KIPS Trans. Softw. and Data Eng.
Vol.7, No.9 pp.325~334 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2018.7.9.325

A Spreadsheet Application that Enables to Flexibly Change Mappings in Requirement Traceability Matrix

Serin Jeong† · Seonah Lee††

ABSTRACT

Requirement traceability should be continuously maintained in software development and evolution. However, it is usually updated in practice in the quality assurance phase. The gap between “is” and “should” exists due to the fact that developers must invest considerable effort to update requirement traceability while being able to obtain only marginal benefit from the updated traceability. To close this gap, we propose a spreadsheet application that enables developers to flexibly change mappings in a requirement traceability matrix. In this way, developers can reduce their effort in updating the requirement traceability matrix, but still obtain the common form of a requirement traceability matrix on a spreadsheet. The proposed application maintains the mappings between two artifacts on each sheet so that, whenever an artifact item changes, developers can instantly insert the relevant mapping changes. Then, when developers desire the common form of a requirement traceability matrix, the proposed application calculates the mappings among several artifacts and creates the matrix. The application also checks traceability errors and calculates the metrics so that developers can understand the completeness of the matrix. To understand the applicability of the proposed approach, we conducted a case study, which shows that the proposed application can be applied to the real project and easily incorporate the mapping changes.

Keywords : Requirement Engineering, Requirement Traceability, Traceability Matrix, RTM

요구사항 추적성 매트릭스에서 유연한 맵핑 변경을 가능하게 하는 스프레드시트 애플리케이션

정 세 린† · 이 선 아††

요 약

요구사항 추적성은 개발과 유지보수 과정 동안 지속적으로 관리해야 한다. 그러나, 실제에서는 품질 보증을 점검하는 단계에서 갱신한다. 이러한 차이는 개발자가 추적성을 갱신하는 노력에 비해 추적성을 통해 얻는 혜택이 적기 때문이다. 이러한 노력 대비 보상의 관점에서 우리는 일반적으로 사용하는 스프레드시트 형태의 요구사항 추적성 매트릭스에서 맵핑을 유연하게 바꿀 수 있는 방법을 제안한다. 제안의 목적은 개발자가 요구사항 추적성을 갱신하는데 들이는 노력을 줄이는 것이다. 제안 방법은 먼저, 각 시트에 두 산출물 간의 관계만을 기입하여 변경이 발생할 때, 개발자가 변경된 맵핑을 즉시 반영할 수 있도록 한다. 다음, 개발자가 원하는 시점에서 제안 방법은 자동적으로 모든 산출물의 관계를 계산하여 추적성 매트릭스를 생성한다. 또한 누락된 맵핑 관계를 색상으로 표시하고 척도를 계산하여 개발자가 추적성 매트릭스의 완전성을 파악하도록 돕는다. 우리는 제안 방법의 적용가능성을 파악하기 위하여 사례 연구를 수행하였다. 사례 연구는 제안한 요구사항 추적성 매트릭스가 실제 프로젝트에 적용 가능하며 변경된 맵핑 관계를 쉽게 수용함을 보여준다.

키워드 : 요구공학, 요구사항 추적성, 추적성 매트릭스, RTM

1. 서 론

요구사항 추적성은 요구사항을 구현하는 여러 산출물을 연관 짓는 능력이다[1]. 요구사항 추적성은 개발하는 시스템이 요구사항을 얼마나 잘 수용하였는지 파악하기 위해 필요한다. 개발자들은 요구사항 추적성을 수립하기 위해 요구사항 추적성 매트릭스(Requirement Traceability Matrix, 이하 RTM)를 사용한다. 추적성 매트릭스는 개별 요구사항과 다른 산출물 간의 논리적 맵핑을 기술한다. 이러한 추적성 수립과 관리를 위해서 일반적으로 간편한 형태의 스프레스시트를 사용한다. 스프레드시트 형태의 전형적인 추적성 매트릭스의 형태는 Table 1과 같다.

† 비 회 원 : 경상대학교 정보과학과 석사과정
†† 종신회원 : 경상대학교 기계항공정보공학부 항공우주및소프트웨어전공 조교수
Manuscript Received : March 7, 2018
Accepted : March 27, 2018
* Corresponding Author : Seonah Lee(saleese@gnu.ac.kr)

Table 1. A Typical Requirement Traceability Matrix [2]

Requirement	Design module	Implementation file	Test item
RE001	DE003	IM002	TE002
…	…	…	…

전형적인 추적성 매트릭스는 각 산출물에 대해서 열을 만들고 산출물 항목 간에 맵핑을 행으로 배열한다. 예로서 표1은 요구사항, 설계모듈, 구현파일, 시험항목의 열이 존재하고, 각각의 산출물 항목 ID간에 맵핑이 행으로 표현된다 [2]. 이러한 매트릭스에서는 임의의 산출물을 변경, 세분화, 추가, 삭제할 경우, 모든 산출물의 관계를 다시 맵핑 해야 한다. 그 결과, 담당자는 변경이 발생할 때마다 전체 매트릭스를 다시 작성해야 하므로 지속적인 추적성 관리에 어려움을 겪는다 [1, 2]. 또한 RTM을 위한 많은 도구[5, 6]가 개발되었지만, 이러한 도구들은 오히려 더 많은 노력이 요구된다.

우리는 흔히 사용하는 스프레드시트에서의 새로운 형식의 RTM 자동화를 제안한다. 제안하는 RTM에서는 개발자가 두 개의 산출물 간 변경 사항을 즉시 반영할 수 있도록 개별 스프레드시트에 한 쌍의 산출물 간의 맵핑을 기입하도록 하였다. 이후, 필요한 시점에서 맵핑시트의 집합을 바탕으로 전형적인 추적성 매트릭스 형태를 자동 생성할 수 있다. 우리가 제안하는 추적성 매트릭스는 맵핑 관계 분할을 통해 변경에 대한 반영이 용이하고, 전형적인 추적성 매트릭스를 자동으로 생성해 줌으로써 추적성 생성의 노력을 줄인다. 또한 일관되지 않은 추적성 링크를 색상으로 표시하여 사용자가 이슈를 쉽게 발견할 수 있도록 한다.

본 논문에서 제안하는 RTM이 변경 사항에 대해 유연하게 대처할 수 있는지 확인하기 위해 사례연구를 수행한다. 이를 위해 오픈소스 프로젝트인 네브마인에 대해 초기 RTM을 작성하였다. 다음으로 이후에 발생한 변경 사항을 RTM에 반영하여 이를 유연하게 대처할 수 있는지를 확인하였다. 제안하는 RTM에서는 사용자가 변경과 관련된 맵핑 관계만을 수정하면 원하는 시점에서 해당 변경 반영 및 기존의 맵핑 관계가 유지된 RTM이 자동 생성된다. 또한 사용자의 검토가 용이하도록 변경 및 누락이 발생된 맵핑 관계를 색상으로 표시한다. 그 결과, 본 논문에서 제안하는 RTM이 변경 사항에 대해 보다 유연하게 대처할 수 있음을 확인하였다.

본 논문의 구성은 다음과 같다. 먼저 2절에서는 요구사항의 변경 타입을 논의한다. 다음 3절에서는 제안 방법을 설명한다. 4절에서는 제안 방법을 적용한 예제를 보인다. 5절에서는 관련 연구를 논의한다. 6절에서는 결론과 향후 연구 방향을 논의한다.

2. 요구사항의 변경 타입

기존 논문 [3, 4]은 요구사항의 변경 타입을 제시한다. 그 중 맵핑 변경과 관련된 변경 타입은 다음과 같이 일곱 가지로 정리할 수 있다.

