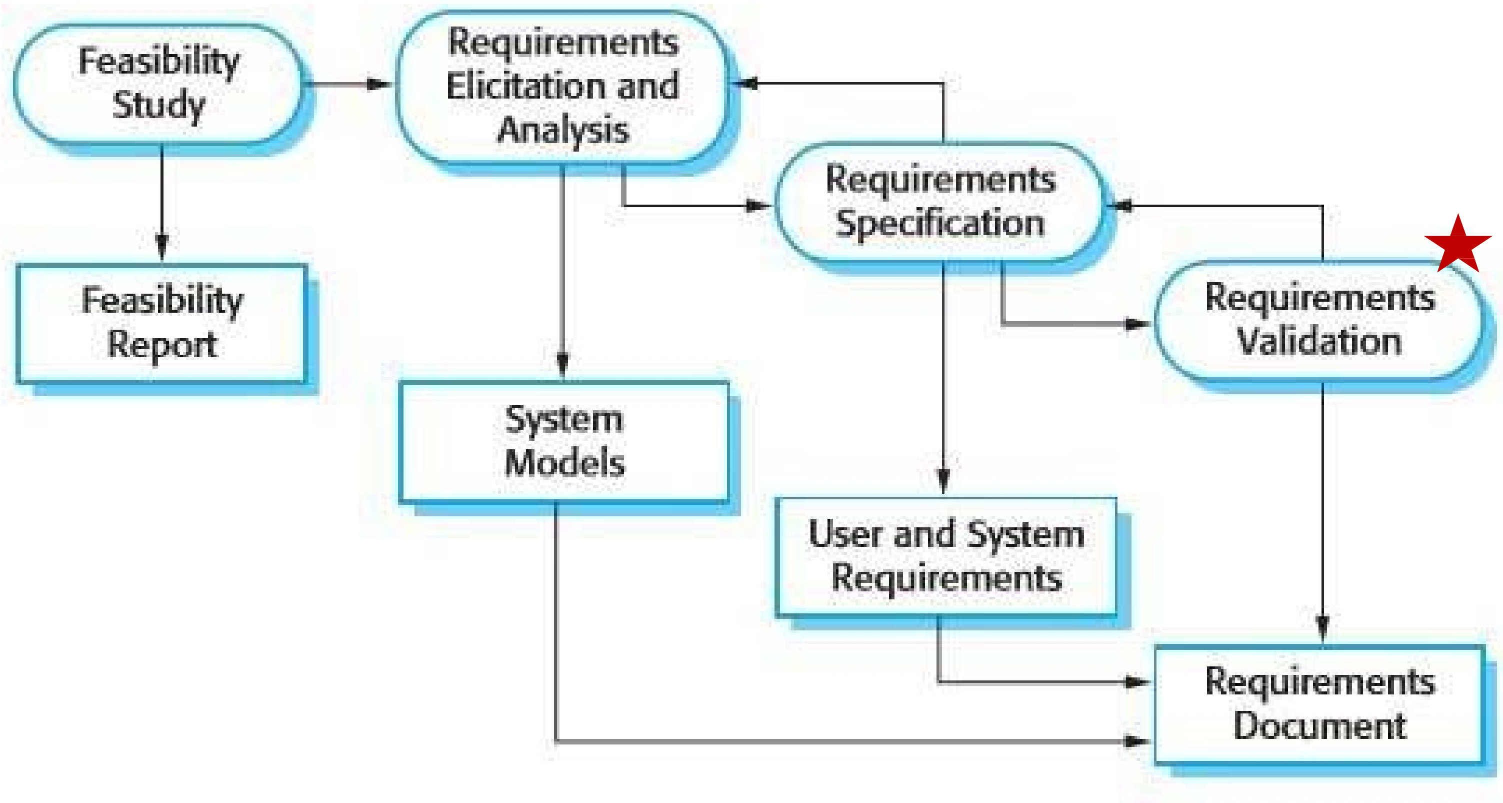
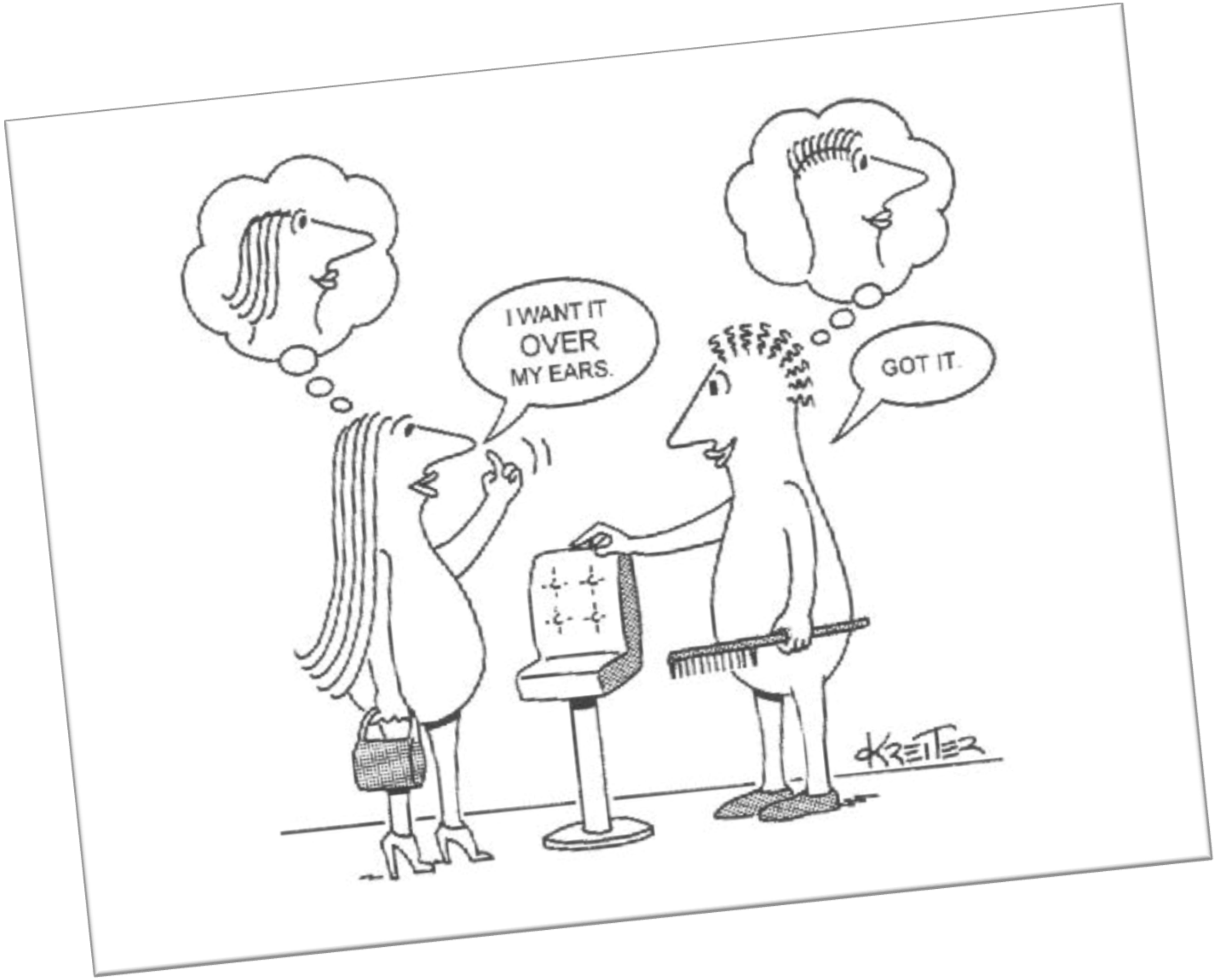
9. Requirements Validation

Requirements Engineering Process

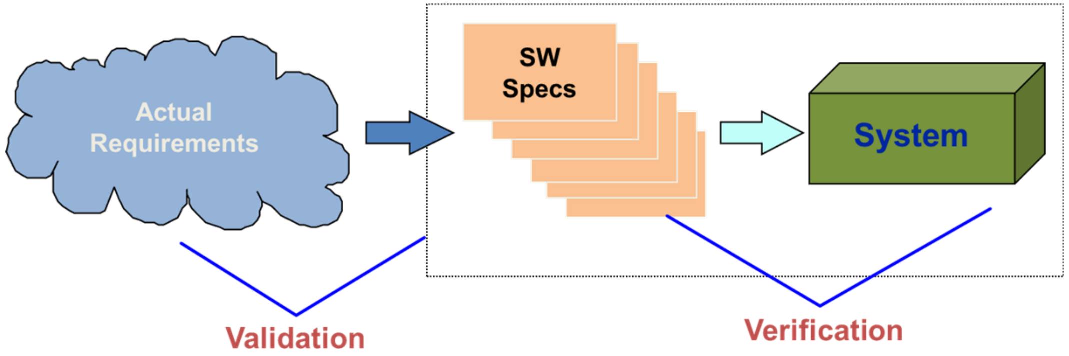



I WANT IT OVER MY EARS.
GOT IT.
KREITER

Verification and Validation in SDLC

- **Validation**: “Does the software system meets the user's real needs?”
  - Are we building the right software?
  - Does our design meet the spec?
  - Does our implementation meet the spec?
  - Does the delivered system do what we said it would do?
  - Are our requirements models consistent with one another?

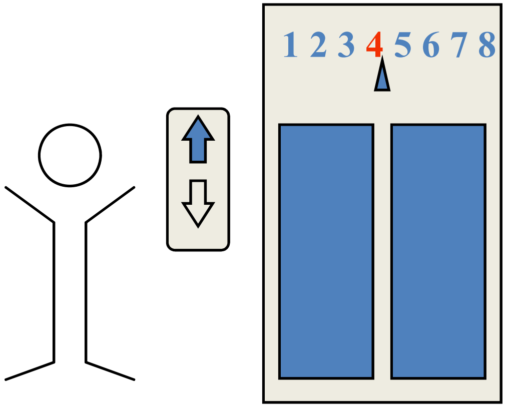
- **Verification**: “Does the software system meets the requirements specifications?”
  - Are we building the software right?
  - Does our problem statement accurately capture the real problem?
  - Did we account for the needs of all the stakeholders?



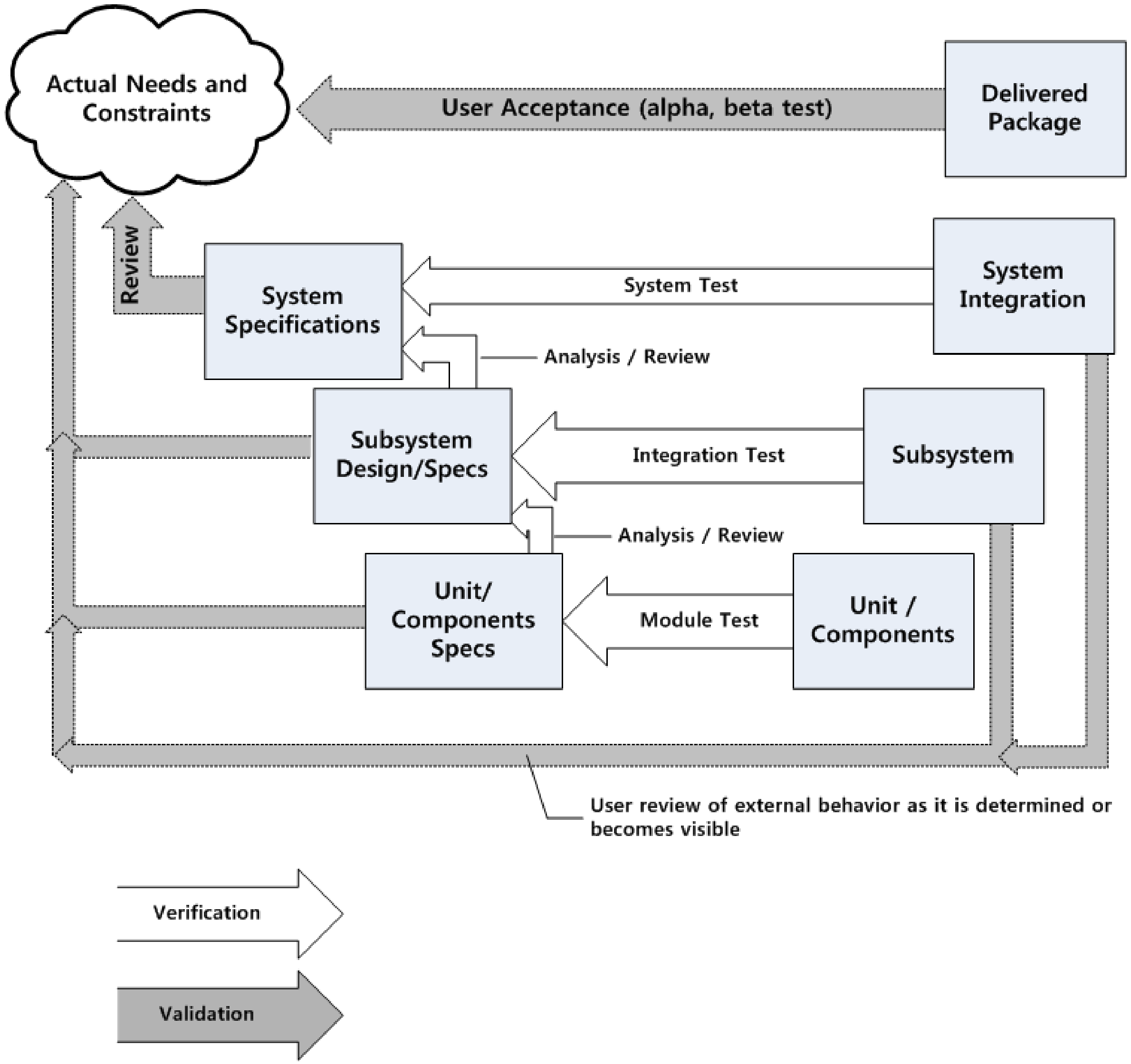
V&V Depends on the Specification

- Unverifiable (but validatable) specification: “If a user presses a request button at floor i, an available elevator must arrive at floor i soon.“

