우아한 성능감퇴를 위한 임베디드 시스템의 유용도 설계

(Utility Design for Graceful Degradation in Embedded Systems)

강 민 구 † 박 기 진 ††
(Minkoo Kang) (Kiejin Park)

요 약 임베디드 시스템의 신인도(Dependability)를 높이기 위해 기존 컴퓨터시스템에서 주로 사용되는 결함허용(Fault-tolerant) 기법을 그대로 적용시키는 것은 임베디드 시스템의 엄격한 비용 제약과 설치 공간 부족 등을 고려할 때 적합하지 않다. 본 논문에서는 각 시스템 구성요소(Component)들의 여분(Redundancy)을 최소한도로 사용하는 임베디드 시스템에 적합한 소프트웨어 결함허용 기법을 제안한다. 이를 위하여 임베디드 시스템의 신인도를 반영하기 위한 기준인 유용도(Utility) 척도를 정의하고, 실제 시스템의 결함허용을 위해 각각의 시스템 구성요소들의 재구성(Reconfiguration) 조합에 대한 유용도 평가를 수행하였다. 이러한 유용도 평가는 일반적으로는 지수복잡성(Exponential Complexity)을 가지게 되나, 본 논문에서는 각각의 구성요소에 대한 소프트웨어 수준의 계층적 그룹화 개념을 이용하여 복잡도를 크게 감소시켰다. 이를 통해 임베디드 시스템의 일부 부품에 결함이 발생했을 시, 시스템의 고장(Failure)을 방지할 수 있도록 전체 시스템 단계에서 가능한 최대 유용도를 제공하는 구성조합으로의 재구성 작업을 가능하게 하였다.

키워드 : 신인도, 생존성, 임베디드 소프트웨어, 시스템 재구성, 유용도

Abstract As embedded system has strict cost and space constraints, it is impossible to apply conventional fault-tolerant techniques directly for increasing the dependability of embedded system. In this paper, we propose software fault-tolerant mechanism which requires only mimimum redundancy of system component. We define an utility metric that reflects the dependability of each embedded system component, and then measure the defined utility of each reconfiguration combinations to provide fault tolerance. The proposed utility evaluation process shows exponential complexity. However we reduce the complexity by hierachical subgrouping at the software level of each component. When some components of embedded system are failed, reconfiguration operation changes the system state from current faulty state to pre-calculated one which has maximum utility combination.

Key words : Dependability, Survivability, Embedded software, System reconfiguration, Utility

1. 서 론

최근 컴퓨터시스템 사용자들은 점점 원하는 시간 내에 원하는 서비스를 받을 수 있는 가능성이 보장되는 고신인도(High Dependability: Reliability, Availability, Safety 등) 시스템을 요구하고 있다[1]. 고신인도 시스템을 구축한다는 것은, 시스템이 고장(Failure) 상태에 이르지 않도록 지속적으로 관리할 수 있다는 의미이며, 이러한 고신인도 시스템에 대한 요구는 기존 컴퓨터시스템은 물론 최근 각광 받고 있는 임베디드 시스템(Embedded System)에도 확산 적용되고 있다. 임베디드 시스템이란 미리 정해진 특정한 기능을 수행하기 위하여, 하드웨어와 소프트웨어를 조합한 컴퓨터 제어시스템을 말하며, 응용프로그램에 따라 여러 기능을 수행할 수 있는 기존 컴퓨터시스템(예: PC)과는 구별되는 '내장된' 시스템의 성격을 지닌다. 이러한 임베디드 시스템은 프로세서, 센서, 액츄에이터 및 소프트웨어 등으로 구성되어 있으며, 분산성(Distributed), 실시간성(Real-time)

· 본 연구는 한국학술진흥재단 2006년 정부(교육인적자원부)의 재원으로 한국학술 진흥재단의 지원을 받아 수행된 연구임 (KRF-2006-331-D00438)
† 학생회원 : 아주대학교 공과대학 산업정보시스템공학부
ozige@ajou.ac.kr
†† 종신회원 : 아주대학교 공과대학 산업정보시스템공학부 교수
kiejin@ajou.ac.kr
논문접수 : 2006년 1월 19일
심사완료 : 2006년 11월 13일

과 같은 성격을 지니고 있다[2].

시스템의 고장은 근원적으로는 시스템에 내재하는 결함(Fault)으로부터 비롯되며, 미처 제거되지 못한 결함이 발현하여 오류(Error)를 야기시킨다. 오류가 지속적으로 발생할 경우, 결국에는 시스템 서비스 중지 상태에 이르게 되므로, 이러한 시스템 고장을 막기 위해서는, 근원적으로 결함을 다루어야 한다. 기존 컴퓨터시스템에서는 아래와 같은 4 가지의 대표적인 결함처리 기술이 사용되고 있다[3,4].