(1) 생성[create]: 새로운 요구사항을 생성한다.
(2) 세분화[refine]: 하나의 요구 항목의 의미를 유지한 채 추가적인 요구사항으로 상세화한다.
(3) 분할[decompose]: 하나의 요구 사항을 여러 요구사항으로 분할한다.
(4) 통합[merge]: 여러 요구 사항을 하나의 요구 사항으로 합친다.
(5) 대체[Replacement]: 하나의 요구 항목을 폐지하고 다른 요구 사항으로 바꾼다.
(6) 삭제[delete]: 기존 요구사항을 삭제한다.
(7) 비활성화[Inactivate]: 요구사항을 비활성화 한다.

우리는 상기 일곱 가지의 변경 타입이 RTM에 존재하는 모든 산출물 항목에서 발생할 수 있다고 본다. 이러한 관점에서 맵핑된 두 항목 X:Y 대해 항목 X에 변경 사항이 발생할 경우, 다른 항목 Y에서 발생할 수 있는 맵핑 변경의 경우의 수를 Table 2와 같이 정리하였다. 각 경우의 수에 대한 설명은 다음과 같다.

Table 2. Mapping Changes According to Item Changes

Item of X	Ex		Change of Mapping
	X	Y	
create	Add X	Maintain Y	Mapping with existing item
	Add X	Add Y	Create new item
refine	Maintain X	Maintain Y	Maintain existing item
	Add X-C1	Maintain Y	Mapping with existing item
	Add X-C2	Add Y	Create new item
decompose	Delete X	Delete Y	Remove existing item
	Add X-C1	Maintain Y	Mapping with existing item
	Add X-C2	Add Y	Create new item
merge	Delete X1	Maintain Y	Maintain existing item
	Delete X2	Delete Y	Remove existing item
	Add X	Maintain Y	Mapping with existing item
	Add X	Add Y	Create new item
Replacement	Delete X	Maintain Y	Maintain existing item
	Delete X	Delete Y	Remove existing item
	Add X1	Maintain Y	Maintain existing item
	Add X1	Add Y	Create new item
delete	Delete X	Maintain Y	Maintain existing item
	Delete X	Delete Y	Remove existing item
Inactivate	Maintain X	None Y	None mapping

(1) 생성: X항목이 생성되면, 그에 맵핑되는 Y항목은 기존에 있을 수도 있고 새로운 항목을 생성하여 X 항목과 맵핑할 수 있다.
(2) 세분화: X항목이 세분화되면, X항목은 유지되나, 이와 연관된 세부항목(예: X-C1, X-C2)을 생성할 수 있다. 이 때, 맵핑되는 Y항목은 기존 항목이거나 새로 추가되는 항목일 수 있다.
(3) 분할: X항목이 분할되면, X항목과 그에 맵핑된 Y항목이 제거된다. 제거된 X항목은 여러 요구사항(X-C1,

X-C2)으로 분할된다. 이 때, 맵핑되는 Y항목은 기존 항목이거나 새로 추가된 항목일 수 있다.

(4) 통합: 여러 요구사항(X1, X2)이 통합되면, 그에 맵핑되는 Y항목은 다른 항목과의 맵핑으로 유지될 수도 있고 X항목과 함께 제거될 수 있다. 이 때 통합된 X항목이 추가되고 이에 맵핑되는 Y항목은 기존에 있을 수 도 있고 새로 추가되는 항목일 수 있다.

(5) 대체: X항목이 대체되면, 그에 맵핑되는 Y항목은 다른 항목과의 맵핑으로 유지될 수도 있고 X항목과 함께 제거될 수 있다. 이 때 대체된 X1항목이 추가되고 이에 맵핑되는 Y항목은 기존에 있을 수 도 있고 새로 추가되는 항목일 수 있다.

(6) 삭제: X항목이 삭제되면, 그에 맵핑되는 Y항목은 다른 항목과의 맵핑으로 유지될 수도 있고 X항목의 제거에 따라 함께 삭제될 수도 있다.

(7) 비활성화: X항목이 비활성화되면, X항목에 맵핑되는 Y는 존재하지 않는다.

우리는 이러한 두 항목 간의 맵핑 변경을 수시로 기입할 수 있도록 하고, 변경된 항목 간의 맵핑 관계에서 RTM을 자동으로 생성하는 방법을 제안하고자 한다.

3. 제안 요구사항 추적성 매트릭스

우리가 제안하는 스프레드시트 형태의 요구사항 추적성 매트릭스는 개별 시트에 두 개의 산출물 간의 맵핑을 표현하고, 이러한 맵핑 집합을 바탕으로 전체 추적성 매트릭스를 자동으로 생성한다.

Fig. 1은 최종 RTM을 자동으로 생성하기 위해 본 논문에서 제안하는 단계이다. RTM생성은 개발자가 맵핑 관계 정보와 최종 RTM의 형태를 결정짓기 위한 기준을 입력한 상태에서 시작한다. 이후 RTM생성 단계는 다음과 같다. 첫번째, 어떤 산출물을 우선으로 맵핑을 수행할지 기준을 결정한다. 두번째, 기준 산출물에 따라 각 산출물 간의 맵핑 관계를 정렬하여 Table 1과 같은 전형적인 RTM을 생성한다. 세번째, 생성된 RTM에 대한 오류 검사를 통해 추적성 링크가 누락된 셀을 표시하여 최종 RTM의 점검을 도와준다.

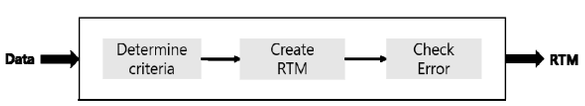
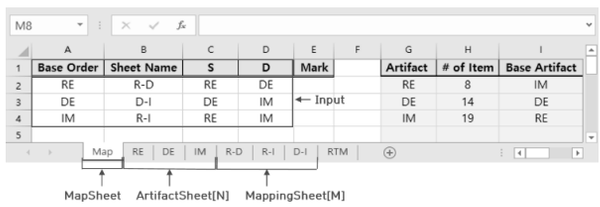


Fig. 1. The Steps to Automatically Create the Final RTM

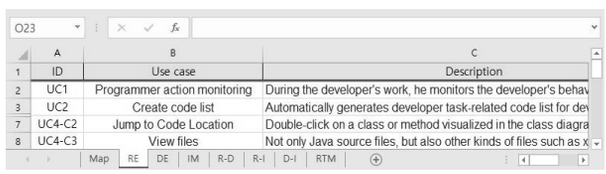
3.1 개발자의 데이터 입력

사용자는 최종 RTM을 생성하기 위해 다음 3가지 정보를 사전에 입력해야 한다.

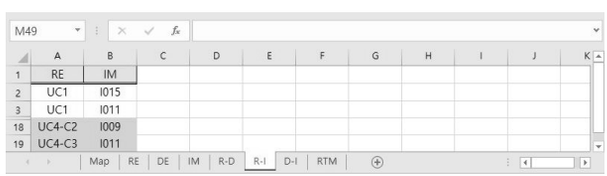
- MapSheet: 전체 산출물 관계 입력
- ArtifactSheet[N]: 각 산출물 항목의 정보 입력
- MappingSheet[M]: 두 개의 산출물 항목 간의 맵핑 정보 입력



(a) Example of MapSheet



(b) Example of ArtifactSheet



(c) Example of MappingSheet

Fig. 2. Data Input

Fig. 2는 입력해야 하는 데이터의 예를 보여준다. 먼저 Fig. 2(a)의 MapSheet에는 최종 RTM에 출력할 산출물 순서를 "기준 순서" 열에 입력한다. "기준 순서" 열은 MapSheet의 첫 번째 열로서 최종 RTM의 기대 출력 순서로 N개의 산출물 나열한다. 다음, 산출물 간의 맵핑 정보를 입력한다. 이를 위하여 두 번째 열인 "시트명"에는 맵핑 관계를 포함하고 있는 MappingSheet[M]의 이름을 입력한다. 다음 세번째와 네번째 열인 "S"열과 "D"열에는 MappingSheet[M]의 추적성 맵핑 대상인 두 개의 산출물의 이름을 기입한다. 예를 들어 요구명세서를 RE, 설계서를 DE라고 하고, 그 맵핑 관계를 포함하는 시트를 R-D라고 한다면, "시트명"에는 R-D를, 그리고 "S"에는 RE를, "D"에는 DE를 입력한다.

