An Introduction to Computer Science

Jingde Cheng

Southern University of Science and Technology

An Introduction to Computer Science

- Computer Science: What Is It and Why Study It?
- Computation: What Is It and Why Study It?
- Computability
- Computational Complexity (CS101A class only)
- Algorithms
- Data, Information, and Knowledge, and Their Representations
- Data Storage
- Computer Architecture
- Data Manipulation in Computer Systems
- Programming Languages and Compilers
- Operating Systems
- System Software and Application Software
- Software Engineering (CS101A class only)
- Knowledge Engineering and Artificial Intelligence (CS101A class only)
- Information Security Engineering (CS101A class only)

Software Engineering: What Is It and Why Study It ?

Software Engineering: What Is It and Why Study It ?

Software Engineering: What Is It?

- The seminal definition [Friedrich L. Bauer, 1968]
  - [Software Engineering is] "the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines."

History of software engineering

The notion of 'software engineering' was first proposed in 1968 at a conference held to discuss what was then called the 'software crisis' (Naur and Randell, 1969). It became clear that individual approaches to program development did not scale up to large and complex software systems. These were unreliable, cost more than expected, and were delivered late.

Throughout the 1970s and 1980s, a variety of new software engineering techniques and methods were developed, such as structured programming, information hiding and object-oriented development. Tools and standard notations were developed and are now extensively used.

http://www.SoftwareEngineering-9.com/Web/History/

Software: What Is It?

- Software [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "A generic term for those components of a computer system that are intangible rather than physical. It is most commonly used to refer to the programs executed by a computer system as distinct from the physical hardware of that computer system, and to encompass both symbolic and executable forms for such programs."
- Software [IEEE Standard Computer Dictionary, 610, 1991]
  - "Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system."

Software Engineering: What Is It?

- Software Engineering [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "The entire range of activities used to design and develop software, with some connotation of "good practice". Topics encompassed include user requirements elicitation, software requirements definition, architectural and detailed design, program specification, program development using some recognized approach such as …, and the development and use of software engineering environments. Further, software engineering is generally expected to address the practical problems of software development, including those encountered with large or complex systems."

Software Engineering: What Is It?

- Software Engineering [IEEE Standard Computer Dictionary, 610, 1991]
  - "(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1)."
- Software Engineering [B-CS-11]
  - Software Engineering is the branch of computer science that seeks principles to guide the development of large, complex software systems. The problems faced when developing such systems are more than enlarged versions of those problems faced when writing small programs.

Software Reliability

- Reliability [The Oxford English Dictionary, 2nd Edition, 1989]
  - "The quality of being reliable, reliableness."
- Reliability [IEEE Standard Computer Dictionary, 610, 1991]
  - "The ability of a system or component to perform its required functions under stated conditions for a specified period of time."
- Reliability [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "The ability of a computer system to perform its required functions for a given period of time. It is often quoted in terms of percentage of uptime, but may be more usefully expressed as MTBF (mean time between failures.)"

Software Reliability Engineering

- Software Reliability [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "A measure of the extent to which a software system can be expected to deliver usable services when those services are demanded. Software reliability differs considerably from program "correctness". Correctness is the static property that a program is consistent with its specification, while reliability is related to the dynamic demands that are made upon the system and the ability to produce a satisfactory response to those demands."
- Software Reliability Engineering (SRE)
  - To provide principles, methodologies, and tools for designing, developing, operating, and maintaining (function-)reliable software systems.

Software Engineering: What Is It? [SEPA7-10]

QUICK LOOK

What is it? Computer software is the product that software professionals build and then support over the long term. It encompasses programs that execute within a computer of any size and architecture, content that is presented as the computer programs execute, and descriptive information in both hard copy and virtual forms that encompass virtually any electronic media. Software engineering encompasses a process, a collection of methods (practice) and an array of tools that allow professionals to build high-quality computer software.

Who does it? Software engineers build and support software, and virtually everyone in the industrialized world uses it either directly or indirectly.

