유스 케이스를 통한 요구사항 관리 적용

Roger Oberg, Leslee Probasco 및 Maria Ericsson

Rational Software 백서

TP 505(버전 1.4)

Rational®
the software development company

목차

사용자가 요구사항 관리에 대해 잘 모르거나 다소 익숙한 정도이고 요구사항 프로세스 개선에 관심이 있는 경우, 이 문서는 자체 접근 방식을 개발할 프레임워크를 제공합니다.

> **유스 케이스 및 소프트웨어 요구사항 스펙(SRS)**
>
> 독자에게 요구사항 관리 활동이 보다 의미 있도록 작성자는 Rational Software Unified Process 의 특정 문서 유형과 기타 요구사항 관리 아티팩트 및 업계 표준 UML(Unified Modeling Language)을 선택했는데, 둘 다 유스 케이스 기반 소프트웨어 엔지니어링 프로세스를 권장합니다.
>
> 따라서 여기서는 소프트웨어 요구사항을 지정하기 위한 유스 케이스 기반 접근 방식을 설명합니다. 유스 케이스 모델과 유스 케이스 이외에, 또는 이를 대신하여 이런 활동을 보다 일반적인 소프트웨어 요구사항 스펙(예: IEEE 표준)과 함께 사용할 수도 있습니다.

프로세스 추이별 소프트웨어 및 시스템 개발

대부분의 소프트웨어 및 시스템 개발 팀의 경우, 과거의 보다 자유 분방했던 시절과 비교할 때 1990 년대는 프로세스 위주였습니다. 효과적인 소프트웨어 개발 프로세스를 측정하고 인증하기 위한 표준이 도입되어 널리 보급되었습니다. 소프트웨어 개발 프로세스에 관한 많은 서적과 기사, 비즈니스 프로세스 모델링 및 리엔지니어링 관련 자료가 공개되었습니다. 소프트웨어 도구 수가 늘어나면서 효과적인 소프트웨어 개발 프로세스를 정의하고 적용하는 데 도움을 주었습니다. 지난 10 년간 세계 경제의 소프트웨어에 대한 의존도가 심화되면서 개발 프로세스를 사용하고 시스템의 품질이 개선되었습니다.

그렇다면, 오늘날 소프트웨어 프로젝트의 높은 실패율을 어떻게 설명할 수 있습니까? 대부분은 아니더라도 많은 소프트웨어 프로젝트가 여전히 지연, 예산 초과 및 품질 문제점에 시달리는 이유는 무엇입니까? 비즈니스, 국가 경제 및 일일 활동의 시스템 의존도가 늘어나는 상황에서 빌드하는 시스템의 품질을 어떻게 개선할 수 있습니까?

항상 그렇듯이 해답은 직업에 적용된 사람, 도구 및 프로세스에 있습니다. 소프트웨어 개발의 지속적인 문제점에 대한 솔루션으로 요구사항 관리가 제안되는 경우가 많지만 이런 원칙 사례를 개선하는 데는 상대적으로 거의 중점을 두지 않았습니다.

이 문서에서는 효과적인 요구사항 관리 프로세스의 요소를 제공하고 성공적 구현을 방해하는 몇 가지 장애물을 강조합니다.

요구사항 관리는 소프트웨어 전용 프로젝트와 소프트웨어가 최종 결과의 일부에 지나지 않거나 전혀 포함되지 않는 프로젝트에 동등하게 적용됩니다. 편의상 지금부터 이 문서에서 "시스템"이라는 용어는 이 중 일부 또는 전부를 의미합니다. 그러나 이 문서는 단독으로 또는 하드웨어와 공동으로 요구사항 관리를 복잡하게 하는 소프트웨어 개발의 추상적 특성에 주로 초점을 맞추고 있습니다.

요구사항을 관리하는 이유

간단히 말하면 요구사항을 관리하는 시스템 개발 팀은 프로젝트가 성공을 거두길 바라므로 요구사항을 관리합니다. 프로젝트의 요구사항을 충족시키는 것이 성공을 정의합니다. 요구사항을 관리하지 못하면 이런 목표를 충족시킬 가능성이 줄어듭니다.

최근의 증거가 이를 뒷받침합니다.

- Standish Group 의 1994 년과 1997 년의 CHAOS 보고서는 프로젝트 실패의 가장 중요한 요인은 요구사항과 관련이 있다고 밝혔습니다. [1]

- 1997 년 12 월에 Computer Industry Daily 는 미국과 영국에 있는 500 명의 IT 관리자를 대상으로 한 Sequent Computer Systems, Inc 에 관한 연구 보고서에서 응답자의 76%가 자신의 직무 수행 중 완벽한 프로젝트 실패를 경험한 바 있다고 하였습니다. 프로젝트 실패의 원인으로 가장 빈번히 언급된 것은 "사용자 요구사항의 변화"였습니다. [2]

실패 방지는 요구사항을 관리하는 충분한 동기 부여 요인이어야 합니다. 프로젝트 성공 가능성과 요구사항 관리의 기타 이점을 증대시키는 것도 마찬가지로 동기 부여를 할 수 있습니다. 더 나아가 Standish Group 의 CHAOS 보고서는 요구사항 관리 역시 프로젝트 성공과 가장 관련이 깊은 요인이라고 밝혔습니다.

요구사항의 개념

요구사항 관리를 이해하는 첫 번째 단계는 공통 용어에 관한 합의에 도달하는 것입니다. Rational 은 요구사항을 "(빌드 중인) 시스템이 준수해야 하는 조건 또는 기능"으로 정의합니다. IEEE(Institute of Electronics and Electrical Engineers)는 유사한 정의를 사용합니다.

유명한 요구사항 엔지니어링 작성자인 Merlin Dorfman 과 Richard H. Thayer 는 소프트웨어에 고유한(반드시 여기에 국한되지는 않음) 호환 가능하면서 보다 세부화된 정의를 제공합니다.

"다음과 같이 소프트웨어 요구사항을 정의할 수 있습니다.

- 문제점을 해결하거나 목표를 달성하기 위해 사용자가 필요로 하는 소프트웨어 기능.
- 계약, 스펙, 표준 또는 기타 공식적으로 부과되는 문서를 충족시키기 위해 시스템 또는 시스템 컴포넌트가 충족하거나 보유해야 하는 소프트웨어 기능." [3]

요구사항 관리의 개념

요구사항은 빌드 중인 시스템이 준수해야 하는 사항이고 일부 요구사항 세트를 준수하느냐가 프로젝트의 성공이나 실패를 정의하므로 요구사항의 개념을 파악하여 이를 기록하고 구성하며, 요구사항이 변경될 경우에는 이를 추적하는 것이 적절합니다.

요구사항 관리를 다른 방식으로 설명하면 다음과 같습니다.

- 시스템의 소프트웨어 요구사항을 도출, 구성 및 문서화하기 위한 조직적 접근 방식
- 시스템의 변경되는 요구사항에 관해 고객과 프로젝트 팀 간에 계약을 설정하고 유지보수하는 프로세스.

이 정의는 Dorfman 과 Thayer 의 정의와 유사하며, IEEE 의 "소프트웨어 요구사항 엔지니어링"과도 유사합니다. 요구사항 엔지니어링에는 소프트웨어 요구사항 도출, 분석, 구체화, 검증 및 관리가 포함되며 "소프트웨어 요구사항 관리"는 이런 모든 관련 활동을 계획하고 제어합니다. [4] 이런 모든 활동은 여기 제공되어 있고 Rational Software 에서 교육하는 요구사항 관리에 대한 정의에 통합됩니다. 주요 차이점은 "엔지니어링" 대신 "관리"라는 단어를 선택한 것에 있습니다. 관리는 관련된 모든 활동에 대한 보다 적절한 설명이며, 이해 당사자(stakeholder)와 프로젝트 팀 간의 계약을 유지보수하기 위해 변경사항 추적의 중요성을 정확히 강조합니다.

팀이 이해 당사자(stakeholder) 요청을 도출하거나 발견하는 데 사용하는 활동 세트로 정의된 "도출"이란 용어에 익숙하지 않은 사용자의 경우, 요청 이면의 실제 요구를 판별하고 해당 요구를 충족시키기 위해 시스템에 배치할 수 있는 적절한 요구사항 세트에 도달하십시오.

요구사항 관리의 문제점

그렇다면, 시스템이 시스템에 대해 설정된 기대에 부응하도록 하기 위한 프로세스의 경우 어려운 점은 무엇입니까? 실제 프로젝트에서 실행할 때 어려움이 드러납니다. 그림 1 은 개발자, 관리자 및 품질 보증 담당자를 대상으로 한 1996 년의 조사 결과를 표시합니다. 가장 빈번히 언급된 요구사항 관련 문제점을 경험한 응답자의 백분율을 표시합니다.

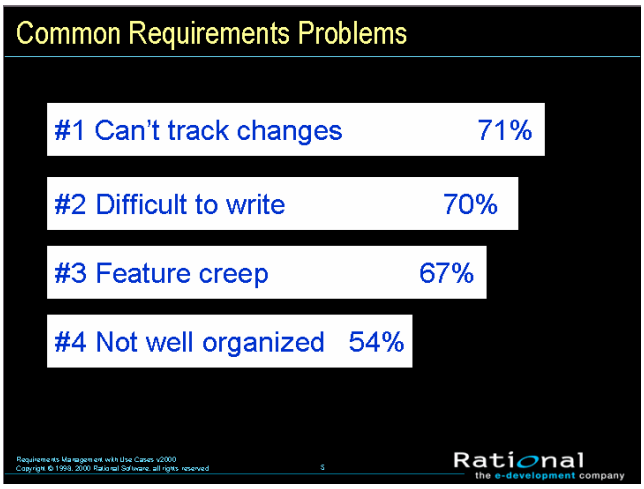

그림 1—일반적인 요구사항 문제점

보다 포괄적인 문제점 목록은 다음과 같습니다.

- 요구사항은 항상 명백한 것은 아니며, 소스가 다양합니다.
- 요구사항은 항상 단어로 명백하게 표현하는 것이 쉽지 않습니다.
- 세부사항의 여러 레벨에서 서로 다른 많은 유형의 요구사항이 있습니다.
- 제어되지 않을 경우 여러 가지의 요구사항이 관리 불가능하게 될 수 있습니다.
- 요구사항은 서로 관련되어 있고 프로세스의 다른 인도물과도 다양한 방법으로 관련되어 있습니다.
- 요구사항은 고유한 특성 또는 특성 값을 가지고 있습니다. 예를 들어, 요구사항은 똑같이 중요하지도 않고 충족시키는 것이 똑같이 쉽지도 않습니다.
- 이해 관계자와 책임자가 많습니다. 이는 다기능 그룹에서 요구사항을 관리해야 함을 의미합니다.
- 요구사항은 변경됩니다.
- 요구사항은 시간에 민감할 수 있습니다.