Fig. 2(b)는 산출물 항목 정보를 입력한다. Fig 2.b에서의 ArtifactSheet[N]는 각 산출물 이름을 탭명으로 하는 N개의 시트로, 산출물의 항목에 대한 ID, 이름을 기술하는 두 개의 열을 가진다.

Fig. 2(c)는 산출물간 맵핑 정보를 입력한다. MappingSheet[M]은 맵핑 관계 이름을 탭명으로 하는 시트로, 맵핑 관계를 구성하는 두 산출물의 항목ID를 기술하는 열을 가진다. 맵핑 관계에 변경이 발생하면 초록색으로, 의도적으로 항목을 더 이상 맵핑하지 않았을 경우 회색으로 표시한다.

3.2 기준 결정 (Step #1)

Step #1은 최종 RTM을 생성하기 위해 읽어 들이는 산출물의 순서를 지정한다. RTM 생성 시 맵핑 관계를 정렬하는데 가장 적은 계산을 할 수 있도록 산출물 내 항목이 가장 많은 산출물의 순서를 자동 결정한다.

Fig. 2(a)에서 회색 열로 표시된 부분이 기준 결정을 수행

```
   Input:
   - MapSheet: A Sheet contain the all artifact
relationship
   : Col[1] lists N airifact names in the expected output
order of the final RTM
   : Col[2,3,4] lists the name of the MappingSheet[M] and
the names of the two aritifacts contained in that sheet
   - ArtifactSheet[N]: The sheets of number of N that
named by each artifact name
    : Col[1,2] lists the ID and name of the item in the
artifact
   Output:
   - MapSheet(Col[9]): The order of the products to be
calculated first in the final RTM
   Variable:
   - artifactSet: The set of artifact name
   - artifactArray[N]: A class object list having an
artifact name and a number of items included in the
artifact as members
   Algorithm (Decide_Base)
   Begin
      For (i = 2; i < M; i++)
         artifactSet.add(MapSheet(3:i))
      artifactSet.add(MapSheet(4:i))
   End For
   print artifactSet to MapSheet(7:)
   For (i = 2; i < N; i++)
      numOfItems = getNumOfItems(ArtifactSheet
      (MapSheet(7:i)))
      print numOfItems to MapSheet(8:i)
      artifactArray.add(MapSheet(7:i), MapSheet(8:i))
   End For
      Sort_Descending(artifactArray)
      For (i = 0; i < N; i++)
      print artifactArray.name(i) to Map(9:i)
   End For
   End
```

Fig. 3. An Algorithm for Deciding the Order of Sheets to be Read (Step #1)

한 결과이다. 산출물 열은 사용자가 입력한 두 산출물의 맵핑에 존재하는 산출물을 보여준다. 항목 개수는 산출물 열의 산출물에 존재하는 항목 개수를 보여준다. 기준 산출물 열은 산출물 항목의 개수가 많은 산출물에서 적은 순서로 정렬한 결과를 보여준다.

예를 들어 현재 맵핑 관계를 구성하고 있는 산출물은 RE, DE, IM이고 각 산출물의 항목 개수는 각각 8개, 14개, 19개이다. 이를 내림차순 정렬하면 IM이 가장 많은 항목의 개수를 가지고 있으므로 첫번째 기준 산출물이 된다. 이후 IM 다음으로 항목의 개수가 많은 DE, RE 순으로 기준 산출물이 된다.

이러한 기준 결정을 수행하는 알고리즘은 Fig. 3과 같다. 기준 결정의 입력은 Fig. 2에서 빨간색으로 표시한 부분이며, 출력은 MapSheet의 회색 열이다. Fig 3의 알고리즘은 먼저 MapSheet Col [3, 4]에 존재하는 산출물 이름을 읽어서 산출물 이름의 집합을 표현하는 집합 변수 artifactSet에 저장 후, 이를 7번째 열에 있는 “산출물” 열에 기입한다. 이후 산출물

```
   Input:
   - MapSheet:  A Sheet contain the all artifact
relationship
   :  Col[1] lists N airifact names in the expected output
order of the final RTM
   :  Col[2,3,4] lists the name of the MappingSheet[M] and
the names of the two aritifacts contained in that sheet
   - ArtifactSheet[N]: The Sheets of number of N that
named by each artifact name
    : Col[1,2] lists the ID and name of the item in the
artifact
   - MappingSheet[M]: The sheets of number of M that named
by mapping-relationship name
   : Col [1,2] lists the item IDs of the two aritifacts
contained in the mapping relationship
   Output:
   - RTMSheet:  A sheet that create the final RTM by
arranging each mapping relationship
   Variable :
   - baseNum: Specify the line number of Col [1] in
MapSheet
   - artifactArray[N]: A class object list having an
artifact name and a number of items included in the
artifact as members
   Algorithm (Create_RTM)
   Begin
      int baseNum = 2
      # Create an RTM title
      For (i = 1; i < (N - 1) * 2; i++)
         If (i Mod 2 == 0)
            print “Description” to RTMSheet(i:1)
         Else
            print  MapSheet(1:baseNum)  to  RTMSheet(i:1)

         baseNum = baseNum + 1
       End If
   End For
   # Create RTM
      For (i = 1; i < N; i ++)
      Copy_ID(artifactArray(i).name, i)
   End For
   Copy_Description()
   End
```

Fig. 4. An Algorithm for Creating the Final RTM (Step #2)

의 항목 개수를 얻기 위해 7번째 열의 산출물 이름을 탭명으로 하는 시트에서 항목의 개수를 읽어들여, 이를 8번째 열에 기입한다. 마지막으로 7, 8번째 열을 산출물 이름과 개수를 가지는 객체의 배열 artifactArray에 저장한 후, 항목 개수에 따라 내림차순 정렬하여 그 결과를 순서대로 9번째 열에 있는“기준 산출물” 열에 기입하여 기준 산출물을 결정한다.

기준 결정을 완료 후 산출물 열과 기준 순서 열을 비교하여 오류를 검사한다. 만약 산출물 목록에 없는 산출물 이름이 기준 순서에 존재하면 에러 메시지를 발생시킨다.

3.3 RTM생성 (Step #2)

기준 산출물에 따라 MappingSheet[M]의 맵핑 관계를

RTMSheet로 복사하여 전형적인 형태의 RTM을 생성한다. RTM생성 알고리즘은 Fig. 4와 같으며 레이블 생성(Create_RTM 함수), 맵핑 관계 복사(Copy_ID 함수), 산출물의 항목 정보 복사(Copy_Description 함수)로 구성된다. 즉 Create_RTM 함수에서 레이블을 생성 후 Copy_ID 함수를 활용하여 맵핑 관계를 복사하고 Copy_Description 함수를 활용하여 산출물의 항목 정보를 복사하여 RTM을 완성한다.

