한국방위산업학회
제18권 제1호
2011년 6월

재사용성 제고를 위한 요구사항 분석 및 명세 기법

김 동환[1†] 김 영환[2] 김 진사[3]

내용목차

1†. LIG Nex1 연구위원
(교신저자 Tel: 031-288-9142 Fax: 031-288-9208 E-mail: kimdonghwan@lignex1.com)
2 LIG Nex1 수석연구원
3 LIG Nex1 선임연구원

논문접수일: 2011년 4월 27일 게재확정일:2011년 6월 27일
논문수정일 (1차: 2011년 6월 22일)

Korea Association of
Defense Industry Studies
Vol.18, No.1, June 2011

Requirements Analysis and Specification for Enhancing Reusability

Kim, Dong Hwan[1†] Kim, Young Hwan[2] Kim, Chin Sa[3]

Abstract

The quality of requirements specification mainly impacts quality including a system's reusability and components. Faults in the specification could cause a number of problems in the system development, such as cost and schedule overruns.

Therefore, the developer has the ability to determine the specification with completeness, consistency, testability and traceability from vague user requirements. In order to develop a good specification with high reusability, various requirements analysis and specification method have been used by the developers in developing the system.

There are two traditional analysis and specification methods, which are Structured Analysis and Use Case Model. There are, however, disadvantages when using these two methods, such as the inadequacy for defense system, data modeling and formalism. These limitations could cause a developer to create poor specifications and systems.

To resolve this disadvantage, this paper proposed to combine this approach with the existing methods in order to maximize the advantages of the other methods. This method is called Use-Case Event Modeling. It enables the developer to analyze the user requirements and specify a well defined specification with high productivity and reusability.

<*Key Words*> *Requirements, Structured Analysis, Use-Case Modeling, Reusability*

1. 서론

소프트웨어 위기(Software Crisis)는 소프트웨어 투자에 들어가는 비용의 대부분이 소프트웨어 유지보수 단계에서 발생하고 있다는 것이 주된 원인으로 제기되었다. 이는 결국 고 품질의 소프트웨어를 신속히 개발할 수 있는 방법론의 연구로 이어져 체계공학과 함께 소프트웨어 공학이라는 관리적 및 기술적 접근방법의 중요성을 인식하기에 이르렀다. 특히 국방 및 항공분야와 같은 복잡하고 대규모 시스템에서 더욱 심각한 문제로 제기되고 있다. 무기체계가 점점 하드웨어 중심에서 소프트웨어의 중심의 체계로 발전하고 있는 현재에도 국방 무기체계 개발에 매우 중요한 이슈라 해도 과언이 아닐 것이다.

이로 인해 소프트웨어의 재사용성을 높이는 것은 국가 및 사회적으로 중요한 경쟁력이 되고 있다. 특히 전투체계와 같이 유사한 도메인에 존재하는 다양한 응용체계 개발에서의 재사용성은 매우 중요한 요소로 인식되고 있다 [1]. 특히 코드의 재사용 뿐 만 아니라 개발초기 단계인 요구사항을 분석하고 명세화하는 단계부터 재사용을 고려하는 것이 재사용을 극대화할 수 있는 방법으로 인식되고 있다. 이는 코드를 재사용하는 단순한 접근방법을 넘어서 소프트웨어 프로덕트라인(Software Product Line)의 개념을 도입하여 전체 개발수명주기를 재사용하고자하는 접근방법으로 발전하고 있다[5].

특히 요구사항 분석단계의 결함은 시스템 품질 및 개발일정에 미치는 영향이 매우 높다는 것은 통계적으로 입증된 사실이다[4]. 따라서 시스템의 개발 초기 단계인 요구사항 분석에서부터 재사용성을 고려한 접근이 시스템의 재사용성에 매우 중요하며, 표준화되고 정형화된 요구사항 분석 및 명세기법은 재사용성을 높이는 중요한 요소라고 할 수 있다.

이를 위해 본고에서는 기존 요구사항 분석 및 명세 기법의 장점을 이용하여 추상화 수준에 따라 논리적인 기능단위를 추출하여 이를 중심으로 요구사항을 분석하고, 정형화된 명세 기법에 의해 명세 하는 방법을 제시함으로써 소프트웨어 프로덕트라인의 품질을 제고하고, 공통성 및 가변성을 용이하게 식별할 수 있는 방법을 제시하고자 한다.

2. 요구사항에 대한 이해

2.1 요구사항의 정의

IEEE 용어집에 따르면 요구사항을 사용자관점에서 문제를 해결하거나 목적을 달성하기 위하여 사용자가 필요로 하는 조건 및 기능이라고 정의하고 있고, 개발자관점에서는 시스템 또는 시스템 컴포넌트가 처리하거나 충족시켜야 하는 조건 또는 기능이라고 정의하고 있다.

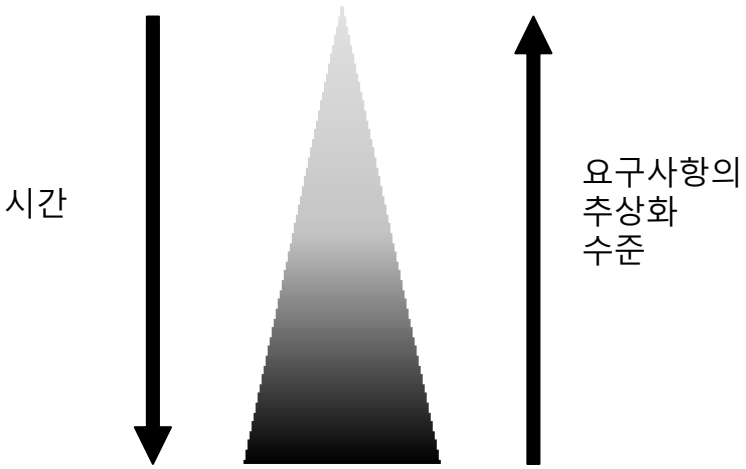
위의 정의로부터 다양한 관점에서의 사용자 요구사항을 도출하고, 이를 개발자 관점에서 일관성 있게 공유할 수 있는 형태로 요구사항을 정의할 필요가 있다는 것을 이해하는 것이 필요하다. 즉, 요구사항은 관련된 다양한 사용자로부터 발생되며, 이를 분석하여 개발자관점에서 시스템이 만족해야 할 요구사항을 기록하고 기술하여야 한다. 이를 위해 요구공학에서는 요구사항 사항을 도출하고, 분석하고, 명세화하는 작업을 구분하고 있으

며, 각 단계에서의 요구사항은 동일한 형태라기보다 변환된 형태로 존재하고 기록되어야 한다. 다시 말해 시스템 개발자는 사용자의 요구사항을 시스템 개발을 위한 정형화된 형태로 정의할 수 있어야 한다는 것을 이해할 필요가 있다.