이런 문제점이 부적절한 요구사항 관리 및 프로세스 스킬과 결합되고 사용하기 쉬운 도구가 부족한 경우, 많은 팀은 요구사항을 제대로 관리할 수 없다는 절망감에 빠지게 됩니다. Rational Software 는 팀에 요구사항 관리 스킬 및 프로세스를 교육할 수 있는 전문 지식을 개발했습니다. 또한 Rational RequisitePro 는 효과적인 요구사항 관리를 자동화하는 액세스 가능한 도구입니다.

요구사항 관리 스킬

앞서 언급한 문제점을 해결하기 위해 Rational 은 핵심 스킬 개발을 권장합니다. 이런 스킬은 아래에 표시되어 있는데, 여기서는 순서대로 나타나지만 효과적인 요구사항 관리 프로세스에서는 다양한 순서로 연속적으로 적용됩니다. 여기서는 새 프로젝트의 첫 번째 반복에 적용할 순서대로 표시되어 있습니다.

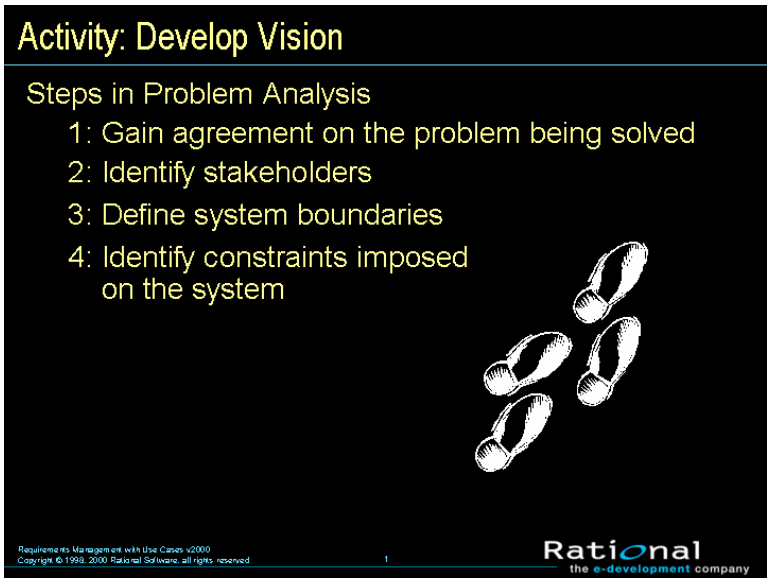

그림 2—문제점 분석 단계

핵심 스킬 1: 문제점 분석

문제점 분석은 비즈니스 문제점, 대상 초기 이해 당사자(stakeholder)의 요구를 이해하고 상위 레벨 솔루션을 제안하기 위해 수행됩니다. 이러한 추론 및 분석 조치는 "문제점 이면의 문제점"을 찾습니다.

문제점을 분석하는 동안 실제 문제점의 설명에 대한 합의에 도달하고 이해 당사자(stakeholder)를 식별합니다. 초기 솔루션 경계와 제한조건은 기술적 관점과 비즈니스 관점 둘 다에서 정의됩니다. 해당되는 경우, 프로젝트의 비즈니스 사례는 시스템에서 예상되는 투자 수익률을 분석합니다.

핵심 스킬 2: 이해 당사자의 요구 이해

요구사항은 소스가 다양합니다. 요구사항의 소스는 프로젝트 결과물에 관심이 있는 어떤 사람이라도 될 수 있습니다. 고객, 파트너, 일반 사용자 및 도메인 전문가는 요구사항의 일부 소스입니다. 관리, 프로젝트 팀 구성원, 비즈니스 정책 및 규제 기관이 소스일 수도 있습니다.

소스가 누구인지 판별하는 방법, 해당 소스에 액세스하는 방법 및 해당 소스로부터 정보를 도출하는 방법을 아는 것이 중요합니다. 이 정보의 1 차 소스의 역할을 하는 개인을 프로젝트에서는 "이해 당사자(stakeholder)"라고 합니다.

회사에서 내부적으로 사용할 정보 시스템을 개발 중인 경우, 개발 팀에 일반 사용자 경험이 있는 사람과 비즈니스 도메인 전문가를 포함시킬 수 있습니다. 시스템 레벨이 아닌 비즈니스 모델 레벨에서 논의를 시작하는

경우가 아주 많습니다. 시장에 판매할 제품을 개발 중인 경우, 해당 시장에서 고객의 요구사항을 제대로 이해할 수 있도록 마켓팅 인원을 광범위하게 기용할 수 있습니다.

요구사항 도출 기법으로는 인터뷰, 브레인스토밍, 개념 프로토타입 생성, 질문지 및 경쟁적 분석이 있습니다. 요구사항 도출 결과는 텍스트와 그래픽으로 설명되고 서로 상대적으로 우선순위가 지정된 요청 또는 요구 목록입니다.

핵심 스킬 3: 시스템 정의

시스템을 정의하는 것은 이해 당사자(stakeholder) 요구 이해를 빌드하는 시스템의 의미있는 설명으로 변환하여 구성하는 것을 의미합니다. 시스템 정의 초기에는 요구사항 구성, 문서 형식, 언어 형식성, 요구사항의 정도, 요청 우선순위와 예상 노력, 기술 및 관리 위험성, 범위를 결정합니다. 이 활동의 일부로, 가장 중요한 이해 당사자(stakeholder) 요청에 직접 관련되는 초기 프로토타입 및 디자인 모델이 포함될 수 있습니다.

문서라는 단어를 일반적으로 사용할 때 내재되어 있는 인식된 제한사항을 피하기 위해 "문서" 대신 "설명"이란 단어를 사용합니다. 설명은 기록된 문서, 전자 파일, 그림일 수 있으며, 또는 시스템 자체는 제외하고 시스템 요구사항을 커뮤니케이션하기 위한 기타 표시일 수 있습니다.

시스템 정의 결과는 자연어 및 그래픽으로 된 시스템 설명입니다. 일부 제안된 설명 형식은 이후 섹션에 제공되어 있습니다.

원칙 55: 보다 정규 모델 이전에 자연어 작성

정규 모델을 먼저 작성하는 경우, 솔루션 시스템 대신 모델을 설명하는 자연어를 작성하는 경향이 있습니다. 다음 예제를 고려하십시오.

> 장거리 전화를 하려면 사용자가 수화기를 들어야 합니다. 시스템은 발신음으로 응답합니다. 사용자는 "9"번을 눌러야 합니다. 시스템은 변별적 발신음으로 응답합니다.…
>
> 시스템이 네 가지 상태(대기, 발신음, 변별적 발신음 및 연결)로 구성됩니다. 대기 상태에서 발신음 상태가 되려면 수화기를 드십시오. 발신음 상태에서 변별적 발신음 상태가 되려면 "9"번을 누르십시오.

후자의 예제에서 텍스트는 독자에게 전혀 도움이 되지 않습니다.

—Alan M. Davis, 201 Principles of Software Development, 1995

핵심 스킬 4: 시스템 범위 관리

프로젝트에 할당된 요구사항 세트가 프로젝트 범위를 정의합니다. 프로젝트 범위를 주어진 자원(시간, 인력 및 자금)에 맞게 관리하는 것이 성공적인 프로젝트 관리의 핵심입니다. 범위 관리는 반복적 또는 점진적 개발을 필요로 하는 연속적 활동으로, 프로젝트 범위를 보다 관리하기 쉬운 작은 부분으로 분할합니다.

우선순위, 노력 및 위험성과 같은 요구사항 속성을 요구사항 포함 협상을 위한 기초로 사용하는 것은 특히 범위 관리에 유용한 기법입니다. 요구사항 자체보다 속성에 초점을 맞추면 협상 시 논쟁의 여지가 있는 부분이 완화됩니다.

팀 리더에게 협상 스킬을 훈련하고 프로젝트에서 고객측 뿐 아니라 조직 내의 챔피언을 가지면 좋습니다. 제품 및 프로젝트 챔피언은 사용 가능한 자원 이상의 범위 변경을 거부하거나 추가 범위에 맞게 자원을 확대할 수 있는 조직적 능력이 있어야 합니다.

핵심 스킬 5: 시스템 정의 정제

합의된 상위 레벨 시스템 정의와 충분히 인지된 초기 범위를 사용하여 보다 정제된 시스템 정의에 자원을 투자하는 것이 가능하면서도 경제적입니다. 시스템 정의 정제에는 두 가지 핵심 고려사항이 포함되는데, 상위 레벨 시스템 정의에 대한 보다 자세한 설명을 개발하는 것과 시스템이 이해 당사자(stakeholder) 요구에 부응하고 설명된 대로 작동하는지 확인하는 것입니다.

설명은 프로젝트 팀에 중요한 참조 자료인 경우가 많습니다. 독자를 염두에 둘 때 최상의 설명이 제공됩니다. 흔한 실수는 특히, 독자가 합의를 얻는 데 필요한 비판적 사고를 할 수 없거나 꺼려하는 경우에 복잡한 정의를 사용하여 빌드할 복잡한 내용을 표시하는 것입니다. 따라서 프로젝트 팀 내부 사람과 외부 사람 모두에게 시스템의 목적을 설명하는 것이 어려워집니다. 대신, 다른 독자에게는 다른 유형의 설명을 생성해야 할 수도 있습니다. 이 문서에는 제안된 형식의 세부 자연어, 정규 텍스트 및 그래픽 설명이 포함되어 있습니다. 설명 형식이 설정되면, 프로젝트 라이프사이클 전반에서 정제가 계속됩니다.

핵심 스킬 6: 요구사항 변경 관리

아무리 신중히 요구사항을 정의한다 하더라도 요구사항은 변경됩니다. 사실, 일부 요구사항 변경은 바람직합니다. 이는 팀에서 이해 당사자(stakeholder)를 참여시키고 있음을 의미합니다. 변경되는 요구사항을 수용하는 것은 팀의 이해 당사자에 대한 반응도와 운영상의 유연성(프로젝트 성공에 기여하는 속성)을 측정하는 척도입니다. 변경이 적이 아니라 변경을 관리하지 않는 것이 적입니다.

