DOI: 10.3745/KIPSTD.2008.15-D.4.505

그래픽 언어를 이용한 임베디드 시스템의 단일 요구사항 모델링

오 정 섭[†] · 이 홍 석[††] · 박 현 상[†††] · 김 장 복[†††] · 최 경 희[††††] · 정 기 현[†††††]

요 약

임베디드 시스템에 대한 RBT(Requirement-Based Testing)를 수행하기 위하여 정확한 요구사항 명세서가 존재해야 한다. 그러나 고객이 자연어로 작성하는 요구사항은 모호성, 부정확성, 불일치성을 가지고 있다. 이를 해결하기 위해서 UML, Simulink등의 모델링 언어를 이용하여 요구사항을 다시 모델링 하지만, 이 과정에서 요구사항을 use-case단위로 조합하고 재해석하는 단점이 생겨나게 되었다. 본 논문에서는 임베디드 시스템에 대한 고객의 요구사항을 use-case단위의 조합이나 재해석이 필요 없이 곧바로 모델로 표현할 수 있는 그래픽 언어를 이용한 1:1 요구사항 모델링 기법을 소개한다. 제안한 방법은 1) 임베디드 시스템의 요구사항을 자연어가 아닌 의미가 분명한 그래픽 언어를 이용하여 표현하고, 2) 하나의 요구사항을 하나의 그래픽 모델로 표현한다. 또한 제안한 방법은 시스템의 입출력을 기반으로 "what-to-do"만을 기술하기 때문에 상위레벨의 요구사항이나 하위레벨의 요구사항 모두에 적용할 수 있는 기법이다. 이 기법이 적용된 REED라는 도구를 통하여 실제 프로젝트에 적용한 예를 살펴본다.

키워드 : 요구사항, 모델링, 단일 요구사항, REED, RBT

A Single Requirement Modeling with Graphical Language for Embedded System

Jungsup Oh[†] · Hongseok Lee[††] · Hyunsang Park[†††] · Jangbok Kim[†††]
Kyunghee Choi[††††] · Kihyun Jung[†††††]

ABSTRACT

In order to do requirement-based testing of embedded system, we must have correct requirement specifications. But, natural language requirements of a client have ambiguity, inaccuracy, and inconsistency. To solve these problems, natural language requirements are modeled with modeling language such as UML and Simulink. During a modeling phase, the requirements are rearranged and retranslated in use-case. These activities are disadvantages of modeling. In this paper, we propose the technique, which is how to model a embedded system requirement into a model without rearranging and retranslating. This technique 1) represent a embedded system requirement with graphical language, and 2) model a requirement into a model. Because this technique only describes "what-to-do" of the requirement, this technique is useful to not only the low-level requirements but also the high-level requirements. We show some example systems modeled by REED, which has adopted this technique.

Key Words : Requirement, Modeling, Single Requirement, REED, RBT

1. 서 론

최근에 임베디드 시스템의 품질에 대한 관심이 매우 높아지고 있다. 좋은 품질을 가진 제품을 생산하기 위해서, 생산자는 제품이 고객의 요구대로 만들어 졌는지를 테스트 해야 한다.

최종 제품이 올바르게 만들어졌는지를 테스트 하기 위하여 Requirement-Based Testing(RBT)이 등장하게 되었다. RBT는 생산된 제품이 고객의 요구사항을 만족하는지를 검사하는 것이다. James Bach는 자신의 글에서 RBT를 위해서는 최소한 다음의 4가지는 만족해야 한다고 언급하였다.[1]

1. 정해진 요구사항 없이는 어떠한 테스팅도 불가능하다. *(Without stated requirements, no testing is possible)*
2. 소프트웨어 제품은 정해진 요구사항을 만족해야 한다. *(A software product must satisfy its stated requirements.)*
3. 모든 테스트 케이스는 하나 이상의 정해진 요구사항을 추적할 수 있거나 그 반대도 가능해야 한다.

† 준 회 원 : 아주대학교 대학원 컴퓨터공학과 박사과정
†† 준 회 원 : 아주대학교 대학원 전자공학과 박사과정
††† 준 회 원 : 아주대학교 정보통신전문대학원 정보통신공학과 박사과정
†††† 정 회 원 : 아주대학교 정보통신전문대학원 교수
††††† 정 회 원 : 아주대학교 전자공학부 교수
논문접수 : 2008년 4월 28일
수 정 일 : 1차 2008년 5월 22일, 2차 2008년 6월 13일
심사완료 : 2008년 6월 14일

(All test cases should be traceable to one or more stated requirements, and vice versa.)

4. 요구사항은 테스트 가능한 형태로 작성되어야 한다. *(Requirements must be stated in testable terms.)*

위의 요구사항 중에서 1, 2, 3번을 만족하기 위한 관리 도구들은 많이 존재한다[2]. 이들 중에서 Telelogic의 DOORS[3], Borland의 Caliber[4], Compuware의 Optimal Trace[5]등의 도구들이 많이 사용된다.이러한 요구사항 관리 도구들은 Hoffmann이 제시한 요구사항 관리 도구들이 가져야 하는 기능을 많이 포함한다.[6] 이러한 도구들을 통하여 요구사항의 변경 이력이나 기준선(baseline) 관리, 그리고 상위 요구사항과의 추적성 등을 관리 할 수 있다. 그러나 일반적으로 자연어로 기술되어 지는 요구사항은 모호성(ambiguity and impreciseness), 부정확성(inaccuracy), 불일치성(inconsistency and incompleteness)을 가지고 있다. 자연어에 포함되어 있는 단점 때문에 제품이 고객이 원하지 않는 방향으로 제품이 개발되기도 한다.