2.2 요구사항의 형태 및 종류

요구사항의 정의에서 알 수 있듯이 요구사항은 다양한 형태로 존재하며, 관점에 따라 다양한 종류가 존재한다. 개발자가 이러한 요구사항의 형태 및 종류를 이해하고 요구사항을 분석하고 명세화하는 것은 양질의 요구사항명세서를 개발하는데 중요한 요소라 할 수 있다.

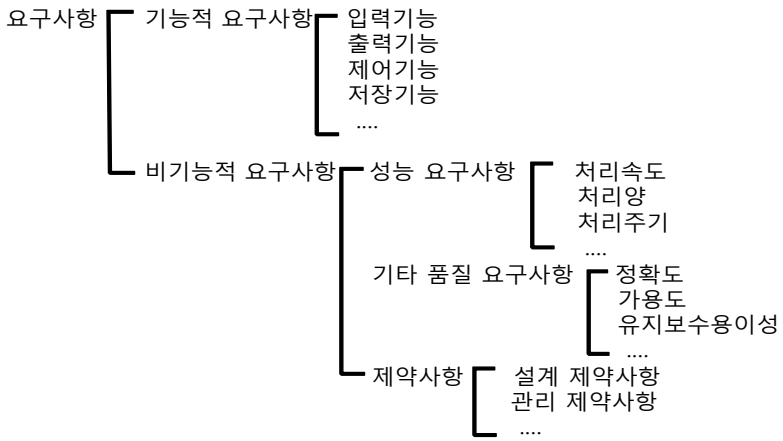
요구사항은 개발단계에 따라 그 형태가 추상화된 형태에서 상세한 형태로 발전되며 구체화되거나 어떤 경우에는 변경이 이루어져 요구사항의 형태가 변경될 수 있다. <그림 1>은 개발단계의 진행에 따른 요구사항의 추상화 수준을 그림으로 표현하고 있다.



<그림 1> 시간에 따른 요구사항의 추상화수준

따라서 이렇게 구체화되고 변경되는 요구사항을 잘 수용하기 위해서는 요구사항을 논리적인 단위로 식별하고 추상화 수준을 관리함으로써 시스템의 품질 및
개발일정을 만족시킬 수 있다[7].

또한 요구사항은 크게 기능적 및 비기능적 요구사항으로 구분할 수 있다. 두 가지의 요구사항 분류를 올바르게 이해하고 이를 구분하여 요구사항을 도출하는 것 또한 시스템의 품질을 좌우하는 매우 중요한 요소이다. 기능적 요구사항은 개발할 시스템 자체 즉, What(무엇)에 대한 요구사항이라면, 비기능적 요구사항은 기능적 요구사항을 How(어떻게) 개발할 것 인가에 대한 요구사항을 의미한다. <그림 2>는 요구사항에 대한 종류에 대해 전형적인 분류표를 나타내고 있다.



<그림 2> 요구사항의 종류

사용자 관점에서 기능적 요구사항은 시스템이 응당 수행해야 할 요구사항이며, 비기능적 요구사항은 시스템의 품질이나 제약사항을 규정짓는 요구사항이라고 할 수 있다. 개발자 관점에서 기능적 및 비기능적 요구사항 모두에 관심을 갖고 분석을 수행해야 하지만, 비기능적인 요구사항은 설계와 관련되는 정량적인 요구사항이어서 재사용성 관점에서는 그 의미가 크지 않다. 따라서 본고에서 요구사항을 도출하고 명세화하는 기법은 기능적 요구사항의 분석 및 명세에 초점이 맞추어져 있으며, 비기능적 요구사항의 분석 및 명세는 제외한다.

2.3 요구사항 명세서의 조건

요구사항을 도출하고, 이를 분석한 후 요구사항을 기술한 문서가 요구사항 명세서이다. 요구사항 명세서는 요구분석을 통해 공학적인 활동을 수행한 후 대상 시스템의 최종 요구 성능을 기술한 문서이므로 단순히 사용자의 요구사항을 정리한 문서가 아닌 공학적 개발결과물로서 다음과 같은 주요 조건을 갖추어야 한다[2].

첫째, 명세서는 논리적이어야 한다. 요구사항은 가능한 특정한 구현방법에 의존적이지 않고 논리적인 관점에서 정의되어야 한다. 특정 구현방법에 의존적인 요소가 포함된다는 것은 그 만큼 다양한 대안을 설계하기 어려울 수 있고, 기술 변화에 취약한 명세서가 될 수 있다는 것을 의미한다.

둘째, 명세서는 일관성이 있어야 한다. 명세서에 기술된 각 부분이 상호 일치해야 한다. 명세서의 불일치 요소가 존재한다는 것은 결함이 존재한다는 것을 내포하고 있기 때문에 명세서의 일관성은 매우 중요한 조건이다.

셋째, 명세서는 완전해야 한다. 완전한 시스템이 없듯이 완전한 명세서는 존재하지 않는다. 명세서가 완전해야 한다는 의미는 상대적인 의미로서 요구사항의 누락이 없도록

명세화되어야 하는 당위성에 대한 조건이다. 따라서 개발자는 요구사항의 누락이 없도록 요구사항을 분석하고 명세화해야 한다는 의미를 표현한 것이다.

넷째, 명세서는 테스트가 가능해야 한다. 명세서는 최종 시스템이 개발된 후 시스템의 시험을 위한 기본 잣대 또는 심지어 계약서의 역할을 한다. 따라서 명세서가 테스트가 가능하도록 명세가 되는 것은 중요한 조건 중에 하나이다.

다섯째, 명세서는 추적이 용이해야 한다. 명세서는 설계의 기준이 되는 문서로서, 단순한 요구조건을 제시하는 것이 아니라 설계의 각 구성품과 연계되어 추적이 용이하도록 구성되어야 한다. 이는 요구사항을 논리적인 기능단위로 잘 조직화해서 구성을 하고, 설계와 연계된 개념들이 포함되어야 한다는 것을 의미한다.

상기의 요구사항 명세서가 가져야 할 특성들을 유지한다는 것은 요구사항명세서에 존재하는 결함을 조기에 제거하는 것이기도 하지만, 재사용성을 높이는 중요한 요소가 된다.