Why is it important? Software is important because it affects nearly every aspect of our lives and has become pervasive in our commerce, our culture, and our everyday activities. Software engineering is important because it enables us to build complex systems in a timely manner and with high quality.

What are the steps? You build computer software like you build any successful product, by applying an agile, adaptable process that leads to a high-quality result that meets the needs of the people who will use the product. You apply a software engineering approach.

What is the work product? From the point of view of a software engineer, the work product is the set of programs, content (data), and other work products that are computer software. But from the user's viewpoint, the work product is the resultant information that somehow makes the user's world better.

How do I ensure that I've done it right? Read the remainder of this book, select those ideas that are applicable to the software that you build, and apply them to your work.

Software: What Is It? [SEPA7-10]

- Software is both a product and a vehicle that delivers a product
  - Software takes on a dual role. It is a product, and at the same time, the vehicle for delivering a product.
- Software as a product
  - Software delivers the computing potential embodied by computer hardware or more broadly, by a network of computers that are accessible by local hardware.
- Software as a vehicle
  - As the vehicle used to deliver the product, software acts as the basis for the control of the computer (operating systems), the communication of information (networks), and the creation and control of other programs (software tools and environments).

Software: What Is It? [SEPA7-10]

- Software (textbook description)
  - (1) *instructions* (computer *programs*) that when executed provide desired features, functions, and performance, (2) *data structures* that enable the programs to adequately manipulate information, and (3) *documentation* that describes the operation and use of the programs.
- Special characteristics of software (software is a logical rather than a physical system element)
  - Software is developed or engineered, it is not manufactured in the classical sense.
  - Software does not "wear out", but it does deteriorate (due to change).
  - Although the industry is moving toward component-based construction, most software continues to be custom-built.

Software: What Is NOT It?

Software is NOT just programs!

Software Engineering (development) is NOT just programming!

Software Engineers are NOT just programmers!

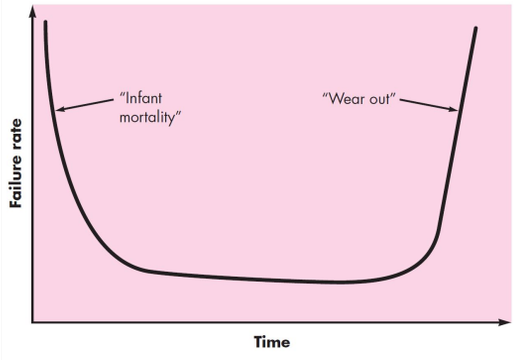
Questions about Software

- **Old questions**
  - Today, the questions that were asked of the lone programmer are the same questions that are asked when modern computer-based systems are built.
- **The questions**
  - Why does it take so long to get software finished?
  - Why are development costs so high?
  - Why can't we find all errors before we give the software to our customers?
  - Why do we spend so much time and effort maintaining existing programs?
  - Why do we continue to have difficulty in measuring progress as software is being developed and maintained?

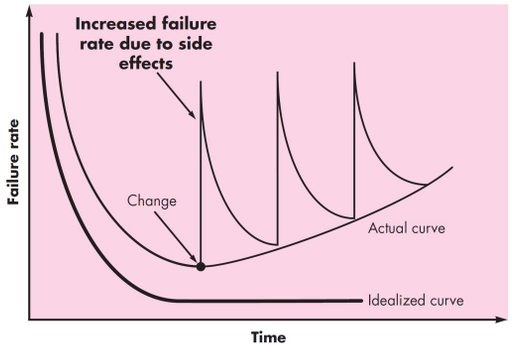
Questions about Software [S-SE-11]

Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable, and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What are the fundamental software engineering activities?	Software specification, software development, software validation, and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
What is the difference between software engineering and system engineering?	System engineering is concerned with all aspects of computer-based systems development including hardware, software, and process engineering. Software engineering is part of this more general process.
What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times, and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs; 40% are testing costs. For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another.
What differences has the Web made to software engineering?	The Web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.

