비기능적 요구사항 관점의 소형 무인지상차량 성능 향상 연구

A Study on Efficiency Improvement for SUGV
with a Practical View Point of Non-Functional Requirements

서 진 원* 김 영 철* 김 장 한*
Seo, Jin-Won Kim, R. Young-Chul Kim, Jang-Han

ABSTRACT

In the next near future, the human would like to use the small unmanned ground vehicle(SUGV) on the diverse fields. Specially the world of today is tried to apply with operating the task on very difficult working environments such as some dangerous or unreachable area[5]. To work this task, this vehicle should be guaranteed with the high level of reliability, safety, and performance. In this paper, we propose to focus on not only the functional requirements, but also the non-functional requirements based on software architecture at the design stage for developing the embedded system. Through focusing on the non-functional requirements on this software architecture, we can obtain the design goal of the target system and also show the enhancement of reliability, safety and performance with 'Vtune' performance analysis tool.

주요기술용어(주제어) : Software Architecture(소프트웨어 아키텍처), Non-functional Requirement(비기능적 요구 사항), SUGV(소형무인지상차량), 'VTune' Performance Analysis Tool(성능 분석기)

1. 머리말

최근 미군은 팩봇(PackBot)[1], 탈론(TALON)[2] 스워드 등 1,000여대의 다양한 군사용 로봇을 이라크와 아프가니스탄의 치안 유지에 투입하고 있다. 국내는 국방과학연구소에서 자율주행로봇에 대한 연구가 활발히 진행 중이다. 미군은 미래전투체제(FCS : Future Combat System)[3]로 군의 무기를 첨단화를 추진 중이다. 그중에 소형무인지상차량(SUGV : Small Unmanned Ground Vehicle)[4~7]은 도시지형, 터널, 하수구, 동굴 등에서 군사작전을 수행할 수 있는 소형, 경량, 휴대용 무인지상차량(UGV)이다. SUGV는 인력이 집중적이고 위험성이 높은 일(도시지능, 감시 및 정찰(ISR) 임무, 화학/독성화학제품(TIC)/독성산업물자(TIM), 정찰 등)과 같은 군인이 직접 위험한 곳에서 몸을 드러내고 작업을 수행하기 어려운 분야에 적용되고 있다[5]. 이러한 작업을 수행하기 위해서는 높은 신뢰성 및 안전성이 보장되어야 한다.

일반적인 시스템 개발을 위해서는 기능적인 요소들이 강조된다. 하지만 특수한 작업을 하는 시스템은 비기능적인 요구사항이 더 크게 작용 할 수 있다[8,10]. 특히 전장에서 위험한 환경에 SUGV시스템의 비기능적

† 2008년 4월 15일 접수~2008년 6월 20일 게재승인
* 홍익대학교(Hongik University)
주저자 이메일 : suh@selab.hongik.ac.kr

요구사항이 더욱 인명이나 재산 손실등의 직접적인 피해를 막을 수 있을 것이다. 결국 시스템의 기능적 면도 중요하지만 신뢰성, 성능, 안전성, 모듈성 등과 같은 비기능적 요구사항도 중요하다. 비기능적인 요구사항도 시스템 개발단계에 추가하여 시스템을 품질을 향상 시킬 수 있다. 이는 개발하고자 하는 타겟 시스템의 설계 목적(Design Goal)에 초점을 두고 개발할 수 있기 때문이다. 또한 아키텍처를 관점으로 시스템에 접근하여 설계에 대한 검증을 하면 안전성을 높일 수 있다.

본 논문은 소프트웨어 아키텍처 기반의 임베디드 시스템 설계에 있어서 비기능적인 요소를 설계단계에서 적용과 아키텍처 단계에서 비기능적 요구사항에 따른 시스템간의 차이를 통해 성능 향상과 안전성 및 신뢰성을 보다 효과적으로 높이는 방법을 제안한다.

논문의 구성은 다음과 같이 되어 있다. 2장에서는 관련된 소프트웨어 아키텍처를, 3장에서는 아키텍처의 비기능적 요구에 관하여 언급한다. 4장에서는 소프트웨어 아키텍처 기반의 임베디드 시스템의 적용사례를, 마지막으로 5장은 앞의 내용을 토대로 결론을 내 놓았다.

