T-SASD : 추적성을 제공하는 SASD 개발 프로세스의 제안

(T-SASD : Traceable SASD Development Process)

서 영 주 (Youngju Seo) 은 애 천 (Ae-cheoun Eun)

요 약 SASD (Structured analysis and structured design)은 시스템의 분석과 디자인을 돕기 위한 탑-다운 접근법의 일종으로, 사용자들의 이해를 돕기 위한 기호 체계를 사용하여 시스템을 단계별로 분석 및 디자인할 수 있게 해주는 기법이다. 사용자는 이 기법을 통해 구성하고자 하는 시스템의 전반적인 설계 및 개발에 대한 체계적인 개발을 할 수 있다. 본 논문에서는 SASD에 RM+라고 하는 요구사항 관리 와 추적성을 제공해주는 프로그램을 적용하는 것으로 요구사항 분석 단계부터 디자인, 구현 및 테스트 단계까지의 추적성을 확인할 수 있는 확장된 SASD 개발 프로세스를 제안한다. 사례 연구에서는 본 논문에서 제안하는 개발 프로세스를 따라 프로그램을 개발하는 과정에서 어떻게 요구사항부터 테스팅까지 각 단계들이 수행되며 단계별 추적성이 어떻게 제공되는지를 보여줄 것이다.

키워드 : **소프트웨어 엔지니어링, 추적성, 구조적 분석 방법론**

1 서 론

소프트웨어 공학 기법 중 하나인 SASD (Structured analysis and structured design)[1, 2]는 시스템을 기능 계층으로 표기하는 소프트웨어 공학 방법론으로써 시스템의 분석과 디자인을 돕기 위한 탑-다운 접근법을 취하며 화살표, 도형 등 사람의 이해를 돕기 위한 기호들을 사용하여 이를 통해 사용자가 시스템을 개발하는 것을 돕는 기법이다. 이 기법을 통해 개발의 질을 높이고 시스템 고장의 위험성을 줄일 수 있으며 실제 구현 시 필요한 개발 문서를 작성할 수 있다.

본 논문에서는 시스템의 분석 및 디자인 단계에 적용하는 SASD 기법에 요구사항 분석 단계에서 사용할 RM+라고 하는 도구를 이용하여 요구사항 분석 단계부터 디자인, 구현 및 테스트 단계까지의 추적성을 확인할 수 있는 확장된 SASD 개발 프로세스인 T-SASD 를 제안하며 각 기법, 도구에 대한 설명과 이들 개별 적용의 한계점을 설명하고 T-SASD 에서 어떻게 이를 보완할 수 있는지 T-SASD 의 개발 프로세스의 적용을 사례 연구를 통해 보일 것이다.

논문의 구성은 다음과 같다. 1 장의 서론에 이어 2 장에서는 본 개발 프로세스에 사용된 RM+와 SASD 의 설명 및 각각의 적용 방법에 대해 설명하고 개별 적용의 한계점에 대해 설명하겠다. 그리고 3 장에서 RM+와 SASD 를 결합한 T-SASD 의 제안 및 이의 적용에 대한 사례 연구를 보이며, 마지막으로 4 장에서 결론을 내린다.

2 관련연구

2.1 RM+

소프트웨어 공학에서 요구사항의 관리는 중요한 문제로 이미 여러 논문들에서[3, 4, 5] 요구사항의 관리나 추적성에 대해서 다루고 있다. 이는 요구사항 관리와 추적성이 요구사항을 기반으로 하는 시스템의 개발 환경에서 얼마나 중요성에 대하여 말하고 있다. 추적성의 중요성은 요구사항이 개발 중에도 계속하여 바뀌기 때문에 나타난다. 이는 요구사항이 도출된 후에도 개발 과정을 지속적으로 관리하고 추적해야 하기에 더욱 중요한 작업이라 할 수 있다..

RM+는 e SQUARE 사에서 만든 요구사항 기반 통합 시스템 엔지니어링 도구로써, V모델[6] 중 요구사항 정의 및 분석, 시스템 디자인 및 시스템의 검증 등의 기능을 지원한다. 또한 이러한 과정들을 수행하며 작성하게 되는 요구사항들과 디자인, 검증사항간의 추적성을 제공하여 각 단계들이 서로 어떻게 연동되는지를 확인할 수 있게 해준다.