1) 결함 방지(Prevention): 결함 발생을 예방하는 기술로 디자인/생산 단계에서 적용
2) 결함 허용(Tolerance): 결함이 존재함에도 불구하고 올바른 서비스를 지속적으로 제공하는 기술
3) 결함 제거(Removal): 결함 제거를 위한 검증(Verification), 진단(Diagnosis), 정정(Correction)의 3단계로 구성
4) 결함 예측(Forecasting): 시스템에 대한 일련의 평가에 의해 현재의 결함 또는 앞으로 발생할 결함의 빈도를 예측

이들 중에서 결함허용 기술은 결함이나 오류를 찾고(Detection) 복구(Recovery)하는 방법을 사용하여, 앞서 말한 4 가지 기술 중에서 가장 활발히 연구/응용되는 분야이다[5].

기존의 컴퓨터시스템에 성공적으로 적용되었던 대표적인 결함허용 기법들은 TMR(Triple Modular Redundancy) 이나 복구블록(Recovery Block)처럼 하드웨어 및 소프트웨어적으로 여분(Redundancy)을 두는 기법 등이었으나, 이러한 기법들을 임베디드 시스템에 그대로 적용시키기에는 여러 가지 이유로 무리가 따른다. 임베디드 시스템의 열악한 가동 환경(예: 온도, 압력, 충돌, 기후 등) 및 서비스 중단 시 사용자의 생명에 지장을 줄 수 있다는 점(예: 자동차, 항공기, 원자력발전소 등)은 높은 시스템 신인도를 요구하게 된다. 또한 실제 구현 시에는 여분의 하드웨어 및 소프트웨어를 설치할 공간, 비용 및 전력이 부족하다는 문제점이 있다. 따라서 그림 1에서 보는 바와 같이 임베디드 시스템의 결함허용 기술은 기존의 컴퓨터시스템에서 요구되었던 완전한 동작(Full Operation)으로의 복귀를 수행하기보다는 여러 제약조건들을 고려하여 어느 정도의 성능 감퇴(Degraded Operation)를 허용하는 방향으로 설정해야 할 필요성이 있다[6,7].

여분 사용을 최소화하면서 시스템의 서비스를 유지하기 위해서는 결함이 발생한 시스템을 재구성(Reconfiguration)하여 결함의 영향을 최소화하는 작업이 필수적이며, 이를 위하여 시스템의 모든 가능한 구성조합을 고려해야만 한다. 그러나 이러한 모든 구성조합을 찾는다는 것은 거의 불가능하다. 예를 들어 N개의 부품(프로세서, 센서, 액츄에이터 및 소프트웨어)으로 이루어진 시스템에서 가능한 구성조합의 수는 2^N개이므로, N의 값이 큰 복잡한 시스템일수록 모든 구성조합을 고려하여 일정 시간 내에 결함복구를 위한 특정 재구성 조합을 찾는 것은 어려운 문제이다. 이를 해결하기 위해 본 논문에서는 임베디드 시스템을 구성하는 하드웨어 및 소프트웨어 부품들을 소프트웨어 중심의 서브셋(Subset)들로 묶어, 부품들의 고장에 따른 시스템 재구성 작업을 위해 고려해야 할 구성조합(Configurations)의 수를 축약하여 계산하였으며, 도출된 각 구성조합들의 유용도(Utility)를 산출하여, 재구성 가능한 임베디드 시스템의 유용성을 분석하였다. 본 논문의 2장에서는 관련 연구를 언급하고, 3장에서는 제안된 임베디드 시스템의 결함허용 방법을 설명한 후, 모노레일 제어 시스템 분석 사례를 제시하였으며, 4장에서는 레고 블록을 사용한 프로토타입을 제작하여 제안된 방법의 성능 평가를 수행하였고, 5장에서는 결론을 내렸다.

2. 관련 연구

여분 사용이 최소화된 임베디드 시스템에서 하드웨어/소프트웨어 요소들을 재구성함으로써, 결함이 발생한 시

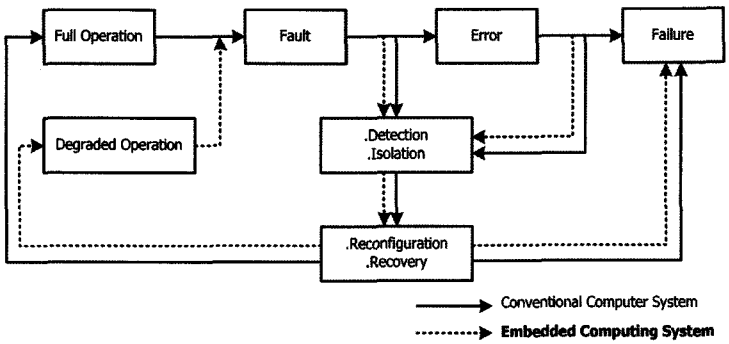

그림 1 임베디드 시스템의 결함 허용 구조

스템의 성능을 우아하게 감퇴(Graceful Degradation) 시킬 수 있다[8]. 우아한 성능감퇴 개념은 1) 결함을 감지하는 작업, 2) 결함이 감지된 부품이 시스템에 미치는 영향을 제거(Isolation)하는 작업, 3) 나머지 부품들의 상호 작용을 고려하여 올바른 서비스를 지속적으로 제공할 수 있도록 시스템을 재구성하는 작업이 동시에 이루어질 때 가능하다. 부품 결함으로 인한 시스템 재구성 작업은 적절한 하드웨어/소프트웨어 요소를 선택하여, 새롭게 배정(Allocation)하는 것을 뜻하며, 이러한 문제는 일반적으로 지수적(Exponential)으로 증가하는 복잡도를 가진다. 그렇기 때문에 각 시스템별로 재구성 작업을 얻기 위해서는 각 경우마다 발견적 기법(Heuristic)을 고안하여 해결해야 한다.

