개발방법론개요

|  |  |  |
| --- | --- | --- |
| http://tfile.nate.com/download.asp?FileID=27676529 |  | 방법론이란 무엇인가? |
|  |  | 최근의 정보시스템은 **복잡화, 대형화, 전략정보시스템화** 로 발전해 오면서 과거의 개발방식인 無방법론, 주먹구구식, 가내수공업형으로는 대응하기가 매우 어려워졌다. 특히, 기업의 Business System은 업무의 효율을 높이는 **MIS** 에서 정보시스템을 이용한 경쟁우위 창출의 개념인 **SIS** 로 요구사항의 수준이 상승되면서 시스템 전략의 계획에서부터 철저한 사용자의 요구분석과 설계를 통해 자동화 Tool에 의한 개발 공정에 이르기까지 그 생산 방법의 자동화 및 체계화가 필요하게 되었다.  이것이 시스템 구축의 공학화와 이의 근거인 정보시스템 개발 방법론의 등장 배경이라 할 수 있다. 실제로 기업의 정보시스템 개발을 주 업무로 하는 시스템 통합 작업은 많은 수의 기술자와 긴 일정을 통해 수행되고 있다. 따라서 한 두명의 개발자에 의지하기 보다는 팀을 통제하며 목표로 하는 시스템을 완성해 갈 수 있는 관리와 통제 즉, **프로젝트의 관리** 와 이의 바탕이 되는 **개발 방법론** 이 반드시 기반이 되어야 한다.   정보시스템 개발 방법론은 **정보시스템을 개발하기 위한 작업방법이나, 절차, 산출물, 기법 등을 논리적으로 정리해 놓은 체계** 를 말한다. 마치 음식을 만들 때 요리책을 보고, 익히고, 따라하듯이, 개발자들은 방법론을 이해하고 참조하면서 시스템의 계획, 분석, 설계, 구현, 운영의 SDLC(System Development Life Cycle)를 따라 정보시스템 개발을 수행하게 된다. 방법론은 바로 시스템 개발의 **이론적 기반** 이라고 할 수 있다. 방법론의 내용을 구성하는 주요 구성요소로는 다음과 같은 것들이 있다.   * **작업절차** - 프로젝트 수행시 이루어지는 작업단계의 체계 * **작업방법** - 각 단계별로 해야할 일들의 구체적인 설명 * **산출물** - 단계별로 만들어야 할 산출물의 목록과 작성방법, 또는 양식 등 * **기법** - 각 단계별로 작업 수행시에 소요되는 기술 또는 기법의 설명 * **관리** - 프로젝트 관리자의 입장에서 수행해야 할 작업 * **Tool** - 상황에 따른 필요 Tool과 그 적용방법   [그림 1]은 이러한 방법론의 구성요소를 구조화한 것이며, [그림 2]는 전체 개발 Life Cycle에 따른 CASE와의 관계를 설명하는 그림이다.  http://tfile.nate.com/download.asp?FileID=27676530 [그림 1] 방법론의 구성요소   http://tfile.nate.com/download.asp?FileID=27676531 [그림 2] 방법론과 CASE의 관계 |
|  |  | CASE Tool(특히 I-CASE Tool)은 개발 프로젝트에서 전체 공정의 산출물 작성과 코드생성 등을 자동화하고 일관성을 보장하는 도구이다. CASE Tool 자체도 작업 내용의 정형화된 흐름과 논리에 따라 구성되는데, 이것도 일종의 방법론이라 할 수 있다. 즉, CASE Tool은 방법론이 그 논리적 바탕이 되는 것이다. 반대로 방법론은 그 내용이 CASE Tool을 사용하여 따르지 않으면 효과성이 크게 떨어지는 것이 사실이다. 따라서 둘은 상호보완관계에 있다고 볼 수 있다.   방법론은 그 자체로는 기술이 아니지만 이를 표준화하고 이해함으로써 프로젝트의 생산성을 높이는 수단이 된다. 따라서 시스템을 개발하고 통합하는 업체의 입장에서는 방법론을 보유하는 것이 하나의 노하우고 경쟁무기가 된다.  외국의 정보시스템 컨설팅 업체 및 국내의 대표적인 SI업체들은 물론 당연히 이러한 방법론을 가지고 시스템 개발 프로젝트를 수행하고 있다. 대표적인 사례로는 METHOD/1(앤더슨 컨설팅), NAVIGATOR(언스트 앤 영), IEM(제임스마틴), 4FRONT(딜로이트), 이노베이터(삼성SDS), POS-IEM(포스데이타), SLC(LG-EDS), HIST4FRONT(한진정보통신) 등이 있다.  방법론을 만드는 것은 축적된 경험과 노하우를 오랜기간에 걸쳐 연구하고 정리하여야 하는 작업이다. 하지만 국내 업체들의 경우, 대부분 역사가 짧고 기술력이 풍부하지 못하여 외국의 방법론을 도입하는 예가 대부분이다. 일부는 원본을 들여와 자사의 환경에 맞게 개조(Customizing)하는 경우도 있다. 자체 개발의 경우도 있다고 하지만 그 내용의 신뢰성은 다소 떨어지는 것으로 파악된다. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| http://tfile.nate.com/download.asp?FileID=27676529 |  | 왜 방법론이 필요한가? |
