[UCI]I804:24011-200000503913

2021年 8月
博士學位論文

소프트웨어 신뢰성 모형과 순차적 확률비 검정을 적용한 신뢰성 연구

朝鮮大學校 大學院

電算統計學科

李 多 惠

소프트웨어 신뢰성 모형과 순차적 확률비 검정을 적용한 신뢰성 연구

A Study on the Reliability of Software Reliability Models Using Sequential Probability Ratio Test

2021年 8月 27日

朝鮮大學校 大學院

電算統計學科

李 多 惠

소프트웨어 신뢰성 모형과 순차적 확률비 검정을 적용한 신뢰성 연구

指導敎授 張 仁 弘

이 論文을 理學 博士學位 申請 論文으로 提出함

2021年 4月

朝鮮大學校 大學院

電算統計學科

李 多 惠

李多惠의 博士學位論文을 認准함

委員長	朝鮮大學校	敎 授	裵 相 賢	(인)
委 員	慶星大學校	敎 授	鄭 璣 文	(인)
委 員	朝鮮大學校	敎 授	權 容 萬	(인)
委 員	朝鮮大學校	敎 授	金 世 珍	(인)
委 員	朝鮮大學校	敎 授	張 仁 弘	(인)

2021年 6月

朝鮮大學校 大學院

목 차

표 목 차

그 림 목 차

ABSTRACT

A Study on the Reliability of Software Reliability Models Using Sequential Probability Ratio Test

Lee, Da Hye
Advisor : Prof. Chang, In Hong, Ph.D.
Department of Computer Sceience and statistics,
Graduate School of Chosun University

With the development of information technologies such as big Data, machine Learning, and artificial intelligence (AI), software is being used in various fields. In particular, the internet of things (IoT), which combines the Internet and things, has opened the IoT era, steadily expanding its use and form over the past decade. Recently, artificial intelligence of things (AIoT), which combines IoT and AI, is drawing attention. IoT products provide personalized services through artificial intelligence and are used in home appliances, automobiles, and medical equipment, and the software industry is booming.

As software and various fields converge, new industries have emerged, which is drawing attention as a key industry in the future. For example, smart factories utilize IoT to digitize factory records and data into electronic documents and manage the incoming and outgoing of products in real time. Smart medicine helps patients check their health conditions and receive treatment in an appropriate time through self-diagnostic software. However, security and reliability are critical because these systems are vulnerable to cyber attacks, especially manufacturing equipment and medical devices.

The reliability of the software is the primary measure by which consumers and

developers can determine the operational stability of the software. To estimate the reliability of software, a software reliability growth model (SRGM) is used as a tool. The homogenous Poisson process (HPP) is followed if the failure rate does not change over time of use and remains constant. However, in general, the probability of a product failing increases as its usage time increases, and the quality decreases. This is the case for most software and follows the non-homogenous poisson process (NHPP). The software reliability growth models have various forms depending on the environment of the software and the assumptions considered, expressed in m(t), a unique mean value function.

In this paper, we introduce different types of software reliability models and propose a sequential probability ratio test as a technique for determining software reliability. For most existing software reliability growth models, it is assumed that the test time is theoretically the same as the designed test time and the actual test time. However, if an error occurs in the code responsible for the test, the time between the theoretical test time and the actual test time differ because the error is delayed in correcting the code. Therefore, in this paper, we introduce a software reliability growth model that assumes that test debugging is delayed due to syntax errors that occur in the test code.

Furthermore, most software reliability growth models based on the non-homogeneous poisson process assume that failures occur independently. However, sometimes software failures occur dependently. For example, if an error occurs in a particular class within the code implementing the software, it also occur in other classes referencing that class, which lead to a chain of failures. In this case, the failure will occur dependent. Therefore, in this paper, we introduce a software reliability growth model that assumes such a subordinate failure.

In this paper, we propose a sequential probability ratio test (SPRT) as a statistical technique for determining software reliability. To demonstrate the efficiency of the SPRT, we first apply the SPRT to the parameters of the model to anlaysis the parameter sensitivity. We determine the reliability of the dataset by applying SPRT for the remaining parameters except for sensitive parameters. We also compare the SPRT results based on the level of to estimate the reliability of the dataset, and finally propose the level of δ based on this result.

The composition of this paper is as follows. Chapter 2 addresses the general concept of reliability and distribution function of reliability. Chapter 3 introduces the concept of software reliability and the kinds of software reliability models, among which we focus on the non-homogeneous Poisson process model. In particular, we compare criterias by fitting datasets to existing models, along with new types of models, and select optimal models. Chapter 4 addresses the concept of SPRT, and procedures for estimating software reliability using SPRT. Furthermore, after examining the sensitivity of parameters according to the level of δ in a sequential probability ratio test, we propose optimal parameters of model to apply the final SPRT, and we show conclusions and suggestions in Chapter 5.

Ⅰ. 서론

1.1. 연구 배경

빅 데이터(Big data), 머신러닝(Machine learning), 인공지능(Artificial intelligence; AI) 등과 같은 정보 기술이 개발되면서 소프트웨어는 여러 분야에서 다양한 형태로 사용되고 있다. 특히 인터넷과 사물이 결합한 사물인터넷(Internet of Things; IoT)은 IoT 시대를 열어 지난 십여 년간 꾸준히 사용 분야와 형태가 확장됐다. 최근에는 IoT와 AI를 융합한 사물지능(Artificial intelligence of Things; AIoT)이 주목받고 있다. 인공지능을 통해 AIoT 제품은 개인 맞춤형 서비스를 제공하며 가전제품, 자동차, 의료장비 등에 사용되면서 소프트웨어 산업은 호황을 맞이하였다.

코로나19의 확산은 비대면 시대의 문을 열었다. 2020년 11월, 처음으로 G20 정상회담이 비대면 영상 회의로 진행됐으며 코로나19로 위기를 맞은 기업은 방역 및 업무 효율성을 고려하여 자동화 시스템으로 교체하거나 비대면 서비스로 전환하는 추세이다. 단순 업무뿐만 아니라 교육, 공연 심지어는 경매까지 소프트웨어를 통해 온라인에서 진행된다. 이는 다양한 소프트웨어가 개발되는 발판을 마련하였다.

2020년 7월 14일, 정부는 다양한 분야의 스마트 및 비대면 인프라를 구축하기 위해 2021년 디지털 뉴딜(Digital new deal) 정책을 발표하였다(기획재정부, 2020). 디지털 뉴딜 정책은 여러 질환을 진단하는 소프트웨어 개발을 착수하며 IoT 기기를 활용한 노인 건강관리 사업을 시행하고, 중소기업에 비대면 서비스 및 영상 회의를 위한 소프트웨어를 제공하겠다고 계획함에 따라 국내 소프트웨어 산업은 날개를 달게 됐다.

소프트웨어와 다양한 분야가 융합되면서 새로운 산업이 등장했으며 이는 미래 핵심 산업으로 주목받고 있다. 예를 들면 스마트공장은 IoT를 활용하여 공장의 기록과 데이터를 전자문서로 디지털화하고, 제품의 입출고를 실시간으로 관리한다. 스마트 의료는 자가 진단 소프트웨어를 통해 환자 본인이 건강 상태를 확인하고, 적절한 시기에 치료를 받을 수 있도록 도움을 준다. 그러나 이런 시스템은 사이버 공격에 노출되기 쉬우므로 보안과 신뢰성이 매우 중요하며 특히, 제조 장비와 의료기기는 안정적인 운용이 필수적이다.

소프트웨어에서 오류가 발생하면 단순히 서비스가 지연되는 문제뿐만 아니라 인명 사고나 경제적 손실을 초래할 수 있으므로 소프트웨어 신뢰성 확보가 중요하다.

소프트웨어의 오류가 유발한 대표적인 사고로 AT&T(American Telephone and Telegraph)사의 통신 장애 사고와 테락25(Therac-25) 사고가 있다(김종하, 2014). AT&T는 업데이트 과정에서 새로 설치한 소프트웨어의 코드 한 줄에서 발생한 오류가 전체 시스템의 네트워크 장애를 일으켰다. 이는 AT&T뿐만 아니라 다양한 통신 판매업까지 치명적인 경제적 손실을 보았다. 테락25 사고는 소프트웨어의 오류가 유발한 최악의 인명 사고 중 하나로 꼽힌다. 테락25는 캐나다의 ACEL(Atomic Energy of Canada Limited)사가 개발한 선형가속기로 방사선 치료를 위한 의료기기이다. 테락25는 1985년부터 1987년까지 치료 과정에서 방사성 물질 피폭 사고를 유발하여 사망까지 이르는 인명 피해를 줬으며 그 원인은 소프트웨어의 오류로 밝혀졌다.

앞서 언급한 사고 외에도 전 세계적으로 다양한 사고가 발생하였으며 이는 소프트웨어의 신뢰성 및 규격 기준의 필요성을 느꼈으며 이를 계기로 소프트웨어 산업의 관련 규정을 정립하였다. 소프트웨어의 신뢰성은 소비자와 개발자가 소프트웨어의 운용 안정성을 판단할 수 있는 주요 척도이다. 소프트웨어의 신뢰성을 추정하기 위해 소프트웨어 신뢰성 성장 모형(Software reliability growth model; SRGM)이 도구로 사용된다. 사용 시간에 따라 고장률이 변하지 않고, 일정한 값으로 유지되면 동질 포아송 과정(Homogeneous poisson process; HPP)을 따르게 된다. 그러나 일반적으로 제품은 사용 시간이 증가할수록 고장이 발생할 확률은 증가하고, 품질은 저하된다. 대부분의 소프트웨어는 여기에 해당하며 비동질성 포아송 과정(Non-homogeneous poisson process; NHPP)을 따르게 된다.

소프트웨어 신뢰성 성장 모형은 소프트웨어의 환경 및 고려하는 가정에 따라 다양한 형태를 가지고 있으며 이는 고유한 평균값함수(Mean value function)인 $m(t)$로 표현한다. 소프트웨어 신뢰성 성장 모형에 관한 연구는 1960년대부터 시작됐다. Goel 외 (1979a)는 지수분포(Exponential distribution)를 기반으로 하는 Goel-Okumoto 모형을 개발하여 소프트웨어의 고장 수를 추정하였으며 이는 소프트웨어 신뢰성 연구의 초석을 이루었다. 소프트웨어 신뢰성 성장 모형 연구 초기는 다양한 분포와 굴곡 형태의 결함 검출율 함수를 고려하는 연구가 주를 이루었다. Ohba 외(1984b)와 Yamada 외(1984)는 Inflection S-shaped 모형을 제안하였다. Pham 외(1997b)는 지수함수의 결함 검출율 함수를 갖는 비감소 Inflection S-shaped 모형을 제안하였으며 PZ(Pham-Zhang) 모형으로 잘 알려져 있다. 또한 Pham 외(2003a)는 테스팅 커버리지(Testing coverage)를 고려하는 일반화 된

NHPP 소프트웨어 신뢰성 성장 모형을 제안하였다.

2000년대에 들어서면서 소프트웨어 신뢰성 성장 모형의 연구는 함수의 분포와 모양뿐만 아니라 다양한 환경을 고려하는 형태로 확장되었다. 예를 들면 소비자가 실질적으로 사용하는 운용 환경(Operating environment)은 다양한 하드웨어, 운영체제, 소프트웨어 등을 포함하고 있으나 테스트 환경은 통제된 환경이기 때문에 운용 환경과 상이하다. 이에 대하여 어떤 분포를 갖는 확률변수를 모형의 모수로 도입하여 운용 환경을 설명하는 모형에 관한 연구가 활발히 진행되었다. Teng 외(2006)는 운용 환경의 불확실성을 고려하는 소프트웨어 신뢰성 성장 모형을 제안하였다. Pham(2014b)은 loglog 결함 검출율 함수를 갖는 테스팅 커버리지 신뢰성 성장 모형에 불확실한 운용 환경을 고려하여 확장하였다. Inoue 외(2016)는 불확실한 운용 환경뿐만 아니라 변화점(Change-point)을 고려한 신뢰성 성장 모형을 제시하였다. 이 밖에도 Chang 외(2014), Song 외(2017a, 2017b, 2017c, 2019)에서 운용 환경의 불확실성을 고려하는 다양한 모형을 제안하였다.

최근에는 머신러닝 및 다양한 기법을 활용한 소프트웨어 신뢰성 연구가 활발하게 이루어지고 있다. Caiuta 외(2017)는 메타러닝(Meta learning)을 활용하여 소프트웨어 신뢰성 성장 모형을 분류 및 선택하는 방법을 연구하였고, Tamura 외(2016a, 2016b)는 딥 러닝을 기반으로 최적의 릴리스 정책(Release policy)과 모형을 선택하는 연구를 제안하였다. Wang 외(2018)는 순환신경망(Recurrent nueral network; RNN)을 활용하여 소프트웨어 신뢰성을 추정하는 모형을 제안하였다. 김윤수 외(2020)는 심층신경망(Deep nueral network; DNN)을 적용한 모형을 제안하였으며 NHPP를 기반으로 하는 기존의 모형과 적합도를 비교하였다.

그 밖에는 소프트웨어 신뢰성을 예측하는 통계적 기법을 제안하는 연구가 있는데 Minamino 외(2016)는 변화점을 고려하는 소프트웨어 신뢰성 모형을 제안하였으며 Rani 외(2019)는 변화점과 불완전한 디버깅을 고려하는 신뢰성 모형을 개발하였다. Zeephongsekul 외(2016)는 최대 우도 추정법(Maximum likelihood estimation)을 사용하여 NHPP 소프트웨어 신뢰성 모형의 모수를 추정하였다. 신뢰성 성장 모형의 모수를 추정하는데 다양한 알고리즘이 사용된다. Cadini 외(2016)는 베이지안 몬테칼로 알고리즘(Bayesian monte carlo algorithm)을 적용한 고장률 추정을 제안하였으며 Yaghoobi(2020)은 기존의 차등 진화(Differential evolution; DE) 알고리즘의 성능을 향상한 수정된 차등 진화(Modified differential evolution; MDE) 알고리즘을 소프트웨어 신뢰성 연구에 적용하였다.

본 연구에서는 새로운 유형의 소프트웨어 신뢰성 성장 모형을 소개하고, 신뢰성을 추정하기 위한 통계적 기법을 제안하고자 한다. 순차적 확률비 검정(Sequential probability ratio test; SPRT)은 1943년 컬럼비아 대학교에서 통계 연구 모임을 담당하던 Wald(1943, 1947)가 고안한 통계 추론 시스템이며 후에 육·해군의 새로운 기술을 분석하는 통계 보고서를 작성하는데 이 기법을 제안하면서 주목을 받게 되었다. 순차적 확률비 검정은 데이터가 수집되는 즉시 검정하여 결론을 내리는 통계적 프로세스로 고전적인 가설 검정 방법보다 훨씬 적은 데이터로 검정을 할 수 있어 경제적인 측면에서 혁신적인 기법으로 주목받았다.

Stieber(1997)는 처음으로 소프트웨어 신뢰성을 추정하는데 순차적 확률비 검정의 접근을 제안하였다. 순차적 확률비 검정의 가설과 검정통계량을 소프트웨어 신뢰성 성장 모형의 관점에서 재정의하였다. Prasad 외(2013)는 순차적 확률비 검정을 Inflection S-shaped 모형에 적용하여 소프트웨어 신뢰성을 추정하였으며 Gutta 외(2014)와 Kotha 외(2014)는 Preto type 모형에 순차적 확률비 검정을 적용하였고, Smitha 외(2014)와 Murali Mohan 외(2015)는 Burr type 모형에 순차적 확률비 검정을 적용하여 소프트웨어 신뢰성을 판단하였다. 순차적 확률비 검정을 적용하기 위해선 등간척도를 구성하는 δ와 위험확률(Risk's probability)인 α, β의 가정을 필요로 한다. 특히 δ는 테스터가 주관적으로 결정하기 때문에 결과에 민감한 영향을 미칠 수 있다. 예를 들어 δ의 수준에 따라 시스템이 채택에서 기각으로 혹은 기각에서 채택으로 신뢰성의 판단이 번복된다면 결과를 신뢰할 수 없으며 δ의 수준에 따른 왜곡이 우려된다. 그러나 지금까지 연구된 순차적 확률비 검정을 적용한 소프트웨어 신뢰성 연구를 살펴보면 δ에 대해 객관적인 기준이 제시되어 있지 않다. 실질적인 개발 환경에서 순차적 확률비 검정을 통해 소프트웨어 신뢰성을 효율적으로 판단하기 위해서는 이에 관한 연구가 선행되어야 한다.

1.2. 연구 방법 및 내용

본 논문에서는 기존과 다른 유형의 소프트웨어 신뢰성 모형을 소개하며 소프트웨어 신뢰성을 판단하는 기법으로 순차적 확률비 검정을 제안하고자 한다. 기존에 사용되는 대부분의 소프트웨어 신뢰성 성장 모형에서 테스트 시간 t는 이론적으로 설계한 테스트 시간과 실질적인 테스트 시간이 동일함을 가정한다. 그러나 테스트를 담당하는 코드에 오류가 발생하면 코드를 수정하는데 시간이 지연되므로 이론적인 테스트 시간과 실질적인 테스트 시간은 서로 상이하다. 따라서 본 논문에서는 테스트 코드에서 발생한 구문 오류로 인해 테스트 디버깅이 지연됨을 가정하는 소프트웨어 신뢰성 성장 모형을 소개한다.

비동질성 포아송 과정을 기반으로 하는 대부분의 소프트웨어 신뢰성 성장 모형은 고장이 독립적으로 발생한다고 가정한다. 하지만 때때로 소프트웨어의 고장은 종속적으로 발생한다. 예를 들어 소프트웨어를 구현하는 코드 내에서 특정 클래스에 에러가 발생한 경우, 해당 클래스를 참조하는 다른 클래스에도 에러가 발생할 수 있으며 이는 연쇄적인 고장을 초래할 수 있다. 이 경우, 고장은 종속적으로 발생하게 된다. 본 논문에서는 이처럼 종속적으로 고장이 발생함을 가정하는 소프트웨어 신뢰성 성장 모형을 소개한다.

본 논문에서는 소프트웨어 신뢰성을 판단하는 통계적 기법으로 순차적 확률비 검정을 제안한다. 순차적 확률비 검정의 효율성을 입증하기 위해 먼저 모형의 모수에 순차적 확률비 검정을 적용하여 모수별 민감도를 파악한다. 민감한 모수를 제외한 나머지 모수에 대하여 순차적 확률비 검정을 적용해 데이터 세트의 신뢰성을 판단한다. 또한 등간척도를 구성하는 δ의 수준에 따른 검정 결과를 비교하며 데이터 세트의 신뢰도를 추정하고, 최종적으로는 이 결과를 기반으로 δ의 수준을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 일반적인 신뢰성의 개념과 신뢰성의 분포함수 종류를 다룬다. 3장에서는 소프트웨어에서 사용되는 신뢰성의 개념과 소프트웨어 신뢰성 모형의 종류를 소개하며 그 중 비동질성 포아송 과정 모형을 중점적으로 다룬다. 특히 새로운 유형의 모형 소개와 더불어 기존의 모형에 데이터 세트를 적합하여 척도를 비교하고, 최적의 모형을 선택한다. 4장에서는 순차적 확률비 검정 개념, 순차적 확률비 검정을 이용한 소프트웨어 신뢰성 추정의 절차를 다룬다. 또한 순차적 확률비 검정에서 δ의 수준에 따른 모수의 민감도를 살펴본 후, 최종적인 검정에서 통계량을 구성할 모형별 최적의 모수를 제안하며 검정 결과를 다루고, 5장에서는 결론 및 제언을 보인다.

Ⅱ. 신뢰성

2.1. 신뢰성 개념 및 역사

사용자는 제품, 장비, 시스템 등을 최대한 오랫동안 안정적으로 사용하길 원한다. 저렴한 제품이든 비싼 제품이든 성능이 아무리 뛰어나도 사용자가 기대한 만큼의 시간 동안 사용할 수 없다면 제품의 가치는 떨어지게 된다. 예컨대 수천만 원에서 수억 원씩 하는 자동차에 결함이 발생하여 운행이 어렵거나 원인 모를 화재가 발생한다면 이는 사용자의 안전 문제와 직결되므로 제품에 대한 소비자의 신뢰도는 추락하게 된다. 소비자는 신뢰도 있는 제품을 구매하기 위해 가격 및 성능뿐만 아니라 제품의 보증 기간, 신뢰성과 같은 안정적인 운용을 설명하는 척도를 기준으로 제품의 품질을 판단한다. 일반적으로 신뢰성 공학(Reliability engineering)에서 장비 또는 시스템이 주어진 시간 내에 목적을 달성할 확률, 설계한 대로 작동할 확률 등과 같이 정의하며 국제 표준화 기구(International Organization for Standardization; ISO)에서는 “주어진 환경 및 작동 조건에서 명시된 기간 동안 필요한 기능을 수행하는 능력”으로 정의한다(ISO, 1986). 최근에는 가용성(Availability), 신인성(Dependability), 안전성(Safety), 보전성(Maintainability) 등의 개념을 포함하는 신뢰성 연구가 진행되고 있다.

신뢰성 연구의 역사를 살펴보면 1930년대의 장비나 시스템에 관하여 다양한 수명 분포 연구를 시작됐다. 스웨덴의 물리학자인 Waloddi Weibull이 와이블분포(Weibull distribution)를 이용해 부품의 파괴 강도를 설명하였고, 와이블분포는 신뢰성 공학의 주요 분포로 자리 잡았다(Weibull, 1951). 1910년대부터 1940년대까지는 제1차 세계대전과 제2차 세계대전을 겪으면서 비행기 산업 시장이 폭발적으로 성장하였다. 이때 누적된 고장 데이터를 기반으로 비행기에 대한 신뢰도 기준이 정립되었으며 신뢰성 연구 도약의 계기를 마련하였다(정해성 외, 2003). 1960년대~1970년대에는 제2차 세계대전이 끝난 뒤인 냉전 시대로 미국과 소련의 우주 경쟁이 활발하던 시기이다. 소련이 인공위성 스푸트니크(Sputnik) 1호를 세계 최초로 발사하면서 미국은 소련을 견제하기 위해 미국 항공우주국(National Aeronautics and Space Administration, NASA)을 창설하였고, 인공위성 개발 연구를 위해 대규모의 국가 예산을 투입하였다. 개발 과정에서 인공위성의 신뢰도를 최대치로 끌어올리기 위한 노력 끝에 주요 신뢰성 이론과 기법이 개발되었으며 결과물은 현재까지도 신뢰성 공학에 많은 영향을 끼치고 있다.

2.2. 신뢰성의 분포 함수

2.2.1. 신뢰성 함수와 특징

신뢰성 공학은 통계적 이론에 의존적이며 대부분은 부품의 고장 시간에 대한 분포를 모형화하거나 시간을 확률 변수로 가정한 모형화를 중점적으로 다룬다. 분포 함수는 함수의 특성을 나타내는 모수(Parameter)를 갖는다. 모수는 매개변수로 불리기도 하며 위치모수(Location parameter), 척도모수(Scale parameter), 형상모수(Shape parameter)로 분류할 수 있다.

신뢰성을 설명하는 함수로 확률밀도함수(Probability density function), 누적분포함수(Cumulative distribution function), 신뢰도함수(Reliability function), 고장률함수(Failure rate function)가 있다. 고장 시간이나 수명을 나타내는 T를 확률변수로 가정하여 함수들을 이용해 T의 변화에 따른 고장 수나 고장 확률을 추정한다.

신뢰도함수는 생존 함수(Survival function)라고도 불리며 시스템의 작동 시간이 t이상이 될 확률로 정의된다. 신뢰도함수 수식은 다음과 같이 나타낸다.

$$\begin{aligned} R(t) = \Pr[T > t] &= \int_t^{\infty} f(s)ds \\ &= 1 - \int_0^t f(s)ds \\ &= 1 - F(t). \end{aligned}$$

여기서 $F(t)$는 누적분포함수이다.

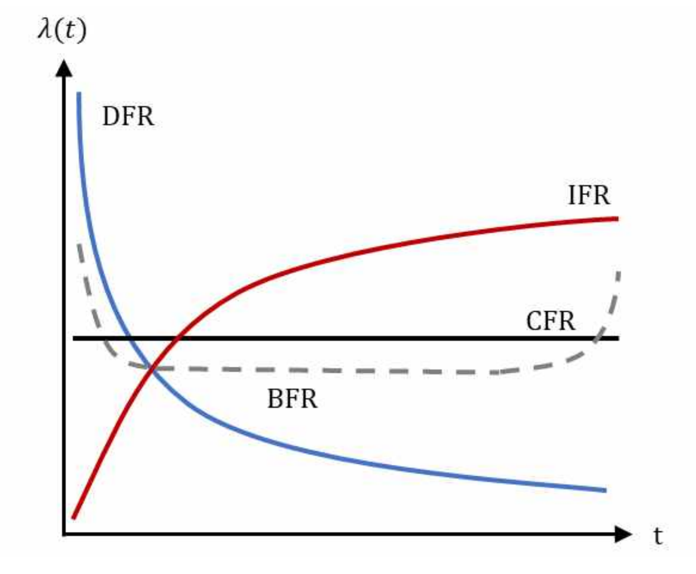
고장강도함수(Failure intensity function)는 위험률 함수(Hazard rate function)라고도 일컬으며 시점 t 직후에 고장을 일으킬 수 있는 단위 시간에 대한 조건부 평균 고장 비율로 정의하고, 수식은 다음과 같이 나타낸다.

$$\begin{aligned} \lambda(t) &= \lim_{\Delta t \to 0} \frac{P[t < T \leq t + \Delta t | T > t]}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{F(t+\Delta t) - F(t)}{\Delta t} \frac{1}{R(t)} \\ &= \frac{f(t)}{R(t)} \\ &= \frac{f(t)}{1 - F(t)}. \end{aligned}$$

고장강도함수는 형태에 따라 다음과 같이 분류할 수 있다(Hamada 외, 2008).

1. 증가함수 형태(Increasing failure rate; IFR) : 시간에 따라 순간 고장률이 증가하며 고장 수도 증가한다.
2. 감소함수 형태(Decreasing failure rate; DFR) : 시간에 따라 순간 고장률이 감소하며 고장 수도 감소한다.
3. 욕조형(Bathtub failure rate; BFR) : 초기에는 순간 고장률이 높으며 일정 기간 순간 고장률이 감소한 상태로 유지됐다가 다시 증가하는 형태를 띤다.
4. 일정한 고장률(Constant failure rate; CFR) : 순간 고장률이 일정하게 유지되며 고장 수도 비교적 일정하게 유지된다.

시스템의 전체 수명 주기에 대하여 DFR은 초기 고장기에 해당하며 CFR은 우발 고장기, IFR은 마모 고장기에 해당한다. 시스템의 수명 주기에 따라 고장률의 형태가 변하기 때문에 각 주기에 따른 적절한 대응책을 세운다면 제품의 신뢰성을 향상하는 데 큰 도움이 된다. <그림 Ⅰ-1>은 유형별 고장강도함수를 나타낸다.



<그림 Ⅰ-1> 고장강도함수의 형태에 따른 분류

평균 수명은 시스템에서 고장이 발생할 때까지의 평균 시간을 의미하며 시스템의 수리 가능 여부에 따라 수리가 가능한 시스템의 평균수명시간(Mean time between failure; MTBF), 수리할 수 없는 시스템의 평균수명시간(Mean time to failure; MTTF)으로 분류한다. 평균고장시간(MTTF)은 다음과 같은 수식으로 표현한다.

$$MTTF = \int_0^\infty tf(t)dt.$$

이 밖에도 시스템의 신뢰성을 설명하는데 백분위수(Percentile), 평균잔여수명(Mean residual life) 등이 사용된다.

고장 데이터도 여러 형태로 분류할 수 있는데 그중 가장 간단한 데이터 형태는 성공(Pass)과 실패(Fail)로만 분류하는 베르누이(Bernouii) 데이터가 있으며 그 밖에도 고장 수 데이터(Failure count data), 고장 시간 데이터(Failure time data), 열화 데이터(Degradation data)가 있다. 데이터의 수집 형태에 따라서는 모든 표본으로부터 관측하여 얻은 완전 데이터(Complete data)와 데이터 수집이 도중에 중단되는 불완전 데이터(Incomplete data)로 분류한다. 모든 표본에 대해서 목표 시간까지 데이터를 수집하는 것은 어려우므로 완전 데이터보다는 불완전 데이터가 많이 사용된다. 불완전 데이터는 데이터 수집 중단 시점의 속성에 따라 정시 중단 데이터(Type I censoring data), 정수 중단 데이터(Type II censoring data), 임의 중단 데이터(Random censoring data)로 분류할 수 있다(Epstein 외, 2008).

본 논문에서는 보편적으로 사용되는 몇 가지의 분포를 소개하며 그 밖의 다양한 분포함수와 시스템의 분류에 따른 수명 분포는 Bain(1978), Cohen 외(1988), Pham(2007), 박동호 외(2015)에서 기술하고 있다.

2.2.2. 고장분포

2.2.2.1. 지수분포

지수분포(Exponential distribution)는 무기억성(Memoryless property)을 갖는 연속 확률 분포로 신뢰성 공학에서 중요한 역할을 하고 있다. 무기억성은 어떠한 시점부터 소요되는 시간이 과거의 시간에 영향을 받지 않음을 의미한다. 즉, 지수분포를 따르는 제품의 잔여 수명은 과거의 사용 시간은 고려하지 않고, 새 제품을 기준으로 남은 수명을 추정할 수 있다. 신뢰성 공학에서는 이를 노화 영향(Aging effect)이 없다고 표현한다. 지수분포를 따르는 제품은 시간 t의 영향을 받지 않기 때문에 고장강도함수 $\lambda(t)$는 상수와 같으며 유일한 CFR 분포이다(Barlow 외, 1996).

Drenick(1960)은 충분히 오랜 시간 동작하고, 상당히 크고 다양한 부품으로 조립됐다는 조건을 전제로 시스템의 고장 발생 간격이 지수분포를 따른다는 것을 증명하였다. 지수분포의 확장된 형태로 감마분포(Gamma distribution), 와이블분포(Weibull distribution)가 있다.

지수분포의 확률밀도함수는 다음과 같으며 평균은 $\dfrac{1}{\lambda}$이고, 분산은 $\dfrac{1}{\lambda^2}$이다.

확률밀도함수 : $f(t) = \lambda e^{-\lambda t},\ t > 0.$

지수분포의 신뢰도 관련 함수는 다음과 같다.

