Software Engineering Principles

Fall, 2021

jehong@chungbuk.ac.kr

2021년 채용의 변화 (1/2)

비대면 채용의 보편화

기존에도 AI면접(AI역량검사)과 화상면접 등이 시행되고 있었으나, 코로나19의 발생 이후로 많은 기업이 채용 과정에 비대면 방식을 도입하고 있습니다. 기업은 온라인 테스트 전형과 화상면접, AI면접(AI역량검사) 등 비대면 방식을 활용하고 있으며, 이러한 비대면 채용 방식은 코로나19의 감염 예방뿐 아니라 공정성을 강화하고 지원자의 역량을 더욱 정확하게 평가할 수 있어 계속해서 활성화될 것으로 전망합니다.

NEWS **AI 필기시험에 화상면접… 유통가 하반기 '비대면' 채용 대세**

식품 · 유통업계가 하반기 공개채용에 돌입했다. 코로나바이러스 감염증(코로나19) 장기화로 취업 시장이 얼어붙은 가운데 채용에 나선 기업들은 채용 과정에 비대면 방식을 속속 도입하고 있다. 2020.09.23. 조선비즈

디지털 인재 채용 확대

4차 산업혁명과 코로나19의 발생으로 디지털 시대로의 전환이 빠르게 진행되고 있습니다. 다양한 분야에 신기술이 접목되어 우리의 생활은 더욱 편리해졌습니다. 정부에서는 4차 산업혁명 도래에 따라 신기술 분야 인재 양성과 신기술 개발을 추진하고 있으며, 이는 코로나19의 발생 이후 비대면 · 디지털 사회에 대비가 필요한 상황이 되면서 그 속도가 더욱 빨라졌습니다. 이러한 상황에 맞춰 기업에서도 디지털 분야의 채용을 확대하고 인재 확보를 위해 노력하고 있습니다.

NEWS **은행권, 디지털 인재 채용 '활발'…"경력자모십니다"**

은행들이 경력직 인재를 향해 연말 채용 문을 활짝 열었다. 특히 비대면 거래 증가와 함께 몸값이 훌쩍 뛴 디지털 분야 인재 채용에 집중하는 모습이다. 은행권 전반에 인원 감축의 칼바람이 불고 있으나 한편에서는 '긴급 수혈이 이뤄지고 있는 것. 금융시장의 변화에 따른 경영 효율화와 디지털 금융으로의 전환에 속도를 맞추기 위한 행보라는 분석이다. 2020.12.17. 오늘경제

2021년 채용의 변화 (2/2)

수시 · 상시 채용의 정착

코로나19의 발생 이후, 채용 제도의 변화에도 영향을 미쳤습니다. 그동안 공채의 축소와 수시 · 상시 채용의 확대가 이어지는 추세였지만 코로나19로 인해 변화의 속도가 급격히 빨라졌습니다. 코로나19의 확산 이후 코로나19 바이러스 확산 방지와 경기 침체로 2020년 상반기와 하반기의 채용 계획이 대부분 연기되거나 수시채용으로 축소 전환되었습니다.

NEWS 딱 맞는 인재 콕 집어 선발...공채 밀어낸 수시채용

경영환경이 급변하면서 채용 방식에도 변화가 일고 있다. 공채를 줄이고 사람이 필요할 때마다 채용하는 수시(상시)채용 방식이 늘어나고 있다. 수시채용은 시기나 절차에 얽매이지 않고 필요 인력이 생기면 그때 그때 공고를 내고 채용방는 방식이다. 지난해 현대차그룹이 정기 공채를 폐지한 데 이어 SK그룹과 KT 등 주요 대기업이 수시채용을 도입하고 있다. 2020.08.29. 중앙선데이

높아진 인턴 경쟁

인턴에는 인턴 기간 후 채용 기회를 주는 채용형 인턴과 인턴 기간 만료 후에 채용으로 연계되지는 않지만 청년들의 단기 일자리 확대와 직장체험을 위해 활용되는 체험형 인턴으로 나뉩니다. 채용 시장에서 구직자들의 직무 관련 경험과 역량을 집중적으로 평가하면서 신입 구직자에게 인턴 경험은 직무를 체험할 수 있어 더욱 중요해지고 있습니다. 또한, 코로나19의 발생 이후 대졸 신입공채가 예년 대비 줄어들고, 대신 필요한 직무에 대해 경력자와 채용형 인턴을 중심으로 수시채용하는 비율이 높아져 인턴 지원자들의 경쟁은 갈수록 치열해지고 있습니다.

