RECOMMENDED SKILLS AND KNOWLEDGE FOR SOFTWARE ENGINEERS

–By Steve Tockey

컴퓨터공학부

200711422 김부년

Contents

1. Introduction.
2. Computer Science Versus Software Engineering, From First Principles.
3. Recommended Software Engineering skills and knowledge.
    1. Computing Theory
    2. Software Practice
    3. Engineering Economy
    4. Customer and Business Environment
4. Practical Implications.
5. Summary.

1. Introduction

- 컴퓨터 과학에 대한 적절한 기술이나 지식의 구성을 정립해야 하는 이유가 산업에 걸쳐서 있다.
  - the Computing Sciences Accreditation Board has published its “Criteria for Accrediting Programs in Computer Science”
  - A survey of curricula available on the World Wide Web. -> 소프트웨어 엔지니어링을 위한 적절한 지식들이나 기술들을 무엇이 구성하는지에 대한 합의된 바가 거의 없었다.

2. Computer Science Versus Software Engineering, From First Principles.

- Science :

  - a department of systematized knowledge as an object of study; knowledge or system of knowledge covering general truths or the operation of general laws, especially as obtained and tested through scientific method.

- Engineering :

  - the profession in which a knowledge of the mathematical and natural sciences gained study, experience, and practice is applied with judgement to develop way to utilize, economically, the materials and forces of nature for the benefit of mankind.

2. Computer Science Versus Software Engineering, From First Principles.

- Science -> 지식을 추구한다
- Engineering -> 사람에게 이로움을 주기 위해 그 지식을 적용한다.

(For Example)

- Chemistry -> 우리가 관찰 할 수 있는 화학작용에 대한 지식을 좀더 이해하기 편하도록 체계화 하여 정립시켜 놓은 것.
- Chemistry Engineering -> 화학적 지식과 함께 (공학적인)경제의 이해를 동반하는 것.
  - Ex . 압력용기 디자인의 고안, Waste-Heat-Removal 매커니즘.

2. Computer Science Versus Software Engineering, From First Principles.

- 즉, 과학(Science)이라는 가지와 공학(Engineering)이라는 기술적 분야의 가지는 관련이 있지만 구분 됨.

  과학이라는 가지는 그 학문에 대한 이론적인 지식을 계속 확장시켜 나가는 것이라면 공학이라는 가지는 그와 같은 이론적 지식을 실용적이고 경제적으로 적용한 것.

- Engineering = Scientific theory + Practice
  + (Engineering) Economy

2. Computer Science Versus Software Engineering, From First Principles.

- Computer science :

  a department of systematized knowledge about computing as an object of study; a system of knowledge covering general truths or the operation of general laws of computing especially as obtained and tested through scientific method.

- Software Engineering :

  the profession in which a knowledge of the mathematical and computing sciences gained by study, experience, and practice is applied with judgement to develop ways to utilize, economically, computing systems for the benefit of mankind.

2. Computer Science Versus Software Engineering, From First Principles.

- Software Engineering = Computing theory + Practice + (Engineering) Economy

3. Recommended Software Engineering skills and knowledge.

- 사전적인 의미에서의 "Skill"과 "Knowledge"
  - Skill : a learned power of doing something competently; a developed aptitude or ability
  - Knowledge : facts or ideas acquired by study, investigation, observation, or experience.

※ 누구도 모든 엔지니어가 "이상적" 인 위치까지 기대하지 않는다!!

3.1 Computing Theory.

- Knowledge of computing theory allows software engineers to:
  - Propose a larger number of diverse designs than would otherwise be possible.
  - Identify and discard proposed design that could not work (because they violate some known theory) earlier than otherwise possible.

3.1 Computing Theory.

Recommended computing theory skills and knowledge

- Programming language concepts
- Data structure concepts
- Database system concepts
- Relational Algebra
- Operation system concepts
- Software architectures
- Computer architectures
- Automata theory and Petri nets

- Computability theory and Turing machine theory
- Complexity theory
- Linguistics and parsing theory
- Computer graphics
- Set theory
- Predicate logic
- Formal proofs
- Induction

3.2 Software Practice

Recommended software product engineering skills and knowledge

Requirements, analysis, and equirements engineering
Software design
Code optimization and semantics preserving transformations
Human-computer interaction, and usability engineering
Specific programming language
Debugging techniques
Software-software and software-hardware integration
Product family engineering techniques and reuse techniques
CASE/CASE tools

3.2 Software Practice

Recommended software quality assurance skills and knowledge

Task kick-offs, previews, and readiness reviews
Peer reviews, inspection, and walk-throughs
Software project audits
Requirements tracing/Quality Function Deployment (QFD)
Software testing techniques
Proofs of correctness
Process definition and process improvement techniques
Statistical process control
Technology innvation

3.2 Software Practice

Recommended software product deployment skills and knowledge

User documentation techniques
Product packaging techniques
System conversion techniques
Customer support techniques
General technology transfer issues

3.2 Software Practice

Recommended software engineering management skills and knowledge

- Risk assessment and risk management
- Project planning
- Alternative software lifecycles
- Organizational structures
- Organizational behavior
- Project tracking and oversight
- Cost management, schedule management, and resource management
- Metrics, goal–question–metric paradigm, and measurement theory
- Configuration management and change management
- Supplier and subcontract management
- Effective meeting skills
- Effective communication skills
- Negotiation skills

3.3 Engineering Economy

- Economy : thrifty and efficient use of resources.

- Engineering economy is applied microeconomics, where the fundamental question is, “Is it in the best interest of the enterprise to invest its limited resources in a proposed technical endeavor, or would the same investment produce a higher return elsewhere?”

- Business적인 관점에서 Engineering 의 최후의 목표는 최소 비용으로 최대의 이익을 창출하는 것이다.

3.3 Engineering Economy

- Leon Levy.
  - Software economics has often been misconceives as the mean of estimating the cost of programming projects. But economics is primarily a science of choice, and software economics should provide methods and models for analyzing the choices that software projects must make.
  - In any software project there is always a balance between short term and long term concerns…economic methods can help us make enlightened choices.

3.3 Engineering Economy

Recommended engineering economy skills and knowledge

Time value of money (interest)
Economic equivalence
Inflation
Depreciation
Income taxes
Decision making among alternatives
Decision making under risk and uncertainty
Evaluating public alternatives
Evaluating public activities
Breakeven
Optimization

3.4 Customer and Business Environment

- Who is the customer and what is their business?
- What do they use our products and services for?
- When, where, and why are our products and services used?
- Are our products and services being used in a way different than originally intended? If so, why?
- How do our products and services affect the customers' business?
- What external restrictions or regulations impact the ability to deliver products and services to the customer(s)?

3.4 Customer and Business Environment

Recommended engineering economy skills and knowledge

Customer satisfaction assessment techniques
Competitive benchmarking techniques
Technical communication
Intellectual property law
Ethics and professionalism

4. Practical implications

- 질적으로 나은 사람을 제공하는 것은 소프트웨어 학위 프로그램의 첫 목표가 되어야 한다.
- 이러한 지식과 기술을 배우기 위해선 정형화된 소프트웨어 엔지니어링 학위 커리큘럼의 기초 형식이 요구된다.
- 이러한 간단한 바람으로부터의 조언을 지적한 과정으로 바꾸기 위해서 산업과 대학이 연동하는 포럼 같은 것을 해야한다.

5. Summary

The end.

- Thanks -