신뢰도함수 : $R(t) = e^{-\lambda t}$

누적분포함수 : $F(t) = 1 - e^{-\lambda t}$

고장강도함수 : $\lambda(t) = \lambda,\ \lambda > 0$

2.2.2.2. 감마분포

감마분포(Gamma distribution)는 지수분포의 확장된 형태로 신뢰성 공학에서 많이 쓰이는 분포이다. 만약 어떤 제품의 고장 발생 시간이 $\exp(\lambda)$를 따른다면 이때 단위 시간 t동안 발생하는 고장 수의 분포는 감마분포를 따르게 된다.

감마분포는 두 개의 모수인 α와 β를 가지며 그 확률밀도함수는 다음과 같다.

확률밀도함수 : $f(t) = \dfrac{1}{\Gamma(\alpha)\beta^{\alpha}} t^{\alpha-1} e^{-\frac{t}{\beta}}$, $t > 0,\ \alpha, \beta > 0.$

감마분포의 평균은 $\alpha\beta$이며 분산은 $\alpha\beta^2$이다. 여기서 α는 형상모수(Shape parameter), β는 척도모수(Scale parameter)이다. α값에 따라 다음과 같이 세 가지의 고장 확률 분포로 분류된다. 특히 감마분포에서 $\alpha = 1$이면 지수분포와 같은 확률밀도함수를 갖게 된다.

DFR : $0 < \alpha < 1$

CFR : $\alpha = 1$

IFR : $\alpha > 1$

감마분포의 신뢰도 관련 함수는 다음과 같다.

신뢰도함수 : $R(t) = \exp\left(-\dfrac{t}{\beta}\right) \sum_{i=0}^{\alpha-1} \left(\dfrac{t}{\beta}\right)^i \dfrac{1}{i!}$

누적분포함수 : $F(t) = 1 - \exp\left(-\dfrac{t}{\beta}\right) \sum_{i=0}^{\alpha-1} \left(\dfrac{t}{\beta}\right)^i \dfrac{1}{i!}$

고장강도함수 : $\lambda(t) = \dfrac{\dfrac{t^{\alpha-1}}{\beta^{\alpha}\Gamma(\alpha)} \exp\left(-\dfrac{t}{\beta}\right)}{\exp\left(-\dfrac{t}{\beta}\right) \sum_{i=0}^{\alpha-1} \left(\dfrac{t}{\beta}\right)^i \dfrac{1}{i!}}$

2.2.2.3. 와이블분포

와이블분포(Weibull distribution)는 지수분포의 확장된 형태이자 레일리분포(Rayleigh distribution)의 일반화 한 형태로 신뢰성 공학에서 지수분포, 감마분포와 함께 상당히 많이 쓰이는 분포이다. 감마분포와 함께 고장강도함수의 세 가지 유형(DFR, CFR, IFR)을 모형화하기 위해 많이 쓰인다.

와이블분포는 세 개의 모수 γ, β, θ를 가지며 그 확률밀도함수와 특성은 다음과 같다.

확률밀도함수 : $f(t)=\left(\frac{\beta}{\theta^{\beta}}\right)(t-\gamma)^{\beta-1}\exp\left[-\left(\frac{t-\gamma}{\theta}\right)^{\beta}\right]$, $0 \leqq \gamma \leqq t,\ \ \theta,\beta > 0.$

와이블분포의 평균은 $\dfrac{\Gamma\left(1+\frac{1}{\beta}\right)}{\theta}$이며 분산은 $\dfrac{1}{\theta^2}\left[\Gamma\left(1+\frac{2}{\beta}\right)-\left(\Gamma\left(1+\frac{1}{\beta}\right)\right)^2\right]$이다. 여기서 γ는 위치모수(Location parameter)이며 β는 형상모수, θ는 척도모수이다. 보통 γ는 0으로 설정하고, 나머지 두 개의 모수 β와 θ로 구성된 와이블분포가 많이 쓰인다. 와이블분포는 β의 값에 따라 고장 확률 분포의 유형이 구분된다.

DFR : $0 < \beta < 1$

CFR : $\beta = 1$

IFR : $\beta > 1$

또한, 특수한 형태로 $\beta = 3.44$일 때는 정규분포에 근사하며 $\beta = 2$인 경우 레일리분포와 같다. 와이블분포의 신뢰도 관련 함수는 다음과 같다.

신뢰도함수 : $R(t)=\exp\left[-\left(\frac{t-\gamma}{\theta}\right)^{\beta}\right]$

누적분포함수 : $F(t)=1-\exp\left[-\left(\frac{t-\gamma}{\theta}\right)^{\beta}\right]$

고장강도함수 : $\lambda(t)=\dfrac{\beta(t-\gamma)^{\beta-1}}{\theta^{\beta}}$

2.2.2.4. 레일리분포

레일리분포(Rayleigh distribution)는 와이블분포에서 $\gamma=0$으로 두 개의 모수를 갖는 특수한 형태로 1880년 영국 물리학자 Rayleigh가 제안하였다. 레일리분포는 주로 음향, 전자파 특징의 모형화를 위해 사용됐으나 신뢰성 공학에서는 고장 확률이 점차 증가하는 장비의 모형화에 적합하여 자주 사용되고 있다.

레일리분포의 확률밀도함수와 특성은 다음과 같다.

확률밀도함수 : $f(t)=\frac{t}{\sigma^2}\exp\left[-\frac{t^2}{2\sigma^2}\right]$

레일리분포의 평균은 $\sigma\sqrt{\frac{\pi}{2}}$ 이며 분산은 $\frac{4-\pi}{2}\sigma^2$이다.

레일리분포의 신뢰도 관련 함수는 다음과 같다.

신뢰도함수 : $R(t)=\exp\left[-\frac{\sigma t^2}{2}\right]$

누적분포함수 : $F(t)=1-\exp\left[-\frac{\sigma t^2}{2}\right]$

고장강도함수 : $\lambda(t)=\frac{t}{\sigma^2}$

2.2.2.5. 정규분포

정규분포(Normal distribution)는 통계학 및 자료 분석 측면에서 가장 골간적이다. 표본의 개수가 충분히 많은 경우, 중심 극한 정리(Central limit theorem)에 의하여 정규분포를 따르게 되므로 표본의 특성을 이용해 모수를 추정할 수 있다. 또한, 표준화가 쉬워 서로 다른 분포를 비교하기 수월하다.

정규분포의 확률밀도함수와 특성은 다음과 같다.

확률밀도함수 : $f(t)=\frac{1}{\sqrt{2\pi}\,\sigma}\exp\left[-\frac{(t-\mu)^2}{2\sigma^2}\right],\ -\infty<t<\infty.$

정규분포의 평균은 μ이며 분산은 σ^2이다. 여기서 μ는 위치모수, σ^2은 형상모수이다. 정규분포를 표준화한 표준정규분포(Standard normal distribution)는 $\Phi(z)=\frac{1}{\sqrt{2\pi}}\exp\left[-\frac{z^2}{2}\right]$이며 평균은 0이고 분산은 1이다.

정규분포의 신뢰도 관련 함수는 다음과 같다.

신뢰도함수 : $R(t)=\Phi\left(\frac{t-\mu}{\sigma}\right)$

누적분포함수 : $F(t)=1-\Phi\left(\frac{t-\mu}{\sigma}\right)$

고장강도함수 : $\lambda(t)=\frac{\left(\frac{1}{\sigma}\right)\left(\frac{t-\mu}{\sigma}\right)}{1-\Phi\left(\frac{t-\mu}{\sigma}\right)}$

제품의 고장강도함수가 IFR 분포를 따르면 고장 확률밀도함수는 정규분포를 따르게 된다.

2.2.2.6. 로그정규분포

로그정규분포(Log normal distribution)는 정규분포와 흡사한 형태를 보이고 있으며 대칭이거나 한쪽으로 치우친 고장분포에 적합하다. 부식이나 전이, 스트레스나 피로도가 고장의 원인이거나 특정 시간이 지나면 고장 확률이 감소하는 부품을 모형화하는데 쉽다.

로그정규분포의 확률밀도함수와 특성은 다음과 같다.

확률밀도함수 : $f(t) = \frac{1}{\sqrt{2\pi}\sigma t}\exp\left[-\frac{(\ln t-\mu)^2}{2\sigma^2}\right]$, $-\infty < t < \infty$.

로그정규분포의 평균은 $\exp\left[\mu+\frac{\sigma^2}{2}\right]$이고, 분산은 $\exp(2\mu+\sigma^2)(\exp(\sigma^2)-1)$이다.

로그정규분포의 신뢰도 관련 함수는 다음과 같다.

신뢰도함수 : $R(t) = 1-\Phi\left(\frac{\ln t-\mu}{\sigma}\right)$

누적분포함수 : $F(t) = \Phi\left(\frac{\ln t-\mu}{\sigma}\right)$

고장강도함수 : $\lambda(t) = \frac{\left(\frac{1}{\sigma}\right)\left(\frac{\ln t-\mu}{\sigma}\right)}{1-\Phi\left(\frac{\ln t-\mu}{\sigma}\right)}$

Ⅲ. 소프트웨어 신뢰성

3.1. 소프트웨어 개발

과거에는 하드웨어의 중요성이 컸으며 소프트웨어는 부수적인 기능을 제공하는 정도로만 여겨졌으나 현재는 코로나19의 여파로 소프트웨어를 통해 재택근무를 하고, 학교 수업을 듣는 등 비대면 서비스의 핵심적인 역할을 하고 있다. 코로나19가 종식돼도 기업은 소프트웨어를 활용한 서비스의 비대면화를 지속적으로 지원할 것으로 기대되며 소프트웨어 시장의 규모는 더 커질 것으로 전망된다. 그러나 소프트웨어 산업이 주목받는다고 무분별하게 소프트웨어를 과잉생산할 경우, 제품의 저품질이 우려된다. 다양한 산업과 결합한 소프트웨어는 오류 발생 시 경제적 손실뿐만 아니라 인명 피해까지 초래할 수 있으므로 소프트웨어의 주요 기능뿐만 아니라 신뢰성 확보는 필수 불가결이다.

일반적으로 소프트웨어는 <그림 Ⅲ-1>과 같은 개발 과정을 거치게 된다(Singpurwalla 외, 2012).

<그림 Ⅲ-1> 소프트웨어 개발 과정

소프트웨어의 개발 과정은 소프트웨어의 완성도를 높이기 위해 5단계로 나눠 체계적으로 개발 및 관리한다.

ⅰ) 요구사항 분석(Requirements analysis)

사용자의 요구사항에 따라 소프트웨어 개발의 목적, 주요 기능, 사용 조건 및 환경 등을 검토 및 조율하여 이를 정의하는 요구 명세서를 작성한다. 요구 명세서를 기반으로 소프트웨어의 개발 방법과 개발 자원 및 비용을 예측한다.

ⅱ) 설계(Design)

소프트웨어의 구조(Architecture)를 설계하는 단계로 내부 모듈 간의 관계와 구조, 각 모듈의 알고리즘을 정의하기 위한 소프트웨어 설계도와 프로토콜 설계도를 작성한다. 데이터베이스를 설계하고, 소프트웨어 사용자의 인터페이스(Interface)를

정의하여 UI 설계도를 작성한다. 소프트웨어 구현을 위해 알고리즘과 자료구조를 구체화한다.

iii) 구현(Programming)

소프트웨어를 구현할 언어와 개발 환경을 선택하여 코딩한다. 구현과 사후관리가 쉽도록 간결하게 코드를 작성한다.

iv) 테스팅(Testing)

소프트웨어가 주어진 기능을 제대로 수행하는지를 확인하기 위해 검증하는 단계이며 단위 모듈을 테스트하는 단위 테스팅, 모듈을 통합하여 테스트하는 통합 테스팅, 요구 명세서와 일치하는지를 테스트하는 적합성 테스팅, 전반적인 시스템의 성능과 보안 및 복구 등을 테스트하는 시스템 테스팅의 단계를 거친다. 각 단계의 테스팅을 거쳐 최대한의 오류를 발견 및 수정하여 소프트웨어의 품질과 신뢰성을 향상한다.

v) 유지보수(Maintenance)

사용자의 추가 요구사항이나 수정할 사항을 반영하며 운용 과정에서 소프트웨어에 오류나 장애가 발생 시 수정 및 복구, 보고를 하는 등의 지속적인 모니터링을 수행한다.

3.2. 소프트웨어 신뢰성 모형

3.2.1. 오류 유입 모형

오류 유입(Error seeding)은 오류를 기반으로 하는 테스팅 기법이다. 오류는 조작하지 않은 프로그램 고유 오류(Indigenous error)와 테스팅 과정에서 인위적으로 삽입된 유입 오류(Seeding error)로 구분된다. 인위적인 오류를 프로그램 내에 삽입하고, 테스트를 통해 이 오류의 검출을 관찰하여 인위적인 오류의 검출율을 기반으로 프로그램의 잔여 결함 수를 추정한다. 오류 유입 모형의 기본적인 가정은 다음과 같다.

i) 삽입된 오류는 프로그램의 고유 오류를 대표할 수 있어야 한다.

ii) 테스트 환경이 운용 환경을 대표할 수 있어야 한다.

iii) 삽입된 오류의 정보가 프로그램 내에 알려져서는 안 된다. 프로그램 내에 삽입된 정보가 알려질 경우, 삽입된 오류를 목표(Target)로 하여 프로그램이 수리될 수 있기 때문이다.

iv) 오류를 삽입하기 위한 소스 코드의 가용성을 만족해야 한다.

그러나 실질적으로 위 가정을 만족하기 어려우므로 다른 기법으로 대체하는 경우가 많다. 오류 유입 모형의 대표적인 예로 Mills의 오류 유입 모형(Mills, 1970), Cai’s 모형(Cai, 1998), Tohma의 초기하분포 모형(Tohma 외, 1991)이 있다.

Mills의 오류 유입 모형은 프로그램에 오류를 삽입하여 소프트웨어의 오류 수를 추정하는 오류 시딩 방법을 고려하였다. 고유 오류와 유입 오류로 구성된 디버깅 데이터로부터 알려지지 않은 고유 오류 수를 추정할 수 있다.

Cai의 모형은 Mills의 모형을 수정한 형태로 소프트웨어를 두 부분으로 나누어 소프트웨어의 잔여 오류 수를 추정하는 데 사용된다.

Tohma의 초기하분포 모형은 초기하분포를 기반으로 하며 테스트 또는 디버깅 프로세스를 시작할 때, 초기에 프로그램 내에 존재하는 결함 수를 추정하기 위한 모형이다.

3.2.2. 고장률 모형

고장률(Failure rate) 모형은 고장 간격에서 결함당 프로그램 고장률과 그 변화를 연구하는 데 사용된다. 잔여 결함 수가 변경되면 고장률도 함께 변경된다. 결함 수는 이산변수(Discrete variable)이므로 고장률도 고장 시간에 불연속적인 이산함수이다. 고장률 모형으로는 Jelinski and Moranda 모형(Jelinski 외, 1972), Modified Schick and Wolverton 모형(Sukert, 1977), Schick and Wolverton 모형(Schick 외, 1978), Goel and Okumoto imperfect debugging 모형(Goel, 1979b) 등이 있으며 <표 III-1>에서는 대표적인 고장률 모형과 모형식을 나타내고 있다.

<표 Ⅲ-1> 고장률 모형 예시

모형	모형식
J-M(Jelinski and Moranda) 모형 (Jelinski 외, 1972)	$\lambda(t_i)=\phi[N-(i-1)]$
S-W(Schick and Wolverton) 모형 (Schick 외, 1978)	$\lambda(t_i)=\phi[N-(i-1)]t_i$
J-M(Jelinski-Moranda) geometric 모형 (Moranda, 1979)	$\lambda(t_i)=Dk^{i-1}$
Modified Schick-Wolverton 모형 (Sukert, 1977)	$\lambda(t_i)=\phi[N-n_{i-1}]t_i$
Goel-Okumoto imperfect debugging 모형 (Goel and Okumoto, 1979b)	$\lambda(t_i)=\phi[N-p(i-1)]$

<표 III-1>에서 $i=1,2,...,N$이며 N은 프로그램의 초기 고장 수, ϕ는 비례상수, D는 프로그램의 초기 고장률, k는 모수 수를 의미한다. 또한, t_i는 $i-1$번째와 i번째의 고장 간의 시간 간격을 의미하고, p는 결함이 발생 됐을 때 제거될 확률을 의미한다.

앞서 언급한 고장률 모형의 예시 중 가장 대표적인 Jelinski-Moranda(J-M) 모형은 초기에 개발된 소프트웨어 신뢰성 모형 중 하나로 기존의 많은 모형은 J-M 모형을 확장하여 개발됐다.

J-M 모형의 가정은 다음과 같다.

ⅰ) 프로그램 내에는 알려지지 않은 고정 된 상수인 N개의 초기 오류가 존재한다.

ⅱ) 프로그램의 각 오류는 독립적이며 테스트 중에 고장을 일으킬 확률이 동일하다.

ⅲ) 고장 발생 사이의 시간 간격은 서로 독립이다.

ⅳ) 고장이 발생할 때마다 해당 오류는 확실하게 제거된다.

ⅴ) 고장의 원인이 되는 오류는 즉시 제거되며 검출된 오류를 제거하는 도중에 새로운 오류는 발생하지 않는다.

ⅵ) 고장 간격 동안의 소프트웨어 고장률은 일정하며 프로그램의 잔여 오류 수에 비례한다.

기존의 많은 모형이 J-M 모형을 기반으로 확장한 형태이기 때문에 위 가정을 기본으로 요구한다.

3.2.3. 곡선 접합 모형

곡선 적합(Curve fitting) 모형은 통계적인 분석 방법인 회귀분석(Regression analysis)을 사용하여 소프트웨어의 복잡성과 오류 수, 변경 횟수, 고장률 간의 관계를 연구하는 데 쓰인다. 곡선 적합 모형은 선형회귀, 비선형회귀, 시계열분석 방법을 사용하여 종속변수(Dependent variable)와 독립변수(Independent variable) 간의 관계를 파악한다. 예를 들면 종속변수 Y는 오류 수로 정의하고, 독립변수 t는 유지·보수 단계에서 변경된 모듈의 수, 고장 사이의 시간, 프로그래머의 기술, 프로그램 크기 등으로 고려할 수 있다.

단순선형회귀(Simple linear regression) 모형의 수식은 다음과 같이 표현한다.

$$Y = b + at + \epsilon.$$

여기서 a는 기울기, b는 절편, ϵ은 오차항을 의미한다. 그러나 일반적으로 선형보다는 비선형 데이터가 훨씬 많으며 이 경우에는 비선형회귀곡선(Non-linear regressive curve) 모형을 사용한다. 자주 사용되는 비선형회귀곡선의 예는 <표 III-2>에서 나타낸다.

<표 Ⅲ-2> 비선형회귀곡선 모형 예시

모형	모형식
2차 모형	$Y = b + a_1 t + a_2 t^2 + \epsilon$
3차 모형	$Y = b + a_1 t + a_2 t^2 + a_3 t^3 + \epsilon$
지수 모형	$Y = be^{at} + \epsilon$
대수 모형	$Y = b + a \ln t + \epsilon$
로지스틱 모형	$Y = 1/(k + ba^t) + \epsilon$

그 밖의 곡선 접합 모형에 관한 대표적인 예로는 복잡도 추정 모형(Belady 외, 1976), 고장률 추정 모형(Miller 외, 1985) 등이 있다.

3.2.4. 신뢰성 성장 모형

신뢰성 성장(Reliability growth) 모형은 테스트를 통해 프로그램의 신뢰성을 측정 및 예측한다. 신뢰성 성장 모형은 시간이나 테스트 횟수의 함수로 구현되며 시스템의 신뢰성 혹은 고장률을 나타낸다. <표 III-3>에서는 신뢰성 성장 모형의 예로 Coutinho 모형과 Wall and Ferguson 모형의 모형식을 나타내고 있다.

<표 Ⅲ-3> 신뢰성 성장 모형 예시

모형	모형식
Coutinho 모형(Coutinho 1973)	$\lambda(t)=\frac{N(t)}{t}=\beta_0 t^{-\beta_1}$
Wall and Ferguson 모형(Wall 1977)	$m(t)=\alpha_0[b(t)]^{\beta}$ $\lambda(t)=m'(t)=\alpha_0\beta b'(t)[b(t)]^{\beta-1}$

Coutinho 모형은 소프트웨어 테스트 프로세스를 나타내기 위해 Duane 성장 모형을 적용하였으며 검출된 누적 결함 수 대비 누적 테스트 수에 대한 수정 횟수를 log-log paper에 표시하였다. 여기서 $N(t)$는 누적 고장 수를 나타내고, t는 총 테스트 시간을 의미한다. 여기서 β_0와 β_1는 알려지지 않은 Coutinho 모형의 모수이다.

Wall and Ferguson 모형은 테스트 중 소프트웨어의 고장률을 예측하기 위해 Weibull 성장 모형의 변형된 형태를 제안하였다. α_0와 β는 알려지지 않은 모수이다. $b(t)$ 함수는 테스트 횟수 혹은 총 테스트 시간으로 얻을 수 있으며 평균값함수 $m(t)$를 미분하면 시간 t에서의 고장강도함수인 $\lambda(t)$를 얻을 수 있다.

3.2.5. 시계열 모형

시계열(Time series)은 시간의 흐름에 따른 데이터의 변동을 시각화하고, 수학적인 모형의 적합을 통해 데이터 해석 및 향후 예측하는 통계적 분석 기법이다. 공학, 주가 예측, 환율, 유가 변동 등의 분야에서 많이 사용된다. 시계열의 수학적 모형으로는 자기회귀모형(Autoregressive model; AR), 이동평균모형(Moving average model; MA), 자기회귀이동평균모형(Autoregressive moving average model; ARMA), 자기회귀누적이동평균모형(Autoregressive integrated moving average model; ARIMA)이 있으며 자기상관계수(Autocorrelation; AC)와 편자기상관계수(Partial autocorrelation: PAC)를 통해 모형을 결정한다(Box 외, 2015).

AR 모형은 AC는 점점 작아지는 형태를 보이며 PAC의 돌출된 부분의 수로 차수를 결정한다. MA 모형은 PAC는 점점 작아지는 형태를 보이며 AC의 돌출된 부분의 수로 차수를 결정한다. ARMA 모형은 자기회귀모형과 이동평균모형의 혼합된 형태로 AC와 PAC가 점점 작아지고, AC와 PAC의 유의한 수로 차수를 결정한다. ARIMA 모형은 ARMA 모형을 차분(Differencing)한 형태이다.

AR(p)는 다음과 같은 수식으로 표현한다.

$$Y_t = \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \cdots + \alpha_p Y_{t-p} + \epsilon_t.$$

MA(q)는 다음과 같은 수식으로 표현한다.

$$Y_t = \epsilon_t - \beta_1 \epsilon_{t-1} - \beta_2 \epsilon_{t-2} - \cdots - \beta_q \epsilon_{t-q}.$$

ARMA(p,q)는 AR(p)와 MA(q)의 혼합된 형태이므로 다음과 같이 표현할 수 있다.

$$Y_t = \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \cdots + \alpha_p Y_{t-p} - \beta_1 \epsilon_{t-1} - \beta_2 \epsilon_{t-2} - \cdots - \beta_q \epsilon_{t-q} + \epsilon_t.$$

3.3. NHPP 소프트웨어 신뢰성 모형

3.3.1. 포아송 과정

포아송 과정(Poisson process)은 신뢰성 공학에서 가장 기초적인 확률과정(Stochastic process) 중 하나로 주어진 구간 내의 사건 발생 횟수 n은 포아송분포이고, 발생 횟수가 독립증분(Independent increment) 및 정상성(Stationary increment)을 보이는 확률과정을 의미한다. 이때, 사건 발생 횟수 n은 모수 λ를 갖는 포아송분포를 따르며 다음과 같은 수식으로 표현할 수 있다.

$$\Pr\{N(t)=n\}=\frac{(\lambda(t))^n}{n!}e^{-\lambda(t)},\ n=0,1,2,\ \cdots .$$

포아송 과정의 모수 λ가 시간의 흐름에 따라 변동이 없는 상수라면 동질성 포아송 과정(Homogeneous poisson process; HPP)으로 분류하고, 시간의 흐름에 따라 변화한다면 비동질성 포아송 과정(Non-homogeneous poisson process; NHPP)으로 구분할 수 있다. 대부분의 소프트웨어 신뢰성 성장 모형 연구는 비동질성 포아송 과정을 기반으로 한다.

3.3.1.1. 동질성 포아송 과정

동질성 포아송 과정(HPP)은 포아송 과정에서의 모수인 λ가 시간의 흐름에 따라 변동이 없는 상수로 가정할 때의 확률과정을 의미하며 다음과 같은 가정이 필요하다.

i) 구간 s에 대한 고장 수는 평균이 λs인 포아송분포를 따른다.

ii) 겹치지 않는 시간에 대해 발생한 사건의 수는 독립이다.
즉, $N(t)$와 $N(t+s)-N(t)$는 서로 독립이다. 여기서 $t+s>t$이면 $N(t+s)\geqq N(t)$이다.

iii) 초기값은 $N(0)=0$로 주어진다.

모든 가정이 충족되면 다음과 같은 특성함수를 갖게 된다.

$$\Pr\{N(t+s)-N(t)=n\}=\frac{e^{-\lambda s}(\lambda s)^n}{n!},\ n=0,1,2,\ \cdots .$$

신뢰도함수 : $R(t)=e^{-\lambda t}$

누적분포함수 : $F(t)=1-e^{-\lambda t}$

3.3.1.2. 비동질성 포아송 과정

비동질성 포아송 과정(NHPP)은 포아송 과정에서의 모수인 λ가 시간의 흐름에 따라 변동이 있는 확률과정을 의미하며 다음과 같은 가정이 필요하다.

ⅰ) 구간 s에 대한 고장 수의 평균은 $\int_t^{t+s} \lambda(t)dt,\ (t+s>t)$로 표현한다.

ⅱ) 겹치지 않는 시간에 대해 발생한 사건의 수는 독립이다.
즉, $N(t)$와 $N(t+s)-N(t)$는 서로 독립이다. 여기서 $t+s \geqq t$이면 $N(t+s) \geqq N(t)$이다.

ⅲ) 초기값은 $N(0)=0$로 주어진다.

위와 같은 가정이 충족되면 다음과 같은 특성함수를 갖게 된다.

$$\Pr\{N(t)=n\} = \frac{e^{-m(t)}[m(t)]^n}{n!},\ n=0,1,2,\cdots.$$

신뢰도함수 : $R(t)=e^{-m(t)}$

누적분포함수 : $F(t)=1-e^{-m(t)}$

여기서 $\lambda(t)$는 고장강도함수로 만약 감소함수라면 고장 확률이 감소하여 품질이 향상됨을 의미하고, 증가함수라면 고장 확률이 증가하여 품질이 떨어짐을 의미한다. 고장 수는 독립증분임을 가정하지만, 시간의 간격에 대해서는 독립증분을 가정하지 않는다. 비동질성 포아송 과정은 확률적, 통계적 모형 및 열화를 모형화하여 수학적으로 해결할 수 있어 신뢰성 분야에서 중요한 역할을 하고 있다.

비동질성 포아송 과정 소프트웨어 신뢰성 성장 모형은 시간 t까지 발생한 소프트웨어의 고장 수 $N(t)$가 NHPP를 따른다고 가정한다. NHPP 소프트웨어 신뢰성 성장 모형의 주요 목표는 시간 t까지의 고장 수를 예측하는 것이며 반영하는 가정에 따라 다양한 형태의 평균값함수를 갖게 된다. NHPP를 가정하는 소프트웨어 신뢰성 성장 모형에서 $N(t)$는 평균이 $m(t)$인 포아송분포를 갖게 되며 다음과 같은 수식으로 표현할 수 있다.

$$\Pr\{N(t)=n\} = \frac{[m(t)]^n}{n!}e^{-m(t)}\quad n=0,1,2,\cdots.$$

여기서 평균값함수 $m(t)$는 다음과 같이 고장강도함수 $\lambda(t)$로 표현이 가능하다.

$$m(t)=\int_0^t \lambda(s)ds.$$

NHPP 신뢰성 성장 모형의 신뢰도함수 $R(t)$는 다음과 같이 주어진다.

$$R(t) = e^{-m(t)} = e^{-\int_0^t \lambda(s)ds}.$$

일반적인 NHPP 소프트웨어 신뢰성 성장 모형은 다음과 같은 가정을 따른다.

i) 소프트웨어의 고장 발생은 NHPP를 따른다.

ii) 소프트웨어 고장 강도 비율은 항상 해당 시점의 소프트웨어에 잔여 고장 수에 비례한다.

iii) 소프트웨어에서 오류가 발생하면 즉시 디버깅 작업이 수행된다.

iv) 디버깅 프로세스 동안 오류를 완벽하게 제거하지 못할 수 있으므로 불완전한 디버깅 강도 비율 함수에 의하여 소프트웨어 시스템에 새로운 오류가 도입될 수 있다.

v) 불완전한 디버깅 비율은 테스트가 진행될수록 감소하고 테스트 단계가 끝날 무렵에는 무시할 수 있는 수준으로 간주한다. 이는 테스트가 진행될수록 부서의 경험과 지식이 증가하기 때문이다.

3.3.2. NHPP Exponential 모형

3.3.2.1. Goel-Okumoto 모형

Goel-Okumoto 모형(Goel 외, 1979a)은 NHPP Exponential 모형이라고도 불리며 다음과 같은 가정을 만족한다.

i) 고장은 독립적으로 발생한다.

ii) 검출된 고장 수는 항상 현재 고장 수에 비례한다.

iii) 검출되어 격리한 오류는 다음 테스트 전에 제거된다.

iv) 소프트웨어에서 고장이 발생하면 이를 유발한 오류는 즉시 제거되며 새로운 오류는 발생하지 않는다.

Goel-Okumoto 모형은 다음과 같은 미분방정식을 기반으로 얻을 수 있다.

$$\frac{dm(t)}{dt} = b[a - m(t)].$$

평균값함수와 고장강도함수는 다음과 같이 주어진다.

$$m(t) = a(1 - e^{-bt}),$$

$$\lambda(t) = abe^{-bt}.$$

여기서 a는 소프트웨어 테스트 전에 남아있는 총 기대 고장 수를 의미하며 b는 고장 검출율을 의미한다.