- Verifiable specification: “If a user presses a request button at floor i, an available elevator must arrive at floor i within 30 seconds“



V-Model of V&V Activities in SDLC



V&V for Requirements Models

- **Verification**
  - *"Is the model well-formed?"*
  - *"Are the parts of the model consistent with one another?"*

- **Validation:**
  - **Animation** of the model on small examples is possible.
  - 'What if' questions:
    - Reasoning about the consequences of particular requirements;
    - Reasoning about the effect of possible changes
    - "Will the system ever do the following,"
  - State exploration
    - E.g., use **model checking** to find traces that satisfy some property

- Generation techniques for requirements validation
  - Prototyping (Simulation)
  - Test-case generation
  - Review

Reviews, Walkthroughs, Inspections

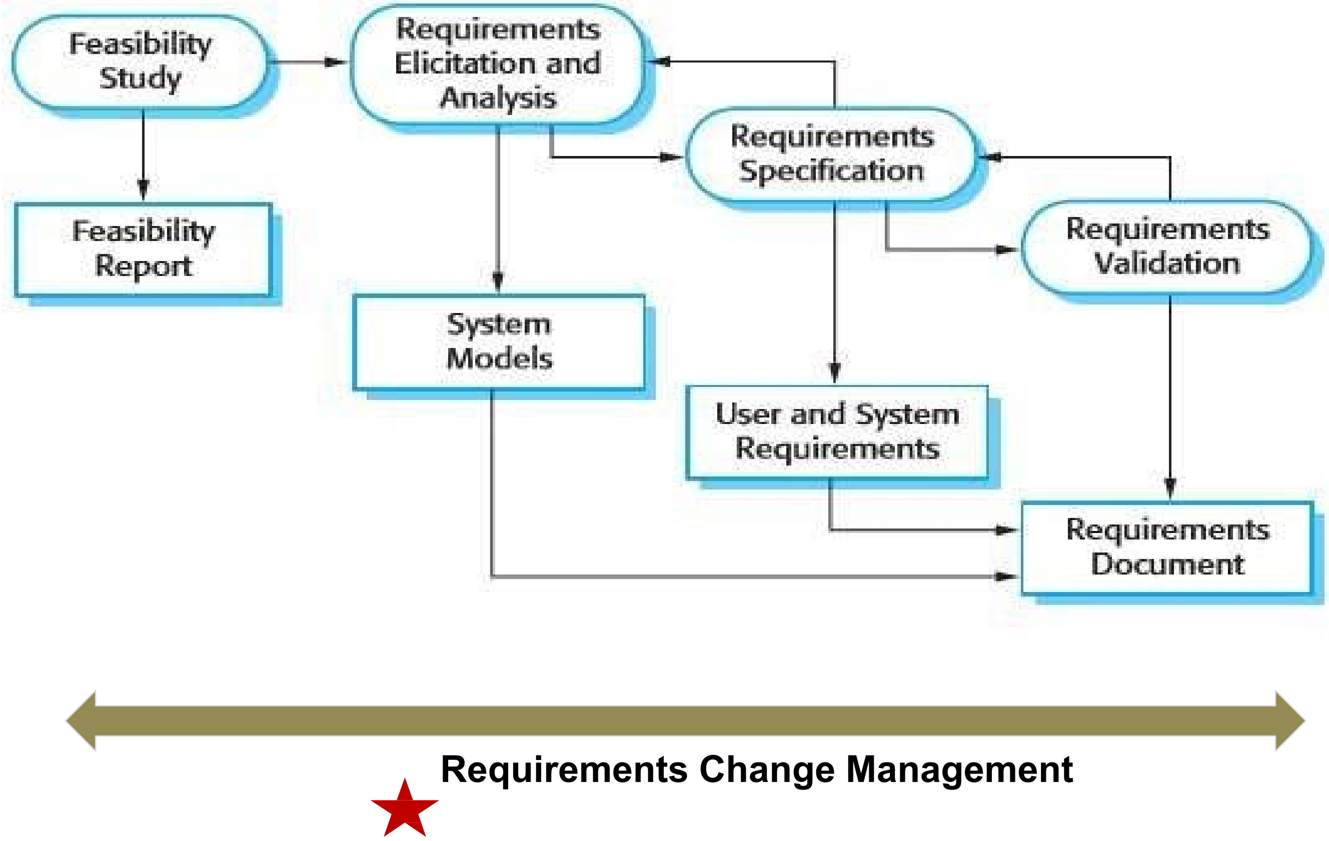
- **Management Reviews**
  - Preliminary design review (PDR), critical design review (CDR), formal technical review (FTR), formal business review (FBR), etc.
  - Used to provide confidence that the design is sound
  - Attended by management and sponsors (customers)

- **Walkthroughs**
  - Developer technique (usually informal)
  - Used by development teams to improve quality of product
  - Focusing on finding defects

- **(Fagan) Inspections**
  - A process management tool
  - Used to improve quality of the development process
  - Collect defect data to analyze the quality of the process
  - Written output is important

10. Requirements Change Management

Requirements Engineering Process

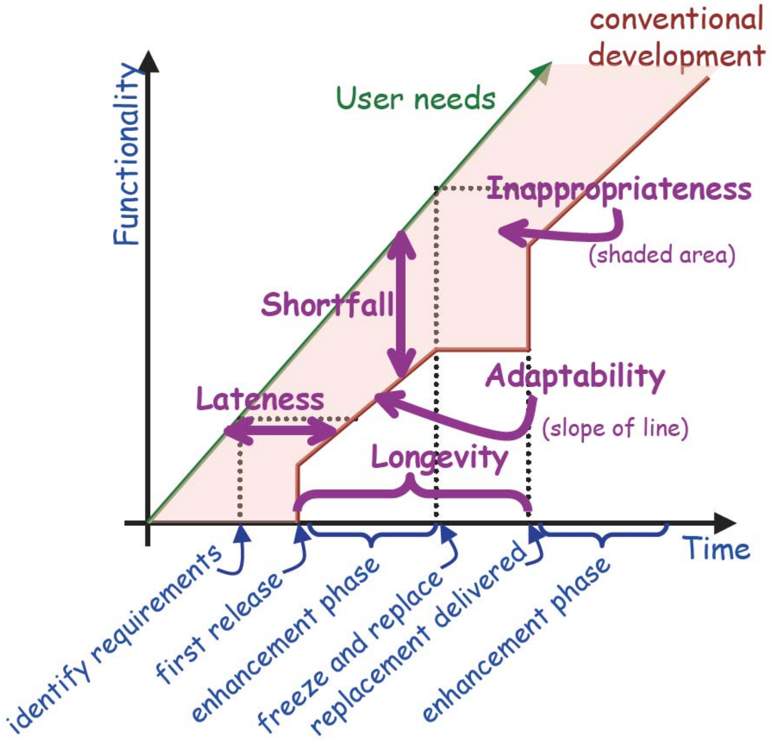

Feasibility Study
Requirements Elicitation and Analysis
Requirements Specification
Requirements Validation
Feasibility Report
System Models
User and System Requirements
Requirements Document
Requirements Change Management

Laws of Program Evolution

- **Continuing Change**
  - Any software that reflects some external reality undergoes continual change or becomes progressively less useful
    - Change continues until it is judged more cost effective to replace the system
- **Increasing Complexity**
  - As software evolves, its complexity increases
- **Fundamental Law of Program Evolution**
  - Software evolution is self-regulating
    - With statistically determinable trends and invariants
- **Conservation of Organizational Stability**
  - During the active life of a software system, the work output of a development project is roughly constant, regardless of resources
- **Conservation of Familiarity**
  - The amount of change in successive releases is roughly constant

Requirements Growth Model

- **Davis's model(1988):**
  - User needs evolve continuously
    - May not be linear or continuous (hence no scale shown)
  - Traditional development always lags behind needs growth
    - First release implements only part of the original requirements
    - Functional enhancement adds new functionality
    - Eventually, further enhancement becomes too costly, and a replacement is planned
    - The replacement also only implements part of its requirements,
    - and so on...



Software Aging

- **Causes of Software Aging**
  - Failure to update the software to meet changing needs
    - Customers switch to a new product, if benefits outweigh switching costs
  - Changes to software tend to reduce its coherence

- **Costs of Software Aging**
  - Owners of aging software find it hard to keep up with the marketplace
  - Deterioration in space/time performance due to deteriorating structure
  - Aging software gets more buggy
    - Each “bug fix” introduces more errors than it fixes

- **Ways of Increasing longevity**
  - Design for change
    - Design patterns
    - Architecture styles
  - Document the software carefully
  - Requirements and designs should be reviewed by those responsible for its maintenance
  - Software Rejuvenation

Software Maintenance

- Maintenance philosophies
  - "Throw-it-over-the-wall" : someone else is responsible for maintenance
    - Investment in knowledge and experience is lost
    - Maintenance becomes a reverse engineering challenge
  - "Mission orientation" : development team make a long term commitment to maintaining/enhancing the software

- **Basili's maintenance process models**:
  - **Quick-fix model**
    - Changes made at the code level, as easily as possible
    - Rapidly degrades the structure of the software
  - **Iterative enhancement model**
    - Changes made based on an analysis of the existing system
    - Attempts to control complexity and maintain good design
  - **Full-reuse model**
    - Starts with requirements for the new system, reusing as much as possible
    - Needs a mature reuse culture to be successful

Managing Requirements Change

- Managers need to respond to requirements change
  - Adding new requirements during development
  - Modifying requirements during development
  - Removing requirements during development

- **Key techniques**
  - Change Management (Process)
  - Release Planning
  - Requirements Prioritization
  - Requirements Traceability
  - Architectural Stability

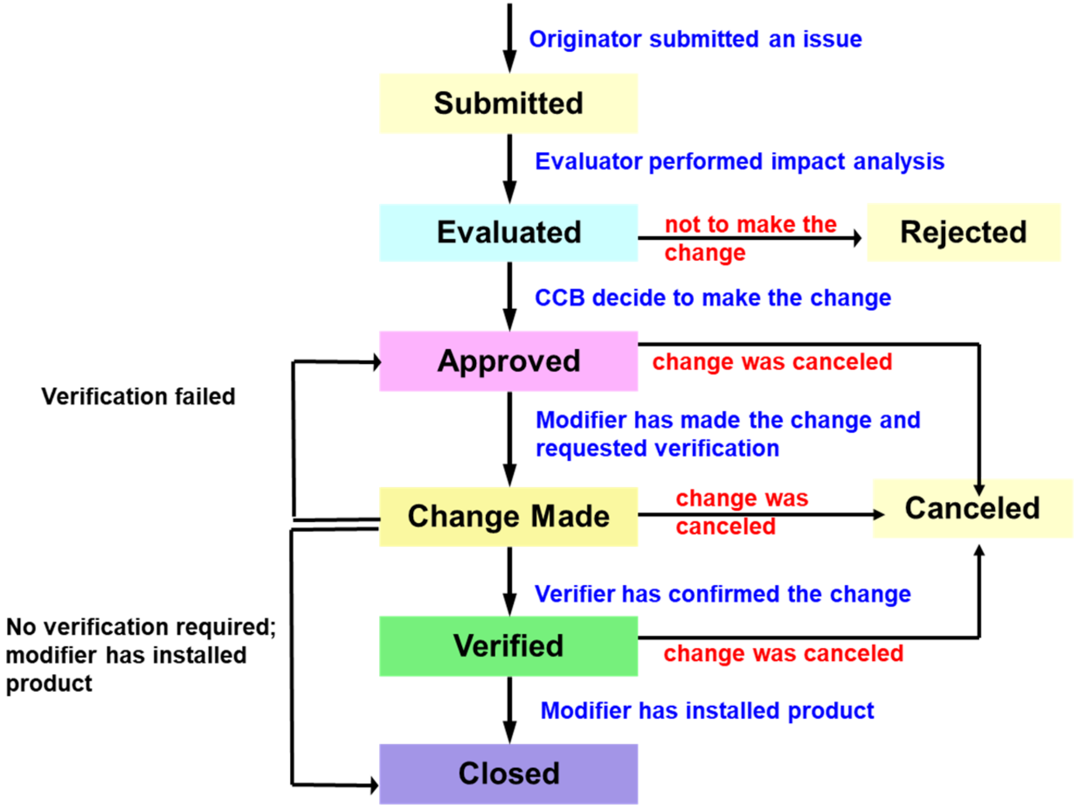
Change Management

- **Configuration Management**
  - Each distinct product is a Configuration Item (CI)
  - Each configuration item is placed under version control
  - Control which version of each CI belongs to which build of the system