또한, 기존의 도구들은 James Bach가 언급한 내용 중에 4번째 요구사항인 테스트 가능한 형태로 작성되어지는 것도 항상 보장하지 못한다. 이를 보장하기 위한 방법으로는 의미가 분명한 그래픽 언어를 사용하여 요구사항을 다시 모델링하는 방법들이 사용되고 있다. 대표적인 모델링 방법으로는 UML[7]이나 Simulink[8] 등이 있다.

UML이나 Simulink는 use-case 중심으로 모델링을 수행하기 때문에 모델링 하는 엔지니어는 전체 요구사항을 이해하고 분석하여 use-case 중심으로 다시 구조화하는 작업이 필요하게 된다. 비록 고객의 요구사항과 모델 사이의 추적성을 제공한다고 하더라도 고객은 자신의 요구사항이 제대로 모델링이 되었는지 알기 힘들다. 고객은 모호한 자연어로 작성된 요구사항과 자신이 잘 이해하지 못하는 모델을 m:n의 관계로 요구사항 추적표를 통해서 비교해 보아야 하는 매우 어려운 위치에 놓이게 된다. 이러한 과정에서 제품의 오류가 많이 생산된다[9].

위와 같은 문제를 해결하기 위해서는 1) 요구사항은 자연어가 아닌 의미가 분명한 그래픽 언어를 이용하여 표현해야 하고, 2) 하나의 요구사항은 하나의 그래픽 모델로 표현할 수 있어야 한다. 요구사항을 이렇게 작성하면, 고객은 자신이 제시한 자연어로 작성된 요구사항이 의미가 분명한 그래픽적인 표기법으로 모델링된 결과를 검증 할 수 있다. 하나의 요구사항을 하나의 그래픽 모델로 표현한다면 보다 명확하고 분명하게 고객에서부터 개발자까지 모든 담당자들이 요구사항을 잘 이해할 수 있을 것이다. 따라서 본 논문에서는 단일 요구사항에 대한 모델링 방법을 제안하고 이 방법의 장점에 대해서 언급할 예정이다.

본 논문의 2장에서는 일반적인 요구사항 관리 도구들과 모델링 언어들의 특징을 살펴보고 3장에서는 단일 요구사항에 대한 모델링 방법을 살펴볼 예정이다. 4장에서는 실제로 단일 요구사항에 대한 모델링을 할 수 있는 도구인 REED에 대해서 살펴볼 예정이다. 5장에서는 REED를 사용하여 실제 시스템을 모델링한 통계를 소개할 예정이다. 끝으로 6장에서는 결론과 향후 과제에 대해서 언급할 것이다.

2. 요구사항 관리도구와 모델링 언어

Telelogic DOORS는 현재 가장 많이 사용되는 요구사항 관리 도구 이다. DOORS는 상위 요구사항과 하위 요구사항 사이의 관계를 연결, 추적, 분석하는데 매우 편리한 기능을 제공한다. 또한 DOORS는 변경되는 요구사항에 대하여 변경이력이나 기준선(baseline) 관리 등의 기능을 제공하여 효과적으로 잘 관리할 수 있도록 지원한다. DOORS는 문서와 비슷한 형태의 편집을 지원한다. 각각의 요구사항을 문서를 작성하듯이 편집할 수 있고, MS-Word와의 호환성도 가진다[10].

Borland는 소프트웨어 요구사항을 생성하고 관리하기 위한 2가지 제품을 제공한다. Borland Caliber DefineIT과 Borland CaliberRM이다. 이 둘은 함께 쓰여도 되고 각각 사용될 수도 있다. Borland Caliber DefineIT은 정확하고 완전한 소프트웨어 요구사항을 정의하기 위한 소프트웨어 요구사항 정의 소프트웨어이다. Borland CaliberRM은 소프트웨어 개발 주기 동안에 요구사항이 고객의 필요를 제대로 반영하는지를 관리할 수 있는 요구사항 관리 소프트웨어이다. Borland Caliber는 'business scenario'라는 개념을 제공한다. 'Business scenario'란 요구사항과 연관된 use-case에 대한 시나리오를 작성하여서 요구사항이 올바르게 작성되었는지를 관리할 수 있는 기능이다[11].

Optimal Trace는 비지니스 요구사항을 정의하고 관리하는 도구이다. Optimal Trace는 "structured requirements"라는 개념을 도입하였다. 이 방식은 소프트웨어 요구사항을 visual storyboard를 이용하여 고객의 관점으로 생성하는 것이다. Optimal Trace의 다른 점은 다른 도구들은 요구사항을 관리하는데 초점을 맞추고 있는 반면에 요구사항을 생성하는데 초점을 맞추고 있다는 것이다[12].