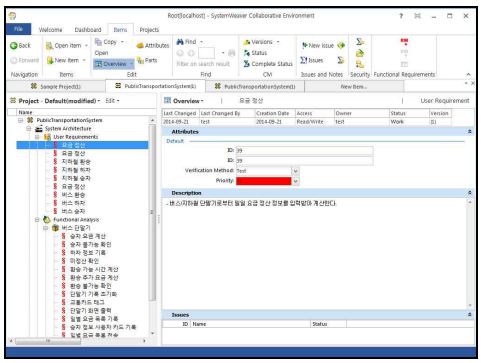


그림 1 RM+의 실행 화면

그림 2 User requirement 와 Functional requirement 간의 대응을 통한 추적성 제공

그림 1 은 RM+의 실행 화면을 보여준다. 사용자는 RM+를 이용해 User requirement를 정의할 수 있으며, 각 User requirement들에 대하여 여러 개의 Functional requirement들을 추가로 정의할 수 있다. 그림 2는 User requirement 와 이에 대한 Functional requirement들의 대응을 보여준다. 왼쪽에 나열된 버스 승차, 버스 하차, 버스 환승, 요금 정산이 User requirement를 나타내며, 가로축으로 나열된 것들이 Functional requirement들을 나타낸다.

이를 통해 하나의 User requirement가 어떤 Functional requirement들로 이루어져있는지를 확인할 수 있으며, 어느 한 쪽의 요구사항의 변경이 발생할 경우에 영향을 받는 항목들이 어떤 것들이 있는지 쉽게 파악할 수 있다. 또한 각 요구사항들에 대한 테스트 케이스의 작성과 이들의 대응도 지원되어 그림 3은 User requirement와 이에 해당하는 System test specification을 보여주고 있으며 각 test에 대한 세부 Description도 작성 가능하게 되어있다. 그림 4는 Functional requirement와 이에 해당하는 Functional test specification간의 연관관계도를 나타낼 수 있으며, 또한 세부 Description을 명시하는 기능도 내장되어있다.

또한 그림 5에서 볼 수 있듯이 요구사항들에 대한 Logical model의 design을 지원하는 기능도 갖추고 있으며. 이를 활용하면 각 Functional requirement에 따른 구조적인 설계가 가능하다.

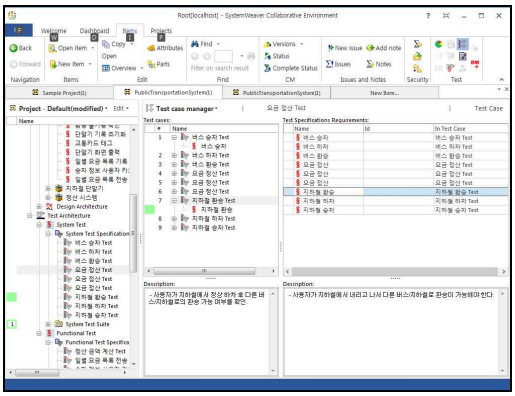


그림 3 User requirement 와 System test specification 간의 추적성 확인

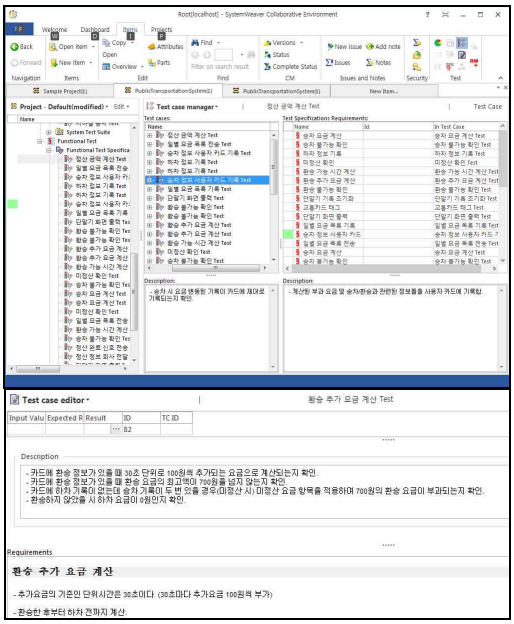


그림 4 Functional requirement 와 Functional test specification 간의 추적성 확인

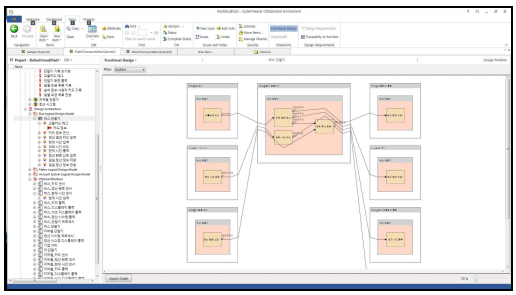


그림 5 RM+를 통해 수행한 Logical mode 의 design

2.2 SASD

SASD는 시스템을 기능 계층으로 표기하는 소프트웨어 공학 방법론으로써 탑-다운 접근법과 사용자의 이해를 돕기 위한 기호들을 이용하여 대규모의 복잡한 시스템을 개발 및 유지보수하기 쉽게 해주는 기법이다.