2. 관련 연구

가. 소프트웨어 아키텍처

소프트웨어 아키텍처란 개발하려고 하는 소프트웨어 시스템에 대한 핵심적인 기능적, 비기능적 요구 사항을 담고 있는 추상화 된 구조이다. 활용 방안에는 소프트웨어가 구현되기 전에 아키텍처가 제공하는 다양한 모형으로부터 여러 품질 특성을 추론하고 이를 바탕으로 품질을 향상시킬 수 있다[8].

본 논문에서는 구현할 소프트웨어의 기능성 외에 비기능성을 효과적으로 표현하고자 한다.

비기능적 요구사항들의 요건들 중 사용자 요구에서 사람의 생명과 가장 밀접하게 관련 있는 안전성과 테스크들을 정확하게 수행할 수 있도록 하는 신뢰성, 사용자가 전문적인 지식을 가지고 있어야 하는가를 언급하는 사용성, 타겟 시스템의 하드웨어 플랫폼이 제약점을 가지고 있는가를 제시하는 구현성, 시스템이 작동하는 동안 다른 모듈이 응답할 수 있는 지를 수행성으로 판단하여 소프트웨어 아키텍처 기반의 임베디드 시스템을 개발하려 노력하고자 한다.

나. 기능과 비기능 요구사항

요구사항은 크게 기능 요구사항과 비기능 요구사항으로 구분될 수 있다. 기능 요구사항이란 프로젝트에서 요구되는 기능인 이동, 좌턴, 우턴, 탐색 등과 같은 것들이다. 비기능 요구사항은 시스템의 속성(Properties) 및 제약 사항(Constraints)과 같이 시스템의 아키텍쳐 설계와 품질지표를 도출하는데 필요한 사용자 요구사항이다.

[표 1] 기능적 요구사항과 비기능적 요구사항 비교

기능적 요구사항	비기능적요구사항
- 시스템의 개발 범위 - 시스템의 목적 - 시스템이 제공해야 하는 기능들 - 시스템의 사용 방법	- **성능** : 응답속도, 시간당 데이터 처리량 등 - **신뢰성** : 데이터의 무결성 또는 정보처리의 정확성 등 - **보안성** : 시스템에 대한 비권한자의 접근과 자료의 유출 방지 기능 - **운영 편의성** : 장비, 유지보수 방법 등 운영 및 유지보수 환경

표 1은 기능적 요구사항과 비기능적 요구사항 비교이다. 일반적으로 비기능 요구사항은 소프트웨어의 품질 속성에 대한 내용이다. 품질은 주관적인 특성을 가지고 있다. 따라서 사용자가 직접적으로 품질 기준을 정하는 경우는 드물다. 프로젝트 팀은 사용자의 요구사항과 더불어 시스템 분석가로서의 역할을 통해 비기능적 요구사항에 대한 정량적인 자료를 만들어 내야 한다.

비기능 요구사항은 시스템 설계뿐만 아니라 인수조건의 기본이 되는 사용자 요구사항으로, 기능적 요구사항과 함께 시스템의 성패에 중요한 역할을 하므로 정량적이고 검증 가능하도록 기술하고 합의해야 한다. 비기능 요구사항은 개별적인 업무 특성 외에도 전체

적인 시스템의 분석을 통해 도출될 수 있다.

3. 소프트웨어 아키텍쳐의 비기능적 요구

가. 요구 설계 체크리스트

본 논문에서는 개발단계 중 아키텍쳐 단계를 고려하여 비기능적 요구사항들에 따른 각 모듈마다의 check list를 설정하였다. check list의 구성은 처음에는 Use Case name을 설정하여 Use Case를 식별하게 되며 participating actors를 설정하여 이벤트에 영향을 주는 액터들을 식별하여 정의한다. Flow of event는 이벤트간의 사건을 나열하여 시스템 안에 있는 상호작용들을 구별할 수 있게 된다. Entry condition에서는 처음에 이벤트가 시작하는 조건을 설명하여 정의하고, Exit condition에서는 시작하는 이벤트가 종료되어버리는 조건을 나타내도록 정의한다. 마지막으로 Non-functional requirement에선 모듈들이 기능적인 행위 외에 다른 상태를 고려 할 수 있도록 나열하여 설명하여 정의한다. 표 2는 비기능적 요구 check list 기본 형태이다[10].