시스템을 구성하는 부품의 여분을 두지 않고, 시스템의 일부 부품에 고장이 발생한 경우 나머지 부품들이 고장난 부품의 여분이 아님에도 불구하고 고장난 부품의 기능을 어느 정도 대신할 수 있다는 기능적 대안(Functional Alternative) 개념을 제시하여 임베디드 시스템의 신인도를 향상시키는 연구가 선행되었다[9-11]. 임베디드 시스템을 구성하는 각 부품의 고장 여부에 따라 발생하는 시스템의 상태를 구성조합이라고 정의하고, 이러한 구성조합의 수를 줄이기 위해서 시스템을 구성하는 모든 하드웨어 부품들을 각 부품 간의 데이타 흐름을 파악한 다음, 비슷한 특성을 가지는 부품들로 나누어 묶어 여러 개의 Feature Subset으로 분류하였다. 일반적으로 N개의 부품으로 구성된 시스템의 구성조합의 수는 2^N이지만, m개의 Feature Subset으로 분류하게 되면 구성조합의 수는 $m*2^k$(k는 한 Feature Subset에서의 최대 부품 수)으로 줄어들게 된다. 또한 임베디드 시스템의 설계에 주로 요청되는 신인도와 성능을 동시에 확보하기 위해, 이 두 가지가 모두 고려된 통합 척도로서 유용도를 정의하였고, 엘리베이터 제어 시스템, 네비게이션 시스템의 사례 연구를 통해 Feature Subset, 구성조합의 감소 및 유용도 평가 과정을 설명하였다. 하지만 구성조합의 수를 줄인다는 것은 Feature Subset이 전체 시스템과는 무관하게 독자적으로 재구성 작업을 수행할 수 있다는 것이 전제되어야 하며, 이를 위해서는 Feature Subset 내에 반드시 소프트웨어 요소가 포함되어 있어야만 하나 이를 고려하지 않았다. [12]에서는 임베디드 시스템의 신뢰도 최적화를 위해, 어떠한 부품의 여분을 얼마나 두어야 하는 가에 대한 방법이 제안되었고, 이를 위해 비용 제약하에서 시스템의 신뢰도를 최적화하는 4개의 모델과 신뢰도 제약하에서 비용을 최적화하는 1개의 모델을 설계하여 실험하였다.

시스템의 전체 부품의 고장 발생 여부를 모두 한꺼번에 고려하지 않고, 각각의 Feature Subset 내에 포함된 부품들의 고장만을 고려하는 Feature Subset 개념을 적용하여 실질적인 임베디드 시스템을 구현하기 위해서는 각각의 Feature Subset 내의 부품들의 고장만을 고려할 수 있는 소프트웨어 요소가 반드시 필요하다. 또한 복잡한 시스템일수록 요구되는 기능이 증가함에 따라 Feature Subset의 수도 급격히 증가하게 되므로 구성조합의 수 또한 지수적으로 증가할 가능성이 높다. 따라서 기능적 대안 기법이 실제로 적용되기 위해서는 각 시스템에서 결정적(Deterministic)으로 Feature Subset의 수가 정해질 필요성이 있다.

이에 본 논문에서는 소프트웨어 중심의 서브셋 개념을 적용하여, 임베디드 시스템의 재구성 작업을 설계 및 구현하였다. 즉 발생 가능한 시스템 결함에 따른 구성조합을 모두 파악하고, 각 구성조합을 대상으로 시스템을 재구성할 수 있는 소프트웨어 결함허용 기법을 제안하였으며, 각 구성조합에 대해 시스템 전체의 유용도 평가를 수행하였다.

3. 시스템 모델

본 논문에서 다루고자 하는 임베디드 시스템은 프로세서, 센서, 액츄에이터 및 소프트웨어 부품들로 구성되어 있으며, 결함 발생에 대비한 하드웨어/소프트웨어의 여분은 없다고 가정하였다. 또한 임베디드 시스템을 구성하는 각 부품의 결함 발생은 Fail-fast, Fail-silent 하며 부품 간 데이타 통신 네트워크의 고장은 배제하였고, 각 부품의 고장 발생 여부는 즉각적으로 감지된다고 가정하였다.

여분을 사용하지 않는 임베디드 시스템의 재구성 작업은 디자인 단계와 운영 단계로 구분하여 진행하였다. 우선 디자인 단계에서는 1) 임베디드 시스템을 구성하는 각 부품을 정의하고, 2) 부품 간의 데이타 흐름을 파악하여 소프트웨어 중심의 서브셋을 도출한 뒤, 3) 서브셋별로 부품의 고장 여부에 따라 발생 가능한 구성조합을 모두 파악하고, 4) 각 구성조합의 유용도 평가를 수행한다. 이를 통해 임베디드 시스템의 일부 부품이 결함 발생 시 가능한 최대의 유용도를 제공하는 구성조합으로의 시스템 상태 천이를 결정할 수 있다. 임베디드 시스템의 운영 단계에서는 디자인 단계에서 수행된 각 구성조합의 유용도 평가를 바탕으로 1) 부품의 결함을 감지하고, 2) 나머지 가용한 부품을 파악하여, 3) 해당 구성조합으로의 재구성 작업을 통해 부품의 결함을 복구한다.