3. 기존 요구사항 분석 및 명세기법

3.1 구조적 분석기법

구조적 분석 기법은 정보체계를 전통적으로 분석하고 명세화 하는 기법으로 사용되어 왔으며, 객체지향 설계가 나오기 전까지 매우 유용한 분석방법으로 사용되어 왔다[8]. 구조적 분석 기법은 자료흐름도와 자료사전이라는 도구를 활용하여 개발하고자 하는 시스템을 분석하고 명세화하고 있다. <그림 3>은 구조적 분석 기법의 기본적인 개념을 설명하고 있다.

구조적 분석 기법은 자료흐름도(Data Flow Diagram)라는 도구를 이용하여 시스템의 기능을 분석하고 명세화한다. 자료흐름도는 시스템의 기능을 데이터를 입력 받아 처리 또는 변환하여 결과를 내는 프로세스(Process)들 간의 데이타흐름으로 시스템의 기능적 요구사항을 표현한다. 이때 프로세스는 상태를 유지하거나 데이터를 관리할 필요가 있는 경우 데이터를 자료저장소(Data Store)에 유지한다.

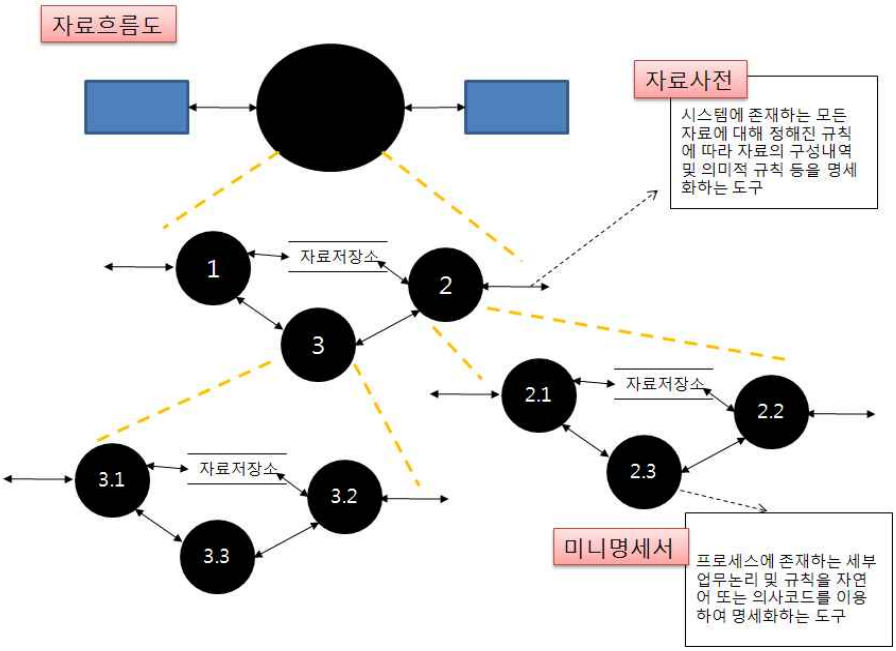
자료흐름도는 시스템을 하나의 프로세스로 표현하고 시스템과의 인터페이스 되는 다른 외부객체(External)와의 입출력관계를 표현하는 배경도(Context Diagram)를 시작으로 시스템 프로세스를 분할하여 하위 자료흐름도를 구성하여 여러 단계의 자료흐름도를 이용하여 추상화레벨을 유지한다.

부가적으로 프로세스의 입출력되는 자료흐름(Data Flow)과 자료저장소의 세부적인 구성내역 및 포맷을 기술하기 위해 자료사전(Data Dictionary)이라 도구를 이용하여 데이터의 상세 내역을 명세화하고, 최하위 프로세스에 대한 상세 시나리오를 표현하기 위해 미니명세서(Mini-Spec)이라는 도구를 활용하여 세부적인 업무 로직을 명세화한다.

구조적 분석 기법은 요구사항을 프로세스와 프로세스의 입출력을 명확히 정의하고 명세화하는 정형화 기법으로 많이 활용되었으며, 여러 조직이나 사람간의 인터페이스가 존재하는 물리적인 시스템의 기능을 분석하고 명세화하는 경우에 매우 유용하다.

그러나 새롭게 구축하려고 하는 시스템이나 무기체계와 같이 여러 조직이나 사용자간

의 인터페이스가 많지 않은 경우는 구조적 분석 기법이 요구하는 기능 의 계층 구조를 모델링하는 것이 용이치 않으며, 객체지향(Object-Oriented) 개념이 아닌 프로시져(Procedure) 기반의 설계의 개념을 기반으로 하고 있어 무기체계 설계에 적합한 객체지향 개념의 설계와는 쉽게 맞지 않는 부분이 존재한다[3].



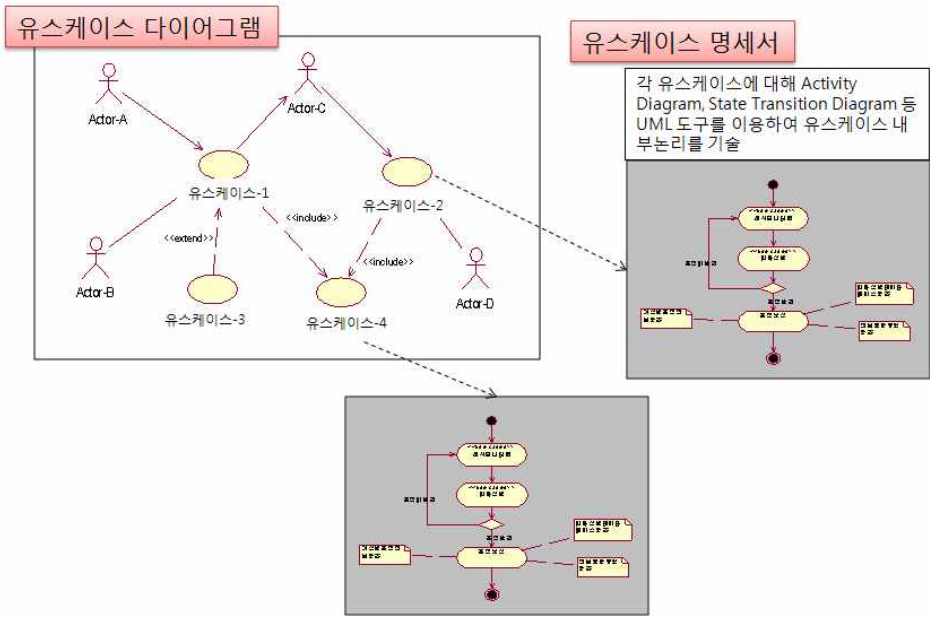
<그림 3> 구조적 분석 기법

3.2 유스케이스 모델링

유스케이스 모델링은 표준화 그룹중의 하나인 OMG(Object Management Group)에서 시스템의 분석 및 설계를 위한 도구로 제정한 UML(Unified Modeling Language)의 유스케이스를 기반으로 시스템을 분석 및 명세화하는 방법이다.