|  |  | 정보시스템의 구축 또는 개발은 소프트웨어의 개발과는 차원이 다르다. 정보시스템은 단순 소프트웨어의 결합이 아니라 소프트웨어, 하드웨어, 시스템 소프트웨어, 네트워크, DBMS 등 모든 시스템 기술 요소가 결합된 형태이다. 따라서 시스템의 통합은 단순 프로그램 기술의 개발이나 절차의 개선 등으로 효과를 볼 수 있는 것이 아닌 것이다.  따라서 다음과 같은 과제가 설정될 수 있다.   * **정보시스템의 구축에서 개인의 경험이나 능력(skill)에 의존하는 부분을 줄일 수는 없는가?** * **최신의 기술(technology)을 IT전략에 반영시키는 효과적인 방법이 없는가?** * **다양한 능력을 가지고 있고, 입장이 서로 다른 사람들을 조직적이고도 효과적으로 정보시스템 구축에 참여시킬 수는 없는가?**   다음과 같은 성과를 위해 방법론의 적용이 필요하다고 볼 수 있다.  대규모 Business System 구축 프로젝트의 수행최근의 시스템(특히 Business System)은 더욱 복잡해지고 규모도 커지고 있다. 뿐만 아니라, 과거처럼 업무의 자동화 차원에 머물지 않고 **정보시스템을 이용한 기업의 전략적 경쟁우위를 달성** 하기 한다는 것을 목표로 하고 있다.  따라서 과거의 방식, 일부 구성원의 능력에 의존하는 프로젝트 수행은 이에 대응하기 어려울 뿐아니라, 일관된 진행과 관리의 어려움으로 품질좋은 시스템을 기대할 수 없게 된다.  프로젝트의 효과적인 관리방법론에 앞서 이러한 정보시스템 프로젝트에는 **관리** 가 매우 중요하다. 설사 개발자들의 기술력이 조금 떨어지는 한이 있더라도 풍부한 경험과 능력이 있는 프로젝트 관리자가 있다면 여러 환경 요인의 조화를 통해 부족한 부분을 어느 정도 해소하는 것이 가능할 수 있다.  물론 이러한 관리의 능력은 당연히 잘 구성된 방법론이 있으면 더 좋아질 수 있는 것은 명백하다. 잘 짜여진 작업수행의 틀에 맞추어 관리하고 팀원들에게도 방법론을 숙지시킨다면 의사소통이 훨씬 수월해져, 프로젝트의 계획과 진척관리, 통제 등이 잘 될 수 밖에 없다.  프로젝트 경험의 축적경험의 축적이라는 것은 인적 자원을 잘 관리하고 작업 성과들을 잘 모아두는 정도가 아마 전부라고 생각될지도 모른다. 그러나, 이런 원시적인 방법으로 경쟁력을 달성할 수 없는 것은 당연하다.  일례로, 선진국 유수의 IS 컨설팅 업체의 방법론을 통한 경험의 축적을 보면 그 까닭을 알 수 있다. 이들은 프로젝트를 수주하고 잘 진행하여 성공적으로 끝마치는 데에도 많은 노력을 기울이지만, 그 후에 그간의 작업을 모으고 협의하여 결과를 방법론에 다시 반영하는 절차를 되풀이 한다. 이것은 회사 전체의 기술력으로 공유되어 기술력의 상승을 가져온다. 다국적 기업이라면 이러한 작업이 각 지역 사이트로부터 모아져 그야말로 엄청난 노하우로 축적되는 것이다.  방법론을 통해 시스템 개발 프로젝트의 경험들을 정연하게 정리하는 것은 가장 합리적인 **경험 축적** 의 수단이 될 수 있다.  시스템 개발 생산성의 향상표준화된 방법론을 통해 팀장(PM) 뿐아니라 팀원들간에도 작업 수행의 틀이 공유되어 불필요한 중복, 불일치, 일관성의 상실, 의견차이 등의 낭비를 해소할 수 있으므로 해서 개발 생산성이 높아진다. 많은 프로젝트 관리자들이 **의사 소통** 문제가 가장 프로젝트를 어려움으로 몰아넣는 요인이라고 말하고 있다. 표준화된 방법론의 존재는 팀원간, 또는 팀간 지식의 공유를 이루도록하여 생산성 향상에 기여한다. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| http://tfile.nate.com/download.asp?FileID=27676529 |  | 패러다임에 따른 방법론의 비교 |
|  |  | 위에서 언급한 상용 방법론들은 정보기술 패러다임 변화에 따라 여러가지 유형의 개발 방법론들을 기반으로 하고 있다. 특히, 소프트웨어공학의 발전사를 보면 크게 3가지의 방법론이 출현하였음을 알 수 있는데, 과거 방법론이 없던 無방법론 시대에서 구조적 분석/설계(또는 방법론), 정보공학, 객체지향 방법론 등이 그것이다   |  |  |  | | --- | --- | --- | | http://tfile.nate.com/download.asp?FileID=27676532 |  | **구조적 방법론(Structured Methodology)** | |  |  | |  | | --- | | 구조적 방법론의 시작은 구조적 프로그래밍의 개념이 탄생하면서 부터이고, 구조적 프로그래밍의 탄생은 소프트웨어공학의 출범을 알리는 것이었다.   1950년대에 시작한 소프트웨어 개발의 역사는 구조적이지 못한 무원칙의 상향식 개발 방식이었다. 개발자의 손이 가는대로 코딩을 해 나갔으며, 로직의 구성 또한 자신만이 알아볼 수 있는 그런 것이었다. 이것은 개발 생산성의 저하와 유지보수의 어려움 등 소프트웨어를 개발하는데 있어서 거의 모든 부분에 문제점을 갖게 만들었고 결국은 “소프트웨어 위기”라는 용어의 탄생을 불러왔다.   1960년대 말, “GO TO” 논쟁, 즉 GO TO 문을 쓰지 말고 구조적으로 프로그래밍을 하자라는 주장이 호응을 얻으면서, 분석과 설계도 구조적으로 하자라는 의견으로 확대되면서 구조적 방법론의 틀이 완성되어 갔다.   구조적 프로그래밍의 개념은 다음과 같다.   * **코드가 계층적인 형식과 제한된 구조로 작성된 순서대로 순차적으로 실행함** * **알고리즘을 기술하는데는 순차(sequencing), 선택(selection), 반복(iteration) 구조면 충분하며, 단일입구/단일출구의 처리구조를 가짐** * **철저한 모듈화로 추상화와 정보은닉을 이루어 프로그램의 구조를 읽기 쉽게 단순화함**   구조적 방법론의 상세한 내용은 본 강좌의 “구조적 분석 및 설계”의 내용을 참조하길 바라며 간단히 그 특징을 정리하면 다음과 같다.   * **데이터 흐름 지향, 즉 프로세스 위주의 분석과 설계 방식** * **모듈의 분할과 정복에 의한 하향식 설계 방식** * **SDLC의 구조를 가진 폭포수 모델이 기본** * **소프트웨어의 개발이 목표인 프로세스와 산출물의 구성** * **데이터의 구성에 대한 설계 방안이 부족** * **프로젝트 관리 및 조직, 역할 등 방법론적 다른 요소들의 정의가 없음**   위의 특성들로 구조적 방법론의 단점을 추정할 수 있는데, 이것은 가장 오래전에 규정된 방법론이며, 단순한 소프트웨어 개발을 목표로 한다는 점에서 기인한 것이다. | | |  |  |  | | http://tfile.nate.com/download.asp?FileID=27676532 |  | **정보공학 방법론(Information Engineering Methodology)** | |  |  | |  | | --- | | 구조적 방법론은 그 후 오랫동안 소프트웨어 개발의 방법론으로 사용되어 왔지만 정보기술의 발전에 따라 새로운 방법론의 필요가 생겼다.   1970년대 이후 소프트웨어는 기업에서도 많이 응용되었고 80년대를 거치면서 경영정보시스템(MIS;Management Information System), 의사결정지원시스템(DSS;Decision Support System), 임원정보시스템(EIS;Executive Information System) 등의 형태로 발전하여 갔고, 급기야 전략정보시스템(SIS;Strategic Information System)의 수준에 이르렀다. 이것은 단지 소프트웨어 뿐 아니라 데이터베이스, 네트워크, 운영조직 등 방대한 주변 정보기술들과 연계되면서 정보시스템(Information System)이라는 개념의 한 요소에 포함되버리고 말았다.   이런 정보시스템은 그 규모도 클 뿐 아니라 구성내용도 복잡해서 개발의 조직도 커지고 개발기간도 몇 년 단위로 늘어났다. 한편으로, 정보기술은 급격히 발전해서 4GT, CASE, 등의 자동화 도구, 다이어그래밍 도구의 등장으로 이것들을 활용할 필요도 생겨났다.   80년대 중반 James Martin에 의해 **정보공학(Information Enfineering)** 의 체계가 정리되면서 정보공학은 본격적으로 활용되었다.   정보공학 방법론의 상세한 내용은 본 강좌의 “정보공학”의 내용을 참조하길 바라며 간단히 그 특징을 정리하면 다음과 같다.   * **소프트웨어공학의 기본적 사상, 즉 하향식, 모듈화, 분할과 정복, 폭포수 등은 동일** * **정보공학은 개별 소프트웨어가 아닌 기업에서 사용되는 정보시스템을 목표로 함** * **단순한 업무지원 시스템의 개발은 의미가 없고 기업의 경영전략을 창출하는 정보전략 계획(ISP; Information Strategy Planning) 작업이 포함됨** * **기업의 업무처리에는 안정된 데이터베이스가 중요하므로 데이터 중심의 분석과 설계를 진행** * **CASE 등 자동화를 요구**   정보공학은 대규모 정보시스템의 개발에 가장 적합한 방법론으로 인정되어 왔으나, 경직되고 복잡한 구조로 인해 많은 단점들이 생겨났고, 때 맞춰 객체지향의 개념이 바람을 타고 정보공학의 영역을 허물어 가고 있다. | | |  |  |  | | http://tfile.nate.com/download.asp?FileID=27676532 |  | **객체지향 방법론(Object Oriented Methodology)** | |  |  | |  | | --- | | 모든 방법론이 그렇듯이 객체지향 방법론도 객체지향 프로그래밍으로부터 출발했다. 추상화와 캡슐화, 상속, 정보은닉 등 객체를 중심으로 프로그래밍 구조를 단순화하고 재사용성을 강화하는 프로그래밍 방식의 유용성이 증명되면서, 분석과 설계도 객체지향의 원칙을 적용하려는 것이 객체지향 분석/설계요, 모든 제반 환경이 결합되어 객체지향 방법론이 된 것이다.   구조적 방법론이나 정보공학 방법론 모두 프로세스와 데이터를 분리하여 처리한다는 단점은 이를 통합하여 처리하는 객체지향의 등장에 가장 큰 배경이 되었다. 프로세스와 데이터가 분됨으로써 이를 분석하여 설계와 개발로 연계시키는데 많은 애로를 겪어야 하며 복잡한 기법들이 활용되어야 한다. 또한 완성된 시스템에 대한 요구사항의 확인 결과를 너무 늦게 확인할 수 있다는 점, 산출물의 양과 복잡도가 크다는 점 등 여러가지 문제를 객체지향 개발 방법론에서 해결하고 있다.   객체지향 방법론의 상세한 내용은 본 강좌의 객체지향에 대한 여러 내용들을 참조하길 바라며 간단히 그 특징을 정리하면 다음과 같다.   * **반복적, 그리고 점증적인(Iterative and Incremental) 개발 방식** * **재사용성의 강조** * **쉽고 표준화된 표기법 존재** * **분산객체 기술의 완벽한 지원** * **OODBMS의 원활한 연계 등 아직 개선의 여지가 많음**   최근에 많은 프로젝트들이 점차 객체지향 방법론으로 진행되고 있으며, 객체지향 모델링 언어의 표준인 UML(Unified Modeling Language)를 기반으로 RUP(Rational Unified Process) 등 상용 방법론들이 등장하여 활용되고 있다. | | |  |  |  | | http://tfile.nate.com/download.asp?FileID=27676532 |  | **방법론간의 비교** | |  |  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 이 외에도 기술의 발전에 따라 RAD(Rapid Appilcation Development) 방법론, 클라이언트/서버(Client/Server) 방법론, 패키지(Package) 개발 방법론, 웹(Web) 개발 방법론 등 여러 방법론 유형들이 나올 수 있지만, 아무래도 위의 3가지가 가장 큰 주류라 하겠다. 또한 최근들어 컴포넌트 기반 개발(CBD;Component Based Development) 방식이 큰 반향을 얻고 있고 있고, 조만간 표준화된 방법론도 등장(이미 여러 방법론들이 나와 있지만)할 것으로 보여 컴포넌트 방법론이 제 4의 비교 대상에 올려야 할지도 모르겠다.   |  |  |  | | --- | --- | --- | | 구조적 방법론 | 정보공학 방법론 | 객체지향 방법론 | | 프로세스 모델링 중심   모듈화가 관건   일부 모듈의 재사용 가능   프로그래밍 기법에 치우침   비정형적 접근방식, 잘 연계되지 않음   소규모 프로젝트 중심   프로그래머 중심   개발자의 능력이 프로그램의 품질을 좌우   호스트 중심, 3GL | 데이터 모델링 중심   엔티티 식별이 관건   데이터의 재사용 가능   기업의 전략 측면 중시, 산출물 중심   구조적인 연계   대규모 프로젝트 중심   분석가 중심   프로그래밍은 자동화될 수 있다는 확신   통합 CASE 중심 | 데이터와 프로세스를 함께 모델링   객체의 식별이 관건   거의 모든 것이 재사용됨   기업의 전략 측면 포함   모든 단계가 Seamless하게 연결   모든 프로젝트에 적합   분석가/설계자/프로그래머와의 협동 중심   Visual Programming과 컴포넌트에 의한 빠른 개발 방식   분산 C/S환경, Round-Trip CASE, 다양한 개발툴 | | | |

|  |  |  |
| --- | --- | --- |
| http://tfile.nate.com/download.asp?FileID=27676529 |  | 방법론의 사례(국내 H사의 경우) |
|  |  | 현재 SI업체에서 많이 사용되고 있는 상용 방법론의 이해를 돕기 위해 시스템 통합 업체인 국내 H사의 정보시스템 개발 방법론인 methodH(가칭)를 일부 소개한다. 방법론의 실제 모습과 내용 구성을 이해하는데 도움이 되리라 생각한다.  방법론은 대개 책자(매뉴얼)가 주된 이용 매체이다. 수권 또는 수십권으로 된 책자를 읽고 참조하여 프로젝트를 수행한다. 어떤 경우는 CD-ROM이나 인트라넷에 정보를 담아 이용할 수 있도록 하기도 한다.   [그림 3]에 보이는 것이 methodH의 전체 구조이다. methodH는, 물론 SDLC 전부를 커버하고 있으며 정보시스템 전략의 결정에서부터 계획, 분석, 설계, 구현, 운영과 관리자의 역할과 작업, 패키지 개발 등의 거의 모든 개발 범위를 포괄하고 있다.  http://tfile.nate.com/download.asp?FileID=27676533  [그림 3] methodH의 구조 |
|  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 그림에서 윗(노란) 부분은 **Base** 시리즈에 해당된다. 여기에는 프로젝트 관리자를 위한 모듈인 **manager** , 정보전략계획에 해당하는 **strategy** , 분석/설계에 해당하는 **designer** , 상세설계와 구현에 해당하는 **builder** , 테스트와 운영에 해당하는 **implementer** 가 포함된다. 그리고 다음 아래의 블럭들은 이 Base 시리즈와 유사하지만 고유 개념을 가지고 있는 파생 방법론으로서, **methodH-IE** 시리즈는 제임스마틴의 정보공학 이론을 적용한 정보공학 방법론이다. 그 아래의 **methodH-C/S** 시리즈는 최근 많은 시스템에 적용되고 있는 개념인 클라이언트 서버 개발의 기술적 환경과 분산 환경 등을 고려한 방법론이다. 그 다음 **methodH-RAD** 시리즈는 Rapid Aplication Development, '신속개발'이라고 하는 ,즉 빠르게 시스템을 개발하는 방법에 관한 방법론이다.   여기서는 이 중에서 정보공학 부분인 **methodH-IE** 에 대해 정리해본다.   |  |  |  | | --- | --- | --- | | http://tfile.nate.com/download.asp?FileID=27676532 |  | **정보공학** | |  |  | |  | | --- | | IE(Information Engineering) 즉, 정보공학이란 **기업 전체 또는 기업의 주요 부문간 정보시스템의 계획, 분석, 설계 및 구축을 위한 정형화된 기술의 집합을 연계하여 응용하는 개념** 을 말한다. 