二項型 模型을 利用한 應用 소프트웨어의 信賴性 評價에 관한 硏究
-A Study on Reliability Evaluation of Application Software using Binomial-Type Model-

趙 盛 健*
李 相 鐵**

Abstract

Computer software users develop and utilize their application software by themselves since processing methods are different by quantity and quality of the information. The developed model needs input data and error numbers generated during the testing phases. However, total error numbers of the existing model and each error time was needed as data for developing the new model.

But, maximum likelihood estimation must be used to exponential model of binomial-type and estimating of parameters by using the searched data. Parameter estimation can be done with trial and error or simulation.

1. 序 論

1.1 硏究目的

기업이 현장에서 발생하는 多量의 情報를 처리하여 결과를 意思決定의 資料로 사용하기 위하여 업무의 처리 형태나 요구하는 情報의 형식에 맞게 데이타를 加工處理할 수 있는 應用 소프트웨어를 開發하여 사용하고 있다. 그러나 이렇게 開發된 應用 소프트웨어는 開發段階에서나 情報를 처리하는 과정에서 어느 정도의 信賴性을 갖고 있는지 評價하지도 않고 발생되는 데이타를 처리하여 意思決定의 資料로 제공되고 있는 것이 현실이다. 이렇게 현장에서 開發된 수많은 應用 소프트웨어를 開發者나 管理者는 信賴性을 評價하지 않고 현장의 데이타를 처리하고 있는 것은 지금까지의 대부분의 應用 소프트웨어가 단순한 데이타의 처리에 사용되었고, 또 실제 처리된 情報가 意思決定者의 意思決定의 資料로 사용되지도 못했기 때문이라고 생각된다. 그러나 지금은 현장에서 처리되는 情報가 意思決定의 資料로 사용되고 있기도 하지만 開發된 應用 소프트웨어의 信賴性은 문제 삼지 않고 있는데 그것은 開發된 應用 소프트웨어의 信賴性을 간단하게 評價해볼 수 있는 應用 소프트웨어의 信賴性 評價模型이 開發되어 있지 않다는 것과 開發된 대부분의 기존 信賴性 評價模型이 시스템 소프트웨어(system software)의 信賴性을 評價하는 模型이기 때문에 應用 소프트웨어에 적용하기가 어렵기 때문이라고 생각된다. 그러나 기존의 信賴性 評價模型을 이용하여 信賴性을 評價하기 위해서는 應用 소프트웨어에서 발생하는 誤謬 데이타를 기존 開發된 模型에 적용할 수 있도록 조사하여야 하는데 그 조사방법은 조금은 복잡하지만 應用 소프트웨어의 테스트 과정에서 정확하게 조사만 한다면 應用 소프트웨어의 信賴性도 기존의 시스템 소프트웨어 信賴性 評價方法으로 評價해 볼 수 있을 것이다. 따라서 本 硏究에서는 이런 관점에서 應用 소프트웨어를 開發하여 테스트하는 과정에서 발생되는 誤謬를 기존의 二項型 指數模型에 적용하기 위해 필요한 데이타의 調査方法으로 조사하고 旣存模型에 적용하기 위해 데이타를 재정리하여 母數를 推定해 보므로써 현장 데이타를 이용해 信賴性을 評價할 수 있는 방법을 제시하고자 한다.

1.2 硏究方法 및 範圍

應用 소프트웨어에서 처리하는 데이타는 시스템 소프트웨어에서 처리하는 데이타와는 달리 다양화된 非定型

*慶南專門大學 電子計算科 副敎授
**東明專門大學 工業經營科 副敎授
접수 : 1992. 4. 20.
확정 : 1992. 5. 2.