채용형 인턴 시행 기업 예시채용형 인턴 시행 기업

KB증권	6주간 현업부서 실습에서 우수한 평가를 받은 인턴에게 신입공채 시 서류 · 필기시험 면제
KT	마케팅&세일즈, 네트워크, IT(정보기술), R&D(연구개발) 총 4개 모집분야 인턴십 수료자 중 임원 면접 통과 시 신입사원으로 채용
네이버 비즈니스 플랫폼	6주간 인턴십 후 우수 평가를 받은 지원자를 신입사원으로 채용

2021년 채용트렌드 인재상의 변화 (코로나19 이후)

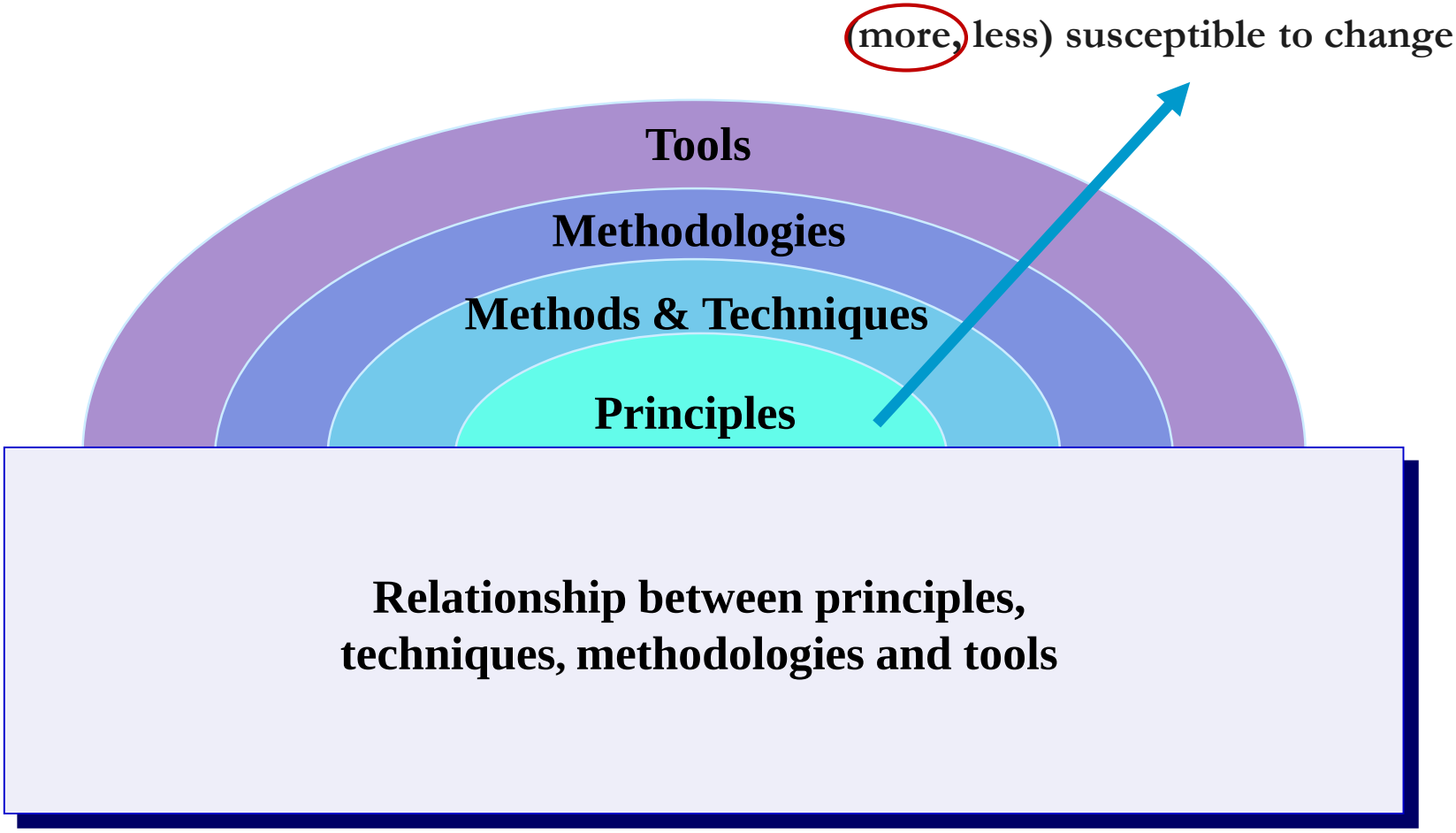
Topics Covered

7 Peinciples

- Rigor and formality
- Separation of concerns
- Modularity
- Abstraction
- Anticipation of changes
- Generality
- Incrementality

Case Study

Software Engineering Principles



Software Engineering Principles

Principles

- Rigor and formality
- Separation of concerns
- Modularity
- Abstraction
- Anticipation of changes
- Generality
- Incrementality

Principles form the basis of methods, techniques, methodologies and tools

Principles may be used in all phases of software development

Modularity is the cornerstone principle supporting software design

Rigor and Formality

Rigor is a necessary complement to creativity that increases our confidence in our developments.

Formality is rigor at the highest degree.

Application of process

- Rigorous documentation of development steps helps project management and assessment of timeliness.