정보공학에는 다음과 같은 구별되는 특징이 있다.   * **기업 중심** - 일반적인 소프트웨어 개발이 아니라 기업이나 다른 조직들의 정보시스템의 통합에 그 촛점이 맞추어져 있다. * **정보전략계획(ISP ; Information Strategy Planning)** - 경영층의 요구를 시스템에 반영하고, 경쟁 우위를 달성하기 위한 시스템을 개발한기 위해 ISP의 수행이 반드시 필요하다. * **데이타 중심** - 기업은 업무활동 보다는 처리하는 데이타에 보다 안정성이 있으며 이의 분석, 설계가 잘되면 시스템의 개발 뿐아니라 유지보수도 훨씬 용이하다는 생각이다. 따라서 데이타의 분석/설계에 보다 치중하는 경향이 있다. * **공학적 접근** - 기본적으로 정보공학은 공학적 개발방법이며 자동화 도구, CASE Tool을 이용하여 코드의 자동생성을 목표로 하고 있다. 따라서 수작업이 훨씬 줄어든다. * **사용자 참여** - 긴 라이프싸이클에도 사용자의 요구사항과 그 변화를 충분히 수용하기 위해 사용자의 적극적인 참여를 강조하고 있다. 매 단계마다 사용자의 확인을 받아야만 다음 단계로 넘어갈 수 있도록 정의되어 있다. * **데이타와 프로세스의 상관분석** - 데이타의 중점을 두어 데이타와 프로세스를 별개의 작업으로 병행 진행한 뒤에는 서로간의 오류를 상관분석을 통해 검증한다. 이에는 CRUD매트릭스를 비롯한 여러 기법들이 동원된다.   methodH-IE도 정보공학에 기반을 두고 있으므로 제임의 마틴의 정보공학 4단계와 유사하게 4개의 큰 절차(모듈)로 구분된다. **PlanningIE** , **AnalysisIE** , **Business Systems ImplementationIE** , **TransitionIE** 등이 그것이다. | | |  |  |  | | http://tfile.nate.com/download.asp?FileID=27676532 |  | **PlanningIE** | |  |  | |  | | --- | | PlanningIE는 정보전략계획 즉, ISP에 해당되는 작업단계이다. 이 단계에서는 내부와 외부의 경영환경을 분석하고, 현행 정보시스템의 조사와 평가를 통해 현안(문제점)을 도출하며, 이의 해결방안과 새로운 시스템 아키텍쳐를 만들어 낸다.  이 단계는 다음 [그림 4]와 같은 단계로 이루어진다.  http://tfile.nate.com/download.asp?FileID=27676534 [그림 4] PlanningIE의 절차 |  |  | | --- | | **Phase 1** 에서는 기업의 경영 및 시스템 전반에 걸친 현황과 경영목표(Goal), 주요성공요인(CSF ; Critical Success Factor), 현안(Issues)을 파악하기 위해 업무분장, 보고서, 각종 장표 등의 경영자료를 분석한다. 또한 경영층 및 관리층의 인터뷰를 수행한다. 이에 따라 주제영역과 업무기능을 정의하는 초기 모델링이 이루어진다. 주요 산출물로는 조직도, 사업장 정의서, 상품 및 서비스 정의서, 산업동향분석, 경제성분석, 경영목표 정의서, 주요성공요인 정의서, 현안 정의서, 업무영역 정의서, 현안 및 기회분석 등이 있다.   **Phase 2** 에서는 현 정보시스템(어플리케이션, 데이타베이스, 네트워크 등의 각종 정보기술들)의 사용수준과 기능적, 기술적 품질을 조사하여 평가하고 문제점들을 찾아낸다. 또한 정보시스템 부문(주로 전산실이 대상) 조직의 역사, 방향, 구조, 기능 등을 평가한다. 주요 산출물로는 현행 어플리케이션/데이타베이스/정보기술 정의서, 현행 어플리케이션/데이타베이스/정보기술 기능 및 기술 평가서, 정보시스템 부문 조직 정의서, 정보시스템 부문 기능 정의서, 정보시스템 부문 기능현안 정의서 등이 있다.   **Phase 3** 에서는 기업의 정보시스템 지원수준에 대해 외부(주로 경쟁업체)와의 비교한 관점을 얻기 위해 벤치마킹을 수행한다. 또한 주요 외부의 시스템 사용자(고객, 공급자, 공공기관 등)의 만족도를 평가하고, 일반적인 정보기술 및 업계에서 전문적으로 사용되는 특정 정보기술에 대해 정리한다. 주요 산출물로는 어플리케이션/데이타베이스/정보기술/ 정보자원 전략 분석, 경쟁사 정보시스템 현안, 업계 정보기술 동향 등이 있다.   **Phase 4** 에서는 지금까지 도출한 요구사항과 현안등을 우선순위에 따라 정리하고 이에 대한 기회요인 및 해결방안을 제시한다. 이를 통해 신규로 개발할 정보시스템의 타당성 및 방향성을 구체화할 수 있게 된다. 그리고 향후 정보시스템에 관련된 전략적 비전을 사용자측과 함께 정의한다. 주요 산출물로는 주요현안 정의서, 정보시스템 기회분석, 정보시스템 비전, 정보시스템 목표 정의서, 정보시스템 주요성공요인 정의서 등이 있 다.   **Phase 5** 에서는 상위 수준의 데이타, 기능 및 프로세스를 정의하고 모델링을 한다. 뿐만 아니라, 이에 따른 논리적인 업무영역을 확정한다. 