정형기법에 근간하는 LTL[14], CTL[15]등을 이용하여 모델을 생성하는 기법도 많이 사용되고 있다. LTL이나 CTL이 가지는 표현의 한계를 언급하지 않더라도, 사용이 너무 어렵고, 입출력 기반의 임베디드 시스템을 모델링 하는 데는 적합하지 않다. 따라서 로직을 기반한 방법보다는 그래픽 언어를 이용한 모델링 기법이 더욱 많이 사용되고 있다.

요구사항을 그래픽 언어로 표기하기 위하여 가장 널리 사용되는 방법으로 UML이 있다. UML 2.0은 class 다이어그램이나 statemachine 다이어그램 등 13가지 다이어그램을 통해서 시스템을 모델링 하는 언어이다. Class 다이어그램 등을 통해서 시스템을 정적인 관점에서 모델링을 할 수 있고, statemachine 다이어그램 등을 통해서 시스템을 행위적인 관점에서 모델링을 할 수 있다. 정적인 관점과 행위적인 관점에서 모델링 할 경우, 시스템의 구조뿐 아니라 시스템의 동작에 대한 명확한 모델링이 가능해 진다. 또한 UML은 많은 사용자 층을 확보하고 있고, IBM의 Rational Rose, RequisitePro, Borland의 Together, Telelogic의 Rhapsody 등 많은 도구에

서 지원된다. 이 도구들 중의 일부는 UML로부터 프로그래머가 작성해야 할 코드를 자동으로 생성하는 하위방향으로의 번역 기능까지도 제공한다. 그러나 UML은 요구사항이 13개의 다이어그램에 걸쳐서 표현되기 때문에 고객이 요구사항을 파악하기에 매우 어렵다[16].

Mathworks의 Simulink는 동적인 시스템을 모델링, 시뮬레이션, 분석하기 위한 소프트웨어이다. Simulink는 시스템을 설계하는데 필요한 여러 가지 컴포넌트를 제공한다. 예를 들면, AND/OR 게이트에서부터 신호 생성기, 수학 연산자에 이르기까지 다양한 컴포넌트를 제공한다. 이들 컴포넌트를 이용하여 시스템을 쉽게 모델링 할 수 있다. 시뮬레이션을 수행하여 시스템의 동작을 미리 예측해 볼 수도 있고, 설계된 시스템의 오류를 미리 찾아낼 수도 있다. 또한 다양한 부가 기능을 통하여 요구사항 관리기능, 검증 및 확인 기능도 가능하다. Simulink는 임베디드 시스템의 모델링에 탁월한 기능을 제공하지만, "how-to-do"에 기반한 모델링을 수행해야 한다. 즉, Simulink로 표현하기 위해서는 시스템의 출력이 "어떻게 만들어진다"는 것을 모두 표현해야 한다. 하지만, 상위레벨의 요구사항은 "how-to-do"를 모두 표현할 수 없다. 상위레벨에서의 요구사항에서는 "어떤 출력이 나와야 한다"고 정의하는 것이지, 그것이 "어떻게 만들어져 출력되어야 한다"고 정의하지 않을 수도 있기 때문이다.

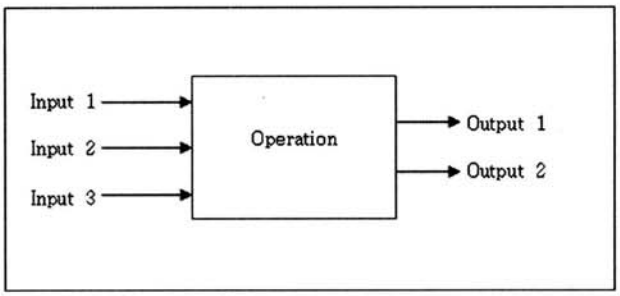
3. 임베디드 시스템의 단일 요구사항 모델링 기법

본 논문에서 제안하는 그래픽 언어를 이용한 요구사항 모델링 방법은 다음의 2가지 요구사항을 만족한다.

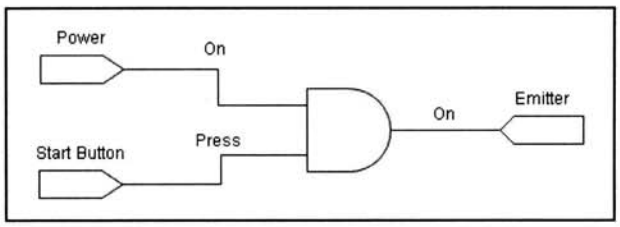
1. 자연어로 작성되어지는 요구사항을 의미상으로 분명한 그래픽 언어를 사용하여 요구사항 다이어그램으로 표현할 수 있어야 한다.
2. 하나의 요구사항에 대하여 하나의 요구사항 다이어그램으로 표현할 수 있어야 한다.

임베디드 시스템의 요구사항은 시스템이 받은 입력을 어떻게 처리해서 어떤 출력을 내보내야 한다는 것에 대한 정의이다. 즉, 임베디드 시스템의 요구사항은 입력과 출력, 그리고 그 값들을 다루기 위한 몇 가지 연산으로 이루어진다. 입력이란 시스템 외부의 사용자로부터 발생되어 들어오는 입력일 수도 있고, 내부의 다른 요구사항으로부터 발생되어 들어오는 이벤트나 상태의 변경이 될 수도 있다. 출력이란 시스템이 외부의 사용자에게 보여주는 출력일 수도 있고, 다른 요구사항으로 내보내는 이벤트나 상태의 변경이 될 수도 있다. 즉, 임베디드 시스템의 각각의 요구사항은 (그림 1)과 같이 입력과 출력 중심으로 표현할 수 있다.