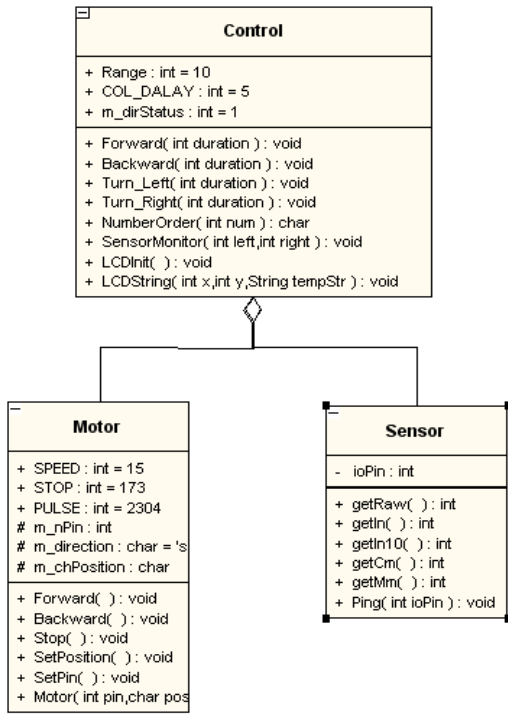
[표 2] 요구 설계 체크리스트

use case name		
participating actor		
flow of events		
entry condition		
exit condition		
Non-functional requirement		

나. 비기능적 요구사항 기반 임베디드 S/W 개발

기존의 방법[10,13]을 참조하여 SUGV 시스템을 기능적인 요구사항 기반으로 설계하면 그림 1과 같다.

본 시스템의 요구분석에서 시나리오에 따른 각 모듈에 관한 내용을 클래스 다이어그램으로 설계 한다. 클래스 다이어그램의 기능적 요구사항에서는 센서모듈(Sensor Module), 제어모듈(Control Module), 모터모듈(Motor Module)을 고려한다.



[그림 1] 기능적인 요소를 적용한 클래스 다이어그램

로봇이 기본적으로 진행할 수 있는 행동인 기능적 요구는 앞으로 전진, 왼쪽 센서에 물체를 감지하면 우회하고 오른쪽 센서에 물체를 감지하면 왼쪽으로 비켜간다. 그리고 두 개의 센서가 동시에 감지했을 때 후진을 하고 난 뒤 왼쪽으로 비켜간다는 4가지 시나리오가 있다.

비기능적 요구사항 중 안전성, 신뢰성, 성능, 사용, 구현을 고려하여 표 3~5와 같이 각각 모듈의 조건들을 만들 수 있다.

여기서의 비기능적 요구는 각 모듈마다의 제약 조건들 중 센서모듈에서의 신뢰성은 센서의 감지거리가 무한하지 않고 일정한 거리를 유지하고, 수행성으로서는 SUGV의 동작 중이나 정지하고 있을 때에도 센서는 항상 감지를 하고 있어야 한다는 제약을 준다. 센서의 또 하나 제약으로는 SUGV의 회전 동작은 처음과 나중의 사이각을 약 30도를 유지해야 한다.