3.3.2.2. Musa exponential 모형

Goel-Okumoto 모형과 유사한 Musa exponential 모형(Musa, 1987)은 소프트웨어의 실행 시간(CPU time)과 실제 달력 시간(Calender time)을 고려하였으며 다음과 같은 미분방정식을 통해 모형을 얻을 수 있다.

$$\frac{dm(t)}{dt} = \frac{c}{nT}[a - m(t)].$$

Musa exponential 모형의 평균값함수와 고장강도함수는 다음과 같다.

$$m(t) = a(1 - e^{-\frac{ct}{nT}}),$$

$$\lambda(t) = \frac{c}{nT}[a - m(t)].$$

여기서 a는 소프트웨어 고장 수, c는 테스트 컴프레션(Compression) 계수, T는 테스트 초기의 평균 고장 시간, n은 소프트웨어 수명 내에서 발생할 수 있는 총 고장 수, t는 테스트를 수행하는데 소요된 전체 CPU 시간을 의미한다.

3.3.2.3. Hyperexponential 성장 모형

Ohba(1984a)가 제안한 Hyperexponential 성장 모형은 사용 여부에 따른 모듈, 복잡도 정도에 따른 모듈, 하드웨어와 상호작용 여부에 따른 모듈과 같은 모듈 클러스터로 구성된 소프트웨어에 대하여 각 모듈 클러스터는 서로 다른 초기 고장 수와 고장률을 갖는다고 가정한다. 여기서 Hyperexponential 분포는 지수분포 합과 같음을 유의해야 하며 Hyperexponential 성장 모형의 평균값함수와 고장강도함수는 다음과 같다.

$$m(t) = \sum_{i=1}^{n} a_i |1 - e^{-b_i t}|,$$

$$\lambda(t) = \sum_{i=1}^{n} a_i b_i e^{-b_i t}.$$

여기서 a는 모듈 클러스터의 수, a_i는 i번째 클러스터의 초기 오류 수, b_i는 i번째 클러스터의 각 오류에 대한 고장률을 의미한다.

3.3.2.4. Yamda-Osaki exponential 성장 모형

Yamada와 Osaki(1985)가 제안한 Yamda-Osaki exponential 성장 모형은 소프트웨어를 k개의 모듈로 나눈 형태로 지수 성장 모형을 확장한 모형이다. 서로 다른 모듈 내의 고장 확률은 다르며 각 모듈에 대해 검출된 기대 오류 수는 지수분포를 따른다고 가정한다. Yamda-Osaki exponential 성장 모형의 평균값함수와 고장강도함수는 다음과 같다.

$$m(t) = a \sum_{i=1}^{k} p_i [1 - e^{-b_i t}],$$

$$\lambda(t) = a \sum_{i=1}^{k} b_i p_i [1 - e^{-b_i t}].$$

여기서 k는 모듈 수, a는 소프트웨어 테스트 전에 남아있는 총 기대 고장 수, b_i는 i번째 모듈 내에서 한 고장에 대한 고장 검출율, p_i는 i번째 모듈에 대한 고장률을 의미한다.

3.3.3. NHPP S-shaped 모형

NHPP S-shaped(S형) 모형의 소프트웨어 신뢰도 성장 곡선은 S형 곡선의 형태를 보인다. 고장 검출율은 시간의 흐름에 따라 변하는데 테스트가 시작된 후 특정 시점에서 최대가 되며 그 이후에는 급격하게 감소한다. S-shaped 모형은 Inflection S-shaped, Delayed S-shaped 등으로 분류할 수 있다.

NHPP S-shaped 모형은 각 고장에 대한 고장 검출율이 서로 다르며 소프트웨어에서 고장이 발생할 때 원인이 되는 오류가 즉시 제거되고, 새로운 오류는 발생하지 않음을 기본적으로 가정한다. NHPP를 기반으로 하는 S-shaped 모형은 다음과 같은 미분방정식을 풀어내어 평균값함수를 얻을 수 있다.

$$\frac{dm(t)}{dt} = b(t)[a - m(t)],$$

$$m(t) = a[1 - e^{-\int_0^t b(u)du}].$$

여기서 a는 테스트를 하기 전에 소프트웨어에 남아있는 총 기대 고장 수를 의미하며 $b(t)$는 결함 당 고장 검출율을 의미한다. 환경에 따라 $b(t)$는 다양하게 가정하므로 모형별 평균값함수는 서로 다르게 주어진다.

3.3.3.1. Delayed S-shaped 모형

Yamada 외(1984)는 소프트웨어의 고장 수의 성장 곡선이 S형인 NHPP 기반 Delayed S-shaped 모형을 개발하였으며 다음과 같은 가정을 필요로 한다.

i) 소프트웨어의 모든 오류는 상호 독립적이다.
ii) 고장 검출율은 항상 소프트웨어의 현재 오류 수에 비례한다.
iii) 고장 검출율은 항상 일정하다.
iv) 소프트웨어의 초기 오류는 확률변수이다.
v) 소프트웨어는 소프트웨어에 존재하는 오류로 인해 임의의 시간에 고장이 발생한다.
vi) $(i-1)$번째 고장과 i 번째 고장 사이의 시간은 $(i-1)$번째 고장까지의 시간에 따라 다르다.
vii) 고장이 발생할 때마다 고장을 일으킨 오류는 즉시 제거되고 다른 오류는 발생하지 않는다.

Delayed S-shaped 모형의 평균값함수와 고장강도함수는 NHPP 기반 S-shaped 모형의 평균값함수를 도출하기 위한 미분방정식으로부터 $b(t)=\frac{b^2t}{bt+1}$ 을 대입하여 해를 구하면 다음과 같이 얻을 수 있다.

$$m(t)=a[1-(1+bt)e^{-bt}],$$

$$\lambda(t)=ab^2te^{-bt}.$$

여기서 a는 테스트하기 전에 소프트웨어에 남아있는 총 기대 고장 수를 의미하며 b는 정상 상태에서 결함 당 고장 검출율을 의미한다.

3.3.3.2. Inflection S-shaped 모형

Ohba(1984b)가 제안한 Inflection S-shaped 모형은 다음과 같은 가정을 요구한다.

ⅰ) 특정 오류는 다른 오류를 제거하기 전에 검출되지 않는다.

ⅱ) 고장 검출율은 항상 소프트웨어의 현재 발견된 오류 수에 비례한다.

ⅲ) 발견 가능한 각 오류의 고장률은 일정하며 동일하다.

ⅳ) 격리한 오류는 완전히 제거할 수 있다.

Inflection S-shaped 모형의 평균값함수와 고장강도함수는 미분방정식에 $b(t)=\frac{b}{1+\beta e^{-bt}}$ 를 대입하여 해를 구하면 다음과 같이 얻을 수 있다.

$$m(t)=\frac{a(1-e^{-bt})}{1+\beta e^{-bt}},$$

$$\lambda(t)=\frac{ab(1+\beta)e^{-bt}}{(1+\beta e^{-bt})^2}.$$

여기서 a는 테스트하기 전에 소프트웨어에 남아있는 총 기대 고장 수를 의미하며 b는 고장 검출율, β는 변곡계수를 나타낸다.

3.3.4. Testing effort NHPP 모형

테스트 노력(Testing effort)은 소프트웨어를 테스트하는 데 소모된 자원(인력, CPU 시간 등)을 의미한다. 소프트웨어 신뢰성 연구 초기에는 테스트 노력을 고려하지 않았으나 Yamada 외(1986)는 테스트 노력을 고려하는 다양한 모형을 제안하였다. 테스트 노력 모형은 주로 지수형이나 레일리 곡선으로 표현한다. Testing effort NHPP 모형은 다음과 같은 미분방정식을 풀어내면 얻을 수 있다.

$$\frac{dm(t)}{dt} = b(t)r[a - m(t)],$$

$$m(t) = a\left[1 - e^{-r\int_0^t b(u)du}\right].$$

여기서 r은 테스트 노력 당 고장 검출율, $b(t)$는 테스트 시간 t 시점에서의 테스트 노력 소비 함수를 의미한다. Yamada 외(1986)는 $b(t)$의 가정에 따라 Yamada exponential 모형, Yamada rayleigh 모형으로 분류하였으며 다음과 같은 가정을 기반으로 한다.

i) 소프트웨어에 남아있는 오류가 있으면 임의의 시간에 고장이 발생한다.

ii) 고장이 발생할 때마다 고장을 일으킨 오류가 즉시 제거되고 새로운 오류는 도입되지 않는다.

iii) 테스트 노력 비용은 지수 곡선 혹은 레일리 곡선으로 나타낸다.

iv) 현재의 테스트 노력에 대한 시간 $(t, t+\Delta t]$에서 발견된 소프트웨어 기대 고장 수는 남아있는 소프트웨어의 기대 고장 수에 비례한다.

v) 소프트웨어 테스트에서 발견된 고장은 NHPP를 기반으로 모형화한다.

3.3.4.1. Yamada exponential 모형

Yamada exponential 모형의 평균값함수와 고장강도함수는 다음과 같다.

$$m(t) = a(1 - e^{-r\alpha(1 - e^{(-\beta t)})}),$$

$$\lambda(t) = ar\alpha\beta e^{-r\alpha(1 - e^{-\beta t}) - \beta t}.$$

여기서 $a(t) = a$, $b(t) = r\alpha\beta e^{-\beta t}$로 주어지며 α와 β는 테스트 노력 함수의 모수이고, r는 테스트 노력 당 오류 발견율이다.

3.3.4.2. Yamada rayleigh 모형

Yamada rayleigh 모형의 평균값함수와 고장강도함수는 다음과 같다.

$$m(t) = a\left(1 - e^{-r\alpha[1 - e^{(-\beta t^2/2)}]}\right),$$

$$\lambda(t) = ar\alpha\beta t e^{-r\alpha(1 - e^{-\frac{\beta}{2}t^2}) - \frac{\beta}{2}t^2},$$

여기서 $a(t) = a$, $b(t) = r\alpha\beta t e^{-\beta t^2/2}$로 주어지며 α와 β는 테스트 노력 함수의 모수이고, r는 테스트 노력 당 오류 발견율이다.

3.3.5. NHPP imperfect debugging 모형

NHPP 소프트웨어 신뢰성 성장 모형은 디버깅 프로세스 함수 $a(t)$와 고장 검출율 함수 $b(t)$로 구성되어 있다. 앞서 언급한 모형은 $a(t)=a$인 완벽한 디버깅을 가정한다. Yamada 외(1984)는 불완전한 디버깅을 고려하여 $b(t)=b$로 가정하고, 일정한 고장 검출율을 갖는 NHPP imperfect debugging 모형을 제안하였으며 다음과 같은 가정을 기반으로 한다.

ⅰ) 발견된 오류가 제거되면 새로운 오류가 발생할 수 있다.

ⅱ) 오류를 발견할 확률은 소프트웨어에 남아있는 오류 수에 비례한다.

Yamada 외(1984)가 제안한 NHPP imperfect debugging 모형은 다음과 같은 미분방정식을 풀어내면 얻을 수 있다.

$$\frac{dm(t)}{dt}=b[a(t)-m(t)]. \tag{1}$$

$m(t)$에 대해서 초기값은 $m(0)=0$로 주어진다. $a(t)$는 소프트웨어 테스팅 시간 t 동안 발생하는 고장함수를 의미한다. 미분방정식을 풀어 얻은 $m(t)$는 다음과 같은 수식으로도 표현되며 $a(t)$ 함수의 형태에 따라 다양한 평균값함수를 갖게 된다.

$$m(t)=be^{-bt}\int_0^t a(s)e^{bs}ds.$$

Yamada 외(1984)는 다음과 같은 불안전한 디버깅 모형을 제안하였다.

3.3.5.1. Yamada imperfect debugging 모형 1

Yamada imperfect debugging 모형 1의 디버깅 프로세스 함수 $a(t)$는 다음과 같다.

$$a(t)=ae^{\alpha t}.$$

$a(t)$를 미분방정식 수식(1)에 대입하여 풀어내면 최종적인 평균값함수는 다음과 같다.

$$m(t)=\frac{ab}{\alpha+b}(e^{\alpha t}-e^{-bt}).$$

여기서 a는 테스트하기 전에 소프트웨어에 존재하는 총 기대 결함 수이며 b는 고장 검출율, α는 초기 고장에 도입된 고장 수의 증가율을 의미한다.

3.3.5.2. Yamada imperfect debugging 모형 2

Yamada imperfect debugging 모형 2의 디버깅 프로세스 함수 $a(t)$는 다음과 같다.

$$a(t) = a(1+\alpha t).$$

$a(t)$를 미분방정식 수식(1)에 대입하여 풀어내면 최종적인 평균값함수는 다음과 같다.

$$m(t) = a(1-e^{-bt})\left(1-\frac{\alpha}{b}\right)+a\alpha t.$$

여기서 a는 테스트하기 전에 소프트웨어에 존재하는 총 기대 결함 수이며 b는 고장 검출율, α는 초기 고장에 도입된 고장 수의 증가율을 의미한다.

3.3.6. Generalized imperfect debugging fault detection 모형

소프트웨어 신뢰성 모형화 과정은 세 단계로 분류된다. 첫 번째 단계에서는 통계적인 확률적 과정을 통해 시간 t에서 발견된 소프트웨어의 누적 고장 수를 나타내는 $\{N(t), t \geq 0\}$를 설명하며 이는 평균값함수로 표현할 수 있다. 모형 대부분의 평균값함수 $m(t)$는 NHPP를 기본적으로 가정한다. 두 번째 단계에서는 평균값함수를 좀 더 세분화하여 정의한다. 모형이 요구하는 가정을 토대로 오류 수 함수인 $a(t)$와 고장 검출율 함수 $b(t)$를 구체적으로 가정하여 평균값함수를 구성한다. 두 함수의 모수는 수학적 추론이나 물리적인 특성으로부터 결정할 수 있으나 대부분은 통계적인 모형을 통해 데이터를 분석하여 추론한다. 마지막 단계에서는 앞서 정의한 확률과정을 기반으로 하는 평균값함수를 이용하여 실제 데이터 세트를 분석한다.

수십 년간 소프트웨어 신뢰성 연구가 진행되면서 다양한 모형이 개발되었는데 이를 통합하여 하나의 일반화한 모형을 개발하는 시도가 있었다. 대부분의 신뢰성 모형은 고장 검출율이 현재 시점에 존재하는 오류 수에 비례한다는 가정을 기반으로 하였으나 Pham과 Nordmann(1997a)은 불완전한 디버깅과 고장 검출율을 고려하는 일반화 된 NHPP 소프트웨어 신뢰성 모형을 제안하였으며 다음과 같은 가정을 필요로 한다.

ⅰ) 고장 검출율은 고장에 따라 다르다.

ⅱ) 소프트웨어 고장이 발생할 때, 고장의 원인이 되는 소프트웨어 오류는 즉시 제거되고 새로운 오류가 발생할 수 있다.

Pham 외(1997a)는 다음과 같은 미분방정식을 통해 일반화한 imperfect debugging fault detection 모형을 제안하였다.

$$\frac{dm(t)}{dt} = b(t)[a(t) - m(t)]. \tag{2}$$

여기서 디버깅 시작점인 t_0에 대하여 초기값은 $m(t_0) = m_0$로 주어지며 $m(t)$는 다음과 같다.

$$m(t) = e^{-B(t)}\left[m_0 + \int_{t_0}^{t} a(\tau)b(\tau)e^{B(\tau)}d\tau\right].$$

$a(t)$와 $b(t)$의 형태에 따라 $m(t)$는 고유의 함수로 표현되며 $B(t)$는 다음과 같이 주어진다.

$$B(t) = \int_{t_0}^{t} b(\tau) d\tau.$$

3.3.6.1. Pham-Zhang 모형

Pham(1997b)이 제안한 Pham-Zhang 모형은 다음과 같은 가정을 기반으로 한다.

ⅰ) 오류 도입률은 테스트 시간의 지수함수로 테스트가 끝나가는 시기보다 초기에 고장 수가 더 빨리 증가함을 가정한다.

ⅱ) Inflection S형 고장 검출율은 비감소함수다.

다음과 같은 $a(t)$와 $b(t)$를 미분방정식 수식(2)에 대입하면 Pham-Zhang 모형의 평균값함수를 얻을 수 있다.

$$a(t) = c + a(1 - e^{-\alpha t}),$$

$$b(t) = \frac{b}{1 + \beta e^{-bt}},$$

$$m(t) = \frac{1}{(1 + \beta e^{-bt})}\left((c+a)(1 - e^{-bt}) - \frac{ab}{b - \alpha}(e^{-\alpha t} - e^{-bt})\right).$$

여기서 a는 테스트하기 전에 소프트웨어에 존재하는 총 기대 결함 수이며 b는 고장 검출율, α는 초기 고장에 도입된 고장 수의 증가율, β는 변곡계수를 의미한다.

3.3.6.2. Pham-Nordmann-Zhang 모형

Pham 외(1999)가 제안한 PNZ 모형은 다음과 같은 시간 종속적인 고장함수 $a(t)$와 비감소 S형 곡선을 갖는 고장 검출율 함수 $b(t)$를 제안하였다.

$$a(t) = a(1 + \alpha t),$$

$$b(t) = \frac{b}{1 + \beta e^{-bt}}.$$

$a(t)$와 $b(t)$를 미분방정식 수식(2)에 대입하여 해를 구하게 되면 다음과 같은 평균값함수를 얻을 수 있다.

$$m(t)=\frac{a}{1+\beta e^{-bt}}\left[\left(1-e^{-bt}\right)\left(1-\frac{\alpha}{b}\right)+\alpha t\right].$$

여기서 a는 테스트하기 전에 소프트웨어에 존재하는 총 기대 결함 수이며 b는 고장 검출율, α는 초기 고장에 도입된 고장 수의 증가율, β는 변곡계수를 의미한다. 초기값은 $m(0)=0$로 주어진다.

3.3.6.3. Pham exponential imperfect debugging 모형

Pham(2000)이 제안한 Pham exponential imperfect debugging 모형은 다음과 같은 가정을 필요로 한다.

ⅰ) 도입률은 테스트 시간의 지수함수이다.

ⅱ) Inflection S-shaped 모형의 고장 검출율은 비감소함수다.

미분방정식 수식(2)에 다음과 같은 $a(t)$와 $b(t)$를 대입하면 최종적인 평균값함수를 얻을 수 있다.

$$a(t)=\alpha e^{\beta t},$$

$$b(t)=\frac{b}{1+ce^{-bt}},$$

$$m(t)=\frac{\alpha b}{b+\beta}\left(\frac{e^{(\beta+b)t}-1}{e^{bt}+c}\right).$$

여기서 a는 테스트하기 전에 소프트웨어에 존재하는 총 기대 결함 수이며 b는 고장 검출율, α는 초기 고장에 도입된 고장 수의 증가율, β는 변곡계수를 의미한다.

3.3.7. Testing coverage 모형

테스팅 커버리지(Testing coverage)는 소프트웨어를 구성하는 코드의 테스트 정도를 측정하는 지표로 코드 커버리지(Code coverage)로 불리기도 한다. 테스팅 커버리지를 기준으로 개발자는 향후 소프트웨어 신뢰성을 향상하는 데 필요로 하는 테스트 노력을 추정할 수 있다. 또한 소비자에게는 소프트웨어 신뢰성을 판단하는 척도로도 사용되기 때문에 개발자와 소비자 모두에게 중요한 지표로 작용한다 (Pham, 2003).

Pham 외(2003)는 테스팅 커버리지를 고려하는 소프트웨어 신뢰성 성장 모형의 평균값함수를 얻기 위해 다음과 같은 미분방정식을 제안하였다.

$$\frac{dm(t)}{dt} = \frac{c'(t)}{1-c(t)}[a(t)-m(t)]. \tag{3}$$

여기서 $a(t)$는 총 오류 수 함수를 의미하고, $c(t)$는 테스트 시간 t에 대한 함수로 코드 커버리지의 백분율을 나타낸다. $1-c(t)$는 시간 t까지의 테스트에서 아직 다루지 않은 코드의 백분율을 의미하고, $c'(t)$은 코드 커버리지의 비율을 나타낸다.

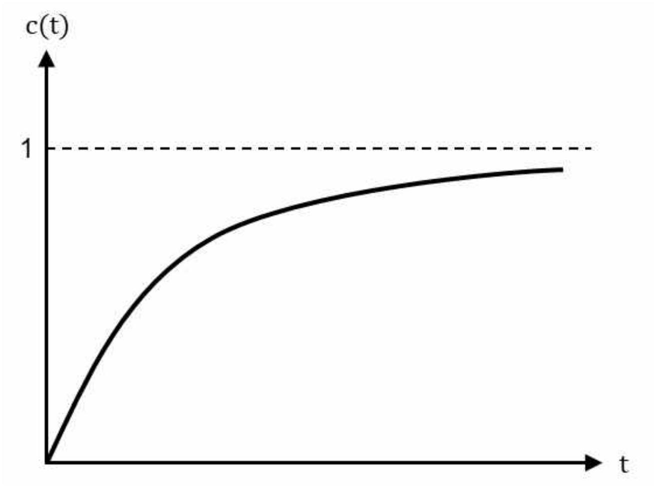
미분방정식을 풀어내면 다음과 같이 테스팅 커버리지 모형의 평균값함수를 얻을 수 있다.

$$m(t) = e^{-B(t)}\left(m_0 + \int_{t_0}^{t} a(\tau)e^{B(\tau)}\frac{c'(\tau)}{1-c(\tau)}d\tau\right).$$

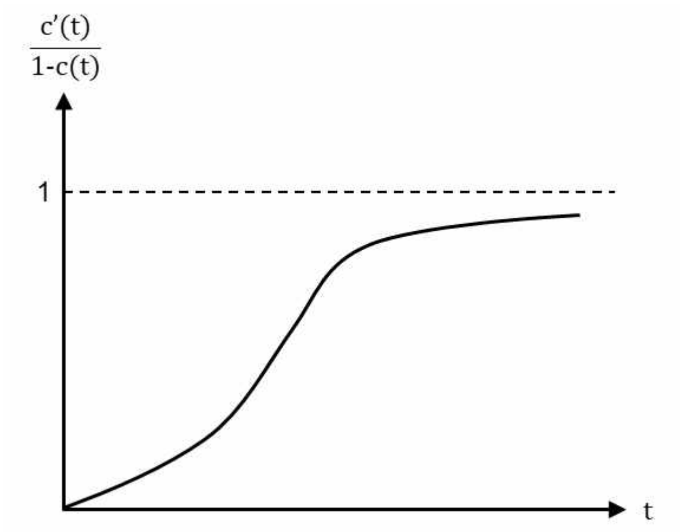
여기서 $m(t)$는 디버깅 시작점 t_0에 대하여 초기값은 $m(t_0)=m_0$로 주어진다. $B(t)$는 고장 검출율 함수인 $\frac{c'(t)}{1-c(t)}$를 t에 대해서 적분한 형태로 다음과 같이 주어진다.

$$B(t) = \int_{t_0}^{t}\frac{c'(\tau)}{1-c(\tau)}d\tau.$$

만약 $c(t)$가 시간 t에 대하여 음이 아닌 오목 함수(Concave function)라면 $c(t)$는 <그림 III-2>와 같으며 $\frac{c'(t)}{1-c(t)}$는 <그림 III-3>과 같은 S형 곡선을 갖게 된다.



<그림 Ⅲ-2> $c(t)$ 함수



<그림 Ⅲ-3> 고장 검출율 함수

3.3.7.1. PZ coverage 모형

Pham 외(2003)가 제안한 PZ coverage 모형에서 사용되는 테스팅 커버리지 함수 $c(t)$는 다음과 같이 주어진다.

$$c(t) = 1 - (1+bt)^{e^{-bt}}.$$

총 오류 수 함수(Error content function)는 다음과 같이 주어진다.

$$a(t) = a(1+\alpha t).$$

미분방정식 수식(3)에 $c(t)$와 $a(t)$를 대입하면 평균값함수를 다음과 같이 구할 수 있다.

$$m(t) = a\left(1+\alpha t - \frac{bt+1}{e^{bt}}\right) - \frac{a\alpha(1+bt)}{be^{bt+1}}\left(\ln(bt+1) + \sum_{i=0}^{\infty}\frac{(1+bt)^{i+1}-1}{(i+1)!(i+1)}\right).$$

여기서 a는 테스트하기 전에 소프트웨어에 존재하는 총 기대 결함 수이며 b는 고장 검출율, α는 초기 고장에 도입된 고장 수의 증가율을 의미한다. 또한, $m(t)$에 대해서 초기값은 $m(0)=0$로 주어진다.

3.3.8. 운용 환경의 불확실성을 고려하는 NHPP 모형

소비자는 다양한 운용 환경에서 제품을 사용하게 된다. 운용 환경은 노트북, 태블릿과 같은 하드웨어(Hardware; HW)부터 윈도우(Window), 리눅스(Linux), 맥(MAC) 등과 같은 운영체제(Operating system; OS), 백그라운드(Background)에서 실행 중인 프로그램 등 모든 요소를 포함한다. 이에 반면 테스트 환경은 상당한 요소가 통제되어 있으므로 실제 소비자가 사용하는 운용 환경의 모든 요소를 반영하지 못한다. 따라서 실제 운용 환경에서는 테스터가 예측하지 못한 돌발적인 고장이 발생할 수 있다. 이를 운용 환경의 불확실성(Uncertainty)으로 표현하며 평균값함수에 어떤 분포를 갖는 확률변수를 도입하여 운용 환경의 불확실성을 설명한다.

3.3.8.1. Vtub-shaped fault detection rate 모형

Pham(2014a)은 운용 환경의 불확실성을 고려하는 소프트웨어 신뢰성 성장 모형을 얻기 위해 수식 (4)와 같은 미분방정식을 제안하였다.

$$\frac{dm(t)}{dt} = \eta b(t)[a(t) - m(t)]. \tag{4}$$

총 오류 수 함수 $a(t)$와 결함 탐지율 함수 $b(t)$는 다음과 같이 주어지며, 수식(4)에 각 함수를 대입하면 최종적인 평균값함수를 얻을 수 있다.

$$a(t) = N,\ b(t) = b\ln(a)t^{b-1}a^{t^b},$$

$$m(t) = N\left[1 - \left(\frac{\beta}{\beta + a^{t^b} - 1}\right)^{\alpha}\right].$$

여기서 N은 테스트 전에 존재하는 기대 고장 수를 나타내고, a, b, α는 $b(t)$의 모수이며 $b(t)$는 V형 형태의 함수이다. η는 운용 환경의 불확실성을 의미하는 확률변수로 감마분포를 따른다고 가정한다.

3.3.8.2. Three parameter fault detection rate 모형

Song 외(2017a) 운용 환경의 불확실성을 고려하여 세 개의 모수로 구성된 결함 탐지율 함수를 갖는 NHPP 소프트웨어 신뢰성 성장 모형을 제안하였으며 다음과 같은 가정을 고려한다.

ⅰ) 소프트웨어의 결함의 발생 및 제거는 NHPP를 따른다.

ⅱ) 소프트웨어의 결함은 소프트웨어 실행 중 고장을 초래할 수 있다.

ⅲ) 소프트웨어 고장률은 당시 소프트웨어에 남아있는 고장 수에 비례한다.

ⅳ) 소프트웨어의 고장이 감지되면 고장을 유발한 오류는 즉시 제거된다.

ⅴ) 결함 탐지율 함수는 단위 고장 검출율 함수인 $b(t)$와 독립이며 단위가 없는 확률변수 η의 곱으로 표현한다.

평균값함수는 수식 (4)에 수식 (5)를 대입하여 풀어내면 얻을 수 있다.

$$a(t) = N,\ b(t) = \frac{a}{1+ce^{-bt}}, \tag{5}$$

$$m(t) = N\left[1-\left(\frac{\beta}{\beta-\frac{a}{b}\ln\left(\frac{(1+c)e^{-bt}}{1+ce^{-bt}}\right)}\right)\right]$$

여기서 총 오류 수 함수 $a(t)$의 N은 테스트 전에 존재하는 기대 고장 수를 나타내고, η는 운용 환경의 불확실성을 의미하는 확률변수로 지수분포를 따른다고 가정한다. $b(t)$는 결함 탐지율 함수로 a, b, c를 모수로 갖는 변곡 S형 곡선의 비감소 함수다.

3.3.8.3. S형 성장 곡선 모형

Song 외(2017b)는 운용 환경의 불확실성을 고려하는 S형 성장 곡선의 NHPP 신뢰성 모형을 제안하였으며 요구되는 가정은 Three parameter fault detection rate 모형과 동일하다.

$$a(t) = N, \ b(t) = \frac{a^2 t}{1+at}, \ a, b > 0. \tag{6}$$

여기서 총 오류 수 함수 $a(t)$의 N은 테스트 전에 존재하는 기대 고장 수를 나타내고, η는 모수 α와 β를 갖는 일반화 확률밀도함수를 따른다. $b(t)$ 함수는 모수 a와 b를 갖는 와이블분포이며 $b<1$이면 감소하고, $b>1$이면 증가하며 $b=1$일 때 상수의 형태를 갖는다.

수식 (6)을 수식 (4)에 대입하여 풀어내면 다음과 같은 S형 성장 곡선 형태의 평균값함수를 얻을 수 있다.

$$m(t) = N\left(1 - \frac{\beta}{\beta + at - \ln(1+at)}\right)^{\alpha}.$$

3.3.8.4. Weibull fault detection rate 모형

Song 외(2017c)는 운용 환경의 불확실성을 고려하는 와이블분포의 결함 탐지율 함수를 갖는 NHPP 신뢰성 성장 모형을 제시하였으며 요구되는 가정은 Three parameter fault detection rate 모형과 동일하다.

$$a(t) = N, \ b(t) = a^b b t^{b-1}, \ a, b > 0. \tag{7}$$

여기서 N은 테스트 전에 존재하는 기대 고장 수를 나타내고, η는 모수 α와 β를 갖는 일반화 확률밀도함수를 따른다. $b(t)$ 함수는 모수 a와 b를 갖는 와이블분포이며 $b<1$이면 감소하고, $b>1$이면 증가하며 $b=1$일 때 상수로 나타난다.