(그림 1)과 같이 각각의 요구사항에 대해서 입력(Input)과 출력(Output)을 정의하고, 이들 사이의 관계를 특정한 의미를 가지고 있는 연산(Operation)들로 연결하면 임베디드 시스템의 요구사항을 명확하게 정의할 수 있다.



(그림 1) 입/출력 기반의 요구사항



(그림 2) 요구사항 다이어그램의 예제

다음과 같은 아주 간단한 전자레인지의 요구사항이 있다고 가정해 보자.

[요구사항] 전자레인지는 'Start'버튼을 누르면 전자파를 발산해야 한다.

위의 요구사항은 (그림 2)와 같이 의미가 분명한 요구사항 다이어그램으로 모델링 될 수 있다. 자연어로 기술된 요구사항에는 나타나지 않았지만 'Power'가 'On'이어야 한다는 의미까지 다이어그램에 나타냄으로써 보다 명확한 요구사항이 되었다. (그림 2)에서 입력은 'Power', 'Start Button'이고 출력은 'Emitter'이다. 이와 같이 2개의 입력이 모두 만족되어야 출력이 발생하는 것을 의미상으로 분명하게 표현한 것이다. 또한 (그림 2)의 다이어그램에서 보듯이 'Power'가 'On'인 것, 'Emiiter'가 'On'인 것 만을 기술하도록 하였고, 'Emitter'가 'On'이 되기 위해서 어떠한 신호를 어디에 보내야 하는지에 대한 내용은 기술하지 않는다. 이처럼 요구사항의 "what-to-do"만을 기술하여 표현 할 수 있다.

요구사항 다이어그램은 입력과 출력을 나타내는 엔티티 블록과 연산을 나타내는 연산 블록, 그리고 이들 사이의 연결을 나타내는 링크로 구성되어 진다.

3.1 엔티티 블록(Entity Block)

엔티티 블록은 요구사항 다이어그램에서 입력과 출력을 나타내는 것으로, REED에는 약 10개 가량의 엔티티 블록을 정의하고 있다. 대표적인 것들은 <표 1>과 같다.

엔티티 블록은 프로그래밍관점에서 바라보았을 때, 식별자(identifier)에 해당한다. 따라서 각각의 엔티티 블록은 <name>이라는 이름을 가지고 있다. 이 이름을 가지고 다른 요구사항과 연결되어 질 수 있다.

위의 <표 1>에 있는 블록들 중에서 입력장치와 상수는 입력으로만 사용되고, 출력장치는 출력으로만 사용되며, 그

〈표 1〉 엔티티 블록의 예

기호	이름	의미
<name>	입력장치	시스템의 입력을 표현한다. 예를 들면, 버튼, 센서 값 등이 이에 속한다. 입력장치의 이름은 <name>이다.
<name>	출력장치	시스템의 출력을 표현한다. 예를 들면, 모터, LED 값 등이 이에 속한다. 출력장치의 이름은 <name>이다.
<name>	메모리	요구사항의 기술을 용이하게 하기 위해서 사용되는 변수이다. 변수의 이름은 <name>이다.
<name>	내부 이벤트	<name>이라는 이벤트를 발생 시킨다.
C	상수	연결된 링크에 적힌 상수 값을 의미한다.

외의 메모리와 내부 이벤트는 입력과 출력에 모두 사용될 수 있다. 입력과 출력에 모두 사용되는 메모리와 내부 이벤트와 같은 경우는 하나의 요구사항의 출력이 다른 요구사항의 입력으로 사용될 수 있다.

3.2 연산 블록(Operation Block)

연산 블록은 요구사항이 처리해야 하는 작업을 수행하기 위해서 해야 하는 연산(operation)들을 표현한다. REED에는 약 20개 가량의 연산 블록을 정의하였으며, 이들의 대표적인 예는 다음의 <표 2>와 같다.

연산 블록은 프로그래밍 관점에서 바라보았을 때, 함수에 해당한다. 연산 블록의 입력은 엔티티 블록일 수도 있고, 다른 연산 블록의 출력일 수도 있다. 그러나 연산 블록은 항상 입력과 출력을 필요로 하기 때문에 제일 앞이나 뒤에 사용될 수는 없다.

위의 <표 2>에 있는 블록들 중에서 맨 뒤의 Loop와 Sequential 블록을 따로 FC(Flow of Control) 블록이라고 부른다. 다른 블록은 출력 포트가 여러 개가 있어도 동시에 여러 개의 출력을 내보내거나 여러 개의 출력 중에서 하나의 출력 만을 내보내지만, 이 블록들은 출력을 하나씩 순차적으로 내보내야 하는 기능을 수행한다. 이 블록들을 이용하여 요구사항을 효율적으로 작성할 수 있다.

3.3 링크(Link)

각각의 블록을 연결하기 위해서 링크를 사용한다. 링크는 방향성을 가지고 있다. 각각의 블록은 블록의 왼쪽에 위치한 포트가 입력 포트, 블록의 오른쪽에 위치한 포트가 출력 포트다. 링크는 블록의 오른쪽에 위치한 출력 포트에서 출발하여 블록의 왼쪽에 위치한 입력 포트로 연결된다. 따라서 링크는 출력 포트에서 입력 포트로의 방향성이 존재한다.