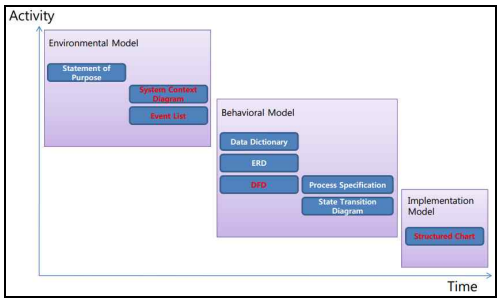


그림 6 SASD 개발 프로세스

전반적인 SASD를 통한 개발 과정은 그림 6에 나와있으며 이를 각 단계별로 보면 Environmental Model 단계에서 구현할 시스템의 범위와 외부와의 상호작용 범위를 정의한다. 이를 위해 SCD (System Context Diagram)나 Event List등을 사용한다. 이후 Behavioral Model 단계에서 모델의 내부 동작이나 시스템의 데이터 엔티티를 나타내며 이를 위해 DFD (Data Flow Diagram)와 같은 다이어그램을 사용하게 되는데, 이를 활용하여 최종적으로 Implementation Model 단계에서 모듈간의 호출 관계, 데이터의 흐름 등을 Structured chart로 표기한다.

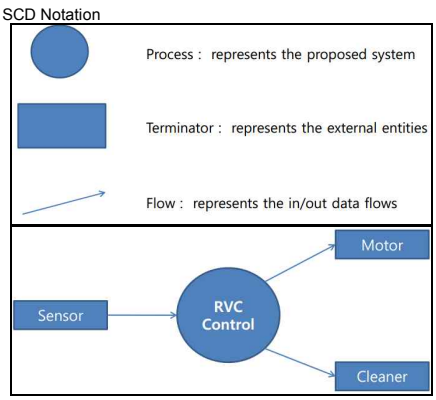


그림 7 System Context Diagram

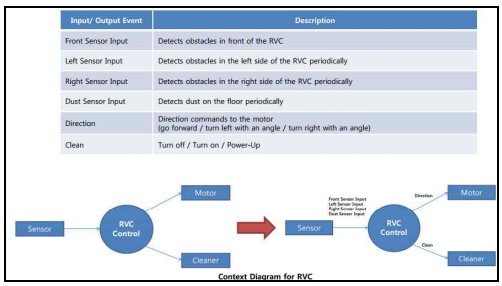


그림 8 Event List 와 Event List 의 정보를 대응시킨 System Context Diagram

Environmental Model에서 SCD와 Event List의 실제 적용 사례를 보면 그림 7, 8과 같이 SCD를 이용해 시스템의 입력부와 컨트롤, 그리고 출력부를 구분하고 Event List를 이용해 실제 시스템이 어떤 데이터를 입력으로 받으며 어떤 출력을 내보내는지를 명확히 하는 것으로 시작하여 그림 9의 DFD를 단계별로 그려 나가서 시스템의 내부 동작을 프로세스 별로 나타내고 그림 10에 나와있는 것처럼 Process Specification을 이용해 각 프로세스 내부의 동작을 설명한다.