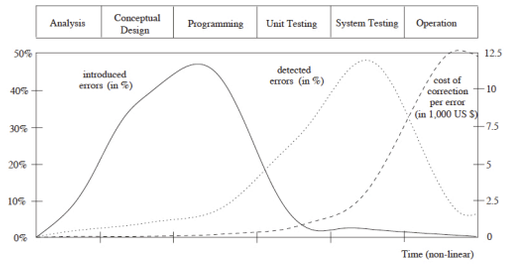
Failure Curve for Hardware [SEPA7-10]



Failure Curve for Software [SEPA7-10]

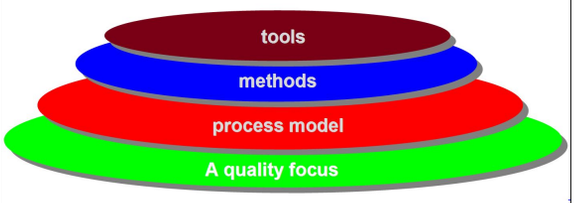


Software Errors [PMC-08]



Software lifecycle and error introduction, detection, and repair costs

Software Engineering: A Layered Technology [SEPA7-10]



Software Engineering: A Layered Technology

- **Quality focus**
  - The bedrock that supports software engineering is a *quality focus*.
- **Process**
  - The foundation for software engineering is the process layer.
  - *Process* defines a framework that must be established for effective delivery of software engineering technology.
  - The software engineering process forms the basis for management control of software projects and establishes the context in which technical methods are applied, work products are produced, milestones are established, quality is ensured, and change is properly managed.

Software Engineering: A Layered Technology

- **Methods**
  - Software engineering *methods* provide the technical how-to's for building software.
  - Software engineering methods rely on a set of basic principles that govern each area of the technology and include modelling activities and other descriptive techniques.
- **Tools**
  - Software engineering *tools* provide automated or semi-automated support for the process and the methods.

Quality Evaluation Criteria in Traditional Software Engineering

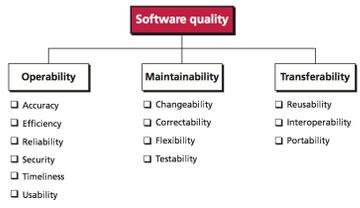
- **The role of quality evaluation criteria**
  - The principles, methodologies, and tools of any engineering discipline are closely dependent on the quality evaluation criteria for the products of that engineering area.
- **Traditional quality evaluation criteria in traditional Software Engineering**
  - Traditional SE is established based on the following main quality evaluation criteria for software systems: *reliability*, *safety*, *efficiency*, *maintainability*, *usability*, *cost/performance ratio*.

Essential Attributes of Good Software [S-SE-11]

Product characteristics	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security, and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable, and compatible with other systems that they use.

Software Quality [F-CS-18]

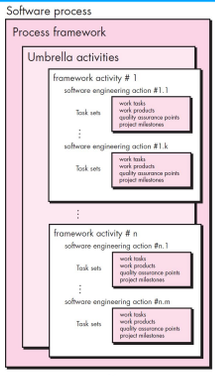
Figure 10.10 *Quality factors*



Software Development Process: What Is It?

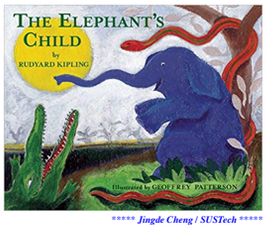
- **Process**
  - A *process* is a collection of activities, actions, and tasks that are performed when some work product is to be created.
  - "A process defines who is doing what when and how to reach a certain goal."
- **5W1H [R. Kipling, The Elephant's Child, 1902]**
  - 5W1H: Who, Why, What, Where, When, How

Software Development Process Framework [SEPA7-10]

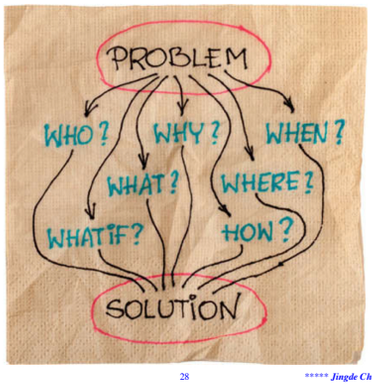


5W1H [R. Kipling, The Elephant's Child, 1902]

- "I keep six honest serving-men.
  (They taught me all I knew);
  Their names are What and Why and When
  And How and Where and Who."
- "私は，６人の正直な召使いを持っている。彼らは，私の知りたいことを何でも教えてくれた。その名前は，What，Why，When，How，Where，Who である。"



6W1H



Software Development Process: What Is It?