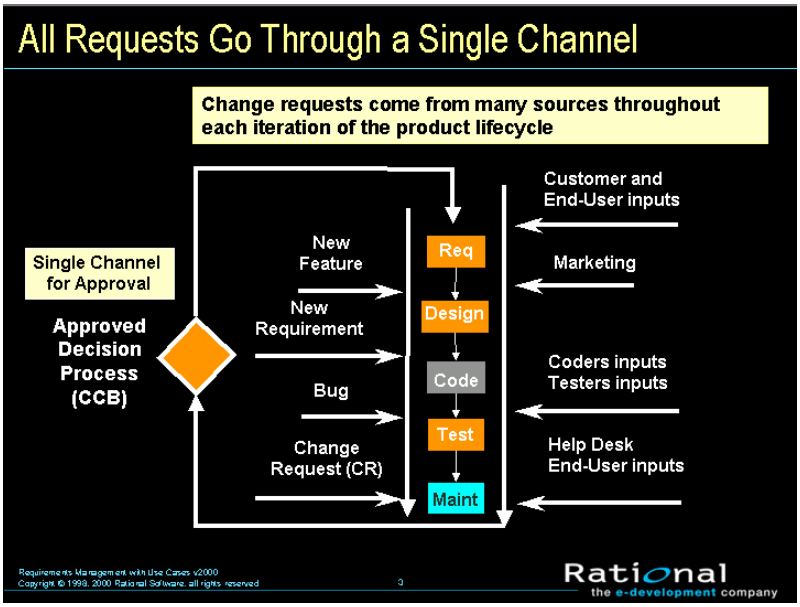

그림 3—변경 관리 프로세스

변경된 요구사항은 특정 기능을 구현하는 데 다소 시간이 소요되어야 함을 의미하며 하나의 요구사항에 대한 변경은 다른 요구사항에도 영향을 줄 수 있습니다. 요구사항 관리에는 기준선 설정, 각 요구사항의 히스토리 추적, 추적에 중요한 종속성 판별, 관련 항목 간에 추적 가능한 관계 설정 및 버전 제어 유지보수와 같은 활동이 포함됩니다. 그림 3 이 설명하는 바와 같이 변경 제어 또는 승인 프로세스를 설정하는 것도 중요하며, 지정된 팀 구성원은 제안된 모든 변경사항을 검토해야 합니다. 변경 제어를 수행하는 이 단일 채널을 변경 제어 위원회(CCB)라고 하는 경우도 있습니다.

중요한 요구사항 개념

프로젝트에 요구사항 관리 스킬을 적용하려면 프로젝트에 참여한 모든 사람이 특정 요구사항 관리 개념을 이해하는 것이 유용합니다. 여기에는 다음이 포함됩니다.

- 요구사항 유형
- 다기능 팀
- 추적성
- 다차원 속성
- 변경 히스토리

요구사항 유형

시스템이 대형화되고 복잡해질수록 요구사항 유형이 더 많이 나타납니다. 요구사항 유형은 단지 요구사항 클래스입니다. 팀은 요구사항 유형을 식별하여 다양한 요구사항을 의미있고 보다 쉽게 관리할 수 있는 그룹으로 구성할 수 있습니다. 서로 다른 요구사항 유형을 한 프로젝트에 설정하면 팀 구성원이 변경 요청을 분류하고 보다 분명하게 커뮤니케이션하는 데 도움이 됩니다.

일반적으로, 하나의 요구사항 유형을 다른 유형으로 나누거나 분해할 수 있습니다. 비즈니스 규칙 및 비전 설명은 팀이 사용자 요구, 기능 및 제품 요구사항 유형을 도출하는 상위 레벨 요구사항 유형일 수 있습니다. 유스 케이스 및 기타 모델링 양식은 소프트웨어 요구사항으로 분해하고 분석 및 디자인 모델에 표시할 수 있는 디자인 요구사항을 주도합니다. 테스트 요구사항은 소프트웨어 요구사항에서 파생되며 특정 테스트 프로시저로 분해됩니다. 주어진 프로젝트에 수백, 수천 또는 수만 개의 요구사항 인스턴스가 있는 경우, 요구사항을 유형으로 분류하면 프로젝트를 보다 쉽게 관리할 수 있습니다.

다기능 팀

단일 비즈니스 그룹에서 관리할 수 있는 다른 프로세스(예: 테스트 또는 응용프로그램 모델링)와 달리 요구사항 관리에는 전문 지식을 개발 프로세스에 컨트리뷰션할 수 있는 모든 사람이 포함되어야 합니다. 고객 및 비즈니스 기대를 표시하는 사람이 포함되어야 합니다. 개발 관리자, 제품 관리자, 분석가, 시스템 엔지니어 및 고객도 참여해야 합니다. 요구사항 팀에는 시스템 솔루션을 작성하는 사람(엔지니어, 설계자, 디자이너, 프로그래머, 전문 기술 작성자 및 기타 전문 기술 컨트리뷰터)도 포함되어야 합니다. 테스터 및 기타 품질 보증(QA) 담당자도 중요한 팀 구성원으로 간주해야 합니다.

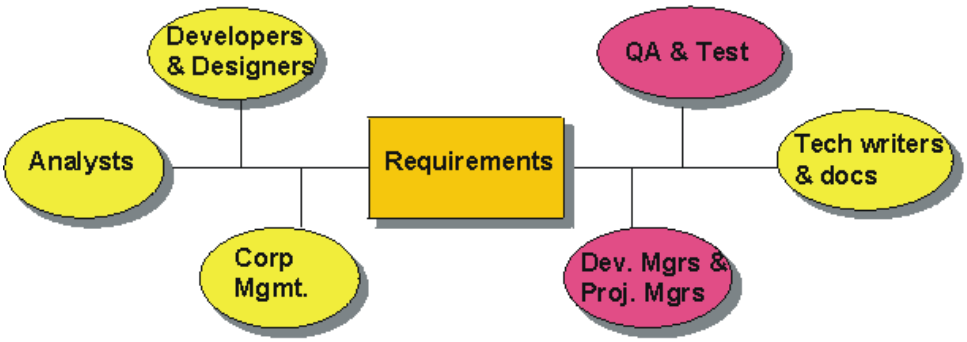

그림 4—다기능 요구사항 팀

대개 요구사항 유형을 작성하고 유지보수하는 책임을 기능 영역별로 할당할 수 있으며, 이를 통해 더 나아가 대형 프로젝트를 보다 잘 관리할 수 있습니다. 요구사항 관리의 다기능 네이처는 원칙의 보다 도전적인 측면 중 하나입니다.

추적성

요구사항 유형의 설명에 나타난 대로, 요구사항의 단일 표현식은 독립적이지 않습니다. 이해 당사자(stakeholder) 요청은 해당 요청을 충족시키기 위해 제안된 제품 기능과 관련됩니다. 제품 기능은 기능적 동작과 비기능적 동작 관점에서 기능을 지정하는 개별 요구사항과 관련됩니다. 테스트 케이스는 테스트 케이스에서 확인하고 유효성을 검증하는 요구사항과 관련됩니다. 요구사항은 다른 요구사항에 종속될 수 있으며, 상호 배타적일 수 있습니다. 변경에 따른 영향을 판별하고 시스템이 기대에 부응한다고 확신하려면, 팀은 추적성 관계를 이해하고 문서화하며 유지보수해야 합니다. 추적성은 요구사항 관리에서 구현할 가장 어려운 개념 중 하나이며, 변경을 수용하려면 필수적입니다. 명확한 요구사항 유형을 설정하고 다기능 참여를 통합하면 추적성을 보다 쉽게 구현하고 유지보수할 수 있습니다. 서로 다른 요구사항 추적성 전략에 관한 자세한 정보는 "유스 케이스로 요구사항을 관리하기 위한 추적성 전략" 백서를 참조하십시오.[5]

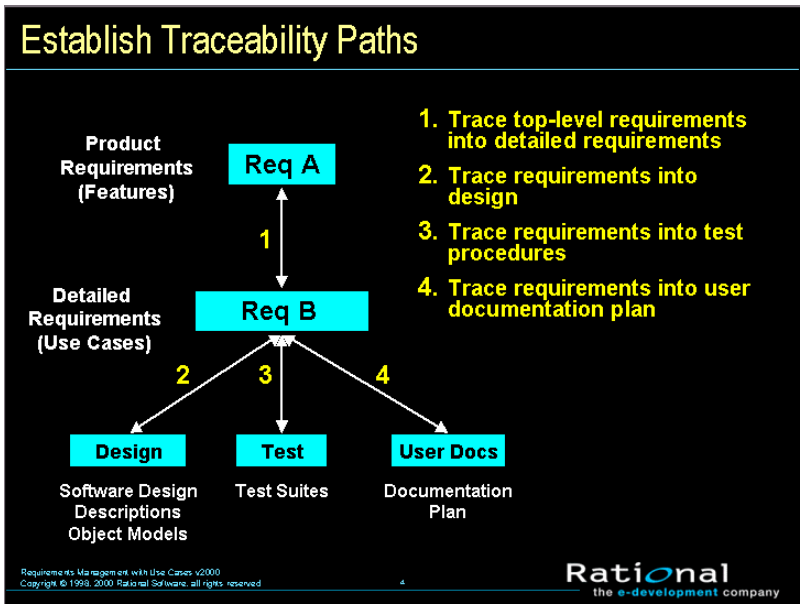

그림 5—요구사항 추적성

다차원 속성

요구사항 유형마다 속성이 있으며 개별 요구사항마다 속성 값이 다릅니다. 예를 들어, 요구사항에는 우선순위가 지정되고 소스 및 근거로 식별되며, 기능 영역 내 특정 하위 팀에 지정되고, 난이도가 지정되거나 시스템의 특정 반복과 연관될 수 있습니다. 이를 설명하기 위해 그림 6 은 학습 프로젝트의 기능 요구사항 유형 속성을 Rational RequisitePro 요구사항 관리 도구에 표시합니다. 화면 제목이 암시하듯 요구사항 유형과 각 유형의 속성이 전체 프로젝트에 대해 정의되어 팀 전체에서 사용 일관성이 유지됩니다.

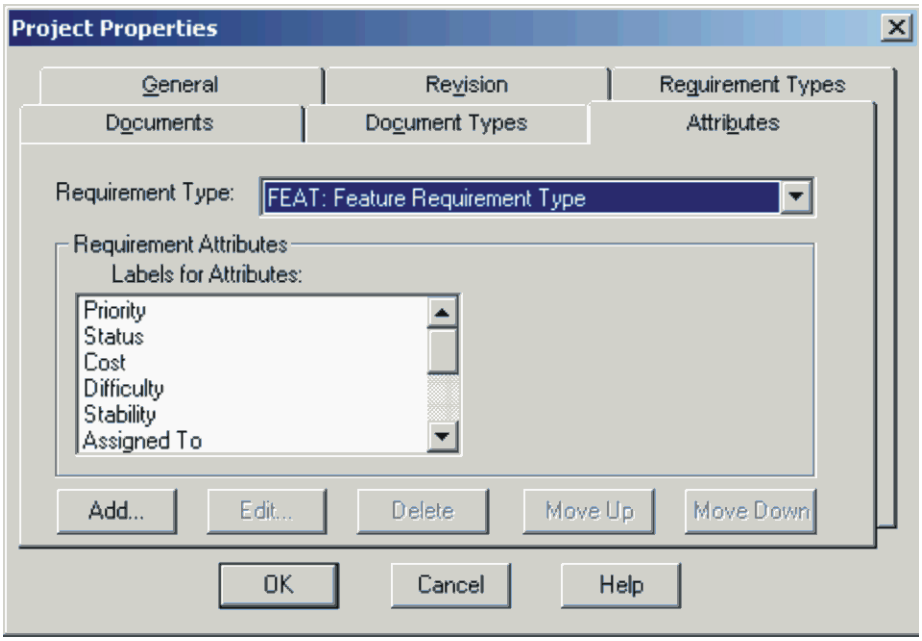

그림 6—기능의 요구사항 속성 정의

그림 7 에서는 기능 요구사항의 인스턴스는 RequisitePro 의 특정 프로젝트에 대해 표시됩니다. 각 요구사항의 전체 텍스트를 표시하지 않더라도 속성 값을 통해 각 요구사항에 대해 많이 배울 수 있음에 유의하십시오. 이 경우, 우선순위와 난이도(분명 다른 팀 구성원이 지정)는 팀이 사용 가능한 자원과 시간에 프로젝트 범위 지정을 시작하도록 돕고 이해 당사자(stakeholder) 우선순위와 난이도 속성 값에 반영된 대략적인 노력 추정치를 둘 다 고려합니다. 요구사항의 세부 유형에서 우선순위와 노력 속성은 범위를 추가로 정제하기 위한 보다 특정한 값(예: 예상 시간, 코드 행 등)을 가질 수 있습니다. 서로 다른 요구사항 유형(각각 자체 속성을 가짐)이 복합된 요구사항의 다차원 측면은 많은 요구사항을 구성하고 프로젝트의 전체 범위를 관리하는 데 필수적입니다.

