고신뢰성 차량 임베디드 컴퓨팅 시스템의 백업 최소화 방안

박기진 · 김광섭 · 최석호
아주대학교 산업정보시스템공학부

A Mechanism of Minimizing Backups for Highly Dependable Vehicle Embedded Computing Systems

Kiejin Park, Gwang-sub Kim, Seokho Choi
Division of Industrial & Information Systems Engineering, Ajou University

Abstract

It is not easy to apply fault-tolerant techniques which are used in conventional computer systems successfully to the field of embedded computing system directly. In this paper, we study on the way of minimizing hardware and/or software backups for vehicle embedded computing systems. First, we group parts that constitute vehicle embedded systems and next feature subset is determined using the grouping information derived. The possibility of implementing graceful degradation capability in vehicle embedded systems is verified.

1. 서론

무어의 법칙에 따른 반도체 기술의 발전으로 인해, 과거에 하드웨어가 수행했던 기능을 소프트웨어가 대신 처리하는 임베디드 컴퓨팅 패러다임이 급속히 확산되고 있으며, 임베디드 소프트웨어의 복잡도 증가로 인해 임베디드 시스템을 구성하는 하드웨어보다는 소프트웨어적인 측면이 시스템 개발 비용을 결정하는 주요 요인으로 작용하고 있다. 예를 들어 자동차 산업 분야는 대표적인 분산 임베디드 제어 응용(Instrumentation & Control) 중에 하나이며, 차량 한 대에 수백 개의 마이크로프로세서가 탑재(즉, 복잡한 분산 실시간 임베디드 컴퓨팅 시스템)되고 있는 상황이다.

분산성(Distributed), 실시간성(Real-time) 등 소프트웨어적인 요구 조건이 기존의 컴퓨터 시스템 보다 엄격한 임베디드 컴퓨팅 시스템은 하드웨어적인 결함에 의해 고장이 발생할 가능성 보다는 오히려 소프트웨어적인 원인에 의해 고장이 야기될 가능성이 상당히 높기 때문에, 임베디드 컴퓨팅 시스템을 구성하는 하드웨어와 소프트웨어의 정상적인 동작을 보장하기 위해서는, 하드웨어 부품과 소프트웨어 로직 등의 결함허용(Fault-tolerant) 능력을 반드시 확보해야 한다.

하지만, 이론상 결함이 없는 임베디드 컴퓨팅 시스템의 개발은 불가능하며, 설령 개발이 가능하다 할지라도, 하드웨어와 소프트웨어 개발 및 테스트 비용이 막대할 것으로 예상되므로, 어느 정도 결함이 포함될 가능성이 있다는 가정 하에, 임베디드 컴퓨팅 시스템의 신인도(Dependability: "신인도" 라는 용어는 아직 학계에서 일반화되지 않았으며, 본 논문에서는

본 연구는 ㈜ NGV의 "2005년도 차세대자동차 선행기술" 과제의 연구비 지원으로 수행되었음

"의존도" 라는 직역보다는 "신인도" 라는 의역을 사용함)를 설계하는 것이 바람직하다. 즉 결함이 있더라도 시스템의 원래 목적이나 기능을 달성할 수 있는 임베디드 컴퓨팅 시스템에 적합한 결함허용 개념을 도입하여야 한다.

기존 결함허용 컴퓨터시스템의 신인도 평가 척도로는 1) 신뢰도(Reliability) 2) 가용도(Availability) 3) 안전도(Safety) 4) 유지보수도(Maintainability) 5) 생존도(Survivability) 등이 있으며, 응용 시스템에 따라 추구해야 하는 신인도 척도가 달라진다[1]. 예를 들어 비행기의 경우, 이륙해서 착륙하는 시간 동안에는 반드시 정상적 가동해야 하므로, 일정기간 동안 시스템이 정상적으로 작동하는 확률을 제시하는 신뢰도에 초점을 맞추어 시스템을 개발해야 하며, 통신시스템의 경우 일정 기간(예: 1년) 동안 전화를 못 걸거나 못 받는 시간의 합이 최소화되는 개념을 설명하는 가용도 측면에서 설계가 이루어져야 한다. 일반적으로 가용도가 99.9999% (6 Nines) 이면, 교환기를 1년간 가동할 경우 약 30초 정도의 다운타임을 갖는다[2].