수식 (7)을 수식 (4)에 대입하여 풀어내면 수식 (8)과 같은 평균값함수를 얻을 수 있다.

$$m(t) = N\left[1 - \frac{\beta}{\beta + (at)^b}\right]^{\alpha}. \tag{8}$$

3.3.8.5. 결함 제거 확률의 영향을 받는 S형 곡선 모형

Song 외(2018)는 운용 환경의 불확실성을 고려하면서 결함 제거 확률에 영향을 받는 S형의 결함 탐지율 함수를 제안하였으며 요구되는 가정은 Three parameter fault detection rate 모형과 동일하다.

$$a(t) = N,\ b(t) = \frac{ap}{1+\gamma e^{-bpt}},\ a,b,\gamma > 0,\ 0 < p < 1. \tag{9}$$

여기서 N은 테스트 전에 존재하는 기대 고장 수를 나타내고, η는 모수 α와 β를 갖는 일반화 확률밀도함수를 따른다. $b(t)$ 함수는 결함 제거 확률의 영향을 받는 결함 탐지율 함수로 S형 함수이며 모수 $a,\ p,\ \gamma,\ b$로 구성된다. 수식 (9)를 수식 (4)에 대입하여 풀어내면 수식 (10)과 같은 평균값함수를 얻을 수 있다.

$$m(t) = N\left[1 - \frac{\beta}{\beta - \frac{a}{b}\ln\left(\frac{(1+\gamma)e^{-bpt}}{1+\gamma e^{-bpt}}\right)}\right]^{\alpha}. \tag{10}$$

3.3.8.6. 테스팅 커버리지 모형

Chang 외(2003)은 NHPP 테스팅 커버리지의 일반화 모형에 운용 환경의 불확실성을 고려하였으며 요구되는 가정은 기존의 NHPP 테스팅 커버리지 일반화 모형과 동일하고, 다음과 같은 미분방정식을 기반으로 평균값함수를 얻는다.

$$\frac{dm(t)}{dt} = \eta\left(\frac{c'(t)}{1-c(t)}\right)[N - m(t)].$$

여기서 테스팅 커버리지 함수 $c(t)$는 다음과 같이 주어지며 η는 모수 α와 β를 갖는 감마분포를 따르며 최종적인 평균값함수는 수식(11)과 같다.

$$c(t) = 1 - e^{-(at)^b},\ a,b > 0,$$

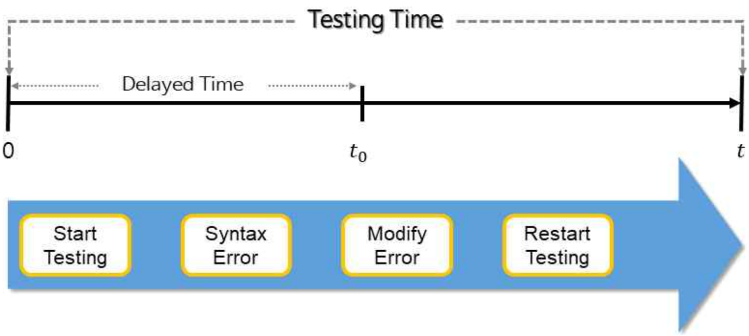
$$m(t) = N\left[1 - \left(\frac{\beta}{\beta + (at)^b}\right)^{\alpha}\right]. \tag{11}$$

수식(8)과 수식(11)에서 α의 값이 1이면 평균값함수는 $N\left[1 - \left(\frac{\beta}{\beta + (at)^b}\right)\right]$로 같게 된다. 그러나 평균값함수가 동일해도 서로 다른 환경을 가정하며 평균값함수가 도출되는 미분방정식이 다르므로 완전히 같은 함수라고 볼 수 없다.

3.4. 새로운 유형의 모형 및 비교

3.4.1. Delayed time by syntax error 모형

소프트웨어의 개발이 마무리되면 소프트웨어를 평가하기 위한 테스트 계획을 세운다. 테스트 기간이 길어지면 소프트웨어의 신뢰성은 높아지나 상당한 인력, 비용, 시간이 소모된다. 반면에 테스트 기간이 짧아지면 경제적 비용은 절감하나 소프트웨어의 신뢰성 보장은 어렵다. 따라서 소프트웨어 신뢰성과 경제적 비용을 고려하여 적절한 테스트 기간 t를 결정해야 한다. 이론적으로는 시간 t동안 테스트를 진행하지만, 실질적인 테스트 기간은 t보다 더 짧을 수 있다. 예를 들면 테스트 코드에 개발자가 인지하지 못한 오탈자가 있거나 문법적인 오류가 존재하는 경우, 테스트는 오류를 수정할 때까지 실행되지 않으므로 테스트는 지연되며 실질적인 테스트 기간은 계획했던 t보다 더 짧아지게 된다.



<그림 Ⅲ-4> 테스트 시간 t의 구조

<그림 Ⅲ-4>는 구문 오류로 인해 지연되는 테스트 시간 t의 구조를 나타낸다. 이론적인 테스트 시간은 t이지만 실질적인 테스트 시간은 지연되는 시간 t_0을 제외해야 하므로 $t-t_0$가 실제 테스트 시간이 되며 $0<t_0<t$이다.

Lee 외(2018)는 구문 오류로 인해 지연되는 테스트 시간을 고려하는 NHPP 소프트웨어 신뢰성 성장 모형을 제안하였으며 평균값함수는 다음과 같은 미분방정식을 풀어내면 얻을 수 있다.

$$\frac{dm(t)}{dt} = b(t)[a(t)-m(t)], \tag{12}$$

$$m(t) = N(1 - e^{-\int_{t_0}^{t} b(s)\,ds}) = N\left(1 - \frac{\beta}{\beta + \int_{t_0}^{t} b(s)ds}\right)^{\alpha}.$$

여기서 t_0는 오류가 발생한 코드를 수정한 후의 디버깅 시작점을 나타낸다. 총 오류 수 함수인 $a(t)$와 결함 탐지율 함수 $b(t)$는 다음과 같이 주어진다.

$$a(t) = N,\ \ b(t) = a^b b t^{b-1} \quad a, b > 0.$$

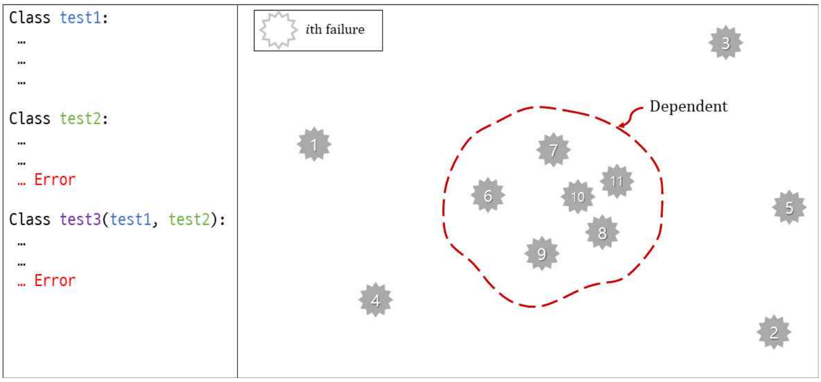
여기서 N은 테스트 전에 존재하는 기대 고장 수를 나타내고, 결함 탐지율 $b(t)$ 함수는 모수 a, b로 구성된다. $a(t)$와 $b(t)$를 수식(12)에 대입하여 미분방정식을 풀어내면 모형의 최종 평균값함수는 다음과 같다.

$$m(t) = N\left(1 - \frac{\beta}{\beta + a(t - t_0)^b}\right)^{\alpha}.$$

이때 $m(t)$에 대한 초기값은 $m(0) = 0$로 주어진다.

3.4.2. Dependent failure 모형

NHPP를 따르는 대부분의 소프트웨어 신뢰성 성장 모형은 소프트웨어의 고장이 독립적으로 발생한다고 가정한다. 그러나 때때로 소프트웨어의 고장은 종속적으로 발생하며 먼저 발생한 고장이 다음에 발생할 고장에 영향을 준다. 예를 들면 소프트웨어 코드 내에서 특정 클래스에 문법 오류가 발생했을 때, 코드를 수정하여도 오류가 발생했던 클래스를 참조하는 또 다른 클래스에서 오류가 발생하여 고장이 연쇄적으로 나타날 수 있다. 이 경우, 소프트웨어의 고장은 종속적으로 발생한다.



<그림 Ⅲ-5> 종속적으로 발생하는 소프트웨어의 고장 구조

<그림 Ⅲ-5>는 소프트웨어의 종속적인 고장 발생의 구조를 나타낸다.

소프트웨어의 고장이 종속적임을 가정하는 소프트웨어 신뢰성 성장 모형의 평균값함수는 다음과 같은 미분방정식을 풀어내면 얻을 수 있다(Lee 외, 2020).

$$\frac{dm(t)}{dt} = b(t)[a(t) - m(t)]m(t), \tag{13}$$

총 오류 수 함수 $a(t)$와 결함 탐지율 함수 $b(t)$는 다음과 같다.

$$a(t) = a(\alpha t + 1),\ b(t) = \frac{b}{b + ce^{-bt}}.$$

여기서 $a(t)$는 $\alpha = 0$일 때의 $a(t) = a$인 특수한 유형으로 고려한다. $b(t)$는 b, c를 모수로 갖는 비감소함수이다.

$a(t)$와 $b(t)$를 수식 (13)에 대입하면 최종적인 평균값함수는 다음과 같이 주어진다.

$$m(t)=\frac{a}{1+\frac{a}{h}\left(\frac{b+c}{c+be^{bt}}\right)^{\frac{a}{b}}}.$$

이때 $m(t)$에 대한 초기값은 $m(0)=0$로 주어진다.

3.4.3. 적합도 척도

실제 데이터를 각 모형의 평균값함수 $m(t)$에 적합하여 Matlab을 이용해 최소 제곱 추정(Least squares estimation; LSE) 방법으로 모수 추정값을 얻을 수 있다. 모형의 적합도를 비교하기 위해서는 모수 추정값을 다시 평균값함수에 대입하여 척도를 계산한다. 자주 사용되는 척도로 평균 제곱 오차(Mean squared error; MSE), 예측비위험(Predictive ratio risk; PRR), 예측력(Predictive power; PP), R^2 등이 있다(Pham 2000, Pham 2006a, Pham 2006b, Pham 2014a, Akaike 1974, Pillai 외 1997).

첫 번째, 평균제곱오차(MSE)는 다음과 같다.

$$MSE=\frac{\sum_{i=1}^{n}\left(\hat{m}(t_i)-y_i\right)^2}{N-m}.$$

여기서 N은 데이터의 관측 수, m은 모형의 모수 수를 나타내고, $\hat{m}(t_i)$는 모수 추정치를 대입한 평균값함수에서 t_i일 때 예측값, y_i는 t_i에서의 실제 데이터 값을 의미한다. 즉, MSE는 데이터의 관측 수와 모형의 모수 수를 고려하여 모형의 추정치와 실제 데이터의 거리를 측정하는 척도이다. 일반적으로 모수의 수가 클수록 적합도가 더 우수한 경향을 보이는데 MSE는 모수의 수에 대한 페널티를 부여하므로 모수의 수가 서로 다른 모형의 적합도를 비교하는데 합리적이다.

두 번째, 예측비위험(PRR)은 다음과 같다.

$$PRR=\sum_{i=1}^{n}\left(\frac{\hat{m}(t_i)-y_i}{\hat{m}(t_i)}\right)^2.$$

모형 예측값을 고려하여 모형 예측값과 실제 데이터 값의 거리를 측정한다.

세 번째, 예측력(PP)은 다음과 같다.

$$PP = \sum_{i=1}^{n}\left(\frac{\hat{m}(t_i)-y_i}{y_i}\right)^2.$$

실제 데이터 값을 고려하여 모형 예측값과 실제 데이터 값의 거리를 측정한다.

네 번째, 결정계수 R^2(R-squared)은 회귀직선의 적합도를 평가하는데 많이 쓰이며 다음과 같이 주어진다.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_i-\hat{m}(t_i)\right)^2}{\sum_{i=1}^{n}\left(y_i-\bar{y}\right)^2}.$$

여기서 $\bar{y}$는 실제 데이터 세트의 전체 평균을 의미한다.

다섯 번째, Akaike의 정보 기준(Akaike's information criteria; AIC)은 다음과 같이 정의한다.

$$AIC = -2\ln L + 2N.$$

여기서 N은 자유도이며 우도 함수 L과 $\ln L$은 다음과 같이 주어진다.

$$L = \prod_{i=1}^{n}\frac{(\hat{m}(t_i)-\hat{m}(t_{i-1}))^{y_i-y_{i-1}}}{(y_i-y_{i-1})!}e^{-(\hat{m}(t_i)-\hat{m}(t_{i-1}))},$$

$$\ln L = \sum_{i=1}^{n}\left\{(y_i-y_{i-1})\ln\left(\hat{m}(t_i)-\hat{m}(t_{i-1})\right)-\left(\hat{m}(t_i)-\hat{m}(t_{i-1})\right)-\ln((y_i-y_{i-1})!)\right\}.$$

여섯 번째, 절대 오차 합(Sum of absolute error; SAE)은 예측값과 실제 데이터 값의 절대 거리를 측정하며 다음과 같이 표현한다.

$$SAE = \sum_{i=1}^{n}\left|\hat{m}(t_i)-y_i\right|.$$

일곱 번째, 변동(Variation)은 예측 편향의 표준편차로 다음과 같이 주어진다.

$$Variation = \sqrt{\frac{\sum_{i=1}^{n}\left(\left(\hat{m}(t_i)-y_i\right)-Bias\right)^2}{n-1}}.$$

여기서 편향(Bias)은 다음과 같다.

$$Bias = \sum_{i=1}^{n} \left| \frac{\hat{m}(t_i) - y_i}{n} \right|.$$

여덟 번째, 평균 제곱 오차의 근(Root mean square prediction error; RMSPE)은 다음과 같다.

$$RMSPE = \sqrt{Variation^2 + Bias^2}.$$

결정계수 R^2은 1에 가까울수록 모형이 우수함을 나타내며 나머지 척도의 값은 작을수록 모형이 우수함을 나타낸다.

앞서 언급한 여덟 개의 척도는 모형의 적합도를 비교하는 척도이며 예측력을 판단하는 척도로 $MSE_{prediction}$, $AIC_{prediction}$, $PP_{prediction}$을 사용하기도 한다(Li 외, 2019). 전체 데이터 중 일부만 사용하여 평균값함수의 모수를 추정하며, 나머지 데이터는 예측력을 비교하는 데 사용한다. 예를 들면 $MSE_{prediction}$은 다음과 같다.

$$MSE = \frac{1}{n-m+1-N} \sum_{i=m}^{n} \left(\hat{m}(t_i) - y_i \right)^2.$$

전체 데이터 세트 $t_i = 1, 2, \cdots, n$에 대하여 $t_i = 1, 2, \cdots, m-1$까지의 데이터는 평균값함수의 모수를 추정하는 데 사용하고, 나머지인 $t_i = m, \cdots, n$까지의 데이터는 평균값함수 $m(t)$에 의한 예측값 $\hat{m}(t_i)$를 추정하여 실제 데이터와의 거리를 측정하는 척도를 계산한다. 여기서 $t_{m-1} < t_n$이며 N은 평균값함수의 모수 수를 나타낸다. 마찬가지로 $AIC_{prediction}$과 $PP_{prediction}$도 같은 방법으로 척도를 계산한다.

3.4.4. 적합도 비교 및 결과

본 논문에서는 비교 모형 집단, Delayed time by syntax error 모형(STX), Dependent failure 모형(DPF)에 데이터 세트를 적합하여 모수를 추정하고, 척도를 측정하여 적합도를 판단한다.

<표 III-4>와 <표 III-5>는 각 데이터 세트를 나타낸다. 데이터 세트 1은 Tandem Computers (Wood, 1996)의 주요 소프트웨어에서 추출한 데이터로 총 100개의 고장이 발견됐다. 데이터 세트 2는 ABC 소프트웨어 회사의 온라인 통신 시스템(Online communication system; OCS) 프로젝트로부터 추출됐다(Pham 외, 2003a). 12주 동안 주 단위로 검출된 고장 횟수를 기록한 데이터 세트이며 총 55개의 고장이 발견됐다.

<표 Ⅲ-4> 데이터 세트 1(DS 1)

TIME	CUM.F
1	16
2	24
3	27
4	33
5	41
6	49
7	54
8	58
9	69
10	75
11	81
12	86
13	90
14	93
15	96
16	98
17	99
18	100
19	100
20	100

<표 Ⅲ-5> 데이터 세트 2(DS 2)

TIME	F	CUM.F
1	10	10
2	2	12
3	4	16
4	6	22
5	6	28
6	8	36
7	4	40
8	3	43
9	1	44
10	6	50
11	1	51
12	4	55

<표 Ⅲ-6>은 여덟 개의 비교 모형(1-8)과 Delayed time by syntax error 모형(STX), Dependent failure 모형(DPF)의 평균값함수 $m(t)$를 나타낸다.

모형별 모수의 추정치는 Matlab을 사용하여 LSE 방법을 사용하였다. <표 Ⅲ-7>과 <표 Ⅲ-8>은 각 데이터 세트를 적합하여 얻은 모형별 모수 추정치를 나타내고, <그림 Ⅲ-6>과 <그림 Ⅲ-7>은 각 데이터 세트에 따른 모형별 평균값함수를 도식화한다.

<표 Ⅲ-6> 각 모형의 평균값함수

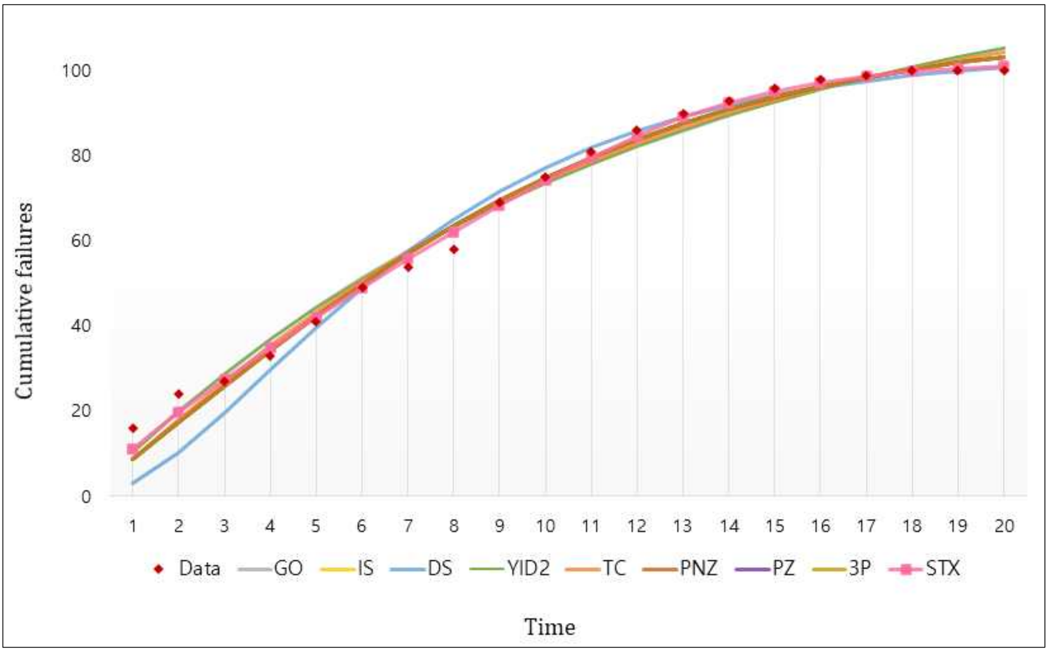
No	Model	$m(t)$
1	GO	$a(1-e^{-bt})$
2	IS	$\frac{a(1-e^{-bt})}{1+\beta e^{-bt}}$
3	DS	$a(1-(1+bt)e^{-bt})$
4	YID 2	$a(1-e^{-bt})(1-\frac{\alpha}{b})+a\alpha t$
5	TC	$N\left[1-\left(\frac{\beta}{\beta+(at)^b}\right)^{\alpha}\right]$
6	PNZ	$\frac{a}{(1+\beta e^{-bt})}\left((1-e^{-bt})(1-\frac{\alpha}{b})+\alpha t\right)$
7	PZ	$\frac{1}{(1+\beta e^{-bt})}\left((c+a)(1-e^{-bt})-\frac{ab}{b-\alpha}(e^{-\alpha t}-e^{-bt})\right)$
8	3P	$N\left[1-\left(\frac{\beta}{\beta-\frac{a}{b}\ln\left(\frac{(1+c)e^{-bt}}{1+ce^{-bt}}\right)}\right)\right]$
9	STX	$N\left(1-\frac{\beta}{\beta+a(t-t_0)^b}\right)^{\alpha}$
10	DPF	$\frac{a}{1+\left(\frac{a}{h}\right)\left(\frac{b+c}{c+be^{bt}}\right)^{\frac{a}{b}}}$

<표 Ⅲ-7> 데이터 세트 1에 대한 모형별 모수 추정 결과

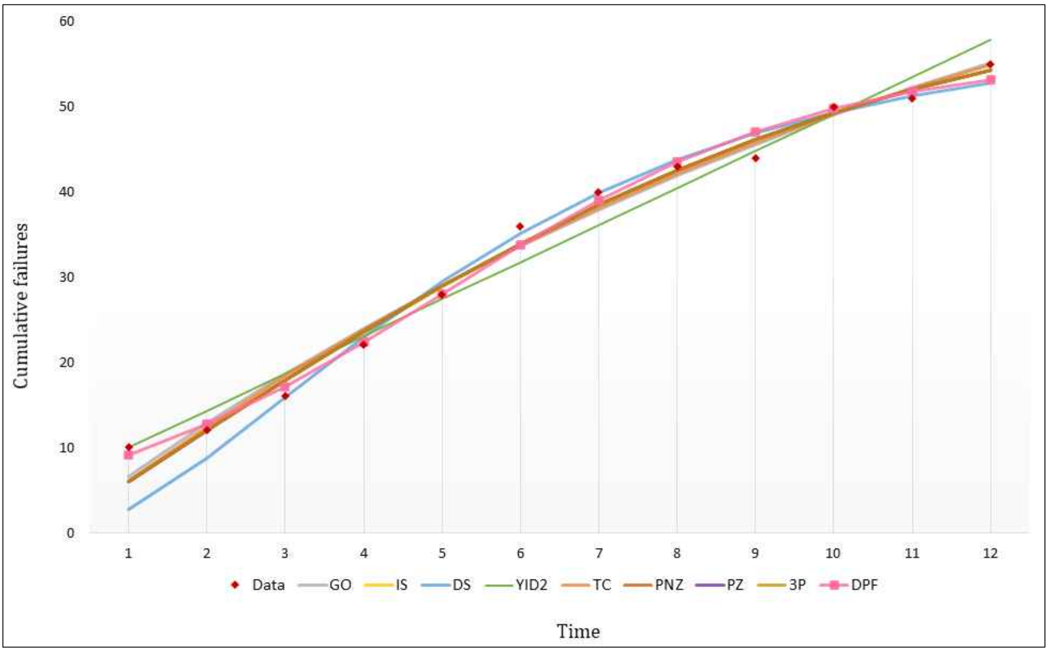
Model	Parameter	SSE
GO	$\hat{a}=129.511$, $\hat{b}=0.084$	232.5177
IS	$\hat{a}=110.841$, $\hat{b}=0.172$, $\hat{\beta}=1.204$	179.5844
DS	$\hat{a}=104.049$, $\hat{b}=0.265$	505.1447
YID 2	$\hat{a}=129.400$, $\hat{b}=0.084$, $\hat{\alpha}=0.0001$	232.6899
TC	$\hat{a}=0.0003$, $\hat{b}=1.111$, $\hat{\alpha}=9915.2$, $\hat{\beta}=15.820$, $\hat{N}=118.466$	217.4253
PNZ	$\hat{a}=110.679$, $\hat{b}=0.172$, $\hat{\alpha}=0.0001$, $\hat{\beta}=1.199$	179.7893
PZ	$\hat{a}=0.0003$, $\hat{b}=0.172$, $\hat{\alpha}=10000$, $\hat{\beta}=1.204$, $\hat{c}=110.841$	179.5844
3P	$\hat{a}=391.999$, $\hat{b}=0.172$, $\hat{\beta}=0.716$, $\hat{N}=110.855$, $\hat{c}=7014.4$	179.5844
STX	$\hat{a}=0.0001$, $\hat{b}=6.976$, $\hat{\alpha}=0.120$, $\hat{\beta}=11119$, $\hat{N}=102.445$, $\hat{t_0}=0.00001$	75.7939

<표 Ⅲ-8> 데이터 세트 2에 대한 모형별 모수 추정 결과

Model	Parameter	SSE
GO	$\hat{a}=94.344$, $\hat{b}=0.0733$	40.2448
IS	$\hat{a}=65.781$, $\hat{b}=0.206$, $\hat{\beta}=1.293$	36.4997
DS	$\hat{a}=57.478$, $\hat{b}=0.344$	82.0956
YID 2	$\hat{a}=5.749$, $\hat{b}=52.415$, $\hat{\alpha}=0.756$	69.7821
TC	$\hat{a}=0.005$, $\hat{b}=1.075$, $\hat{\alpha}=2001$, $\hat{\beta}=84.681$, $\hat{N}=80.373$	39.4937
PNZ	$\hat{a}=64.922$, $\hat{b}=0.208$, $\hat{\alpha}=0.001$, $\hat{\beta}=1.286$	36.5058
PZ	$\hat{a}=7.617$, $\hat{b}=0.210$, $\hat{\alpha}=0.005$, $\hat{\beta}=1.321$, $\hat{c}=64.992$	36.5072
3P	$\hat{a}=0.05496$, $\hat{b}=0.2072$, $\hat{\beta}=0.0245$, $\hat{N}=68.5181$, $\hat{c}=25.0097$	36.5002
DPF	$\hat{a}=55.893$, $\hat{b}=0.004$, $\hat{c}=0.548$, $\hat{h}=7.274$	22.5607



<그림 Ⅲ-6> 데이터 세트 1에 대한 모형별 평균값함수



<그림 Ⅲ-7> 데이터 세트 2에 대한 모형별 평균값함수

<표 III-9>와 <표 III-10>은 각 데이터 세트와 비교 모형(STX, DPF)의 추정된 적합도 결과를 나타낸다. <표 III-9>에서 STX 모형의 R^2은 0.9953으로 가장 크게 나타났으며 나머지 기준은 모두 다른 기존 모형과 비교해 가장 작게 나타났다. 따라서 데이터 세트 1에 대해서 STX 모형의 적합도가 가장 우수함을 알 수 있다. <표 III-10>에서 DPF 모형의 AIC 값은 58.6958로 전체 아홉 개의 모형 중 세 번째로 작은 값으로 나타났다. R^2은 0.9919로 가장 높게 나타났으며 나머지 기준은 다른 기존 모형과 비교해 가장 작게 나타났다. 따라서 데이터 세트 2에 대해서 DPF 모형의 적합도가 가장 우수함을 알 수 있다.

NHPP 소프트웨어 신뢰성 성장 모형의 $100(1-\alpha)$%에 대한 양측 신뢰구간(Confidence intervals)은 다음과 같은 수식을 통해 근사적으로 추정할 수 있다(Pham, 2007).

$$\hat{m}(t) \pm z_{\alpha/2}\sqrt{\hat{m}(t)}.$$

여기서 $z_{a/2}$는 표준정규분포(Standard normal distribution)의 $100(1-\alpha)$분위수를 의미한다.

<표 III-11>과 <표 III-12>는 데이터 세트에 따른 STX 모형과 DPF 모형의 $\alpha=0.05$에 대한 신뢰구간 하한(Lower-limit confidence interval; LC), 상한(Upper-limit confidence interval; UC) 및 평균값함수를 나타내며 <그림 III-8>과 <그림 III-9>는 이를 시각화한다.