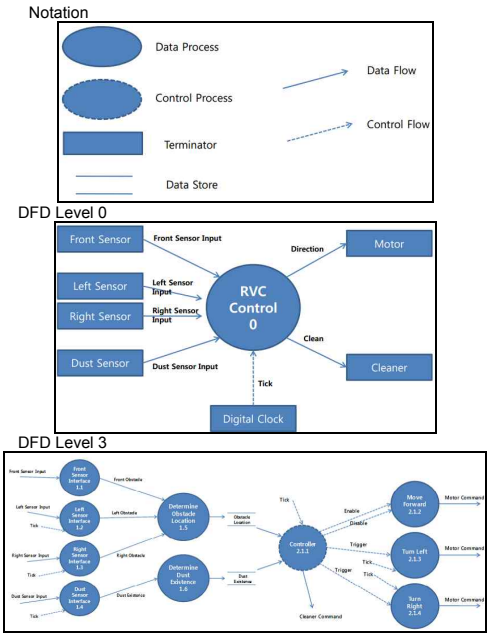


그림 9 Data Flow Diagram 의 표기법과 단계별 수행 DFD level 0 과 DFD level 3 의 결과

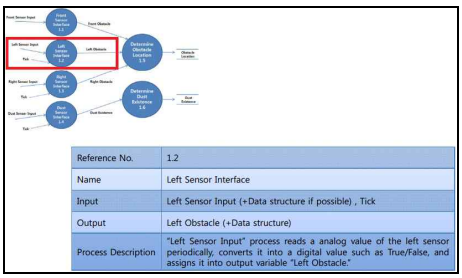


그림 10 각 프로세스별 Process Specification 을 작성한 모습

그리고 마지막으로 최종 단계의 DFD 를 그림 11 에 나와있는 Structured Chart 의 형태로 변환하는 것으로 프로세스간의 데이터 전달 및 호출 관계를 나타낸다.

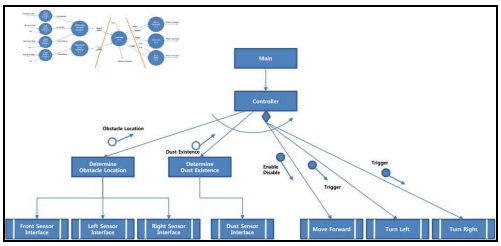


그림 11 Data Flow Diagram 을
Structured Chart 로 변환한 모습

이와 같이 SASD를 적용하여 시스템을 개발할 경우 중간 목표와 결과물이 뚜렷하여 프로젝트의 진행 상황을 확인하기 쉽고, 프로세스들의 수행과 데이터의 전달로 이루어져있기 때문에 시스템의 동작을 생각하기 쉽다는 장점이 있다.

2.3 개별 적용의 한계점

RM+의 경우 RM+만을 이용하여 개발을 수행할 경우, 요구사항으로부터 Logical Design을 작성할 때 현재 RM+에서 수행하는 방식대로 요구사항의 상세 설명만을 보고 작성하기에는 부족함이 있다. 특정한 프로세스를 따라 모델을 디자인하는 것이 아니라서 모델을 디자인하는 개발자의 실력에 따라 만들어지는 Logical Design의 질적 차이가 클 수 있다. 또한 사용되는 데이터의 표현이나 구조화에 대한 표기 방법의 지원이 미흡하다는 등의 문제점이 존재한다. 따라서 직접 코드를 작성할 수준의 구체화 된 디자인 문서를 생성하는 데는 부족함이 있다.

SASD는 이미 오래 전부터 사용되어온 방법론으로 개발을 수행하는 데에는 충분하지만 요구사항이 이미 존재하여 이를 기반으로 개발을 수행하는 것이므로 요구사항의 관리, 도출에 대한 부분이 고려되어있지 않으며, 많은 다이어그램과 리스트 등의 문서가 생성되므로 이전의 요구사항들에 대한 추적성의 제공이 어렵다. 따라서 본 논문에서는 RM+의 부족한 디자인의 구체화와 SASD에서 부족한 요구사항의 도출 지원과 추적성을 보완하기 위하여 이 둘을 서로 합쳐서 부족한 부분을 보완할 수 있는 T-SASD라고 하는 새로운 개발 프로세스를 제안한다.

3 T-SASD의 제안

3.1 T-SASD

T-SASD 는 RM+의 부족한 디자인의 구체화와 SASD 에서 부족한 요구사항의 도출 지원과 추적성을 보완하기 위하여 이 둘을 병행하여 사용하는 개발 프로세스로 RM+의 장점인 요구사항의 관리 및 추적성 제공과 디자인 기능을 사용하여 시스템의 요구사항 도출 및 분석 단계를 수행하고 여기서 나온 문서들을 이용하여 SASD 의 개발을 수행하고 또한 SASD 단계에서 나온 각 DFD 와 Process specification 과 같은 문서들 다시 RM+의 요구사항들과 대응시키는 것으로 RM+나 SASD 단독으로는 제공하기 어려운 요구사항-프로세스 수준의 추적성을 제공할 수 있다.