Create_RTM은 RTMSheet의 첫번째 행에 레이블을 생성한다. 각 레이블의 항목은 다음과 같다. 첫번째, 홀수 번째 레이블에는 MapSheet의 기준 순서 열의 산출물 이름을 순서대로 출력한다. 두번째, 짝수 번째 레이블에는 각 산출물 항목에 대한 설명을 입력하는 'Description'을 출력한다.

하위절 3.3.1절에서는 맵핑 관계를 복사하는 함수 Copy_ID를 그리고 3.3.2절에서는 산출물 항목에 대한 정보를 복사하는 함수 Copy_Description를 설명한다.

1) RTM생성: Copy_ID 함수

RTMSheet의 레이블이 생성되면 MappingSheet[M]에 포함된 맵핑 관계를 RTMSheet로 복사한다. 이 때 색상으로 표시된 맵핑 관계의 변경 여부를 함께 복사한다. 복사는 첫번째 기준 산출물을 포함하는 MappingSheet[M]부터 수행한다. 첫번째 기준 산출물을 포함하는 MappingSheet[M]의 복사가 끝나면 두번째 기준 산출물을 포함하는 MappingSheet[M]를 복사한다. 이 때 이전에 복사된 맵핑 관계를 기준으로 추적성 링크를 유지하며 삽입한다. 또한 새로 추가되거나 누락된 맵핑 관계가 있으면 RTM의 마지막 행에 추가한다. 이 때의 스텝을 간단하게 정리하면 다음과 같다.

a) 기준 산출을 포함하고 복사하지 않은 맵핑 관계인지 검사한다.
b) 첫 번째 복사인지 검사한다.
c) 첫 번째 복사인 경우 MappingSheet의 맵핑 관계를 RTMSheet로 단순 복사한다.
d) 첫 번째 이후의 복사인 경우 이전에 복사된 맵핑 관계를 기준으로 정렬하여 복사한다.
e) 모든 MappingSheet의 복사가 수행되면 종료한다.

2) RTM생성: Copy_Description 함수

RTMSheet로 모든 맵핑 관계의 복사가 완료되면 각 산출물 항목에 해당하는 정보를 출력한다. RTMSheet의 산출물 ID와 ArtifactSheet[N]의 산출물 ID를 비교하여 해당하는 ID에 대한 설명을 복사한다. 이 때의 스텝을 간단하게 정리하면 다음과 같다.

(1) ArtifactSheet의 산출물 항목 ID와 RTMSheet의 산출물 항목 ID를 비교한다.
(2) 동일한 ID인 경우 ArtifactSheet에서 해당 ID의 Description을 RTMSheet로 복사한다.

3.4 오류 검사 (Step #3)

오류 검사는 생성된 RTM에서 맵핑 관계가 누락된 링크를 찾아 빨간색으로 표시한다. 오류 검사 알고리즘은 Fig. 5와 같다. 해당 알고리즘은 RTMSheet를 순회하여 빈 셀이면서 구현 항목이 'na'이 아닌 셀을 맵핑 관계가 누락된 추적성 링크로 식별한다.

```
Input:
- RTMSheet: A sheet that create the final RTM by arranging each mapping relationship
Output:
- Marking mapping relationships with missing traceability links in RTMSheet in red
Variable
- RTMLength: The maximum length of a row in RTMSheet
- imColNum: Save the column number in the RTMSheet where Col [:1] is IM
Algorithm (Check_Error)
Begin
   RTMLength = 0
For (i = 1; i < (N - 1) * 2; i++)
   If (RTMLength < numOfItems(RTMSheet(i:))
      RTMLength = numOfItems(RTMSheet(i:))
      End If
   End For
   For (j = 1; j < (N - 1) * 2; j++)
      If (RTMSheet(j:1) == "IM")
      colNum = j
   End If
   For (i = 2; i < RTMLength; i++)
     If (RTMSheet(i:j) == "")&&(RTMSheet(colNum:i) != "na")
            color RTMSheet(i:j) in red
     End If
   End For
End For
End
```

Fig. 5. An Algorithm for Checking Errors (Step #3)

또한 생성된 RTM에서 내부적 변경이 발생한 셀은 초록색으로 표시하고 중간에 비활성화 된 셀은 회색으로 표시한다.

3.5 척도

최종 요구사항 매트릭스의 결과를 바탕으로 구현율과 추적률을 자동으로 계산한다. 구현율은 전체 구현 항목 중에서 실제 구현이 된 항목의 비율이다. 구현율(I_r)은 전체 구현 항목 개수(I_{ac})에서 미수용 된 구현 항목 개수(NI_c)를 뺀 값을 전체 구현 항목 개수로 나누어 백분율로 표현한 것이다. 구현율을 수식으로 나타내면 다음과 같다.

$$I_r = \frac{I_{ac} - NI_c}{I_{ac}} \times 100 \quad (1)$$

추적률은 요구사항 매트릭스의 전체 링크 중에서 누락된 링크를 제외한 링크의 비율이다. 추적률(T_r)은 전체 추적성 링크 개수(R_{ac})에서 누락된 링크 개수(M_c)를 뺀 값을 전체 추적성 링크 개수로 나누어 백분율로 표현한 것이다. 추적률을 수식으로 나타내면 다음과 같다.

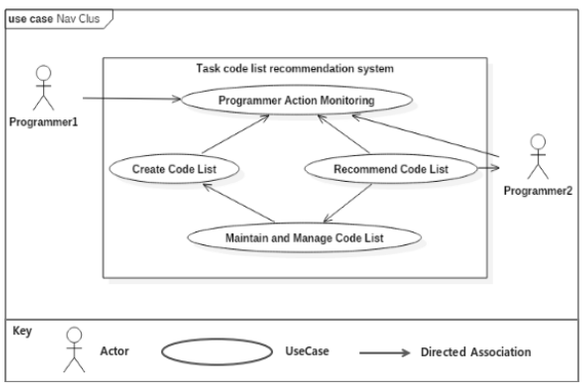


Fig. 6. A Use Case Diagram of a Software Project for Our Case Study

$$T_r = \frac{R_{ac} - M_c}{R_{ac}} \times 100 \quad (2)$$

계산된 구현율과 추적률은 RTM생성 시트에 최종 요구사항 추적성 매트릭스와 함께 표현된다.

최종으로 출력된 RTM은 기존의 맵핑 정보를 바탕으로 최대한 추적성을 완성하여 보여준다. 개발자는 이를 검토, 수정하여 RTM을 완성할 수 있다.

4. 사례 연구

우리는 제안한 RTM이 기존의 전형적인 RTM보다 변경사항에 대해서 유연하게 대처할 수 있는지를 확인하기 위하여 사례 연구를 진행하였다. 사례 연구는 본 저자 중 한명이 과거에 개발한 오픈 소스를 대상으로 진행하였다[1]. 분석 대상이 된 네브마인 도구는 개발자들의 코드 탐색을 돕는 도구로, Fig. 6과 같이 네 가지 유스케이스를 갖는다. 각 유스케이스의 간략한 설명은 다음과 같다.

- UC1: 개발자의 행위를 모니터링하여 상호 이벤트를 자동으로 수집한다.
- UC2: 수집한 이벤트로부터 개발자들의 태스크와 관련된 코드 목록을 생성한다.
- UC3: 생성된 태스크 관련 목록을 유지 관리한다.
- UC4: 현재 개발자가 수행하는 작업과 관련된 태스크 코드 목록을 추천한다.

시스템에서 필요로 하는 컴포넌트는 열 세가지로 식별된다. 각 컴포넌트의 설명은 다음과 같다.