3.1 소프트웨어 중심의 서브셋

소프트웨어 중심의 서브셋을 도출하기 위해 우선적으로 임베디드 시스템을 구성하는 각 부품에 대한 정의가 필요하다. 임베디드 시스템이 제공하는 기능에 따라 조

금씩 차이가 있겠지만 임베디드 시스템은 대체로 프로세서, 센서, 액츄에이터 및 소프트웨어로 구성된다(예: 차량 Anti-lock Braking System을 구성하는 ECU, 휠 속도 센서, ABS 액츄에이터). 목표하는 기능에 맞게 정확히 정의된 부품들은 임베디드 시스템의 운영 단계에서 일정한 방향으로 데이타 흐름을 발생시킨다. 소프트웨어 중심의 서브셋은 해당 소프트웨어의 입출력 데이타 흐름을 발생시키는 모든 부품들의 집합이다.

한편 서브셋 내의 각 부품에서 발생되는 데이타 흐름은 중요도에 따라 강한 의존성(Strong Dependence) 데이타 흐름과 약한 의존성(Weak Dependence) 데이타 흐름으로 분류된다(그림 3 참조). 강한 의존성 데이타 흐름은 해당하는 부품이 고장 발생 시 전체 시스템 또는 해당 서브셋의 최소한의 기능까지도 상실하는 경우를 말하며, 예로 차량의 엔진 제어기를 들 수 있다. 반면 약한 의존성 데이타 흐름은 해당하는 부품의 고장에도 소프트웨어적으로 결함을 허용할 수 있는 경우를 말하며, 예를 들어 모노레일 제어 시스템의 Rail Sensor의 경우, 일부 Sensor의 고장에도 모노레일의 주행속도가 느려질 뿐, 최소한의 기능인 운송은 계속할 수 있는 것을 의미한다.

3.2 구성조합

소프트웨어 중심의 서브셋이 결정되면, 각 서브셋 별로 부품의 고장 여부에 따라 발생 가능한 각 구성조합의 수가 산출되며, 서브셋으로 분류하기 이전보다 그 수가 대폭 감소한다. 이를 수식으로 표현하면 서브셋 i 에서 고려해야 할 부품의 수가 k_i개라고 할 때, 해당 서브셋에서 모든 부품이 정상 동작하는 경우를 제외하면 고려해야 할 구성조합의 수는 $2^{k_i}-1$개이고, 이때 전체 시스템의 구성조합의 수는 식 (1)과 같이 표현된다.

$$\sum_{i=1}^{m}(2^{k_i}-1)+1=\sum_{i=1}^{m}2^{k_i}-m+1,$$

m은 서브셋의 수 (1)

한편 서브셋이 목표하는 기능을 위해 반드시 동작해야만 하는 부품들(예: ABS 시스템의 ECU, ABS 액츄에이터)의 고장이 발생할 경우에는 곧바로 서브셋이 목표하는 기능을 제공할 수 없게 되므로, 소프트웨어 결함 허용 기법을 적용한다는 것이 사실상 무의미하기 때문에 각 서브셋 별로 고려해야 할 부품의 수, k_i는 더욱 작아지게 되므로 전체 시스템의 구성조합의 수를 더욱 줄일 수 있었다.

3.3 유용도 함수

임베디드 시스템이 작동하는 도중, 일부 부품에 고장이 발생하게 되면, 시스템은 동작 가능한 부품들의 목록을 바탕으로 가능한 최대의 유용도를 가지는 구성조합으로 재구성 작업을 실시해야 한다. 이를 위해서는 가능한 모든 구성조합 별로 유용도 평가가 선행되어야 하며, 이는 다음에서 설명하는 3 단계를 거쳐 이루어진다(표 1 참조).

1) 각 부품의 유용도 평가: 동작 여부에 따라 해당 부품은 0 또는 1의 유용도 값을 가짐.
2) 서브셋의 유용도 평가: 해당 서브셋이 목표하는 기능을 위해 반드시 동작해야 하는 부품들이 모두 동작한다면, 나머지 부가적인 부품들의 유용도 값(U_{Com_1}, U_{Com_2}, …, U_{Com_k})과 서브셋의 유용도 함수(f_{Subset})에 의해 0에서 1사이의 값으로 평가됨.
3) 전체 시스템의 유용도 평가: 각 서브셋의 유용도 값(U_{Subset_1}, U_{Subset_2}, …, U_{Subset_m})과 시스템 유용도 함수(U_{System})에 의해 0에서 1사이의 값으로 결정된다.