비교적 고가의 개발 비용이 투입될 수 있고, 가동 전원이 풍부하고, 사용 환경이 우수한 곳에서 원하는 신인도를 달성할 수 있는 컴퓨터 시스템에 대한 개발 및 상용 분야의 응용이 지난 40여 년간 활발히 이루어졌고, 또한 상당히 성공적이라 평할 수 있지만[3], 기존 컴퓨터시스템에 적용했던 4가지 대표적인 결함허용 개념,

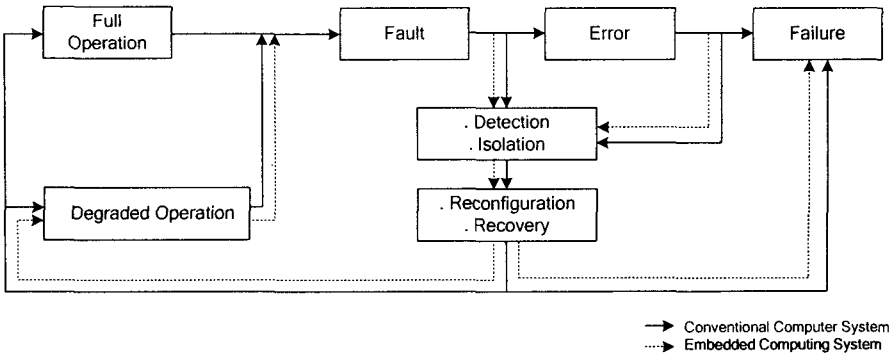
- 하드웨어 결함허용: 별도의 여분(Redundancy) 하드웨어를 추가하여 주(Primary) 하드웨어에 결함이 발생할 경우, 여분의 하드웨어(Backup)가 주 하드웨어의 역할을 대신 수행하는 방식
- 소프트웨어 결함허용: 연산 수행 결과를 승인 검사(Acceptance Test)를 통과시켜, 소프트웨어적인 계산 결과를 검증하는 방법인 복구 블록(Recovery Block), 혹은 여러 개의 소프트웨어 버전을 동시에 수행하여, 수행된 결과를 투표기를 이용해서 결정하는 N-version 프로그래밍 등을 사용하는 방식
- 정보 결함허용: 패리티, Cyclic Redundancy Code 와 같이 코딩 기법을 이용하는 방식
- 시간 결함허용: 주 연산 수행 후 남아도는 잉여 CPU 능력(여분용량: Spare Capacity)을 결함 감지 등에 이용하는 방식

등을 임베디드 컴퓨팅 시스템 분야(예: 자동차, 항공, 우주, 군사, 산업제어, 정보통신, 정보가전 등)에 그대로 적용하기에는 다음에 설명하는 바와 같이 무리가 있다.

- 임베디드 컴퓨팅 시스템의 사용 환경이 전산실에서 주로 가동되는 기존 컴퓨터시스템에 비해 상당히 열악하다. (예: 온도, 압력, 충돌, 기후)
- 목표하는 기능이 동작하지 않을 경우 사용자의 생명에 지장을 준다(예: Anti-lock Braking System)
- 여분의 하드웨어 혹은 소프트웨어를 설치할 공간 혹운 전원 등이 부족하다(예: 로봇, 정보통신 기기)
- 소프트웨어의 복잡도가 매우 높다(예: 실시간 컴퓨팅, 네트워크 컴퓨팅)

다시 말하면, 임베디드 컴퓨팅 시스템에서는 공간적/시간적 제약으로 인해 이미 알려진 하드웨어 및 소프트웨어 결함허용 방식을 사용하기 어렵다는 의미이다.