- C01: 이클립스 개발 환경에서 개발자의 인터랙션을 모니터링하고 인터랙션 이벤트 기록을 생성한다.
- C02: 인터랙션 이벤트 기록을 네비게이션 시퀀스 단위까지 축적하고 있다가 하나의 네비게이션 시퀀스를 형성하면, 이를 그 다음 필터 컴포넌트로 전송한다.
- C03: 전송받은 네비게이션 시퀀스를 일정 단위로 축적하고 있다가 정기적으로(약 3분 마다) 서버에 있는 Collection Manager 로 데이터를 전송한다.
- C04: 서버에 있는 모듈로 전송받은 데이타를 DB에 저장한다.
- C05: 요청이 들어오면 해당 요청에 맞는 코드 목록을 전송한다.
- C06: 컬렉션이 일정한 문맥으로 묶이도록 정기적으로 관리한다.
- C07: 코드 목록을 저장, 유지하는 데이터 베이스이다.
- C08: 개발자의 인터랙션 이벤트 기록을 바탕으로 해당 개발자의 일정 네비게이션 컨텍스트를 형성하는 컴포넌트로, 해당 컨텍스트를 형성하면 다음 필터 컴포넌트로 전송한다.
- C09: 개발자의 네비게이션 컨텍스트를 바탕으로 DB에 가장 유사한 컨텍스트의 코드 목록을 추천해 달라고 요청하며, DB로부터 가장 유사한 컨텍스트의 코드 목록을 받게 되면 해당 데이터를 Filter에게 전송한다.
- C10: Collection Selector 가 요청한 문맥에 따라 받았던 최근의 코드 목록을 임시로 저장하여 캐쉬 역할과 같이 유사한 문맥에 대하여 빠른 시간 내 응답할 수 있도록 한다.
- C11: 추천 목록이 있을 때 개발자의 작업에 방해되지 않도록 추천 목록이 있음을 알려주기 위해서 추천 내역을 보여주기 전에, 조용히 추천 목록 있음을 표시한다.
- C12: 추천 목록 중 중요하지 않은 목록을 걸러내고 중요한 코드만을 화면에 보이도록 한다.
- C13: 개발자의 이클립스 개발 환경에서 개발자에게 추천 코드 목록을 가시화하여 보여준다. 개발자 활동과 유사한 컨텍스트의 코드 목록을 보여 주어 개발자의 코드 네비게이션을 돕는다.

우리는 여기서 UC4에 초점을 맞추어 설명을 진행한다. 5.1절에서는 초기 RTM 생성에 대해서 설명한다. 5.2절과 5.3절에서는 각각의 다른 개발 단계에서 발생한 변경 사항에 대한 RTM에서의 처리를 설명한다. 5.4절에서는 이 사례 연구에 대한 결과를 논의한다.

4.1 초기 RTM 생성

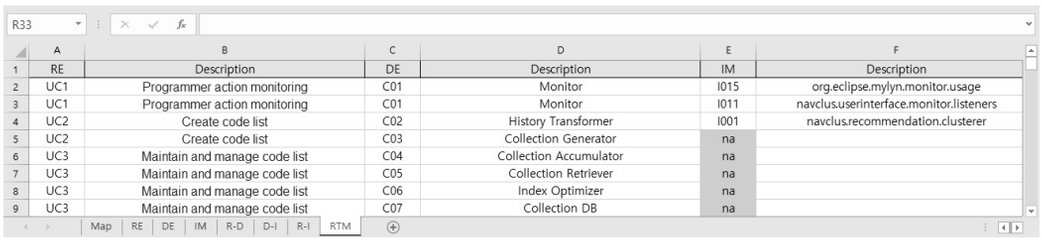
초기 RTM 생성을 위하여 산출물 정보와 산출물 간의 맵핑 관계를 입력하여 추적성 매트릭스를 생성한다. 하위 1)절부터 3)절은 추적성 매트릭스를 생성하는 작업을 설명한다.

1) 산출물 정보 입력

우리는 각 산출물 시트에 산출물 정보를 입력하였다. 먼저 RE 시트에는 유스케이스의 ID와 설명을 기입하였다. 다음 DE 시트에는 설계 모듈 ID와 컴포넌트 설명을 입력하였다. 그 중 UC4와 연관되는 컴포넌트 목록을 설명하면 다음과 같다.

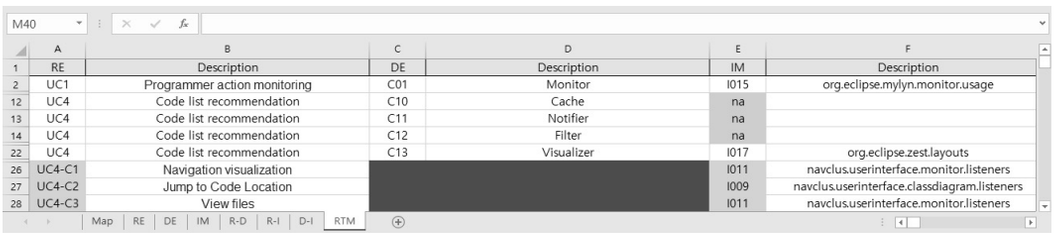
- C08은 개발자 인터랙션 이벤트 기록을 바탕으로 해당 개발자의 네비게이션 컨텍스트를 형성한다.
- C09는 네비게이션 컨텍스트를 바탕으로 가장 유사한 컨텍스트의 코드 목록 추천을 요청한다.
- C11은 추천 목록이 있을 때 개발자에게 추천 목록이 있음을 표시한다.
- C13은 개발자의 이클립스 개발 환경에서 개발자에게 추

1) https://marketplace.eclipse.org/content/navmine



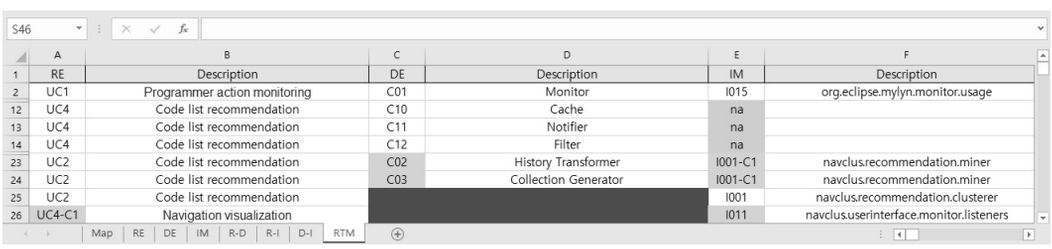
	A	B	C	D	E	F
1	RE	Description	DE	Description	IM	Description
2	UC1	Programmer action monitoring	C01	Monitor	I015	org.eclipse.mylyn.monitor.usage
3	UC1	Programmer action monitoring	C01	Monitor	I011	navclus.userinterface.monitor.listeners
4	UC2	Create code list	C02	History Transformer	I001	navclus.recommendation.clusterer
5	UC2	Create code list	C03	Collection Generator	na	
6	UC3	Maintain and manage code list	C04	Collection Accumulator	na	
7	UC3	Maintain and manage code list	C05	Collection Retriever	na	
8	UC3	Maintain and manage code list	C06	Index Optimizer	na	
9	UC3	Maintain and manage code list	C07	Collection DB	na	

(a) Initial Traceability Matrix



	A	B	C	D	E	F
1	RE	Description	DE	Description	IM	Description
2	UC1	Programmer action monitoring	C01	Monitor	I015	org.eclipse.mylyn.monitor.usage
12	UC4	Code list recommendation	C10	Cache	na	
13	UC4	Code list recommendation	C11	Notifier	na	
14	UC4	Code list recommendation	C12	Filter	na	
22	UC4	Code list recommendation	C13	Visualizer	I017	org.eclipse.zest.layouts
26	UC4-C1	Navigation visualization			I011	navclus.userinterface.monitor.listeners
27	UC4-C2	Jump to Code Location			I009	navclus.userinterface.classdiagram.listeners
28	UC4-C3	View files			I011	navclus.userinterface.monitor.listeners