유스케이스는 시스템의 쓰임새로서 액터(Actor)라는 시스템 밖에 존재하는 외부객체가 요구하는 시스템의 기능단위이다. <그림 4>는 유스케이스 모델링을 위한 유스케이스 다이어그램과 명세서를 표현한 그림이다.

유스케이스 다이어그램은 액터와 각 액터와 연관되는 유스케이스와의 관계를 나타내는 모델링도구다. 유스케이스 다이어그램은 유스케이스와 액터와의 관계를 나타내는 것 이외에는 어떠한 내용도 포함되지 않으며 유스케이스에 대한 세부적인 시나리오나 비즈니스 로직은 유스케이스 명세서에 정의한다. 유스케이스 명세서는 UML의 다른 모델링 도구나 자연어 형태로 명세화한다. 유스케이스의 시나리오를 분석 후 각 유스케이스에서 공통적으로 사용하는 유스케이스는 별도의 유스케이스로 분할하여 공통 유스케이스로 모델링하여 표현한다.



<그림 4> 유스케이스 모델링

유스케이스 모델링은 시스템의 기능적 요구사항을 유스케이스라는 기능단위로 식별하고 이를 기반으로 시스템을 분석하고 명세화함으로써 설계와의 추적성을 명확하게 하고, 재사용성을 제고할 수 있는 논리적 기능의 식별이 가능하게 한다.

그러나 시스템의 기능적 요구사항의 중요한 요소 중의 하나인 데이터에 대한 모델링을 위한 방법은 제시하지 않으며, 유스케이스라는 기능단위를 식별하는 것이 매우 중요한 작업임에도 이에 대한 방법에 대해서는 언급이 없다. 또한 유스케이스 명칭과 유스케이스 명세서만을 가지고는 요구사항에 대한 검증이 용이치 않다[6].

4. 제안 요구사항분석 및 명세기법

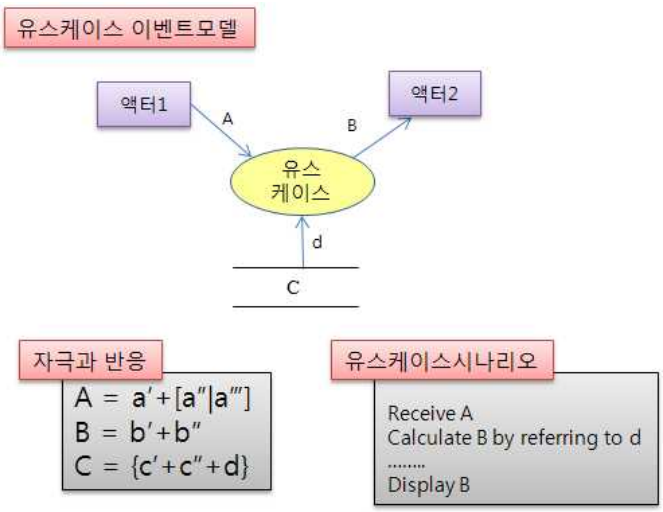
따라서 본고에서는 상술한 각 기법의 장점을 살려 재사용성 및 품질을 높일 수 있는 논리적이고 정형화된 분석 및 명세기법을 제안하고자 한다. 즉, 구조적 분석 기법의 정형화된 명세 및 데이터모델 기법과 유스케이스 모델링의 유스케이스라는 논리적인 기능단위의 장점들을 통합적으로 활용하여 유스케이스 이벤트 모델링이라는 분석 및 명세 기법을 제안하고자 한다.

유스케이스 이벤트 모델링은 크게 유스케이스의 이벤트 모델 분석, 요구사항의 검증, 요구사항 명세서 작성 작업절차에 따라 요구공학의 요구사항 도출, 분석, 검증 및 명세 업무를 효과적으로 수행하게 함으로써 재사용성 및 품질 높은 양질의 요구사항명세서를 신속하게 작성할 수 있도록 지원한다.

또한 유스케이스 이벤트 모델링은 시스템 관점이 아니라 소프트웨어 관점에서도 동일한 개념에 의해 적용할 수 있어 유스케이스의 추상화 수준을 동일한 방법에 의해 명세할 수 있도록 지원하여 체계 및 소프트웨어 형상항목의 요구사항 명세에 동일한 방법에 의해 추상화 수준 및 추적용이성을 유지하면서 요구사항명세서를 개발할 수 있도록 지원한다.

4.1 유스케이스 이벤트 모델

유스케이스 이벤트 모델은 기존 유스케이스 다이어그램을 구조적 분석 기법의 장점을 이용하여 확장한 모델로서 논리적인 시스템의 기능단위인 유스케이스를 정형화한 형태로 모델링하는 도구이다. <그림 5>는 유스케이스 이벤트모델과 관련 도구들을 보여주고 있다.



<그림 5> 유스케이스 이벤트모델 및 관련 도구

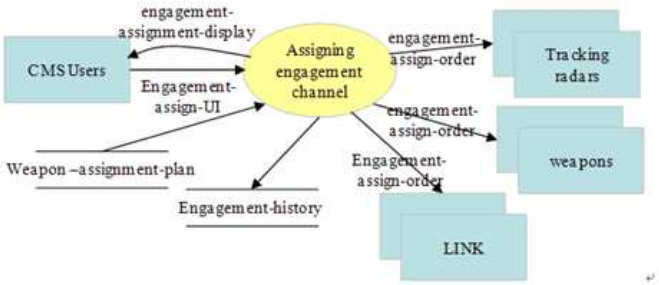
유스케이스 이벤트 모델은 유스케이스의 관련 액터와 액터와의 자극과 반응(입출력데이타) 관계를 표현한다. 각 자극과 반응은 자료사전이라는 구조적 분석 기법의 도구를 활용하여 데이터의 구성내역, 포맷 등을 기술함으로써 유스케이스의 기능을 정확하게 설명한다. 또한 자료사전은 유스케이스 시나리오를 명확하게 이해할 수 있도록 하고 내부 비즈니스 로직을 정확하게 명세할 수 있도록 지원한다.