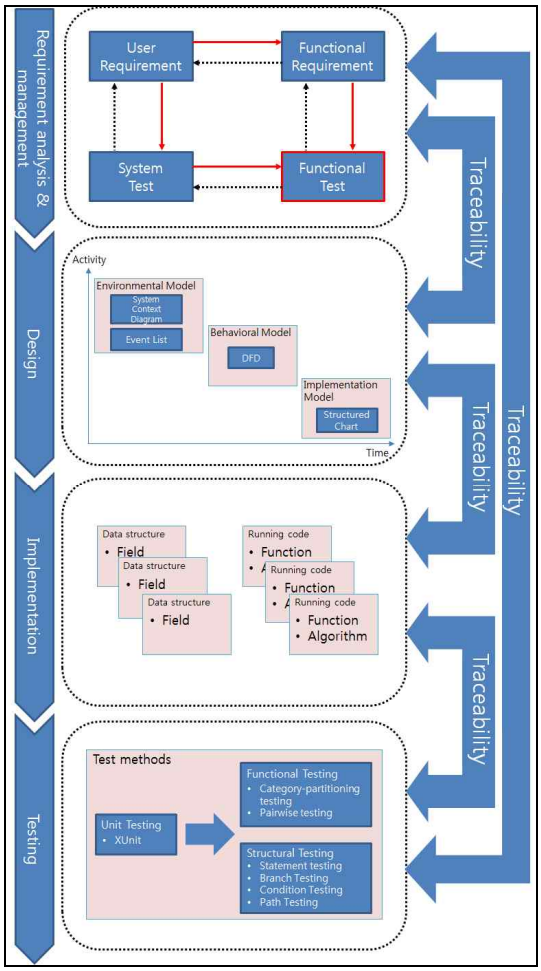


그림 12 T-SASD 프로세스

그림 12 가 전반적인 T-SASD 프로세스의 적용 과정을 보여준다. RM+를 이용한 요구사항 도출 단계에서는

- User requirement 와 Functional requirement
- User requirement 와 System test specification
- Functional requirement 와 Functional test specification
- System test specification 과 Functional test specification

각 항목들의 연관관계를 보여주고 이에 대한 추적성을 제공할 수 있으며, 요구사항들로부터 SASD 를 수행하여 SCD, Event List, DFD 와 Structured Chart 를 작성하며 이 때 RM+의 Logical Design 과 SASD 의 DFD level 0 과의 대응, Functional requirement 와 DFD level 3 과의 대응을 통해 요구사항들이 디자인 단계의 어떤 프로세스들과 대응되는지에 대한 추적성을 제공할 수 있다.

구현 단계에서는 각 소스코드들의 자료구조들과 함수들을 SASD 의 Event List, DFD level 3 의 각 프로세스들에 할당하는 것으로 요구사항, 디자인, 소스코드 간의 추적성을 제공할 수 있으며 최종 테스트 단계에서 각 소스코드 내 함수들에 대한 유닛 테스트의 작성, Functional test specification 에 대한 테스트 케이스 작성, System test specification 에 대한 테스트 시나리오 작성과 같은 방식으로 요구사항-테스트까지의 대응 관계 확인, 추적성 제공이 가능하다.

3.2 Case study

RM+와 SASD 를 결합한 프로세스를 개발 하기에 앞서 실제 실험에 이것을 적용하기 위하여 교통카드시스템을 대상으로 개발 프로세스를 진행하였다. 그림 13 과 같이 먼저 RM+에서의 4 가지 부분에 해당하는 것들을 SASD 의 기법들과 연동하여 작성하였다.

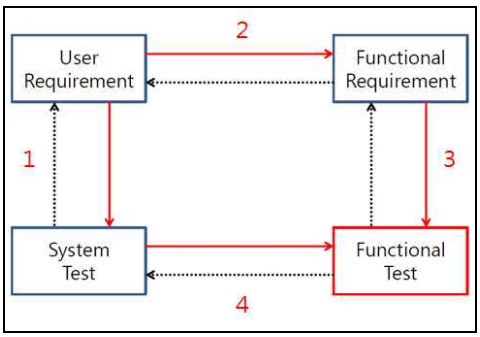


그림 13 RM+의 요구사항 관리 프로세스

그림 13 의 세부 파트에 해당하는 내용은 본 논문의 부록에 첨부되어있다. 각 번호가 부록 번호와 동일하다. 그림 14 에 나와있는 왼쪽 그림은 RM+의 Logical Design 에의한 그림이며 Logical Design 내부의 박스들은 Physical Structure 들이며, 이것은 해당 프로그램의 물리적 기능을 하는 부분을 나타낸다. 이를테면 버스 단말기의 카드 센서나, 출력 부 등이 이에 해당한다. 그리고 그림 14 의 우측에 해당되는 것이 SASD 의 DFD Level 0 에 해당되는 다이어그램이며 여기서 다루는 심볼들이 RM+의 Logical 디자인의 요소들과 1 대 1 로 대응된다.