주요 산출물로는 주제 영역도, 업무기능 정의서, 프로세스 정의서, 프로세스 분할도, 프로세스 의존도, 엔티티/프로세스 관계도, 업무영역 정의서 등이 있다.   **Phase 6** 에서는 데이타, 어플리케이션, 정보기술과 정보시스템 조직의 현행의 상태와 그간의 분석과 모델링을 통해 도출된 신규 모델과의 차이를 식별해낸다. 그리고 이러한 차이를 토대로 구 시스템을 신 시스템으로 효과적으로 연계시킬 수 있는 전이방안 및 신 시스템의 아키텍쳐를 작성한다. 주요 산출물로는 데이타 전이 정의서, 데이타 전이 방안, 어플리케이션 전이 정의서, 어플리 케이션 전이 방안, 정보기술 전이 정의서, 정보기술 전이 방안, 정보시스템 조직 전이 정의서, 정보시스템 조직 전이 방안 등이 있다.   **Phase 7** 에서는 각 업무영역별 개발 접근방안을 만들고 이에 대한 타당성 분석을 통해 업무영역별 프로젝트를 결정하고 논리적 우선순위를 부여한다. 또한 다음 분석/설계 단계를 위한 자원계획을 수립한다. 주요 산출물로는 개발 접근방안 정의서, 정보기술 접근방안 정의서, 정보시스템 조직 접근방안 정의서, 개발 프로젝트 정의서, 정보 기술 프로젝트 정의서, 조직 프로젝트 정의서, 자원 계획서 등이 있다.   **Phase 8** 에서는 다음 분석/설계 단계를 위한 상세한 세부계획을 수립한다. 주요 산출물로는 프로젝트 헌장, 프로젝트 작업계획, 비용 및 효과분석, 비용 계획서 등이 있다.   여기까지가 정보전략계획(ISP) 부분에 해당하는 PlanningIE 부분이다. 이 부분은 전략정보시스템 구축 차원에서 필요한 내용이며, 별도의 프로젝트로 수주하고 수행하기도 한다. 따라서, 이 부분을 생략하고 바로 다음 단계인 분석 및 설계, AnalysisIE를 진행하기도 한다. | | |  |  |  | | http://tfile.nate.com/download.asp?FileID=27676532 |  | **AnalysisIE** | |  |  | |  | | --- | | AnalysisIE는 데이타와 프로세스 모델링을 통해 사용자의 요구사항을 파악하고, 상관분석과 사용자 검토회를 통해 모델을 검증, 보완한다. 뿐만 아니라 정보기술과 환경에 대한 분석, 설계도 실시한다.  이 단계는 다음 [그림 5]와 같은 단계로 이루어진다.  http://tfile.nate.com/download.asp?FileID=27676535 [그림 5] AnalysisIE의 절차 |  |  | | --- | | **Phase 1** 에서는 분석/설계를 수행하기 위한 프로젝트 수행 준비와 계획이 이루어진다. 그리고 ISP를 수행하지 않았을 경우를 대비한 축약된 ISP의 작업들을 거치도록 되어 있다. 물론 ISP를 수행했다면 생략 가능하다.   **Phase 2** 에서는 상세한 데이타 요구사항 분석을 위해 데이타 모델링을 하는 단계이다. 대표적으로 ERD(Entity Relationship Diagram)를 그린다. 데이타 모델링을 하기 위해서는 먼저 정보의 수집이 선행되어야 한다. 즉, 기업의 경영정보를 충분히 이해하기 위해 각종 문서, 보고서, 장표, 업무분장, 시스템 산출물 등을 수집하여 읽고 분석, 정리한다. 또한 현업에 대한 인터뷰와 Workshop, 또는 현장 관찰과 같은 여러 방법이 동원된다. 이러한 작업에 일반적으로 가장 많은 시간과 노력이 투입되게 된다.  이러한 작업이 끝나면, 또는 병행하여 엔티티를 추출하는데, 중요하면서도 관리의 대상인 명사를 중심으로 도출한다. 엔티티는 DB에서 하나의 Table에 해당하는 것으로 보아도 되며, 이 단계에서는 업무적인 관점에 의한 것이라야 한다. 엔티티 도출 후에는 엔티티들과의 관계와 속성에 대한 정의를 하고 이를 ERD의 규칙에 따라 그림으로 그린다.  주요 산출물로는 엔티티 목록, 엔티티 관계도, 엔티티 속성 정의서 등이 있다.   **Phase 3** 에서는 데이타 모델링과 병행하여 프로세스 모델링을 하는 단계이다. 프로세스란 업무 활동으로서, 모델링을 위해 시스템적 관점에서 보면 서브시스템이나 상위 프로그램에 해당하는 동사위주의 단어를 주로 추출한다.  일반적으로 프로세스 모델링은 이러한 기능의 분할이나 의존관계를 표현하는 것으로서 완성할 수 있다. 프로세스의 분할은 프로세스 분할도로, 프로세스간의 의존관계는 프로세스 의존도를 이용하여 표현한다.  흔히 조직도와 기능의 분할도가 거의 비슷한 형태를 띄게 되지만 조직구성과 기능구조가 일치하는 것은 아니다. 물론 프로세스 모델링에 앞서 데이타 모델링처럼 정보의 수집이 선행되어야 할 것이다. 주요 산출물로는 업무기능 정의서, 프로세스 정의서, 단위 프로세스 정의서, 프로세스 분할도, 프로세스 의존도 등이 있다.   **Phase 4** 에서는 데이타와 프로세스의 상관분석이 실행된다. 이 단계의 목표는 이전 단계들에서 만들어진 데이타와 프로세스 모델을 검증하고 수정하는 것과, 초기 로직을 생성해내는 것, 두가지가 있다. 검증 방법은 주로 CRUD 매트릭스를 통해 프로세스가 엔티티를 Create, Read, Update, Delete하는 관계를 파악하여 잘못된 프로세스와 엔티티를 찾아낸다. 