특히 자료사전은 유스케이스의 수준을 결정하고 재사용성 등을 높이기 위한 매우 중요한 역할을 담당한다. <그림 6>은 자료사전의 표기법과 그 의미를 설명하고 있다. 자료사전에는 모든 입출력 인터페이스 및 내부 저장데이타에 대한 기능적인 내용 뿐 만 아니라 비기능적 요구사항을 모두 기입함으로써 유스케이스의 기능 특성을 정확하게 규정할 수 있다.

<그림 7>은 함정 전투체계 분석에서 유스케이스 이벤트 모델에 따라 교전채널 할당에 대한 이벤트를 처리하는 유스케이스 이벤트 모델의 예를 나타내고 있다.

A = a + b + c	A 는 a, b, c 데이터 항목으로 구성
A = $_n${a + b + c}$_m$	A 는 a, b, c 데이터 항목으로 구성되며 여러 데이터 어커런스가 존재한다는 의미 n : 최소 어커런스 수(default : 0) m : 최대 어커런스 수 (default : ∞)
A = {a + (b) + c}	b 데이터 항목은 옵션
A = {a + [b | c]}	b와 c 데이터 항목은 선택적
A = ["a" | "b" | "c"]	A는 데이터 요소로서 "a", "b", or 'c"라는 값을 가짐
A = * ----- *	A 데이터에 대한 기타 설명

<그림 6> 자료사전의 표기법



Name	Composition or Description
engagement-assign-order	= [SAM-assign-order | Gun-assign-order | SSM-assign-order | SAAM-assign-order | SLT-assign-order | ECM-assign-order | tracking-radar-assign-order | IRST-assign-order | CHAFF-assign-order | HEL-assign-order]
Engagement-assign-UI	= * the user interface for assigning engag
engagement-assignment-display	= * the text display for assigning CMS us
engagement-history	= * the log data for recording engagement

Scenario Description

- Receive engagement order through [engagement-assign-UI]
- Send [engagement-assign-order] for target tracking by referring to [weapon-assignment-plan]
- Send [engagement-assign-order] for weapon by referring to [engagement-plan]
- Display [engagement-assignment-display] for console by referring to [weapon-assignment-plan]
- Send [unit-engagement-order] for other units by referring to [weapon-assignment-plan]
- Create [engagement-history]

<그림 7> 유스케이스 이벤트 모델의 예

상기의 예는 교전채널을 할당해야 하는 이벤트를 중심으로 전투체계 사용자, 센서 및 무장과 같은 관련 액터를 식별하고, 이벤트에 따른 자극과 반응의 관계를 통해 논리적인 기능단위로서의 교전채널할당이라는 체계수준의 유스케이스를 식별하고, 이 유스케이스의 입출력에 대한 정의와 내부 시나리오를 기술하여 정확한 요구사항을 명세화하는 것을 보여주고 있다.

구조적 분석 기법에서는 상술한 예와 같은 논리적인 기능단위를 도출하기 위해서 전투체계 전체를 하나의 프로세스로 모델링한 배경도로부터 하위 프로세스를 분석하여 최하위 프로세스를 도출해야 한다. 이러한 방법으로 하위 프로세스를 도출하는 것이 쉽지

않으며 기능의 분할이 분석자의 경험에 의해 다르게 도출될 수 있다. 유스케이스 모델링의 경우는 기능 계층구조를 요구하지는 않으나 유스케이스를 도출하는 방법을 제시하지 않고 있어 논리적인 기능단위를 동일한 개념으로 도출할 수 없는 문제점을 안고 있으며, 유스케이스의 시나리오만으로 정확한 유스케이스의 입출력관계를 파악하기 어려워 요구사항의 검증이 용이치 않다는 문제점이 있다.

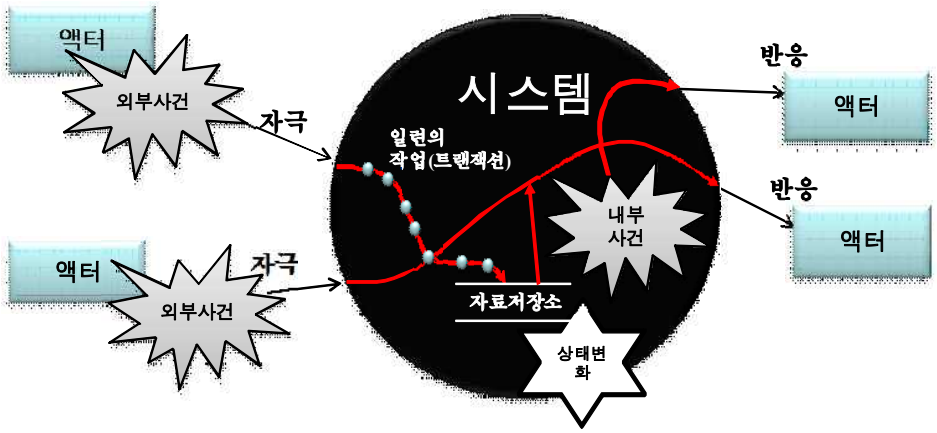
4.2 유스케이스의 식별 기법

요구사항 분석의 가장 중요한 첫 번째 행위는 시스템의 범위를 결정하고, 시스템이 수행할 유스케이스를 식별하는 것이다. 이는 시스템을 사용하거나 인터페이스 해야 할 외부객체(사람, 장치, 타 시스템)들을 결정하고, 외부객체로부터 시스템에 자극을 유발하는 이벤트와 내부에서 발생하는 이벤트를 분석함으로써 쉽게 도출이 가능하다.

외부객체를 식별하는 방법은 구조적 분석 기법이나 유스케이스 모델링에서의 액터를 식별하는 것과 동일한 방법에 의해 이루어진다. 따라서 이벤트를 식별하고 이 이벤트에 대한 시스템이 반응해야 할 트랜잭션을 도출하면 이 트랜잭션이 시스템의 유스케이스가 되는 것이다.

이벤트는 액터들이 특정 시점에 시스템을 필요로 하거나 시스템에 입력을 해야 하는 시스템 외부에서 발생하는 어커런스(Occurrence)로서 시스템에 주는 자극으로 구체화된다. 전투체계의 예를 들면 전투체계 사용자가 임무계획을 수립해야 하거나, 탐색레이타로부터 표적이 발생하는 것을 의미한다. 이벤트는 외부객체로부터 발생이 될 수도 있지만, 특정 시점에 시스템 내부에서 발생하는 어커런스와 같은 내부적으로 발생하는 이벤트도 존재한다.