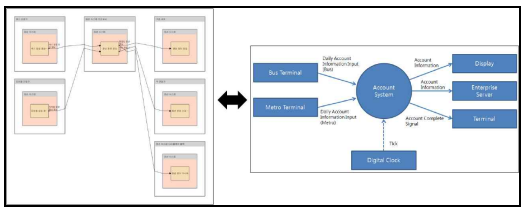


그림 14 Logical Design 과 DFD Level 0 의 대응관계

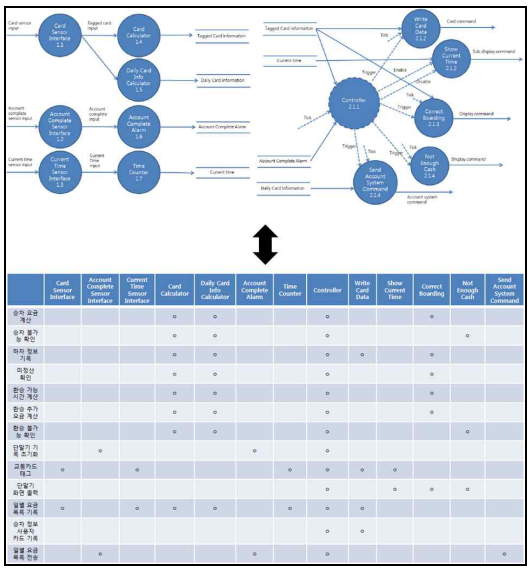


그림 15 DFD Level3 과 Functional requirement 의 관계

그림 15 는 RM+에서의 Functional requirement 가 DFD Level 3 의 각 프로세스와 어떤 식으로 대응 되는가를 보여준다. 또한 해당 프로세스와 Code 간의 관계를 나타내는 것 또한 중요한데 이 부분은 부록 5 번과 6 번에 기록된 그림에서 확인할 수 있다. 이렇게 하면 구현 부분에서의 RM+와 SASD 간의 추적성은 만족한다고 할 수 있다. 마지막으로 테스트 케이스를 커버하는 부분은 아래의 그림 16 에 나타난 표의 것과 같은 경우들을 수행하며 확인할 수 있도록 RM+에서 해당 Test case coverage 를 확인할 수 있는 기능을 아래 그림과 같이 제공하고 있다. 본 사례 연구에서는 Functional requirement 에 대한 테스트의 수행 결과만을 보여주고 있지만 실제 코드 수준에서의 유닛 테스트의 작성이나 Structural testing 의 적용 및 대응도 충분히 보여줄 수 있을 것이다.

그림 16 Test case 의 수행 결과에 대한 RM+의 추적성 및 Coverage 제공

4 Conclusion and future work

본 논문에서는 시스템을 개발하는데 있어서 중요한 요구사항의 관리와 이에 대한 추적성을 제공할 수 있는 RM+와 시스템을 디자인하기 위한 SASD 기법에 대한 설명과 각각의 한계점, 그리고 이를 보완하기 위해 두 가지를 함께 적용하는 개발 프로세스인 T-SASD 를 제안하고 이를 적용하여 간단한 시스템을 개발하는 내용을 다루었다.

더 큰 시스템의 개발에 있어서는 더 잦은 요구사항의 변경이 발생할 것이며 디자인의 질이 더욱 중요할 것이므로 본 논문에서 보여주었던 것 보다 T-SASD 를 이용한 개발 프로세스를 적용하였을 시 더 좋은 효과를 볼 수 있을 것이라 생각된다.

또한 본 논문에서 제안하는 T-SASD 는 요구사항의 관리와 관련된 부분을 RM+를 통해 제공하고 상세한 디자인을 하는 부분에서 SASD 를 적용하는 방식을 사용하고 있는데 이는 바꿔서 말하면 디자인 과정을 SASD 가 아닌 다른 기법을 적용할 수도 있다는 것으로 디자인 과정을 OOAD (Object oriented analysis and design)와 같은 다른 방식을 적용할 수도 있을 것이다. 이렇게 한다면 다른 개발 방법론을 적용하더라도 위와 같은 수준의 요구사항에 대한 추적성을 제공할 수 있을 것이다.