그림 1은 본 연구에서 제안하는 임베디드 컴퓨팅 시스템의 결험허용 구조를 표시하고 있으며, 점선으로 표시된 흐름이 본 연구에서 고려



<그림 1> 임베디드 컴퓨팅 시스템의 결함허용 구조

하고 있는 임베디드 컴퓨팅 시스템의 결함허용 구조이다. 기존의 결함허용 컴퓨터 시스템에 사용하는 결함 감지 (Detection), 결함 분리 (Isolation), 결함 복구(Recovery), 및 시스템 재구성(Reconfiguration) 등의 복잡한 결함허용 기법을 적용하여, 완전한 동작(Full Operation)으로 복귀하기 보다는 일정 수준의 성능 감퇴를 허용하는 방향(Degraded Operation)으로 임베디드 컴퓨팅 시스템의 신인도를 확보하고자 한다. 결함이나 오류가 감지된 후, 미리 설계된 특정 구성으로 동작할 수 있게 하기 위해서, 임베디드 컴퓨팅 시스템에서 가능한 동작 구성을 미리 파악하여야 하며, 이 때 각 구성들에 대한 유용도(Utility) 평가가 이루어져야 한다.

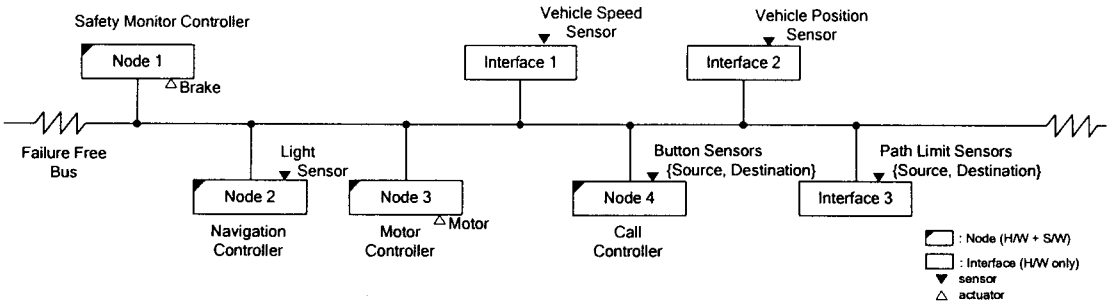
2. 관련 연구

시스템을 구성하는 모든 개별 부품의 고장으로 인해 고려해야 할 시스템 구성조합(Configuration)의 수는 지수적으로 많지만, 이들 모두를 시스템 설계 시에 고려한다는 것은 불가능하다. 예를 들어 N 개의 부품(Sensor, Actuator, Module 등)으로 구성된 시스템은 모두 2^N 개의 서로 다른 시스템 구성조합을 가지며, 각 구성조합을 이용한 재구성 작업은 NP-complete 문제이므로, 발견적 기법(Heuristic)을 고안해야만 한다[4,5].

[6]에서는 각 부품 그룹들 사이의 관계(Feature Subset)를 정의하여, 그림 1에 나타난 재구성 작업을 위한 구성조합의 유용도를 계산하였으며, 시스템이 영구적 고장으로 장애를 받고 있는 경우, 재구성 절차에 의해 고장난 시스템을 스스로 치유하고자 하였다. 이러한 자가치유(Self-healing) 개념[7]이 적용된 자동차를 설계하기 위해서는, 자동차에 내재하는 센서 출력 값의 이상에 대해 결함 모델링을 통해서, 이들의 발현 소스, 발현 주기, 제어 시스템의 반응(센서 이상, 연비/출력 등의 감소) 등에 대한 정의가 필요하다.