Application of product

- Mathematical (formal) analysis of program correctness.
- Systematic (rigorous) test data derivation.

Separation of Concerns

A way to deal with inherent complexity.

Various ways in which concerns may be separated.

Application of process

- Go through phases one after the other (as in waterfall)
  - separation of concerns by separating activities with respect to time

Application of product

- Keep product requirements separately
  - functionality
  - performance
  - user interface and usability

May we miss some global optimization due to the separation of concerns?

Modularity

A complex system may be divided into simpler pieces called *modules*.

A system that is composed of modules is called *modular*.

Supports application of 'separation of concerns'

- when dealing with a module we can ignore details of other modules

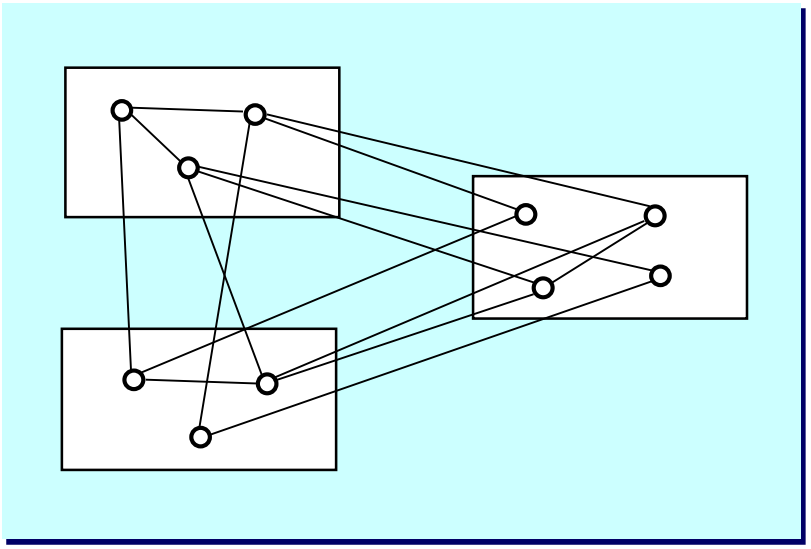
Three goals

- Decomposability
  - Divide and conquer
- Composability
  - Starting bottom up from elementary components
- Capability of understanding
  - Each part of a system separately