이러한 이벤트의 발생으로 외부객체가 시스템에 가하는 자극은 구체화된 데이터형태로 입력되며, 시스템은 이 자극(입력데이타)에 대해 일련의 작업을 수행하고 외부객체로 반응(출력데이타)을 보이거나 내부 시스템의 상태(자료저장소)를 변화시키게 된다. <그림 8>은 이러한 이벤트 발생, 이벤트에 따른 자극과 시스템이 반응하는 개념을 표현한 것을 보여주고 있다.



<그림 8> 이벤트와 이벤트에 따른 시스템 동작모형

이벤트의 식별을 통해 자극과 반응을 식별하면 유스케이스의 식별이 이루어 진다. 따라서 이벤트 수준을 결정하는 것이 유스케이스의 수준이 되며, 이벤트의 수준을 결정한다는 것은 이벤트에 따른 자극과 반응이라는 데이터들의 수준을 결정함으로써 결정될 수 있다. 즉, 유스케이스의 수준은 시스템의 입력 및 출력 데이터를 분석하여 추상화 및 상세화를 통해 결정되며 이를 통해 요구사항의 수준, 가변성 및 공통성을 조절할 수 있는 방법을 제공하게 된다.

예를 들어 전투체계의 경우 플랫폼에 따라 표적을 탐지하는 센서들이 다양한 형태로 다르게 존재한다. 각 센서로부터 표적이 발생하는 것은 각 센서 액터로 부터의 단일 이벤트로 간주할 수 있다. 따라서 A센서로부터 표적이 발생하여 표적정보가 시스템에 자극으로 전달되며, 이를 처리하기 위한 유스케이스로서 A센서의 표적을 처리하는 유스케이스가 존재할 것이다. 그러나 이러한 유스케이스는 전투체계 관점에서 별개의 유스케이스로 처리해도 가능하지만 논리적인 하나의 기능단위로 식별한다면 너무 많은 유스케이스가 존재하고 시스템 수준에서의 유스케이스로는 너무 상세한 유스케이스들이 되어 버린다. 또한 플랫폼마다 다른 센서들의 조합을 구성해야 하는 경우 이들을 선택적으로 구성해야 하므로 요구사항의 재사용성 관점에서 적합하지 않게 된다.

따라서 이러한 경우는 유스케이스는 하나로 식별하되 들어오는 자극과 반응을 자료사전의 선택적 구성내역을 이용하여 표현함으로써, 시스템수준의 유스케이스를 도출하고, 가변성에 쉽게 대응할 수 있도록 구성할 수 있다. <그림 9>는 이러한 자극과 반응의 추상화 레벨을 조정하여 시스템 수준의 유스케이스와 다양한 가변적인 요소에 대응할 수 있다는 개념을 설명하고 있다.

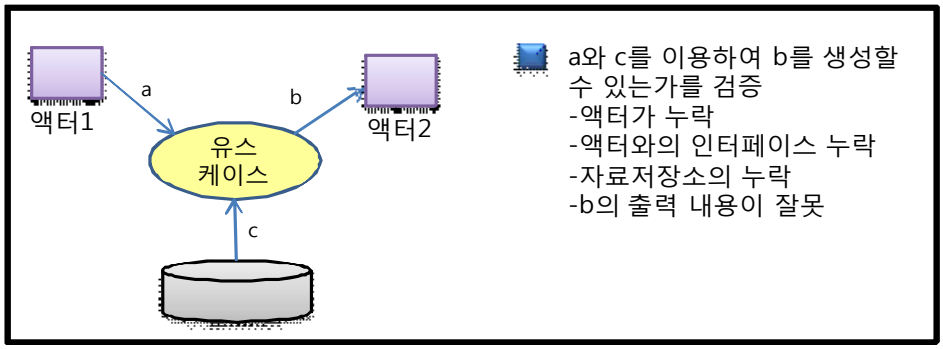
<그림 9> 자극과 반응의 추상화

상기 그림은 액터로부터의 자극과 반응을 자료사전을 통해 추상화하여 이벤트의 수준을 조정하고, 결국 유스케이스의 수준을 조정할 수 있다는 것을 의미한다. 이렇게 자료사전을 이용하여 자극과 반응을 추상화함으로써, 유스케이스의 재사용성을 높일 뿐만 아니라 시스템 구성요소(시스템, 서브시스템, 컴포넌트 등)의 수준에 따라 논리적인 기능단위를 균형 있게 도출하고 유스케이스 이벤트 모델을 통해 정확하게 명세화할 수 있도록 한다.

4.3 요구사항의 검증방법

유스케이스 이벤트 모델을 유스케이스의 이름과 시나리오 뿐만 아니라 유스케이스의 입출력관계를 자료사전을 통해 정형적으로 명세화하게 함으로써 요구사항 분석결과에 대한 검증을 수행할 수 있다.

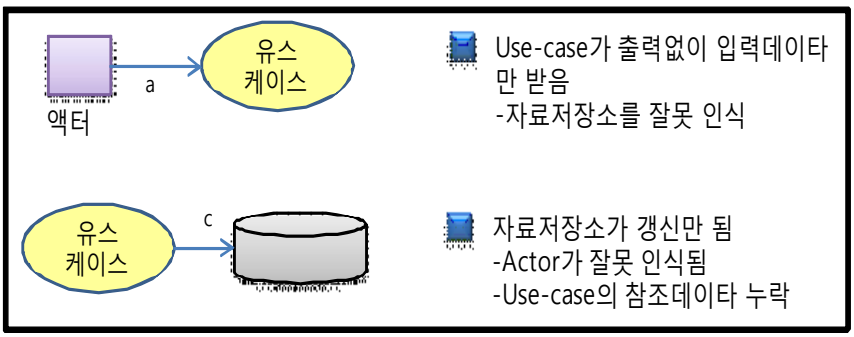
첫째, 자료보전법칙이라고 부르는 검증방법으로 반응을 생성하기 위해 입력되는 데이터는 충분히 존재하는가를 검증하는 것이다. 이는 액터의 식별이 누락되었거나 액터로부터의 이벤트 식별이 충분하지 않다거나, 참조해야 할 자료저장소의 식별이 누락되었거나, 반응의 내용에 결함이 존재한다거나 하는 오류들을 발견할 수 있도록 한다. <그림 10>은 자료보전법칙에 의한 검증 예이다.