링크에는 2가지 정보가 포함되어 전달된다. 하나는 '값'이고, 다른 하나는 '활성화/비활성화' 신호 이다. '값'이란 우리가 일반적으로 이해할 수 있는 값을 의미한다. 이 '값'은 메모리가 가지고 있는 값일 수도 있고, 고객에게 직접적으로 보여지는 값일 수도 있다. 또한 연산 블록에 의해서 계산된 값일 수도 있다. '활성화/비활성화' 신호는 일종의 제어 신호

〈표 2〉 연산 블록의 예

기호	이름	의미
EN, X1, X2, X3, X4, Assignment	Assignment	EN 포트로 어떤 값이 입력될 때, 입력 포트 X_1~X_4으로 입력된 값을 사용하여 함수를 실행한 후, 그 값을 출력 포트로 출력한다.
EN, Bypass	Bypass	EN 포트가 활성화 되었을 때, 입력으로 들어온 값을 출력으로 내보낸다.
EN, =, T, F	Equal	EN 포트가 활성화 되었을 때, 입력으로 들어온 2개의 값이 같으면 출력 T를 활성화 하고, 입력 값이 서로 다르면 출력 F를 활성화 한다.
	Flow-AND	입력포트가 모두 활성화 되면 출력 포트가 활성화 되고, 그렇지 않으면 출력포트는 활성화 되지 않는다.
Table name, EN, <<N:1 Mapping>>	Table	EN 포트가 활성화 되었을 때, 입력 값에 대응하는 값을 테이블에서 찾아서 출력 값으로 내보낸다.
Start, Entry, Loop, Do, Stop, Exit	Loop	'Loop'는 동일한 작업이 반복되는 것을 표현할 때 사용하는 객체이다. 'Start' 포트로 입력되면, 'Entry'에 연결된 작업을 수행한 후, 'Do'에 연결된 작업을 'Stop'으로 입력이 발생할 때까지 반복한다는 것을 의미한다. 'Stop'에 입력이 발생하면, 'Exit'에 연결된 작업을 수행하고, 종료된다.
Sequential, 1, 2, 3, 4	Sequential	'Sequential'은 일련의 행동을 차례대로 기술할 때 사용된다. 각 출력 포트에 연결된 작업을 번호순으로 차례대로 수행하라는 것을 가리킨다.

이다. 모든 블록들은 '활성화' 신호를 받아야 동작할 수 있다. 연산 블록은 자신의 기능에 맞게 출력신호의 '활성화/비활성화' 여부를 결정하여 출력한다.

링크에는 특정 조건이나 값이 같이 기록되어 질 수 있다. 링크에 조건이 기록되었다면 링크는 조건이 참일 경우에만 '활성화' 신호를 전달하고, 조건이 거짓일 경우에는 '비활성화' 신호를 전달한다. 링크에 값이 기록되었다면, 링크가 활성화가 되었을 때, 링크에 기록된 값을 같이 전달한다. 링크에 값이 기록되는 경우는 앞의 블록에서 값이 전달되어 들어오지 않고, '활성화/비활성화' 신호만 전달되는 경우에만 해당된다.

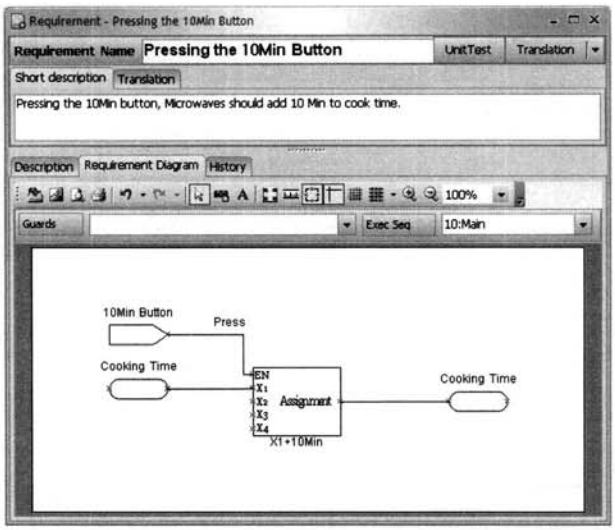
4. REED : REquirement EDitor

임베디드 시스템의 하나의 요구사항을 하나의 모델로 1:1 대응 시키기 위한 도구로 REED(REquirement EDitor)라는 도구가 개발되었다. REED는 3절에서 제시한 2가지의 요구사항을 만족하는 도구이다. 즉, 자연어로 작성된 하나의 요구사항을 하나의 요구사항 다이어그램으로 그래픽 언어를 이용하여 표현하여 함께 관리할 수 있다. REED에 사용되는 모든 블록들은 "how-to-do"가 없는 "what-to-do"에 해당하는 의미를 가지고 있다. 즉, 각 블록들은 어떤 입력에 대하여 어떤 출력이 생성된다는 것은 기술되나 그 출력이 어떻게 만들어진다는 것을 표현하지는 않는다. REED의 블록들은 "what-to-do"만을 표현하기 때문에, 요구사항 작성시에 시스템의 아주 상세한 기능을 모두 표현하지 않아도 요구사항을 명확하게 표현할 수 있다. 따라서 REED는 상위레벨의 요구사항도 표현할 수 있다. 즉, 자연어로 기술되어 지는 모든 임베디드 시스템의 요구사항은 상위레벨 요구사항에서부터 하위레벨 요구사항까지 모두 표현 가능하다.