3.4 모노레일 제어 시스템 분석사례

본 절에서는 임베디드 시스템 모델링 과정의 정확한 이해를 돕기 위해 모노레일 제어시스템을 사례로 제시한다. 모노레일은 임의의 A, B 두 지점을 왕복 운행한다고 가정한다. 모노레일 제어 시스템은 프로세서, 센서, 액츄에이터 및 소프트웨어로 이루어져 있으며, 해당 시스템의 구조도에 의하여 모노레일 제어 시스템을 구성하는 모든 부품을 정의하였다(그림 2 참조). 한편 모노레일 제어 시스템을 구성하는 전체 부품의 수는 모두 17+n(n은 Rail Sensor의 수)개이므로, 발생 가능한 모든 구성조합의 수는 2^{17+n}이다. 하지만 일부 부품의 결함 발생 시에 발생 가능한 모든 구성조합에 대해 재구성

표 1 유용도 평가의 3 단계

단계	유용도 평가의 조건	유용도 함수
각 부품의 유용도 평가	해당 부품이 동작하는 경우	$U_{Com} = 1$
	해당 부품이 동작하지 않는 경우	$U_{Com} = 0$
서브셋의 유용도 평가	목표하는 기능을 위해 반드시 동작해야 하는 부품이 모두 동작하는 경우	$U_{Subset} = f_{Subset}(U_{Com\_1}, U_{Com\_2}, \cdots, U_{Com\_k})$
	위의 조건에 맞지 않는 경우	$U_{Subset} = 0$
전체 시스템의 유용도 평가	목표하는 기능을 위해 반드시 동작해야 하는 서브셋이 모두 동작하는 경우	$U_{System} = f_{System}(U_{Subset\_1}, U_{Subset\_2}, \cdots, U_{Subset\_m})$
	위의 조건에 맞지 않는 경우	$U_{System} = 0$

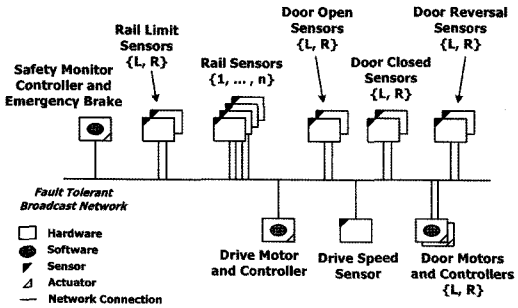

그림 2 모노레일 제어 시스템의 구조도

알고리즘을 수행하는 것은 비현실적이다.

소프트웨어 중심의 서브셋을 결정하기 위해 모노레일 제어 시스템의 소프트웨어 요소 및 각각의 기능을 우선적으로 파악해야 한다. 그 중 Drive Controller 소프트웨어는 모노레일의 원활한 주행을 위한 속도 데이타(Drive Speed Sensor)와 위치 데이타(Rail Sensors) 및 탑승자의 안전을 위한 데이타(Door Closed Sensors, Safety Control Subset)를 필요로 한다. 마찬가지로 Safety Monitor Controller 소프트웨어와 Door Controller 소프트웨어에 필요한 데이타를 파악하였으며, 각 소프트웨어의 기능을 원활히 제공하기 위한 부품들을 그룹핑한 것이 소프트웨어 중심의 서브셋이다(그림 3 참조). 이 과정에서 서브셋이 목표하는 기능을 위해 반드시 동작해야 하는 부품들의 데이타 흐름은 강한 의존성(Strong Dependence, 실선)으로 표현하고, 그렇지 않은 부품들의 데이타 흐름은 약한 의존성(Weak Dependence, 점선)으로 표현하였다.

그림 3에서 보는 바와 같이 소프트웨어 중심의 서브셋이 결정된 뒤에는 각 서브셋 별로 고려해야할 구성조합의 수를 계산해야 한다. 서브셋의 목표하는 기능을 위해 필요한 최소한의 부품들, 즉 강한 의존성 데이타 흐름을 발생시키는 부품들이 고장일 경우에는 서브셋이 목표하는 기능은 물론 전체 시스템이 곧바로 고장 상태에 도달하게 되므로, 이들의 고장은 소프트웨어 결함허용 범위에서 제외하였다. 그러므로 각 서브셋 내에서 약한 의존성 데이타 흐름을 발생시키는 부품들만이 소프트웨어 결함허용 기법의 직접적인 적용 대상이 된다. 표 2에서 서브셋 내에서 고장을 고려해야 하는 부품의 수가 n일 때 구성조합의 수가 2^n-1인 이유는 n개의 부품이 모두 정상 동작하는 경우의 구성조합을 배제하였기 때문이며, 이는 총 구성조합의 수를 계산할 때 1을 더한 이유와 동일하다(수식 (1) 참조).