[8]에서는 임베디드 컴퓨팅 시스템을 구성하는 부품 일부에서 문제가 생기면, 나머지 정상 동작 부품들이 고장난 부품의 기능을 대신 수행할 수도 있다는 기능적 대안(Functional Alternative) 개념을 제시했으며, 이는 백업을 설치하는 기존의 결함허용 개념과는 근본적으로 차이가 있다고 할 수 있다. 백업이 아닌 기능적 대안 개념이 적용된 엘리베이터 제어기의 경우, 엘리베이터 구성 부품의 75%가 고장이 나더라도 원래의 주목적(Primary Objective: 승객을 안전하게 목적지까지 이동시키는 작업)의 달성 가능함을 설명하였다.

멀티프로세서 시스템(예: 다수의 임베디드 프로세서를 탑재하고 있는 차량)에서 데드라인 있는 비선점형 태스크 스케줄링 문제는 NP-Hard로 알려져 있다. 멀티프로세서로 구성된



<그림 2> AGV 주행 제어 시스템 구조

시스템에서 특정 프로세서에 고장이 발생하는 경우, 이에 대한 해결 방안으로 여분 용량을 둘 경우, 시스템의 이용률이 낮아지는 단점이 있는 반면에 과부하 관리 기법을 사용할 경우, 몇몇 태스크의 데드라인이 위배될 가능성이 높아진다. 이러한 문제를 해결하기 위해서 [9]에서는 시스템의 성능이 저하된 상태로 가동할 수 있는 (m,k)-firm Guarantee(총 m 개의 태스크 중에서 k 개의 태스크 데드라인이 지켜지는 것을 보장하는 시스템)의 개념을 제시하여 분석하였다.

[10]에서는 하드웨어와 소프트웨어를 모두 고려하는 임베디드 시스템의 신뢰도를 계산하는 모델을 제시한 후 이에 대한 최적화 해를 구했으며, 기존의 최적화 모델과는 달리 하드웨어와 소프트웨어가 서로 종속적이라는 가정하에 분석을 수행했지만, 임베디드 컴퓨팅 시스템에는 하드웨어 혹은 소프트웨어 백업을 설치할 수 있는 충분한 여유가 없다는 점을 고려하지 않았다.

[11]에서는 프로세스 제어 응용을 위한 우아한 성능 감퇴를 허용하는 컴퓨터 시스템의 Performability 를 평가하였다. 결함이 발행한 부품 혹은 시스템의 발견 및 제거에 대한 Coverage 확률을 고려하였으며, 이를 통해 최적 수리 기법의 제시는 물론 최적 여분 프로세서 수를 산정하였다.

고신뢰성 차량 임베디드 컴퓨팅 시스템에 관한 본 논문은 기존의 무거운 결함허용 기법을 사용하지 않고, 비교적 가벼운 임베디드 시스템에 적합한 하드웨어 혹은 소프트웨어 백업을 최소로 사용하는 결함허용 구조 및 알고리즘 도출에 관한 기초 연구이며, 도출된 결함허용 알고리즘의 실용화 가능성 검토를 위해, 그림 2와 같은 구조를 가지는 샘플 AGV(Automatic Guided Vehicle)를 대상으로 핵심 부품이 고장이 날 경우에도 임베디드 컴퓨팅 시스템의 원래의 제공하여야 하는 주기능을 성능이 감퇴된 상태로 수행할 수 있는 가능성을 검토하였다.

3. 우아한 성능 감퇴(Graceful Degradation)를 고려한 차량 임베디드 컴퓨팅 시스템

그림 2 는 본 논문에서 고려하고 있는 간단한 AGV 주행 제어 시스템 구조를 나타내고 있다. 대상 시스템은 하드웨어와 소프트웨어가 모두 있는 컴퓨팅이 가능한 노드와 센서 혹은 액츄에이터만으로 이루어진 인터페이스로 구성되 있으며, 노드와 인터페이스는 신뢰성이 높은 네트워크 상에서 상호 데이터를 주고 받는다. 예를 들어, Node 1은 안전성 감시 제어기와 브레이크로 이루어져 있으며, 사용자의 안전에 문제가 생길 경우에는 즉각 자동차의 주행을 멈추게 한다. 그 밖에 주행 제어기, 모터 제어기, 호출 제어기 등이 있으며, 이들은 다양한 인터페이스를 통해서 차량의 속도, 위치, 경로, 운행 요청 등의 정보를 수집하여, 모터