<그림 10> 자료보존 법칙의 예

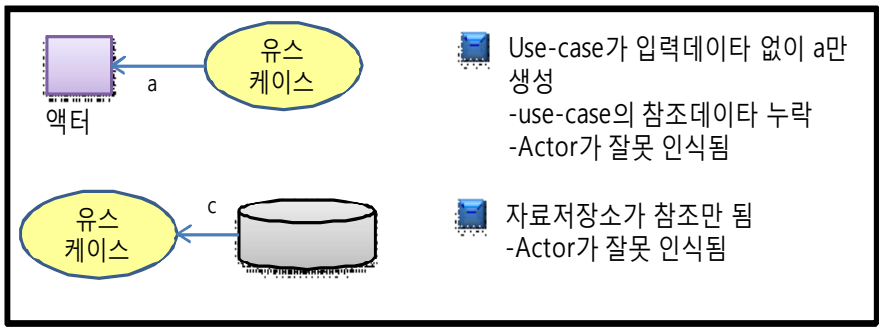
둘째, 균형법칙이라고 부르는 검증방법으로 유스케이스의 입출력 관계, 자료사전, 유스케이스 시나리오를 검증하여 유스케이스 이벤트 모델을 정의하는 명세서들이 상호 일치하는지 검증함으로써 요구사항의 일관성 및 완전성을 검증할 수 있다. 예를 들어 유스케이스 이벤트 모델에 입력되는 자극이 존재하나 자료사전에 정의가 되어 있지 않거나, 유스케이스 시나리오에 외부에 출력되는 반응이 존재하나 유스케이스 이벤트 모델에는 누락이 되었다는 것은 요구사항의 명세가 잘못되었거나 요구사항을 명확히 이해하지 못하고 있는 반증일 수 있다.

셋째, White Hole이라고 부르는 검증방법으로 자극이 없이 반응을 보이는 유스케이스 또는 참조만 되는 자료저장소는 없는지 검증하는 것이다. 이는 유스케이스의 자극이나 참조하는 자료저장소가 누락되었거나, 액터를 잘못 식별하는 결함을 발견할 수 있는 아주 유용한 검증방법이다. <그림 11>은 White Hole에 의한 검증 예이다.



<그림 11> White-Hole의 예

넷째, Black Hole이라고 부르는 검증방법으로 자극만 존재하고 반응이 없는 유스케이스 또는 참조되지 않고 갱신만 되는 자료저장소는 없는지 검증하는 것이다. 이는 While Hole과 같이 유스케이스와 액터, 자료저장소 들을 잘못 식별하는 결함을 발견할 수 있다. <그림 12>는 Black Hole에 의한 검증 예이다.



<그림 12> Black-Hole의 예

상기의 검증방법을 통해 요구사항을 잘못 인식하거나, 누락되거나, 논리적이지 않거나, 시스템의 개발범위를 잘못 인식하는 등의 결함들을 발견함으로써 정확하고 재사용성이 가능한 양질의 요구명세서를 도출할 수 있다.

4.4 요구사항명세서의 완성

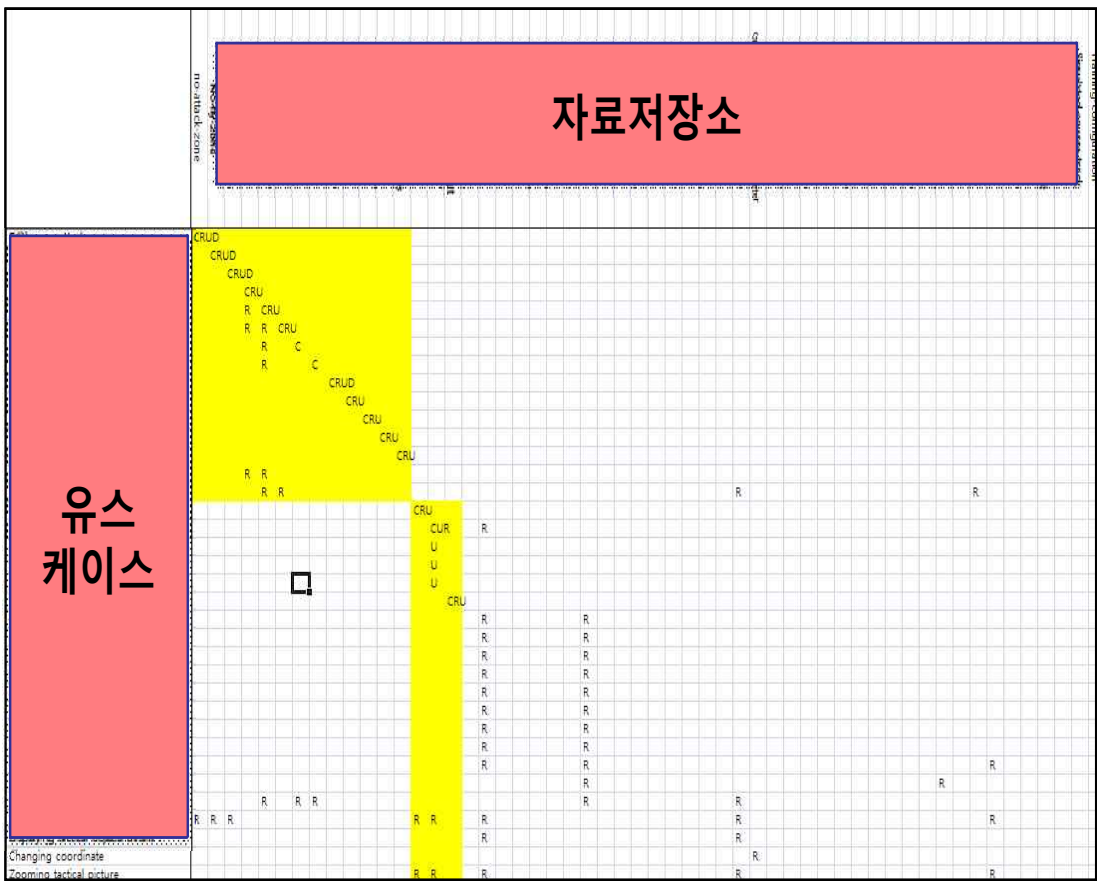
요구사항명세서는 시스템 요구분석을 통해 분석한 내용을 특정 구현 방법에 의존적이지 않게 정형화된 형태로 기술한 공식적인 문서로서, 기능적 요구사항의 명세 관점에서 시스템에 대한 능력 요구사항(Capability Requirements), 외부 및 내부 인터페이스(Interface Requirements), 내부 저장 데이터(Data Requirements)에 대한 내용을 기술하는 것이 핵심이다.