<표 Ⅲ-9> 데이터 세트 1에 대한 모형별 적합도

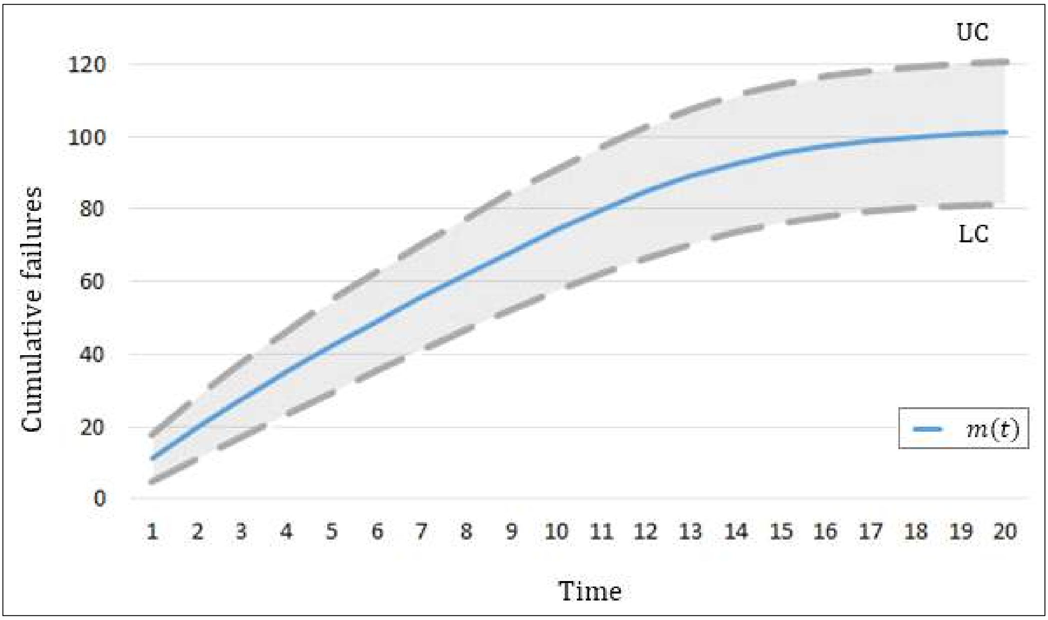
Model	MSE	PRR	PP	R^2	SAE	AIC	Variation	RMSPE
GO	12.9177	0.3712	0.2021	0.9857	61.3458	91.8213	3.4976	3.4982
IS	10.5638	0.8674	0.3047	0.9890	45.2128	90.2698	3.0417	3.0728
DS	28.0636	19.6801	1.0832	0.9689	63.1991	111.2568	4.9439	5.1458
YID2	13.6876	0.3726	0.2022	0.9857	61.3744	93.8697	3.4988	3.4995
TC	14.4950	0.8375	0.3014	0.9866	53.9273	97.8758	3.3668	3.3820
PNZ	11.2368	0.8648	0.3042	0.9889	45.3028	92.3017	3.0437	3.0745
PZ	11.9723	0.8674	0.3047	0.9890	45.2125	94.2698	3.0417	3.0728
3P	11.9723	0.8669	0.3046	0.9890	45.2171	94.2682	3.0418	3.0728
STX	5.4139	0.2542	0.1383	0.9953	27.0759	86.5700	1.9870	1.9968

<표 Ⅲ-10> 데이터 세트 2에 대한 모형별 적합도

Model	MSE	PRR	PP	R^2	SAE	AIC	Variation	RMSPE
GO	4.0245	0.2932	0.1627	0.9855	19.4170	57.7076	1.9120	1.9127
IS	4.0555	0.4815	0.1905	0.9868	17.0520	60.1451	1.8126	1.8208
DS	8.2096	7.3679	0.6177	0.9704	20.9540	69.6251	2.6305	2.7236
YID2	7.7536	0.0893	0.1027	0.9748	24.4096	58.2593	2.5187	2.5187
TC	5.6420	0.4307	0.1888	0.9857	18.3723	64.2519	1.8906	1.8945
PNZ	4.5632	0.4818	0.1906	0.9868	17.0566	62.1389	1.8128	1.8210
PZ	5.2153	0.4890	0.1917	0.9868	17.0459	64.1689	1.8125	1.8210
3P	5.2143	0.4812	0.1905	0.9868	17.0484	64.1366	1.8131	1.8209
DPF	2.8201	0.0276	0.0270	0.9919	12.7389	58.6958	1.4321	1.4321

<표 Ⅲ-11> STX 모형의 95% 신뢰구간(DS 1)

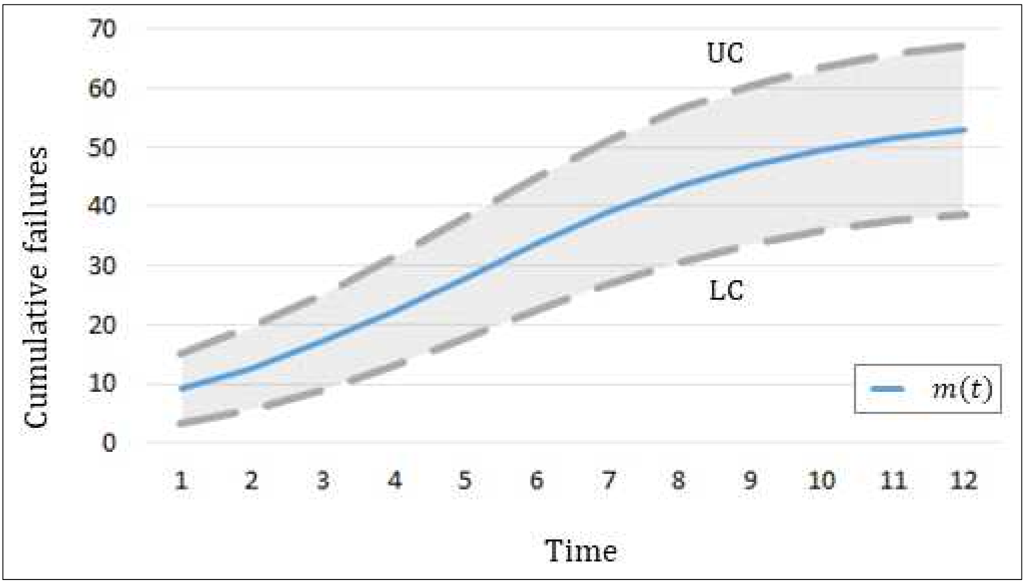
Time	$\widehat{m}(t)$	LC	UC	Time	$\widehat{m}(t)$	LC	UC
1	11.0907	4.5635	17.6179	11	79.8466	62.3330	97.3603
2	19.7148	11.0123	28.4173	12	84.8181	66.7674	102.8687
3	27.6015	17.3044	37.8986	13	89.1055	70.6043	107.6068
4	35.0443	23.4416	46.6469	14	92.6209	73.7583	111.4836
5	42.1715	29.4436	54.8995	15	95.3604	76.2209	114.5000
6	49.0516	35.3246	62.7785	16	97.4029	78.0595	116.7464
7	55.7185	41.0884	70.3486	17	98.8761	79.3869	118.3653
8	62.1779	46.7230	77.6328	18	99.9166	80.3251	119.5080
9	68.4031	52.1930	84.6133	19	100.6439	80.9813	120.3065
10	74.3287	57.4310	91.2263	20	101.1512	81.4391	120.8634



<그림 Ⅲ-8> 데이터 세트 1에 대한 STX 모형의 95% 신뢰구간

<표 Ⅲ-12> DPF 모형의 95% 신뢰구간(DS 2)

Time	$\widehat{m}(t)$	LC	UC
1	9.1317	3.2089	15.0544
2	12.6825	5.7026	19.6624
3	17.1280	9.0165	25.2395
4	22.3485	13.0830	31.6141
5	28.0333	17.6560	38.4106
6	33.7329	22.3494	45.1163
7	38.9897	26.7514	51.2281
8	43.4751	30.5520	56.3983
9	47.0529	33.6085	60.4973
10	49.7548	35.9298	63.5798
11	51.7108	37.6167	65.8050
12	53.0833	38.8033	67.3632



<그림 Ⅲ-9> 데이터 세트 2에 대한 DPF 모형의 95% 신뢰구간

Ⅳ. 순차적 확률비 검정

4.1. Wald의 순차적 확률비 검정

순차적 확률비 검정(Sequential probability ratio test; SPRT) 혹은 축차확률비 검정으로 불리는 이 통계적 기법은 Wald(1947)가 1947년 육·해군의 새로운 기술에 대한 보고서를 작성하는 과정에서 제안하여 통계적 추론 방법의 혁신으로 주목받았다. 확률변수 x에 대하여 $f(x,\theta)$가 주어질 때, 알려지지 않은 모수 θ에 대하여 $H_0 : \theta = \theta_0$, $H_1 : \theta = \theta_1$을 가설로 설정한다. 이때, $p_0 = f(x,\theta_0)$, $p_1 = f(x,\theta_1)$로 주어지며 분포 vs 분포의 기준으로 설정하거나 특정 모수를 기준으로 가설을 설정할 수 있다.

귀무가설의 기각 및 채택 여부를 판단할 순차적 확률비 검정통계량은 p_0와 p_1의 확률비인 p_1/p_0 으로 정의하며 검정을 통해 세 가지의 결론을 내릴 수 있다.

$$B < \frac{p_1}{p_0} < A. \tag{14}$$

첫 번째, 검정통계량인 p_1/p_0이 수식 (14)를 만족한다면 다음 검정을 할 때까지 데이터 수집을 계속 진행한다(Continue)는 결론을 내리게 된다.

$$\frac{p_1}{p_0} \geqq A. \tag{15}$$

두 번째, 검정통계량이 수식 (15)를 만족한다면 귀무가설 H_0는 기각하고, 대립가설 H_1을 채택한다.

$$\frac{p_1}{p_0} \leqq B. \tag{16}$$

세 번째, 검정통계량이 수식 (16)을 만족한다면 귀무가설 H_0는 채택하고, 대립가설 H_1을 기각한다.

여기서 A와 B는 귀무가설의 채택역 및 기각역을 결정하는 임계값으로 상수이며 위험확률(Risk probability)인 α와 β에 의존적이다. 통계적으로 α는 1종 오류, β는 2종 오류를 의미하며 품질 관리 분야에서는 α와 β를 생산자 위험(Producer's risk)과 소비자 위험(Consumer's risk)이라고도 일컫는다.

A와 B를 α와 β로 표현하면 다음 수식(17)-(20)과 같다.

$$1-\beta \geqq A\alpha, \tag{17}$$

$$A \leqq \frac{1-\beta}{\alpha}, \tag{18}$$

$$\beta \leqq (1-\alpha)B, \tag{19}$$

$$B \geqq \frac{\beta}{1-\alpha}. \tag{20}$$

4.2. 순차적 확률비 검정을 적용한 소프트웨어 신뢰성 추정

4.2.1. 소프트웨어 신뢰성에서의 순차적 확률비 검정 이론

Stiber(1997)는 소프트웨어 신뢰성에 순차적 확률비 검정을 처음으로 적용하였다. 일반적인 NHPP 소프트웨어 신뢰성 성장 모형은 다음과 같이 표현할 수 있다.

$$\Pr\{N(t)=n\}=\frac{[m(t)]^n e^{-m(t)}}{n!},\ \ n=0,1,2,\ldots.$$

여기서 $m(t)$는 시간 t에서 검출된 기대 결함 수를 의미하며 다음과 같은 수식으로 계산한다.

$$m(t)=\int_0^t \lambda(s)ds.$$

$N_L(t)$와 $N_U(t)$는 수식 (21)과 같이 나타낼 수 있다.

$$N_L(t)=at-b_1,\ \ N_U(t)=at+b_2. \tag{21}$$

여기서 a, b_1, b_2는 다음과 같이 주어진다.

$$a=\frac{\lambda_1-\lambda_0}{\ln\left(\frac{\lambda_1}{\lambda_0}\right)}, \tag{22}$$

$$b_1=\frac{\ln\left(\frac{1-\alpha}{\beta}\right)}{\ln\left(\frac{\lambda_1}{\lambda_0}\right)},\ \ b_2=\frac{\ln\left(\frac{1-\beta}{\alpha}\right)}{\ln\left(\frac{\lambda_1}{\lambda_0}\right)}. \tag{23}$$

수식 (22)와 (23)의 λ_0와 λ_1은 다음과 같이 주어진다.

$$\lambda_0=\frac{\lambda\ln(q)}{q-1},\ \ \lambda_1=q\frac{\lambda\ln(q)}{q-1}.$$

여기서 $q=\lambda_0/\lambda_1$이다. 소프트웨어 신뢰성에서의 순차적 확률비 검정통계량 p_1/p_0을 계산하기 위해 p_0와 p_1은 다음과 같이 쓸 수 있다.

$$p_0=\frac{e^{-m_0(t)}[m_0(t)]^{N(t)}}{N(t)!},\ \ p_1=\frac{e^{-m_1(t)}[m_1(t)]^{N(t)}}{N(t)!}. \tag{24}$$

수식 (24)를 수식 (14)에 대입하면 $N(t)$에 대하여 다음과 같이 재정의할 수 있다.

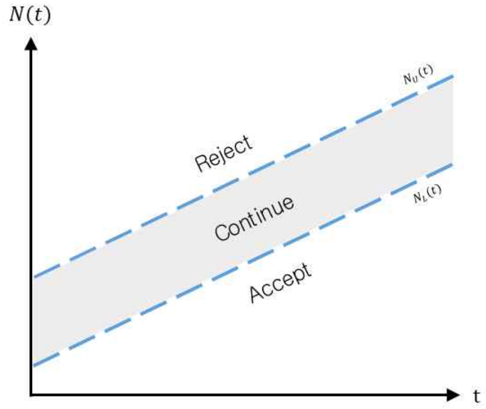
$$B < \frac{\dfrac{e^{-m_0(t)}[m_0(t)]^{N(t)}}{N(t)!}}{\dfrac{e^{-m_1(t)}[m_1(t)]^{N(t)}}{N(t)!}} < A.$$

수식 (14)-(17)을 기반으로 A와 B는 α와 β로 표현 할 수 있으며 최종적으로 수식 (25)와 같이 정리할 수 있다.

$$\frac{\beta}{1-\alpha} < \frac{\dfrac{e^{-m_0(t)}[m_0(t)]^{N(t)}}{N(t)!}}{\dfrac{e^{-m_1(t)}[m_1(t)]^{N(t)}}{N(t)!}} < \frac{1-\beta}{\alpha},$$

$$\frac{\ln\left(\dfrac{\beta}{1-\alpha}\right)+m_1(t)-m_0(t)}{\ln m_1(t)-\ln m_0(t)} < N(t) < \frac{\ln\left(\dfrac{1-\beta}{\alpha}\right)+m_1(t)-m_0(t)}{\ln m_1(t)-\ln m_0(t)}. \tag{25}$$

순차적 확률비 검정을 시행하기 위해서는 가설을 설정하고, 채택역과 기각역을 계산하기 위해 위험확률인 α와 β를 가정해야 한다. 귀무가설과 대립가설은 분포함수, $\lambda(t)$, 모수 등을 기준으로 설정할 수 있다. 본 논문에서는 모수를 기준으로 가설을 설정한다. 예를 들면 모수 a의 추정치를 기준으로 δ만큼의 등간척도를 구성하여 $H_0 : a-\delta$, $H_1 : a+\delta$인 가설을 설정할 수 있다. 이때, 위험확률은 $0 \leqq \alpha < 1-\beta$를 만족하며 일반적으로 α, β는 $\alpha=\beta$이면서 0.1 혹은 0.05로 설정한다(Wald 1947, Stiber 1997). 모수를 기준으로 가설을 설정할 경우, 모형의 전체 혹은 일부의 모수에 대해 등간척도를 적용하여 $m_0(t)$와 $m_1(t)$를 구성한다.



<그림 Ⅳ-1> 순차적 확률비 검정의 구조

<그림 Ⅳ-1>은 4.1의 순차적 확률비 검정 기각역과 채택역을 시각화한다. <그림 Ⅳ-1>의 $N_L(t)$와 $N_U(t)$는 신뢰 영역의 하한과 상한 경계를 나타내며 검정통계량 값이 속하는 영역을 기준으로 소프트웨어의 신뢰성을 판단한다. 만약 $N(t) \leq N_L(t)$로 $N(t)$가 “Accept”에 존재하면 해당 시스템을 채택한다. 반면에 $N(t) \geq N_U(t)$로 $N(t)$가 “Reject”에 존재하면 해당 시스템을 기각한다. 여기서 $N(t)$는 시간 t에서의 실제 고장 수를 의미한다.

순차적 확률비 검정은 수식 (25)를 기준으로 데이터를 수집하는 매 순간마다 검정하여 결론을 내린다. 고전적인 가설 검정에 비하여 아주 적은 데이터로 시점마다 검정하기 때문에 조기에 결론을 내리고, 데이터 수집을 종료할 수 있어 훨씬 경제적이다. 본 논문에서는 소프트웨어의 신뢰성을 판단하는 방법으로 순차적 확률비 검정을 제안하며 수치적 예제를 통해 순차적 확률비 검정의 효율성을 입증한다. 또한, 순차적 확률비 검정의 통계량을 추정하기 위해 사용되는 가정의 객관적인 기준을 제시한다.

4.2.2. 수치적 예제

본 논문에서는 STX 모형과 DPF 모형을 이용하여 실제 소프트웨어 데이터의 신뢰성을 순차적 확률비 검정으로 판단하고자 하며 절차는 다음과 같다. 먼저 각 모형의 모수에 순차적 확률비 검정을 개별적으로 적용해 모수별 순차적 확률비 검정 결과 및 δ의 수준에 따른 채택역과 기각역의 민감도를 살펴본다. 최종적으로 민감한 모수를 제외한 나머지 모수를 통합 적용하여 순차적 확률비 검정 시행 및 소프트웨어 신뢰성을 판단한다.

순차적 확률비 검정을 적용할 데이터 세트는 STX 모형과 DPF 모형의 적합도를 추정할 때 사용한 데이터 세트 1(DS 1)과 데이터 세트 2(DS 2)를 사용한다.

4.2.2.1. δ의 수준에 따른 모형의 모수별 순차적 확률비 검정의 민감도 분석

Wald(1947)와 Stiber(1997)는 위험확률인 α, β에 대해서 0.05 혹은 0.1의 수준을 권장하였으나 δ에 대해서는 구체적인 기준을 제시하지 않았다. Prasad 외(2013), Gutta 외(2014), Kotha 외(2014), Smitha 외(2014), Murali Mohan 외(2015)가 소프트웨어 신뢰성 성장 모형에 순차적 확률비 검정을 적용한 연구를 살펴보면 δ의 가정에 대한 객관적인 근거가 부족하다. 또한, 순차적 확률비 검정은 전체 모수 혹은 일부 모수를 선택하여 $m_0(t)$와 $m_1(t)$를 구성할 수 있는데 모형마다 결과에 영향을 주는 민감한 모수가 존재하므로 모수의 선별이 중요하다. 본 논문에서는 STX 모형과 DPF 모형에 대하여 등간척도를 구성하는 δ의 수준에 따라 $m_0(t)$, $m_1(t)$, $m(t)$의 변화를 살펴본 후, 순차적 확률비 검정에서 사용할 각 모형의 모수 선택 및 적절한 δ의 수준을 제안한다.

<표 IV-1>은 순차적 확률비 검정을 적용하기 위해 각 모형의 모수에 대한 δ의 가정을 나타내고 있다. 예를 들어 STX 모형의 모수 a에 대하여 $a_0 = \hat{a} - \delta$, $a_1 = \hat{a} + \delta$와 같은 등간척도를 구성하는데 이때 $\delta = 0.000005$, 0.000010, 0.000015인 세 개의 수준으로 가정하며 위험확률인 α와 β는 각각 0.1로 설정한다.

<표 Ⅳ-1> 순차적 확률비 검정을 위한 모수별 가정

<table>
<tr><th rowspan="2">Model</th><th rowspan="2">Parameter</th><th colspan="3">δ</th><th colspan="2">Risk Probability</th></tr>
<tr><th>Case 1</th><th>Case 2</th><th>Case 3</th><th>α</th><th>β</th></tr>
<tr><td rowspan="6">STX</td><td>a</td><td>0.000005</td><td>0.000010</td><td>0.000015</td><td rowspan="10">0.1</td><td rowspan="10">0.1</td></tr>
<tr><td>b</td><td>0.1</td><td>0.5</td><td>1.0</td></tr>
<tr><td>α</td><td>0.013</td><td>0.015</td><td>0.017</td></tr>
<tr><td>β</td><td>500</td><td>1000</td><td>1500</td></tr>
<tr><td>N</td><td>1</td><td>5</td><td>10</td></tr>
<tr><td>t_0</td><td>0.0000005</td><td>0.0000010</td><td>0.0000015</td></tr>
<tr><td rowspan="4">DPF</td><td>a</td><td>1</td><td>5</td><td>10</td></tr>
<tr><td>b</td><td>0.0001</td><td>0.0005</td><td>0.0010</td></tr>
<tr><td>c</td><td>0.40</td><td>0.45</td><td>0.50</td></tr>
<tr><td>h</td><td>0.1</td><td>0.5</td><td>1.0</td></tr>
</table>

<표 IV-2>부터 <표 IV-7>까지는 STX 모형의 모수에 대하여 δ의 수준에 따른 순차적 확률비 검정의 결과를 나타내며 <그림 IV-2>부터 <그림 IV-7>은 각 모수의 δ에 따른 $m_0(t)$, $m(t)$, $m_1(t)$를 나타내고, <그림 IV-8>과 <그림 IV-9>는 각 모수의 δ에 따른 채택역과 기각역을 나타낸다.

<표 IV-2>의 a와 <표 IV-6>의 N은 모든 δ에 대해서 “Continue(다음 검정을 할 때까지 데이터를 계속 수집한다)”라는 결론을 도출했다. <표 IV-3>의 b, <표 IV-5>의 β, <표 IV-7>의 t_0는 채택역과 기각역을 산출할 만큼의 $m_0(t)$와 $m_1(t)$의 차이가 있지 않아 채택역과 기각역을 보일 수 없어 생략한다. 마지막으로 <표 IV-4>의 α는 모든 δ에 대해 $t=1$인 시점에서 “Reject(신뢰성을 기각하며 테스트를 종료한다)”라는 결론이 나타났다. 그러나 α는 순차적 확률비 검정에 매우 민감하게 반응한다. <그림 IV-4>를 보면 모수 α의 $m_0(t)$는 다른 모수와 달리 $m_0(t)$가 $m_1(t)$보다 위에 분포하고 있다. 민감도가 심한 모수는 순차적 확률비 검정을 적용하기 어려워 δ의 범위가 매우 제한적이다. 모수 α에 적용되는 δ도 0.013, 0.015, 0.017로 다른 모수에 비하여 δ의 범위가 매우 좁다. 따라서 STX 모형에 순차적 확률비 검정을 시행할 경우, 왜곡이 우려되므로 α에 대해 엄격한 판단이 필요하다.

<표 IV-8>부터 <표 IV-11>까지는 DPF 모형의 모수에 대하여 δ의 수준에 따른 순차적 확률비 검정의 결과를 나타내며 <그림 IV-10>부터 <그림 IV-13>은 각 모수의 δ에 따른 $m_0(t)$, $m(t)$, $m_1(t)$를 나타내고, <그림 IV-14>부터 <그림 IV-16>은 각 모수의 δ에 따른 채택역과 기각역을 나타낸다.

<표 IV-8>의 a, <표 IV-9>의 b, <표 IV-11>의 h는 모든 δ에 대해서 “Continue(다음 검정을 할 때까지 데이터를 계속 수집한다)”라는 결론이 나타났다.

<표 IV-10>에서 c는 모든 δ에 대해 $t=1$인 시점에서 “Accept(신뢰성을 채택하며 테스트를 종료한다)”라는 결론이 나타났다. 그러나 <그림 IV-12>에서 c에 대한 $m_0(t)$는 다른 모수와 달리 $m_0(t)$의 편차가 큰 것으로 나타났다. 이는 모수 c가 감소할수록 평균값함수 $m(t)$가 민감하게 반응함을 의미한다. 따라서 DPF 모형으로 순차적 확률비 검정을 시행할 경우, 왜곡이 우려되므로 c에 대한 엄격한 판단이 필요하다.

<표 Ⅳ-2> STX 모형 모수 a의 δ에 따른 순차적 확률비 검정 결과(DS 1)

T	$N(t)$	$m(t)$	$\delta=0.000005$					$\delta=0.000010$					$\delta=0.000015$				
			$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result
1	16	11.0907	11.0227	11.1558	-171.8603	194.0385		10.9514	11.2183	-80.1607	102.3293		10.8765	11.2783	-49.4973	71.6497	
2	24	19.7148	19.5938	19.8305	-163.2376	202.6615		19.4671	19.9416	-71.5418	110.9485		19.3340	20.0482	-40.8846	80.2625	
3	27	27.6015	27.4322	27.7636	-155.3551	210.5502		27.2548	27.9190	-63.6611	118.8322		27.0685	28.0683	-33.0092	88.1399	
4	33	35.0443	34.8292	35.2500	-147.9341	218.0125		34.6040	35.4473	-56.2330	126.2809		34.3675	35.6369	-25.5830	95.5797	
5	41	42.1715	41.9129	42.4190	-140.8961	225.2270		41.6420	42.6563	-49.1537	133.4480		41.3575	42.8843	-18.4941	102.7267	
6	49	49.0516	48.7512	49.3389	-134.3008	232.3898		48.4365	49.6145	-42.4188	140.4652		48.1061	49.8792	-11.7164	109.6910	
7	54	55.7185	55.3787	56.0436	-128.3886	239.8095		55.0226	56.3552	-36.1308	147.5032		54.6485	56.6545	-5.3061	116.5971	
8	58	62.1779	61.8021	62.5371	-123.6756	248.0134		61.4082	62.8814	-30.5434	154.8271		60.9942	63.2118	0.5709	123.6219	
9	69	68.4031	67.9975	68.7906	-121.0939	257.8804		67.5718	69.1614	-26.1337	162.8607		67.1239	69.5169	5.5904	131.0365	
10	75	74.3286	73.9032	74.7342	-122.1794	270.8153		73.4558	75.1216	-23.6983	172.2695		72.9842	75.4924	9.2043	139.2582	
11	81	79.8466	79.4166	80.2553	-129.3036	288.9740	Continue	78.9629	80.6445	-24.4693	184.0707	Continue	78.4831	81.0158	10.5636	148.9219	Continue
12	86	84.8180	84.4031	85.2106	-145.9448	315.5572		83.9632	85.5827	-30.2432	199.7840		83.4959	85.9362	8.4380	160.9823	
13	90	89.1055	88.7264	89.4621	-176.9975	355.1849		88.3221	89.7981	-43.5170	221.6331		87.8898	90.1155	1.1391	176.8570	
14	93	92.6209	92.2939	92.9264	-229.1320	414.3516		91.9428	93.2124	-67.6423	252.7946		91.5647	93.4810	-13.5663	198.6054	
15	96	95.3604	95.0932	95.6084	-311.2383	501.9395		94.8042	95.8392	-107.0296	297.6711		94.4907	96.0546	-38.5793	229.1202	
16	98	97.4029	97.1938	97.5958	-434.9820	629.7714		96.9663	97.7743	-167.4261	362.1656		96.7178	97.9399	-77.6545	272.3097	
17	99	98.8761	98.7175	99.0216	-615.4733	813.2123		98.5441	99.1557	-256.2654	453.9646		98.3536	99.2796	-135.6416	333.2733	
18	100	99.9166	99.7986	100.0244	-872.0280	1071.8510		99.6690	100.1235	-383.0646	582.8567		99.5260	100.2147	-218.7595	418.4994	
19	100	100.6439	100.5569	100.7232	-1229.0061	1430.2861		100.4611	100.7958	-559.8525	761.1092		100.3549	100.8625	-334.8868	536.1038	
20	100	101.1512	101.0872	101.2095	-1716.7297	1919.0263		101.0165	101.2627	-801.6303	1003.9095		100.9381	101.3115	-493.8669	696.1163	

<표 Ⅳ-3> STX 모형 모수 b의 δ에 따른 순차적 확률비 검정 결과(DS 1)

T	$N(t)$	$m(t)$	$\delta=0.1$					$\delta=0.5$					$\delta=1.0$				
			$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result
1	16	11.0907	11.0907	11.0907	-	-		11.0907	11.0907	-	-		11.0907	11.0907	-	-	
2	24	19.7148	19.5515	19.8794	-	-		18.9117	20.5520	-	-		18.1413	21.4247	-	-	
3	27	27.6015	27.2400	27.9678	-	-		25.8408	29.4822	-	-		24.1924	31.4909	-	-	
4	33	35.0443	34.4662	35.6320	-	-		32.2475	38.0832	-	-		29.6739	41.3850	-	-	
5	41	42.1715	41.3653	42.9934	-	-		38.2912	46.4428	-	-		34.7671	51.1409	-	-	
6	49	49.0516	48.0102	50.1151	-	-		44.0586	54.5983	-	-		39.5702	60.7394	-	-	
7	54	55.7185	54.4397	57.0257	-	-		49.6017	62.5420	-	-		44.1436	70.0636	-	-	
8	58	62.1779	60.6669	63.7214	-	-		54.9514	70.2031	-	-		48.5267	78.8149	-	-	
9	69	68.4031	66.6778	70.1595	-	-		60.1223	77.4195	-	-		52.7463	86.4715	-	-	
10	75	74.3286	72.4271	76.2488	-	-		65.1141	83.9283	-	-		56.8195	92.4657	-	-	
11	81	79.8466	77.8344	81.8489	-	-		69.9113	89.4243	-	-		60.7566	96.5797	-	-	
12	86	84.8180	82.7887	86.7913	-	-		74.4822	93.6984	-	-		64.5612	99.1006	-	-	
13	90	89.1055	87.1688	90.9311	-	-		78.7802	96.7573	-	-		68.2308	100.5424	-	-	
14	93	92.6209	90.8770	94.2055	-	-		82.7481	98.8049	-	-		71.7577	101.3460	-	-	
15	96	95.3604	93.8743	96.6600	-	-		86.3270	100.1188	-	-		75.1286	101.7951	-	-	
16	98	97.4029	96.1939	98.4227	-	-		89.4690	100.9459	-	-		78.3262	102.0505	-	-	
17	99	98.8761	97.9259	99.6527	-	-		92.1484	101.4652	-	-		81.3301	102.1992	-	-	
18	100	99.9166	99.1866	100.4978	-	-		94.3693	101.7939	-	-		84.1196	102.2880	-	-	
19	100	100.6439	100.0902	101.0757	-	-		96.1635	102.0047	-	-		86.6758	102.3424	-	-	
20	100	101.1512	100.7333	101.4717	-	-		97.5826	102.1421	-	-		88.9845	102.3765	-	-	

<표 Ⅳ-4> STX 모형 모수 α의 δ에 따른 순차적 확률비 검정 결과(DS 1)

T	$N(t)$	$m(t)$	$\delta=0.013$					$\delta=0.015$					$\delta=0.017$				
			$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result
1	16	11.0907	14.1110	8.7169	15.7597	6.6368		14.6437	8.3998	15.1872	7.2808		15.1965	8.0942	14.7631	7.7868	
2	24	19.7148	23.5682	16.4914	-	-		24.2245	16.0446	-	-		24.8990	15.6099	-	-	
3	27	27.6015	31.8152	23.9459	-	-		32.5183	23.4282	-	-		33.2369	22.9217	-	-	
4	33	35.0443	39.3628	31.1995	-	-		40.0729	30.6466	-	-		40.7958	30.1036	-	-	
5	41	42.1715	46.4279	38.3054	-	-		47.1198	37.7429	-	-		47.8220	37.1887	-	-	
6	49	49.0516	53.1254	45.2901	-	-		53.7815	44.7376	-	-		54.4457	44.1919	-	-	
7	54	55.7185	59.5186	52.1610	-	-		60.1258	51.6343	-	-		60.7392	51.1128	-	-	
8	58	62.1779	65.6340	58.9038	-	-		66.1824	58.4156	-	-		66.7355	57.9315	-	-	
9	69	68.4031	71.4627	65.4746	-	-		71.9454	65.0353	-	-		72.4313	64.5990	-	-	
10	75	74.3286	76.9575	71.7896	-	-	Reject	77.3701	71.4068	-	-	Reject	77.7849	71.0260	-	-	Reject
11	81	79.8466	82.0317	77.7197	-	-		82.3732	77.3976	-	-		82.7160	77.0768	-	-	
12	86	84.8180	86.5709	83.1007	-	-		86.8438	82.8396	-	-		87.1175	82.5793	-	-	
13	90	89.1055	90.4624	87.7690	-	-		90.6730	87.5652	-	-		90.8841	87.3618	-	-	
14	93	92.6209	93.6380	91.6149	-	-		93.7955	91.4611	-	-		93.9532	91.3076	-	-	
15	96	95.3604	96.1036	94.6230	-	-		96.2185	94.5100	-	-		96.3335	94.3972	-	-	
16	98	97.4029	97.9370	96.8718	-	-		98.0194	96.7904	-	-		98.1019	96.7090	-	-	
17	99	98.8761	99.2566	98.4970	-	-		99.3153	98.4388	-	-		99.3740	98.3807	-	-	
18	100	99.9166	100.1874	99.6464	-	-		100.2292	99.6049	-	-		100.2709	99.5634	-	-	
19	100	100.6439	100.8375	100.4507	-	-		100.8673	100.4210	-	-		100.8971	100.3913	-	-	
20	100	101.1512	101.2906	101.0121	-	-		101.3121	100.9907	-	-		101.3335	100.9693	-	-	