- **Baseline**
  - A stable version of a document or system
    - Safe to share among the team
  - Formal approval process for changes should be incorporated into the next baseline

Change Management Process

- **Change Management Process**
  - All proposed changes are submitted formally as change requests
  - A review board reviews these periodically and decides which to accept


Originator submitted an issue
Submitted
Evaluator performed impact analysis
Evaluated
not to make the change
Rejected
CCB decide to make the change
Approved
change was canceled
Verification failed
Modifier has made the change and requested verification
Change Made
change was canceled
Canceled
Verifier has confirmed the change
No verification required; modifier has installed product
Verified
change was canceled
Modifier has installed product
Closed

Requirements Traceability

- **From IEEE-STD-830.1998:**
  - **Backward traceability**
    - To previous stages of development
    - The origin of each requirement should be clear
  - **Forward traceability**
    - To all documents spawned by the SRS
    - Facilitation of referencing of each requirement in future documentation

- **From DOD-STD-2167A:**
  - A requirements specification is traceable if:
    1) It contains or implements all applicable stipulations in predecessor document
    2) A given term, acronym, or abbreviation means the same thing in all documents
    3) A given item or concept is referred to by the same name in the documents
    4) All material in the successor document has its basis in the predecessor document, that is, no untraceable material has been introduced
    5) The two documents do not contradict one another

Importance of Traceability

- **Verification and Validation**
  - Assessing adequacy of test suite
  - Assessing conformance to requirements
  - Assessing completeness, consistency and impact analysis
  - Investigating high level behavior impact on detailed specifications
  - Detecting requirements conflicts
  - Checking consistency of decision making across the lifecycle

- **Maintenance**
  - Assessing change requests
  - Tracing design rationale

- **Document access**
  - Ability to find information quickly in large documents

- **Process visibility**
  - Ability to see how the software was developed
  - Provides an audit trail

- **Management**
  - Change management
  - Risk management
  - Control of the development process

Traceability Difficulties

- **Cost**
  - Very little automated support
  - Full traceability is very expensive and time-consuming

- **Delayed gratification**
  - The people defining traceability links are not the people who benefit from it
    - Development vs. V&V
  - Much of the benefit comes late in the lifecycle
    - Testing, integration, maintenance

- **Size and diversity**
  - Huge range of different document types, tools, decisions and responsibilities
  - No common schema exists for classifying and cataloging these
  - In practice, traceability concentrates only on baselined requirements

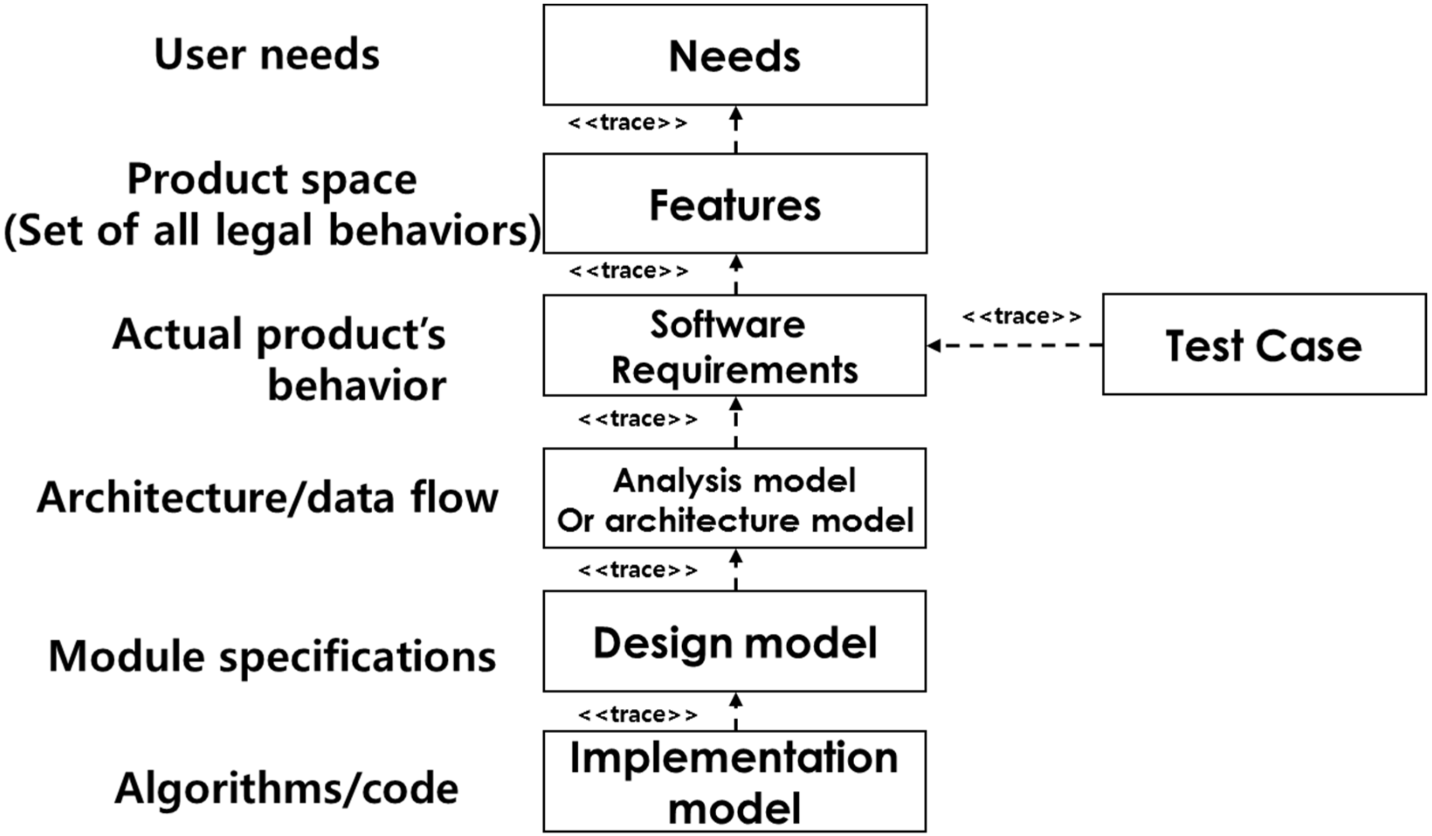
Traceability in Practice

- **Coverage**
  - Forward: Links from requirements forward to designs, code, test cases,
  - Backward: Links back from designs, code, test cases to requirements
  - links between requirements at different levels

- **Traceability process**
  - Assign each sentence or paragraph a unique id number
  - Manually identify linkages
  - Use manual tables to record linkages in a document
  - Use a traceability tool (database) for project wide traceability
  - Tool then offers ability to
    - Follow links
    - Find missing links
    - Measure overall traceability

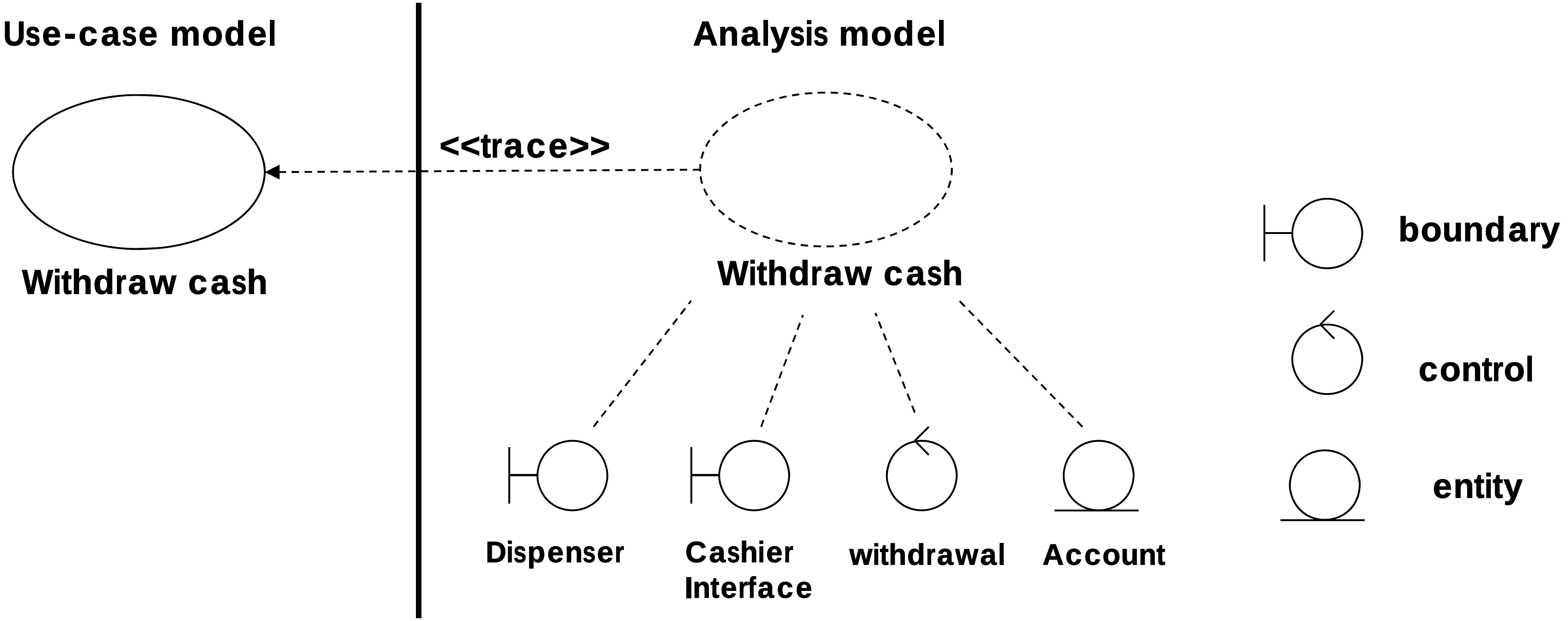
Example : Requirements Traceability

- When a high level artifact derives a refined artifact, **Traceability link** should be generated between two artifacts.



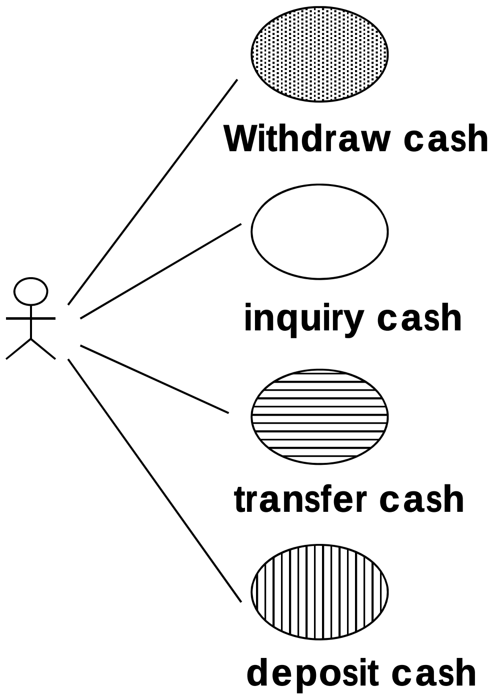
Traceability Link Example: An ATM System

- Using **Use Case**
  - For a use case, finding participating class based on categorization of application classes (boundary, control, entity)

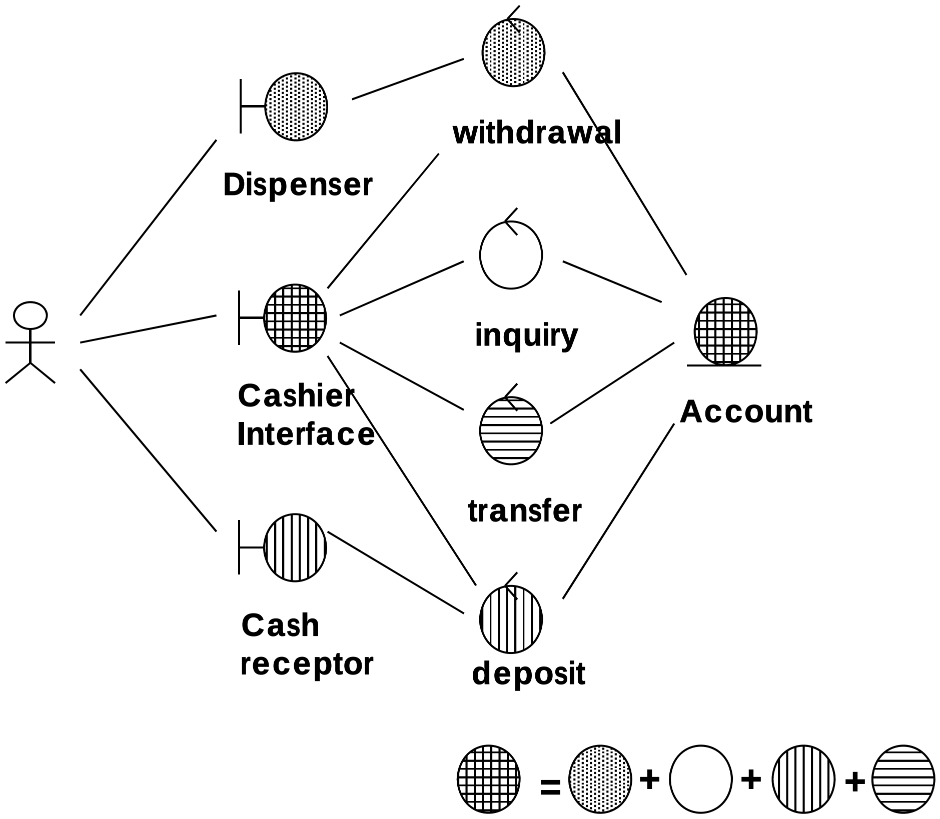


- Each **use case** derives a participating **analysis case**.