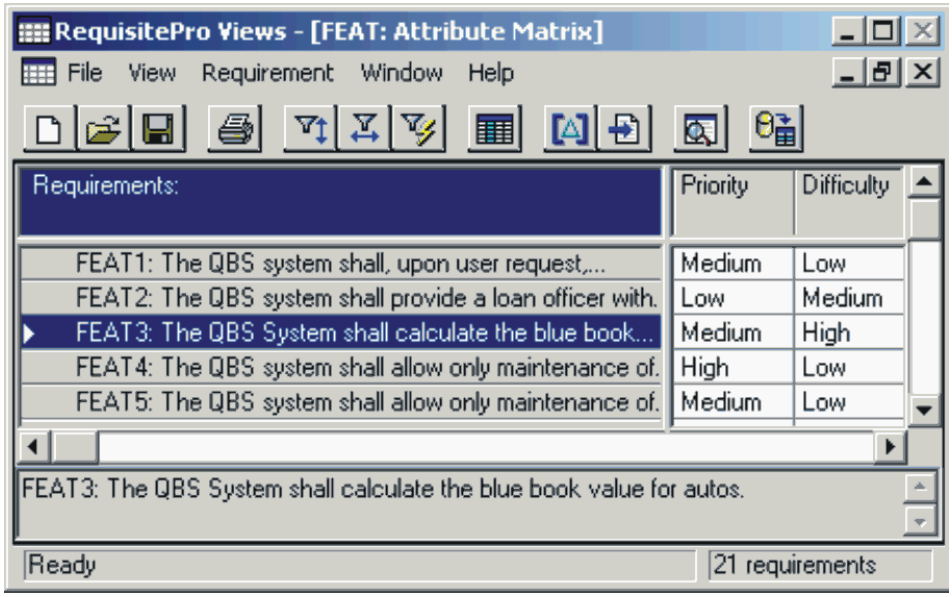

그림 7—각 요구사항의 속성 값 설정

변경 히스토리

개별 요구사항과 요구사항 콜렉션 둘 다 시간 경과에 따라 의미 있는 히스토리를 갖고 있습니다. 변화하는 환경과 발전하는 기술과 보조를 맞추려면 변경은 당연하고 바람직합니다. 프로젝트 요구사항 버전을 기록해 두면 팀 리더가 프로젝트 변경 이유(예: 새 시스템 릴리스)를 캡처할 수 있습니다. 요구사항 콜렉션이 소프트웨어의 특정 버전과 연관될 수 있음을 이해하면 점진적으로 변경을 관리하여 위험성을 줄이고 이정표 충족 가능성을 개선할 수 있습니다. 개별 요구사항이 발전함에 따라 히스토리(변경 내용, 이유, 시기 및 권한 부여자)를 이해하는 것이 중요합니다.

요구사항 관리를 작업에 배치

요구사항 관리는 문제점을 성공적으로 식별하고 해결하기 위해 위에 제공된 핵심 스킬과 개념을 이용합니다.

고객의 요구를 충족하는 시스템을 빌드하려면, 프로젝트 팀은 먼저 시스템이 해결할 문제점을 정의해야 합니다. 그 다음에, 팀은 비즈니스 및 사용자 요구를 도출, 설명하고 우선순위를 결정하는 이해 당사자(stakeholder)를 식별해야 합니다. 이 상위 레벨 기대 또는 요구 세트를 통해 제품 또는 시스템 기능 세트에 대한 합의가 이루어져야 합니다.

고객과 개발 팀이 둘 다 이해하는 양식으로 세부 소프트웨어 요구사항을 작성해야 합니다. 고객의 언어를 사용하여 이런 소프트웨어 요구사항을 설명하는 것이 고객의 이해와 합의에 도달하는 데 가장 효율적임을 파악하였습니다. 이런 세부 소프트웨어 요구사항은 구현 및 유효성 검증에 필요한 테스트 계획 및 프로시저에는 물론 시스템 디자인 스펙의 입력으로 사용됩니다. 또한 소프트웨어 요구사항은 초기 사용자 문서 계획 및 디자인을 주도해야 합니다.

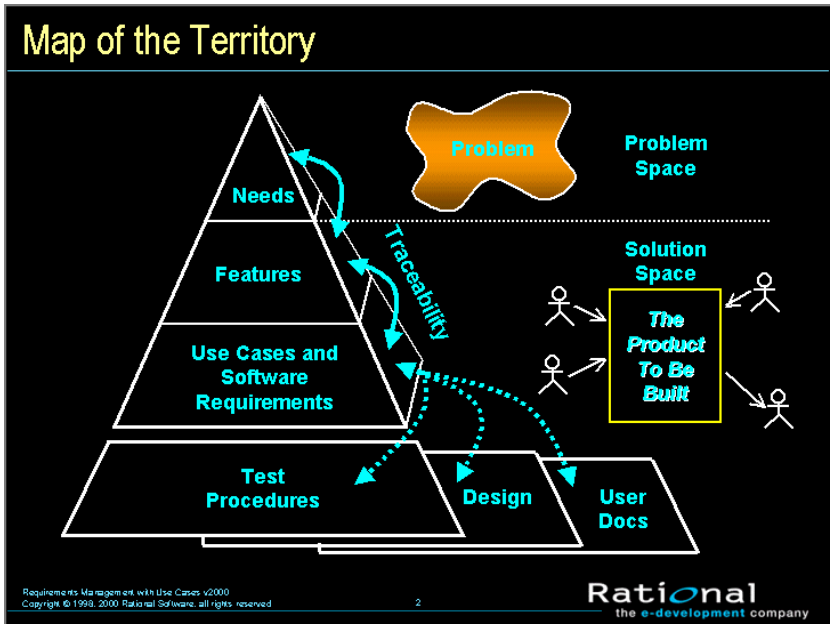

그림 8—요구사항 관리 구조 개요

이를 용이하게 하려면 프로젝트 팀은 다음을 수행해야 합니다.

- 프로젝트의 공통 용어에 합의

- 시스템이 해결할 문제점은 물론 1 차 기능을 설명하는 시스템의 비전 개발
- 최소 다섯 개의 중요한 영역(기능성, 사용성, 신뢰성, 성능 및 지원 가능성)에서 이해 당사자(stakeholder)의 요구 도출
- 사용할 요구사항 유형 판별
- 각 요구사항 유형의 속성과 값 선택
- 요구사항을 설명하는 형식 선택
- 하나 이상의 요구사항 유형을 작성, 컨트리뷰션하거나 보기만 하는 팀 구성원 식별
- 필요한 추적성 결정
- 요구사항 변경을 제안, 검토 및 해결하기 위한 프로시저 설정
- 요구사항 히스토리를 추적하기 위한 메커니즘 개발
- 팀 구성원 및 관리를 위한 진행상태 및 상태 보고서 작성

이런 필수적인 요구사항 관리 활동은 업종, 개발 방법 또는 요구사항 도구와 관계 없습니다. 또한 이러한 요구사항 관리 활동은 유연하므로, 이를 통해 가장 엄밀하고 가장 신속한 응용프로그램 개발 환경에서 효과적인 요구사항 관리가 가능합니다.

문서에 대한 몇 가지 주의사항

문서로 요구사항을 설명하겠다는 결정은 신중을 기해야 합니다. 한편으로 기록은 널리 허용되는 커뮤니케이션 양식이며 대부분의 사람들에게는 자연스런 일입니다. 다른 한편으로 프로젝트의 목적은 문서가 아닌 시스템을 생성하는 것입니다.

상식과 경험을 통해 이 결정은 요구사항을 문서화할 것인지 여부가 아니라 요구사항을 문서화하는 방법에 대한 것임을 알 수 있습니다. 문서 템플리트는 요구사항 관리를 위한 일관된 형식을 제공합니다. Rational RequisitePro 는 이런 템플리트 및 문서 내 요구사항을 모든 프로젝트 요구사항이 들어 있는 데이터베이스에 링크시키는 추가 기능을 제공합니다. 이 고유 기능을 통해 요구사항을 관계형 데이터베이스에서 보다 쉽게 액세스할 수 있고 쉽게 관리할 수 있도록 하는 동시에 자연스럽게 문서화할 수 있습니다.

요구사항 관리: 활동

요구사항 관리는 무수한 도메인별 경로를 따를 수 있습니다. 다음 접근 방식은 각각의 핵심 요구사항 관리 스킬에 적용되지만 임의 도메인에 적용할 수 있는 여섯 개의 세부 활동을 규정합니다.

다음 활동 다이어그램은 Rational Unified Process [6] 요구사항 원칙에서 발췌한 것입니다. 이 활동은 그림 9 에 요약된 바와 같이 역할, 타스크 및 아티팩트(입력 또는 출력) 관점에서 표현됩니다. 이 문서에 함께 표시되는 텍스트는 요구 사항 관리 프로세스 개선에 대한 생각과 관심을 유발하기 위해 각 활동을 간략히 설명합니다. Rational Unified Process 에 대한 자세한 정보는 www.ibm.com/software/rational을 참조하십시오.

