Prototyping: alternative systems development methodology

by J M Carey

200614163
김은우
2008년 9월 22일

목 차

- Consensus definition
- Rationale for prototyping
- Prototyping environments
- The types of prototyping
- Advantages of prototyping
- Disadvantages of prototyping
- Determination of when to prototype
- Methodology
- Incorporating human-factors guidelines into prototyping
- Two case studies with prototyping
- Summary
- My opinion

Consensus definition (1)

- 일반적인 정의

1. 최상의 시스템 구축 모델에 위한 사용자 요구사항 결정을 위한 전략
2. 개발될 시스템의 모델 또는 표본 설계
3. 총체적 생산 이전에 수행되는 디자인의 정확도 평가도구
4. 개발 주기 내에 사용자를 포함하자는 이상: 전통적인 개발 주기는 초기 단계에서만 사용자와 개발자의 커뮤니케이션이 이루어지고 이후로는 독립적이고 배제된다. 이러한 개념상 차이점의 이유와 타당성에 대해서는 뒷 부분에서 알아본다.

Consensus definition (2)

- 다양한 정의들의 공통점

1. 최종시스템(초기 Prototype을 사용하여 발전된 형태)의 모델: 이는 곧 초기 형태의 포로토타입이 개발되어 발전되고 진화된 형태의 최종 결과물로서의 최종시스템을 말한다.
2. 사용자의 소프트웨어 개발 과정 개입
3. 전통적인 개발 생명주기 방식보다 빠른 Information System 생산: 왜 빠른지에 대해서는 뒷부분에서 알아보도록 한다.

Consensus definition (3)

‘Prototyping’ is the process of quickly building a model of the final software system, which is used primarily as a communication tool to assess and meet the information needs of the user.: 프로토타입핑이란 주로 사용자의 정보에 대한 요구사항을 평가하여 해결하고 만족시키기 위한 커뮤니케이션툴로 사용되는, 최종 소프트웨어 시스템의 한 모델을 빠르게 구축하는 과정이다.

Consensus definition (4)

- 소프트웨어 시스템이나 컴퓨터 하드웨어 시스템을 본격적으로 생산하기 전에 그 타당성의 검증이나 성능 평가를 위해 미리 시험삼아 만들어 보는 모형제작방법.
- 프로토타이핑이란 개발자들과 사용자들의 의사소통상의 효과를 증진시키기 위하여 취하는 시스템개발 기법이다. 프로토타이핑 기법을 수행할 때 중요한 점은 개발자와 사용자 간의 상호이해 및 지식교환을 위한 작업이라는 점을 명심해야 한다. 일반적인 분석방법을 취할 경우 양자간에 서로 다른 이해를 가져올 수 있으므로 프로토타입이라는 의사소통도구를 만들자는 것이다.
- 프로토타이핑은 그 목적에 따라 여러가지 형태가 있을 수 있다. 먼저, 오직 사용자의 요구분석이 목적인 경우 폐기처분용 프로토타입을 만들 수 있고, 둘째 가급적 빨리 개발해야 하는 경우 4GL 등을 써서 개발하는 quick and dirty 프로토타입이 있다. 셋째 상세설계와 구현까지 마친 다음 대량생산에 임하기 앞서 시험용으로 개발된 프로토타입이 있을 수 있다. 넷째로, 입출력의 사례를 보여줄 뿐 실제 데이타도 없고 절차논리도 구현되지 않는 프로토타입이 있으며, 마지막으로 개발된 프로토타입을 계속 진화시켜 나감으로써 최종적인 시스템으로 발전시키는 진화형 프로토타입이 있다.
- 이 프로토타입은 각 단계마다 사용자의 요구를 분석하며 프로토타이핑을 반복하기 때문에 기존의 전통적인 생명주기를 무시하게 된다. 결국 진정한 프로토타입은 바로 진화적 프로토타입이 되어야 할 것이다. 특히 RAD 기법으로 개발할 경우 각 단계의 프로토타입이 실제 완성된 시스템으로 이어져야 한다는 사실은 중요하다.

Rationale(근거, 타당성) for prototyping

전통적인 소프트웨어 개발 방식의 문제점

1. 사용자들의 Information Needs의 이해가 불분명하고 간결하지 못함. (요구사항을 미리 명세하지 못함): 전에 말했듯이 사용자들의 개발과정에 있어서 참여도는 초기 단계에서만 이루어지기 때문에 전반적인 이해도가 부족하게 되고 다소 전문적 지식의 필요성이 있으므로 정확한 의사표현과 요구 명시가 어렵게 됨.
2. 전통적 기능 명세는 기술적이고 시간소비
3. 개발 팀 규모가 커질수록 의사소통의 어려움
4. 배우고 사용함의 어려움
5. 문서화 강조 ->시간소비, 변화(개발환경과 IT분야에서의 발전, 사회적 흐름과 양상에 맞춰나가야함)에 따른 부정확성, 고비용 -> 사용자의 불만 야기