Use-case model

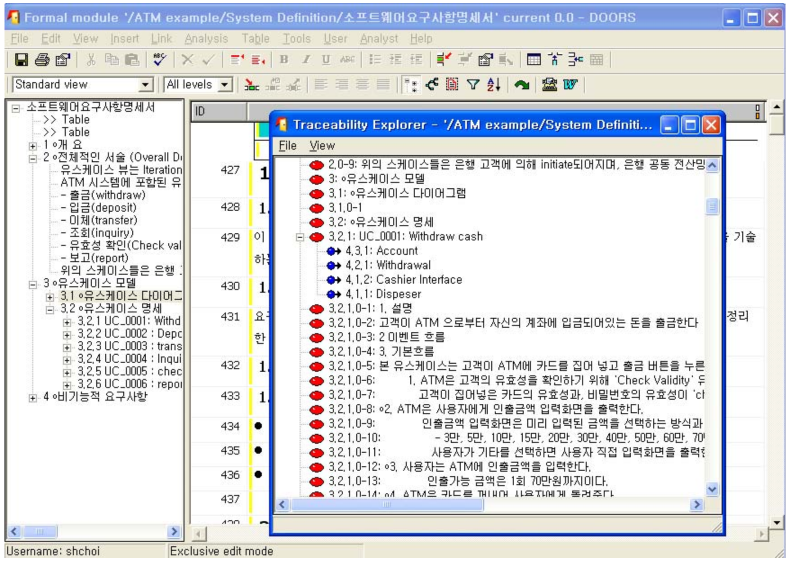


Analysis model



- Tracing the link using **DOORS** (Use case to analysis model)

소프트웨어 요구사항 명세서의 해당 Use Case 절을 해당되는 분석클래스와 연결한다.

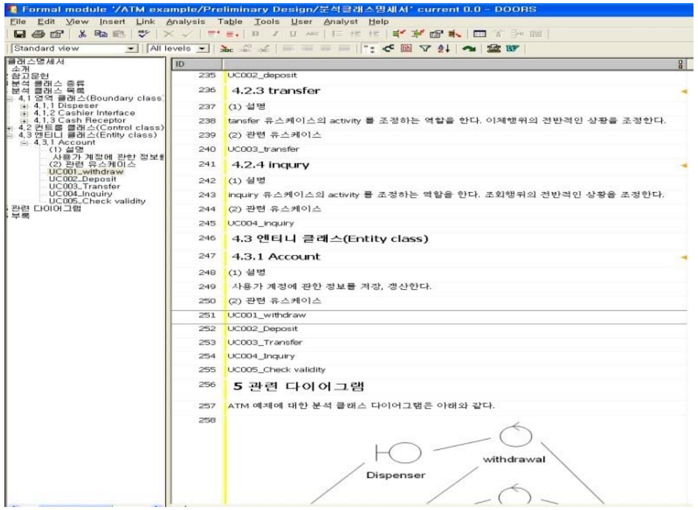

Formal module '/ATM example/System Definition/소프트웨어요구사항명세서' current 0.0 - DOORS
File Edit View Insert Link Analysis Table Tools User Analyst Help
Standard view
All levels
ID
Traceability Explorer - '/ATM example/System Definiti...
File View
Username: shchoi
Exclusive edit mode

소프트웨어요구사항명세서
>> Table
>> Table
1 ◦개 요
2 ◦전체적인 서술 (Overall D
유스케이스 뷰는 Iteration
ATM 시스템에 포함된 유
- 출금(withdraw)
- 입금(deposit)
- 이체(transfer)
- 조회(inquiry)
- 유효성 확인(Check val
- 보고(report)
위의 스케이스들은 은행
3 ◦유스케이스 모델
3.1 ◦유스케이스 다이어그
3.2 ◦유스케이스 명세
3.2.1 UC_0001: Withd
3.2.2 UC_0002 : Depo
3.2.3 UC_0003 : trans
3.2.4 UC_0004 : Inqui
3.2.5 UC_0005 : chec
3.2.6 UC_0006 : repo
4 ◦비기능적 요구사항

2.0-9: 위의 스케이스들은 은행 고객에 의해 initiate되어지며, 은행 공동 전산망
3: ◦유스케이스 모델
3.1: ◦유스케이스 다이어그램
3.1.0-1
3.2: ◦유스케이스 명세
3.2.1: UC_0001: Withdraw cash
4.3.1: Account
4.2.1: Withdrawal
4.1.2: Cashier Interface
4.1.1: Dispeser
3.2.1.0-1: 1. 설명
3.2.1.0-2: 고객이 ATM 으로부터 자신의 계좌에 입금되어있는 돈을 출금한다
3.2.1.0-3: 2 이벤트 흐름
3.2.1.0-4: 3. 기본흐름
3.2.1.0-5: 본 유스케이스는 고객이 ATM에 카드를 집어 넣고 출금 버튼을 누른
3.2.1.0-6: 1. ATM은 고객의 유효성을 확인하기 위해 'Check Validity' 유
3.2.1.0-7: 고객이 집어넣은 카드의 유효성과, 비밀번호의 유효성이 'ch
3.2.1.0-8: ◦2. ATM은 사용자에게 인출금액 입력화면을 출력한다.
3.2.1.0-9: 인출금액 입력화면은 미리 입력된 금액을 선택하는 방식과
3.2.1.0-10: - 3만, 5만, 10만, 15만, 20만, 30만, 40만, 50만, 60만, 70
3.2.1.0-11: 사용자가 기타를 선택하면 사용자 직접 입력화면을 출력
3.2.1.0-12: ◦3. 사용자는 ATM에 인출금액을 입력한다.
3.2.1.0-13: 인출가능 금액은 1회 70만원까지이다.
3.2.1.0-14: ◦4. ATM은 카드를 꺼내어 사용자에게 돌려준다.

소프트웨어 요구사항 명세서

Traceability explorer 를 통해 연계상황을 확인할 수 있다.

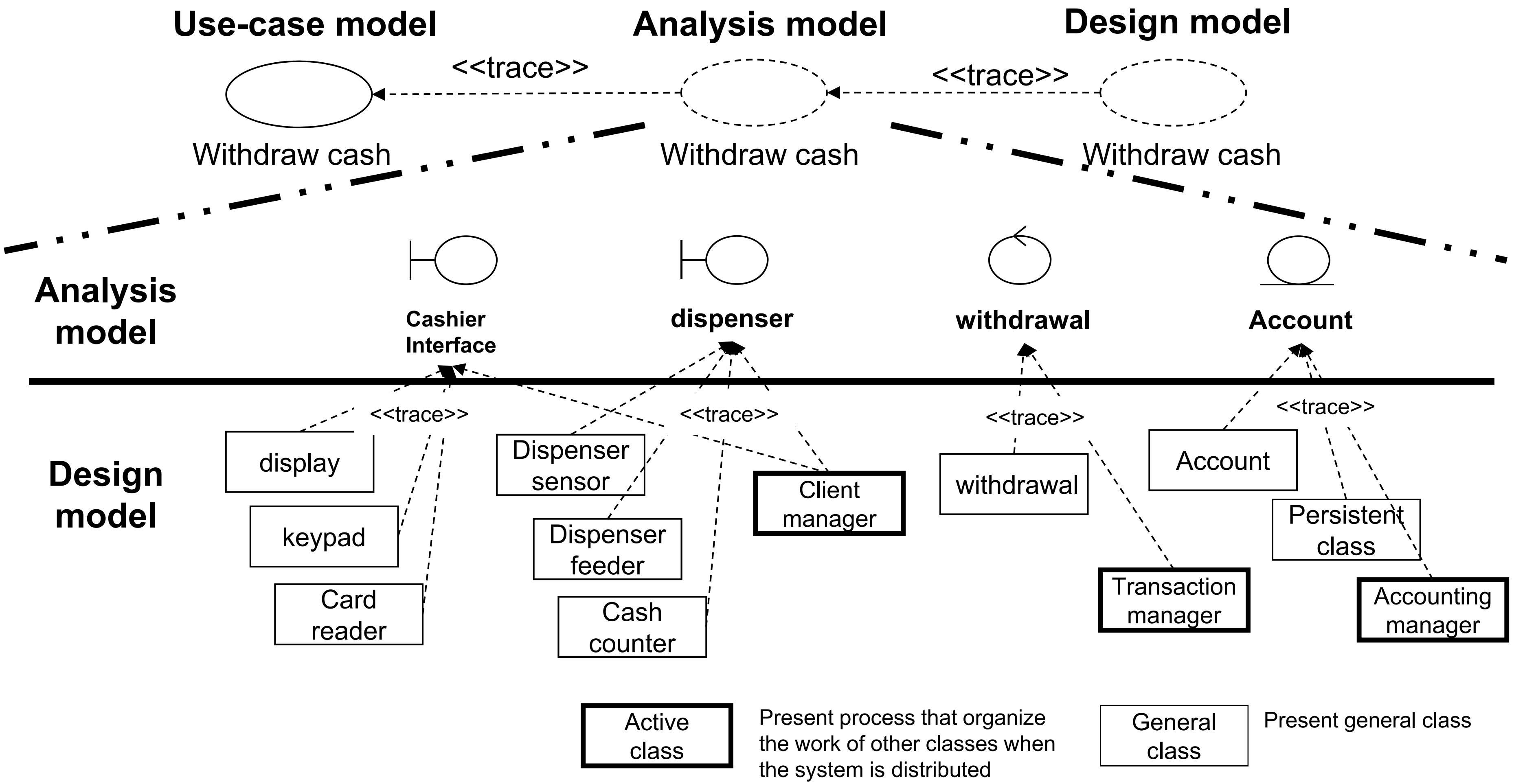

Formal module '/ATM example/Preliminary Design/분석클래스명세서' current 0.0 - DOORS
File Edit View Insert Link Analysis Table Tools User Analyst Help
Standard view
All levels
ID
Dispenser
withdrawal

클래스명세서
소개
참고문헌
분석 클래스 종류
분석 클래스 목록
4.1 영역 클래스(Boundary class)
4.1.1 Dispeser
4.1.2 Cashier Interface
4.1.3 Cash Receptor
4.2 컨트롤 클래스(Control class)
4.3 엔티니 클래스(Entity class)
4.3.1 Account
(1) 설명
사용가 계정에 관한 정보
(2) 관련 유스케이스
UC001_withdraw
UC002_Deposit
UC003_Transfer
UC004_Inquiry
UC005_Check validity
관련 다이어그램
부록

235 UC002_deposit
236 4.2.3 transfer
237 (1) 설명
238 tansfer 유스케이스의 activity 를 조정하는 역할을 한다. 이체행위의 전반적인 상황을 조정한다.
239 (2) 관련 유스케이스
240 UC003_transfer
241 4.2.4 inqury
242 (1) 설명
243 inquiry 유스케이스의 activity 를 조정하는 역할을 한다. 조회행위의 전반적인 상황을 조정한다.
244 (2) 관련 유스케이스
245 UC004_inquiry
246 4.3 엔티니 클래스(Entity class)
247 4.3.1 Account
248 (1) 설명
249 사용가 계정에 관한 정보를 저장, 갱신한다.
250 (2) 관련 유스케이스
251 UC001_withdraw
252 UC002_Deposit
253 UC003_Transfer
254 UC004_Inquiry
255 UC005_Check validity
256 5 관련 다이어그램
257 ATM 예제에 대한 분석 클래스 다이어그램은 아래와 같다.
258

분석 클래스 명세서

- Analysis class derives **design class** in design model.