<표 Ⅳ-5> STX 모형 모수 β의 δ에 따른 순차적 확률비 검정 결과(DS 1)

T	$N(t)$	$m(t)$	$\delta=500$					$\delta=1000$					$\delta=1500$				
			$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result
1	16	11.0907	11.1521	11.0323	–	–		11.2168	10.9767	–	–		11.2853	10.9236	–	–	
2	24	19.7148	19.8239	19.6110	–	–		19.9390	19.5121	–	–		20.0606	19.4177	–	–	
3	27	27.6015	27.7543	27.4562	–	–		27.9154	27.3178	–	–		28.0857	27.1856	–	–	
4	33	35.0443	35.2382	34.8598	–	–		35.4428	34.6840	–	–		35.6589	34.5162	–	–	
5	41	42.1715	42.4049	41.9497	–	–		42.6508	41.7382	–	–		42.9108	41.5363	–	–	
6	49	49.0516	49.3225	48.7939	–	–		49.6082	48.5483	–	–		49.9100	48.3138	–	–	
7	54	55.7185	56.0250	55.4270	–	–		56.3480	55.1491	–	–		56.6893	54.8837	–	–	
8	58	62.1779	62.5166	61.8555	–	–		62.8734	61.5481	–	–		63.2502	61.2544	–	–	
9	69	68.4031	68.7684	68.0552	–	–		69.1528	67.7231	–	–		69.5582	67.4055	–	–	
10	75	74.3286	74.7111	73.9637	–	–		75.1127	73.6149	–	–		75.5354	73.2808	–	–	
11	81	79.8466	80.2320	79.4778	–	–		80.6355	79.1244	–	–		81.0588	78.7850	–	–	
12	86	84.8180	85.1883	84.4623	–	–		85.5742	84.1200	–	–		85.9770	83.7903	–	–	
13	90	89.1055	89.4419	88.7807	–	–		89.7904	88.4665	–	–		90.1521	88.1624	–	–	
14	93	92.6209	92.9091	92.3409	–	–		93.2059	92.0685	–	–		93.5118	91.8035	–	–	
15	96	95.3604	95.5945	95.1316	–	–		95.8340	94.9079	–	–		96.0792	94.6889	–	–	
16	98	97.4029	97.5850	97.2240	–	–		97.7702	97.0481	–	–		97.9588	96.8752	–	–	
17	99	98.8761	99.0135	98.7405	–	–		99.1527	98.6066	–	–		99.2937	98.4743	–	–	
18	100	99.9166	100.0184	99.8157	–	–		100.1212	99.7157	–	–		100.2250	99.6167	–	–	
19	100	100.6439	100.7188	100.5695	–	–		100.7942	100.4957	–	–		100.8701	100.4223	–	–	
20	100	101.1512	101.2062	101.0965	–	–		101.2615	101.0421	–	–		101.3171	100.9879	–	–	

<표 Ⅳ-6> STX 모형 모수 N의 δ에 따른 순차적 확률비 검정 결과(DS 1)

T	$N(t)$	$m(t)$	$\delta=1$					$\delta=5$					$\delta=10$				
			$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result
1	16	11.0907	10.9824	11.1990	-101.4534	123.6341	Continue	10.5494	11.6320	-11.4097	33.5735	Continue	10.0081	12.1733	-0.1635	22.2743	Continue
2	24	19.7148	19.5223	19.9072	-92.8296	132.2579		18.7526	20.6770	-2.7925	42.1907		17.7904	21.6392	8.4331	30.8709	
3	27	27.6015	27.3321	27.8710	-84.9431	140.1444		26.2544	28.9487	5.0880	50.0712		24.9073	30.2958	16.2947	38.7325	
4	33	35.0443	34.7022	35.3863	-77.5006	147.5869		33.3339	36.7546	12.5248	57.5080		31.6235	38.4650	23.7138	46.1516	
5	41	42.1715	41.7599	42.5832	-70.3736	154.7140		40.1133	44.2298	19.6464	64.6296		38.0550	46.2880	30.8184	53.2561	
6	49	49.0516	48.5728	49.5304	-63.4938	161.5938		46.6575	51.4456	26.5210	71.5042		44.2635	53.8396	37.6765	60.1143	
7	54	55.7185	55.1746	56.2624	-56.8270	168.2605		52.9991	58.4379	33.1826	78.1658		50.2796	61.1574	44.3222	66.7600	
8	58	62.1779	61.5709	62.7848	-50.3679	174.7197		59.1432	65.2126	39.6369	84.6201		56.1085	68.2473	50.7610	73.1988	
9	69	68.4031	67.7354	69.0708	-44.1428	180.9447		65.0646	71.7417	45.8572	90.8404		61.7261	75.0802	56.9664	79.4042	
10	75	74.3286	73.6031	75.0542	-38.2175	186.8700		70.7009	77.9564	51.7780	96.7612		67.0732	81.5841	62.8731	85.3109	
11	81	79.8466	79.0672	80.6260	-32.6997	192.3879		75.9496	83.7437	57.2916	102.2748		72.0525	87.6407	68.3735	90.8113	
12	86	84.8180	83.9901	85.6460	-27.7284	197.3591		80.6784	88.9577	62.2591	107.2422		76.5387	93.0974	73.3291	95.7669	
13	90	89.1055	88.2357	89.9753	-23.4411	201.6465		84.7566	93.4545	66.5432	111.5263		80.4076	97.8034	77.6029	100.0407	
14	93	92.6209	91.7168	93.5250	-19.9258	205.1618		88.1004	97.1415	70.0558	115.0389		83.5799	101.6620	81.1071	103.5449	
15	96	95.3604	94.4296	96.2913	-17.1863	207.9012		90.7062	100.0147	72.7931	117.7763		86.0520	104.6689	83.8379	106.2757	
16	98	97.4029	96.4522	98.3537	-15.1439	209.9436		92.6490	102.1569	74.8340	119.8171		87.8951	106.9108	85.8739	108.3117	
17	99	98.8761	97.9109	99.8413	-13.6708	211.4167		94.0503	103.7019	76.3060	121.2891		89.2245	108.5277	87.3424	109.7802	
18	100	99.9166	98.9412	100.8919	-12.6304	212.4571		95.0400	104.7932	77.3456	122.3288		90.1634	109.6697	88.3795	110.8173	
19	100	100.6439	99.6615	101.6263	-11.9031	213.1845		95.7318	105.5560	78.0723	123.0555		90.8197	110.4681	89.1045	111.5423	
20	100	101.1512	100.1639	102.1386	-11.3957	213.6918		96.2144	106.0881	78.5793	123.5624		91.2775	111.0249	89.6102	112.0480	

<표 Ⅳ-7> STX 모형 모수 t_0의 δ에 따른 순차적 확률비 검정 결과(DS 1)

T	$N(t)$	$m(t)$	$\delta=0.0000005$					$\delta=0.0000010$					$\delta=0.0000015$				
			$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result
1	16	11.0907	11.0907	11.0907	-	-		11.0907	11.0907	-	-		11.0907	11.0907	-	-	
2	24	19.7148	19.7148	19.7148	-	-		19.7148	19.7148	-	-		19.7148	19.7148	-	-	
3	27	27.6015	27.6015	27.6015	-	-		27.6015	27.6015	-	-		27.6015	27.6015	-	-	
4	33	35.0443	35.0443	35.0442	-	-		35.0443	35.0442	-	-		35.0443	35.0442	-	-	
5	41	42.1715	42.1715	42.1715	-	-		42.1715	42.1715	-	-		42.1715	42.1715	-	-	
6	49	49.0516	49.0516	49.0516	-	-		49.0516	49.0515	-	-		49.0516	49.0516	-	-	
7	54	55.7185	55.7185	55.7185	-	-		55.7185	55.7185	-	-		55.7185	55.7185	-	-	
8	58	62.1779	62.1779	62.1779	-	-		62.1779	62.1779	-	-		62.1779	62.1779	-	-	
9	69	68.4031	68.4031	68.4031	-	-		68.4031	68.4031	-	-		68.4031	68.4031	-	-	
10	75	74.3286	74.3286	74.3286	-	-		74.3287	74.3286	-	-		74.3287	74.3286	-	-	
11	81	79.8466	79.8466	79.8466	-	-		79.8466	79.8466	-	-		79.8466	79.8466	-	-	
12	86	84.8180	84.8181	84.8180	-	-		84.8181	84.8180	-	-		84.8181	84.8180	-	-	
13	90	89.1055	89.1055	89.1055	-	-		89.1055	89.1055	-	-		89.1055	89.1055	-	-	
14	93	92.6209	92.6209	92.6209	-	-		92.6209	92.6209	-	-		92.6209	92.6209	-	-	
15	96	95.3604	95.3604	95.3604	-	-		95.3604	95.3604	-	-		95.3604	95.3604	-	-	
16	98	97.4029	97.4029	97.4029	-	-		97.4029	97.4029	-	-		97.4029	97.4029	-	-	
17	99	98.8761	98.8761	98.8761	-	-		98.8761	98.8761	-	-		98.8761	98.8761	-	-	
18	100	99.9166	99.9166	99.9166	-	-		99.9166	99.9166	-	-		99.9166	99.9166	-	-	
19	100	100.6439	100.6439	100.6439	-	-		100.6439	100.6439	-	-		100.6439	100.6439	-	-	
20	100	101.1512	101.1512	101.1512	-	-		101.1512	101.1512	-	-		101.1512	101.1512	-	-	

<표 Ⅳ-8> DPF 모형 모수 a의 δ에 따른 순차적 확률비 검정 결과(DS 2)

T	$N(t)$	$m(t)$	$\delta=1$					$\delta=5$					$\delta=10$				
			$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result
1	10	9.1317	9.0496	9.2139	−112.9620	131.2247	Continue	8.7219	9.5444	−15.2548	33.5088	Continue	8.3121	9.9624	−3.0196	21.2443	Continue
2	12	12.6825	12.4893	12.8776	−59.0798	84.4448		11.7361	13.6794	−1.6575	27.0235		10.8348	14.7320	5.5327	19.8349	
3	16	17.1280	16.7771	17.4845	−36.0716	70.3284		15.4272	18.9679	6.5023	27.7711		13.8541	20.9579	11.8535	22.4699	
4	22	22.3485	21.8018	22.9053	−22.1495	66.8475		19.7132	25.2355	13.4639	31.2578		17.3140	28.3876	17.9525	26.8403	
5	28	28.0333	27.2773	28.8031	−12.3364	68.4030		24.3905	32.0208	19.9604	36.1052		21.0851	36.3535	23.9959	32.0632	
6	36	33.7329	32.7876	34.6930	−5.1670	72.6297		29.1591	38.6780	25.9170	41.4724		24.9754	43.9644	29.6943	37.4653	
7	40	38.9897	37.9030	40.0891	−0.1989	78.1705		33.6923	44.6035	31.0611	46.7252		28.7662	50.4705	34.6985	42.5152	
8	43	43.4751	42.3054	44.6531	2.7855	84.1518		37.7223	49.4342	35.1886	51.4408		32.2584	55.5195	38.7948	46.8884	
9	44	47.0529	45.8519	48.2573	4.0715	90.0172		41.0950	53.0955	38.2633	55.4153		35.3118	59.1487	41.9506	50.4696	
10	50	49.7548	48.5582	50.9507	4.0601	95.4296		43.7765	55.7182	40.3983	58.6167		37.8609	61.6141	44.2654	53.2894	
11	51	51.7108	50.5380	52.8803	3.2022	100.1985		45.8217	57.5207	41.7867	61.1127		39.9074	63.2241	45.8995	55.4501	
12	55	53.0833	51.9414	54.2207	1.9101	104.2357		47.3321	58.7236	42.6346	63.0120		41.4984	64.2477	47.0201	57.0740	

<표 Ⅳ-9> DPF 모형 모수 b의 δ에 따른 순차적 확률비 검정 결과(DS 2)

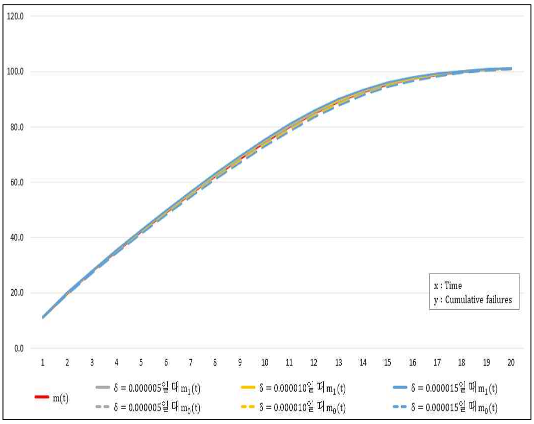
T	$N(t)$	$m(t)$	$\delta=0.0001$					$\delta=0.0005$					$\delta=0.001$				
			$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result
1	10	9.1317	9.0548	9.2090	−120.9845	139.2480	Continue	8.7525	9.5235	−16.8917	35.1568	Continue	8.3857	9.9280	−3.8786	22.1489	Continue
2	12	12.6825	12.4849	12.8822	−57.4431	82.8082		11.7162	13.7029	−1.3443	26.7115		10.8044	14.7764	5.6687	19.7051	
3	16	17.1280	16.7676	17.4927	−34.7782	69.0334		15.3704	18.9918	6.7316	27.5030		13.7287	20.9481	11.8851	22.2848	
4	22	22.3485	21.8029	22.8987	−22.4635	67.1561		19.6731	25.1358	13.3262	31.2598		17.1545	27.9811	17.6373	26.6190	
5	28	28.0333	27.3171	28.7495	−14.9635	71.0179		24.4706	31.5955	19.2832	36.4798		21.0213	35.0431	23.1382	31.7372	
6	36	33.7329	32.9018	34.5535	−11.1370	78.5789		29.5032	37.7070	24.4819	42.3931		25.2056	41.2809	28.1312	37.0389	
7	40	38.9897	38.1248	39.8305	−11.2283	89.1712		34.4531	42.9335	28.5530	48.5228		29.5306	46.1910	32.3302	42.1534	
8	43	43.4751	42.6551	44.2586	−16.0871	102.9909		39.0209	47.0275	31.1269	54.6725		33.7932	49.7130	35.5501	46.9345	
9	44	47.0529	46.3328	47.7287	−26.9959	121.0506		42.9928	50.0165	31.8958	60.9364		37.8005	52.0717	37.6960	51.4158	
10	50	49.7548	49.1597	50.3035	−45.7960	145.2548		46.2706	52.0870	30.5652	67.6778		41.4030	53.5781	38.7058	55.7527	
11	51	51.7108	51.2415	52.1365	−75.2051	178.5806		48.8599	53.4684	26.7523	75.5067		44.5128	54.5100	38.4980	60.1874	
12	55	53.0833	52.7260	53.4024	−119.2978	225.4248		50.8345	54.3663	19.8685	85.2928		47.1038	55.0746	36.9307	65.0401	

<표 Ⅳ-10> DPF 모형 모수 c의 δ에 따른 순차적 확률비 검정 결과(DS 2)

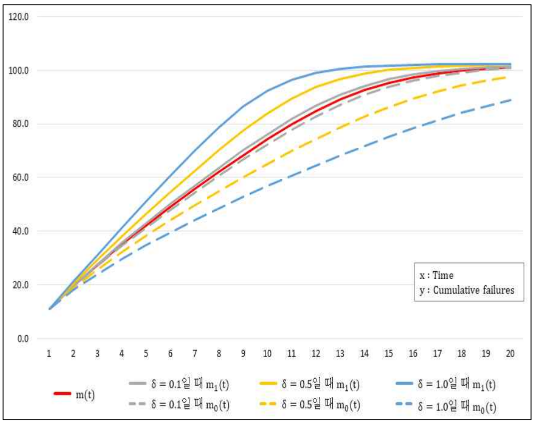
T	$N(t)$	$m(t)$	$\delta=0.4$					$\delta=0.45$					$\delta=0.5$				
			$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result
1	10	9.1317	20.2500	7.9025	15.4571	10.7870	Accpet	30.1353	7.8232	18.1739	14.9153	Accpet	50.6511	7.7520	24.0256	21.6844	Accpet
2	12	12.6825	39.8985	9.6451	-	-	-	51.0776	9.4587	-	-	-	55.8164	9.2926	-	-	-
3	16	17.1280	51.2415	11.6871	-	-	-	55.3226	11.3636	-	-	-	55.8920	11.0766	-	-	-
4	22	22.3485	54.7736	14.0402	-	-	-	55.8306	13.5493	-	-	-	55.8930	13.1153	-	-	-
5	28	28.0333	55.6385	16.6994	-	-	-	55.8863	16.0148	-	-	-	55.8930	15.4099	-	-	-
6	36	33.7329	55.8362	19.6382	-	-	-	55.8923	18.7417	-	-	-	55.8930	17.9482	-	-	-
7	40	38.9897	55.8804	22.8064	-	-	-	55.8929	21.6925	-	-	-	55.8930	20.7022	-	-	-
8	43	43.4751	55.8902	26.1308	-	-	-	55.8930	24.8105	-	-	-	55.8930	23.6276	-	-	-
9	44	47.0529	55.8924	29.5211	-	-	-	55.8930	28.0224	-	-	-	55.8930	26.6656	-	-	-
10	50	49.7548	55.8929	32.8787	-	-	-	55.8930	31.2450	-	-	-	55.8930	29.7463	-	-	-
11	51	51.7108	55.8930	36.1079	-	-	-	55.8930	34.3933	-	-	-	55.8930	32.7958	-	-	-
12	55	53.0833	55.8930	39.1267	-	-	-	55.8930	37.3892	-	-	-	55.8930	35.7420	-	-	-

<표 Ⅳ-11> DPF 모형 모수 h의 δ에 따른 순차적 확률비 검정 결과(DS 2)

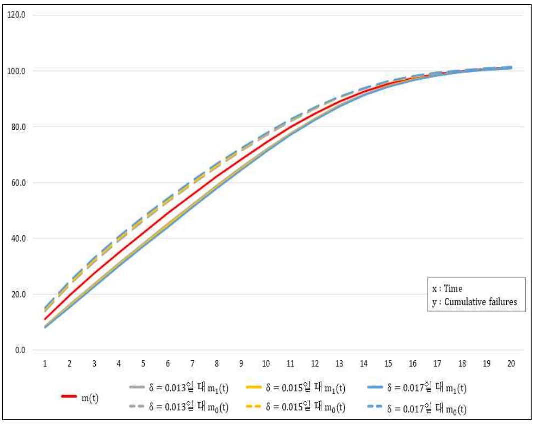
T	$N(t)$	$m(t)$	$\delta=0.1$					$\delta=0.5$					$\delta=1$				
			$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result	$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result
1	10	9.1317	9.0264	9.2365	−86.3805	104.6426		8.6006	9.6510	−9.9522	28.1836		8.0573	10.1589	−0.4124	18.5475	
2	12	12.6825	12.5473	12.8168	−90.6780	116.0411		11.9978	13.3461	−7.9719	33.2919		11.2911	13.9896	2.3390	22.8455	
3	16	17.1280	16.9640	17.2906	−98.0851	132.3387		16.2939	17.9277	−5.8957	40.0914		15.4231	18.6951	5.5864	28.4269	
4	22	22.3485	22.1631	22.5319	−110.7939	155.4880		21.4005	23.2458	−4.2549	48.8758		20.3974	24.0964	9.0112	35.3798	
5	28	28.0333	27.8399	28.2241	−132.2749	188.3380		27.0385	28.9618	−3.9871	59.9654		25.9701	29.8304	12.0006	43.7107	
6	36	33.7329	33.5475	33.9152	−167.8033	235.2653	Continue	32.7738	34.6156	−6.5006	73.8732	Continue	31.7279	35.4306	13.6394	53.4510	Continue
7	40	38.9897	38.8260	39.1503	−225.2189	303.1947		38.1384	39.7631	−13.7217	91.6119		37.1967	40.4689	12.7493	64.8703	
8	43	43.4751	43.3409	43.6065	−316.1624	403.1096		42.7737	44.1054	−28.2319	115.1041		41.9883	44.6747	7.8887	78.7465	
9	44	47.0529	46.9494	47.1541	−458.1323	552.2356		46.5100	47.5365	−53.6257	147.6684		45.8959	47.9699	−2.7892	96.6398	
10	50	49.7548	49.6787	49.8290	−677.7885	777.2961		49.3547	50.1087	−95.1874	194.6489		48.8988	50.4240	−21.8791	121.1942	
11	51	51.7108	51.6569	51.7633	−1016.1043	1119.5245		51.4268	51.9609	−160.9713	264.3581		51.1014	52.1827	−53.2887	156.5690	
12	55	53.0833	53.0461	53.1195	−1536.3194	1642.4850		52.8870	53.2555	−263.4385	369.5806		52.6613	53.4078	−103.0791	209.1465	



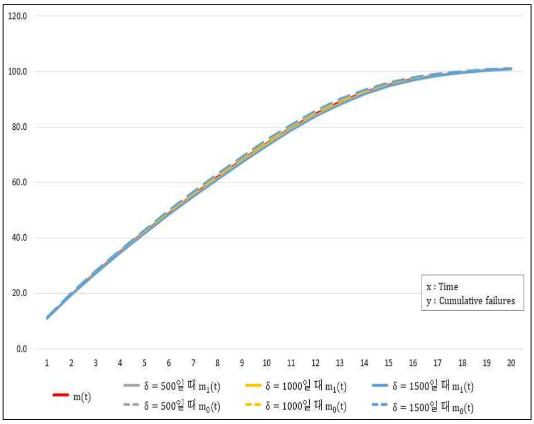
<그림 Ⅳ-2> STX 모형 모수 a의 δ에 따른 $m_0(t)$와 $m_1(t)$



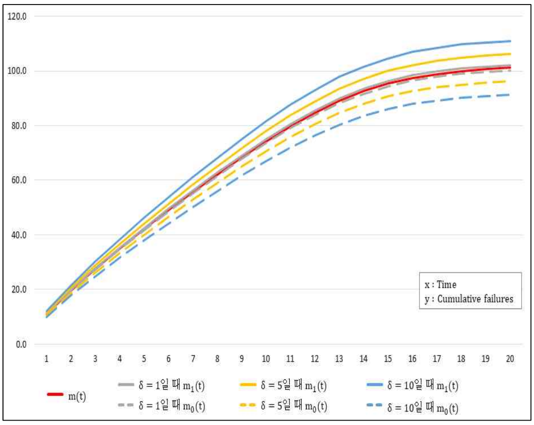
<그림 Ⅳ-3> STX 모형 모수 b의 δ에 따른 $m_0(t)$와 $m_1(t)$



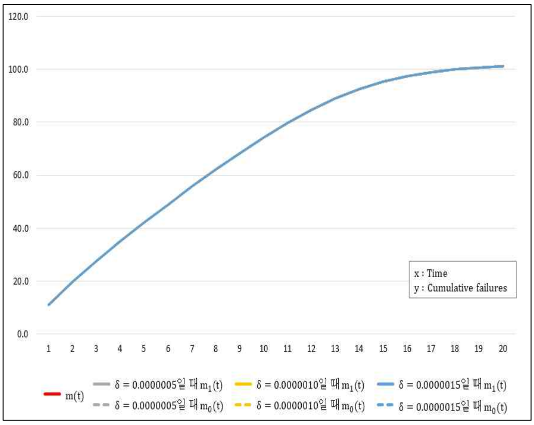
<그림 Ⅳ-4> STX 모형 모수 α의 δ에 따른 $m_0(t)$와 $m_1(t)$



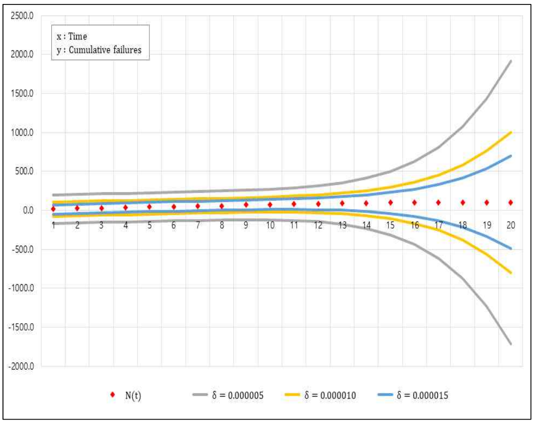
<그림 Ⅳ-5> STX 모형 모수 β의 δ에 따른 $m_0(t)$와 $m_1(t)$

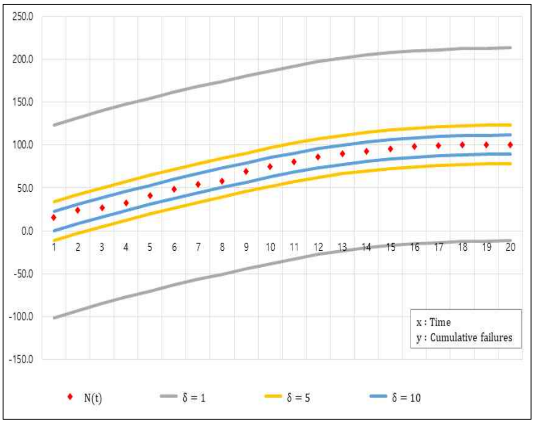


<그림 Ⅳ-6> STX 모형 모수 N의 δ에 따른 $m_0(t)$와 $m_1(t)$

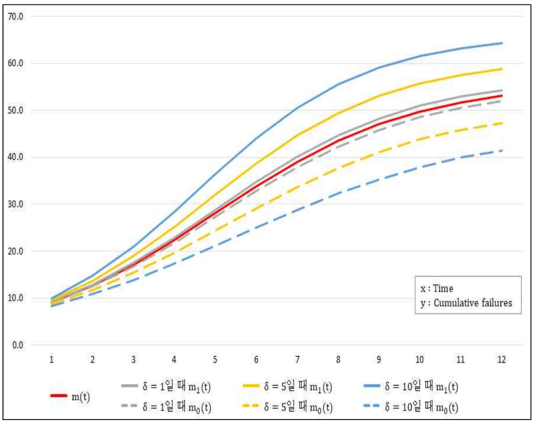


<그림 Ⅳ-7> STX 모형 모수 t_0의 δ에 따른 $m_0(t)$와 $m_1(t)$

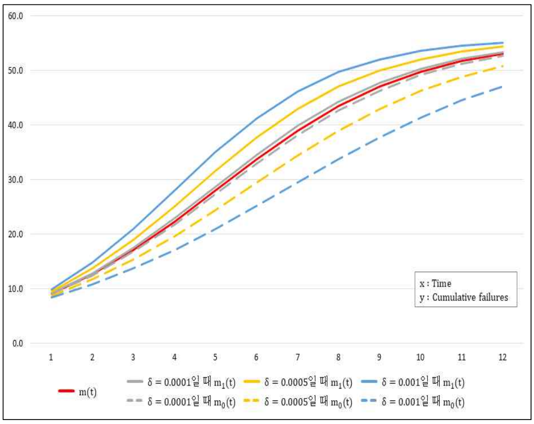




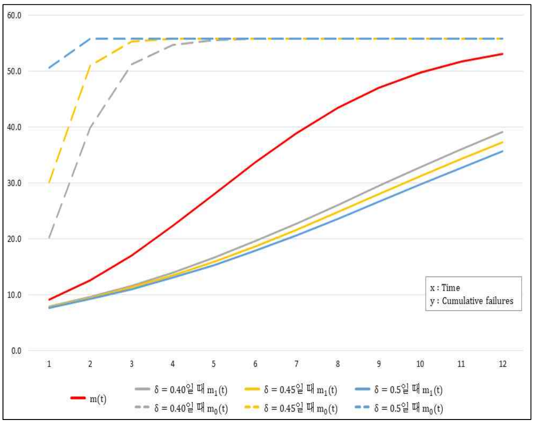
<그림 Ⅳ-8> STX 모형 모수 a의 δ에 따른 채택역과 기각역 <그림 Ⅳ-9> STX 모형 모수 N의 δ에 따른 채택역과 기각역



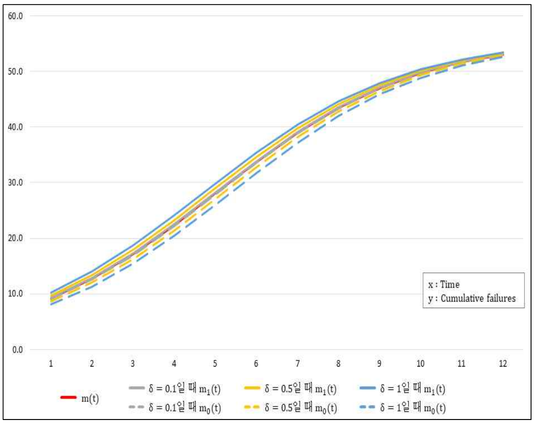
<그림 Ⅳ-10> DPF 모형 모수 a의 δ에 따른 $m_0(t)$와 $m_1(t)$



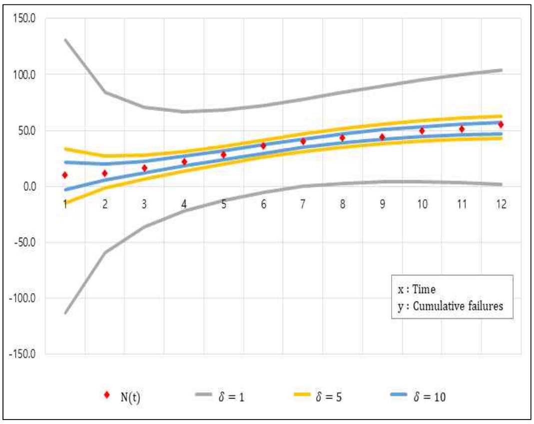
<그림 Ⅳ-11> DPF 모형 모수 b의 δ에 따른 $m_0(t)$와 $m_1(t)$



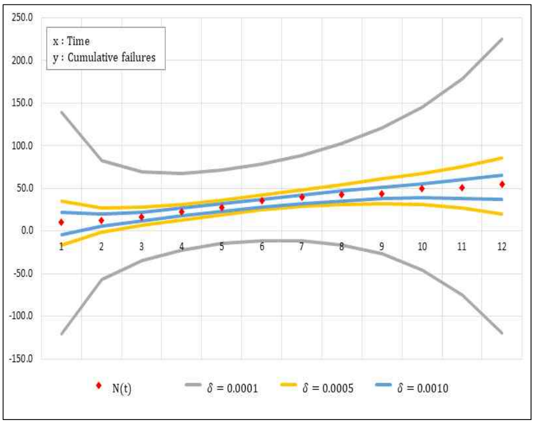
<그림 Ⅳ-12> DPF 모형 모수 c의 δ에 따른 $m_0(t)$와 $m_1(t)$