Drive Control 서브셋의 Rail Sensors의 수를 4개, Left Door Control 서브셋과 Right Door Control 서브셋의 각 Door Open Sensor의 수가 1개라고 가정하면, 표 2에서 보는 바와 같이 총 18(=2^4+2)개의 구성조합이 발생한다. 그림 4는 발생 가능한 모든 구성조합의 유용

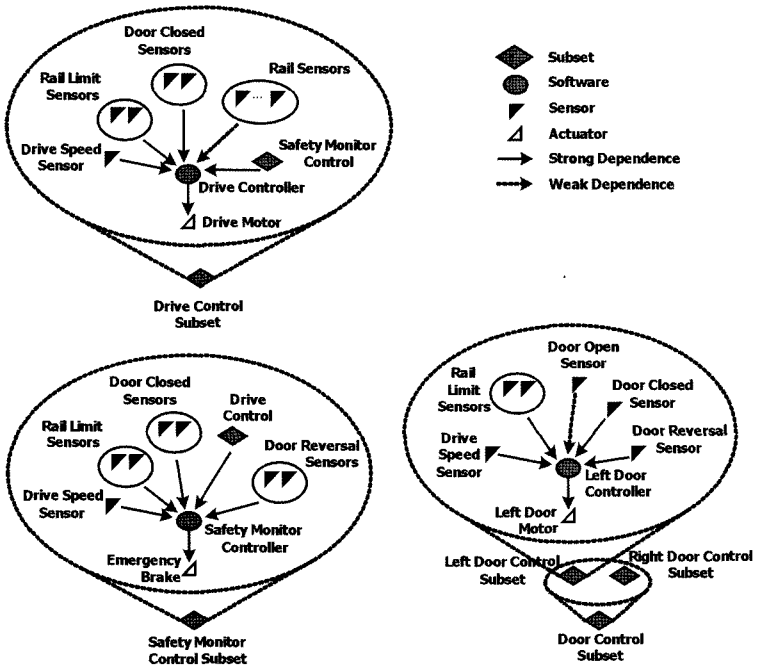

그림 3 모노레일 제어시스템의 소프트웨어 중심의 서브셋

표 2 서브셋 별 발생 가능한 구성조합의 수

구분		약한 의존성의 부품 수	구성조합의 수
서브셋	Drive Control	n	2^n-1
	Left Door Control	1	2^1-1
	Right Door Control	1	2^1-1
	Safety Monitor Control	0	0
모든 부품이 정상인 경우		-	1
합계		n+2	2^n+2

표 3 모노레일 제어 시스템의 유용도 함수

적용 대상	유용도 함수의 적용 조건	유용도 함수
Rail Sensors	적어도 하나 이상의 Rail Sensor가 동작할 경우	$U_{RS}=(U_{RS1}+\cdots+U_{RSn})/n$
	모든 Rail Sensor가 고장일 경우	$U_{RS}=0$
Drive Control	Drive Controller, Drive Motor, Drive Speed Sensor, Rail Limit Sensors, Door Closed Sensors 및 Safety Monitor Control Subset 이 모두 동작하는 경우	$U_{Drive}=0.2+0.8*U_{RS}$
	위의 조건에 맞지 않는 경우	$U_{Drive}=0$
Left(Right) Door Open Sensor	Door Open Sensor가 동작할 경우	$U_{LeftOpen}=1$
	Door Open Sensor가 고장일 경우	$U_{LeftOpen}=0$
Left(Right) Door Control	Left(Right) Door Controller, Left(Right) Door Motor, Drive Speed Sensor, Rail Limit Sensors, Left(Right) Door Closed Sensor 및 Left(Right) Door Reversal Sensor 가 모두 동작하는 경우	$U_{LeftDoor}=0.5+0.5*U_{LeftOpen}$ $U_{RightDoor}=0.5+0.5*U_{RightOpen}$
	위의 조건에 맞지 않는 경우	$U_{LeftDoor}=0$, $U_{RightDoor}=0$
Door Control	둘 중에 적어도 하나가 동작할 경우	$U_{Door}=0.5*U_{LeftDoor}+0.5*U_{RightDoor}$
	양쪽이 모두 고장일 경우	$U_{Door}=0$
Safety Monitor Control	Safety Monitor Controller, Emergency Brake Actuator, Drive Speed Sensor, Rail Limit Sensors, Door Closed Sensors, Drive Control Subset 및 Door Reversal Sensors가 모두 동작할 경우	$U_{Safety}=1$
	위의 조건에 맞지 않는 경우	$U_{Safety}=0$
System Utility	Safety Monitor Control, Drive Control, Door Control 모두 동작할 경우	$U_{System}=0.7*U_{Drive}+0.3*U_{Door}$
	위의 조건에 맞지 않는 경우	$U_{System}=0$