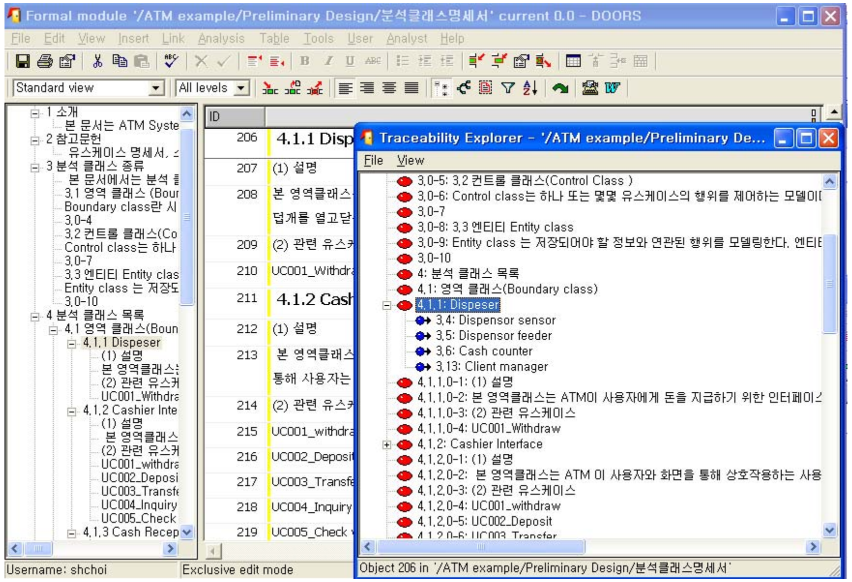


Present process that organize the work of other classes when the system is distributed

Present general class

- Tracing the link using **DOORS** (Analysis model to design model)

분석모델명세서의 해당 클래스와
디자인명세서의 해당 클래스를 연결한다.


Formal module '/ATM example/Preliminary Design/분석클래스명세서' current 0.0 - DOORS

File Edit View Insert Link Analysis Table Tools User Analyst Help
Standard view
All levels

1 소개
본 문서는 ATM Syste
2 참고문헌
유스케이스 명세서,
3 분석 클래스 종류
본 문서에서는 분석
3.1 영역 클래스 (Bour
Boundary class란 시
3.0-4
3.2 컨트롤 클래스(Co
Control class는 하나
3.0-7
3.3 엔티티 Entity clas
Entity class 는 저장되
3.0-10
4 분석 클래스 목록
4.1 영역 클래스(Boun
4.1.1 Dispeser
(1) 설명
본 영역클래스는
(2) 관련 유스케
UC001_Withdra
4.1.2 Cashier Inte
(1) 설명
본 영역클래스
(2) 관련 유스케
UC001_withdra
UC002_Deposi
UC003_Transfe
UC004_Inquiry
UC005_Check
4.1.3 Cash Recep

ID	
206	4.1.1 Disp
207	(1) 설명
208	본 영역클래스 덥개를 열고닫
209	(2) 관련 유스케
210	UC001_Withdra
211	4.1.2 Cash
212	(1) 설명
213	본 영역클래스 통해 사용자는
214	(2) 관련 유스케
215	UC001_withdra
216	UC002_Deposit
217	UC003_Transfe
218	UC004_Inquiry
219	UC005_Check

Username: shchoi
Exclusive edit mode

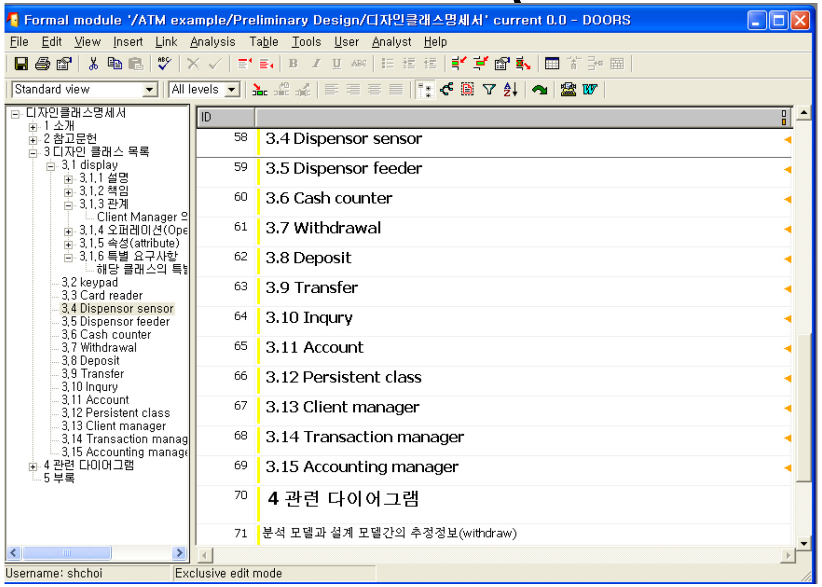
Traceability Explorer - '/ATM example/Preliminary De...

3.0-5: 3.2 컨트롤 클래스(Control Class)
3.0-6: Control class는 하나 또는 몇몇 유스케이스의 행위를 제어하는 모델이
3.0-7
3.0-8: 3.3 엔티티 Entity class
3.0-9: Entity class 는 저장되어야 할 정보와 연관된 행위를 모델링한다. 엔티티
3.0-10
4: 분석 클래스 목록
4.1: 영역 클래스(Boundary class)
4.1.1: Dispeser
3.4: Dispensor sensor
3.5: Dispensor feeder
3.6: Cash counter
3.13: Client manager
4.1.1.0-1: (1) 설명
4.1.1.0-2: 본 영역클래스는 ATM이 사용자에게 돈을 지급하기 위한 인터페이스
4.1.1.0-3: (2) 관련 유스케이스
4.1.1.0-4: UC001_Withdraw
4.1.2: Cashier Interface
4.1.2.0-1: (1) 설명
4.1.2.0-2: 본 영역클래스는 ATM 이 사용자와 화면을 통해 상호작용하는 사용
4.1.2.0-3: (2) 관련 유스케이스
4.1.2.0-4: UC001_withdraw
4.1.2.0-5: UC002_Deposit
4.1.2.0-6: UC003_Transfer

Object 206 in '/ATM example/Preliminary Design/분석클래스명세서'

분석모델 명세서

Traceability explorer 를 통해
연계상황을 확인할 수 있다.


Formal module '/ATM example/Preliminary Design/디자인클래스명세서' current 0.0 - DOORS

File Edit View Insert Link Analysis Table Tools User Analyst Help
Standard view
All levels

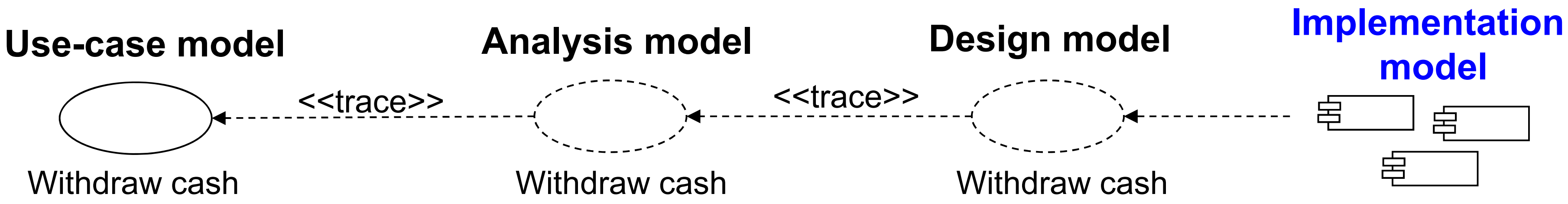
디자인클래스명세서
1 소개
2 참고문헌
3 디자인 클래스 목록
3.1 display
3.1.1 설명
3.1.2 책임
3.1.3 관계
Client Manager 으
3.1.4 오퍼레이션(Ope
3.1.5 속성(attribute)
3.1.6 특별 요구사항
해당 클래스의 특
3.2 keypad
3.3 Card reader
3.4 Dispensor sensor
3.5 Dispensor feeder
3.6 Cash counter
3.7 Withdrawal
3.8 Deposit
3.9 Transfer
3.10 Inqury
3.11 Account
3.12 Persistent class
3.13 Client manager
3.14 Transaction manag
3.15 Accounting manage
4 관련 다이어그램
5 부록

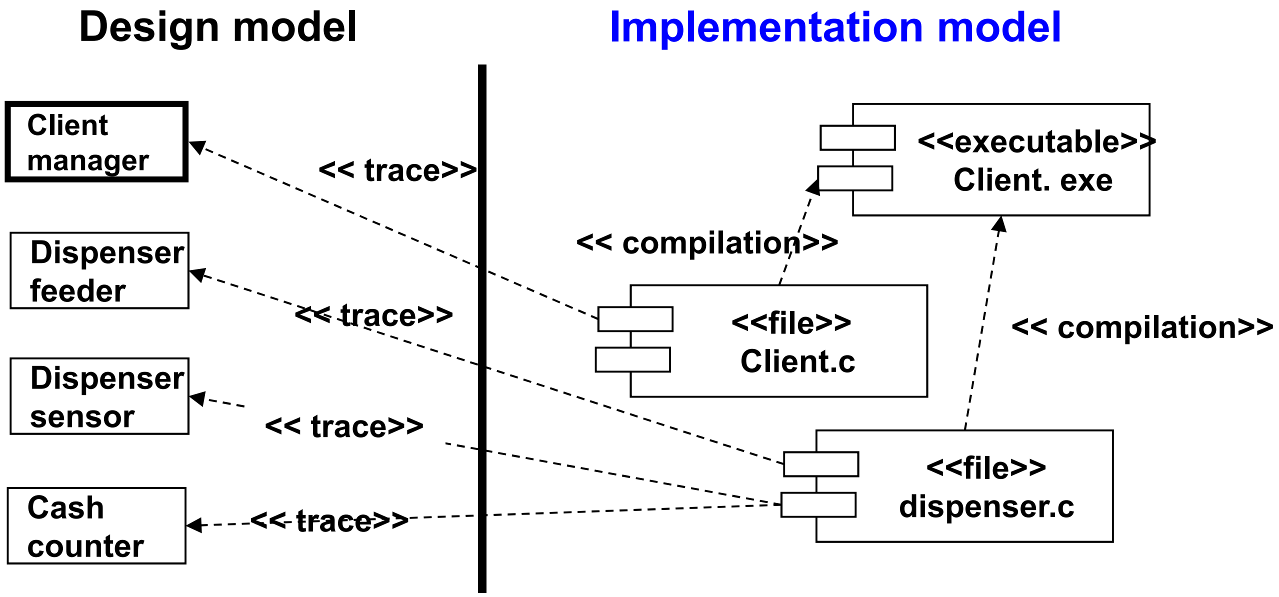
ID	
58	3.4 Dispensor sensor
59	3.5 Dispensor feeder
60	3.6 Cash counter
61	3.7 Withdrawal
62	3.8 Deposit
63	3.9 Transfer
64	3.10 Inqury
65	3.11 Account
66	3.12 Persistent class
67	3.13 Client manager
68	3.14 Transaction manager
69	3.15 Accounting manager
70	4 관련 다이어그램
71	분석 모델과 설계 모델간의 추정정보(withdraw)