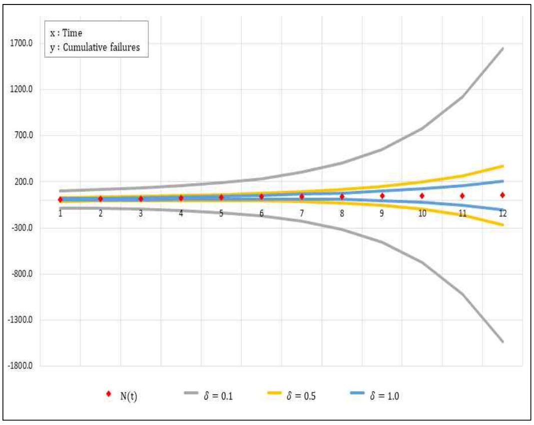
<그림 Ⅳ-13> DPF 모형 모수 h의 δ에 따른 $m_0(t)$와 $m_1(t)$



<그림 Ⅳ-14> DPF 모형 모수 a의 δ에 따른 채택역과 기각역



<그림 Ⅳ-15> DPF 모형 모수 b의 δ에 따른 채택역과 기각역



<그림 Ⅳ-16> DPF 모형 모수 h의 δ에 따른 채택역과 기각역

4.2.2.2. 순차적 확률비 검정 및 신뢰도 분석 결과

STX 모형과 DPF 모형의 모수에 대하여 개별적으로 순차적 확률비 검정을 시행한 결과, STX 모형에서는 모수 α, DPF 모형에서는 모수 c가 가장 민감하게 작용하는 것으로 나타났다(4.2.2.1. 참고). 따라서 STX 모형에서는 모수 α, DPF 모형에서는 모수 c를 제외한 나머지 모수에 대해 순차적 확률비 검정을 적용하여 소프트웨어 신뢰성을 최종적으로 판단한다. 각 모형에 δ를 기준으로 등간척도를 적용하면 $m_0(t)$와 $m_1(t)$는 다음과 같이 표현할 수 있다.

$$m_0(t)=N_0\left(1-\frac{\beta_0}{\beta_0+a_0(t-t_{00})^{b_0}}\right)^{\alpha},\ m_1(t)=N_1\left(1-\frac{\beta_1}{\beta_1+a_1(t-t_{01})^{b_1}}\right)^{\alpha}, \qquad (26)$$

$$m_0(t)=\frac{a_0}{1+\left(\frac{a_0}{h_0}\right)\left(\frac{b_0+c}{c+b_0e^{b_0t}}\right)^{\frac{a_0}{b_0}}},\ m_1(t)=\frac{a_1}{1+\left(\frac{a_1}{h_1}\right)\left(\frac{b_1+c}{c+b_1e^{b_1t}}\right)^{\frac{a_1}{b_1}}}. \qquad (27)$$

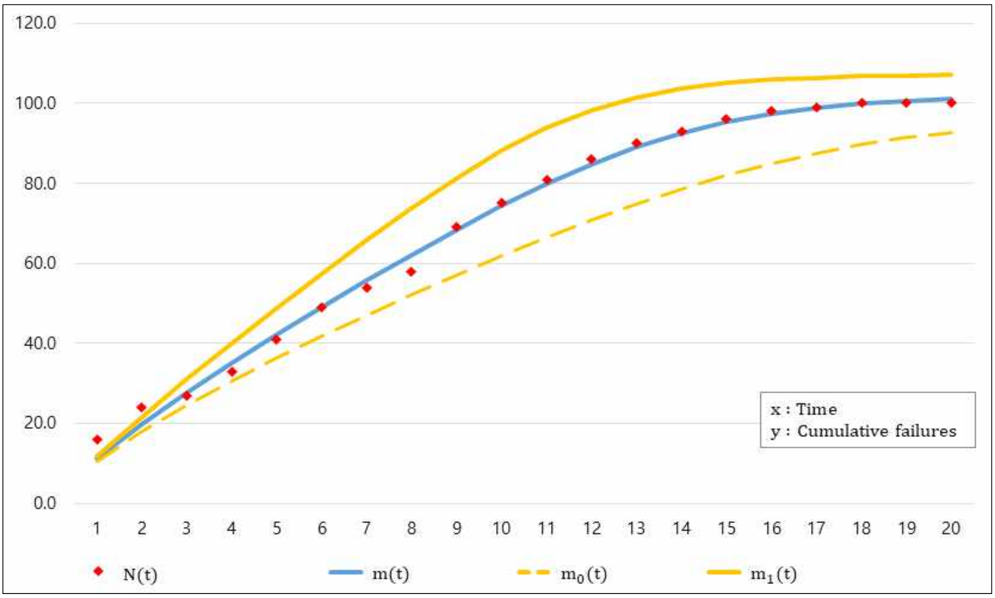
여기서 수식 (26)은 STX 모형의 모수에 등간척도를 적용한 평균값함수이며 수식 (27)은 DPF 모형의 모수에 등간척도를 적용한 평균값함수이다. 각 모수에 대한 δ와 위험확률인 α, β의 가정은 <표 IV-1>의 Case 2를 기준으로 한다. 각 모형의 $m_0(t)$와 $m_1(t)$를 수식 (25)에 대입하면 순차적 확률비 검정의 임계값을 얻을 수 있다.

<표 IV-12>는 STX 모형에서 α를 제외한 다섯 개의 모수를 기준으로 순차적 확률비 검정을 시행한 결과를 나타낸다. 그 결과, $t=1,\cdots,20$인 모든 t에 대하여 “Continue”라는 결론을 도출하였다. <그림 IV-17>은 순차적 확률비 검정 결과인 $m_0(t)$, $m(t)$, $m_1(t)$를 나타내며 <그림 IV-18>은 채택역과 기각역을 나타낸다.

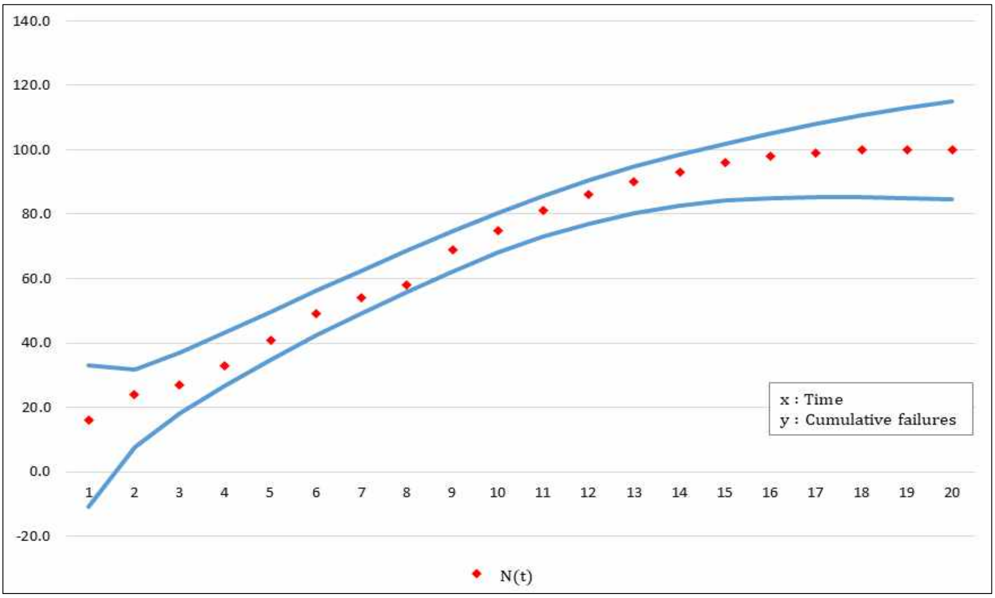
<표 IV-13>은 DPF 모형에서 c를 제외한 세 개의 모수를 기준으로 순차적 확률비 검정을 시행한 결과를 나타낸다. 그 결과, $t=1,\cdots,12$인 모든 t에 대하여 “Continue”라는 결론을 도출하였다. <그림 IV-19>는 순차적 확률비 검정 결과인 $m_0(t)$, $m(t)$, $m_1(t)$를 나타내며 <그림 IV-20>은 채택역과 기각역을 나타낸다.

<표 Ⅳ-12> STX 모형의 모수 a, b, β, N, t_0에 대한 순차적 확률비 검정 결과

T	$N(t)$	$m(t)$	$\alpha=0.1$, $\beta=0.1$				
			$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result
1	16	11.0907	10.5354	11.6448	-10.8639	33.0256	Continue
2	24	19.7148	17.9647	21.5788	7.7298	31.7033	
3	27	27.6015	24.5469	30.9552	18.1548	37.0998	
4	33	35.0443	30.6328	39.9860	26.8559	43.3479	
5	41	42.1715	36.3738	48.7631	34.7704	49.7621	
6	49	49.0516	41.8524	57.3259	42.2000	56.1683	
7	54	55.7185	47.1180	65.6657	49.2600	62.4995	
8	58	62.1779	52.2001	73.7072	55.9681	68.7050	
9	69	68.4031	57.1125	81.2789	62.2597	74.7134	
10	75	74.3286	61.8554	88.1032	67.9953	80.4192	
11	81	79.8466	66.4141	93.8589	72.9945	85.6995	
12	86	84.8180	70.7591	98.3283	77.1112	90.4670	
13	90	89.1055	74.8464	101.5220	80.2997	94.7154	
14	93	92.6209	78.6221	103.6569	82.6151	98.5120	
15	96	95.3604	82.0304	105.0254	84.1632	101.9464	
16	98	97.4029	85.0254	105.8861	85.0606	105.0886	
17	99	98.8761	87.5823	106.4264	85.4234	107.9735	
18	100	99.9166	89.7040	106.7682	85.3713	110.6058	
19	100	100.6439	91.4199	106.9874	85.0267	112.9727	
20	100	101.1512	92.7785	107.1302	84.5058	115.0589	



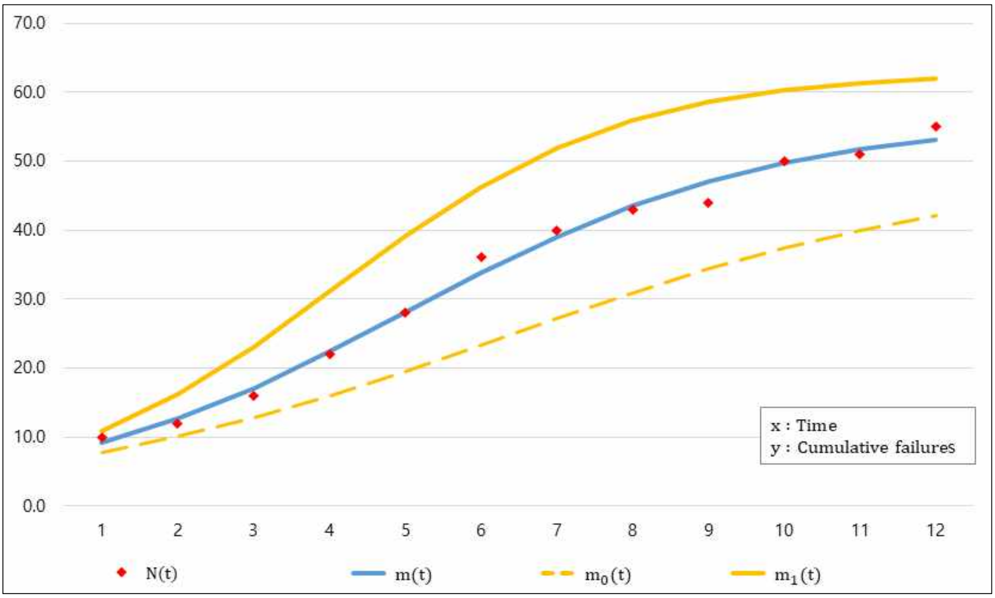
<그림 Ⅳ-17> STX 모형의 순차적 확률비 검정에 대한 $m_0(t)$, $m(t)$, $m_1(t)$



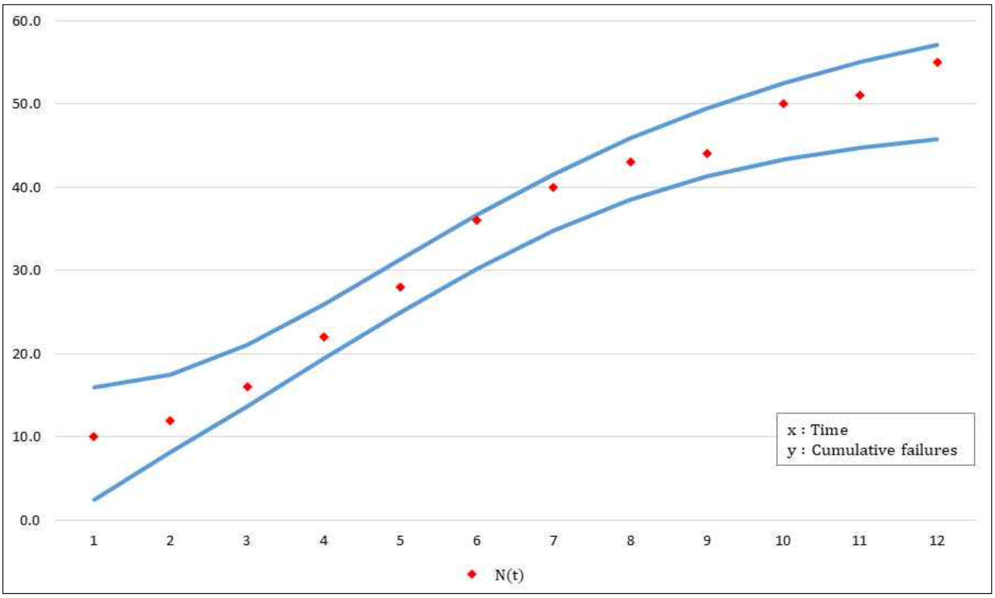
<그림 Ⅳ-18> STX 모형에 대한 순차적 확률비 검정의 채택역과 기각역

<표 Ⅳ-13> DPF 모형의 모수 a, b, h에 대한 순차적 확률비 검정 결과

T	$N(t)$	$m(t)$	$\alpha=0.1,\ \beta=0.1$				
			$m_0(t)$	$m_1(t)$	Accept region	Reject region	Result
1	10	9.1317	7.773831	10.76632	2.442003	15.93601	Continue
2	12	12.6825	10.0353	16.16419	8.247857	17.46654	
3	16	17.1280	12.76089	23.10431	13.72258	21.12521	
4	22	22.3485	15.93511	31.08098	19.38214	25.95995	
5	28	28.0333	19.48619	39.13939	25.02924	31.33022	
6	36	33.7329	23.28327	46.27818	30.27604	36.67322	
7	40	38.9897	27.15076	51.89676	34.80548	41.58859	
8	43	43.4751	30.89846	55.91698	38.47399	45.8825	
9	44	47.0529	34.35762	58.59879	41.28928	49.52028	
10	50	49.7548	37.40895	60.30339	43.34692	52.55037	
11	51	51.7108	39.99393	61.35276	44.77927	55.04877	
12	55	53.0833	42.10922	61.98547	45.72557	57.0915	



<그림 Ⅳ-19> DPF 모형의 순차적 확률비 검정에 대한 $m_0(t)$, $m(t)$, $m_1(t)$



<그림 Ⅳ-20> DPF 모형에 대한 순차적 확률비 검정의 채택역과 기각역

STX 모형과 DPF 모형에 적용한 모수에 대하여 δ의 수준에 따른 순차적 확률비 검정 결과를 살펴보고자 한다. <표 Ⅲ-7>과 <표 Ⅲ-8>의 모형별 모수 추정치를 기준으로 δ는 모수 추정치의 최저 1%에서 최대 20%의 범위로 설정한다. 각 모수에 대해 총 열 개의 수준으로 δ를 구성하며 이는 <표 Ⅳ-14>에서 나타낸다.

<표 Ⅳ-14> 모수별 δ의 가정

Model	Parameter	δ
STX	a	$\delta=\begin{Bmatrix}0.000001, 0.000003, 0.000005, 0.000007, 0.000009,\\ 0.000011, 0.000013, 0.000015, 0.000017, 0.000019\end{Bmatrix}$
	b	$\delta=\{0.05, 0.20, 0.35, 0.50, 0.65, 0.80, 0.95, 1.10, 1.25, 1.40\}$
	β	$\delta=\{100, 300, 500, 700, 900, 1100, 1300, 1500, 1700, 1900\}$
	N	$\delta=\{1, 3, 5, 7, 9, 11, 13, 15, 17, 19\}$
	t_0	$\delta=\begin{Bmatrix}0.0000001, 0.0000003, 0.0000005, 0.0000007, 0.0000009,\\ 0.0000011, 0.0000013, 0.0000015, 0.0000017, 0.0000019\end{Bmatrix}$
DPF	a	$\delta=\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
	b	$\delta=\begin{Bmatrix}0.001, 0.002, 0.003, 0.004, 0.005,\\ 0.006, 0.007, 0.008, 0.009, 0.010\end{Bmatrix}$
	h	$\delta=\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$

<표 Ⅳ-15>부터 <표 Ⅳ-18>은 STX 모형과 DPF 모형의 δ 수준에 따른 열 개의 Case에 대하여 순차적 확률비 검정의 채택역, 기각역 및 결과를 나타내며 <그림 Ⅳ-21>과 <그림 Ⅳ-22>는 이를 시각화하고 있다.

STX 모형에 의한 데이터 세트 1의 순차적 확률비 검정 결과인 <표 Ⅳ-15>와 <표 Ⅳ-16>을 살펴보면 Case 1부터 Case 8까지는 모든 t에 대하여 "Continue"라는 결론을 도출하였다. 반면에 Case 9와 Case 10은 $t=2$일 때 신뢰성을 기각함을 의미하는 "Reject" 결론을 도출하였다. 따라서 데이터 수집 초기에 시스템에 대한 신뢰성을 판단하였기 때문에 순차적 확률비 검정이 효율적임을 입증하였다. <그림 Ⅳ-21>을 보면 δ의 값이 커지면서 "Continue"의 영역은 줄어들고, 채택역과 기각역의 영역이 늘어나면서 시스템을 채택 혹은 기각할 확률이 증가하게 된다. 특히 Case 3부터 "Continue" 영역은 완만하게 감소하였으며 Case 9부터는 근소하게 감소하였다.

DPF 모형에 의한 데이터 세트 2의 순차적 확률비 검정 결과인 <표 Ⅳ-17>과 <표 Ⅳ-18>을 살펴보면 Case 1부터 Case 6까지는 모든 t에 대하여 "Continue"라는 결론을 도출하였다. 반면에 Case 7부터 Case 10은 $t=6$일 때 신뢰성을 기각함을 의미하는 "Reject" 결론을 도출하였다. 따라서 데이터 수집 초기에 시스템에 대한 신뢰성을 판단하였기 때문에 순차적 확률비 검정이 효율적임을 입증하였다. <그림 Ⅳ-22>를 보면 δ의 값이 커지면서 "Continue"의 영역은 줄어들고, 채택역과 기각역의 영역이 늘어나면서 시스템을 채택 혹은 기각할 확률이 증가하게 된다. 특히 Case 3부터 "Continue" 영역은 완만하게 감소하였으며 Case 7부터는 근소하게 감소하였다.

<표 Ⅳ-15> STX 모형의 Case 1-5에 대한 순차적 확률비 검정 결과 채택역 및 기각역(DS 1)

T	$N(t)$	Case 1		Case 2		Case 3		Case 4		Case 5	
		Accept region	Reject region	Accept region	Reject region	Accept region	Reject region	Accept region	Reject region	Accept region	Reject region
1	16	−100.0796	122.2603	−25.9596	48.1345	−11.1350	33.2984	−4.7837	26.9297	−4.7837	26.9297
2	24	−58.5274	97.9563	−4.0200	43.4463	5.7270	33.6949	9.7992	29.6166	9.7992	29.6166
3	27	−39.0865	94.2894	7.9931	47.2146	16.1170	39.1027	19.4936	35.7452	19.4936	35.7452
4	33	−25.3220	95.4110	17.5919	52.5120	24.8653	45.2729	27.8895	42.3022	27.8895	42.3022
5	41	−14.0717	98.4160	26.0904	58.2808	32.8261	51.6053	35.6358	48.8892	35.6358	48.8892
6	49	−4.2537	102.3589	33.9318	64.2127	40.2937	57.9379	42.9595	55.4068	42.9595	55.4068
7	54	4.5836	106.8561	41.2996	70.1894	47.3882	64.2092	49.9493	61.8144	49.9493	61.8144
8	58	12.6081	111.7505	48.2551	76.1543	54.1396	70.3796	56.6119	68.0714	56.6119	68.0714
9	69	19.8046	117.0033	54.7726	82.0670	60.5047	76.3996	62.8798	74.1091	62.8798	74.1091
10	75	26.0040	122.6511	60.7466	87.8834	66.3699	82.1943	68.6149	79.8189	68.6149	79.8189
11	81	30.9028	128.7807	65.9998	93.5467	71.5670	87.6651	73.6415	85.0722	73.6415	85.0722
12	86	34.1147	135.5002	70.3159	98.9941	75.9204	92.7165	77.8202	89.7730	77.8202	89.7730
13	90	35.2839	142.8925	73.5032	104.1754	79.3141	97.2952	81.1138	93.9062	81.1138	93.9062
14	93	34.2475	150.9482	75.4686	109.0661	81.7336	101.4054	83.5871	97.5299	83.5871	97.5299
15	96	31.1747	159.4936	76.2627	113.6568	83.2590	105.0869	85.3559	100.7258	85.3559	100.7258
16	98	26.5830	168.1700	76.0757	117.9253	84.0330	108.3783	86.5436	103.5586	86.5436	103.5586
17	99	21.1943	176.5090	75.1898	121.8214	84.2310	111.2979	87.2670	106.0644	87.2670	106.0644
18	100	15.7142	184.0763	73.9091	125.2804	84.0353	113.8472	87.6364	108.2583	87.6364	108.2583
19	100	10.6597	190.5920	72.4955	128.2528	83.6120	116.0249	87.7546	110.1477	87.7546	110.1477
20	100	6.3064	195.9660	71.1320	130.7265	83.0923	117.8412	87.7118	111.7436	87.7118	111.7436
Result		Continue		Continue		Continue		Continue		Continue	

<표 Ⅳ-16> STX 모형의 Case 6-10에 대한 순차적 확률비 검정 결과 채택역 및 기각역(DS 1)

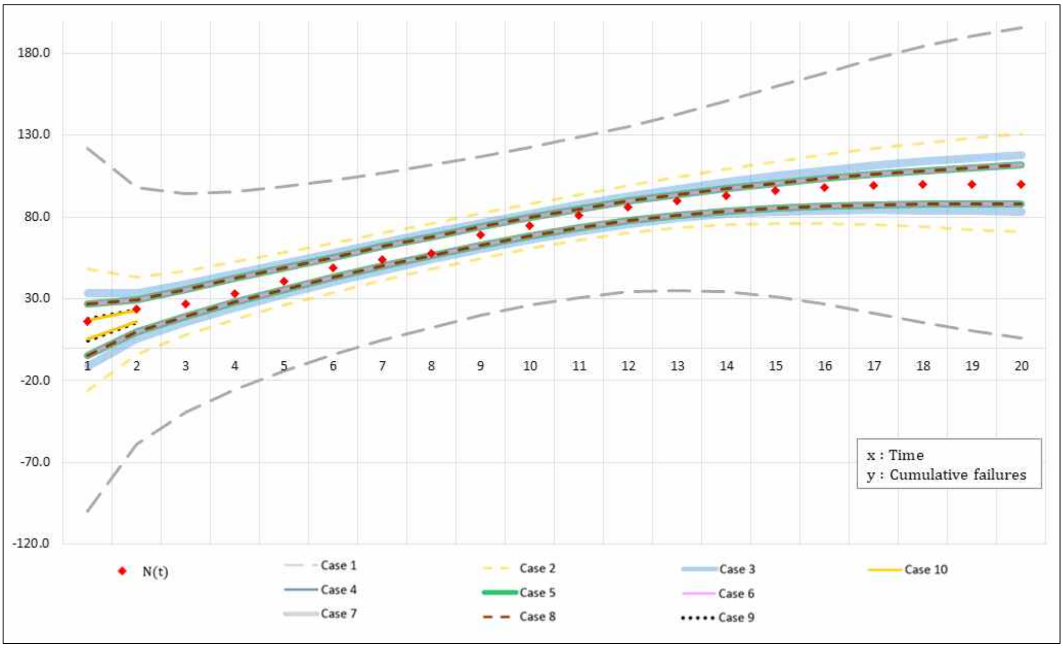
T	$N(t)$	Case 6		Case 7		Case 8		Case 9		Case 10	
		Accept region	Reject region	Accept region	Reject region	Accept region	Reject region	Accept region	Reject region	Accept region	Reject region
1	16	−4.7837	26.9297	−4.7837	26.9297	−4.7837	26.9297	4.5079	17.4635	5.1773	16.7413
2	24	9.7992	29.6166	9.7992	29.6166	9.7992	29.6166	15.6623	23.6940	16.0841	23.2539
3	27	19.4936	35.7452	19.4936	35.7452	19.4936	35.7452				
4	33	27.8895	42.3022	27.8895	42.3022	27.8895	42.3022				
5	41	35.6358	48.8892	35.6358	48.8892	35.6358	48.8892				
6	49	42.9595	55.4068	42.9595	55.4068	42.9595	55.4068				
7	54	49.9493	61.8144	49.9493	61.8144	49.9493	61.8144				
8	58	56.6119	68.0714	56.6119	68.0714	56.6119	68.0714				
9	69	62.8798	74.1091	62.8798	74.1091	62.8798	74.1091				
10	75	68.6149	79.8189	68.6149	79.8189	68.6149	79.8189				
11	81	73.6415	85.0722	73.6415	85.0722	73.6415	85.0722				
12	86	77.8202	89.7730	77.8202	89.7730	77.8202	89.7730				
13	90	81.1138	93.9062	81.1138	93.9062	81.1138	93.9062				
14	93	83.5871	97.5299	83.5871	97.5299	83.5871	97.5299				
15	96	85.3559	100.7258	85.3559	100.7258	85.3559	100.7258				
16	98	86.5436	103.5586	86.5436	103.5586	86.5436	103.5586				
17	99	87.2670	106.0644	87.2670	106.0644	87.2670	106.0644				
18	100	87.6364	108.2583	87.6364	108.2583	87.6364	108.2583				
19	100	87.7546	110.1477	87.7546	110.1477	87.7546	110.1477				
20	100	87.7118	111.7436	87.7118	111.7436	87.7118	111.7436				
Result		Continue		Continue		Continue		Reject		Reject	

<표 Ⅳ-17> DPF 모형의 Case 1-5에 대한 순차적 확률비 검정 결과 채택역 및 기각역(DS 2)

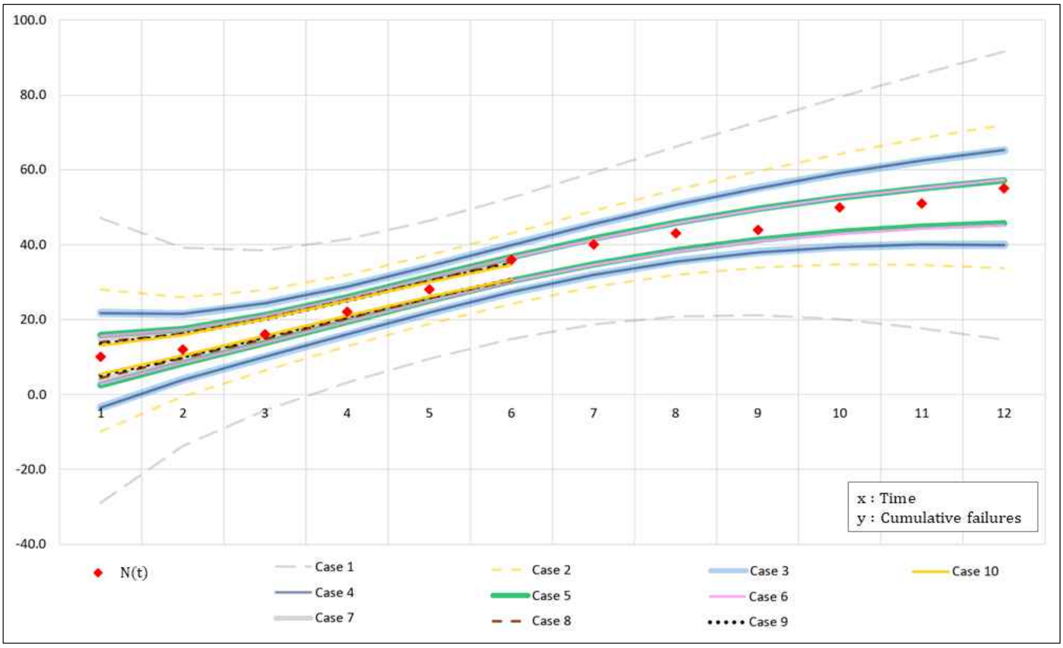
T	$N(t)$	Case 1		Case 2		Case 3		Case 4		Case 5	
		Accept region	Reject region	Accept region	Reject region	Accept region	Reject region	Accept region	Reject region	Accept region	Reject region
1	10	-28.8245	47.0911	-9.8399	28.1158	-3.5049	21.7966	-3.5049	21.7966	2.4420	15.9360
2	12	-13.7208	39.0952	-0.5041	25.9071	3.9198	21.5308	3.9198	21.5308	8.2479	17.4665
3	16	-4.2612	38.5328	6.4565	27.8618	10.0583	24.3378	10.0583	24.3378	13.7226	21.1252
4	22	3.2311	41.4814	12.8101	31.9488	16.0306	28.8049	16.0306	28.8049	19.3821	25.9600
5	28	9.6020	46.4673	18.8134	37.2637	21.8848	34.2046	21.8848	34.2046	25.0292	31.3302
6	36	14.8324	52.6087	24.2308	43.1368	27.3110	39.9348	27.3110	39.9348	30.2760	36.6732
7	40	18.6393	59.2803	28.7053	49.0361	31.9413	45.5066	31.9413	45.5066	34.8055	41.5886
8	43	20.7803	66.0780	31.9721	54.6123	35.5252	50.6094	35.5252	50.6094	38.4740	45.8825
9	44	21.2011	72.7922	33.9514	59.7063	37.9903	55.1159	37.9903	55.1159	41.2893	49.5203
10	50	20.0642	79.3260	34.7439	64.2902	39.4208	59.0263	39.4208	59.0263	43.3469	52.5504
11	51	17.7079	85.5990	34.5764	68.3874	40.0009	62.3951	40.0009	62.3951	44.7793	55.0488
12	55	14.5720	91.4914	33.7383	72.0158	39.9600	65.2797	39.9600	65.2797	45.7256	57.0915
Result		Continue		Continue		Continue		Continue		Continue	

<표 Ⅳ-18> DPF 모형의 Case 6-10에 대한 순차적 확률비 검정 결과 채택역 및 기각역(DS 2)

T	$N(t)$	Case 6		Case 7		Case 8		Case 9		Case 10	
		Accept region	Reject region	Accept region	Reject region	Accept region	Reject region	Accept region	Reject region	Accept region	Reject region
1	10	2.8662	15.5105	3.7912	14.6265	4.4934	13.9715	5.0481	13.4704	5.5005	13.0778
2	12	8.4462	17.2613	9.1356	16.6954	9.6773	16.2961	10.1235	16.0110	10.5058	15.8086
3	16	13.8236	20.9882	14.4290	20.5804	14.9213	20.3142	15.3424	20.1466	15.7173	20.0517
4	22	19.4043	25.8321	19.9478	25.4741	20.3889	25.2412	20.7642	25.0943	21.0953	25.0093
5	28	24.9647	31.1772	25.4065	30.7529	25.7371	30.4366	25.9922	30.1909	26.1927	29.9927
6	36	30.1189	36.4832	30.4123	35.8886	30.5737	35.3866	30.6421	34.9411	30.6416	34.5313
7	40	34.5622	41.3754								
8	43	38.1597	45.6796								
9	44	40.9172	49.3703								
10	50	42.9210	52.4955								
11	51	44.2924	55.1253								
12	55	45.1636	57.3265								
Result		Continue		Reject		Reject		Reject		Reject	



<그림 Ⅳ-21> Case에 따른 STX 모형에 의한 순차적 확률비 검정의 채택역과 기각역



<그림 Ⅳ-22> Case에 따른 DPF 모형에 의한 순차적 확률비 검정의 채택역과 기각역

순차적 확률비 검정의 결과와 실제 데이터 세트 시스템의 신뢰성을 비교하기 위해 데이터 세트의 신뢰도를 추정하고자 한다. NHPP를 기반으로 하는 신뢰도함수(Reliability function)는 소프트웨어 신뢰성 성장 모형의 평균값함수인 $m(t)$를 사용하여 다음과 같이 나타낼 수 있다(Pham 2006a).

$$R(t) = e^{-m(t)}. \tag{28}$$

수식 (28)은 구간 $[0,t]$에서 소프트웨어 오류가 발생하지 않을 확률을 의미한다. 만약 t이후의 시점인 $t+x$가 주어졌을 때의 소프트웨어 신뢰도는 수식 (29)와 같이 조건부 확률 $R(x|t)$로 나타낼 수 있다.

$$R(x|t) = e^{-[m(t+x)-m(t)]}. \tag{29}$$

여기서 $R(x|t)$는 구간 $[t,t+x]$에서 소프트웨어 오류가 발생하지 않을 확률이며 이때 $t \geq 0,\ x > 0$이다.