[표 3] 센서 모듈 체크 리스트

유스케이스 이름	Sensing	
참여 액터	①감지된 센서를 제어모듈에게 통신하여 초음파 센서에 의해 장애물이 감지된다. ②감지된 센서 정보를 제어모듈에게 보내어진다.	
이벤트들의 흐름	SUGV가 작동 중 초음파 센서가 벽이나 장애물에 왼쪽 센서가 먼저 닿았을 경우는 우회전하여 장애물을 피하고 오른쪽 센서가 먼저 닿았을 경우는 좌회전하여 장애물을 피하고 두 개의 센서가 동시에 닿았을 경우에는 후진한 후 우회하여 장애물을 피해간다.	
입구 조건	장애물이 초음파 센서에 감지 당했을 때	
출구 조건	트랙을 한 바퀴 돌고 난 다음에는 완전히 정지한다.	
비기능적 요구사항	신뢰성	센서의 감지거리가 무한하지 않고 일정한 거리를 유지한다.
	성능	SUGV의 동작 중이나 정지하고 있을 때에도 sensor는 항상 감지를 하고 있어야 한다.
	정확성	SUGV의 회전동작은 처음과 나중의 사이각을 약 30도를 유지해야 한다.

[표 4] 모터 모듈 체크 리스트

유스케이스 이름	Moving	
참여 액터	①제어모듈에서 보낸 정보를 모터모듈은 받아들여 전기적인 작동으로 변환한다. ②전기적인 작동에 의해 바퀴가 앞으로 가거나 뒤로 가기를 작동한다.	
이벤트들의 흐름	제어모듈에서 보내온 정보를 모터모듈이 받아 좌회전 할 때는 모터는 역방향으로 회전한 뒤 왼쪽 모터는 정지하고 오른쪽 모터가 회전하여 움직인다. 우회전 할 때도 모터가 역방향으로 회전한 뒤 오른쪽 모터는 정지하고 왼쪽 모터가 회전하여 움직인다.	
입구 조건	제어모듈에서 정보를 보내왔을 때	
출구 조건	트랙을 한 바퀴 돌고 난 다음에는 완전히 정지한다.	
비기능적 요구사항	안전성	SUGV위에 구슬을 올려놓으면 떨어지지 않도록 모터의 속도를 점진적으로 변화한다.
	구현성	센서와 바퀴, 모터, cpu칩, text lcd들이 전부 모듈화 되어 있어 쉽게 분해가 가능하고 조립이 가능하다.

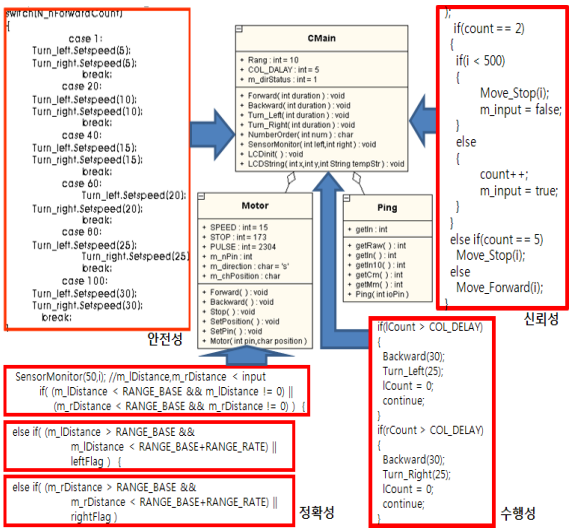
제어모듈에서의 비기능적 요구로서 신뢰성은 센서와 프로세서, 모터는 물리적으로 명확히 연결되어 있어야 하고 승차지점에선 2초간 정지를 하고 정차지점에선 완전히 정지를 하는 제약이다. 수행성으로서는 센서에서 받은 입력을 제어는 반드시 모터에게 물리적인 신호로 보내고, 사용성으로는 센서로부터 입력받으면 제어모듈의 지시명령이 바로 이루어져 모터모듈이 수행되어야 한다.

마지막으로 구현성로는 사용자는 대표적인 객체지향 언어인 JAVA를 통한 센서와 모터에 대한 컨트롤이 구현이 용이하다. 모터모듈의 비기능적 요구인 안전성은 SUGV위에 구슬을 올려놓으면 떨어지지 않도록 모터의 속도를 점진적으로 변화한다는 제약이고 구현성으로는 센서와 바퀴, 모터, cpu칩, text lcd들이 전부 모듈화 되어 있어 쉽게 분해가 가능하고 조립이 가능하다는 제약을 정의할 수 있다.