Username: shchoi
Exclusive edit mode

디자인모델 명세서

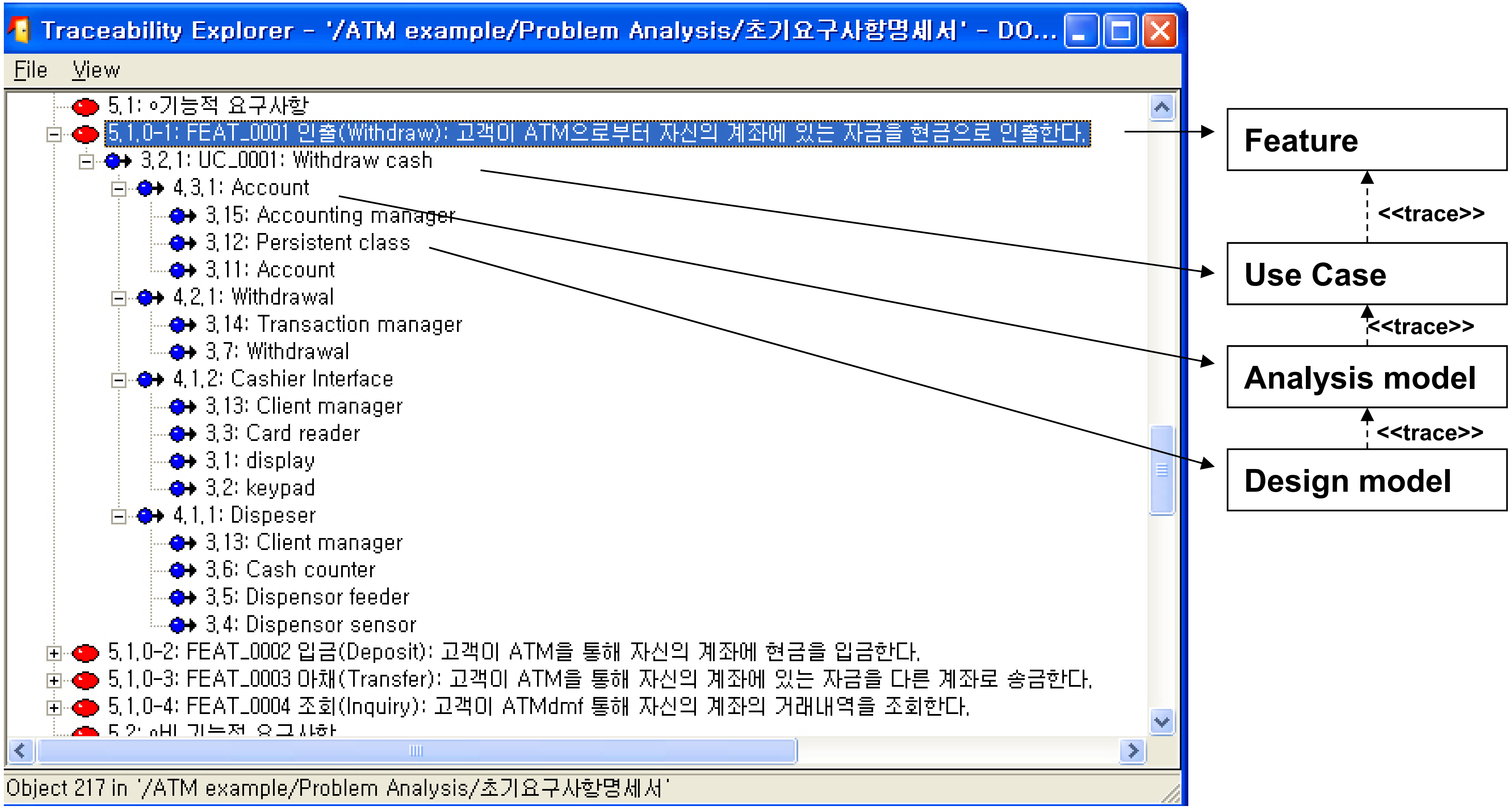
- Design classes derive **components** in implementation model.





Partial implementation model from design model

- Traceability link in DOORS

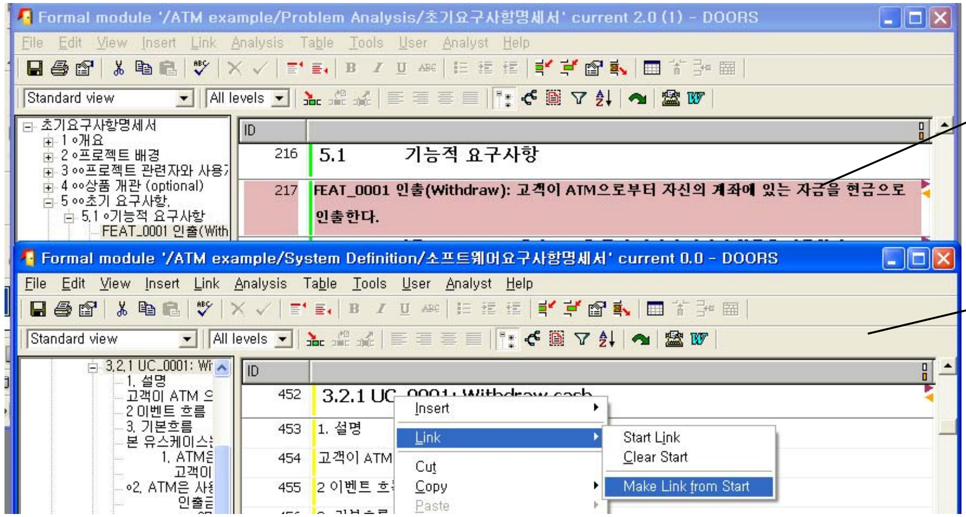


Traceability Explorer - '/ATM example/Problem Analysis/초기요구사항명세서' - DO...

```
5,1: ㅇ기능적 요구사항
5,1,0-1: FEAT_0001 인출(Withdraw): 고객이 ATM으로부터 자신의 계좌에 있는 자금을 현금으로 인출한다.
    3,2,1: UC_0001: Withdraw cash
        4,3,1: Account
            3,15: Accounting manager
            3,12: Persistent class
            3,11: Account
        4,2,1: Withdrawal
            3,14: Transaction manager
            3,7: Withdrawal
        4,1,2: Cashier Interface
            3,13: Client manager
            3,3: Card reader
            3,1: display
            3,2: keypad
        4,1,1: Dispeser
            3,13: Client manager
            3,6: Cash counter
            3,5: Dispensor feeder
            3,4: Dispensor sensor
5,1,0-2: FEAT_0002 입금(Deposit): 고객이 ATM을 통해 자신의 계좌에 현금을 입금한다.
5,1,0-3: FEAT_0003 아채(Transfer): 고객이 ATM을 통해 자신의 계좌에 있는 자금을 다른 계좌로 송금한다.
5,1,0-4: FEAT_0004 조회(Inquiry): 고객이 ATMdmf 통해 자신의 계좌의 거래내역을 조회한다.
5,2: ㅇ비 기능적 요구사항
```

Object 217 in '/ATM example/Problem Analysis/초기요구사항명세서'

- Set **links** between the requirements - **manually**



Formal module '/ATM example/Problem Analysis/초기요구사항명세서' current 2.0 (1) - DOORS

ID	
216	5.1 기능적 요구사항
217	FEAT_0001 인출(Withdraw): 고객이 ATM으로부터 자신의 계좌에 있는 자금을 현금으로 인출한다.

Formal module '/ATM example/System Definition/소프트웨어요구사항명세서' current 0.0 - DOORS

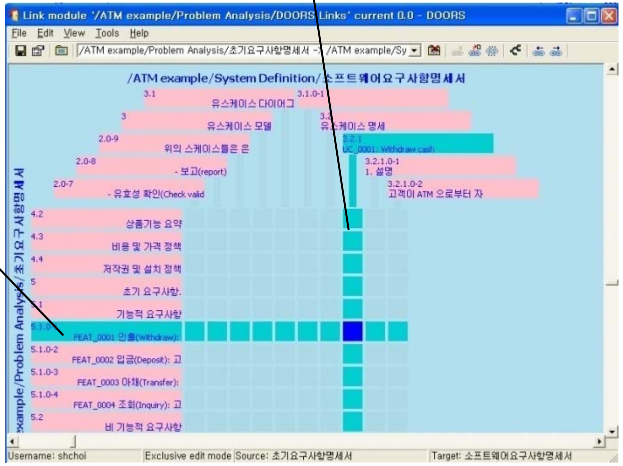
ID	
452	3.2.1 UC_0001: Withdraw cash
453	1. 설명
454	고객이 ATM
455	2 이벤트 흐

초기요구사항 명세서에 있는
feature

소프트웨어 요구사항 명세서에 있는
Use Case

연계설정을 원하는
2개의 모듈을 Open 한 뒤 연결

초기요구사항 명세서 (열)



Link module '/ATM example/Problem Analysis/DOORS Links' current 0.0 - DOORS

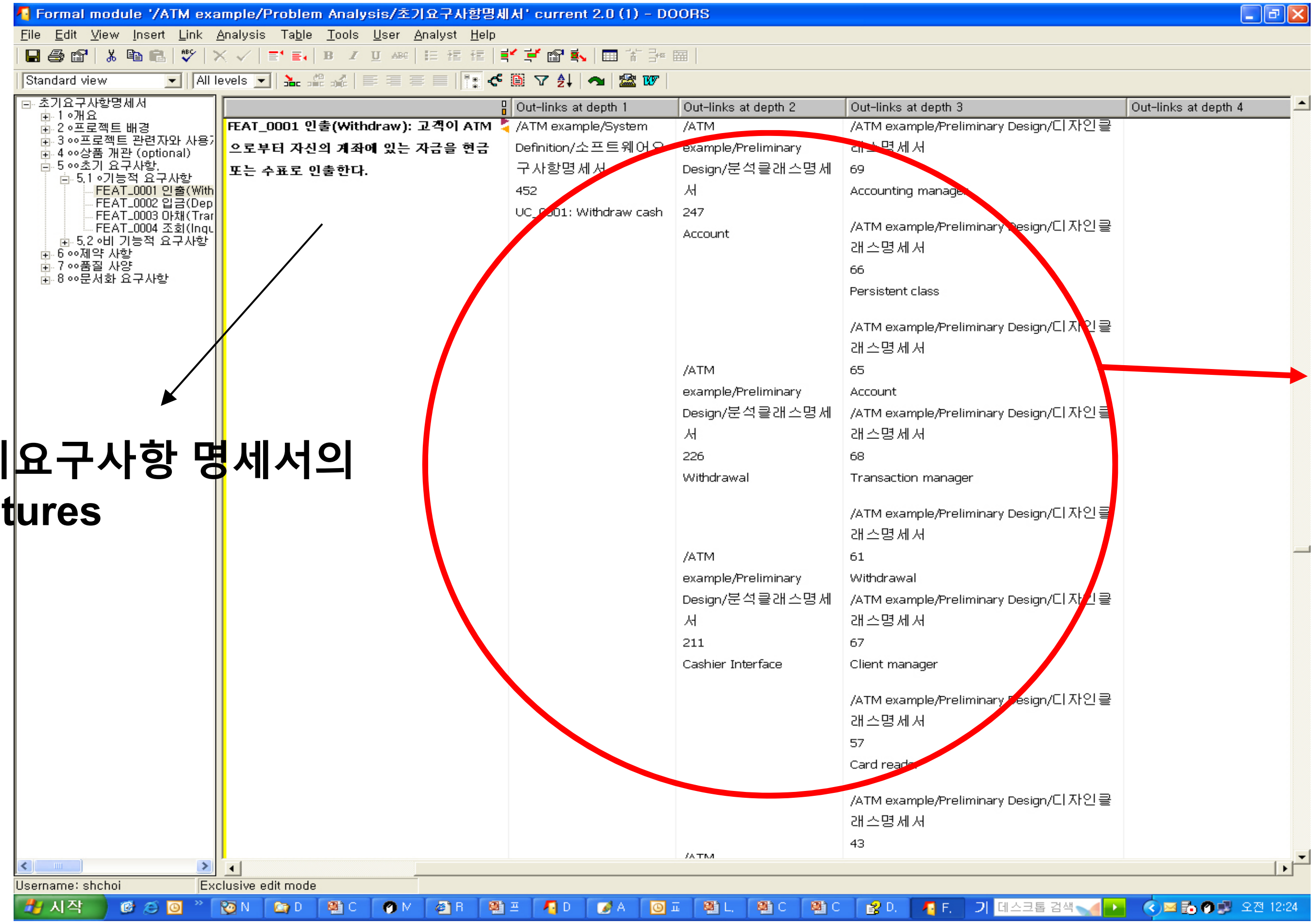
File Edit View Tools Help
/ATM example/Problem Analysis/초기요구사항명세서 -> /ATM example/Sy
/ATM example/System Definition/소프트웨어요구사항명세서
3.1 유스케이스 다이어그
3.1.0-1
3 유스케이스 모델
3.2 유스케이스 명세
2.0-9 위의 스케이스들은 은
3.2.1 UC_0001: Withdraw cash
2.0-8 - 보고(report)
3.2.1.0-1 1. 설명
2.0-7 - 유효성 확인(Check valid
3.2.1.0-2 고객이 ATM 으로부터 자
example/Problem Analysis/초기요구사항명세서
4.2 상품기능 요약
4.3 비용 및 가격 정책
4.4 저작권 및 설치 정책
5 초기 요구사항.
5.1 기능적 요구사항
5.1.0-1 FEAT_0001 인출(Withdraw):
5.1.0-2 FEAT_0002 입금(Deposit): 고
5.1.0-3 FEAT_0003 이체(Transfer):
5.1.0-4 FEAT_0004 조회(Inquiry): 고
5.2 비 기능적 요구사항

Username: shchoi | Exclusive edit mode | Source: 초기요구사항명세서 | Target: 소프트웨어요구사항명세서

소프트웨어
요구사항
명세서 (행)

Link module을 이용한 연결

- View relationships (**Traceability column**)