$R(t)$는 0부터 시작하여 t까지인 구간 $[0,t]$에 대한 신뢰도를 추정하지만 $R(x|t)$는 x를 작게 가정하면 약 t시점에서의 신뢰도를 추정할 수 있다. 순차적 확률비 검정은 매 시점에서 검정을 통해 신뢰성을 판단하는 특성을 고려하여 수식 (29)를 기반으로 데이터 세트의 조건부 신뢰도를 추정한다. 이때 x는 0.1로 가정한다.

<표 Ⅳ-19>와 <그림 Ⅳ-23>은 STX 모형을 기반으로 데이터 세트 1의 신뢰도 $R(x|t)$를 나타낸다. 데이터 세트 1은 전체적으로 조건부 신뢰도가 낮은 것으로 나타났으며 $t=1,\ \cdots,10$일 때 신뢰도가 감소하는 추세를 보였고, $t=10$일 때 신뢰도는 0.1772로 최솟값을 보였다. <표 Ⅲ-4>의 데이터 세트를 살펴보면 $t=1,\ \cdots,10$까지 검출된 고장 수는 그 이후 검출된 고장 수보다 상당히 크게 나타났으며 신뢰도는 이를 뒷받침하고 있다. 특히 $t=1,2$일 때 검출된 고장 수는 각각 16, 8개로 상당히 높게 나타났는데 <표 Ⅳ-16>의 Case 9와 10을 살펴보면 $t=2$일 때 시스템을 기각해야 한다는 결론을 내렸다.

<표 Ⅳ-20>과 <그림 Ⅳ-24>는 DPF 모형을 기반으로 데이터 세트 2의 신뢰도 $R(x|t)$를 나타낸다. 데이터 세트 2의 추정된 신뢰도를 보면 $t=1,\ \cdots,5$일 때 신뢰도가 감소하는 추세를 보였고, $t=5$일 때 신뢰도는 최솟값을 가졌으며 그 이후로는 신뢰도가 점차 증가하였다. <표 Ⅲ-5>의 데이터 세트를 살펴보면 $t=1,\ \cdots,6$까지 검출된 고장 수는 $t=7,\ \cdots,12$에서 검출된 고장 수보다 상당히 크게 나타났으며 신뢰도는 이를 뒷받침하고 있다. 또한 <표 Ⅳ-18>의 Case 7-10을 살펴보면 데이

터 세트의 시스템은 $t=6$일 때 기각해야 한다는 결론을 내렸다.

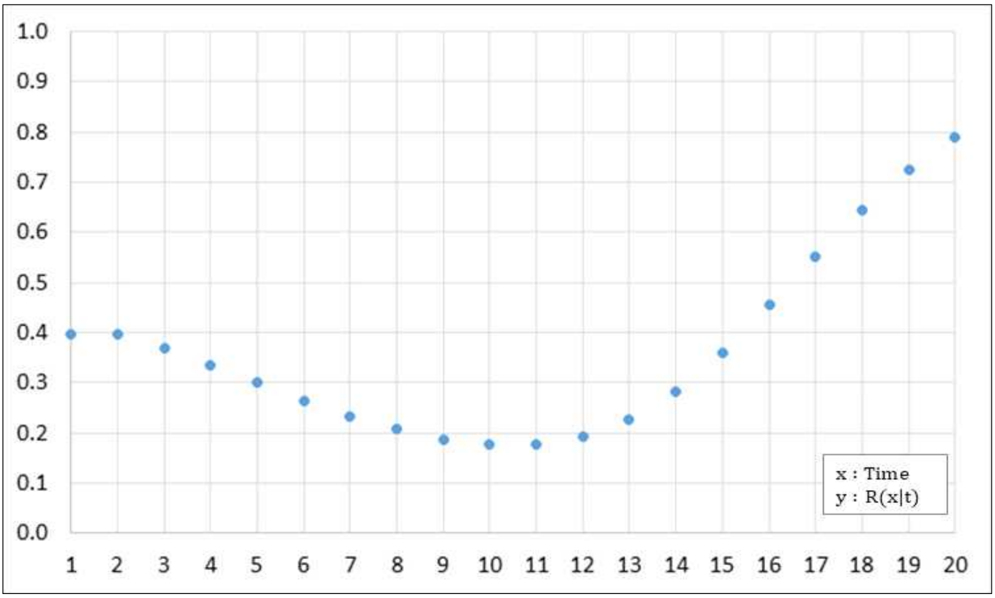
데이터 세트 1과 2에서 δ는 모수 추정치의 약 15% 수준으로 설정할 때의 순차적 확률비 검정 결과는 데이터 세트의 추정된 신뢰도 결과에 부합하였다. 따라서 δ는 모수 추정치의 약 15% 수준으로 가정하는 것이 적절하다.

<표 Ⅳ-19> 데이터 세트 1의 신뢰도

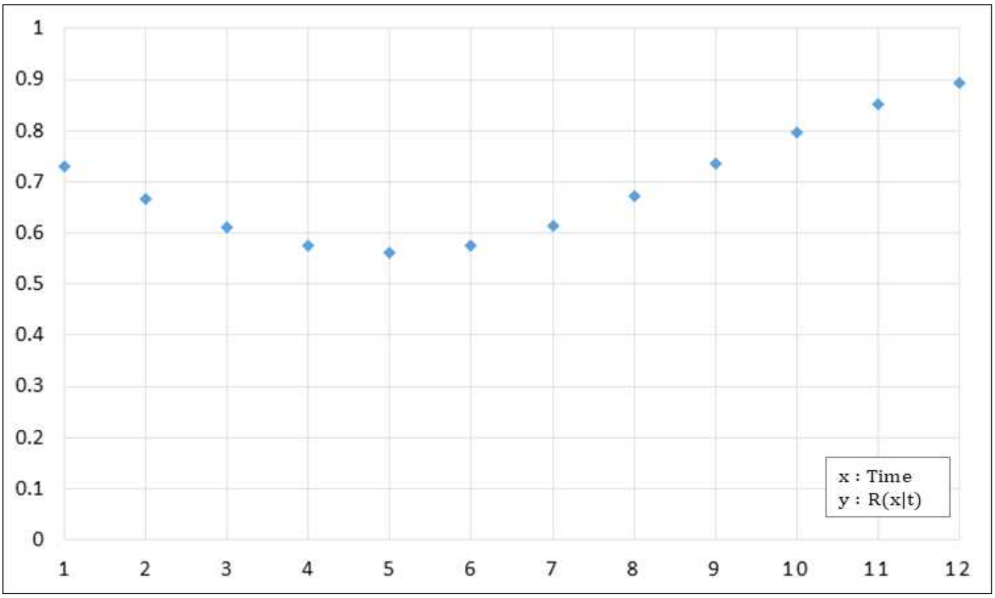
T	$R(x\|t)$	T	$R(x\|t)$
1	0.3981	11	0.1773
2	0.3967	12	0.1923
3	0.3703	13	0.2264
4	0.3360	14	0.2828
5	0.3000	15	0.3610
6	0.2656	16	0.4545
7	0.2344	17	0.5524
8	0.2081	18	0.6443
9	0.1884	19	0.7236
10	0.1772	20	0.7881

<표 Ⅳ-20> 데이터 세트 2의 신뢰도

T	$R(x\|t)$
1	0.7298
2	0.6671
3	0.6121
4	0.5745
5	0.5614
6	0.5756
7	0.6144
8	0.6708
9	0.7349
10	0.7973
11	0.8514
12	0.8947



<그림 Ⅳ-23> 데이터 세트 1의 신뢰도 $R(x|t)$



<그림 Ⅳ-24> 데이터 세트 2의 신뢰도 $R(x|t)$

Ⅴ. 결론 및 제언

본 논문에서는 소프트웨어의 신뢰성을 판단하기 위해 사용되는 새로운 유형의 소프트웨어 신뢰성 성장 모형을 소개하였다. 소프트웨어 테스트 코드에서 발생한 구문 오류로 인해 실질적인 테스트 시간이 지연됨을 가정하여 여섯 개의 모수로 이루어진 평균값함수를 갖는 모형(STX), 소프트웨어를 구성하는 특정 클래스에서 발생한 오류가 이를 참조하는 다른 클래스에 영향을 미침으로써 고장이 종속적으로 발생함을 가정하였으며 네 개의 모수로 이루어진 평균값함수를 갖는 모형(DPF)을 소개하였다. 제안하는 모형의 적합도를 입증하기 위해 실제 데이터를 적합하여 적합도를 비교해본 결과, 데이터 세트 1에서 STX 모형의 모든 척도는 다른 모형에 비하여 가장 우수하게 나타났다. 데이터 세트 2에서 DPF 모형의 AIC 값은 58.6958로 전체 아홉 개의 모형 중 세 번째로 작게 나타났으며 나머지 척도는 다른 모형에 비하여 가장 우수하게 나타났다. 따라서 두 모형은 각 데이터 세트에 대하여 최상의 적합도를 보였다.

다음으로 소프트웨어 신뢰성을 효율적으로 판단하기 위한 기법으로 순차적 확률비 검정을 제안하였다. 먼저 순차적 확률비 검정에 사용될 모형별 최적의 모수를 결정하기 위해 일차적으로 STX 모형과 DPF 모형의 모수에 각각 δ를 세 개의 수준으로 나눠 순차적 확률비 검정을 적용하였다. STX 모형에서는 모수 α, DPF 모형에서는 모수 c의 $m_0(t)$는 다른 모수에 비하여 상이한 형태로 나타났으며 가장 민감한 모수로 밝혀졌다. 따라서 소프트웨어 신뢰성을 판단하는 데 STX 모형의 모수 α, DPF 모형의 모수 c에 관해서는 엄격한 판단이 필요할 것으로 사료되어 최종적인 순차적 확률비 검정에서는 해당 모수를 제외하고 나머지 모수만을 고려하여 검정을 시행하였다. 또한 δ의 영향을 살펴보기 위해 모수 추정값을 기준으로 δ를 최저 1%대에서 최대 20%대의 범위로 가정하고, 열 개의 Case로 분류하여 검정을 시행하였다. 그 결과, STX 모형으로 데이터 세트 1의 순차적 확률비 검정을 시행한 결과, Case 1-8은 데이터 수집 기간 전체 t에 대하여 “Continue”라는 결론을 도출하였으나 Case 9, 10에서는 데이터 수집 도중인 $t=2$일 때 시스템을 기각한다는 “Reject”를 도출하였다. 따라서 순차적 확률비 검정이 소프트웨어의 신뢰성을 판단하는 데 효율적임을 입증하였다. 특히 Case 3부터 “Continue” 영역은 완만하게 감소하였으며 Case 9부터는 근소하게 감소하였다.

DPF 모형으로 데이터 세트 2의 순차적 확률비 검정을 시행한 결과, Case 1-6은 데이터 수집 기간 전체 t에 대하여 “Continue”라는 결론을 도출하였으나 Case 7-10에서는 데이터 수집 도중인 $t=6$일 때 시스템을 기각한다는 “Reject”를 도출하였다. 따라서 순차적 확률비 검정이 소프트웨어의 신뢰성을 판단하는 데 효율적임을 입증하였다. 특히 Case 3부터 “Continue” 영역은 완만하게 감소하였으며 Case 7부터는 아주 근소하게 감소하였다.

즉, δ의 값이 커질수록 채택역과 기각역의 폭은 커지며 “Continue” 영역은 좁아지는데 δ의 값이 커질수록 시스템의 채택 혹은 기각될 확률이 높아지기 때문에 조기에 데이터 수집을 중단할 확률이 높아짐을 알 수 있다. 그러나 “Continue” 영역이 감소하는 데는 한계가 있으며 폭이 점차 좁아지는 것일 뿐 뒤틀리는 수준은 아니므로 “채택에서 기각” 혹은 “기각에서 채택”처럼 결과가 극단적으로 뒤집히는 영향을 보이지는 않았다.

순차적 확률비 검정의 결과와 비교하기 위하여 STX 모형과 DPF 모형으로 각 데이터 세트의 신뢰도를 추정한 결과, 데이터 세트 1은 전반적으로 신뢰도가 낮았으며 $t=1,\cdots,10$에 대해서 신뢰도가 감소하는 추세를 보였다. 데이터 세트 1을 살펴보면 $t=1,2$일 때 고장 수가 상당히 높게 나타났는데 순차적 확률비 검정 결과 Case 9, 10에서 $t=2$일 때 시스템을 기각하였다. 데이터 세트 2는 $t=1,\cdots,5$에 대해서 신뢰도가 감소하는 추세를 보였다. 데이터 세트 2를 살펴보면 $t=1,\cdots,6$일 때 검출된 고장 수가 $t=7,\cdots,12$일 때 발생한 고장 수보다 상당히 높게 나타났는데 순차적 확률비 검정 결과 Case 7-10에서 $t=6$일 때 시스템을 기각하였다. 즉, 추정된 신뢰도는 데이터 세트 1의 Case 9-10 검정 결과, 데이터 세트 2의 Case 7-10 검정 결과를 뒷받침하였다. 따라서 이 결과를 기반으로 δ는 모수 추정값의 약 15%로 설정하는 것이 적절함으로 판단한다.

본 논문에서는 소프트웨어의 신뢰성을 판단하는 데 사용되는 소프트웨어 신뢰성 성장 모형과 검정 기법을 새로이 제안하였다. 포스트 코로나 시대에 다양한 산업에서 소프트웨어의 역할이 커질 것으로 전망되며 본 연구는 소프트웨어 산업에 주요한 역할을 할 것으로 사료한다. 현재까지 다양한 환경을 가정으로 하는 수많은 소프트웨어 신뢰성 성장 모형이 개발되었으며 소프트웨어 신뢰성 성장 모형의 연구는 이제 과도기에 머무르고 있다. 앞으로는 순환신경망, 심층신경망, 장단기 메모리 등의 다양한 딥 러닝 기법을 적용하여 소프트웨어 신뢰성 모형 및 신뢰성 추정의 연구가 확장되길 기대한다.

참 고 문 헌

[1] Akaike, H. A new look at statistical model identification. *IEEE Trans. Autom. Control.* 1974, 19, 716–719.

[2] Bain, L. *Statistical analysis of reliability and life-testing models*; Marcel Dekker, New York & Basel, 1978.

[3] Barlow, R.E.; Proschan, F. *Mathematical theory of reliability*; Society for Industrial and Applied Mathematics, 1996.

[4] Belady, L.A.; Lehman, M.M. A model of large program development. *IBM Syst. J.* 1976, 15, 225–252.

[5] Box, G.E.; Jenkins, G.M.; Reinsel, G.C.; Ljung, G.M. *Time series analysis: forecasting and control*; John Wiley & Sons, 2015.

[6] Cadini, F.; Gioletta, A. A Bayesian Monte Carlo-based algorithm for the estimation of small failure probabilities of systems affected by uncertainties. *Reliab. Eng. Syst. Saf.* 2016, 153, 15–27.

[7] Cai, K.-Y. On estimating the number of defects remaining in software. *J. Syst. Softw.* 1998, 40, 93–114.

[8] Caiuta, R.; Pozo, A.; Vergilio, S.R. Meta-learning based selection of software reliability models. *Autom. Softw. Eng.* 2017, 24, 575–602.

[9] Chang, I.H.; Pham, H.; Lee, S.W.; Song, K.Y. A testing-coverage software reliability model with the uncertainty of operating environments. *Int. J. Syst. Sci. Oper. Logist.* 2014, 1, 220–227.

[10] Cohen, A.C.; Whitten, B.J. *Parameter Estimation in Reliability and Life Span Models*, Marcel Dekker, New York & Basel. 1988.

[11] Coutinho, J.S. Software reliability growth. In Proceedings International Conference on Reliable Software, IEEE Computer Society Press; Los Angeles, 1973.

[12] Drenick, R.F. The Failure Law of Complex Equipment. *J. Soc. Ind. Appl. Math.* 1960, 8, 680–690.

[13] Epstein, B.; Weissman, I. *Mathematical models for systems reliability*; Crc Press, 2008.

[14] Goel, A.L.; Okumoto, K. Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures. *IEEE Trans. Reliab.* 1979a, R-28, 206–211.

[15] Goel, A.L.; Okumoto, K. A Markovian model for reliability and other performance measures of software systems. In Proceedings of the 1979 International Workshop on Managing Requirements Knowledge (MARK); IEEE, 1979b; pp. 769–774.
[16] Gutta, S.; Ravi, S.P. Detection of Reliable Pareto Software Using SPRT. *Int. J. Comput. Sci.* Issues 2014, 11, 130.
[17] Hamada, M.S.; Wilson, A.; Reese, C.S.; Martz, H. *Bayesian reliability*; Springer Science & Business Media, 2008.
[18] Inoue, S.; Ikeda, J.; Yamada, S. Bivariate change-point modeling for software reliability assessment with uncertainty of testing-environment factor. *Ann. Oper. Res.* 2016, 244, 209–220.
[19] ISO 8402 International Standard: Quality Vocabulary. ISO: International Organization for Standardization, Geneva, Switzerland, 1986.
[20] Jelinski, Z.; Moranda, P. SOFTWARE RELIABILITY RESEARCH. In Statistical Computer Performance Evaluation; Elsevier, 1972; pp. 465–484.
[21] Kotha, S.K.; Prasad, R.S. Pareto Type II Software Reliability Growth Model–An Order Statistics Approach. *Int. J. Comput. Sci. Trends Technol.* 2014, 2, 49–54.
[22] Lee, D.H.; Chang, I.H.; Pham, H.; Song, K.Y. A Software Reliability Model Considering the Syntax Error in Uncertainty Environment, Optimal Release Time, and Sensitivity Analysis. *Appl. Sci.* 2018, 8, 1483.
[23] Lee, D.H.; Chang, I.H.; Pham, H. Software Reliability Model with Dependent Failures and SPRT. *Mathematics* 2020, 8, 1366.
[24] Li, Q.; Pham, H. A Generalized Software Reliability Growth Model With Consideration of the Uncertainty of Operating Environments. *IEEE Access* 2019, 7, 84253–84267.
[25] Miller, D.R.; Sofer, A. Completely monotone regression estimation of software failure rate. Proc. International Conference on Software Engineering, IEEE Computer Society Press, Los Angeles, 1985; pp. 343-348.
[26] Mills, H. On the statistical validation of computer programs. IBM FSD Report, 1972.
[27] Minamino, Y.; Inoue, S.; Yamada, S. NHPP-based change-point modeling for software reliability assessment and its application to software development management. Ann. Oper. Res. 2016, 244, 85–101.

[28] Murali Mohan, K. V.; Satya Prasad, R.; Sridevi, G. Detection of Burr Type XII Reliable Software using Sequential Process Ratio Test. *Indian J. Sci. Technol.* 2015, 8.
[29] Musa, J.D.; Iannino, A.; Okumoto, K. *Software Reliability: Measurement, Prediction, and Application*; McGraw-Hill, NewYork, 1987.
[30] Ohba, M. Software reliability analysis models. *IBM J. Res. Dev.* 1984a, 28, 428–443.
[31] Ohba, M.; Yamada, S. S-Shaped Software Reliability Growth Models. In Proceedings of the 4th International Conference on Reliability and Maintainability, Lannion, France, 21–25 May 1984b; pp. 430–436.
[32] Pham, H. A new software reliability model with Vtub-shaped fault-detection rate and the uncertainty of operating environments. *Optimization* 2014a, 63, 1481–1490.
[33] Pham, H. Loglog fault-detection rate and testing coverage software reliability models subject to random environments. *Vietnam J. Comput. Sci.* 2014b, 1, 39–45.
[34] Pham, H.; Deng, C. Predictive-ratio risk criterion for selecting software reliability models, Proc. Ninth International Conf. On Reliability and Quality in Design, August, 2003a; pp. 17-21.
[35] Pham, H.; Normann, L. A generalized NHPP software reliability model. Proc. 3rd ISSAT International Conf. on Reliability and Quality in Design, August, ISSAT Press, Anaheim, 1997a.
[36] Pham, H; Nordmann, L.; Zhang, Z. A general imperfect-software-debugging model with S-shaped fault-detection rate. *IEEE Trans. Reliab.* 1999, 48, 169–175.
[37] Pham, H.; Zhang, X. An NHPP Software Reliability Model and Its Comparison. *Int. J. Reliab. Qual. Saf. Eng.* 1997b, 04, 269–282.
[38] Pham, H.; Zhang, X. NHPP software reliability and cost models with testing coverage. *Eur. J. Oper. Res.* 2003b, 145, 443–454.
[39] Pham, H. *Software reliability*; Springer Science & Business Media, 2000.
[40] Pham, H. *Handbook of Reliability Engineering*; Springer Science & Business Media, 2006a.
[41] Pham, H. *System Software Reliability*; Springer: London, UK, 2006b.
[42] Pillai, K.; Nair, V.S. A model for software development eort and cost estimation. *IEEE Trans. Softw. Eng.* 1997, 23, 485–497.

[43] Prasad, R.S.K.; Rao, K.P.G.; Mohan, G.K. Software Reliability using SPRT: Inflection S-shaped Model. *Int. J. Appl. Innov. Eng. Manag.* **2013**, 2, 349–355.
[44] Rani, P.; Mahapatra, G.S. A Single Change Point Hazard Rate Software Reliability Model with Imperfect Debugging. In Proceedings of the 2019 IEEE International Systems Conference (SysCon); IEEE, **2019**; pp. 1–7.
[45] Schick, G.J.; Wolverton, R.W. An Analysis of Competing Software Reliability Models. *IEEE Trans. Softw. Eng.* **1978**, SE-4, 104–120.
[46] Singpurwalla, N. D.; Wilson, S.P. *Statistical methods in software engineering: reliability and risk*; Springer Science & Business Media, **2012**.
[47] Smitha, C.H.; Prasad, R.S.; Kumar, R.K. Burr Type III Process Model with SPRT for Software Reliability. *Int. J. Innov. Res. Adv. Eng. ISSN.* **2014**, 6, 2349–2763.
[48] Song, K.Y.; Chang, I.H.; Pham, H. A three-parameter fault-detection software reliability model with the uncertainty of operating environments. *J. Syst. Sci. Syst. Eng.* **2017a**, 26, 121–132.
[49] Song, K.Y.; Chang, I.H.; Pham, H. An NHPP Software Reliability Model with S-Shaped Growth Curve Subject to Random Operating Environments and Optimal Release Time. *Appl. Sci.* **2017b**, 7, 1304.
[50] Song, K.Y; Chang, I.H; Pham, H. A Software Reliability Model with a Weibull Fault Detection Rate Function Subject to Operating Environments. *Appl. Sci.* **2017c**, 7, 983.
[51] Song, K.Y.; Chang, I.H.; Pham, H. Optimal Release Time and Sensitivity Analysis Using a New NHPP Software Reliability Model with Probability of Fault Removal Subject to Operating Environments. *Appl. Sci.* **2018**, 8, 714.
[52] Song, K.Y.; Chang, I.H.; Pham, H. A Testing Coverage Model Based on NHPP Software Reliability Considering the Software Operating Environment and the Sensitivity Analysis. *Mathematics* **2019**, 7, 450.
[53] Stieber, H.A. Statistical quality control: how to detect unreliable software components. In Proceedings of the Proceedings The Eighth International Symposium on Software Reliability Engineering; IEEE Comput. Soc; **1997**; pp. 8–12.
[54] Sukert, A. N. An investigation of software reliability models. In Annual Reliability and Maintainability Symposium, Philadelphia, Pa; **1977**; pp. 478–484.

[55] Yaghoobi, T. Parameter optimization of software reliability models using improved differential evolution algorithm. *Mathematics and Computers in Simulation*, 2020, 177, 46-62.

[56] Tamura, Y.; Matsumoto, M.; Yamada, S. Software Reliability Model Selection Based on Deep Learning. In Proceedings of the 2016 International Conference on Industrial Engineering, Management Science and Application (ICIMSA); IEEE, 2016a; pp. 1–5.

[57] Tamura, Y.; Yamada, S. Software Reliability Model Selection Based on Deep Learning with Application to the Optimal Release Problem. *J. Ind. Eng. Manag. Sci.* 2016b, 2016, 43–58.

[58] Teng, X.; Pham, H. A New Methodology for Predicting Software Reliability in the Random Field Environments. *IEEE Trans. Reliab.* 2006, 55, 458–468.

[59] Tohma, Y.; Yamano, H.; Ohba, M.; Jacoby, R. The estimation of parameters of the hypergeometric distribution and its application to the software reliability growth model. *IEEE Trans. Softw. Eng.* 1991, 17, 483–489.

[60] Wald, A. *Sequential analysis of statistical data: theory. A report submitted by the Statistical Research Group, Columbia University to the Applied Mathematics Panel*. National Defense Research Committe; 1943.

[61] Wald, A. Sequential Analysis; JohnWiley and Sons: New York, NY, USA, 1947.

[62] Wall, J. K.; Ferguson, P. A. Pragmatic software reliability prediction. In Annual Reliability and Maintainability Symposium, Philadelphia, Pa; 1977; pp. 485–488.

[63] Wang, J.; Zhang, C. Software reliability prediction using a deep learning model based on the RNN encoder–decoder. *Reliab. Eng. Syst. Saf.* 2018, 170, 73–82.

[64] Weibull, W. A statistical distribution function of wide applicability. Journal of applied mechanics, 1951, 18, 293-297.

[65] Wood, A. Predicting software reliability. Computer (Long. Beach. Calif). 1996, 29, 69–77.

[66] Yamada, S.; Ohba, M.; Osaki, S. S-shaped Software Reliability Growth Models and Their Applications. *IEEE Trans. Reliab.* 1984, R-33, 289–292.

[67] Yamada, S.; Ohtera, H.; Narihisa, H. Software Reliability Growth Models with Testing-Effort. *IEEE Trans. Reliab.* 1986, 35, 19–23.

[68] Yamada, S.; Osaki, S. Software Reliability Growth Modeling: Models and Applications. *IEEE Trans. Softw. Eng.* **1985**, SE-11, 1431-1437.

[69] Zeephongsekul, P.; Jayasinghe, C.L.; Fiondella, L.; Nagaraju, V. Maximum-Likelihood Estimation of Parameters of NHPP Software Reliability Models Using Expectation Conditional Maximization Algorithm. *IEEE Trans. Reliab.* **2016**, 65, 1571-1583.

[70] '「한국판 뉴딜 종합계획」 발표', 기획재정부, 2020.07.14., " https://www.moef.go.kr/nw/nes/detailNesDtaView.do?searchBbsId1=MOSFBBS_000000000028&searchNttId1=MOSF_000000000040637&menuNo=4010100"(2021.01.15. 접속)

[71] 김종하. *역사 속의 소프트웨어 오류*; 에이콘, **2014**;

[72] 김윤수, 장인홍, 이다혜. 심층신경망을 활용한 비모수적 소프트웨어 신뢰성 모형과 NHPP 소프트웨어 신뢰성 성장 모형 비교. *Journal of the Korean Data Analysis Society*, **2020**, 22, 2371-2382.

[73] 박동호, 임재학, 남경현, 정기문. *신뢰성 이론과 수명 분포의 이론*; 자유아카데미, **2015**.

[74] 정해성, 박동호, 김재주. *신뢰성 분석과 응용*; 영지문화사, **2003**.