[표 5] 제어 모듈 체크 리스트

유스케이스 이름	Controlling	
참여 액터	①센서모듈로부터 보내온 정보를 제어모듈이 받아 정보 수집을 한다. ②수집한 정보를 제어모듈은 모터모듈에게 다시 보내어진다.	
이벤트들의 흐름	센서모듈에서 보내온 정보를 제어모듈로부터 수집하여 모터모듈에게 제어하는 정보를 보낸다.	
입구 조건	센서모듈에서 정보를 보내 왔을 때	
출구 조건	트랙을 한 바퀴 돌고 난 다음에는 완전히 정지한다.	
비기능적 요구사항	신뢰성	①센서와 프로세서, 모터는 물리적으로 명확히 연결되어 있어야 한다. ②승차지점에선 2초간 정지를 하고 정차지점에선 완전히 정지를 하여야 한다.
	성능	센서에 받은 입력을 제어모듈은 반드시 모터모듈에게 물리적인 신호로 보내 줘야 한다.
	사용성	센서로부터 입력 받으면 제어모듈의 지시명령이 바로 이루어져 모터모듈이 수행되어진다.
	구현성	사용자는 대표적인 객체지향 언어인 JAVA를 통한 센서와 모터에 대한 컨트롤이 구현이 용이하다.

그림 2는 각 모듈의 체크리스트인 표 3~5를 바탕으로 하여 비기능적 요구를 적용한 것이다. 센서모듈에서는 정확성, 수행성, 성능을 적용하였다. 제어모듈에서는 신뢰성, 성능, 사용성, 구현성을 적용하였다. 모터모듈에서는 안전성과 구현성을 적용한다.



[그림 2] 비기능 요구사항의 추가

4. 적용사례

가. SUGV

본 연구에 사용된 SUGV은 기본적인 마이크로 제어 장치 및 관련 개발 제품을 다루는 Parallx.Inc에서 생산한 제품들을 기반으로 각 모듈을 조합하여 일반적인 초음파 감지 센서 SUGV제품으로 제작하였다. 대표적인 객체지향 언어인 Java를 통한 동작제어가 가능한 SUGV 로봇 시스템에 전송된 파일이 메모리에 저장이 되고 작성된 제어 프로그램의 동작 원리에 맞게 작동을 하게 된다[11].

나. 설계에의 적용

기본적으로 SUGV 로봇은 제어에 필요한 소프트웨어 시스템에 Polling 방식으로 프로그래밍을 하여 비기능적 요구사항인 안전성, 신뢰성, 성능성, 사용성, 구현성에 대해서만 우선으로 적용하였다.

기능적 요구사항의 구현으로서는 제어클래스로서 Forward(), Backward(), Turn_Right(), Turn_Left()을 구현하여 전진과 왼쪽에 장애물이 있을 때는 우회하고, 오른쪽에 장애물이 있을 때는 왼쪽으로 비켜가고 정면에 장애물이 있을 때는 후진을 한 뒤 우회하여 장애물을 피하는 방법을 채택하여 시스템을 구현하였다.

본 논문에서 수행된 비기능적 요구사항의 요건으로서는 안전성이다. 가장 기본적인 Polling 방식을 채택하여 프로그래밍이다. 시스템에서 구슬을 사람으로

```
public void Forward()
{
  if(m_cnPosition == '1')
  {
     CPU.pulseOut(STOP - SPEED,m_nPin);
     switch(N_nForwardCount)
     {
        case 1:
Turn_left.Setspeed(5);
Turn_right.Setspeed(5);
           break;
        case 20:
Turn_left.Setspeed(10);
Turn_right.Setspeed(10);
           break;
        case 40:
Turn_left.Setspeed(15);
Turn_right.Setspeed(15);
           break;
        case 60:
           Turn_left.Setspeed(20);
Turn_right.Setspeed(20);
           break;
        case 80:
Turn_left.Setspeed(25);
           Turn_right.Setspeed(25);
           break;
        case 100:
Turn_left.Setspeed(30);
Turn_right.Setspeed(30);
     break;
     }
  }
}
```