또한 DFD(Data Flow Dagram)가 부가적으로 이용되는데, DFD에는 데이타와 프로세스가 함께 그려지기 때문에 둘간의 관계를 검증할 수 있다. 뿐만 아니라 DFD에는 초기 로직이 드러날 수 있으므로 이 후 로직 구성에 도움이 되는 자료가 된다. 이와 함께 액션 다이아그램(Action Diagram)으로 초기 로직을 작성한다. 주요 산출물로는 엔티티/단위프로세스 관계도, 엔티티목록, 엔티티 관계도, 엔티티 속정 정의서, 단위 프로세스 정의서, 프로세스 분할도, 프로세스 의존도, 단위 프로세스 명세서, 데이타 흐름도, 액션 다이아그램 등이 있다.   **Phase 5** 에서는 네트워크를 비롯한 정보기술 부문의 요구사항을 분석하는 단계이다. 프로그램 개발자들보다는 정보기술 전문가들에 의해 별도의 작업으로 진행되는 경우가 많다. 정보기술의 분석/설계는 언제나 데이타와 기능 부문의 분석 또는 설계가 마무리 되어야 이를 토대로 만들어질 수가 있다. 대개 네트워크 구성도나 구성 명세서 같은 기술 위주의 산출물들이 작성된다. 주요 산출물로는 성능 요구사항 정의서, 통신네트워크 요구사항 정의서, 프로토콜 요구사항 정의서, 하드웨어/소프트웨어 요구사항 정의서, 내/외부 설계 제약조건 정의서, 조직 요구사항 정의서, 보안/감사/통제 요구사항 정의서 등이 있다.   **Phase 6** 에서는 분석/설계의 완료 부분으로서 시스템 개발과 정보기술에 관련된 접근방안을 만들어 타당성 평가를 통해 시스템 구현 단계를 위한 시스템 프로젝트를 결정한다. 그리고 구현 단계의 계획과 우선순위를 정의한다. 주요 산출물로는 시스템 정의서, 개발접근방안 정의서, RFI, 정보기술 접근방안 정의서, 정보시스템 조직 접근방 안 정의서, 개발환경 선정안, 비용 및 효과분석, 업무영향 분석, 위험도 분석, 시스템 우선순위 정의서 등이 있다. | | |  |  |  | | http://tfile.nate.com/download.asp?FileID=27676532 |  | **Business Systems ImplementationIE** | |  |  | |  | | --- | | Business Systems ImplementationIE에서는 기능, 데이타베이스, 기술 및 환경에 대한 본격적인 설계에 해당하는 상세설계와 시스템 개발(코딩), 테스트를 수행한다.  이 단계는 다음 [그림 6]과 같은 단계로 이루어진다.  http://tfile.nate.com/download.asp?FileID=27676536 [그림 6] Business Systems ImplementationIE의 절차 |  |  | | --- | | **Phase 1** 에서는 시스템 설계 및 구현을 위한 각종 표준과 계획을 수립한다. 표준에는 프로젝트 수행조직 및 보고체계 등에 대한 표준, 산출물 작성 표준, 설계 표준, 구현 표준, 산출물 및 작업결과 보관 표준 등이 있을 수 있다. 주요 산출물로는 프로젝트 조직 정의서, 산출물 작성 표준, 프로젝트 보고체계 표준, 프로젝트 작업계획, 시스템 설계/구현 표준 등이 있다.   **Phase 2** 에서는 시스템 기능의 상세 설계를 수행한다. 액션 다이아그램을 이용해 상세한 로직을 작성하고 각 모듈을 식별하며, 모듈간의 흐름을 작성하고, 내/외부 인터페이스, 입출력 화면을 설계한다. 주요 산출물로는 액션 다이아그램, 기능 모듈 정의서, 보안/감사/통제 모듈 정의서, 메뉴 구조도, 사용자/어플리케이션 인터페이스 설계서, 입 력 화면 설계서, 출력 화면 설계서, 내부 인터페이스 관계도 등이 있다.   **Phase 3** 에서는 분석 단계의 데이타 모델링을 근거로 물리 데이타베이스 모델링을 시행하는 단계이다. 여기서는 업무관점의 데이타 모델링을 시스템 관점으로, DBMS를 고려한 형태로 수정한다. 주요 산출물로는 이용경로 데이타 모델, 물리 데이타베이스 목록, 물리 데이타베이스 명세서, 물리 데이타 모델 다이아그램, 트리거/저장 프로 시저 정의서, 물리 데이타베이스 스키마 정의서, 데이타 정의 언어 명세서 등이 있다.   **Phase 4** 에서는 정보기술의 Configuration을 설계하는 단계이다. 그리고 그에 따라 정보기술을 도입하여 설치하는 작업까지 포함한다. 주요 산출물로는 네트워크 구조도, 기술배치 설계서, 조직 영향 정의서, 업무절차 정의서, 정보시스템부문 조직 정의서, RFP, 계약서, 정보기술 시험 결과 보고서 등이 있다.   **Phase 5** 에서는 정보시스템을 구현하는데 필요한 나머지 구현요소들을 설계한다. 구 시스템에서 신 시스템으로 변경하는 과정에서 발생할 수 있는 작업들을 준비하고 계획을 세우는 것들이 포함된다. 데이타 변환 전략, 어플리케이션 전환 전략, 조직 전이 전략, 교육전략, 그 밖의 돌발상황에 대비하기 위한 상황관리 전략 등이 포함된다. 주요 산출물로는 데이타/어플리케이션/조직/교육/시험/상황관리 전략 및 계획, 수작업 변환절차 정의서, 변환작업프로그램 목록, 작업흐름 정의서, 작업 프로그램 명세서, 수작업 업무절차 정의서, 사용절차 정의서, 운영절차 정의서, 사용자 매뉴얼 계획서, 교육 목록 등이 있다.   **Phase 6** 에서는 설계에 따라 데이타베이스를 구축하고 프로그램을 개발(코딩)한다. 