(b) Traceability matrix that reflects changes in requirements



	A	B	C	D	E	F
1	RE	Description	DE	Description	IM	Description
2	UC1	Programmer action monitoring	C01	Monitor	I015	org.eclipse.mylyn.monitor.usage
12	UC4	Code list recommendation	C10	Cache	na	
13	UC4	Code list recommendation	C11	Notifier	na	
14	UC4	Code list recommendation	C12	Filter	na	
23	UC2	Code list recommendation	C02	History Transformer	I001-C1	navclus.recommendation.miner
24	UC2	Code list recommendation	C03	Collection Generator	I001-C1	navclus.recommendation.miner
25	UC2	Code list recommendation			I001	navclus.recommendation.clusterer
26	UC4-C1	Navigation visualization			I011	navclus.userinterface.monitor.listeners

(c) Traceability matrix that reflects changes in design module

Fig. 7. Examples of Requirement Traceability Marix in Our Case Study

천 코드 목록을 시각화 한다.

- 또한 IM시트에 패키지에 대한 ID와 설명을 입력하였다.
- I005는 n.u.m.actions[2] 패키지로 개발자의 인터랙션을 모니터링하는 클래스들을 포함한다.
- I012는 n.u.m.patterns 패키지로 개발자의 인터랙션에 대한 패턴을 도출하는 클래스를 포함한다.
- I013은 n.u.m.selections 패키지로 현 개발자의 패턴을 바탕으로 기존 코드 목록을 선택한다.
- I004은 n.u.classdiagram으로 선택한 기존 코드 목록을 클래스 다이어그램에 보여준다.

참고로, 컴포넌트 C8, C9, C13은 구현이 되었지만 설계 평가에서 우선순위가 낮은 C11은 구현되지 않았다.

2) 산출물 간의 맵핑 관계 입력

다음으로 요구사항과 설계, 설계와 구현 사이의 맵핑 관계를 각각의 시트에 입력하였다.

- R-D시트에 RE 시트의 항목과 DE 시트의 항목 간의 맵핑 관계를 입력하였다. 예를 들어 UC4는 C08, C09, C10, C11, C12, C13과 맵핑된다.
- D-I시트에 DE 시트의 항목과 IM 시트의 항목 사이의 맵핑 관계 및 상태를 입력하였다. 예를 들어 C08은 I005와 맵핑된다. C9는 패키지 I012와 I013과 맵핑된다. C13은 패키지 I004, I006, I007, I008, I009, I010, I016, I017에 맵핑된다. 그러나 C11은 구현되지 않았으므로 회색으로 표기한다.

3) 추적성 매트릭스 생성

해당 사례연구에서 각 산출물의 항목 개수는 요구사항 항목, 설계항목, 구현항목에 대해 각각 9개, 13개, 20개이다. 따라서 산출물 내 항목의 개수가 가장 많은 구현항목이 첫번째 기준 산출물이 되고, 이후 항목의 개수가 많은 설계항목, 요구사항 항목이 순서대로 다음 기준 산출물이 된다. 기준 산출물이 결정되면 버튼을 클릭하여 각 시트의 맵핑 관계를 반영한 추적성 매트릭스를 생성한다. 생성된 초기 RTM은 Fig. 7의 (a)와 같다.

생성된 최종 RTM에서는 변경 사항과 누락된 추적성 링크가 존재하지 않는다. 그러나 구현항목 중 구현하지 않은 항목이 존재하기 때문에 해당 링크는 회색으로 표시된다. 따라서 RTM 초기 버전의 척도를 계산해보면 구현율은 65%이고, 추적률은 100%로 나타난다.

4.2 요구 사항의 변경 대처

구현하는 단계에서 UC4는 다음과 같이 세분화 되었다.

- UC4-C1은 사용자가 네비게이션한 활동에 따라 방문한 코드를 클래스 다이어그램에 가시화한다.

2) 패키지 명이 긴 관계로, 줄여서 알파벳으로 표기하였다.

- UC4-C2는 사용자가 다이어그램에서 클래스를 더블클릭 할 때 그 코드위치로 점핑하여 소스 파일을 에디터에 보여준다.
- UC4-C3은 java 파일뿐만 아니라 xml, html 파일 등 다른 종류의 파일도 클래스 다이어그램에 보여준다.

이와 관련한 변경을 반영한 추적성 매트릭스를 생성하기 위한 작업을 하위절 4.2.1절부터 4.2.3절에서 설명한다.

1) 변경 관련 산출물 정보 입력

세분화된 요구사항을 반영하기 위해 기존 패키지에 새로운 클래스가 추가되는 등 변경이 있었다.

- I009는 n.u.classdiagram.listeners 패키지로서 클래스 다이어그램에서의 사용자의 더블클릭 액션을 모니터링하는 클래스를 추가하였다.
- I011은 n.u.m.listeners 패키지로서 다른 종류의 파일을 여는 사용자의 인터랙션을 모니터링하는 클래스를 추가하였다. 또한 모니터링한 정보를 바로 가시화하는 기능도 추가하였다.

2) 변경 관련 산출물 간의 맵핑 관계 입력

이런 세분화된 관계에 따라 산출물 간의 맵핑 관계를 변경해야 했다. 우리는 현재 구현에 집중하고 있고, 이에 따라 요구사항이 세분화되었기 때문에 R-I시트에서 요구사항과 구현 산출물 간의 맵핑 관계를 변경하였다.

- UC4-C1은 I011에 맵핑된다. UC4-C1을 구현하기 위해 I011에서 사용자의 인터랙션을 즉시 화면에 모니터링하도록 구현하였다.
- UC4-C2는 I009에 맵핑된다. UC4-C2를 구현하기 위해 I009에서 개발자의 더블 클릭 액션을 추가로 인식하도록 구현하였다.
- UC4-C3은 I011에 맵핑된다. UC4-C3를 구현하기 위해 I011에서 다른 종류의 파일을 여는 사용자 동작을 모니터링하도록 구현하였다.

3) 변경 맵핑 관계 관련 추적성 매트릭스 생성

요구사항의 변경을 반영하여 생성된 RTM은 Fig. 7의 (b)와 같다. 변경된 RTM은 초기 RTM의 마지막 행에 새로운 맵핑 관계인 UC4-C1과 I001, UC4-C2와 I009, UC4-C3과 I011이 추가된 모습이다. 우리는 맵핑 관계의 추가를 변경으로 간주하기 때문에 추가된 맵핑 관계를 초록색으로 표시한다. 그러나 추가된 맵핑 관계와 설계 항목 간에는 맵핑이 존재하지 않기 때문에 추가된 행의 설계항목은 붉은색으로 표시된다. 이에 따라 변경이 반영된 RTM의 구현율은 61%이고 추적률은 88%로 나타난다.

4.3 설계 모듈의 변경 대처

다음으로는 소프트웨어 진화 과정에서 설계 모듈의 변경이 발생하였다. 추천을 위해서 원래 클러스터링 기법을 사용하였으나, 정확도와 성능을 높이기 위해 Association Rule Mining 기법과 과 Inverted Index 기법을 사용하도록 설계를 변경하였다. 이 때, 요구사항의 변경은 발생하지 않았으며 변경은 설계 컴포넌트와 구현 파일에서 발생하였다.

이러한 컴포넌트의 변경 사항은 설계 산출물 시트에 일단 반영하였다.