的인 데이타이기 때문에 이런 데이타를 처리하기 위하여 開發되는 소프트웨어는 開發하는 開發者에 따라서 處理方法이 달라질 수 있으며, 또 사용하는 하드웨어(hardware)의 특성과 開發者 자신이 처해 있는 주변여건 등에 의해서 開發되는 應用 소프트웨어의 開發方法이 달라질 수도 있다고 본다. 따라서 동일한 조건에서 定型的인 데이타를 처리하는 시스템 소프트웨어의 信賴性 評價방법보다는 훨씬 다양화될 수 있을 것이다. 이렇게 보면 應用 소프트웨어의 信賴性 評價模型을 어느 조건에서 어떤 방법으로 할 것인가가 문제가 될 수도 있다. 따라서 시스템의 分析이나 設計의 方法에 의해 많은 變數가 작용하기 때문에 하나의 模型(model)을 開發했다고 해도 조건이나 방법을 달리하는 다른 현장에서는 적용하기가 힘들 것으로 본다. 여기서는 이런 문제점을 해결하면서 소프트웨어를 開發하여 사용하는 어느 현장에서나 쉽게 적용해볼 수 있도록 하기 위해서 應用 소프트웨어의 開發段階 중에서 마지막 段階인 테스트 段階에서 발생하는 誤謬發生 데이타를 기존模型에 적용할 수 있는 形式으로 조사하고, 信賴性 評價模型에 적용하여 開發 중인 應用 소프트웨어의 信賴性을 기존의 模型을 이용하여 評價해 보기로 한다. 결국 소프트웨어 開發過程의 마지막 段階인 테스트 段階에서 필요한 데이타를 조사하고 信賴性 評價를 하게 되면 하드웨어가 갖고 있는 특성이나 시스템 分析者가 시스템을 分析하고 設計하면서 발생하는 여러가지 조건들을 變數로 사용하지 않아도 信賴性을 評價할 수 있을 것이기 때문이다. 따라서 本 硏究에서는 誤謬發生 데이타만 應用 소프트웨어의 테스트 과정에서 조사하고 推定에 필요한 母數(parameter)는 기존 模型에서 사용하는 最尤推定法을 使用하여 推定하고 開發 중인 소프트웨어의 信賴性을 評價해 보고자 한다.

2. 二項型 指數模型의 理論的 背景

2.1 記號定義

$f_a(t)$: 확률밀도함수
$F_a(t)$: 누적확률밀도함수
i : 오류발생 순서
m_e : 시간 t_e에서 관측된 오류의 수
$P_m(t)$: t시점에서 m개의 오류가 관측될 수 있는 확률
$Q(t)$: t시점에서 잔존하는 오류의 수로서 확률변수
t : 시간변수
t_i : i번째 오류가 발생한 시간
t_e : 오류발생 관측시간의 끝(테스트의 끝시점)
u_0 : 초기 소프트웨어에 존재하는 고유의 오류 수
$Z_a(t)$: t시점에서의 위험율
β_1 : 모형에 사용되는 수
$\lambda(t)$: 시간 t에서 오류발생밀도를 나타내는 함수

2.2 二項型 指數模型

信賴性을 評價하기 위해 사용하는 二項型 模型에는 指數模型이 사용되는데 이것은 발생되는 誤謬와 誤謬發生 시간간에 분포가 二項分布를 따른다고 假定하고 사용하는 模型이다. 따라서 二項型의 指數模型의 사용은 다음의 가정하에서 가능하게 된다.

假定 1. 언제나 소프트웨어에는 誤謬가 발생한다.
假定 2. 소프트웨어가 갖고 있는 고유의 誤謬를 u_0라고 한다.
假定 3. 모든 발생 誤謬는 독립적으로 발생한다.

위의 가정하에서 $Q(t)=u_0-M(t)$가 되므로 $Z_a(t)$는 (1)式과 같이 나타낼 수 있다.

$$Z_a(t)=f_a(t)/(1-F_a(t)) \quad \cdots\cdots (1)$$

여기서

$$f_a(t)=dF_a(t)/dt$$

이다.

그리고 $F_a(t)$를 數式으로 표현하면 (2)式과 같이 표현된다.

$$F_a(t)=1-\exp[-\int_0^t Z_a(X)dx \quad \cdots\cdots (2)$$

또 $P_m(t+\Delta t)=P[M(t+\Delta t)=m]$이 된다. 이것을 다시 정리해 보면 (3)式과 같이 표현된다.

$$P_m(t+\Delta t)=(u_0-m+1)Z_a(t)\Delta tP_{m-1}(t)+[1-(u_0-m)Z_a(t)\Delta t]P_m(t) \quad \cdots\cdots (3)$$

이 때 m은 0에서 u_0까지이다.