- **Activity**
  - An *activity* strives to achieve a broad objective and is applied regardless of the application domain, size of project, complexity of the effort, or degree of rigor with which software engineering is to be applied.
  - Example: communication with stakeholders.

Software Development Process: What Is It?

- **Action**
  - An *action* encompasses a set of tasks that produce a major work product.
  - Action example: architectural design.
  - Work product example: an architectural design model.
- **Task**
  - A *task* focuses on a small, but well-defined objective that produces a tangible outcome.
  - Task example: conducting a unit test.
  - Tangible outcome example: a unit test report.

Software Development Process: What Is It?

- Process framework
  - A *process framework* establishes the foundation focuses on a complete software engineering process by identifying a small number of framework activities that are applicable to all software projects, regardless of their size or complexity.
- Process framework activities
  - A generic process framework for software engineering encompasses five activities: *Communication, Planning, Modeling, Construction, Deployment*.
  - These five generic framework activities can be used during the development of small, simple programs, the creation of large Web applications, and for the engineering of large, complex computer-based systems. The details of the software process will be quite different in each case, but the framework activities remain the same.

Software Development Process: What Is It?

- Process framework activities
  - *Communication*: customer collaboration and requirement gathering
  - *Planning*: establishes engineering work plan, describes technical risks, lists resources required, work products produced, and defines work schedule
  - *Modeling*: creation of models to help developers and customers understand the requires and software design
  - *Construction*: code generation and testing
  - *Deployment*: software delivered for customer evaluation and feedback

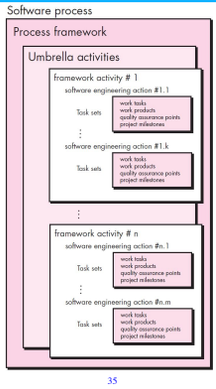
Software Development Process: What Is It?

- Process umbrella activities
  - Software engineering activities are complemented by a number of *umbrella activities*.
  - In general, umbrella activities are applied throughout a software project and help a software team manage and control progress, quality, change, and risk.
  - Typical umbrella activities include: software project tracking and control, risk management, software quality assurance, technical reviews, measurement, software configuration management, reusability management, work product preparation and production.

Software Development Process: What Is It?

- Process umbrella activities
  - Software project tracking and control: allows team to assess progress and take corrective action to maintain schedule
  - Risk management: assess risks that may affect project outcomes or quality
  - Software quality assurance: activities required to maintain software quality
  - Technical reviews: assess engineering work products to uncover and remove errors before they propagate to next activity
  - Measurement: define and collect process, project, and product measures to assist team in delivering software meeting customer needs
  - Software configuration management: manage effects of change
  - Reusability management: defines criteria for work product reuse and establish mechanisms to achieve component reuse
  - Work product preparation and production: activities to create models, documents, logs, forms, lists, etc.

Software Development Process Framework [SEPA7-10]



Software Engineering Practice

- The essence of problem solving [G. Polya, "How to Solve It," 1945]
  - 1. Understand the problem.
  - 2. Plan a solution.
  - 3. Carry out the plan.
  - 4. Examine the result for accuracy.
- The essence of software engineering practice
  - 1. Understand the problem (communication and analysis).
  - 2. Plan a solution (modeling and software design).
  - 3. Carry out the plan (code generation).
  - 4. Examine the result for accuracy (testing and quality assurance).

