

Eş Zamanlı Yazılımlarda Güvenilirlik Analizi : Literatür Taraması

Erkut Tekeli

Çukurova Üniversitesi, Kozan Meslek Yüksekokulu, Adana
etekeli@cu.edu.tr

Özet: Son yıllarda yüksek başarımlı hesaplamalara olan ihtiyaçlar ve donanım bileşenlerindeki teknolojik gelişmeler eş zamanlı (paralel) yazılım teknolojilerinin de gelişmesini tetiklemiştir. Belirli bir anda tek bir bileşenin çalıştığı sıralı (seri) uygulamalar için kullanılan güvenilirlik analizi yöntemleri eş zamanlı yazılım sistemleri için yetersiz kalmıştır. Yapılan çalışmada eş zamanlı yazılım mimarisi temelli güvenilirlik analizini yöntemleri incelenmiştir.

Anahtar Sözcükler: Yazılım Güvenilirlik Analizi, Sistem Güvenilirlik Analizi, Eş Zamanlı Yazılım, Paralel Hesaplama

A Study on Architecture-Based Software Reliability Analysis of Concurrent Softwares

Abstract: In recent years, technological advances in high-performance computations and hardware components have also triggered development of concurrent (parallel) software technologies. Reliability analysis methods are used for sequential applications running in a single component at a given time has been inadequate for concurrent software systems. In this study, architecture-based software reliability analysis methods about concurrent software systems were investigated.

Keywords: Software Reliability Analysis, System Reliability Analysis, Concurrent Software, Parallel Computing

1. Giriş

Kullanıcılar tarafından bir sistemin ömrü, özellikleri ile ilgili verilen hizmetin iki durum arasındaki geçişine göre algılanmaktadır. Durumlardan ilki verilen hizmetin sistem fonksiyonunu yerine getirdiği hatasız hizmet, ikincisi ise verilen hizmetin sistem fonksiyonunu yerine getiremediği hatalı hizmettir [7].

Hatalı bir hizmetten hatasız hizmete geçişi onarım olarak tanımlarken başarısızlığı ise hatasız hizmetten hatalı bir hizmete geçiş olarak tanımlayabiliriz. Buradan yola çıkarak güvenilirliği hatasız hizmetin sürekli olarak verilmesinin ölçümü olarak tanımlayabiliriz.

Elsayed'e göre [1] güvenilirlik, bir ürün veya hizmetin kesinti olmadan tasarım çalışma koşulları altında belirli bir zaman periyodu için düzgün (hatasız) çalışması olasılığıdır.

$R_d(k)$, k adet giriş noktası içeren bir işletim sırasında sistem başarısızlığı olmamasının olasılığı olsun. $R_d(k)$ kesikli zaman sistem güvenilirliğini;

$$R_d(k) = (1 - p)^k \quad (1)$$

olarak göstermek mümkündür. Burada p, önceki girişlerde başarısızlık olmadığı durumlarda sistemin başarısızlık üretmesi şartlı olasılığıdır.

t_e , bir giriş seçimi ile ilişkili işletim süresini olsun. Sürekli zaman sistem güvenilirliği;

$$R(t) = \lim_{t_e \rightarrow \infty} R_d(k) = e^{-\lambda t} \quad (2)$$

eşitliği ile verilir. Burada, $\lambda = \lim_{t_e \rightarrow \infty} \frac{p}{t_e}$ başarısızlık oranı olarak kabul edilmiştir.

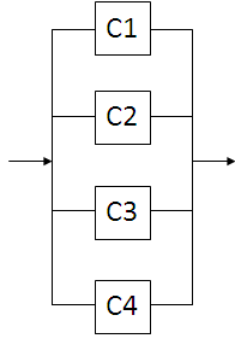
Eğer p, $g_c(p)$ yoğunluk fonksiyonuna sahip sürekli bir rasgele değişken olduğunu ve λ 'nın da $g_c(\lambda)$ yoğunluk fonksiyonuna sahip sürekli bir rasgele değişken olduğunu varsayarsak kesikli ve sürekli durumlarda güvenilirlik için aşağıdaki karma dağılımlar elde edilir.