위의 式들을 정리하면 二項型 指數模型의 평균치함수식은 (4)式과 같고 오류발생밀도함수식은 (5)式과 같으며 信賴度는 (6)式과 같이 나타난다.

$$\mu(t)=u_0[1-\exp(-\beta_1 t)] \quad \cdots\cdots (4)$$

$$\lambda(t)=u_0\beta_1\exp(-\beta_1 t) \quad \cdots\cdots (5)$$

$$R(t)=\exp(-\beta_1 t) \quad \cdots\cdots (6)$$

여기서 二項型 指數模型의 각 函數式에 사용할 u_0와 β_1을 구하기 위해 (7)式과 (8)式을 연립으로 풀어 u_0와 β_1을 구한다.

$$\hat{\beta}_1 t_e+\sum_{i=1}^{m_e} 1/(\hat{u}_0-i+1)=0 \quad \cdots\cdots (7)$$

$$-t_e(\hat{u}_0-m_e)-\sum_{i=1}^{m_e}+(m_e/\hat{\beta}_1)=0 \quad \cdots\cdots (8)$$

위의 두 式에서 u_0와 β_1를 구하는 式을 유도하면 (9)式과 (10)式이 된다.

$$\hat{\beta}_1=m_e/\sum_{i=1}^{m_e} t_i+t_e(\hat{u}_0-m_e) \quad \cdots\cdots (9)$$

$$-[m_e t_e/\{\sum_{i=1}^{m_e} t_i+t_e(\hat{u}_0-m_e)\}]+\sum_{i=1}^{m_e}(1/\hat{u}_0-i+1)=0 \quad \cdots\cdots (10)$$

(9)式과 (10)式을 이용하여 二項型 指數模型의 母數 u_0와 β_1의 추정치를 구하기 위해서는 조사된 데이타를 利用하여 施行錯誤法이나 시뮬레이션을 수행하면서 母數를 推定해야 한다.

3. 데이타의 調査 및 母數推定

3.1 데이타의 調査

二項型의 指數模型에는 모수추정방법을 最尤推定法으로 하여 母數를 推定하여야 하기 때문에 應用 소프트웨어의 테스트 단계에서 발생하는 誤謬를 조사하면서 誤謬發生에 관한 데이타를 기존의 시스템 소프트웨어에 적용 가능한 형태로 조사해야 한다. 따라서 시스템 소프트웨어의 信賴性 評價模型에 사용하는 母數推定에 사용되는 데이타인 誤謬發生時間과 總誤謬發生의 數 그리고 테스트 종료시간을 應用 소프트웨어의 테스트 단위별로 조사하여 시스템 소프트웨어의 信賴性 評價와 같은 방법으로 信賴性을 評價해야 할 것이다. 이렇게 되어야만 기존의 소프트웨어 信賴性 評價模型에서 사용하는 母數推定 방법인 最尤推定의 방법을 적용할 수 있게 되기 때문에 데이타의 조사를 위해 필요한 조치를 應用 소프트웨어 開發段階에서 강구해 두어야 할 것이다.

應用 소프트웨어를 테스트하면서 테스트단계별로 誤謬發生에 대한 데이타를 조사해 정리해 결과는 표-1과 같이 된다.

표-1. 테스트 시행회수별 誤謬發生時間과 誤謬의 수 및 종료시간 (unit : CPU Sec.)

NO	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	12	18	21	28	32	45	27	35	16	68	32	124	187	76	92	96
2	19	22	36	45	41	78	45	58	43	132	76	145	218	122	103	117
3	24	28	53	65	52	93	56	119	75	179	119	187	248	165	162	216
4	36	38	78	71	63	104	98	165	109	216	153	237		212	198	252
5	45	47	92	97	87	143	107	176	143	254	154	264		253	242	
6	51	51	114	99	118	163	129	197	184	261	216	273			256	
7	76	69	126	107	152	187	157	199	201	275	265				278	
8	89	84	154	115	163	246	176	265	254		281					
9	95	92	189	127	174	262	201	276	276							
10	113	101	202	148	201	279	216	279								
11	132	122	226	161	234	281	274	285								
12	154	145	235	173	264	293	276									
13	165	159	236	211	278		295									
14	182	168	243	253	294											
15	201	183	254	273												
16	212	198	265	293												
17	231	219	276	301												
18	257	226	294													
19	273	265	306													
20	297	279														
21	312	302														
22		311														
23		324														
	319	327	314	308	302	301	303	307	299	302	305	297	296	294	293	298

最尤推定法을 이용하여 母數를 推定하기 위하여 조사된 표-1의 데이타를 재정리하여 시스템 소프트웨어의 信賴性 評價에서 사용하는 데이타의 형식으로 바꾸어 정리해 보면 표-2와 같이 된다.

표-2. 誤謬發生 시점에서의 시간 누적치 (unit : CPU Sec.)