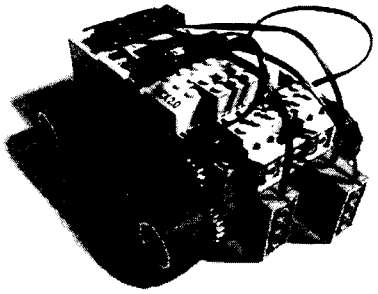
에 동작 명령을 내린다. 우아한 성능 감퇴 기능을 설계하기 위해 다음과 같은 항목을 순차적으로 고려해야 한다.

- AGV 를 구성하는 부품을 그룹핑하여, 이 그룹들 간의 관계를 정의하고 이를 통해, 각 부품 그룹에 해당하는 구성조합을 결정한다.
- 각 구성조합이 발생할 확률 모형과 각 구성 조합에 해당하는 유용도 함수를 정의하여, 전체 시스템의 유용도를 도출한다. 유용도는 우아한 성능 감퇴 기능, 신뢰성, 주행 성능, 생존성 등을 동시에 반영하는 성능 평가 척도로서 사용된다.
- 특정 부품이 고장 날 경우, 이를 검출(Fault Detection) 해내는 능력을 고려한 Coverage 개념을 도입하여, 보다 정확한 안전도 평가기능이 반영되도록 한다. 특히 차량 임베디드 시스템에서 안전도는 중요한 평가 항목이다.

3.1 라인 트레이서의 구성

하드웨어와 소프트웨어를 모두 고려한 우아한 성능 감퇴 기능 구현 가능성 검증을 목표로 그림 2 의 구조를 간략화시킨 라인 트레이싱 AGV 를 제작하였다(그림 3 참조). 샘플 AGV 는 레고에서 제공하는 2 개의 구동 모터(Actuator)와 3 개의 빛 센서 및 RCX 를 이용하였으며, 구동모터는 독립적으로 좌우 캐터필러를 가동하는 데 쓰이며, 3 개의 빛 센서는 2 개의 좌/우 Primary 센서와 1 개의 Backup 센서로 구성되어 있다. RCX 는 빛 센서로부터 입력 값을 받아서 모터를 구동하기 위한 출력 값을 정해주는 8 bits 마이크로 프로세서이다.

자동차와 같이 안전도가 절대적으로 필요한 임베디드 컨트롤 시스템에서는 주기능의 수행 보장을 위해 최소한의 여분 하드웨어 설치가 필수적이므로(예: 실제 자동차 브레이킹 시스템의 경우 이중화로 구현됨), AGV 경로를 정보를 생성하는 센서 고장에 대해서 최소한의 하드웨어적인 이중화를 고려하는 것은 본 논문에서 다루는 차량 임베디드 시스템의 신인도 확보에 반드시 필요하다고 할 수 있다.



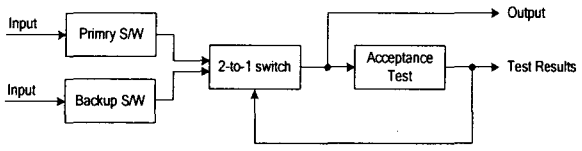
<그림 3> LEGO RCX 를 사용한 샘플 AGV 차량

3.2 소프트웨어 결함 모델(복구 블록)

구성부품의 고장에 따른 각 구성조합 별로 우아한 성능감퇴 기능을 확인하기 위해서, AGV 를 구성하는 중요한 부품 하나인 적외선 센서를 하드웨어적으로 이중화시켜서, 시스템의 주행 성능 및 신뢰도를 높이고자 하였으며, 또한 이들에게 Active-Active 백업 개념을 적용하여, 비록 백업일 지라도 주행 경로 결정 연산 작업에 참여하도록 하였다. 이렇게 함으로써, AGV 경로를 결정하는 연산에 2 개의 센서가 사용되어, AGV 의 주행 속도를 높이는 동시에, 어는 한 특정 센서에 결함이 발생하더라도 원래의 기능을 성능이 감퇴된 상태로 수행할 수 있도록 센서가 모두 정상적으로 동작할 때와 일부 센서에 고장이 발생하여 성능이 저하된 상태에서 수행되는 개별적인 소프트웨어 로직을 구현하였다.