Essential Questions in Software Engineering Practice

- Understand the problem
  - Who has a stake in the solution to the problem? That is, who are the stakeholders?
  - What are the unknowns? What data, functions, and features are required to properly solve the problem?
  - Can the problem be compartmentalized? Is it possible to represent smaller problems that may be easier to understand?
  - Can the problem be represented graphically? Can an analysis model be created?

Essential Questions in Software Engineering Practice

- Plan the solution
  - Have you seen similar problems before? Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?
  - Has a similar problem been solved? If so, are elements of the solution reusable?
  - Can subproblems be defined? If so, are solutions readily apparent for the subproblems?
  - Can you represent a solution in a manner that leads to effective implementation? Can a design model be created?

Essential Questions in Software Engineering Practice

- Carry out the plan
  - Does the solution conform to the plan? Is source code traceable to the design model?
  - Is each component part of the solution provably correct? Have the design and code been reviewed, or better, have correctness proofs been applied to the algorithm?
- Examine the result
  - Is it possible to test each component part of the solution? Has a reasonable testing strategy been implemented?
  - Does the solution produce results that conform to the data, functions, and features that are required? Has the software been validated against all stakeholder requirements?

The Roles of General Principles of Software Development

- Principle: What is it?
  - Principle [The Oxford English Dictionary, 2nd Edtion] "In generalized sense: A fundamental source from which something proceeds; a primary element, force, or law which produces or determines particular results; the ultimate basis upon which the existence of something depends; cause, in the widest sense.".
- The Roles of General Principles of Software Development
  - "principles help you establish a mind-set for solid software engineering practice. " [SEPA7-10].

General Principles of Software Development [D. Hooker, 1996]

- The first principle: The Reason it all exists
  - A software system exists for one reason: to provide value to its users. All decisions should be made with this in mind.
  - Before beginning a software project, be sure the software has a business purpose and that users perceive value in it.
  - All other principles support this one.
- The second principle: KISS (Keep it simple, stupid!)
  - All design should be as simple as possible, but not simpler.
- Albert Einstein's proverb [Albert Einstein, 1938]
  - "Everything should be made as simple as possible, but not simpler."

General Principles of Software Development [D. Hooker, 1996]

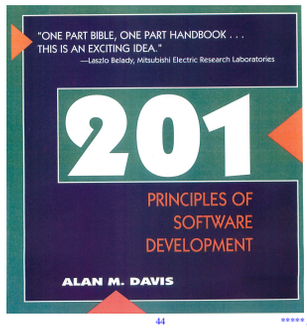
- The third principle: Maintain the vision
  - A clear vision is essential to the success of a software project.
- The fourth principle: What you produce, others will consume
  - Always specify, design, and implement knowing someone else will have to understand what you are doing.
- The fifth principle: Be open to the future
  - Never design yourself into a corner.
  - Always ask "what if," and prepare for all possible answers by creating systems that solve the general problem, not just the specific one.

General Principles of Software Development [D. Hooker, 1996]

- **The sixth principle: Plan ahead for reuse**
  - Planning ahead for reuse reduces the cost and increases the value of both the reusable components and the systems into which they are incorporated.
- **The seventh principle: Think!**
  - Placing clear, complete thought before action almost always produces better results.

201 Principles of Software Development

- A. M. Davis, "201 Principles of Software Development" McGraw-Hill, 1995.



General Principles of Software Development [201P-95]

- Principle 1 Quality Is #1
- Principle 2 Quality Is in the Eyes of the Beholder
- Principle 3 Productivity and Quality Are Inseparable
- Principle 4 High-Quality Software Is Possible
- Principle 5 Don't Try to Retrofit Quality
- Principle 6 Poor Reliability Is Worse Than Poor Efficiency
- Principle 7 Give Products to Customers Early
- Principle 8 Communicate with Customers/Users
- Principle 9 Align Incentives for Developer and Customer
- Principle 10 Plan to Throw One Away

General Principles of Software Development [201P-95]