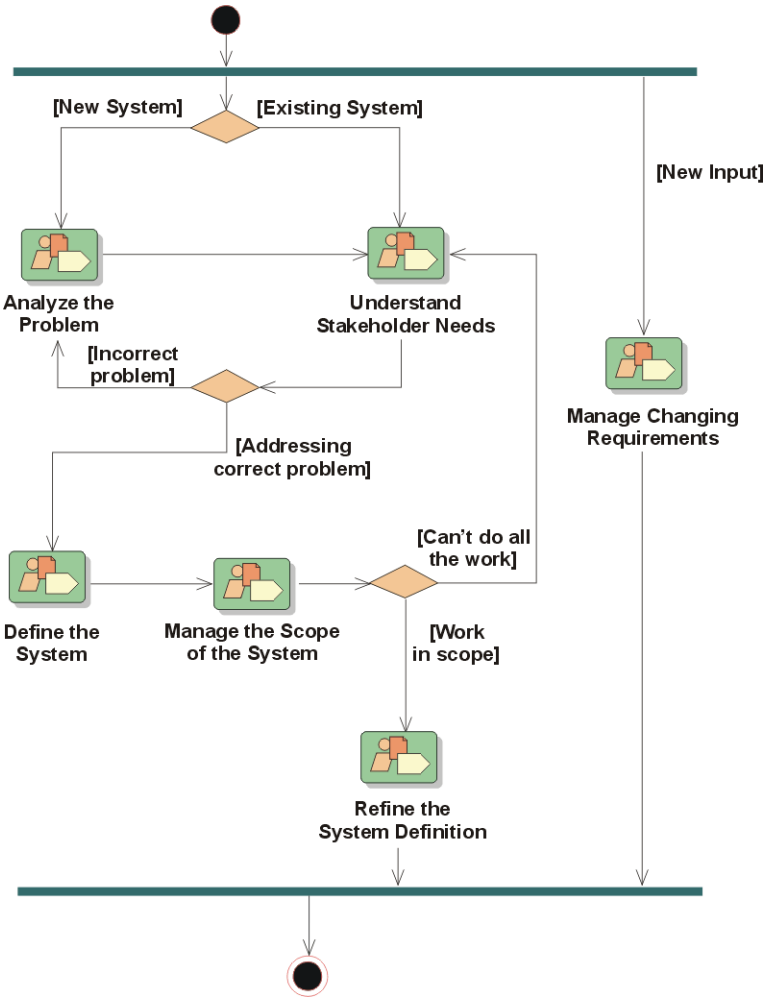

그림 9—Rational Unified Process 의 요구사항 워크플로우

활동: 문제점 분석

문제점 분석에서 기본 활동은 프로젝트의 비전을 개발하는 것입니다. 이 활동의 산출물은 상위 레벨 사용자 또는 빌드될 시스템의 사용자 보기를 식별하는 비전 문서입니다. 비전은 가장 중요한 문제점을 해결하고 핵심 이해 당사자(stakeholder) 요구를 충족하려면 시스템이 보유해야 하는 핵심 기능으로 초기 요구사항을 표현합니다. 시스템 분석가는 이 활동에서 기본 역할을 갖습니다. 시스템 분석가는 문제점 도메인 전문 지식을 보유하고 문제점을 이해하고 있어야 하며, 문제점을 해결할 것으로 믿는 프로세스를 설명할 수 있어야 합니다. 다양한 프로젝트 이해 당사자가 적극적으로 참여해야 하며, 모든 관련 이해 당사자 요청을 고려해야 합니다.

종속성 관리 타스크를 시작하려면 기능을 속성(예: 근거, 상대값 또는 우선순위 및 요청 소스 등)에 지정해야 합니다. 비전이 개발될 때 분석가는 가능한 유스 케이스의 사용자 및 시스템(액터)을 식별합니다. 액터는 유스 케이스 모델의 첫 번째 요소이며, 시스템의 기능적 및 비기능적 기술 요구사항을 정의합니다.

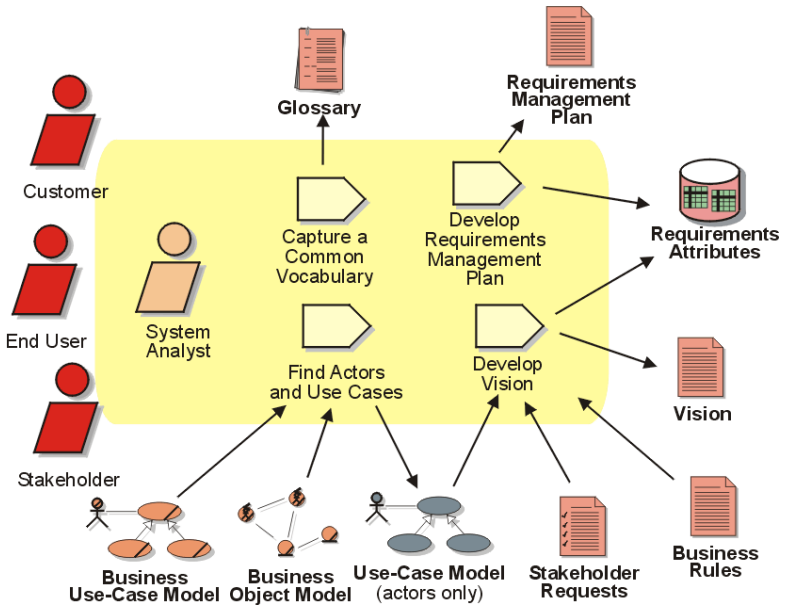

그림 10—문제점 분석

활동: 문제점 분석

시작: 문제점을 인식하는 하나 이상의 이해 당사자(stakeholder)가 활동을 시작합니다.

개발 팀의 시스템 분석가는 세션을 용이하게 하여 초기 이해 당사자가 해결을 원하는 문제점을 설명할 수 있도록 합니다. 인식된 문제점에 대한 간결한 설명에 관한 합의에 도달하는 것이 중요합니다. **문제점 설명의 핵심 요소는** 다음 표에 표시되어 있습니다.

문제점	문제점 정의
이해 당사자에 영향	문제점의 영향을 받는 이해 당사자 나열
문제점의 영향	문제점의 영향 설명
성공적인 솔루션	성공적인 솔루션의 몇 가지 주요 이점을 나열

문제점 설명은 프로젝트의 목적을 간결하게 설명합니다. 문제점 분석은 강력한 이점과 대략적인 예상 비용을 포함하여 모든 이해 당사자(stakeholder) 요청과 초기 비즈니스 사례를 심층 조사하도록 자극합니다. 문제점 설명 정의와 동시에 팀은 일반적으로 사용되는 용어를 추적하고 용어 정의에 합의하여 용어집을 컴파일해야 합니다.

유스 케이스 모델 소개

유스 케이스 모델은 액터, 유스 케이스 및 이들 간의 관계로 이루어져 있습니다. 액터는 일반적인 사용자를 포함하여 시스템과 정보를 교환해야 하는 모든 것을 표시합니다. 액터가 시스템을 사용할 때 시스템은 유스 케이스를 수행합니다. 우수한 유스 케이스는 액터에 대한 측정 가능한 결과를 제공하는 트랜잭션 시퀀스입니다. 유스 케이스 콜렉션은 시스템의 전체 기능성입니다.

> —Jacobson I., Christerson M., Jonsson P., Overgaard G., 객체 지향 소프트웨어 엔지니어링 – 유스 케이스 기반 접근 방식, Addison Wesley – ACM Press, 1992

또한 문제점 분석은 기본 시스템 액터를 식별합니다. 액터는 시스템 또는 시스템과 정보를 교환할 다른 시스템의 사용자입니다. 이 단계에서 문제점 분석은 액터가 시스템과 상호작용할 몇 가지 명백한 방법을 간략히 식별해야 합니다. 설명은 시스템 동작이 아니라 비즈니스 프로세스 위주여야 합니다. 예를 들어, 예산 수립 프로그램은 하나의 액터 유형이 "부서별 예산을 작성"할 수 있도록 하는 반면, 다른 액터는 "부서별 예산 통합"을 할 수 있게 합니다. 시스템 분석가는 나중에 이를 특정 시스템 동작과 보다 의미 있게 맞추는 추가 유스 케이스로 나눌 수 있습니다. 예를 들어, "부서별 예산 작성"의 결과로 "스프레드시트 정보 가져오기" 및 "예산 보기 작성"과 같은 시스템 유스 케이스가 발생할 수 있습니다.

앞서 설명한 문제점 분석 세션은 다른 이해 당사자(stakeholder)와 함께 내부 개발 팀 세션과 섞여 두 번 이상 수행되는 경우가 많습니다. 이해 당사자와의 회의를 수행하는 시스템 분석가는 개발 팀 구성원과 세션을 주도하여 문제점에 대한 기술적 솔루션을 계획하고 초기 이해 당사자 입력에서 기능을 도출하며, 빌드될 시스템의 첫 번째 정의가 되는 비전 설명의 초안을 작성합니다. 초기 이해 당사자들 간에 제안된 솔루션에 대한 이해를 촉진하기 위해 시스템 분석가는 모델링 도구나 수동 드로잉 기법을 사용하여 비전 설명을 보완할 수 있습니다.

시작하는 이해 당사자(stakeholder)는 문제점 설명을 정제하고 가능한 솔루션의 수와 범위를 제한할 수 있도록 여러 지점에서 조언을 받습니다. 이해 당사자 및 시스템 분석가는 핵심 기능의 우선순위를 협상하고 핵심 기능을 개발하는 데 필요한 자원과 노력을 일반적으로 이해하여 이 활동에서 종속성을 관리합니다. 우선순위와 노력 또는 자원이 불가피하게 변경을 예상하더라도 종속성 조기 관리는 개발 라이프사이클 전반에서 계속되는 중요한 패턴을 설정합니다. 이는 범위 관리의 본질이며, 프로젝트 성공의 조기 예측입니다.

몇 개의 초안을 작성한 후에 비전은 팀이 추가 요구사항 도출에 투자해야 할지 여부를 결정해야 하는 지점에 도달합니다. 이즈음에 비즈니스 사례 승인 프로세스가 별도로 시작되었습니다. 이 문서에서는 더 이상 다루지 않지만 비즈니스 사례는 다음을 모두 설명합니다.

- 컨텍스트(제품 도메인, 시장 및 범위)
- 기술적 접근 방식
- 관리 접근 방식(스케줄, 위험성, 성공의 객관적 척도)
- 재무 예측

활동: 이해 당사자의 요구 이해

초기 문제점 분석 결과 추가 투자의 정당성이 입증되면, 이해 당사자(stakeholder)의 요구 이해 활동이 진지하게 시작됩니다. 핵심 타스크는 이해 당사자 요청을 도출하는 것입니다. 기본 산출물은 우선순위가 결정된 이해 당사자 요구와 기능 콜렉션으로, 비전 문서 정제를 가능하게 하고 요구사항 속성에 대한 탁월한 이해를 제공합니다. 또한 이 활동 중에 유스 케이스 및 액터의 관점에서 시스템 논의를 시작할 수 있습니다. 또 다른 중요한 산출물은 팀 구성원 간에 공통 용어 사용을 촉진하기 위해 갱신된 용어집입니다.

