teknik yaşam döngü yaklaşımlarından bağımsız olarak dizayn edilmiş olup, gereksinim analizi, test aktiviteleri gibi teknik faaliyetlere direk etkisi yoktur. PROMOL modelleme dili özellikle bu şekilde dizayn edilmiş olup, sebebi modelleme dilinin her türlü yaşam döngüsü ile kullanılabilmesine olanak sağlamaktır. Ancak, çeşitli teknik faaliyetlere dolaylı olarak pozitif etkisi mevcuttur. Örnek vermek gerekirse, yazılım gereksinim analizinde proje ile ilişkili şahıs ve grupların tespiti öncelikli husustur. PROMOL ile modellenen projelerde faaliyetlerle ilgili kişi ve gruplar açıkça belirtilir. Bu da gereksinim analizi faaliyetinin daha etkinlikle icra edilmesini sağlar.

PROMOL modelleme dilinin şu an için otomatik yazılım desteği yoktur. Geliştirilen proje yönetim modelleri OpenOffice ve MS Office gibi yazım editörleri yardımı ile geliştirilmiştir. Modellerin bu editörler ile geliştirilmesinde herhangi bir güçlükle karşılaşılmamış olmasına rağmen, PROMOL için ayrı bir editörun geliştirilmesinin faydalı olacağını düşünmekteyiz. Araştırmamız böyle bir otomatik modelleme yazılımının geliştirilmesi ile devam edecektir.

7. Açıklamalar

Bu bildirinin kapsamında olan direk veya dolaylı tüm fikir, yorum ve görüşler yazarların şahsi fikirleri olup, bağlı bulundukları hiçbir kurumun direk veya dolaylı olarak resmi veya gayriresmi hiçbir görüşünü temsil etmezler. Bağlı bulundukları kurumlar bu bildiriden doğan her türlü sorumluluktan muaftırlar.

8. Kaynaklar

[1] Contracting for computer software development, FGMSD-80.4, US General Accounting Office, Washington, DC, 1979.

[2] CHAOS, in: The Standish Group Report, Standish Group, West Yarmouth, MA, 1995.

[3] K. E. Emam, and G. Koru, "A Replicated Survey of IT Software Project Failure Rates", *IEEE Software*, 2008, accepted to appear.

[4] C. Jones, "Software Project Management Practices: Failure versus Success",*Crosstalk,* Vol. 17, No. 10, October, 2004

[5] M. W. Evans, A. M. Abela, and T. Beltz, "Seven Characteristics of Dysfunctional Software Projects", *Crosstalk,* Vol. 15, No. 4, April 2002.

[6] T. DeMarco, and T. Lister, *Peopleware: Productive Projects and Teams,* 2nd Edition, Dorset House Publishing Company, New York, NY, 1999

[7] S. Robertson, and J. Robertson, *Requirements-Led Project Management,* Pearson Education Inc., Boston, MA, 2005

[8] G. Weinberg, *Quality Software Management: Volume 3 Congruent Action*, Dorset House, New York, 1994

[9] F. P. Brooks, "No Silver Bullet-Essence and Accident", Chapter 16, *The Mythical Man-Month: Essays on Software Engineering*, Anniversary Edition, Addison Wesley Longman Inc, MA, USA, 1995

[10] C. Jones, "Project Management Tools and Software Failures and Successes"*, Crosstalk,* July, 1998

[11] Project Management Institute, A Guide to the Project Management Body of Knowledge, 3rd Ed., 2004

[12] CMMI Product Team, Capability Maturity Model Integration Version 1.2, Software Engineering Institute, Carnegie Mellon University, August 2006

[13] IEEE Computer Society, A Guide to Software Engineering Body of Knowledge, 2004 Version

[14] J. Pinto, Project Management Handbook, PMI, Jossey Bass Inc., San Francisco, CA, 1998

[15] P. W. G. Morris, J. K. Pinto, The Wiley Guide to Managing Projects, John Wiley and Sons, Inc. Hoboken, New Jersey, 2004, Chapter 13 by Ali Jaafari, pp. 301-302.

[16] K. Molokken, and M. Jorgensen, "A Review of Surveys on Software Effort Estimation", *Proc. of the 2003 Int. Symp. on Empirical Software Engineering, ISESE'03,* 30 Sep.-1 Oct. 2003, Rome, Italy, pp. 223-231.

[17] A. Ergüner, A Novel Project Management Theory and Its Applicability, Masters Thesis, Naval Postgraduate School, Monterey, CA, USA, March 2008.