數	累積	數	累積	數	累積	數	累積	數	累積	數	累積	數	累積	數	累積
1	12	24	347	47	699	70	1067	93	1546	116	2087	139	2757	162	3902
2	19	25	357	48	724	71	1075	94	1562	117	2145	140	2848	163	3932
3	24	26	366	49	738	72	1087	95	1615	118	2147	141	2912	164	4056
4	36	27	370	50	760	73	1108	96	1648	119	2166	142	2959	165	4102
5	45	28	388	51	772	74	1121	97	1666	120	2209	143	2996	166	4145
6	51	29	403	52	800	75	1133	98	1674	121	2232	144	3034	167	4192
7	76	30	411	53	835	76	1171	99	1713	122	2293	145	3041	168	4233
8	89	31	420	54	848	77	1213	100	1733	123	2339	146	3055	169	4366
9	95	32	441	55	872	78	1233	101	1757	124	2350	147	3114	170	4377
10	113	33	464	56	881	79	1253	102	1816	125	2371	148	3158	171	4436
11	132	34	478	57	882	80	1261	103	1833	126	2373	149	3201	172	4472
12	154	35	487	58	889	81	1300	104	1849	127	2439	150	3235	173	4516
13	165	36	502	59	900	82	1309	105	1851	128	2450	151	3236	174	4530
14	182	37	517	60	911	83	1320	106	1863	129	2453	152	3298	175	4552
15	201	38	538	61	922	84	1331	107	1898	130	2459	153	3347	176	4663
16	212	39	545	62	940	85	1355	108	1916	131	2494	154	3363	177	4684
17	231	40	584	63	952	86	1386	109	1927	132	2524	155	3511	178	4783
18	257	41	598	64	988	87	1420	110	1969	133	2556	156	3532	179	4819
19	273	42	621	65	1005	88	1431	111	1978	134	2590	157	3574		
20	297	43	630	66	1025	89	1442	112	2000	135	2624	158	3624		
21	312	44	643	67	1031	90	1469	113	2028	136	2665	159	3651		
22	337	45	667	68	1057	91	1502	114	2047	137	2682	160	3660		
23	341	46	682	69	1059	92	1532	115	2072	138	2735	161	3871		

테스트 종료시간 : 4819 CPU Sec. 총누적치 : 311383

3.2 母數推定

應用 소프트웨어의 테스트 단계에서 조사된 데이타를 재정리하여 얻은 표-2의 데이타로 最尤推定法을 적용하여 母數를 推定하기 위해서는 앞에서 언급한 (9)式과 (10)式을 이용하여 施行錯誤法을 사용해야 한다.

이 방법으로 母數를 推定해 보면 $u_0=221.9$가 되며, $\beta_1=0.1084$가 된다. 따라서 구하고자 하는 平均値函數式은 다음과 같다.

$$\mu(t)=221.9[1-\exp(-0.1084t)] \quad \cdots\cdots (11)$$

그리고 誤謬發生密度函數式은 다음과 같이 된다.

$$\lambda(t)=24.06\exp(-0.1084t) \quad \cdots\cdots (12)$$

또, 信賴度를 구하는 式은 다음과 같다.

$$R(t)=\exp(-0.1084t) \quad \cdots\cdots (13)$$

4. 應用 소프트웨어의 信賴性 評價

應用 소프트웨어의 개발단계에 발생한 誤謬 데이타로 母數를 推定하여 만든 式들을 이용하여 平均値와 誤謬發生密度 그리고 信賴度를 推定해 보면 표-3과 같이 된다.