- 위 사항의 문제점 해결을 위한 몇 가지 진보적 기술이나 해결책 요구 => Prototyping

Prototyping environments (1)

- Automated Development Environments
- Prototyping Toolkits – A Collection of Unintegrated tools

Prototyping environments (2)

- Prototyping 사용의 성공적인 접근 방식을 위한 5가지 단계(by Klinger)

1. 개별적으로 응용프로그램 평가, 이익산출
2. 환경 고려, 알맞은 공식화된 Prototyping Life-cycle의 개발과 문서화
3. 적당한 소프트웨어 Tools의 직원 또는 해당자 훈련
4. 소프트웨어 개발 프로세스 관리와 통제 방식 결정
5. 개발 과정에서 End-user의 참여와 훈련 (※ End-user(Consumer) vs Customer)->참고적으로 동물원를 예로 보면 사료와 관련하여 end-user는 코끼리 customer는 사육사..

The generations of computer languages

- 1st Generation
  - Machine Language
- 2nd Generation
  - Assembly Language
- 3rd Generation
  - C, C++, FORTRAN, Java, Pascal, Ada
- 4th Generation
  - SQL, RPG-II
- 5th Generation
  - Used for artificial intelligence and neural networks – primarily researchers

Two types of prototyping (1)

- Type I – Iterative prototyping

1. 일련의 반복적 Upgrades후의 최종시스템으로 Prototype 사용
2. 일반적으로 개발에 4GL Tool을 이용

- Type II – Throwaway prototyping

1. Prototype은 4GL나 Toolkit으로 개발됨
2. 최종생산품은 3GL로 설계된 Prototype을 기반으로 함

Two types of prototyping (2)

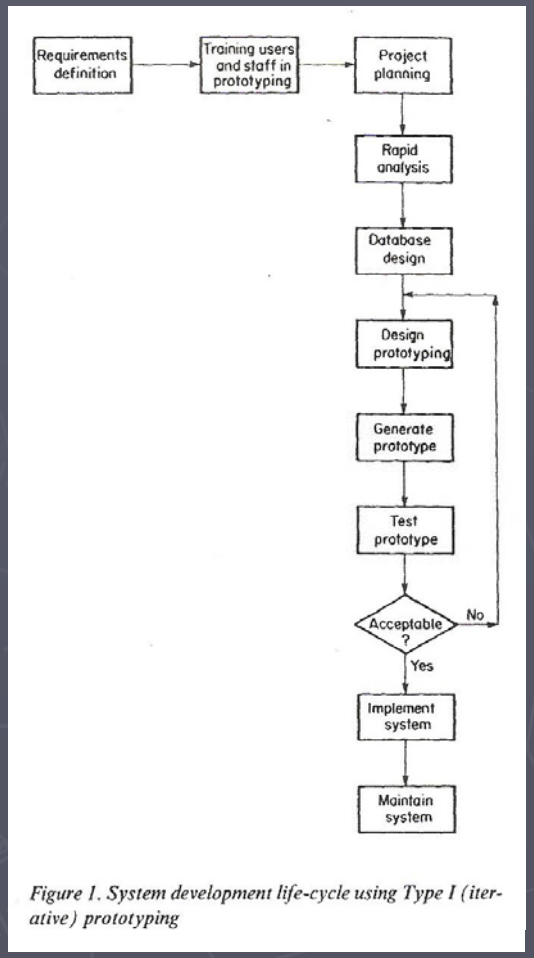

Figure 1. System development life-cycle using Type I (iterative) prototyping

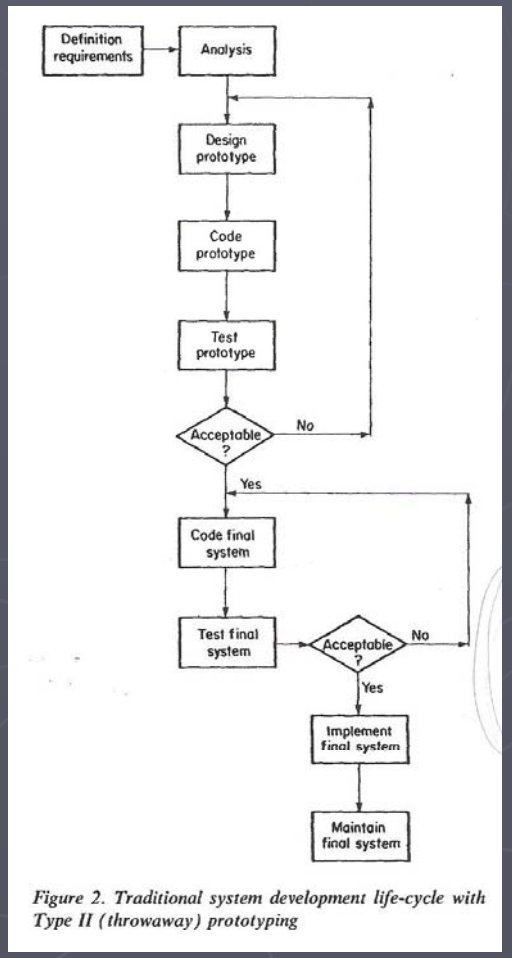

Figure 2. Traditional system development life-cycle with Type II (throwaway) prototyping