	Out-links at depth 1	Out-links at depth 2	Out-links at depth 3	Out-links at depth 4
FEAT_0001 인출(Withdraw): 고객이 ATM 으로부터 자신의 계좌에 있는 자금을 현금 또는 수표로 인출한다.	/ATM example/System Definition/소프트웨어요구사항명세서 452 UC_0001: Withdraw cash	/ATM example/Preliminary Design/분석클래스명세서 247 Account /ATM example/Preliminary Design/분석클래스명세서 226 Withdrawal /ATM example/Preliminary Design/분석클래스명세서 211 Cashier Interface /ATM	/ATM example/Preliminary Design/디자인클래스명세서 69 Accounting manager /ATM example/Preliminary Design/디자인클래스명세서 66 Persistent class /ATM example/Preliminary Design/디자인클래스명세서 65 Account /ATM example/Preliminary Design/디자인클래스명세서 68 Transaction manager /ATM example/Preliminary Design/디자인클래스명세서 61 Withdrawal /ATM example/Preliminary Design/디자인클래스명세서 67 Client manager /ATM example/Preliminary Design/디자인클래스명세서 57 Card reader /ATM example/Preliminary Design/디자인클래스명세서 43	

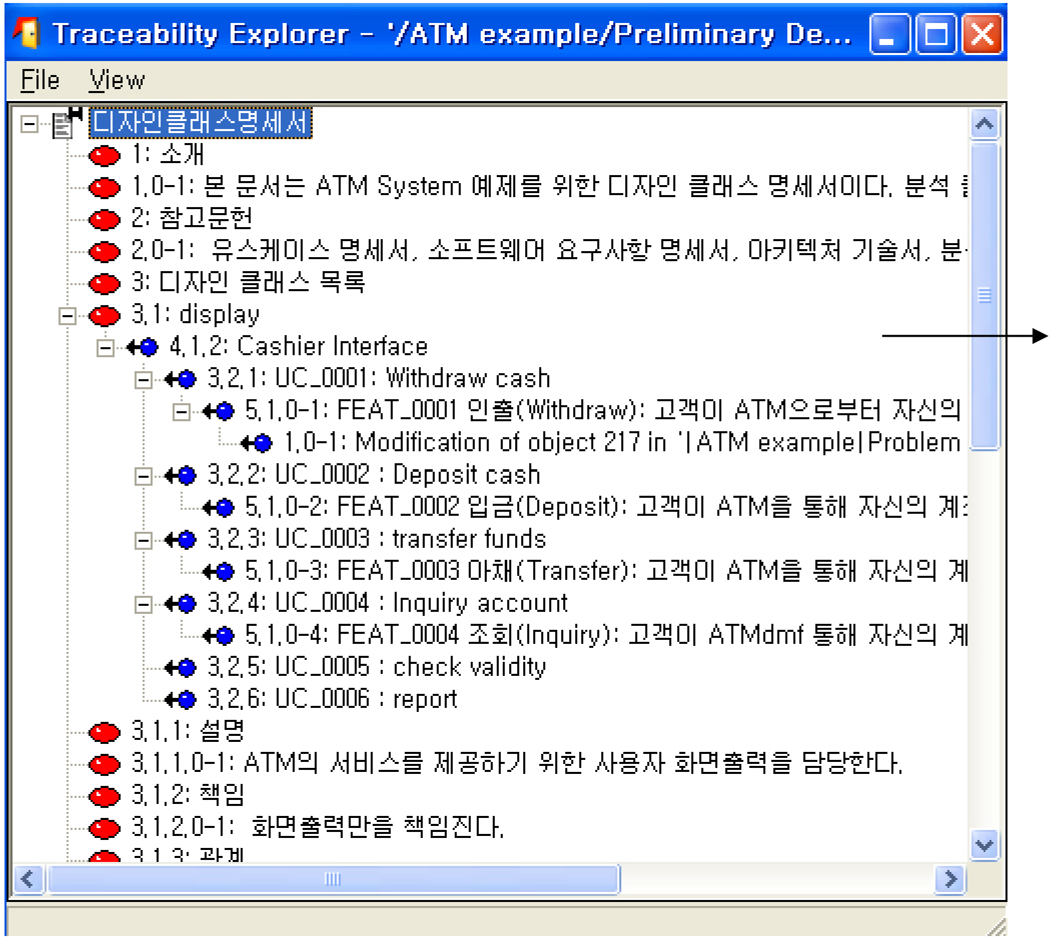
Username: shchoi
Exclusive edit mode

Depth of link

Out link 로 연결된 Use Case, Analysis Class, Design Class 들

초기요구사항 명세서의 Features

- View relationships (**Traceability Explorer**)

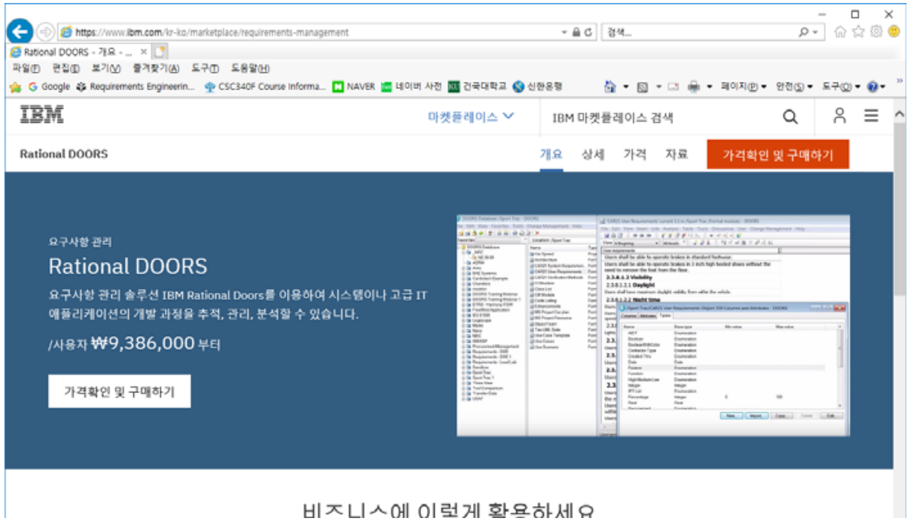


```
디자인클래스명세서
    1: 소개
    1.0-1: 본 문서는 ATM System 예제를 위한 디자인 클래스 명세서이다. 분석
    2: 참고문헌
    2.0-1:  유스케이스 명세서, 소프트웨어 요구사항 명세서, 아키텍처 기술서, 분
    3: 디자인 클래스 목록
    3.1: display
        4.1.2: Cashier Interface
            3.2.1: UC_0001: Withdraw cash
                5.1.0-1: FEAT_0001 인출(Withdraw): 고객이 ATM으로부터 자신의
                    1.0-1: Modification of object 217 in '|ATM example|Problem
            3.2.2: UC_0002 : Deposit cash
                5.1.0-2: FEAT_0002 입금(Deposit): 고객이 ATM을 통해 자신의 계
            3.2.3: UC_0003 : transfer funds
                5.1.0-3: FEAT_0003 아체(Transfer): 고객이 ATM을 통해 자신의 계
            3.2.4: UC_0004 : Inquiry account
                5.1.0-4: FEAT_0004 조회(Inquiry): 고객이 ATMdmf 통해 자신의 계
            3.2.5: UC_0005 : check validity
            3.2.6: UC_0006 : report
    3.1.1: 설명
    3.1.1.0-1: ATM의 서비스를 제공하기 위한 사용자 화면출력을 담당한다.
    3.1.2: 책임
    3.1.2.0-1:  화면출력만을 책임진다.
    3.1.3: 관계
```

"display" class 와 연관된 Use Case 6개, Feature 4개가 존재함을 확인할 수 있다

Requirements Management Tools

- **IBM Rational DOORS**


https://www.ibm.com/kr-ko/marketplace/requirements-management
Rational DOORS - 개요
파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)
IBM
마켓플레이스
IBM 마켓플레이스 검색
Rational DOORS
개요
상세
가격
자료
가격확인 및 구매하기

요구사항 관리

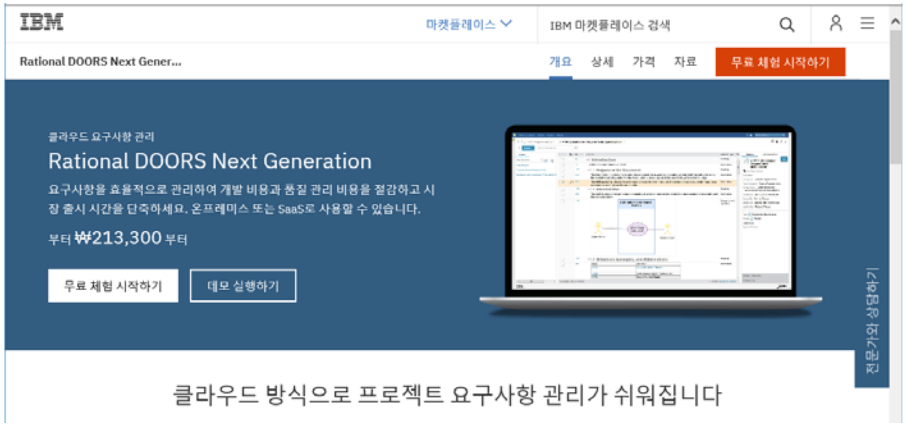
Rational DOORS

요구사항 관리 솔루션 IBM Rational Doors를 이용하여 시스템이나 고급 IT 애플리케이션의 개발 과정을 추적, 관리, 분석할 수 있습니다.

/사용자 ₩9,386,000 부터

가격확인 및 구매하기

비즈니스에 이렇게 활용하세요



클라우드 요구사항 관리

Rational DOORS Next Generation

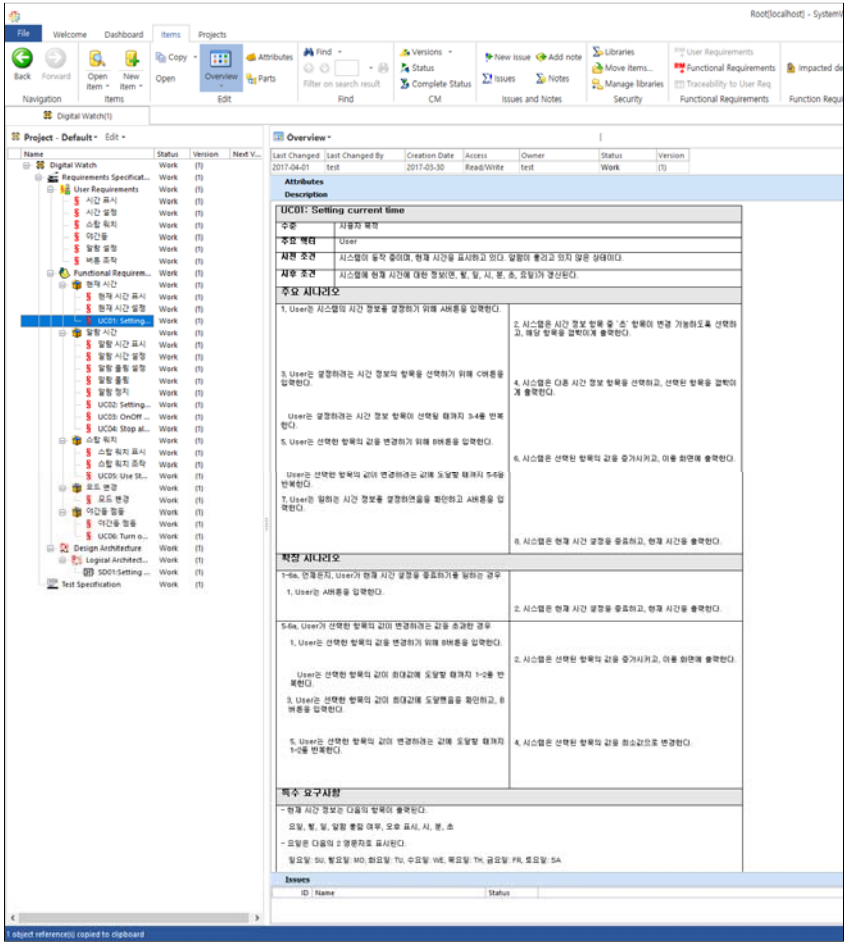
요구사항을 효율적으로 관리하여 개발 비용과 품질 관리 비용을 절감하고 시장 출시 시간을 단축하세요. 온프레미스 또는 SaaS로 사용할 수 있습니다.