표-3. 誤謬發生密度函數를 이용한 推定値

time	誤謬密度	平均値	信賴度	time	誤謬密度	平均値	信賴度	time	誤謬密度	平均値	信賴度
300	21.6	22.8	0.103	8400	1.2	211.2	0.952	16500	0.1	221.3	0.996
600	19.4	43.3	0.195	8700	1.0	212.3	0.957	16800	0.1	221.4	0.996
900	17.4	61.6	0.278	9000	0.9	213.3	0.961	17100	0.1	221.4	0.996
1200	15.6	78.1	0.352	9300	0.8	214.2	0.965	17400	0.0	221.5	0.997
1500	14.0	92.9	0.419	9600	0.8	215.0	0.969	17700		221.5	0.997
1800	12.6	106.1	0.478	9900	0.7	215.7	0.972	18000		221.6	0.997
2100	11.3	118.0	0.532	10200	0.6	216.3	0.975	18300		221.6	0.997
2400	10.1	128.7	0.580	10500	0.5	216.9	0.978	18600		221.6	0.998
2700	9.1	138.3	0.623	10800	0.5	217.4	0.979	18900		221.7	0.998
3000	8.1	146.8	0.662	11100	0.4	217.9	0.981	19200		221.7	0.998
3300	7.3	154.6	0.697	11400	0.4	218.3	0.983	19500		221.7	0.998
3600	6.6	161.5	0.728	11700	0.4	218.6	0.985	19800		221.7	0.998
3900	5.9	167.7	0.756	12000	0.3	219.0	0.986	20100		221.7	0.998
4200	5.3	173.3	0.781	12300	0.3	219.3	0.988	20400		221.8	0.999
4500	4.7	178.3	0.803	12600	0.3	219.6	0.989	20700		221.8	0.999
4800	4.3	182.8	0.824	12900	0.2	219.8	0.990	21000		221.8	0.999
5100	3.8	186.8	0.842	13200	0.2	220.0	0.991	21300		221.8	0.999
5400	3.4	190.4	0.858	13500	0.2	220.2	0.992	21600		221.8	0.999
5700	3.1	193.6	0.873	13800	0.2	220.4	0.993	21900		221.8	0.999
6000	2.8	196.5	0.886	14100	0.2	220.5	0.993	22200		221.8	0.999
6300	2.5	199.1	0.897	14400	0.1	220.7	0.993	22500		221.8	0.999
6600	2.2	201.5	0.908	14700	0.1	220.8	0.994	22800		221.8	0.999
6900	2.0	203.6	0.917	15000	0.1	220.9	0.994	23100		221.9	1.000
7200	1.8	205.5	0.926	15300	0.1	221.0	0.995	23400			
7500	1.6	207.1	0.934	15600	0.1	221.1	0.995	23700			
7800	1.4	208.7	0.940	15900	0.1	221.2	0.995	24000			
8100	1.3	210.0	0.946	16200	0.1	221.3	0.995	24300			

推定된 平均値를 도시해 보면 그림-1과 같이 되며 誤謬發生密度를 도시해 보면 그림-2와 같이 된다. 그리고 信賴度를 圖示해 보면 그림-3과 같이 된다.