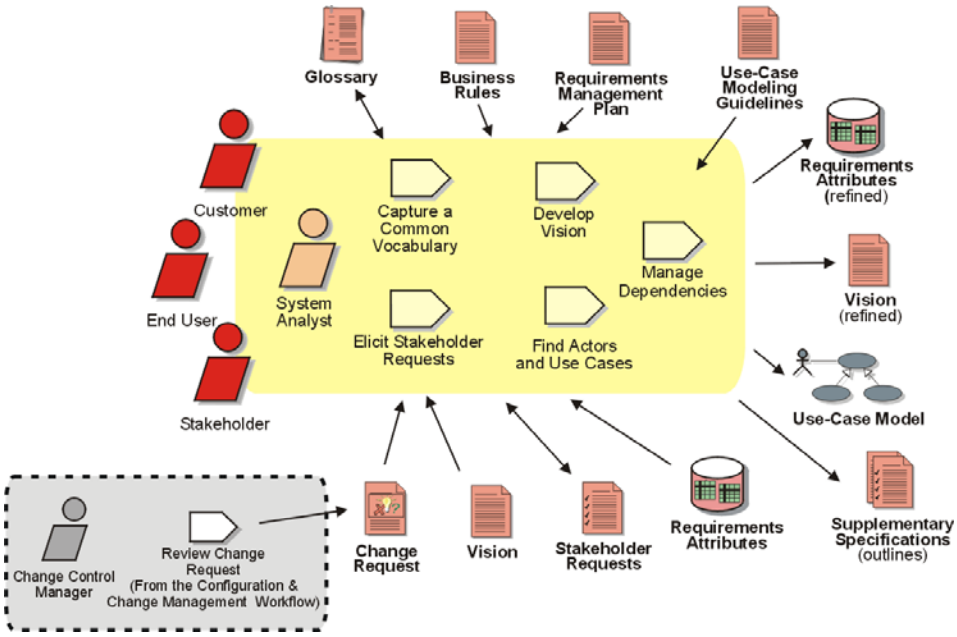

그림 11—이해 당사자의 요구 이해

활동: 이해 당사자의 요구 이해

시스템 분석가 및 핵심 이해 당사자(stakeholder)는 이해 당사자를 추가로 식별하고 이해 당사자의 요청을 도출하며, 인터뷰, 워크샵, 스토리보드, 비즈니스 프로세스 유스 케이스 및 기타 기법을 통해 핵심 요구와 기능을 판별합니다. 한 명 이상의 시스템 분석가가 이런 세션을 용이하게 합니다. 이런 요구사항 워크샵은 가장 유용한 도출 기법 중 하나입니다. 프로세스에는 사용자, 헬프 데스크 담당자, 비즈니스 소유자, 테스터 및 제안된 프로젝트의 결과물과 관련된 다른 사람이 포함됩니다. 이해 당사자 요청은 모호하고 겹치며, 이질적인 경우가 많습니다. 정규 도출 결과 이외에 이해 당사자 요청을 제대로 형식화된 문서, 데이터베이스로부터의 결함 및 개선사항 요청 또는 전자 우편 및 그룹웨어 스레드로 표현할 수 있습니다. 시스템 분석가는 식별된 핵심 이해 당사자의 요구를 기록, 분류하고 우선순위를 결정합니다.

이해 당사자의 요구 이해: "고객 감동"의 출발점

이해 당사자(stakeholder) 요청은 요청하는 이해 당사자의 언어와 형식으로 가능한 최대한 캡처됩니다. 대개 프로세스 교육을 받고 기술적으로 능숙한 프로젝트 팀 구성원이 작성하는 후속 요구사항 유형과 달리 이해 당사자 요청은 제대로 표현되지 못하는 경우가 많습니다. 중복되거나 겹칩니다. 종이 쪽지에서부터 개선사항 요청 데이터베이스에 이르기까지 어디에나 이해 당사자 요청을 표현할 수 있습니다.

분석가(또는 분석가의 역할을 나타내는 팀)는 모든 요청을 검토하고 해석하며, 그룹화하고 다시 입력(다시 작성하지 않고)한 후 이를 핵심 이해 당사자 요구 및 시스템 기능으로 비전 문서에서 변환해야 합니다. 개발에 적용된 엄격함과 도구 가용성에 따라 일부 또는 모든 이해 당사자(stakeholder) 요청, 요구 및 기능 간의 추적성을 적용하여 이해 당사자의 요청과 요구가 시스템의 요구사항을 판별하는 데 고려되는 방법을 이해 당사자가 이해하도록 할 수 있습니다.

이해 당사자(stakeholder)의 요구 이해 활동을 적용하여 요청을 도출하고 이해 당사자의 요구를 충족시키기 위한 진지한 관심을 보여주는 것이 이해 당사자가 개발 팀의 능력을 신뢰할 수 있도록 하는 데 중요할 수 있습니다.

이해 당사자(stakeholder) 요구에 대한 탁월한 이해를 기반으로 개발 팀의 시스템 분석가는 비전 문서를 정제하고 "제품 포지션 설명" 개발에 특별한 주의를 기울입니다. 두 개 또는 세 개의 문장으로 된 이 설명은 프로젝트의 강력한 가치를 확립합니다. 설명에는 의도된 사용자, 이 제품이 해결하는 문제점, 전달하는 이점 및 대체하는 경쟁제품이 포함되어야 합니다. 팀 구성원 모두가 이 프로젝트 주제를 이해해야 합니다. 시스템 분석가는 용어에 대한 공통 이해를 촉진하기 위해 용어집도 갱신합니다.

이해 당사자(stakeholder)의 요구 이해에서 파생된 새로운 기능의 우선순위를 협상하고 새 기능을 개발하는 데 필요한 노력과 자원을 이해할 수 있도록 여러 지점에서 핵심 이해 당사자와 협의합니다. 문제점 분석의 경우와 마찬가지로 이 활동에서 종속성 관리는 범위 관리에 도움을 줍니다. 또한 이해 당사자 요청, 요구 및 시스템 기능 간의 추적성을 설정하므로 이해 당사자는 자신들의 입력이 고려되었음을 확신할 수 있습니다.

활동: 시스템 정의

처음 두 개의 요구사항 활동(문제점 분석 및 이해 당사자의 요구 이해)은 비전 문서에 지정된 기능, 유스 케이스 모델의 첫 번째 아웃라인 및 초기 요구사항 속성을 포함하여 핵심 시스템 정의의 조기 반복을 작성합니다. 다음 활동(시스템 정의)은 기능 정의 정제, 새 액터, 유스 케이스 및 보충 스펙 추가를 통해 상위 레벨 시스템 요구사항에 대한 설명을 완료합니다.

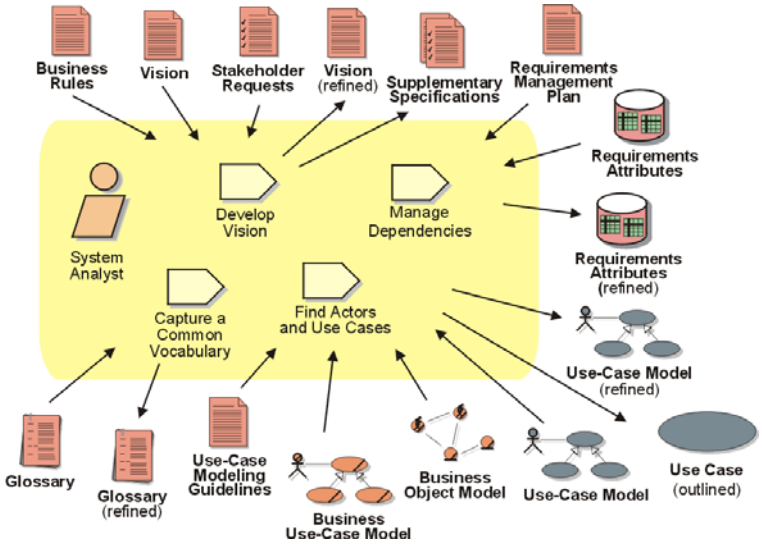

그림 12—시스템 정의

활동: 시스템 정의

유스 케이스 모델 및 보충 스펙에 캡처된 요구사항과 기능을 설명하는 데 사용되는 용어에 대한 현재 이해를 반영하도록 용어집이 갱신됩니다.

시스템 분석가는 정제된 비전에 정의된 기능 세트를 사용하여 시스템의 예상 동작에 대한 사용자의 보기를 고안하는 유스 케이스를 도출하고 설명합니다. 유스 케이스 모델은 선택된 기능이 시스템에서 작동하는 방법에 관한 고객, 사용자, 시스템 개발자 간의 계약으로 작용합니다. 유스 케이스 모델은 시스템 개발자를 위해 현실적 기대와 목적을 설정하며, 고객과 사용자가 시스템이 이런 기대를 충족시킬 것임을 확인하도록 돕습니다.

일부 시스템 요구사항은 유스 케이스에 적합하지 않습니다. 시스템 분석가는 보충 스펙에서 이를 설명합니다. 사용성, 신뢰성, 성능 및 지원 가능성 요구사항과 같은 많은 비기능적 요구사항은 여기서 끝나는 경우가 종종 있습니다. 이런 유형의 많은 비기능적 요구사항은 단일 유스 케이스에 고유함에 유의해야 합니다. 유스 케이스 지정자가 이런 요구사항을 유스 케이스 명세 자체에 배치하고(시스템 정제 활동 참조) 보충 스펙은 시스템 전체의 비기능적 요구사항용으로 남겨 두는 것이 좋습니다.

이 활동에서 시스템 분석가는 우선순위 및 관련 유스 케이스와 같은 보충 요구사항의 속성을 작성하고 설정합니다. 또한 시스템 분석가는 초기 및 새 유스 케이스의 속성 값을 추가하고 갱신합니다.

마지막으로, 시스템 분석가는 보충 스펙에 설명된 요구사항 및 관련 유스 케이스까지 중요한 사용자 요구와 핵심 기능을 추적하여 종속성을 관리합니다.

활동: 시스템 범위 관리

기능 레벨 요구사항을 식별하고 대부분의 액터, 유스 케이스 및 보충 스펙에 지정된 기타 요구사항을 설명한 후에 시스템 분석가는 값을 수집하여 우선순위, 노력, 비용 및 위험성과 같은 요구사항 속성에 가능한 정확히 값을 지정해야 합니다. 이렇게 하면 시스템 릴리스의 초기 범위 판별 방법을 보다 잘 이해할 수 있으며, 설계자가 구조적으로 중요한 유스 케이스를 식별할 수도 있습니다.

프로젝트 및 개발 관리를 통해 동시에 개발되는 반복 계획은 시스템 범위 관리 활동에 처음 나타납니다. 개발 계획이라고도 하는 반복 계획은 릴리스용으로 계획된 반복 횟수와 빈도를 정의합니다. 범위 내에서 가장 위험성이 높은 요소를 조기 반복용으로 계획해야 합니다.