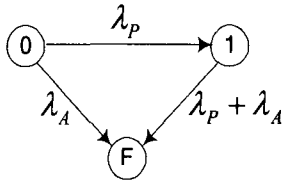
우선 2 개의 소프트웨어 버전이 동시에 수행되는 상황은 그림 4 와 같이 소프트웨어

결함허용 방식의 하나인 복구 블록 (Recovery Block) 으로 모델링 하였다.



<그림 4> 2개의 소프트웨어 버전을 가진 복구 블록 흐름도

시스템의 입력 값은 최초에 주 소프트웨어 버전에서 만 수행되며, 계산 결과는 승인 실험을 통과해야만 정식 출력 값으로 인정된다. 만약 승인이 거부될 경우에는 스위치에 의해 다른 백업 버전으로 재 수행(Fault Detection Trigger)하게 된다. 그림 4에 해당하는 복구 블록의 신뢰도 모델은 그림 5와 같은 마코프 체인으로 변환할 수 있으며, 동시에 2개의 블록이 고장 나거나 혹은 주 버전의 고장을 감지하지 못할 경우에 원하는 결과를 주지 못하는 상황을 표시하고 있다.



<그림 5> 2 버전 복구 블록의 신뢰도 모델

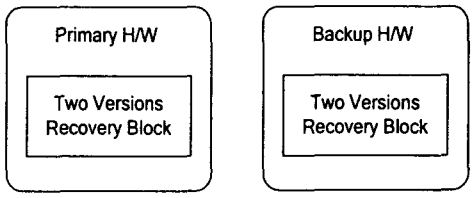
그림 5에 해당하는 2 버전 복구 블록의 신뢰도는 위의 상태 0,1에서 머물 확률의 합으로 식 (1)과 같이 정의된다.

$$R_{RB}(t) = \sum_{i=0}^{1} \lambda_P^i \frac{1}{i!} t^i e^{-(\lambda_P + \lambda_A)t} \qquad (1)$$

3.2 하드웨어/소프트웨어를 동시에 고려한 결함 모델

그림 6는 본 논문에서 고려하고 있는 하드웨어 및 소프트웨어 결함 허용 개념이 혼합된 차량 임베디드 시스템 신인도 향상 구조이다. 이러한 구조에서 특정 임베디드 컨트롤 시스템이 결과값을 산출하지 못할 확률은 $P_{PH}P_{BH} + P_{PS}P_{BS}$ 이며, 각각의 의미는 주 하드웨어와 백업 하드웨어가 동시에 고장 날 확률과 주 소프트웨어와 백업 소프트웨어가 동시에 고장 날 확률의 합의로 정의된다. 이때 사용한 가정한 다음과 같다.

- 모든 소프트웨어 버전의 고장을 다른 소프트웨어 버전의 고장과는 독립적이다.
- 투표기(Voter)는 고장이 나지 않는다.



<그림 6> 하드웨어/소프트웨어 Redundancy 기법을 동시에 고려한 결함허용 구조

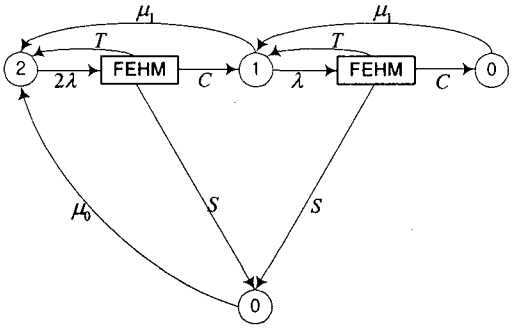
식(2)는 그림 6와 같이 하드웨어적으로는 동일한 백업을 갖추고, 소프트웨어적으로는 다양한 시스템 구성 변화를 반영하는 이중화된 시스템의 고장 확률이다.