$$R_d(k) = \int_0^1 (1 - p)^k g_c(p) dp \quad (3)$$

$$R(t) = \int_0^\infty e^{-\lambda t} g_c(\lambda) d\lambda \quad (4)$$

2. Eş Zamanlı (Paralel) Sistemler

Eş Zamanlı (Paralel bağlı) bir sistemde bileşenlerden bir veya daha fazlası başarısız olsa bile kalan bileşen veya bileşenler sistemin düzgün bir şekilde işlemesine imkân verecektir. Yani eş zamanlı sistemin güvenilirliği herhangi bir bileşenin çalışıyor olması olasılığıdır denilebilir. Eş zamanlı bir sistemin güvenilirlik blok diyagramı şekil 1'te gösterilmiştir. [10]



Şekil 1. Eş zamanlı bir sistemin güvenilirlik blok diyagramı

Eş zamanlı sistemlerde güvenilirliği en az bir bileşenin çalışması olasılığını belirleyerek hesaplayabiliriz.

$$R = P(x_1 + x_2 + \dots + x_n) = 1 - P(\bar{x}_1)P(\bar{x}_2|\bar{x}_1)P(\bar{x}_3|\bar{x}_1\bar{x}_2) \dots \quad (5)$$

Sistemdeki bileşenlerin başarısızlıklarını birbirinden bağımsız kabul edersek (5) eşitliği aşağıdaki şekle dönüşür.

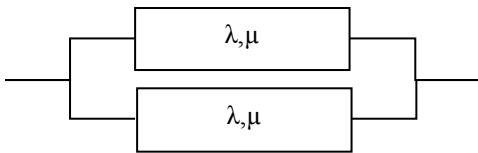
$$R = 1 - P(\bar{x}_1)P(\bar{x}_2)P(\bar{x}_3) \dots P(\bar{x}_n) = 1 - \prod_{i=1}^n P(\bar{x}_i) \quad (6)$$

2.1. Markov Analizi Yöntemi

Markov analizi yöntemi karmaşık sistemlerin analiz edilmesine imkân tanıyan bir yöntemdir. Bu yöntem Markov Zincirleri teorisine dayanır. Sistem bileşenlerinin çalışma, hata veya tamir durumunda olmasına göre analiz yapılır.

2.1.1. Durum Uzaı Diyagramı

Durum uzaı diyagramı bir sistemin güvenilirlik durumlarının grafiksel gösterimidir [5]. Şekil 2'de başarısızlık oranı λ , tamir oranı μ olan ve paralel olarak konumlanmış iki eş bileşen gösterilmektedir. Bu sistemin durum uzaı diyagramını oluşturmaya çalışalım.

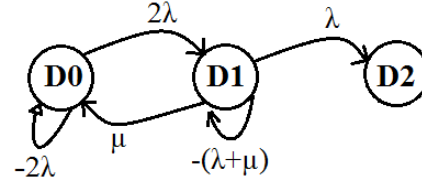


Şekil 2. Paralel konumlanmış iki eş bileşenden oluşan sistem

Bu sistemde 3 durum oluşabilir.

- Durum 0 (D0) : İki bileşen de çalışıyor.
- Durum 1 (D1) : Bileşenlerden biri çalışıyor-yorken diğeri hatalı durumda
- Durum 2 (D2) : Her iki bileşen de hatalı durumda

Durum uzaı diyagramında her bir durum bir düğüm noktası ile belirtilir. Bir durumdan diğeri bir duruma geçiş ise durumlar arasında çizilen oklar ile gösterilir. Şekil 3'te paralel konumlanmış iki eş bileşenin durum uzaı diyagramı görülmektedir.



Şekil 3. Paralel konumlanmış iki eş bileşenin durum uzaı diyagramı

Sistem herhangi bir zamanda bu durumlardan yalnızca bir tanesinde bulunabilir. Ayrıca bir durumdan diğeri bir duruma geçiş sadece 1 adet bileşenin başarısızlığı ile veya tamiri ile mümkündür.