Advantages of prototyping

- 보다 빠른 시스템 개발: 사용자와 개발자간의 의사소통은 사용자의 요구사항을 쉽게 파악하고 이해하도록 하므로 개발에 대한 시간효율을 높인다.
- End-users가 시스템의 학습과 사용이 용이: 전반적인 개발과정의 이해와 요구사항의 접목이 있으므로 사용자는 시스템을 쉽게 이용가능.
- 프로그래밍과 분석의 노력이 적게 듦(요구되는 인력 감소)
- Development backlogs이 감소:※backlog: 개발을 기다리는 프로그램인것, 기업의 시스템등에서 경리나 통계업무뿐만 아니라 경영전략적으로 개발이 필요한 프로그램이 해마다 증가하고 있다. 그런데 설계나 개발에 필요한 인적, 기술적 자원이 충분하지 못하여 프로그램 개발이 이루어지지 못하고 미 개발 프로그램이 적체하는 수가 증가함을 일컸는다.
- End-user의 개입 용이
- 사용자의 시스템 운영 및 수행이 쉬워짐
- 사용자/분석자간의 의사소통 향상
- 사용자 요구사항 결정 용이
- 개발 비용 감소
- 시스템의 정확도 향상, 변화의 요구 감소

Disadvantages of prototyping

- 무리한 사용자 기대감: 개발과정에서 사용자는 요구사항에 대해 의사소통가능하고 그에 따라 요구해결의 빈도가 증가할 수 있으며, 자칫 무리한 요구를 행할 우려도 있다.
- Prototype과 최종 결과물 사이의 일관성 문제: 개발과정에 있어서 요구에 대한 변화와 수정이 있으므로 초기 표본에 대한 최종 결과물의 상이가 일어남
- 최종 시스템의 비효율성
- Lack of attention to good human factors practice
- 적당한 디자인과 분석의 무관심

Determination of when to prototype (1)

- A prime candidate for iterative prototyping

1. is dynamic (always changing)
2. is transaction-processing based
3. contains extensive user dialogues
4. is small versus large
5. is well defined
6. is online
7. is the business

Determination of when to prototype (2)

- A bad candidate for iterative prototyping

1. is stable
2. is decision-support based
3. contains much ad hoc retrieval and reporting
4. is of no predictable form
5. is ill defined
6. is batch
7. makes little use of user dialogues
8. is large and complex
9. is real-time
10. does extensive number crunching
11. is 'about' the business rather than directly involved in transaction processing (i.e., decision support and expert systems)

Methodology

- Prototype의 개발과 완성 중 내제된 4단계

1. Determination of key aspects of system to be prototyped
2. Building the prototype
3. Testing the prototype
4. Using the prototype as a model

Incorporating human-factors guidelines into prototyping (1)

※ human-factors: 인간공학, 인간과 관련된 요소에 대한 연구, 인간이 지니고 있는 여러 가지 속성들을 연구하여 이에 맞는 환경을 제공함. 포로토타입핑과 인간공학의 결합

- Prototyping does nothing to ensure adherence to good human-factors guidelines.
- The issue of “user friendly” or usability has become a huge determinant of system success.
- A system that works perfectly may be rejected if difficult to use
- Note that this is not an issue that applies just to prototypes.

Incorporating human-factors guidelines into prototyping (2)

- Some good guidelines

1. Know your users
2. Use selection not entry
3. Make the system behave predictably
4. Make the system as unobtrusive as possible
5. Use display inertia when carrying out user requests
6. Conserve muscle power
7. Use meaningful error messages
8. Allow for reversing of actions

Two case studies with prototyping

- Case 1 (New Jersey Division of Motor Vehicles

1. Information system 설계를 위해 Ideal이라 불리는 새로운 4GL 선택 -> backlog는 엄청난 시간 요구
2. wrong: Ideal은 new and untested, lttle training staff, Ideal은 right tool이 아니라는 조언자들의 의견 무시
3. fix: 800modules 중 58개를 COBOL로 전환하므로서 시스템의 정상 수행을 이룸, 58modules은 시스템의 85%를 차지

- Case 2 (Town and Country Credit Line: TCCL)

1. Banking card system improvement 계획, Texas Instuments로 부터 IFE(Information Engineering Facility) 구매, Case technology 사용에 있어서 경험있는 programmer/consultants를 고용, Iterative prototyping 사용
2. right: Tool은 Ideal보다 더 많은 발전된 것을 이용, 선택하기 전에 이용할 수 있는 tools의 광범위한 연구가 이루어짐, 다른 경우로부터의 실수를 배움

Summary

- Prototyping은 최종시스템 설계의 한 모델로서 계획되며, 사용자와 개발자간의 요구사항 해결의 커뮤니케이션 도구.
- Protoyping은 강력하고 광범위하게 사용되는 시스템 개발 방법, 성공적인 개발의 척도