부터 ₩213,300 부터

무료 체험 시작하기 | 데모 실행하기

클라우드 방식으로 프로젝트 요구사항 관리가 쉬워집니다

- **ESG PRACTICA RM+**



Last Changed	Last Changed By	Creation Date	Access	Owner	Status	Version
2017-04-01	test	2017-03-30	Read/Write	test	Work	(1)

Attributes

Description

UC01: Setting current time	
수준	사용자 목적
주요 액터	User
사전 조건	시스템이 동작 중이며, 현재 시간을 표시하고 있다. 알람이 울리고 있지 않은 상태이다.
사후 조건	시스템에 현재 시간에 대한 정보(연, 월, 일, 시, 분, 초, 요일)가 갱신된다.
주요 시나리오	
1. User는 시스템의 시간 정보를 설정하기 위해 A버튼을 입력한다.	2. 시스템은 시간 정보 항목 중 '초' 항목이 변경 가능하도록 선택하고, 해당 항목을 깜박이게 출력한다.
3. User는 설정하려는 시간 정보의 항목을 선택하기 위해 C버튼을 입력한다.	4. 시스템은 다음 시간 정보 항목을 선택하고, 선택된 항목을 깜박이게 출력한다.
User는 설정하려는 시간 정보 항목이 선택될 때까지 3-4를 반복한다. 5. User는 선택한 항목의 값을 변경하기 위해 B버튼을 입력한다.	6. 시스템은 선택된 항목의 값을 증가시키고, 이를 화면에 출력한다.
User는 선택한 항목의 값이 변경하려는 값에 도달할 때까지 5-6을 반복한다. 7. User는 원하는 시간 정보를 설정하였음을 확인하고 A버튼을 입력한다.	8. 시스템은 현재 시간 설정을 종료하고, 현재 시간을 출력한다.
확장 시나리오	
1-6a. 언제든지, User가 현재 시간 설정을 종료하기를 원하는 경우 1. User는 A버튼을 입력한다.	2. 시스템은 현재 시간 설정을 종료하고, 현재 시간을 출력한다.
5-6a. User가 선택한 항목의 값이 변경하려는 값을 초과한 경우 1. User는 선택한 항목의 값을 변경하기 위해 B버튼을 입력한다.	2. 시스템은 선택된 항목의 값을 증가시키고, 이를 화면에 출력한다.
User는 선택한 항목의 값이 최대값에 도달할 때까지 1-2를 반복한다. 3. User는 선택한 항목의 값이 최대값에 도달했음을 확인하고, B버튼을 입력한다. 5. User는 선택한 항목의 값이 변경하려는 값에 도달할 때까지 1-2를 반복한다.	4. 시스템은 선택된 항목의 값을 최소값으로 변경한다.
특수 요구사항	
- 현재 시간 정보는 다음의 항목이 출력된다. 요일, 월, 일, 알람 울림 여부, 오후 표시, 시, 분, 초 - 요일은 다음의 2 영문자로 표시된다. 일요일: SU, 월요일: MO, 화요일: TU, 수요일: WE, 목요일: TH, 금요일: FR, 토요일: SA	

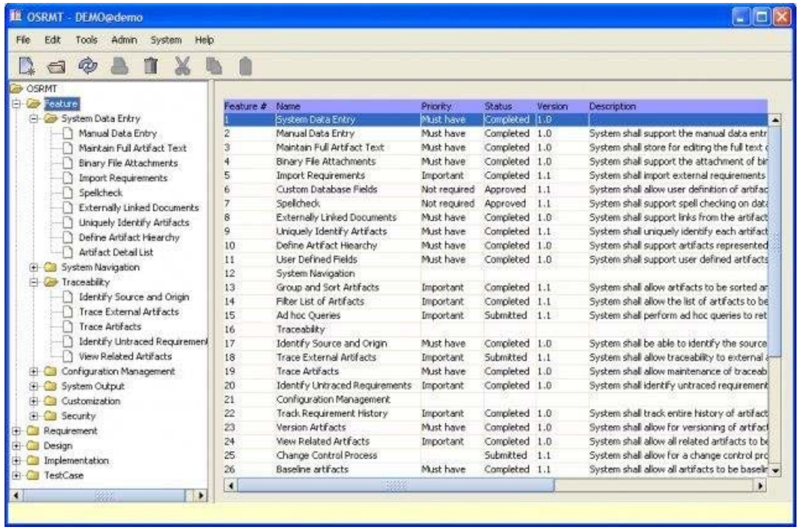
Issues

ID	Name	Status

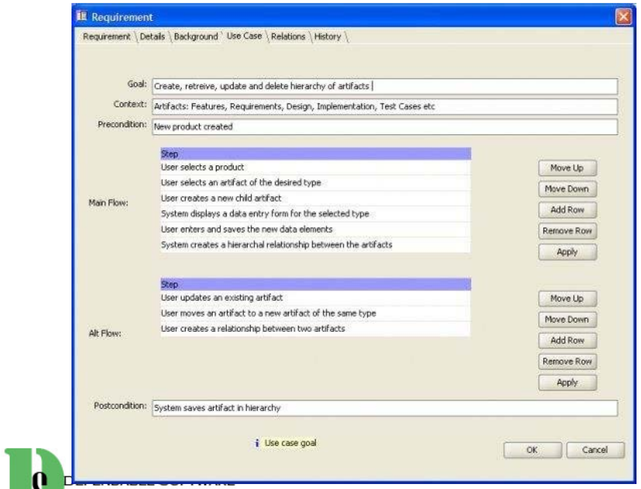
평가판 사용 가능 X

Requirements Management Tools

- **OSRMT**



Feature #	Name	Priority	Status	Version	Description
1	System Data Entry	Must have	Completed	1.0	
2	Manual Data Entry	Must have	Completed	1.0	System shall support the manual data entr
3	Maintain Full Artifact Text	Must have	Completed	1.0	System shall store for editing the full text c
4	Binary File Attachments	Must have	Completed	1.0	System shall support the attachment of bir
5	Import Requirements	Important	Completed	1.1	System shall import external requirements
6	Custom Database Fields	Not required	Approved	1.1	System shall allow user definition of artifac
7	Spellcheck	Not required	Approved	1.1	System shall support spell checking on dat
8	Externally Linked Documents	Must have	Completed	1.0	System shall support links from the artifact
9	Uniquely Identify Artifacts	Must have	Completed	1.1	System shall uniquely identify each artifact
10	Define Artifact Hiearchy	Must have	Completed	1.0	System shall support artifacts represented
11	User Defined Fields	Must have	Completed	1.0	System shall support user defined artifacts
12	System Navigation				
13	Group and Sort Artifacts	Important	Completed	1.1	System shall allow artifacts to be sorted ar
14	Filter List of Artifacts	Important	Completed	1.1	System shall allow the list of artifacts to be
15	Ad hoc Queries	Important	Submitted	1.1	System shall perform ad hoc queries to ret
16	Traceability				
17	Identify Source and Origin	Must have	Completed	1.0	System shall be able to identify the source
18	Trace External Artifacts	Important	Submitted	1.1	System shall allow traceability to external a
19	Trace Artifacts	Must have	Completed	1.0	System shall allow maintenance of traceab
20	Identify Untraced Requirements	Important	Completed	1.0	System shall identify untraced requirement
21	Configuration Management				
22	Track Requirement History	Important	Completed	1.0	System shall track entire history of artifact
23	Version Artifacts	Must have	Completed	1.0	System shall allow for versioning of artifact
24	View Related Artifacts	Important	Completed	1.0	System shall allow all related artifacts to be
25	Change Control Process		Submitted	1.1	System shall allow for a change control prc
26	Baseline artifacts	Must have	Completed	1.1	System shall allow all artifacts to be baselir



Goal: Create, retreive, update and delete hierarchy of artifacts

Context: Artifacts: Features, Requirements, Design, Implementation, Test Cases etc

Precondition: New product created

Main Flow:

Step
User selects a product
User selects an artifact of the desired type
User creates a new child artifact
System displays a data entry form for the selected type
User enters and saves the new data elements
System creates a hierarchal relationship between the artifacts

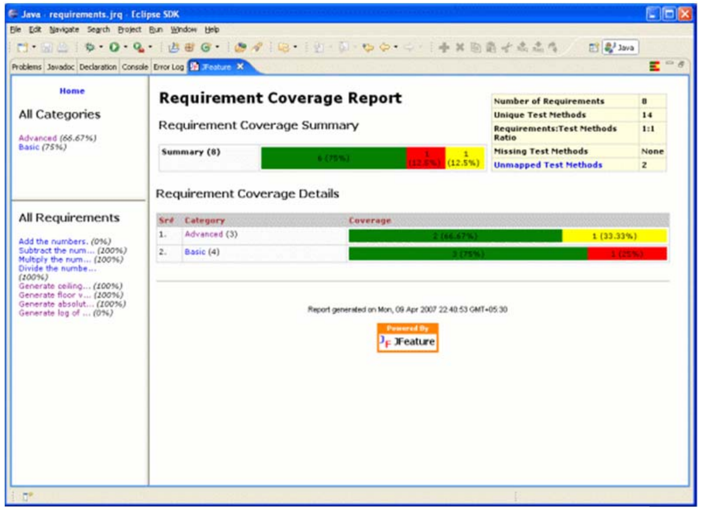
Alt Flow:

Step
User updates an existing artifact
User moves an artifact to a new artifact of the same type
User creates a relationship between two artifacts

Postcondition: System saves artifact in hierarchy

Use case goal

- **JFeatures**



All Categories

Advanced (66.67%)
Basic (75%)

All Requirements

Add the numbers. (0%)
Subtract the num... (100%)
Multiply the num... (100%)
Divide the numbe... (100%)
Generate ceiling... (100%)
Generate floor v... (100%)
Generate absolut... (100%)
Generate log of ... (0%)

Requirement Coverage Report

Requirement Coverage Summary

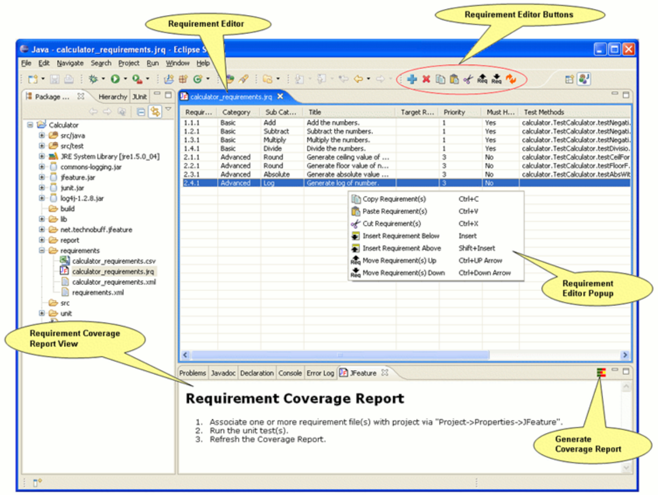
Summary (8)	6 (75%)	1 (12.5%)	1 (12.5%)

Number of Requirements	8
Unique Test Methods	14
Requirements:Test Methods Ratio	1:1
Missing Test Methods	None
Unmapped Test Methods	2

Requirement Coverage Details

Sr#	Category	Coverage	
1.	Advanced (3)	2 (66.67%)	1 (33.33%)
2.	Basic (4)	3 (75%)	1 (25%)

Report generated on Mon, 09 Apr 2007 22:40:53 GMT+05:30


Requirement Editor
Requirement Editor Buttons
Java - calculator_requirements.jrq - Eclipse S
File
Edit
Navigate
Search
Project
Run
Window
Help
Package ...
Hierarchy
JUnit
calculator_requirements.jrq
Calculator
src/java
src/test
JRE System Library [jre1.5.0_04]
commons-logging.jar
jfeature.jar
junit.jar
log4j-1.2.8.jar
build
lib
net.technobuff.jfeature
report
requirements
calculator_requirements.csv
calculator_requirements.jrq
calculator_requirements.xml
requirements.xml
src
unit
Copy Requirement(s) Ctrl+C
Paste Requirement(s) Ctrl+V
Cut Requirement(s) Ctrl+X
Insert Requirement Below Insert
Insert Requirement Above Shift+Insert
Move Requirement(s) Up Ctrl+UP Arrow
Move Requirement(s) Down Ctrl+Down Arrow
Requirement Editor Popup
Requirement Coverage Report View
Problems
Javadoc
Declaration
Console
Error Log
JFeature
Generate Coverage Report

Requir...	Category	Sub Cat...	Title	Target R...	Priority	Must H...	Test Methods
1.1.1	Basic	Add	Add the numbers.		1	Yes	calculator.TestCalculator.testNegati.
1.2.1	Basic	Subtract	Subtract the numbers.		1	Yes	calculator.TestCalculator.testNegati.
1.3.1	Basic	Multiply	Multiply the numbers.		1	Yes	calculator.TestCalculator.testNegati.
1.4.1	Basic	Divide	Divide the numbers.		1	Yes	calculator.TestCalculator.testDivisio.
2.1.1	Advanced	Round	Generate ceiling value of ...		3	No	calculator.TestCalculator.testCeilFor
2.2.1	Advanced	Round	Generate floor value of n...		3	No	calculator.TestCalculator.testFloorF.
2.3.1	Advanced	Absolute	Generate absolute value ...		3	No	calculator.TestCalculator.testAbsWit
2.4.1	Advanced	Log	Generate log of number.		3	No	

Requirement Coverage Report

1. Associate one or more requirement file(s) with project via "Project->Properties->JFeature".
2. Run the unit test(s).
3. Refresh the Coverage Report.