D0 durumundan D1 durumuna geçilirken iki bileşenden herhangi birisinin başarısızlığı yeteceğinden geçiş oranı $\lambda + \lambda = 2\lambda$ 'dir. D1 durumunda hatalı çalışan bileşen tamir edildiğinde tekrar D0 durumuna geçilir ki bu durumda geçiş oranı μ 'dür. Yine D1 durumunda iken çalışan tek bileşen de başarısızlığa uğrarsa D2 durumuna geçilir ve geçiş oranı λ 'dir.

Bir durumda kararlılığı göstermek için diyagramda durumların üzerine çevrimler çizilir. Bu çevrimler negatif bir sayıyla gösterilir ve değeri ilgili durumdan ayrılan geçiş oranlarının toplamıdır.

2.1.2. Geçiş Matrisi

Durum uzaı diyagramından faydalanılarak oluşturulan matrise geçiş matrisi denir. Matrisin elemanı olan a_{ij} , durum j'den durum i'ye geçiş oranını gösterir. Eğer geçiş yoksa $a_{ij}=0$ dir. Geçiş matrisinin özellikleri şunlardır.

- Geçiş matrisi kare matristir.
- Sütunların toplamı 0'dır.
- Matrisin köşegen elemanları durum diyagramındaki çevrimleri gösterir.

İki eş bileşenin paralel olması durumunda geçiş matrisi 3x3 boyutunda olacaktır.

$$T = \begin{bmatrix} -2\lambda & \mu & 0 \\ 2\lambda & -(\lambda + \mu) & 0 \\ 0 & \lambda & 0 \end{bmatrix} \quad (7)$$

2.1.3. Paralel konumdaki iki eş bileşenin MTBF'si

$P_0(t)$, $P_1(t)$ ve $P_2(t)$, t anında sırasıyla D0, D1, D2 durumlarında olma olasılığı olsun. t+Δt anında;

$$P_0(t+\Delta t) = P_0(t) - P_0(t)2\lambda\Delta t + P_1(t)\mu\Delta t \quad (8)$$

(8) eşitliğinde;

$P_0(t)2\lambda\Delta t$, Δt süresince D1 durumuna geçme olasılığını,

$P_1(t)\mu\Delta t$ ise Δt süresince D1 durumundan D0 durumuna geçilme olasılığını gösterir.

Aynı şekilde $P_1(t+\Delta t)$ ve $P_2(t+\Delta t)$ olasılıkları da aşağıda gösterilmiştir;

$$P_1(t+\Delta t)=P_0(t)2\lambda\Delta t +P_1(t) -P_1(t)(\mu+\lambda)\Delta t \quad (9)$$

$$P_2(t+\Delta t) = P_1(t)\mu\Delta t + P_2(t) \quad (10)$$

Sistemin paralel yapısı sebebiyle D0 ve D1 durumlarında sistem çalışır vaziyette olduğundan sistemin t anındaki güvenilirliği;

$$R(t) = P_0(t) + P_1(t) \text{ 'dir.} \quad (11)$$

Bu durumda MTBF (başarısızlıklar arasındaki ortalama zaman);

$$MTBF = \int_0^{\infty} R(t) dt = \int_0^{\infty} P_0(t) dt + \int_0^{\infty} P_1(t) dt \text{ 'dir.} \quad (12)$$

$\Delta t, 0$ 'a yaklaştığında (8), (9), (10) eşitlikleri aşağıdaki hale dönüşür.

$$P_0'(t) = -2\lambda P_0(t) + \mu P_1(t) \quad (13)$$

$$P_1'(t) = 2\lambda P_0(t) - (\lambda + \mu)P_1(t) \quad (14)$$

$$P_2'(t) = \lambda P_1(t) \quad (15)$$

Bu üç eşitliği matris formunda yazarsak;

$$\begin{bmatrix} P_0'(t) \\ P_1'(t) \\ P_2'(t) \end{bmatrix} = \begin{bmatrix} -2\lambda & \mu & 0 \\ 2\lambda & -(\lambda + \mu) & 0 \\ 0 & \lambda & 0 \end{bmatrix} \cdot \begin{bmatrix} P_0(t) \\ P_1(t) \\ P_2(t) \end{bmatrix} \quad (16)$$