범위 관리 활동의 기타 중요한 산출물로는 소프트웨어 아키텍처 문서의 초기 반복 및 분석가와 핵심 이해 당사자(stakeholder)의 시스템 기능성과 프로젝트 자원에 대한 증대된 이해를 반영하는 수정된 비전을 들 수 있습니다.

앞서 언급한 비즈니스 사례 및 반복 계획의 첫 번째 문제와 마찬가지로 소프트웨어 아키텍처 문서는 Rational Unified Process 와 관련되어 있고 Rational Unified Process 의 일부이긴 하지만 요구사항 관리의 아티팩트는 아닙니다. 소프트웨어 아키텍처 문서는 이 문서의 주제가 아닙니다.

> 경험을 통해 범위를 성공적으로 관리하는 핵심은 이해 당사자(stakeholder) 대표와 정기적으로 개방된 마음으로 솔직하게 상호작용하는 것은 물론 이해 당사자 요구, 기능, 유스 케이스 및 보충 스펙에 지정된 기타 요구사항에 지정된 속성 값을 신중히 고려하는 것임을 알 수 있습니다.

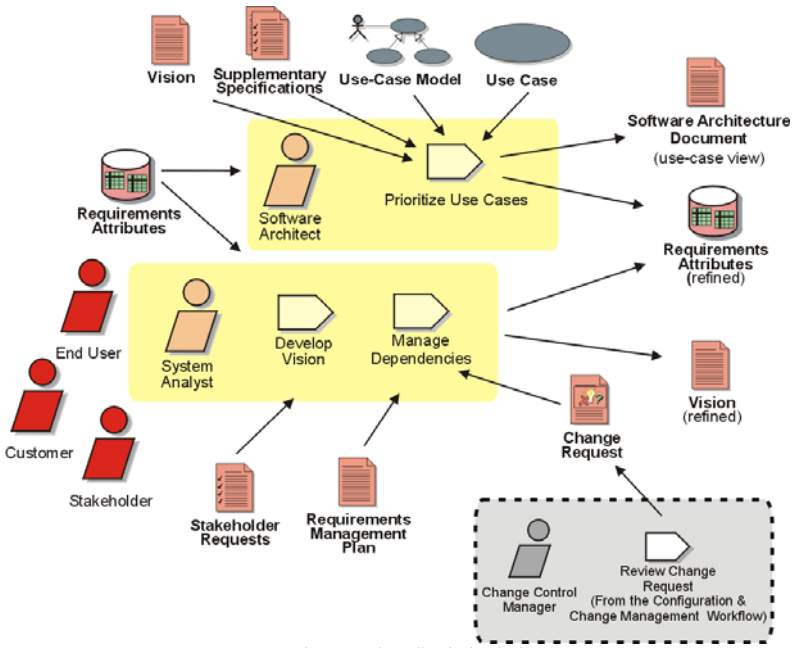

그림 13—시스템 범위 관리

활동: 시스템 범위 관리

설계자는 위험성 적용 범위, 아키텍처 중요성 및 아키텍처 적용 범위에 대해 유스 케이스 우선순위를 결정합니다. 다수의 유스 케이스 및 보충 스펙 요구사항을 사용하여 시스템을 정의할 수 있지만 대개 유스 케이스 서브세트만 우수한 시스템 아키텍처에 중요합니다. 유스 케이스 우선순위가 결정된 상태에서 설계자는 반복 또는 개발 계획을 정제하고 Rational Rose 와 같은 도구로 시스템 아키텍처의 유스 케이스 보기를 모델링합니다.

시스템 분석가는 비전에서 기능의 요구사항 속성을 정제하여 종속성을 관리합니다. 또한 유스 케이스와 보충 스펙에서도 요구사항을 정제합니다. 시스템 분석가는 이해 당사자(stakeholder) 요구, 기능, 유스 케이스 요구사항 및 보충 스펙 요구사항에 대해 적절한 추적성*이 존재하는지 확인합니다. "적절한 추적성"이란 말은 신중합니다. 이 문서의 뒷부분에 있는 추적성에 관한 삽입 텍스트를 참조하십시오.

이 단계는 전체 프로젝트에서 가장 중요한 단계 중 하나입니다. 요구사항, 프로젝트 자원 및 전달 날짜에 대한 진지한 약속을 위해 제안된 시스템에 대한 폭넓은 지식을 처음으로 활용할 수 있습니다. 동시에 이런 요구사항은 지식의 깊이가 증가하면서 변경됨을 이해해야 합니다. 이전 세 개 활동에서 종속성을 관리해 온 경우 이 단계가 훨씬 용이하며, 향후 변경이 용이합니다.

프로젝트 수명 전반에서 상황과 환경이 변경됨에 따라 시스템 분석가가 핵심 이해 당사자(stakeholder)와 수정된 프로젝트 범위와 비전을 협상해야 하므로 이 활동을 여러 번 재검토합니다. 사용 가능한 자원에 맞춰 프로젝트 범위를 관리하는 것은 이해 당사자와 개발 팀 구성원이 이 단계를 사용자의 기대에 대한 복병이나 더 많은 시간과 돈을 벌기 위해 조직을 속이려는 시도가 아닌 자연적인 진행으로 간주하는 경우에만 성공적입니다. 시스템과 시스템의 문제점에 대한 새로운 통찰에 따라 범위 변경이 필요한지 여부를 평가하려면 이 활동이 프로젝트의 주요 이정표에서 반복되어야 합니다. 확약된 요구사항, 예산 및 최종 기한은 변경하기 어렵지만 우선순위가 결정된 유스 케이스, 보충 스펙 및 조기 시스템 반복을 심층 이해하면 불가피하게 범위를 재고하게 됩니다.

다시 한 번, 프로젝트 팀은 변경사항이 발생할 때 프로젝트 라이프사이클 전반에서는 물론 정제 단계에 도달하기 전에 습관적으로 범위 관리를 시작하는 것이 중요합니다. 이해 당사자(stakeholder) 대표는 점점 어려워지는 범위 협상 중에 자신들의 우선순위와 관심사를 진지하게 고려한다는 점을 이해하고 신뢰해야 합니다. 시스템 요구사항이 정제될 때쯤에는 중요한 요구사항만 협상되거나 수정됩니다. 효과적인 범위 관리 습관이 확립되지 않았다면 프로젝트는 "죽음의 행진"(즉, 어쩔 수 없이 지연과 비용 초과를 초래하는 절망적으로 범위를 벗어난 프로젝트)이 될 수 밖에 없습니다.

활동: 시스템 정의 정제

시스템 정의 정제로 진행하면 이 활동 세부사항은 시스템 레벨 유스 케이스가 개괄적으로 설명되고 액터가 최소한 간략히 설명되었다고 가정합니다. 프로젝트 범위 관리를 통해 비전에서 기능은 우선순위가 다시 결정되었으며, 이제 상당히 확고한 예산과 날짜로 이런 기능을 달성할 수 있다고 여겨집니다. 이 활동의 산출물은 세부 유스 케이스에 표현된 시스템 기능성, 수정된 보충 스펙 및 시스템 자체의 조기 반복을 보다 심층 이해하는 것입니다.

분명히 모든 시스템에 사용자 인터페이스가 있는 것은 아니며, 모든 조기 반복에 GUI 요소가 포함되는 것은 아닙니다. 여기서는 이를 조기 반복의 예제로만 사용합니다. 기타 예제로는 프로토타입, 모델 및 스토리보드가 있습니다.

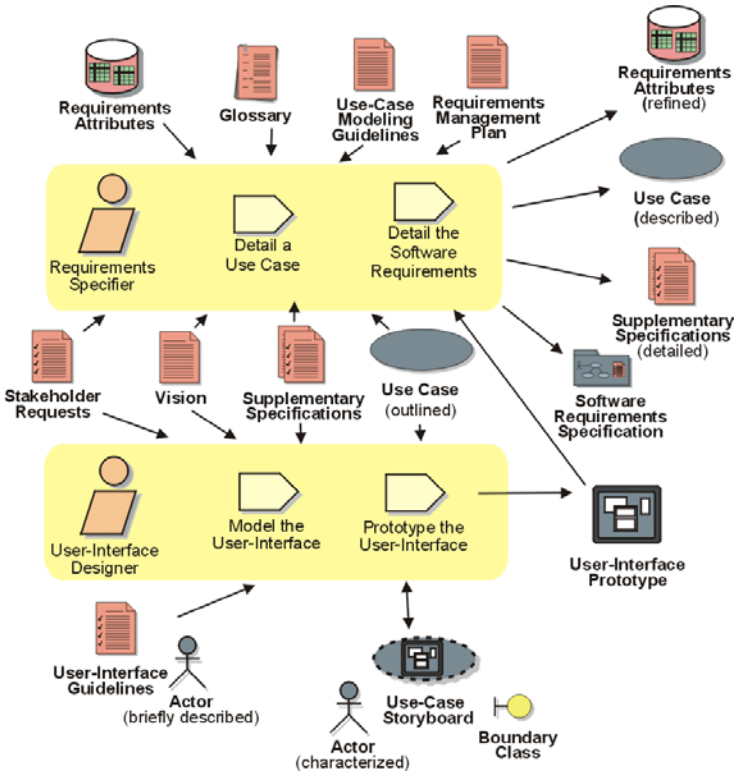

그림 14—시스템 정의 정제

활동: 시스템 정의 정제

유스 케이스 지정자는 이벤트 플로우의 정의, 사전 조건과 사후 조건 및 각 유스 케이스의 기타 텍스트 특성을 자세히 설명합니다. 노력을 최소화하고 판독성을 향상시키려면 표준 문서 형식을 사용하거나 유스 케이스 명세 템플리트를 사용하여 각 유스 케이스에 대한 텍스트 정보를 캡처하는 것이 좋습니다. 체계적인 유스 케이스 명세를 작성하는 것이 시스템 품질에 중요합니다. 이 명세를 개발하려면 이해 당사자(stakeholder) 요구 및 유스 케이스와 관련된 기능을 철저히 이해해야 합니다. 프로젝트 팀의 몇몇 구성원(예: 소프트웨어 엔지니어)이 유스 케이스 작성에 참여하는 것이 바람직합니다.

동시에 유스 케이스 지정자는 유스 케이스에 특정하지 않은 추가 요구사항으로 보충 스펙을 수정합니다.

사용자 인터페이스 디자이너는 시스템의 사용자 인터페이스를 모델링하고 프로토타입화합니다. 이 작업은 유스 케이스 전개와 긴밀한 관련이 있습니다.

유스 케이스 지정자 및 시스템 분석가는 노력, 비용, 위험성 및 제대로 이해된 각 요구사항의 기타 속성 값을 수정합니다.

이 시스템 정의 정제의 결과는 범위 관리 활동의 다른 라운드로 제출됩니다. 시스템에 대해 자세히 알게 되면 우선순위를 변경할 수 있습니다. 명확히 시스템 릴리스의 범위를 검토하고, 필요하면 정제해야 합니다.