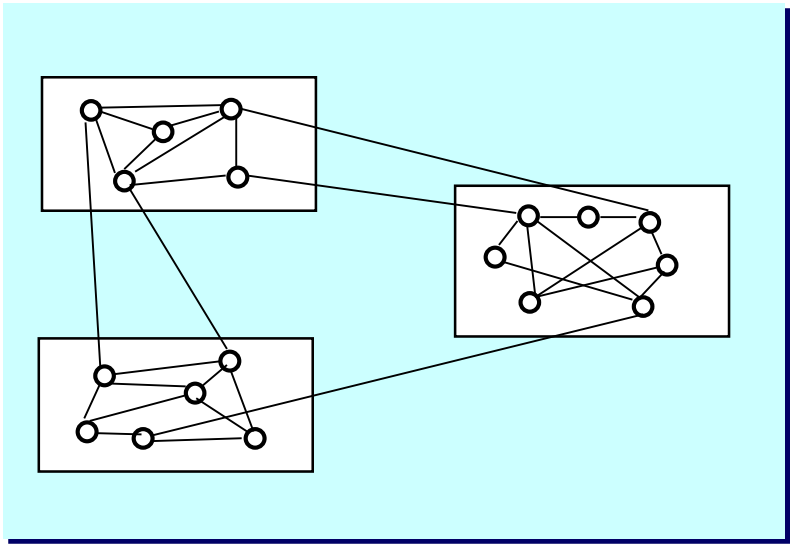
Modularity (Cont.)

To achieve these goals

- High cohesion and low coupling
  - Cohesion: the degree of the relatedness among internal elements of the module.
  - Coupling: the degree of relatedness to other modules



Highly Coupled Structure



High Cohesion and Low Coupling Structure

Abstraction

Identify the important aspects of a phenomenon and ignore its details.

Special case of separation of concerns

History

- Mnemonic
- Macro functions
- Functions and procedures
- Loops, selection
- Abstract data types
- Objects

Helps us concentrate on the problem to solve rather than the way to instruct the machine on how to solve it.

Anticipation of Change

Software undergoes changes constantly.

Likely changes should be isolated in specific portions.

Affects maintainability (evolvability), reusability.

Changes occur in ..

- requirements capturing
- software design
- documentation
- source code
- feedback after delivery

Why "Anticipation of Change" is important in planning phase of software development project ?

Generality

Not necessarily more complex to solve a generalized problem.

Have more potential to be reused.

Already provided by some off-the-shelf packages.

More costly in terms of

- Speed of execution
- Memory requirements
- Development time

Example

- Application server

COTS
Commercial Off-The-Shelf

Incrementality

A process that proceeds in a stepwise fashion.

Evolutionary process

Incremental approach in

- design
- system testing
- verification ...

Examples

- Deliver subsets of a system early to get early feedback from expected users, then add new features incrementally
- Deal first with functionality, then turn to performance
- Deliver a first prototype and then incrementally add effort to turn prototype into product

Case Study in Compiler : Rigor and Formality

Compiler construction is an area where systematic (formal) design methods have been developed

- e.g., BNF for formal description of language syntax
- automata theory

<BNF example>

<statement> ::= <assignment_st> | <call_st> | <compound_st> | <if_st> | ...
<assignment_st> ::= <variable> '=' <exp>
<call_st> ::= <ident> ['(' <exp> { ";" <exp> } ')']
compound_st ::= '{' <statement> {';' [<statement>] } '}'
:

Case Study: Separation of Concerns

Correctness concerns

- producing an object code and appropriate error messages

Efficiency concerns

- compile time efficiency
- run-time efficiency

User-friendliness

- well designed windows and other graphical aids

When designing optimal register allocation algorithms (*runtime efficiency*), no need to worry about runtime diagnostic messages (*user friendliness*)