그리고 단위 시험과 통합 시험을 수행한다. 단위 시험은 개발자 위주의 개별 테스트이고, 통합시험은 전체 시스템들의 연계관계를 위주로 테스트한다. 주요 산출물로는 데이타베이스 시험 계획서, 데이타베이스 시험 결과 보고서, 단위 시험 계획서, 단위 시험 데이타 정의서, 단위 시험 결과 보고서, 통합 시험 계획서, 통합 시험 데이타 정의서, 통합 시험 결과 보고서 등이 있다.   **Phase 7** 에서는 시스템 시험과 인수 시험을 수행한다. 단위 시험은 개별 개발자별 시험이고 통합 시험은 전체 시스템을 결합하여 수행하는 시험이다. 여기까지는 개발 환경인데, 운영환경으로 바뀐 상태에서 통합 시험을 하는 것이 시스템 시험이다. 인수 시험은 사용자가 직접 수행하는 시험이다. 주요 산출물로는 시스템 시험 기준 정의서, 시스템 시험 계획서, 시스템 시험 결과 보고서, 인수 시험 기준 정의서, 인수 시험 계획서, 인수 시험 결과 보고서 등이 있다.   **Phase 8** 에서는 구현을 마무리하고 운영 단계의 계획을 실행한다. 운영 프로젝트를 정의하고 우선순위에 따라 실행계획을 세운다. 주요 산출물로는 운영 우선순위 정의서, 프로젝트 작업계획, 비용 및 효과 분석, 업무영향 분석, 위험도 분석, 업무영역 접근방안 종합 평가서 등이 있다. | | |  |  |  | | http://tfile.nate.com/download.asp?FileID=27676532 |  | **TransitionIE** | |  |  | |  | | --- | | TransitionIE에서는 시스템의 운영 및 그 이후의 작업들에 대해 소개한다.  이 단계는 다음 [그림 7]과 같은 단계로 이루어진다.  http://tfile.nate.com/download.asp?FileID=27676537 [그림 7] TransitionIE의 절차 |  |  | | --- | | **Phase 1** 에서는 완성된 시스템을 가지고 시험 운영을 해보는 단계이다. 시범 시스템을 기존 시스템과 함께 운영해보고 수정하여 전체 신 시스템으로 바꿔 운영하는 방식을 파일럿 운영이라하고, 기존 시스템과 신 시스템을 함께 운영하다가 신 시스템으로 교체하는 방식을 병행 운영이라 한다   **Phase 2** 에서는 최종 시스템을 개통시킨다. 뿐만 아니라 사용자 및 운영자 교육을 실시하고 조직의 변경 부분을 전이시키며, 데이타 변환을 수행한다.   **Phase 3** 에서는 신 시스템의 전반적인 운영과 성능을 평가해보고 수정할 사항이 발생하면 조정에 들어간다. | | | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| http://tfile.nate.com/download.asp?FileID=27676529 |  | 방법론의 현실과 미래 |
|  |  | SI업체 등 대규모 시스템 개발 업체에서 방법론은 이제 필수적인 요소기술로서 다루어지고 있다. 근래에는 사용자의 요구수준도 높아져가고 있어, 방법론을 기반으로 하지 않고는 프로젝트 수주도 어려워지게 되었다.   그러나 국내의 현실은 개발 방법론을 제대로 적용하기 어려운 것이 현실이다.   대부분의 프로젝트가 일정과 비용이 충분하게 주어지지 못하고, 프로젝트 관리자 등 개발팀의 인식이 아직은 프로그램 개발 위주로 되어 있는 까닭이다. 또한, 국내 실정에서 방법론을 자체 개발하기는 어렵고 선진국에서 방법론을 도입하다 보니, 국내실정에 맞지 않는 절차와 양식들이 다수 포함되어 있어, 더더욱 그 적용이 어려운 상황이 되고 만다. 방법론은 정보기술의 변화만큼이나 구성내용의 업그레이드와 피드백이 이루어져야 하지만, 인력이 절대 부족한 국내 개발 업체의 현실상 방법론 구축에 참여한 우수 인력들을 비생산이라 생각되는 방법론 개발 및 유지보수 작업에 오래 묶어둘 수도 없다. 외국의 방법론을 도입한 업체들은 경우에 따라서 방법론을 도입한 인력들 조차 그 내용을 잘 이해하지 못해 사장시켜두는 경우도 많은 것이 사실이다.   제대로 방법론을 적용한다는 것은 국내의 열악한 실정에 미루어 아직은 요원한 것으로 보이지만, 그 노력은 계속되어야 한다. 표준 방법론을 만들고, 지속적으로 유지보수하며, 많은 기술자들이 이용할 수 있도록 전파하고, 또 반복하여 피드백하는 노력이 필요하다. 뿐만 아니라, 유행처럼 나타났다 사라지는 새로운 기술의 신속한 적용을 방법론을 통해서 이룰 수 있도록 해야 한다.   컴포넌트 기반, 객체지향, 인터넷/인트라넷 등의 신개발 방법에 신속히 대처할 수 있는 방법론의 개발이 필요하다. 또한 우리 실정에 보다 잘 맞고 쉽게 적용할 수 있는 방법론의 개발에도 노력을 기울여야 할 것이다. |

출처 : [Tong - ddakzzi님의 Document통](http://tong.nate.com/boxitem/post.do?@idx=3&@base=38912230&@rid=&@sw=1&_tongID=90103&_boxID=1936589&action=clist)

**[출처]** [데이터 주제영역](http://blog.naver.com/samuelc/20107701280) |**작성자** [사무엘](http://blog.naver.com/samuelc)