활동: 요구사항 변경 관리

변경사항이 발생할 때(변경사항은 불가피하게 발생함) 시스템 범위 관리에 대해 설명한 바와 같이 요구사항 변경 관리 활동을 프로젝트 수명 전반에서 계속해서 적용해야 합니다. 이 활동의 산출물로 인해 모든 아티팩트가 수정될 수 있으며, 이는 다시 모든 프로젝트 팀 구성원과 이해 당사자(stakeholder) 간에 효과적인 커뮤니케이션을 필요로 합니다.

이 활동에서는 요구사항 활동에 의해 영향을 받는 추가 아티팩트를 소개합니다. 요구사항 변경은 분석 및 디자인 활동에 표시된 시스템 모델에 당연히 영향을 줍니다. 요구사항 변경은 요구사항이 적절히 구현되었는지 유효성을 검증하기 위해 작성된 테스트에도 영향을 줍니다. 앞의 예제의 경우와 마찬가지로 이런 아티팩트는 Rational Unified Process 의 일부이기는 하지만 이 문서의 주제는 아닙니다. 종속성 관리 프로세스에서 식별된 추적성 관계는 이런 영향을 이해하는 핵심입니다.

추적성

요구사항 필드에서는 추적성으로 이루어지는 것이 많습니다. 관련된 각 스펙, 테스트, 모델 요소 및 궁극적으로 소스 코드 파일까지 개별 고객 요구사항을 추적하는 장점을 촉진하는 것이 많습니다. 분명히 일부 추적성은 성공적인 요구사항 변경 관리의 핵심입니다.

그러나 모든 추적성 양식에는 프로젝트 수명 동안 설정하고 유지보수하기 위한 투자가 필요함을 사전에 알아 두십시오. 모든 다른 투자와 마찬가지로 추적성에도 특정 상황에 따라 수익 체감 지점이 있습니다. 이 문서에서 서로 다른 요구사항 유형 간의 추적 값을 강조합니다. 여기서 시작하는 것이 좋으며, Rational RequisitePro, Rational Rose, Rational SoDA 및 Rational TeamTest 와 같은 도구를 사용하여 자동화할 수 있습니다. 최적의 투자가 될 수 있도록 어느 정도의 요구사항 추적성 레벨을 찾을 수 있습니다.

서로 다른 요구사항 추적성 전략에 관한 자세한 정보는 "유스 케이스로 요구사항을 관리하기 위한 추적성 전략" 백서를 참조하십시오. [6]

요구사항 변경 관리의 또 다른 중요한 개념은 요구사항 히스토리 추적입니다. 요구사항 변경의 네이처 및 근거를 캡처하여 검토자(변경으로 인해 작업에 영향을 받는 소프트웨어 프로젝트 팀의 일원)는 변경에 적절히 응답하는 데 필요한 정보를 수신합니다.

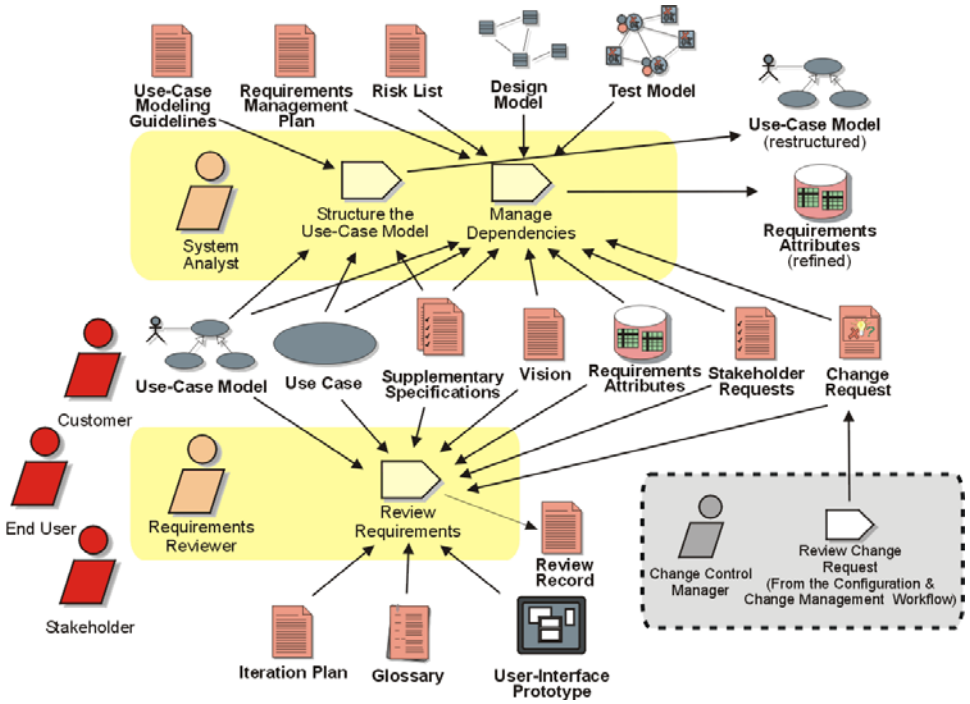

그림 15—요구사항 변경 관리

활동: 요구사항 변경 관리

다양한 이유로 이해 당사자(stakeholder) 또는 프로젝트 팀 구성원이 요구사항 변경 요청을 시작할 수 있습니다. 결함은 물론 요구사항 또는 개선사항 요청 변경을 포함하여 모든 변경 요청(CR)은 동일한 변경 요청 관리(CRM) 프로세스를 통해 모두 채널화되어야 합니다. 최소한 여기에는 중앙 집중식 데이터베이스 시스템에서의 요청 로깅 및 추적이 포함되어야 하며, 중앙 검토 권한을 통해 검토를 실시해야 합니다. 이러한 CRM 프로세스에 대한 자세한 내용은 Rational Unified Process 의 기타 섹션을 참조하십시오.

시스템 분석가는 검토 활동을 조정해야 하며(변경 제어 위원회(CCB)를 통한 조정 선호), 모든 변경 요청을 수집 및 검토하고 이를 다음과 같이 분류해야 합니다.

- 구현 시 요구사항에 영향을 주지 않는 결함
- 일부 기존 요구사항 유형에 대한 수정
- 개선사항 요청

일단 분류되면, 제안된 요구사항 변경에 기타 요구사항 활동에서 설명한 바와 같이 속성과 값이 지정됩니다.

변경 요청 검토 시, 시스템 분석가는 우선순위가 결정된 제안된 요구사항 변경을 이해 당사자(stakeholder) 대표와 프로젝트 팀 구성원으로 구성된 CCB 에 제공합니다. 자원을 초과하는 범위 수정은 거부되거나 날짜 및 예산 약속에 대한 필수 변경사항을 승인할 권한을 가진 이해 당사자 대표에게 이양되어야 합니다.

CCB 는 요구사항 변경을 승인하거나 거부합니다.

시스템 분석가는 요구사항 지정자와 요구사항 변경을 커뮤니케이션하거나 비전, 유스 케이스, 보충 스펙 문서 또는 기타 아티팩트에서 직접 요구사항을 변경합니다.

요구사항 검토자(개발자, 테스터, 관리자 및 기타 프로젝트 팀 구성원)는 요구사항 히스토리를 검토하여 요구사항 변경이 작업에 미치는 영향을 평가합니다. 마지막으로, 변경을 구현하고 권한을 갖고 있는 관련 요구사항을 적절히 변경합니다.

요약

요구사항 관리의 필요성은 새롭지 않습니다. 그렇다면, 무엇 때문에 앞서 설명한 정보를 지금 고려해야 합니까?

먼저, 프로젝트가 고객을 정식으로 만족시키지 못하고 최종 기한을 맞추지 못하며 예산을 초과하는 경우 개발 접근 방식을 다시 고려하는 것이 당연합니다. 이를 수행할 때 요구사항 관련 문제점이 개발 노력을 훼손함을 알게 되면, 보다 나은 요구사항 관리 사례를 고려하는 것이 당연합니다.

두 번째로, 이 문서에 요약된 요구사항 관리 사례는 수천 가지의 집합적 경험을 구체화하며, 요구사항 관리 분야에서 수년간 고객과 접해온 많은 개인의 검토된 의견을 잘 반영합니다. 이들의 컨트리뷰션으로 작성된 이 개요(Rational Unified Process 에서 이에 대한 좀 더 완전한 프리젠테이션 제공)는 요구사항 관리의 "최상의 사례"를 나타냅니다. 요구사항에 관한 보다 입증된 조언은 그 어디에도 없습니다.

Dr. Ivar Jacobson, Dean Leffingwell, Dr. Alan Davis, Ed Yourdon 및 Elemer Magaziner 의 직, 간접적인 기여에 대해 감사드립니다. 무엇보다 수백 개의 개발 프로젝트에서 이런 사례를 적용하고 개선해 온 Rational Software 고객에게 감사드립니다.

마지막으로, 지난 2 년 동안 효과적인 소프트웨어 개발 솔루션 작성 비즈니스에서 선두를 유지해 온 Rational Software 는 요구사항 관리라는 당면 과제에 몰두하여 이 어려운 타스크를 자동화하기 위한 도구를 개발하였습니다. 요구사항 관리의 고질적이면서 파급효과가 큰 문제점은 해결 가능합니다. 이러한 점이 궁극적으로 오늘날 요구사항 관리에서 우수성을 인정받기 시작하는 최상의 이유가 될 수 있습니다.

위 개념에 대한 전체 논의는 Dean Leffingwell 이 소프트웨어 요구사항 관리에 관해 저술한 우수한 서적을 참조하십시오.[7]

참조

[1] CHAOS, The Standish Group International, Inc., Dennis, MA, 1994, 1997.

[2] Computer Industry Daily, December 12, 1997.

[3] Dorfman, M. and R. Thayer, *Software Engineering*, IEEE Computer Society Press, Los Alamitos, CA, 1997 pp. 79.

[4] Dorfman, M. and R. Thayer, *Software Engineering*, IEEE Computer Society Press, Los Alamitos, CA, 1997 pp. 80.

[5] Spence, Ian and Leslee Probasco, *Traceability Strategies for Managing Requirements with Use Cases*, White Paper, Rational Software Corporation, 1998.

[6] Rational Unified Process , Rational Software Corporation, Cupertino, CA, 1999.

[7] Leffingwell, Dean and Don Widrig, *Managing Software Requirements — A Unified Approach*, Addison-Wesley, 2000.

Rational
the software development company

본사 안내:

Rational Software
18880 Homestead Road
Cupertino, CA 95014
전화번호: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
전화번호: (781) 676-2400

수신자 부담 전화번호: (800) 728-1212
전자 우편: info@rational.com
웹: www.rational.com
전세계 지사 안내: www.rational.com/worldwide