$$P_{PH} * P_{BH} + P_{RB} = P_{PH} * P_{BH} + (1 - R_{RB}(t)) \qquad (2)$$

3.2 결함 Coverage 모델

식(2)를 현실적인 문제에 적용하기 위해서는 그림 6으로 모델링된 시스템을 구성하는 부품의 하드웨어적인 고장을 감지하는 확률(Coverage)을 반드시 고려해야 한다. 그림 7은 센서나 액츄에이터의 고장이 발생할 경우 이를 감지해 내는 능력인 결함 Coverage 가 고려된

시스템 모델이다. Coverage는 우아한 성능 감퇴 기능을 자동적으로 구현하기 위한 필수 항목이며, 이를 해석하기 위해서는 FEHM(Fault Error Handling Module)의 설계 및 분석 작업이 이루어져야 한다.



<그림 7> 영구적/임시적 결함을 고려한 Coverage가 고려된 AGV의 결함 모델

4. 결론

본 논문에서는 차량 임베디드 시스템을 구성하는 하드웨어/소프트웨어의 결함을 동시에 고려할 수 있는 우아한 성능감퇴가 가능한 임베디드 컨트롤 시스템의 자가치유 결함 모델을 설계하였으며, 모델의 테스트 베드(Test bed)로 센서와 액츄에이터 및 프로세서를 가진 샘플 AGV 프로토타입을 구현하였다.

향후에는 테스트 베드 상에 다양한 파라미터에 대한 AGV 시뮬레이션을 수행할 예정이며, 또한 본론에서 언급한 각 결함 모델의 확률의 분포를 구할 예정이다. 이를 통해, 특정 구성조합의 발생 가능성이 파악되며, 각 구성조합에서의 유용도가 확률적으로 계산되므로, 전체 시스템의 평균 유용도를 증가 시키는 최적 시스템 결함 허용 구조의 검증이 가능하게 된다.

참고문헌

[1] A. Avizienis, et al., (April 2001), "Fundamental Concepts of Dependability," Research Report N01145, LAAS-CNRS.

[2] E. Marcus and H. Stern, (2003), Blueprints for High Availability: 2nd edition, Wiley, pp. 624.

[3] David B Little, et al., (2003), Implementing Backup and Recovery: The Readiness Guide for the Enterprise, Wiley, pp. 416.

[4] W. Nace and P. Koopman, (Oct. 2000), "A Product Family Approach to Graceful Degradation," Proceedings of Distributed and Parallel Embedded Systems (DIPES 2000). pp.131-140.

[5] W. Nace and P. Koopman, (Oct. 2001), "A Graceful Degradation Framework for Distributed Embedded Systems," Workshop on Reliability in Embedded Systems.

[6] C. Shelton, et al., (Jan. 2003), "A Framework for Scalable Analysis and Design of System-wide Graceful Degradation in Distributed Embedded Systems," WORDS 2003.

[7] P. Koopman, (May 2003), "Elements of the Self-healing System Problem Space," Workshop on Architecting Dependable Systems (WADS03).

[8] C. Shelton and P. Koopman, (July 2004), "Improving System Dependability with Functional Alternative," 2004 International Conference on Dependable Systems and Networks, pp. 295 – 304.

[9] P. Ramanathan, (1997) "Graceful Degradation in Real-time Control Applications using (m,k)-firm Guarantee.

[10] N. Wattanapongsakorn, and S. Levitan, (Sep. 2004), "Reliability Optimization Models for Embedded Systems with Multiple Applications," IEEE Transactions on Reliability, Vol. 53, No. 3, pp. 406-416.

[11] C. Constantinescu, (Dec. 1992) "Predicting Performability of a Fault-tolerant Microcomputer for Process Control," IEEE Transactions on Reliability, Vol. 41, No. 4.