Eşitliğin her iki tarafının integralini alıp çözersek;

$$\int_0^{\infty} P_0(t) dt = \frac{\lambda + \mu}{2\lambda^2} \quad (17)$$

$$\int_0^{\infty} P_1(t) dt = \frac{1}{\lambda} \quad (18)$$

sonuçları bulunur. (17) ve (18) eşitliklerini (12) eşitliğinde yerine koyarak MTBF bulunur.

$$MTBF = \int_0^{\infty} P_0(t) dt + \int_0^{\infty} P_1(t) dt = \frac{\lambda + \mu}{2\lambda^2} + \frac{1}{\lambda} = \frac{3\lambda + \mu}{2\lambda^2} \quad (19)$$

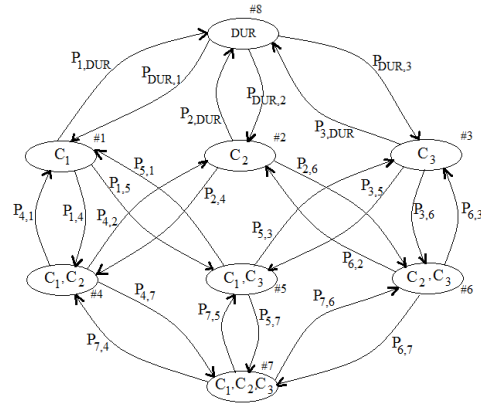
2.2. Eş Zamanlı Yazılım Uygulamalarının Mimari Bazlı Güvenilirlik Analizi

Yazılım mimarisi bir dizi bileşen, bağlantılar ve yapılandırılmaları içeren kavram olarak tanımlanır [9].

Modern yazılımlar genellikle performansı artırmak ve/veya hataya dayanıklılık sağlamak için eş zamanlı (paralel) yazılım mimarisi kullanımı gibi farklı mimariler kullanabilmektedir. Bir yazılım bileşeninin fonksiyonunu mimari açıdan modellemek için yapısal ve davranışsal özelliklerini modellemek gerekir [8]. Bu modellerin analizi sistemin güvenilirliğini etkileyebilecek tasarım sorunlarını ortaya çıkarabilir.

Mimari bazlı güvenilirlik analizleri üzerine yapılan çalışmalar çoğunlukla belirli bir anda tek bir bileşenin çalıştığı sıralı (seri) uygulamaları konu almıştır. Seri yazılımlarda yordamsal (prosedürel) programlama teknik-leri kullanılmaktadır [2, 11]. Fakat günümüzdeki birçok yazılım uygulamasında genellikle nesne tabanlı ve bileşen bazlı yazılım geliştirme paradigmaları kullanılmaktadır. Bu uygulamalarda bileşenlerin eş zamanlı (paralel) çalışmasına da sıklıkla rastlanmaktadır [6].

Şekil 4, üç bileşenli eş zamanlı bir uygulama mimarisinin durum uzayı diyagramıdır. C_i , sistemin i inci bileşeni olsun. Diyagramın her durum noktasında belli sayıda bileşen eş zamanlı olarak çalışmaktadır. <DUR> durumu, hiçbir bileşenin aktif olmadığı durumu gösterebilir. Hiçbir bileşen aktif olmadığı için <DUR> durumunda bir başarısızlık olması söz konusu değildir.



Şekil 4. Eş zamanlı çalışan C_1 , C_2 ve C_3 bileşenlerinden oluşan sistem mimarisinin durum uzayı diyagramı

Şekil 4'deki sistemde güvenilirlik hesaplamasının nasıl yapılabileceğine bakalım.

k, sistemde en az 1 bileşenin çalıştığı durumların sayısı olsun. n ise toplam bileşen sayısını belirtsin. C_i bileşenindeki hataların λ_i başarısızlık oranı ile üssel olarak dağıldığını varsayalım. π_j , sistemin j inci durumda olma olasılığı olsun. π_j , durum uzayı diyagramı kullanılarak Sürekli Zaman Markov Zinciri (Continuous Time Markov Chain - CTMC) modelinin oluşturulup MATLAB yazılımı (veya benzeri bir yazılım) kullanılarak çözülmesi ile elde edilebilir [4].