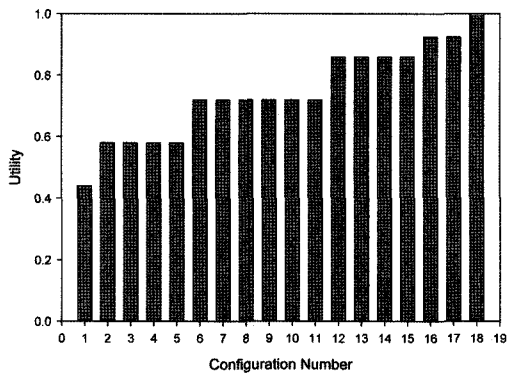

그림 4 모노레일 제어시스템의 각 구성조합별 유용도

도 평가를 수행하여 얻은 그래프이며, 모노레일 제어시스템의 유용도 함수는 표 3과 같다. 모든 Rail Sensors에서 고장이 발생한 경우가 1번 구성조합이며, 18번 구성조합은 모든 부품이 정상 동작할 경우이고, 나머지 구성조합은 Rail Sensors와 Door Open Sensors가 부분적으로 고장이 발생한 경우를 나타낸다. 예를 들어 10번 구성조합은 Rail Sensor 2와 4가 고장이 발생한 경우이며, 표 3에서 설명하는 바에 따라 U_{RS}의 값은 0.5가 된다. 같은 방법으로 각 서브셋의 유용도 값을 계산하면, Drive Control 서브셋의 유용도 값, U_{Drive}는 0.6을 가지게 되며, Door Open Sensors가 모두 정상 동작하므로 U_{Door}는 1의 값을 가지고, U_{Safety} 또한 1의 값을 가진다. 최종적으로 10번 구성조합의 전체 시스템 유용도, U_{System}의 값은 0.72가 된다.

기존의 Feature 중심의 서브셋 개념은 임베디드 시스템의 모든 부품을 서브셋으로 분할하는 기준이 명확하지 못한 것에 비해[9], 본 논문에서 제안된 소프트웨어 중심의 서브셋 개념은 포함된 소프트웨어 요소의 수만큼 서브셋이 결정되므로 임베디드 시스템의 서브셋의 수 및 구성조합의 수를 결정하는 문제가 용이하다. 더군다나 일반적으로 서브셋의 수가 늘어날수록 고려해야 할 구성조합의 수는 줄어들게 되지만 특수한 경우에는 불필요한 서브셋의 증가로 인해 소프트웨어 비용의 증가를 가져오며, 오히려 구성조합의 수가 늘어나는 경우도 있다(i.e., 1개의 하드웨어 부품과 1개의 소프트웨어

가 서브셋을 구성했을 경우). 때문에 본 논문에서 제안된 소프트웨어 중심의 서브셋 개념을 사용하여 임베디드 시스템의 재구성 작업을 위한 구성조합의 수를 도출하는 것이 효과적이라고 판단된다.

4. 실험 및 성능 분석

임베디드 시스템의 결함 발생 시 재구성 작업을 통해 시스템의 신인도 및 성능을 확보할 수 있음을 검증하기 위해, 레고 마인드스톰(Lego Mindstorms)을 이용한 Java 프로그래밍 기반의 AGV(Automatic Guided Vehicle)를 제작하였다. 그림 6에서 보듯이 구현된 AGV에는 빛 센서 2개와 구동 모터 2개 및 소프트웨어 요소로써 RCX 2.0 1개가 사용되었다. AGV 시스템의 소프트웨어 요소가 1개이므로, 전체 시스템에서 소프트웨어 서브셋 또한 1개이다. 이는 3장에서 설명한 구성조합의 수를 줄일 수 없다는 뜻이며, 때문에 본 실험에서는 결함 발생에 따른 시스템 재구성 작업만을 검증하였다.

단일 소프트웨어 서브셋으로 구성된 AGV 프로토타입의 구성요소들 중 강한 의존성 데이타 흐름을 발생하는 부품은 구동 모터 2개와 소프트웨어 요소인 RCX 2.0 1개이며, 나머지 빛 센서 2개는 고장이 발생하더라도 시스템의 기본적인 목적(i.e. 트랙을 따라 주행하는 것)을 제공할 수 있으므로, 약한 의존성 부품이라고 정의하였다. 따라서 고려해야 할 구성조합은 모두 $4(=2^2)$ 가지이며, 각 구성조합은 빛 센서 2개의 고장 발생 여부에 따라 결정된다. 각각의 구성조합별 재구성 동작은 다음과 같다.

- 빛 센서 2개 모두 정상 동작 시(구성조합 1번): 빛 센서 2개의 입력 값에 따라 2개의 구동 모터의 주행 방향을 제어하여, AGV 모델이 라인을 따라 원활히 주행 가능하다(정상).
- 빛 센서 1개만 정상 동작 시(구성조합 2, 3번): 빛 센서의 고장 발생을 감지하게 되면 시스템 재구성 작업에 의해 나머지 정상 동작하는 빛 센서 1개가 트랙의 검은 색 라인을 따라 주행토록 하는 알고리즘으로 전환된다(기능적 대안).
- 빛 센서가 모두 고장 발생 시(구성조합 4번): 모든 빛 센서의 고장을 감지하게 되면 목표하는 주행이 불가능하다고 판단하여 AGV 모델을 정지시킨다(시스템 실패).

AGV 시스템의 재구성 작업 전, 후의 성능 평가를 위하여 구동 모터의 초기 속력을 좌, 우 모두 2m/s로 설정하였으며, 검은 색 라인으로 표시된 트랙을 주행하게 한 실험에서 각 구성조합에 따른 코스 이탈율과 Lap Time을 측정한 결과는 다음 표 4와 같다. 표 4에서 보는 바와 같이 AGV의 일부 부품에 결함이 발생한 경우 성능은 감퇴되더라도 목표하는 기능을 제공하는 것을 확인하였다.