- C02는 기존의 인터랙션 이벤트 기록을 일정 기준 하에 하나의 네비게이션 시퀀스를 형성하였다. 해당 기준을 사용자가 10분 이상 인터랙션을 하지 않는 기준으로 수정하였다.
- C03는 전송받은 네비게이션을 바탕으로 클러스터링을 수행한다. 이를 Association Rule Mining 기법으로 수정하였다.

이와 관련한 변경을 반영한 추적성 매트릭스를 생성하기 위한 작업을 하위절 4.3.1절부터 4.3.3절에서 설명한다.

1) 변경 관련 산출물 정보 입력

이에 따라 변경한 구현 패키지는 다음과 같다.

- I001은 n.recommendation.clusterer 패키지로서 클러스터링 기법을 구현하였다. 그러나 이번 변경에서 클러스터링 기법을 제외함으로써 I001은 다른 패키지로 대체되었다.
- I001-C1은 n.recommendation.miner 패키지로서 새로 도입한 기법인 Association Rule Mining 기법을 구현한다. 해당 패키지는 I001을 대체한다.

2) 변경 관련 산출물 간의 맵핑 관계 입력

설계 항목에 대한 구현 항목의 변경을 RTM에 반영하기 위해 D-I시트에서 설계 항목과 구현 항목 간의 맵핑 관계를 입력하였다.

- C02는 I001-C1과 맵핑된다. C02를 구현하기 위해 I001-C1에서 I001의 클러스터링 기법을 마이닝기법으로 대체하여 구현하였다.
- C03 또한 I001-C1과 맵핑된다. C03를 구현하기 위해 I001-C1에서 I001의 클러스터링 기법을 마이닝기법으로 대체하여 구현하였다.

3) 변경 맵핑 관계 관련 추적성 매트릭스 생성

설계 모듈의 변경을 반영하여 생성된 RTM은 Fig. 7의 (c)와 같다. 해당 변경이 반영되어 생성된 RTM은 4.2절의 RTM에서 C02와 I001, C03과 na의 맵핑 관계가 제거되고 변경된 맵핑 관계인 C02와 I001-C1, C03과 I001-C1이 추가된다. 이 때 변경된 맵핑 관계는 초록색으로 표시가 된다. 그러나 C02와 I001의 맵핑이 제거되면서 UC02와 C02 그리고 I001의 맵핑 관계가 누락되어 해당 행의 설계 항목은 붉은색으로 표시된다. 이에 따라 변경이 반영된 RTM의 구현율은 74%이고 추적률은 85%로 나타난다.

4.4 사례 연구의 논의

4.1절에서 네브마인을 대상으로 초기 RTM을 작성하였다. 4.2절에서는 요구 사항의 변경이 발생하여 해당 변경과 관련된 탭의 내용을 수정하여 RTM을 생성하였다. 그 결과 추가된 맵핑 관계가 초기 RTM의 마지막 행에 추가되었음을 알 수 있었다. 또한 변경에 의해 누락된 맵핑 관계에 대해서는

오류 표시를 해주었다. 이 후 4.3절에서 설계 모듈의 변경이 발생하여 해당 변경과 관련된 탭의 내용을 수정하여 RTM을 생성하였다. 그 결과 4.2절의 최종 RTM에서 변경된 기존의 맵핑 관계는 제거되고 수정된 맵핑 관계가 추가되었다. 우리의 RTM은 변경이 발생하였을 때 사용자가 변경과 관련된 탭의 내용만 수정하면 기존의 맵핑 관계를 유지하여 최종 RTM을 자동 생성해준다.

4절의 사례연구를 통해 본 논문에서 제안하는 RTM이 기존의 RTM보다 변경에 유연하게 대처할 수 있음을 확인하였다. 그러나 기준 산출물과 맵핑된 산출물의 맵핑 관계를 우선으로 RTM을 생성하기 때문에 기준 산출물이 변경되면 다수의 산출물 간의 맵핑 관계가 유지되는 않는 문제가 있다. 또한 여러 산출물의 다대다 맵핑 관계에 있어서. 최종 RTM에서 일부 매핑을 추가 맵핑으로 넣어주는 문제도 있다. 따라서 향후 복잡한 시스템에서의 요구사항 추적성 맵핑을 위하여 추가적인 알고리즘 개선이 필요하다.

5. 관련 연구

관련 연구는 요구사항 추적성 관련 도구 제안, 스프레드시트 형태의 활용, 요구사항 추적성에서의 변경 관리로 나누어서 논의한다.

5.1 요구사항 추적성 관리 도구

최근의 도구 연구들은 산출물 항목 간의 관계를 자동 도출하기 위해 데이터 마이닝 기법을 사용했다[5, 6]. 예를 들어 TraceM 도구는 요구사항 산출물 간의 암묵적인 관계를 탐색하여 오픈 하이퍼미디어를 통해 명시적인 관계를 만든다[5]. Trustrace는 마이닝과 정보검색 기술을 접목하여 소프트웨어 개정 이력에서 요구사항 추적성을 복구한다[6]. 그러나 이러한 방법들은 아직 낮은 정확도를 보이고 있다.

또한 플러그인을 사용하여 추적성을 관리하는 방법을 제시하였다. 예를 들어 Software Artefacts Traceability Analyzer [7]는 Jenkins 플러그인으로 개발 프로세스 동안 지속적인 통합에 따른 추적성 관리를 제공하고자 했으며, 추적성 링크 시각화를 제공하고자 하였다. Capra는 이클립스 플러그인으로 유연한 추적성 관리 도구를 개발하기 위해 프로젝트 및 조직에 의해 맞춤화 될 수 있도록 도구를 구성하고 확장 지점을 제공한다 [8]. 그러나, 이러한 방법들은 플러그인 사용을 위한 환경설정과 학습이 필요하다.

5.2 스프레드시트 형태의 요구사항 활용

Hussain과 동료들은 협업 개발 프로젝트에서의 요구사항을 관리하기 위한 스프레드시트의 활용을 조사하였다 [9]. 그 결과 파키스탄과 미국, 뉴질랜드와 미국, 캐나다의 협업의 두 가지 프로젝트를 조사하여 3가지 형태의 스프레드시트 활용 형태를 밝혀내었다. 그러나 주어진 현상을 연구했을 뿐, 스프레드시트를 활용하기 위한 방법을 제시하지는 않았다.

Marshall과 동료들은 사용자가 제공한 문서로부터 요구사항을 추출하여 스프레드시트에 저장한 후 MBSE로 요구사항을 모델링하였다[10]. 또한 Wohlrab과 동료들은 태국의 중소 소프트웨어 회사들 사이의 요구 공학 문제의 현재 상태와 실무를 조사하는 과정에서 질문 항목에 대한 답변을 정리할 때 스프레드시트를 사용하였다[11]. 그러나 [10, 11]에서는 스프레드시트를 요구사항의 추적성을 위한 방법으로는 사용되지 않았다.

5.3 요구사항 추적성에 변경 사항 반영

김과 동료들은 기존 추적성에서 변경관리 방법을 제시하고 있지 않다고 지적하고 변경 요구사항에 대해서 새로운 ID를 부여하자고 제안하였다[12]. ID에 변경의 원래의 요구사항을 알 수 있도록 기재함으로써 변경 정도를 기존 연구에 비해 용이하게 추정할 수 있다고 주장하였다. 그러나, 그들의 제시 방법은 기존 추적성 방법에 요구 사항 ID를 변경에 따라 어떻게 추가할 것인가의 개선이었다.

Jayaraman와 동료들은 진화하는 요구사항에 따른 추적성의 변경을 반영하기 위해 추적 링크의 시각화를 Vtrace 도구를 통해 제공하였다[13]. Vtrace는 요구사항을 선택하면 해당 요구사항과 관련된 테스트케이스와 설계 ID를 노드로 표현하여 시각화 한다. 그러나, Vtrace는 진화하는 추적 링크의 가시화에 중점을 두고있다.