[그림 3] 제어 클래스의 비기능적인 안전성

간주하여 구슬을 떨어지면 안전성이 만족하지 못한 실험이다. 안전성을 만족하기 위해서는 속도의 변화가 점진적으로 증가하거나 감소하고 주행 중에는 일정한 속도를 유지하는 것이 중요하다. 그 과정 중에서도 속도의 급변화가 이루어지는 부분은 바로 처음 출발할 때와 정지할 때의 속도이다. 정지해 있다가 출발을 할 때 순간적인 펄스를 통해 급출발을 하게 되는데 이것을 제어하기 위해서는 속도를 서서히 올릴 수 있게 해야 한다. 그리고 주행 중의 로봇을 정지시킬 때에는 급제동을 하게 되는데 이것을 제어하기 위해서는 속도를 서서히 줄이는 방법을 채택해야 한다.

비기능적 요구사항의 안전성을 제어하는 방법으로서는 그림 3의 제어클래스 안에 있는 Foward()메소드에서 스위치 케이스(Switch-Case)문을 사용하여 6단계의 속도로 구분하게 되고 점진적으로 속도가 증가하도록 제어하였다. 또 다른 비기능적 요구사항의 신뢰성에서는 로봇이 트랙을 정확하게 목적지까지 수행할 수 있는가 없는가에 중점을 두어 프로그램을 설계하였다. 그 문제에 관해서는 신뢰성을 만족하기 위한 제약 조건으로서 정지해야 할 장소, 바로 승차지점과 정차지점에서의 정지하는 것이다.

제어방법으로서는 그림 4인 제어클래스의 신뢰성을 만족시키기 위해 추가된 on()메소드에서 카운터(Count)변수가 2가 될 경우와 5가 될 경우에 정확히 Move_stop()메소드를 사용하여 정지를 한다. 단 승차지점에서는 5초간 정지 정차지점에서는 완전 정지를 한다.

```
if(!m_left && !m_right)
{
  Move_Backward(i);
  if(countCheck)
  {
     count++;
     countCheck=false;
  }
}
else
{
  System.out.println(count);
  if(count == 2)
  {
     if(i < 500)
     {
        Move_Stop(i);
        m_input = false;
     }
     else
     {
        count++;
        m_input = true;
     }
  }
  else if(count == 5)
     Move_Stop(i);
  else
     Move_Forward(i);
}
```

[그림 4] 제어클래스의 비기능적인 신뢰성

```
if(lCount > COL_DELAY)
{
  Backward(30);
  Turn_Left(25);
  lCount = 0;
  continue;
}
if(rCount > COL_DELAY)
{
  Backward(30);
  Turn_Right(25);
  lCount = 0;
  continue;
}
if(rDistance < RANGE && lDistance < RANGE)
{
  Backward(30);
  Turn_Left(25);
}
else if(rDistance < RANGE)
{
  Backward(10);
  Turn_Left(25);
}
else
  Forward(10);
oldLDistance = lDistance;
oldRDistance = rDistance;
```

[그림 5] 제어클래스의 비기능적인 수행성

그림 5는 비기능적 요구사항의 수행성을 구현해 보면 사각의 코너에 진입하여 좌우로 여러 번 회전을 하였으나 쉽게 빠져 나오지 못하는 상황에 COL_DELAY 상수변수를 지정하여 rCount변수와 lCount변수를 COL_DELAY 변수에 서로 비교하여 이 변수보다 클 경우에는 무조건 뒤로 후진을 한 후 왼쪽방향으로 진행하여 빠져 나올 수 있게 구현함으로서 수행성을 더 만족할 수 있게 제약조건 제시한다.