표 4 정상 동작하는 빛 센서의 수에 따른 AGV의 성능 비교

구 분	빛 센서 2개가 동작 시	빛 센서 1개가 동작 시
평균 속력(m/s)	1.04	0.61
트랙 이탈율(%)	0.0	0.0

5. 결 론

본 연구에서는 여분(Redundancy)을 사용하지 않는 임베디드 시스템의 신인도(Dependability)를 향상시키기 위해, 소프트웨어 결함허용 기법을 이용하여 재구성 가능한 임베디드 시스템의 유용도(Utility) 분석을 수행하였다. 이를 위해 임베디드 시스템의 디자인 단계에서 소프트웨어 중심의 서브셋의 결정 및 구성조합(Configuration)의 수를 줄이는 방법을 제시하였다. 본 연구를 통해 임베디드 시스템의 일부 부품이 고장 발생 시에도 다소 성능감퇴된 운영 상태로의 천이가 가능하였으며, 정량적으로는 임베디드 시스템의 평균 유용도의 향상을

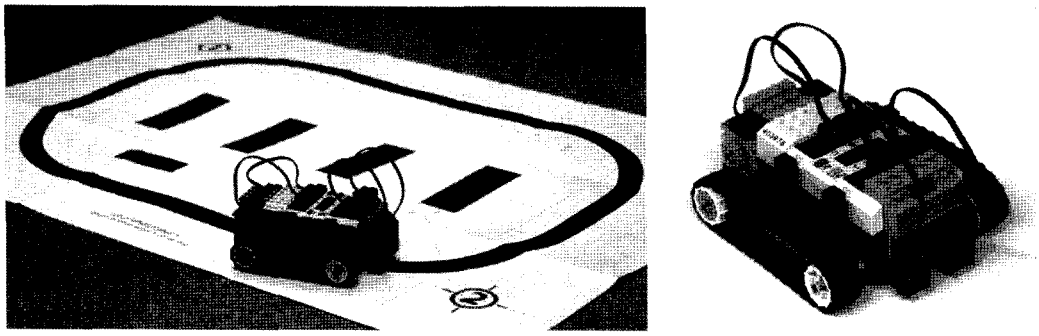

그림 6 레고 마인드스톰을 이용하여 제작한 AGV

가져왔다. 추후에는 임베디드 시스템의 평균 유용도에 영향을 미치는 다양한 요인을 분석하여, 각 부품의 고장 확률을 이용한 구성조합의 발생 확률 정의에 관한 연구 및 임베디드 시스템의 평균 유용도 최적화에 대한 연구를 수행할 예정이다.

참 고 문 헌

[1] J. Knight and K. Sullivan, "On the Definition of Survivability," University of Virginia, Department of Computer Science, *Technical Report* CS-TR-33-00, 2000.

[2] -, "고신뢰성 차량 임베디드 컴퓨팅 시스템의 백업 최소화 방안", 한국신뢰성학회 2005 학술발표대회 논문집, pp 295-301, June 2005.

[3] B. Randell, "System Structure for Software Fault Tolerance," IEEE Transactions on Software Engineering, Vol. SE-1, No. 2, pp. 220-232, June 1975.

[4] M. Bodson, J. Lehoczky, et al., "Control Reconfiguration in the Presence of Software Failures," Proceedings of the 32nd IEEE Conference on Decision and Control, San Antonio, TX, USA, pp. 2284-2289, Dec. 1993.

[5] A. Avizienis, et al., "Fundamental Concepts of Dependability," Research Report N01145, LAAS-CNRS, Apr. 2001.

[6] J. Meyer, "On Evaluating the Performability of Degradable Computing Systems," The Eighth Annual International Conference on Fault-Tolerant Computing (FTCS-8), Toulouse, France, pp. 44-49, June 1978.

[7] P. Ramanathan, "Graceful Degradation in Real-Time Control Applications Using (m, k)-firm Guarantee," 27th Annual international Conferences on Fault-Tolerant Computing, Seattle, WA, USA, pp. 132-141, June 1997.

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

[9] C. Shelton and P. Koopman, "Improving System Dependability with Functional Alternatives," 2004 International Conference on Dependability Systems and Networks, pp. 295-304, July 2004.

[10] C. Shelton, "Scalable Graceful Degradation for Distributed Embedded Systems," Ph.D. dissertation, Dept. of Electrical And Computer Engineering, Carnegie Mellon University, Aug. 2003.

[11] C. Shelton and P. Koopman, "Using Architectural Properties to Model and Measure Graceful Degradation," in Architecting Dependable Systems, LNCS 2677, pp. 267-289, Berlin, 2003.

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

강 민 구

2007년~현재 아주대학교 산업정보시스템공학부. 관심분야는 Embedded System, Fault Tolerant Computing

박 기 진

1989년 한양대학교 산업공학과(공학사) 1991년 포항공과대학교 산업공학과(공학석사). 1991년~1997년 삼성종합기술원, 삼성전자(주) 소프트웨어센터 선임연구원. 1997년~2001년 아주대학교 컴퓨터공학과(공학박사). 2001년~2002년 한국전자통신연구원 네트워크장비시험센터 선임연구원. 2002년~2004년 안양대학교 컴퓨터학과 전임강사. 2004년~현재 아주대학교 산업정보시스템공학부 조교수. 관심분야는 Dependable Embedded Computing, Safety-Critical Computing, Vehicle Embedded Network