5.4 관련 연구 논의

5.1절에서 요구사항 추적성을 관리하는 도구에 대해 논의하였다. [5-8]은 추적성 관리를 위해 새로운 도구를 설치하거나 플러그인을 추가해야 하며 도구를 사용하기 위한 학습이 필요하다. 그러나 우리의 도구는 현재 산업계에서 추적성 관리를 위해 널리 사용되는 스프레드시트에 자동화 방법을 넣었다.

5.2절에서는 스프레드시트를 사용하여 요구사항을 활용하는 방법에 대해 논의하였다. [9]은 지정된 프로젝트의 협업 과정에서 요구사항을 관리하기 위한 스프레드시트의 형태를 제시하였고 [10, 11]에서는 요구사항을 도출하거나 설문 조사 과정에서 스프레드시트를 사용하였다. 그러나 우리의 논문에서는 추적성을 관리하기 위한 스프레드시트 형태의 어플리케이션을 개발하였다.

5.3절에서는 추적성에 변경 사항을 반영하는 방법에 대해 논의하였다. [12, 13]은 요구사항의 변경을 관리하기 위해 각각 변경 ID부여 및 시각화에 초점을 두고 있다. 그러나 우리의 논문은 개발자가 입력한 추적 변경 정보를 반영하여 자동으로 RTM을 생성하는 어플리케이션을 개발하였다.

5.1절은 도구를 설치하여 추적성을 관리하는 방법을 제시하고, 5.2절은 스프레트시트의 사용 사례를 제시한다. 5.3절은 요구사항 변경을 시각화하는 방법을 제시한다. 그러나 본 논문에서는 요구사항 추적성을 관리하기 위해 새로운 도구를 설치할 필요 없이 사용가능한 스프레드시트 형태의 RTM 애플리케이션을 개발하였다. 본 애플리케이션에서 개발자가 변화하는 추적 관계를 쉽게 입력할 수 있다. 애플리케이션은 개발자가 입력한 변경이 발생한 맵핑 관계를 포함한 모든 추적성 링크를 계산하여 전형적인 추적성 매트릭스를 자동으로 생성한다. 개발자는 자동 생성된 RTM을 통하여 추적성의 오류를 쉽게 검토할 수 있다.

6. 결 론

우리는 관리자가 변경된 맵핑을 바로 반영할 수 있고, 필요할 때 전체 요구사항 추적성 매트릭스 형태로 확인할 수 있는 새로운 스프레드 시트의 요구사항 추적성 매트릭스를 제안한다. 제안한 요구사항 추적성 매트릭스는 두 개의 산출물 간의 맵핑만을 포함하는 여러 개의 시트를 포함하고 있으며, 버튼을 클릭하여 전형적인 형태의 요구사항 추적성 매트릭스를 자동 생성할 수 있다. 또한 변경이 발생한 셀은 녹색으로 링크가 누락된 셀은 붉은색으로 표현함으로써 개발자의 검증을 용이하게 하였다.

본 논문에서 제안하는 요구사항 추적성 매트릭스를 평가하기 위해 사례연구를 수행하여 우리의 RTM이 실제로 변경에 유연하게 대처할 수 있다는 것을 확인하였다. 사례연구의 결과 우리는 이러한 요구사항 추적성 매트릭스가 지속적인 맵핑 변경을 관리할 수 있을 것으로 기대한다.

우리는 스프레드시트에 매크로를 적용하여 자동으로 최종 RTM을 생성할 수 있도록 도구를 개발하였다. 그러나 하나의 엑셀파일에 다수의 매크로를 적용하는 과정에서 도구의 로딩 속도가 저하되는 문제점이 발생하였다. 이를 해결하기 위해 향후 현재 알고리즘을 개선함으로써 도구를 최적화 시킬 예정이다. 이후 도구를 공개하고 사용자 피드백을 받아 실용화 하도록 하겠다.

References

[1] J. E. Garcia and A. C. Paiva, “A Requirements-to-Implementation Mapping Tool for Requirements Traceability,” *J. of Software*, Vol.11, No.2, pp.193-200. 2016.

[2] Jongyeol Park, Seunghui Ryu, Serin Jung, Seonah Lee, “Investigation on the difficulties of managing traceability in the evolution of software requirements,” *19th Korea Conference on Software Engineering*, KCSE, 2017.

[3] J. Cleland-Huang, C. K. Chang, and M. Christensen, “Event-based traceability for managing evolutionary change,” *IEEE Trans. on Software Eng.*, Vol.29, No.9, pp.796-810, 2003.

[4] S. Saito, Y. Iimura, H. Tashiro, A. K. Massey, and A. I. Antón, “Visualizing the effects of requirements evolution,” *Proc. 38th Int. Conf. on Software Eng. Companion (ICSE '16)*, ACM, pp.152-161, 2016.

[5] S. A. Sherba, K. M. Anderson, and M. Faisal, “A framework for mapping traceability relationships,” *Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering*. 2003.

[6] N. Ali, Y. G. Guéhéneuc, and G. Antoniol, “Trustrace: Mining software repositories to improve the accuracy of requirement traceability links,” *IEEE Transactions on Software Engineering*, Vol.39, No.5, pp.725-741, 2013.

[7] S. Palihawadana, et al., “Tool support for traceability management of software artefacts with DevOps practices,” *Engineering Research Conference (MERCon), 2017 Moratuwa*. IEEE, 2017.

[8] S. Maro and J. P. Steghöfer, “Capra: A Configurable and Extendable Traceability Management Tool,” *Requirements Engineering Conference (RE), 2016 IEEE 24th International*. IEEE, 2016.

[9] W. Hussain and T. Clear, “Spreadsheets as collaborative technologies in global requirements change management,” Global Software Engineering (ICGSE), 2014.

[10] J. Marshall et al., “Transitioning model based systems engineering to onboard spacecraft electronics,” *Aerospace Conference, 2017 IEEE*. IEEE, 2017.

[11] R. Wohlrab et al., “Collaborative traceability management: Challenges and opportunities,” *Requirements Engineering Conference (RE), 2016 IEEE 24th International*. IEEE, 2016.

[12] Juyoung Kim, Sungyul Rhew, Mansu Hwang, “A Study of Requirement Change Management and Traceability Effect Using Traceability Table,” *The KIPS Transactions: Part D*, Vol.17, No.4, pp.271-282. 2010.

[13] S. C. Jayaraman and M. Anand, “The Impact of Visualizing Traceability Links for Evolving Requirements in Software Maintenance - A Controlled Experiment.”

정 세 린

https://orcid.org/0000-0002-2231-3759

e-mail : rin0608@gnu.ac.kr

2017년 경상대학교 항공우주 및 소프트웨어공학부(학사)

2017년~현 재 경상대학교 정보과학과 석사과정

관심분야 : Requirement Engineering & Software Evolution & Requirement Traceability

이 선 아

https://orcid.org/0000-0002-2004-2924

e-mail : saleese@gnu.ac.kr

1997년 이화여자대학교 전산학(학사)

1999년 이화여자대학교 전산학(석사)

2005년 카네기멜론대학교 소프트웨어공학(석사)

2013년 KAIST 전산학(박사)

1999년~2006년 삼성전자 선임/책임연구원

2013년~2015년 KAIST 연구 조교수

2016년~현 재 경상대학교 항공우주 및 소프트웨어공학전공 정보과학과 조교수, 항공기부품기술연구소 멤버

관심분야 : Software Architecture & Software Repository Mining & Recommendation System & Software Evolution