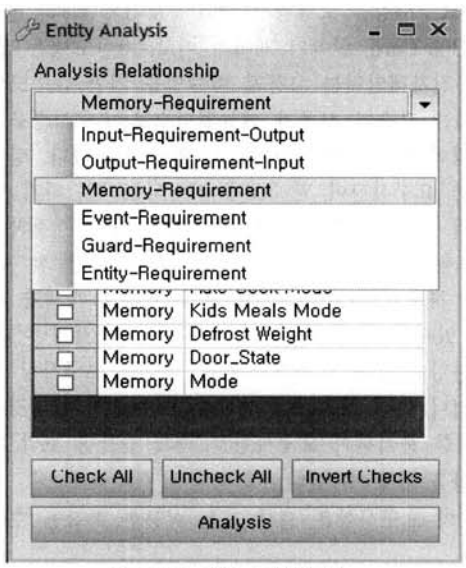
REED는 각각의 요구사항에 대해서 (그림 3)과 같은 윈도우를 제공해 준다. 각각의 윈도우는 Requirement Name, Requirement Short description, Requirement Diagram을 한꺼번에 볼 수 있도록 설계되었다. Requirement Short description 영역은 실제로 자연어로 작성된 요구사항이 기록되는 부분이다. 고객은 자신이 작성한 하나의 요구사항이 어떻게 요구사항 다이어그램으로 모델링 되었는지 확인 할 수 있다.

(그림 3)은 전자레인지의 요구사항 중에서 버튼 중의 하나인 '10Min Button'에 대한 요구사항이다. 전자레인지의 사용자가 '10Min Button'을 누르면 조리시간이 10min 증가해야 하는 것이다. 다이어그램에서 사용자가 직접 시스템에 입력을 발생시키는 '10Min Button'은 입력장치로, 전자레인지가 내부적으로 관리하는 'Cooking Time'은 메모리로 처리하였다. 이렇게 모델링된 다이어그램은 개발자가 개발하는데 매우 명확한 가이드라인을 제시하여 줄 수 있다.

REED는 요구사항 관리 도구가 일반적으로 제공하는 기준선(baseline) 설정, 변경 이력 관리 기능 뿐만이 아니라 여러 가지 유용한 분석 도구를 제공한다. 예를 들면, (그림 3)에 사용된 'Cooking Time'이 어떤 요구사항에서 사용되고



(그림 3) REED에서의 요구사항의 예

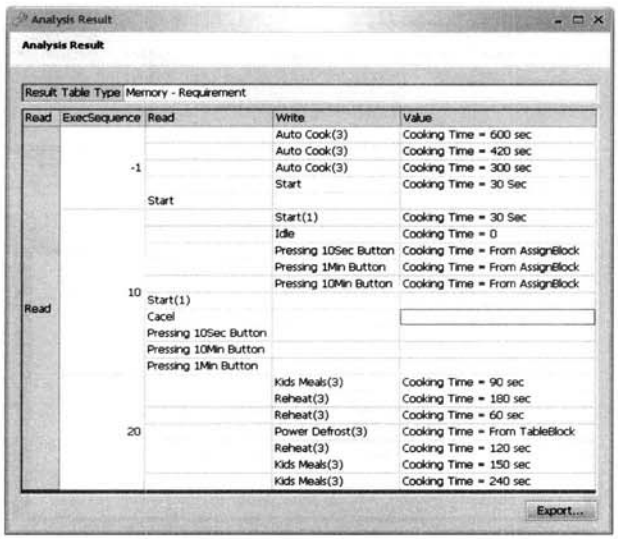


(그림 4) 엔티티 분석기능

있는지 분석할 수 있다. 이러한 방식으로 요구사항을 'input - requirement-output', 'output - requirement - input', 'Memory-requirement' 등 관계 등 6가지 관점에서 분석할 수 있다(그림 4). (그림 5)는 위의 (그림 3)에서 사용된 'Cooking Time'이라는 메모리에 대한 'Memory-Requirement' 관계로서 이 메모리를 어떤 요구사항에서 읽는지(read), 혹은 어떤 요구사항에서 쓰는지(write)를 나타내 준다. (그림 3)의 'Pressing 10Min Button'이라는 요구사항에서는 읽기도 하고 쓰기도 한다는 분석 결과를 보여주고 있다. 또한 Write 부분에는 'Cooking Time'이라는 메모리가 어떤 요구사항에서 어떠한 값으로 설정 되었는지도 보여주고 있다.