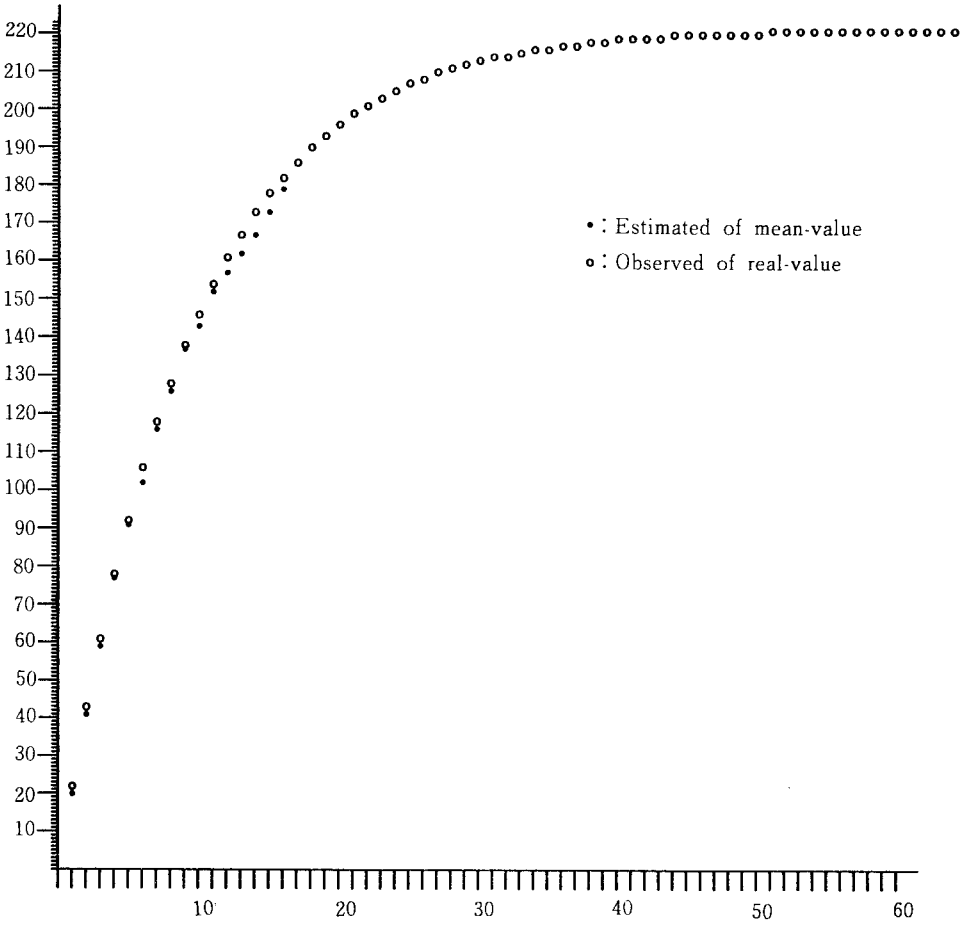

그림-1. 推定値된 平均値의 曲線

실제 관측된 誤謬의 수와 平均値 函數를 이용한 推定値와의 사이에는 큰 차이가 발생하지 않는다는 것은 그림-1에서 알 수 있기 때문에 조사된 데이타를 이용하여 推定한 母數를 誤謬發生密度와 信賴度를 구하는데 사용해도 좋다고 본다. 따라서 구한 母數를 이용하여 개발되고 있는 應用 소프트웨어의 信賴性을 評價해 보기로 한다.

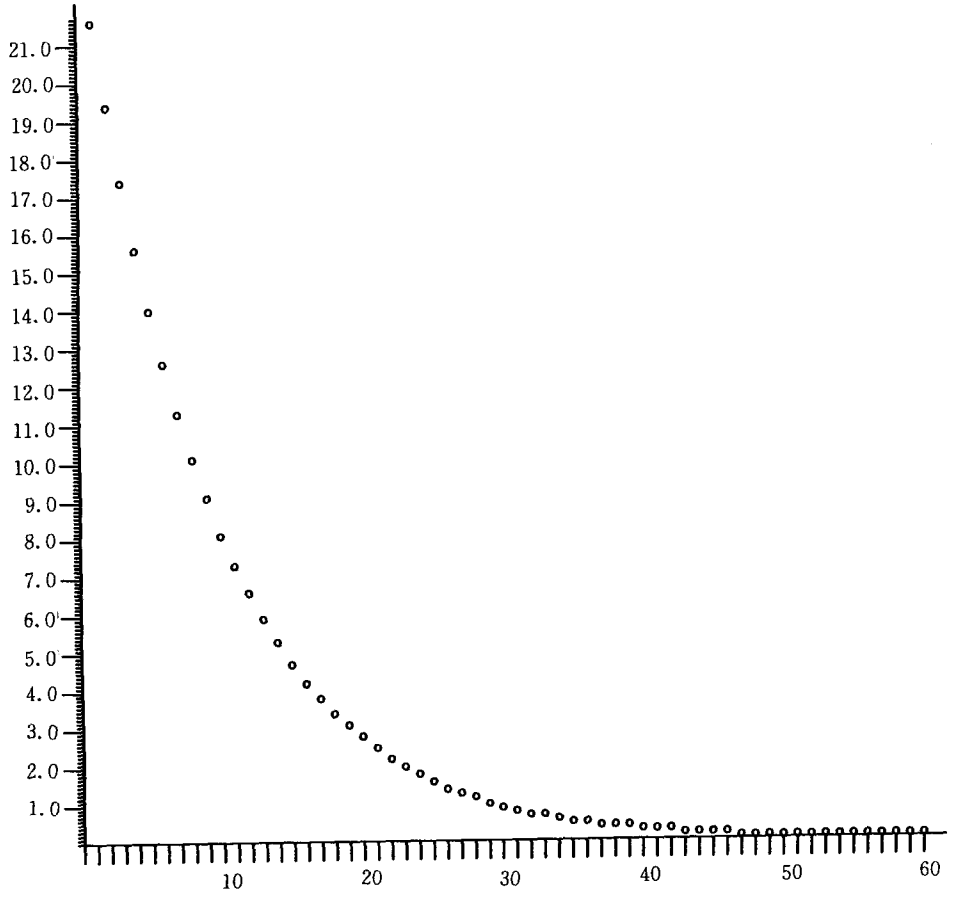

그림-2. 推定値된 誤謬發生密度의 曲線

誤謬發生密度를 보면 17400 CPU Sec.에서 0.0이 되며 이것은 應用 소프트웨어의 테스트 회수로 계산해 보면 약 58회의 테스트를 수행하면서 誤謬를 수정해야 된다는 것을 의미한다. 그러나 현장에서 소프트웨어를 開發하면서 이렇게 많은 테스트를 수행할 수 없기 때문에 소프트웨어의 開發者나 分析者는 어느 시점에서 테스트를 종료하고 현장에서 발생하는 데이타를 처리해야 할 것이다. 따라서 誤謬發生密度가 0이 되는 시점에서 테스트를 종료한다면 30회의 테스트를 수행하는 것이 되며 적당한 시점이라고 볼 수 있다. 이후에 발생하는 誤謬는 실제 발생하는 데이타를 처리하면서 수정하는 것이 바람직하다고 본다.