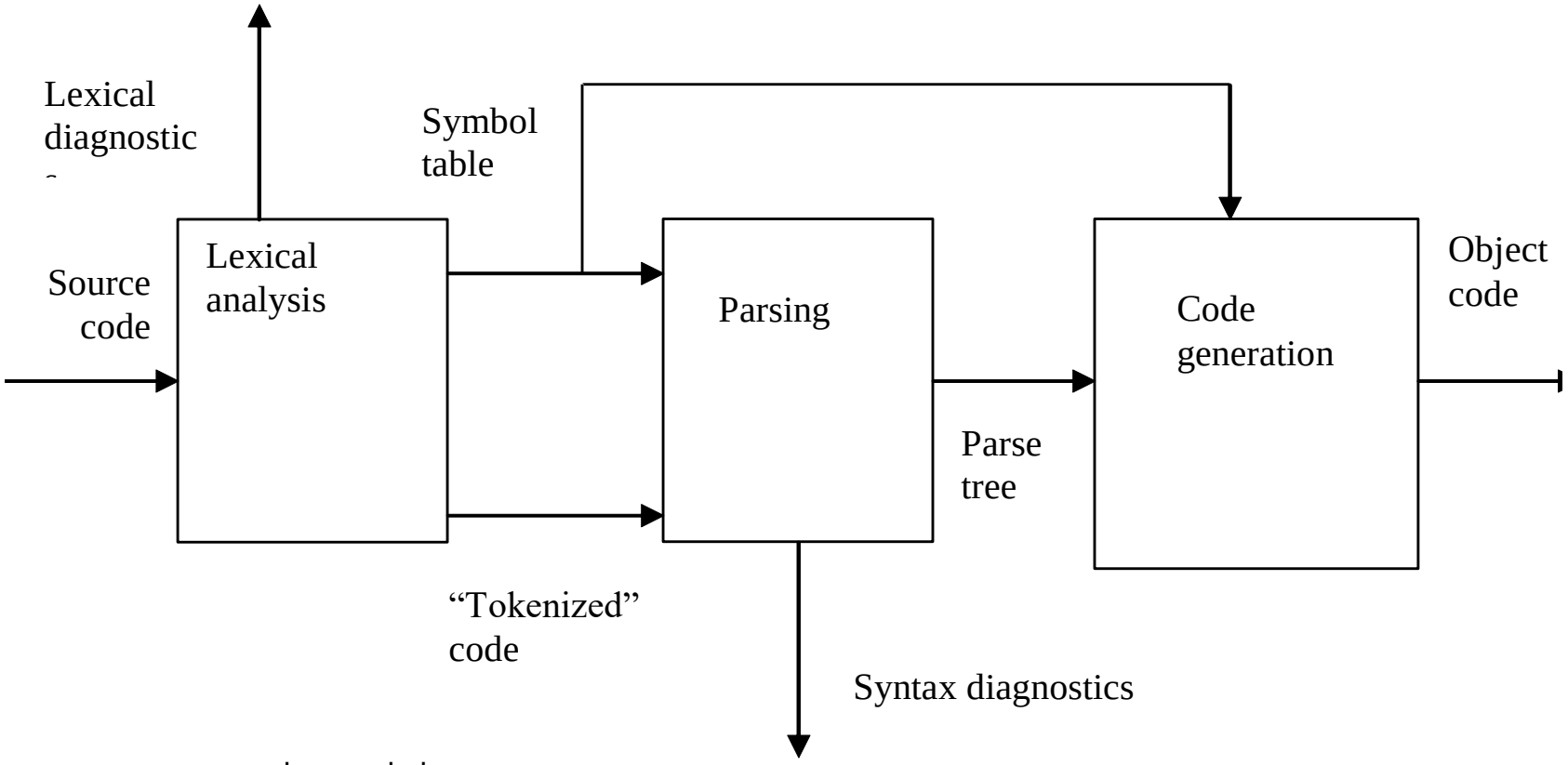
Case Study: Modularity

Compilation process decomposed into phases

- Lexical analysis
- Syntax analysis (parsing)
- Code generation

Phases can be associated with modules

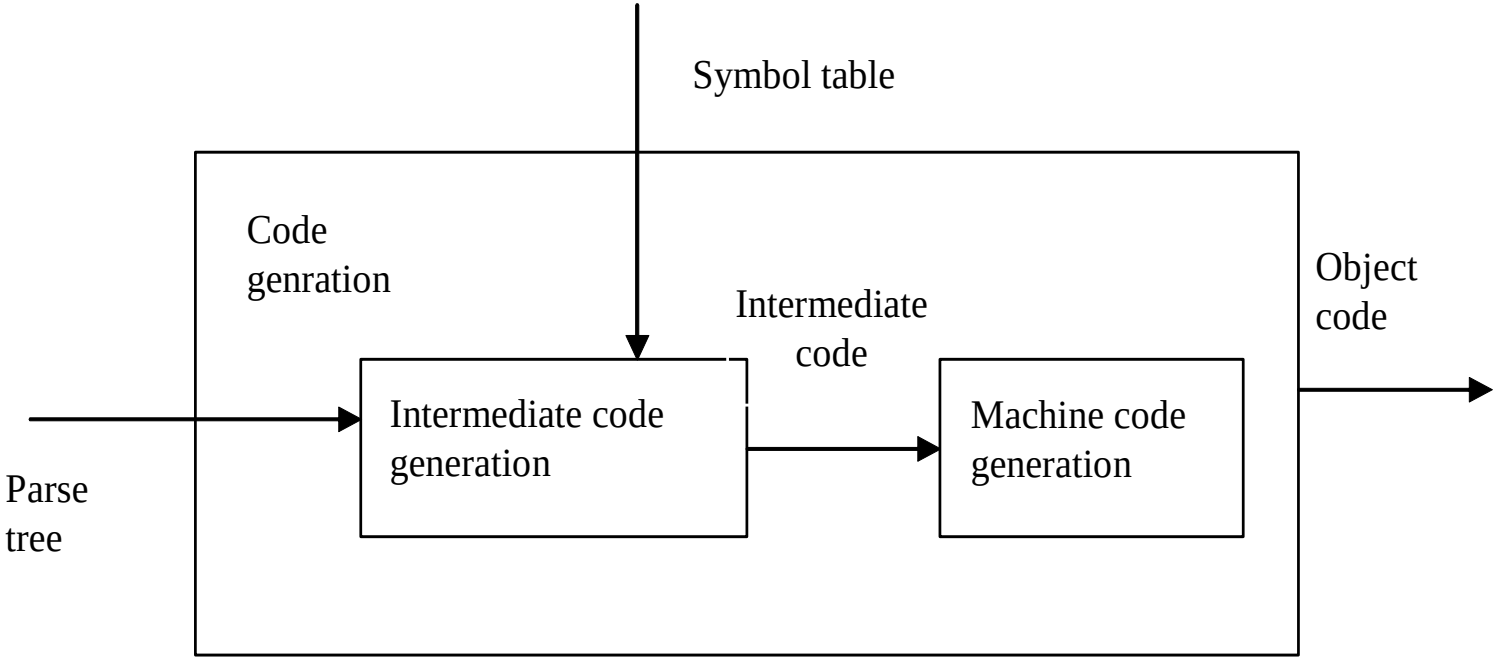
Case Study: Representation of Modular Structure



- boxes represent modules
- directed lines represent interfaces

Module Decomposition may be Iterated

further modularization of code-generation module



Case Study: Abstraction

Applied in many cases

- Abstract syntax to neglect syntactic details such as
  - "begin…end" statement
  - {…} to bracket statement sequences
- Intermediate machine code for code portability
  - e.g., Java Bytecode

What is the abstraction in documentation?

Case Study: Anticipation of Change

Consider possible changes of

- Source language (due to standardization committees)
- Target processor
- I/O devices, I/O statements

Case Study: Generality

Parameterize with respect to target machine (by defining intermediate code)

Parameterize with respect to source language

- Develop compiler generating tools instead of just one compiler
- Compiler compilers

For anticipated change parts, consider generality (parameterize)

Case Study: Incrementality

Incremental development

- deliver first a kernel version for a subset of the source language, then increasingly larger subsets
- deliver compiler with little or no diagnostics/optimizations, then add diagnostics/optimizations

Summary and Discussion

Basic Principles of Software Engineering

- Rigor & formality, Separation of concerns, Modularity, Abstraction, Anticipation of change, Generality, Incrementality.

What is the relation between "Separation of Concerns" and "Anticipation of Change" principles ?

Why modularity principle is cornerstone in software development ?