능력 요구사항은 본고에서 제안한 유스케이스 이벤트 모델을 이용하여 기술함으로써 시스템의 재사용 가능한 논리적이며 정형화된 기능 단위를 기술할 수 있게 된다. 이때 유스케이스를 논리적인 그룹으로 그룹화하여 상위 목차를 구성할 수 있는데, 이는 유스케이스 패키지를 식별함으로써 상위 목차를 구성하는 형태로 명세화한다. 유스케이스 패키지(유스케이스의 논리적인 그룹을 유스케이스 모델링에서는 유스케이스 패키지라고 함)를 구성하는 방법도 직관적인 방법이 아니라 유스케이스의 입출력 상관관계를 고려하여 유스케이스 패키지의 응집도(Cohesion)는 강하고, 결합도(Coupling)를 약하게 구성해야 한다. 이는 유스케이스 패키지가 향후 CSCI(Computer Software Configuration Item) 또는 CSC(Computer Software Component)의 기반이 되며, 재사용의 단위로 참조될 수 있도록 구성될 수 있기 때문이다.

재사용 가능한 논리적인 유스케이스 패키지를 구성하는 방법은 유스케이스와 유스케이스가 참조하는 자료저장소간의 CRUD(Create, Read, Update, Delete) 관계 테이블을 구성하여 결합도가 강한 유스케이스 그룹을 패키지로 구성함으로써, 유스케이스 패키지를 향후 설계에 단계 활용하며, 요구사항의 재사용성을 제고할 수 있도록 지원한다. <그림 13>은 함정 전투체계 분석 시 적용한 유스케이스 패키지를 구성한 예를 보여주고 있다.

본 CRUD 관계 테이블 통해 함정 전투체계의 논리적인 패키지를 도출하고, 또한 유스케이스와 자료저장소 간에 상호참조에 의한 검증을 통해 누락되거나 변경되어야 할 자료 및 유스케이스를 추가로 도출함으로써 논리적이고 완전한 명세서를 구성하게 된다.

자료저장소간의 결합도가 높은 자료저장소를 같이 배열을 하고, 자료저장소를 생성하는 유스케이스들을 대각선으로 배열하고, 유스케이스 패키지 간에 가장 결합도가 약한 그룹을 <그림 13>의 테이블 내부 사각형처럼 식별함으로써 논리적인 유스케이스 패키지를 식별하고 이를 능력 요구사항의 상위목차로 구성한다.



<그림 13> 유스케이스 패키지 구성을 위한 CRUD 테이블의 예

외부 및 내부 인터페이스, 내부 데이터 요구사항은 유스케이스 이벤트 모델을 이용하여 바로 명세화할 수 있다. 외부 인터페이스는 유스케이스 이벤트모델에서 식별된 엑터를 중심으로 자극과 반응에 대한 자료사전을 활용하여 기술함으로써 완성될 수 있으며, 내부 인터페이스 역시 유스케이스 이벤트 모델에서 식별된 유스케이스 간의 자료흐름의 자료사전을 사용하여 바로 명세화하면 된다. 또한 내부 데이터 요구사항은 유스케이스 이벤트 모델의 자료저장소의 자료사전을 이용하여 명세화한다.

5. 결론

요구사항명세서는 사용자의 요구사항을 분석하여 공학적인 관점에서 논리적이고 정형화된 형태로 명세화함으로써 향후 시스템 설계 및 개발의 가장 기반이 되는 정보를 제공하여, 재사용성 및 품질을 제고할 수 있는 기초가 되어야 한다.

요구사항명세서를 단순히 사용자가 요구한 요구사항을 정리하고 조직화하는 정도로 명세화하는 경우 재사용성 및 품질 제고를 위해 요구사항명세서가 갖추어야할 논리성, 완전성, 일관성, 테스트용이성 및 추적용이성과 같은 특성을 만족하지 못하여 전체 시스템 품질에 영향을 주게 된다. 이를 위해 다양한 분석 및 명세방법을 사용하고 있으나 대표적인 분석 및 명세기법 들의 문제점이 존재하고 있다.

본고에서는 이러한 단점을 극복하고 기존 기법들의 장점을 이용하여 재사용성 등의 품질을 제고하기 위한 분석 및 명세기법을 제안하였다. 제안 기법은 유스케이스 이벤트 모델이라는 기능단위를 관련 사람들 간에 동일한 관점에서, 논리적이면서 변경에 용이한 요구사항 단위를 도출하는 방법을 제시하였다. 또한 제안 기법은 입력과 출력 데이터를 명확히 정의하고 이에 따른 시나리오와 검증방법을 통해 정확한 요구사항을 분석 및 명세화하는 방법을 제시함으로써 완전성 및 일관성을 유지하면서 테스트용이성을 증대한다. 또한 데이터 중심 및 모듈화 개념을 적용하여 향후 객체지향 개념에서의 설계 및 개발을 지원하도록 함으로써 추적용이성 및 재사용성을 극대화할 수 있도록 지원한다.

그러나 사업 진행상 요구분석에 요구되는 시간이 부족한 경우나 소프트웨어 수명주기 모델이 진화적 방법을 사용하는 경우, 제안 기법이 요구하는 모든 사항을 명세하기 어려운 사업상의 문제점이 발생할 수 있다. 이러한 경우 논리적인 기능 단위인 유스케이스는 제안한 방법에 따라 식별하고, 자료사전 또는 유스케이스 시나리오는 주어진 시간 내에서 추상화된 수준으로 명세화한 후 상세한 내용은 설계 이후 또는 다음 빌드 일정에 맞추어 분석함으로써 해결할 수 있을 것이다.

참고문헌

[1] 고순주, 박도현, 함정전투체계의 해외 기술동향 및 국내 발전추세에 대한 고찰, 『한국방위산업학회지』, 제16권 제2호, 2009.12, pp.237-268.

[2] Davis, Alan M, *Software Requirements: Objects, Functions, and States, Second Edition*, *Prentice Hall*, 1993, pp.47-50.

[3] FAA, "Structured Analysis and Formal Methods", *FAA System Safety Handbook, Appendix D*, December 30, 2000, pp.D1-D7

[4] G. Spanoudakis, A. Finkelstein, and D. Till, "Overlaps in Requirements Engineering", *Automated Software Engineering*, Vol. 6, No. 2, 1999, pp.171-198.

[5] Gansen Dharmalingam and Knodel Jens, "Identifying Domain-Specific Reusable Components from Existing OO Systems to Support Product Line Migration", *Proceedings of the First International Workshop on Reengineering Towards Product Lines*, March, 2006, pp.27-36.

[6] Grinz, Martin, "Problems and Deficiencies of UML as a Requirements Specification Language", *IWSDD*, '00, 2000, pp.11-22.

[7] Lieberman, Ben, "The Art of Modeling", *www.therationaledge.com*, August, 2003, pp.7-9.

[8] Yourdon, Edward, *Modern Structured Analysis, Prentice Hall*, 1989, pp.25-27.