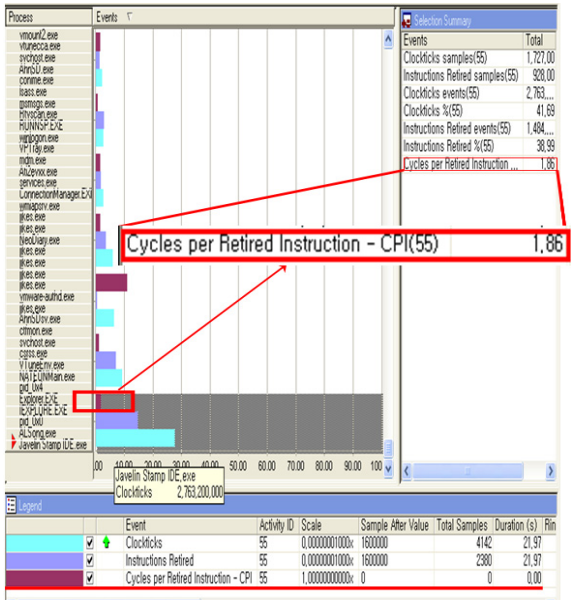
비기능적인 정확성을 구현하면 초음파 센서(Ultrasonic)는 최소 2cm에서 최대 3m터까지 측정이 가능한 센서로, 본 논문의 실험에선 측정거리를 10cm에서 물체를 감지할 수 있을 정도로 제약조건을 만들어 주었다. 그 이상의 거리에서는 실험에 불필요함으로서 비기능적인 정확성을 만족하는 시스템을 확인하였다. 그림 6은 제어클래스에 대한 비기능성 중 정확성에 관한 것이다.

```
for(int i=0; ;i++)
    { SensorMonitor(50,i);
      if( (m_lDistance < RANGE_BASE &&
           m_lDistance != 0) ||
          (m_rDistance < RANGE_BASE &&
          m_rDistance != 0) )
      {   MoveStop(100,i);
          duration = 0;
          leftFlag = false;
          rightFlag = false;        }
      else if( (m_lDistance > RANGE_BASE &&
                m_lDistance <
                RANGE_BASE+RANGE_RATE) ||
                leftFlag ) {
               if(!leftFlag)
               { duration = 0;
                 leftFlag = true;
                 rightFlag = false;       }
               else
               { duration++;    }
               LowSpeed();
               TurnLeft(100,i); }
      else if( (m_rDistance > RANGE_BASE &&
                m_rDistance <
               RANGE_BASE+RANGE_RATE) ||
                rightFlag )
        {
```

[그림 6] 제어클래스의 비기능적인 정확성

5. 성능 평가

최종적으로 논문에서는 비기능적 요구를 적용한 코드와 적용하기 전의 코드를 성능비교 분석을 위해 인텔(intel)사의 성능분석도구인 'VTune' Performance Analyzer를 임베디드 소프트웨어 측정하는데 시도하였다. 성능비교의 핵심은 핫스팟 분석으로 여러 수준의 시간 및 이벤트 샘플링 결과를 보여줌으로서 특정 부분의 리소스 사용량에 따른 성능향상을 보았다[9].

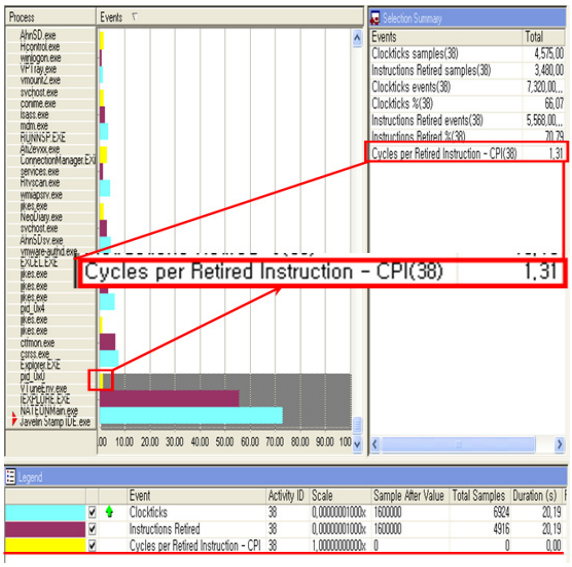


[그림 7] 기능적 요구사항 코드의 성능 측정

SUGV 시스템에서의 코드 수정 전과 코드 수정 후 이벤트 샘플링에서의 CPI(Cycle per Retired Instruction)비율을 통해 성능분석을 보면 Good으로 판단되는 CPI는 0.75정도이고 Poor의 수치는 4정도이다. 여기서 CPI란 그림 7, 8에서 세 개의 Bar중 제일 하단에 있는 그래프로서 이벤트 샘플링 기반으로 성능 조정하여 결정할 수 있도록 체크하는 첫 번째 비율이다. 각 모듈안에서의 CPU사용 시간에 대한 비율을 나타낸 것이다. 처음에는 샘플링 데이터 값은 1.86이고 수정 후 샘플링 데이터 값은 1.31값으로서 수치적으로 성능향상을 확인할 수 있다.