마지막으로 RM+를 처음 사용함에 있어 불편한 점들 및 버그가 있었는데, 처음 사용할 때 불편한 점은, 서버와 클라이언트를 실행시키는 방식이 이해가 잘 되지 않아서. 저장 기능이 있는지 없는지를 계속 찾았던 것 같다. 따로 매뉴얼에 관련 설명을 넣는다던가. 아니면 저장 기능을 추가하는 것이 좋을 것 같았다. 또한 가끔 객체를 추가할 때, 해당 객체가 생성이 되지 않는 문제점이 있었다. Specification 자체가 생성되지 않았던 경우도 있으며, 해당 Specification 의 하위에 추가가 되지 않았던 경우도 있었다. 물론 이 경우는 RM+를 다시 키면 됐었던 것 같다. 또한 생성된 Logical Design 으로부터 C 코드 등을 생성할 수 있는 시스템이 구현되면 더 좋을 것 같다. 이 정도가 RM+ Basic 버전에 사용에 대한 버그 리포트 및 후기이다.

참 고 문 헌

[1] Yourdon, Edward. Modern structured analysis. Prentice Hall PTR, 2000.

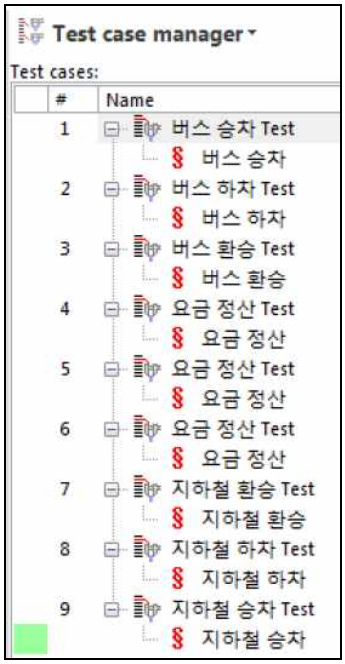
[2] Kendall, Penny A. Introduction to systems analysis and design: a structured approach. Business & Educational Technologies, 1995.

[3] Gotel, Orlena CZ, and Anthony CW Finkelstein. "An analysis of the requirements traceability problem." Requirements Engineering, 1994., Proceedings of the First International Conference on. IEEE, 1994.

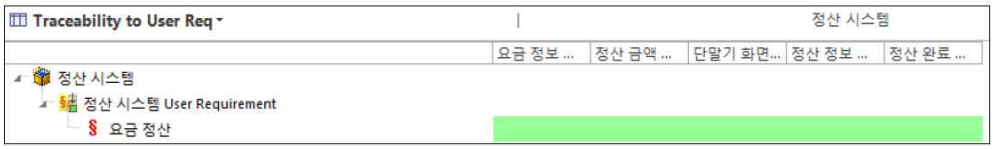
[4] Dorfman, Merlin. "*System and software requirements engineering*." IEEE Computer Society Press Tutorial. 1990.

[5] Gilb, Tom, and Susannah Finzi. Principles of software engineering management. Vol. 4. Reading, MA: Addison-Wesley, 1988.

[6] Londesbrough. I, "A Test Process for all Lifecycles", ICSTW '08. IEEE International Conference on, 2008, pages 327~331.



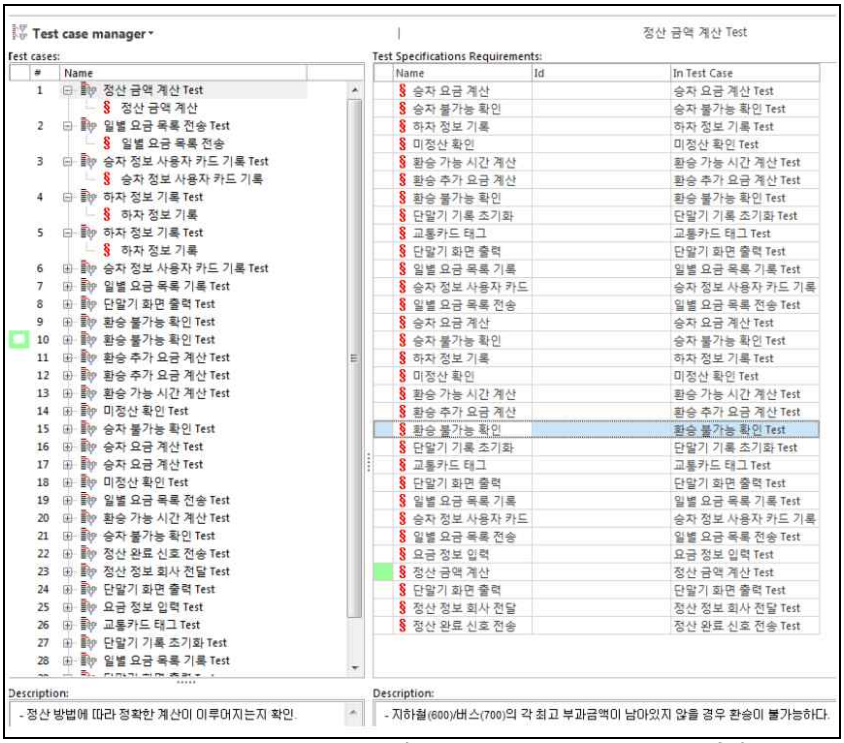
부록– 1 User requirement와 System test specification 관계