5. REED 모델링 적용 사례 분석

REED를 이용하여 3개의 실제 모델과 1개의 가상 모델에



(그림 5) 분석 결과

대한 요구사항을 모델링 하였다 <표 3>. 3개의 실제 모델은 TC, Bus Card, 굴삭기 제어기 이고, 가상모델은 전자레인지 이다. 전자레인지는 사용자 매뉴얼을 기반으로 모델링 하였다. 이는 단순히 사용자 매뉴얼만으로 요구사항을 정의하고 모델링이 가능한지를 알아보기 위한 시도였다. 실제 모델 3가지는 요구사항이 약 300개 가량되는 간단하지 않은 시스템이다. 그러나 각각의 경우에 30개 가량의 정의된 블록만을 사용하여 요구사항 모두를 각각 모델링 할 수 있었다. TC의 경우에 하나의 요구사항에 58개의 객체가 사용된 경우도 있지만, 총 사용된 객체의 종류는 31개 이다. 하나의 요구사항에 많은 객체가 사용된 경우는 요구사항 자체가 복잡하거나 여러 개의 요구사항이 하나로 합쳐진 경우도 있었다. 이를 분석하여 요구사항을 작게 만들면 보다 간결한 요구사항을 만들 수 있다.

TC의 요구사항은 센서 값을 이용한 복잡한 수식 계산이 많은 특징이 있고, Bus Card의 요구사항은 운전자 단말기와 탑승자 단말기 간의 메시지 교환이 많은 특징이 있다. 또한 굴삭기 제어기는 하나의 요구사항은 간단하나 개수가 많은 특징이 있고, 전자레인지는 사용자 매뉴얼 기반의 요구사항이므로 상세한 요구사항이 아닌 상위레벨 요구사항으로 구성되어 있다. 본 연구에서 제시한 방법을 이용하여 서로 다른 특징을 가진 요구사항을 모두 기술할 수 있음을 확인하였다.

각각의 요구사항은 (그림3)과 같이 고객이 제시한 자연어의 요구사항과 그래픽 언어를 이용한 요구사항 다이어그램을 한눈에 볼 수 있기 때문에 고객의 입장에서 자신의 요구사항이 제대로 모델링 되었는지를 확인하기에 매우 용이하다.

또한 자연어로 작성된 요구사항을 요구사항 다이어그램으로 하나씩 모델링 하는 과정을 통하여, 자연어에서 미처 발견하지 못한 모호성, 부정확성, 불일치성을 알아낼 수 있었다. 이렇게 단순히 모델링 하는 과정을 통하여 요구사항을 보다 분명하고, 정확하고, 완전하게 만들 수 있었다. 요구사항 다이어그램이 복잡해 지는 경우는 이를 쉽게 분리할 수 있었고, 결국에는 자연어로 작성된 요구사항이 여러 개의 요구사항을 포함하고 있다는 것을 알아내기도 하였다. 이를 통하여 요구사항을 보다 간결하게 만들기도 하였다. 이렇게 간결하고 분명하게 작성된 요구사항은, 시스템의 개발과정 동안에 요구사항을 변경하거나, 요구사항에 대한 테스트를 수행할 경우에 매우 쉽게 관리할 수 있다.

6. 결론 및 향후 과제

본 논문에서는 하나의 요구사항에 대하여 하나의 모델로 모델링하기 위한 방법과 이를 실제로 구현하고 있는 REED에 대해서 언급하였다. 임베디드 시스템의 요구사항을 자연어가 아닌 의미가 분명한 그래픽 언어를 이용하여 표현하고, 하나의 요구사항을 하나의 요구사항 다이어그램으로 표현하

〈표 3〉 예제 시스템 모델링 통계

		시스템			
		TC	Bus Card	굴삭기 제어기	Oven
개요		자동차에 내부 온도조절장치	버스 요금 계산을 위한 운전자 단말기	굴삭기 제어기용 엔진과 펌프의 마력 매칭을 위한 장비	일반 전자 레인지
장치	입력장치	센서(7종), 스위치(13종)	키패드 GPS	센서(6종), 스위치(15종)	스위치(10종)
	출력장치	Actuator(8종)	LCD 스피커	Solenoid Valve 등 (7종)	LCD Actuator (Motor, Microwave Emitter)
	입/출력장치	CAN	RS-232, 무선 랜 USB	CAN bus (engine 등) Gauge Pannel	
요구사항 총 개수		299	262	373	26
사용된 객체의 종류		31	23	28	20
객체의 최대 사용 개수 (요구사항별)		58	53	37	17
객체의 평균 사용 개수 (요구사항별)		9.49	7.29	4.54	7.96

는 방법에 대해서 설명하였다. 또한 시스템의 입출력을 기반으로 "what-to-do"만을 기술하기 때문에 상위레벨의 요구사항이나 하위레벨의 요구사항 모두에 적용할 수 있다는 점을 예제를 통하여 볼 수 있었다. 이런 모델링 기법을 통하여 고객은 자신의 요구사항이 올바르게 모델링 되었는지 확인할 수 있고, 요구사항 엔지니어링 단계에서의 오류를 많이 줄일 수 있었다. 또한 실제 시스템에 적용해 봄으로서 이러한 모델링 방법이 충분히 가능하고 또한 효과적임을 알아보았다.

이제 이러한 방식으로 모델링된 모델로부터 자동으로 테스트 케이스를 생성하고 이를 테스트 하는 방식을 개발 할 예정이다. 이를 통하여 James Bach가 제시한 요구사항과 테스트 케이스간의 추적성(traceability)을 자동으로 지원할 수 있을 것이다. 이 기능은 RBT를 자동으로 수행하는 중요한 요소가 될 수 있으리라 기대한다.