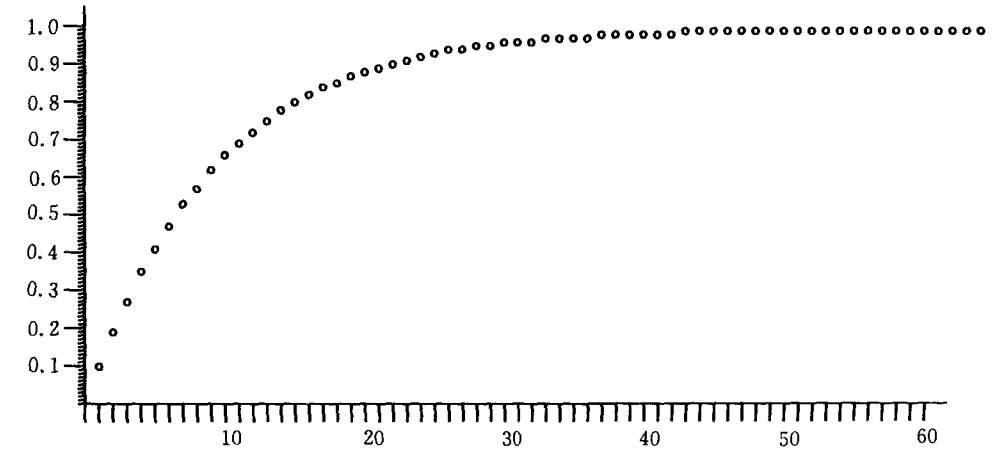

그림-3. 推定値된 信賴度의 曲線

개발되고 있는 應用 소프트웨어가 信賴度 1이 되는 시점은 테스트 회수로 보면 77회의 테스트가 끝나는 시점이 되므로 信賴度를 이용하여 테스트를 종료시키고자 한다면 信賴度가 0.95가 되는 테스트 시점 28회에서 종료하고 실제 발생하는 현장의 데이타를 처리하는 것이 좋다고 본다.

이렇게 보면 誤謬發生密度 0이되는 시점의 테스트 회수와 2회의 차이가 발생하지만 실제 현장에서 開發하는 應用 소프트웨어의 테스트회수에서 본다면 큰 문제가 된다고 볼 수는 없다. 따라서 지금 개발되고 있는 應用 소프트웨어의 실제 현장 데이타 처리시점은 테스트 회수로는 28~30회가 되며 CPU Sec.는 8400 SPU Sec.~9000 CPU Sec.가 됨을 알 수 있다.

그러나 현장에서 개발되고 있는 應用 소프트웨어가 처리해야 할 데이타의 정밀도 및 요구도에 따라서 開發者나 分析者가 테스트 회수를 증가시키거나 감소시킬 수 있을 것으로 보며 또 加工處理해야 할 발생되는 데이타의 量이나 質에 따라서도 달라질 수 있기 때문에 모든 판단은 開發者나 分析者가 해야 할 것으로 본다.

5. 結 論

應用 소프트웨어의 信賴性이 評價되지 못하는 것은 현장에서 개발되는 應用 소프트웨어가 갖는 특수성 때문이라고 생각된다. 그것은 지금까지의 應用 소프트웨어는 현장에서 발생하는 단순한 데이타를 처리하고 있었기 때문에 信賴性을 評價할 필요성을 갖지 못했지만 그러나 지금의 현장 發生 데이타의 처리는 O. R의 기법을 적용하기도 하고 經營者의 意思決定을 직접적으로 돕기 위해 經營情報 시스템을 구축하여 使用하고 있기 때문에 개발되어 사용되는 應用 소프트웨어의 信賴性 評價는 반드시 필요하다고 본다. 따라서 처리된 情報의 信賴性을 높이기 위해서는 開發되는 應用 소프트웨어의 信賴性이 먼저 評價되어야 한다고 본다. 그래서 本 硏究에서는 지금까지 開發된 시스템 소프트웨어의 信賴性 評價模型을 이용하여 현장에서 개발되고 있는 應用 소프트웨어의 信賴性을 評價할 수 있는 방법을 찾아 적용해 보므로서 시스템 소프트웨어 信賴性을 評價하는 模型이 應用 소프트웨어의 信賴性을 評價하는 模型으로 사용되기 위해서는 다음의 두 가지 문제점을 해결해야 할 것이다.