Toplam çalışma süresinin t olduğunu varsayalım. Bu durumda j durumunun ortalama çalışma süresi $t_j = t\pi_j$ olacaktır. Durumdaki aktif çalışan bileşenlerin çalışma süreleri ise t_j/c_j olacaktır. Burada c_j , j durumundaki aktif bileşen sayısıdır. Böylece bir bileşenin sistemdeki toplam çalışma süresi ise şu şekilde bulunabilir;

$$\omega_i(t) = \sum_{j=1}^k \pi_j \frac{t}{c_j} I_{i,j} \quad (20)$$

Burada; $I_{i,j} = \begin{cases} 1, & 1 \leq i \leq n, 1 \leq j \leq k \\ & \text{ve } C_i \text{ durum } j' \text{ de aktif ise} \\ 0, & \text{diğer durumlarda} \end{cases}$

Bileşen i için beklenen güvenilirlik ise aşağıdaki gibidir;

$$R_i(t) = e^{-\lambda_i \omega_i(t)} = e^{-\lambda_i \sum_{j=1}^k \pi_j \frac{t}{c_j} I_{i,j}} \quad (21)$$

(21) eşitliğinden faydalanarak sistemin güvenilirliğini aşağıdaki gibi bulabiliriz.

$$R(t) = \prod_{i=1}^n R_i(t) = \prod_{i=1}^n e^{-\lambda_i \omega_i(t)} = e^{-\sum_{i=1}^n \lambda_i \sum_{j=1}^k \pi_j \frac{t}{c_j} I_{i,j}} \quad (22)$$

3. Sonuç ve Öneriler

Eş zamanlı yazılımların mimari temelli güvenilirlik analizinde bileşen sayısının artmasıyla durum uzayı diyagramında durumların sayısının üssel olarak artacaktır. n adet bileşene sahip eş zamanlı bir uygulamada 2^n adet durum oluşacaktır. Bu da modelin kurulması ve hesaplanmasını zorlaştıracaktır.

4. Kaynaklar

[1] Elsayed E.A., “Reliability Engineering”, Addison Wesley Longman Inc., Reading, Massachusetts, (1996)

[2] Gokhale, S.S., Trivedi, K.S., “Analytical Models for Architecture-Based Software Reliability Prediction: A Unification Framework”, IEEE Transactions on Reliability 55(4), 578-590, (2006).

[3] Goseva-Popstojanova K., Trivedi K.S., “Architecture-based approach to reliability assessment of software systems,” Performance Evaluation, 45(2-3): 578 – 590, (2001)

[4] Hirel C., Tuffin B., Trivedi K.S., “SPNP: Stochastic Petri Nets. Version 6.0”, Lecture Notes in Computer Science 354-357, (2000)

[5] Ireson W.G., Coombs C.F., Moss R.Y., “Handbook of Reliability Engineering and Management”, McGraw-Hill Comp., (1995)

[6] Kharboutly R.E., Gokhale S.S., “Architecture-based Reliability Analysis of Concurrent Software Applications using Stochastic Reward Nets”, The 23rd International Conference on Software Engineering and Knowledge Engineering SEKE 2011, Miami, (2011)

[7] Lyu M.R., “Handbook of Software Reliability Engineering”, IEEE Computer Society Press and McGraw-Hill, (1996)

[8] Medvidoviç N., Taylor R.N., “A Classification and Comparison Framework for Software Architecture Description Languages”, IEEE Transactions on Software Engineering 26(1), 70-93, (2000).

[9] Ramamoorthy S., Rajagopalan S.P., Sathyalakshmi S., “Component-Based Heterogeneous Software Architecture Reliability (Cohar) Modeling”, International Journal on Computer Science and Engineering Vol. 02, No. 04, 1280-1285, (2010)

[10] Roshandel R., “Calculating Architectural Reliability Via Modeling And Analysis” (Phd Thesis), University of Southern California, (2006)

[11] Wang W.L., Wu Y., Chen M.H., “An Architecture-Based Software Reliability Model”, Pacific Rim International Symposium on Dependable Computing, (1999).