Acknowledgment

본 연구는 산업자원부 및 한국부품소재산업진흥원의 부품소재개발사업의 연구결과로 수행되었음.

참 고 문 헌

[1] James Bach, "Risk and Requirements-Based Testing," IEEE Computer, Vol.32, No.6, pp.113-114, June, 1999.
[2] INCOSE Requirements Management Tools Survey, http://www.paper-review.com/tools/rms/read.php
[3] Telelogic AB, http://www.telelogic.com/products/doors/doors/index.cfm
[4] Borland Software Co., http://www.borland.com/us/products/caliber/index.html
[5] Compuware Co., http://www.compuware.com/products/optimaltrace/
[6] Matthias Hoffmann, Nikolaus Kühn, and Margot Bitner, "Requirements for Requirements Management Tools," Proceedings of the 12th IEEE International Requirements Engineering Conference(RE'04), pp.301-308, 2004.
[7] Object Management Group, "Unified Modleling Language (UML), Version 2.1.2", http://www.omg.org/spec/UML/2.1.2/, November 2007.
[8] The MathWorks, Inc., http://www.mathworks.com/products/simulink/
[9] J. Martin, *An information Systems Manifesto*, Prentice Hall, 1984.
[10] Telelogic, "Datasheet : Telelogic DOORS," http://www.telelogic.com/download/get_file.cfm?id=3726, 2005.
[11] Borland, "Integrating Requirements into Software Development," http://www.borland.com/resources/en/pdf/solutions/rdm_integrating_reqs_into_software_dev.pdf, White paper, June 2006.
[12] Compuware, "How Compuware Optimal Trace helps business analysts meet the business challenge," Fact sheet, 2006.
[13] Telelogic, "Key Enabling Technologies of Telelogic Rhapsody," http://modeling.telelogic.com/products/rhapsody/index.cfm
[14] A Pnueli, "A temporal logic of programs." Theoretical Computer Science, 13:45-60, 1981.
[15] E.M.Clarke and E.A.Emerson. "Synthesis of synchronization skeletons for branching time temporal logic," In D. Kozen, editor, Logic of Programs Workshop, number 131 in LNCS. Springer Verlag, 1981.
[16] Mor Peleg and Dov Dori, "The Model Multiplicity Problem: Experimenting with Real-Time Specification Methods," IEEE Transactions on Software Engineering, Vol.26, No.8, pp.742-759, AUGUST, 2000.

오 정 섭

e-mail : jsoh@ajou.ac.kr
1997년 아주대학교 정보 및 컴퓨터공학부 (학사)
1999년 아주대학교 대학원 컴퓨터공학과 (석사)
1999년~현 재 아주대학교 대학원 컴퓨터공학과 박사과정
2001년~2003년 ㈜디오텔 선임연구원
2003년~2006년 ㈜삼성탈레스 책임연구원
관심분야 : 소프트웨어 공학, 요구사항 공학, 임베디드 시스템, 실시간 시스템, 분산 시스템 등

이 홍 석

e-mail : myhong5@ajou.ac.kr
2003년 아주대학교 정보 및 컴퓨터공학부 (공학사)
2003년 아주대학교 전자공학부(공학사)
2005년 아주대학교 전자공학과(공학석사)
2005년~현 재 아주대학교 대학원 전자공학과 박사과정
관심분야 : 정형 검증, 명세 기술, 임베디드 시스템

박 현 상

e-mail : elvaimay@ajou.ac.kr
2005년 아주대학교 정보 및 컴퓨터공학부 (공학사)
2007년 아주대학교 정보통신전문대학원 정보통신공학과(공학석사)
2007년~현 재 아주대학교 정보통신전문대학원 정보통신공학과 박사과정
관심분야 : 소프트웨어 공학, 임베디드 시스템, 운영 체제, 실시간 고속 네트워크 시스템 등

최 경 희

e-mail : khchoi@madang.ajou.ac.kr
1976년 서울대학교 수학교육과(학사)
1979년 프랑스 그랑데꼴 Enseeiht대학(석사)
1982년 프랑스 Paul Sabatier대학 정보공학부 (박사)
1982년~현 재 아주대학교 정보통신전문대학원 교수
관심분야 : 운영 체제, 분산시스템, 실시간 및 멀티미디어시스템 등

김 장 복

e-mail : bbok@ajou.ac.kr
2003년 아주대학교 정보 및 컴퓨터공학부 (학사)
2005년 아주대학교 정보통신전문대학원 정보통신공학과(석사)
2005년~현 재 아주대학교 정보통신전문대학원 정보통신공학과 박사과정
관심분야 : 네트워크 보안, 임베디드시스템, 운영 체제, 임베디드 소프트웨어 테스팅

정 기 현

e-mail : khchung@madang.ajou.ac.kr
1984년 서강대학교 전자공학과(학사)
1988년 미국 Ilinois주립대 EECS(석사)
1990년 미국 Purdue대학 전기전자공학부 (박사)
1991~1992년 현대반도체 연구소
1993년~현 재 아주대학교 전자공학부 교수
관심분야 : 컴퓨터구조, VLSI 설계, 멀티미디어 및 실시간 시스템 등