1) 信賴性 評價模型에 사용할 데이타를 현장에서 應用 소프트웨어를 開發하는 과정에서는 조사하기가 용이하도록 해야 한다.
2) 調査된 데이타를 이용하여 信賴性 評價模型에 사용할 母數를 推定할 때 母數의 推定方法이 현장에서 사용하기 용이해야 하며 推定値를 산출하는 데 시간적인 소모를 줄여야 한다.

따라서 위의 문제점을 해결하면서 간편하게 應用 소프트웨어의 信賴性을 評價하기 위해서는 먼저 데이타의 조사가 간편한 테스트 회수와 誤謬發生의 수를 이용하는 방법이 될 것이며 信賴性 評價模型에 사용되는 母數의 推定方法을 현장에서 적용하기 容易하게 하기 위해서 最小自乘法을 사용할 수 있을 것이다. 그러나 만약 最小自乘法을 사용하게 되면 테스트 회수가 많을수록 母數의 推定値가 정확성을 갖기 때문에 일정 회수의 테스트가 끝난 이후부터 적용이 가능하다는 점만 해결할 수 있다면 실제 현장에서 開發되는 應用 소프트웨어의 信賴性 評價模型으로서는 적절하다고 보며 간편하기 때문에 현장의 활용도도 높을 것으로 본다.

參 考 文 獻

1) 金正子, 趙盛健, "統計的 方法을 利用한 Software品質評價에 관한 硏究," *Journal of the KSQC*, 13(2), 62-63, 1985.
2) 金正子, 趙盛健 "Software 信賴性 Model의 開發과 適用에 관한 硏究," *Journal of the KSQC*, 19(1), 68-69, 1991.
3) 佐和隆光, 回歸分析, 朝倉書店, 1979, pp. 169-171.
4) Geol A. L and Okumoto K, "Time-Dependent Error-Detection Rate Model for Software Reliability and Performance Measures," *IEEE Transactions on Reliability*, R-28(3), 206-211, 1979.
5) Gordon B. Davis and Margrethe H. Olson, *Management Information System*, New York : McGraw-Hill, 1985, pp. 168-171.
6) Robert N. Charette, *Software Engineering Enviroments*, New York : McGraw-Hill, 1988, pp. 308-338.
7) Kazuhira Okumoto, "A Statistical Method for Software Quality Control," *IEEE Transactions on Software Enginnering*, SE-11(12), 1426-1427, Dec., 1985.
8) Musa J. D, "A Theory of Software Reliability and its Application," *IEEE Transactions on Software Enginnering*, SE-1(3), 277-287, 1975.

9) Daniel D. McCracken and William S. Dorn, *Numerical Methods and Fortran Programming*, New York : John Wiley and Sons, Inc., 1964, pp. 262-266.
10) Schick G. J and Wolverton R. W, "An Analysis of Competing Software Reliability Models," *IEEE Transactions on Software Enginnering*, SE-4(2), 395-422, 1978.
11) Ian Sommerville, *Software Enginnering*, New York :Addison-Wesley Publishing Company, 1989, pp. 592-598.
12) Shigeru Yamada and Shunji Osaki, "Software Reliability Growth Modeling : Models and Applications," *IEEE Transactions on Software Enginnering*, SE-11(12), 1432-1433, Dec., 1985.
13) Amrit L. Geol, "Software Reliability Models : Assumptions, Limitations and Application," *IEEE Transactions on Software Engineering*, SE-11(12), 1465-1467, Dec., 1985.
14) John R. Rice, *Numerical Methods, Software and Analysis*, New York : McGraw-Hill, 1983, pp. 56-57.
15) Martin L. Shooman, *Software Enginnering : Design, Reliability and Management*, New York : McGraw-Hill, 1983, pp. 332-336.
16) Robert L. Glass, "Persistent Software Errors," *IEEE Transactions on Software Enginnering*, SE-7(2), 166-167, Mar., 1981.