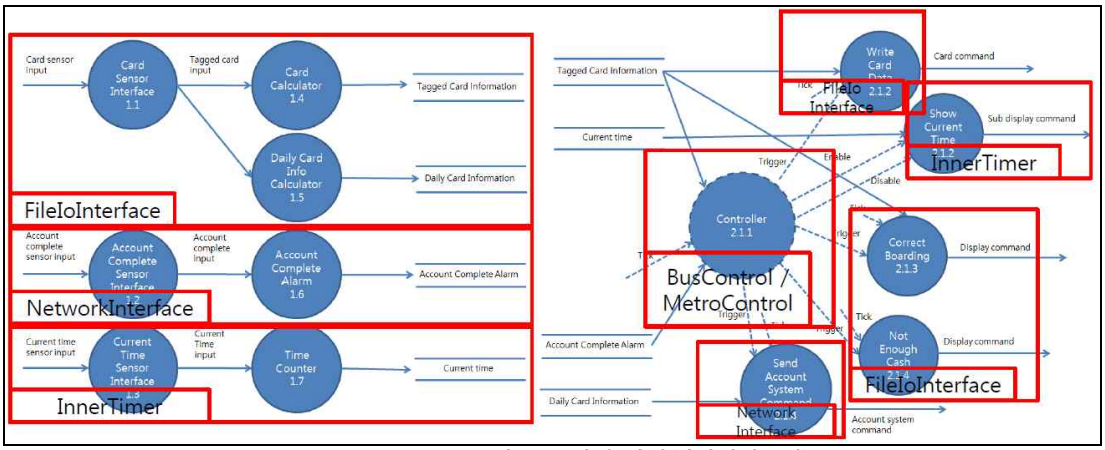
부록-2 User requirement와 Functional requirement 관계

	버스 승차	버스 하차	버스 환승	요금 정산
승차 요금 계산	○			
승차 불가능 확인	○			
하차 정보 기록		○		
미정산 확인		○		
환승 가능 시간 계산	○	○	○	
환승 추가 요금 계산	○	○	○	○
환승 불가능 확인			○	
단말기 기록 초기화				○
교통카드 태그	○	○		
단말기 화면 출력	○	○	○	
일별 요금 목록 기록				○
승차 정보 사용자 카드 기록	○	○	○	
일별 요금 목록 전송				○

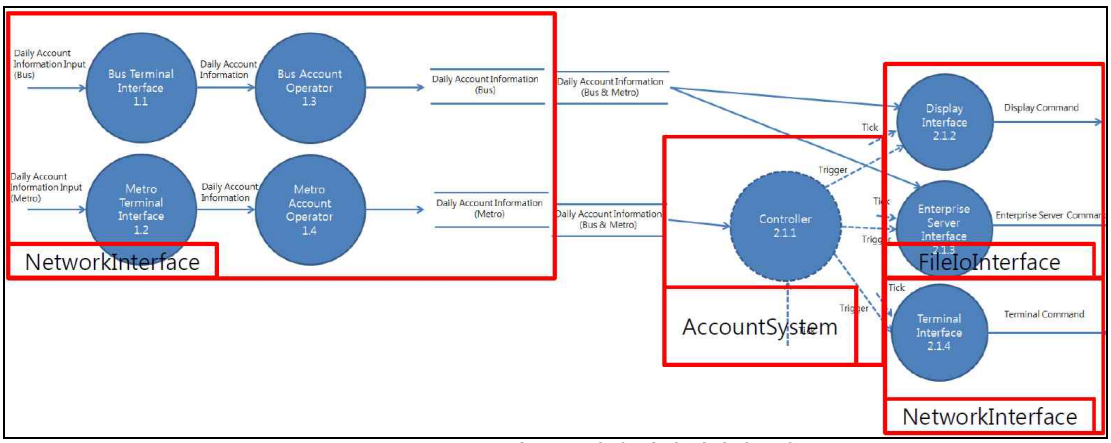
부록-3 System test specification과 Functional test specification 관계



부록-4 Functional requirement와 Functional test specification 관계



부록-5 DFD Level 3과 Code와의 관계(단말기시스템)



부록-6 DFD Level 3과 Code와의 관계(정산시스템)