좋은 CPI 수치는 코드가 최적으로 수행하는 것을

의미하고 나쁜 CPI 수치는 선택된 코드가 프로세서 안에서 좀 더 효율적으로 수행하기 위해 수정하고 개선되어야할 기회가 많아져 최적화 코드가 되고 또한 선택된 코드는 스트링 명령문이나 마이크로 코드(Micro-code)를 포함한다[12].



[그림 8] 비기능 요구 추가된 코드 성능 측정

6. 맺음말

소형무인지상차량은 군인이 직접 위험한 곳에서 몸을 드러내고 작업을 수행하기 어려운 분야에 적용하고 있다[5]. 이러한 작업을 수행하기 위해서는 높은 신뢰성, 안전성이 보장되어야 한다. 본 논문에서는 소프트웨어 아키텍처 기반의 임베디드 시스템 설계에 있어서 비기능적인 요소를 설계단계에서 적용하였다. 이를 통해 아키텍처 단계에서 비기능적 요구사항에 따른 시스템의 성능 향상과 안전성 및 신뢰성이 높아진 것을 확인할 수 있었다. 그리고 성능 측정 도구를 사용하여 원시코드와 변경 후 코드를 가지고 비교하여 CPI 수치를 측정한 후 Good CPI와 Poor CPI로 구별할 수 있게 되었고 기능적 코드보다는 비기능적 코드를 통한 임베디드 S/W개발이 성능이 향상됨을 알 수 있었다.

향후 과제로서는 개발 중인 설계 자동화 도구에 비기능적인 요소도 고려할 수 있도록 적용이 필요하다. 또한 코드상의 적용뿐만 아니라 모델 단계에서의 설계 검증에 대한 연구가 진행 중이다.

후 기

이 논문은 2006학년도 홍익대학교 학술연구진흥비에 의하여 지원되었습니다.

참 고 문 헌

[1] PackBot, http://www.irobot.com/sp.cfm?pageid=109

[2] TALON, http://www.foster-miller.com/lemming.htm

[3] Future Combat System, http://www.fcs.army.mil/

[4] Future Combat System, Small Unmanned Ground Vehicle(SUGV), http://army.mil/fcs/sugv.html/

[5] 진태석, “미국의 국방로봇 최신 동향- 전투체계 및 무인차량 중심으로,”주간기술동향, 통권 1304호, 2007-07-11.

[6] 김우열, 김영철, “A Study on Modeling Heterogeneous Embedded S/W Components based on Model Driven Architecture with Extended xUML”, KIPS, Vol. 14-D, No. 1, 2007. 2.

[7] 손현승, 김우열, 김영철, “이종 소형 무인 지상 차량 개발을 위한 MDA 기반 자동화 방법 연구,” 한국소프트웨어공학회, Vol. 10, No. 1, 417~422, 2008. 2.

[8] Mary Shaw, David Garlan “Software Architecture, Perspectives on an Emerging Discipline”, Prentice Hall, pp. 43~51.

[9] http://www.intel.com/cd/software/products/asmo-na/eng/compilers/220001.htm

[10] Bernd Bruegge, Allen H. Dutoit “International edition Object-Oriented Software Engineering : Using UML, Patterns, and Java”, pp. 121~170.
[11] PARALLAX.INC, “Robotics!, Student workbook version 1.5”, p. 42
[12] VTune(TM) Performance Environment Help, CPI(Cycles per Retired Instruction).
[13] 김우열, 김영철, “Adapting Model Driven Architecture for Modeling Heterogeneous Embedded S/W Components”, ICHIT2006, Vol. 2, 2006. 11.