- Principle 11 Bui1d the Right Kind of Prototype
- Principle 12 Build the Right Features into a Prototype
- Principle 13 Build Throwaway Prototypes Quickly
- Principle 14 Grow Systems Incrementally
- Principle 15 The More Seen, the More Needed
- Principle 16 Change During Development Is Inevitable
- Principle 17 If Possible, Buy Instead of Build
- Principle 18 Build Software So That It Needs a Short Users' Manual
- Principle 19 Every Complex Problem Has a Solution
- Principle 20 Record Your Assumptions

General Principles of Software Development [201P-95]

- Principle 21 Different Languages for Different Phases
- Principle 22 Technique Before Tools
- Principle 23 Use Tools, but Be Realistic
- Principle 24 Give Software Tools to Good Engineers
- Principle 25 CASE Tools Are Expensive
- Principle 26 "Know-When" Is a Important as Know-How
- Principle 27 Stop When You Achieve Your Goal
- Principle 28 Know Formal Methods
- Principle 29 Align Reputation With Organization
- Principle 30 Follow the Lemmings With Care

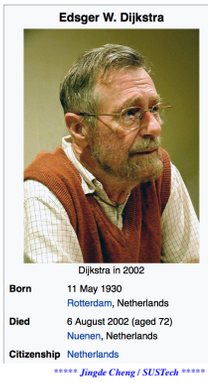
General Principles of Software Development [201P-95]

- Principle 31 Don't Ignore Technology
- Principle 32 Use Documentation Standards
- Principle 33 Every Document Needs a Glossary
- Principle 34 Every Software Document Needs an Index
- Principle 35 Use the Same Name for the Same Concept
- Principle 36 Research-Then-Transfer Doesn't Work
- Principle 37 Take Responsibility

Formal Methods in SE: What Are They and Why Study Them ?

"Program testing can be used to show the presence of bugs, but never to show their absence!"

-- E. W. Dijkstra, "Notes on Structured Programming," 1970.



- **"Today a usual technique is to make a program and then to test it. But: program testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence. The only effective way to raise the confidence level of a program significantly is to give a convincing proof of its correctness."**
- **"But one should not first make the program and then prove its correctness, because then the requirement of providing the proof would only increase the poor programmer's burden. On the contrary: the programmer should let correctness proof and program grows hand in hand."**

-- E. W. Dijkstra, "The Humble Programmer," ACM Turing Award Lecture, 1972.

Because algorithms/programs are logical products, the only correct/effective way to ensure their correctness is the formal verification based on logic!

Ask logic for help!

- **Formal method = Formal specification language + Verification system**
- **"Formal" means "mathematical" or "logical"**
- **Verifying systems are based on logic systems**
- **Note: Programming system = Programming language + Compiler/Interpreter**

"Formal methods should be part of the education of every computer scientist and software engineer, just as the appropriate branch of applied mathematics is a necessary part of the education of all other engineers."

-- FAA (Federal Aviation Authority) and NASA (National Aeronautics and Space Administration), USA

Formal Method: Why Study It ? [B&K-08]

Let us first briefly discuss the role of formal methods. To put it in a nutshell, formal methods can be considered as "the applied mathematics for modeling and analyzing ICT systems". Their aim is to establish system correctness with mathematical rigor. Their great potential has led to an increasing use by engineers of formal methods for the verification of complex software and hardware systems. Besides, formal methods are one of the "highly recommended" verification techniques for software development of safety-critical systems according to, e.g., the best practices standard of the IEC (International Electrotechnical Commission) and standards of the ESA (European Space Agency). The resulting report of an investigation by the FAA (Federal Aviation Authority) and NASA (National Aeronautics and Space Administration) about the use of formal methods concludes that

> *Formal methods should be part of the education of every computer scientist and software engineer, just as the appropriate branch of applied maths is a necessary part of the education of all other engineers.*

Big Challenges in Software Engineering

Big Challenges in Software Engineering

Big Challenges in Software Engineering

- **The fundamentals of SE**
  - **Define the primitive, explicit, measurable, consistent, and complete quality criteria of software systems.**
  - **Establish the logical/mathematical foundation of software systems and SE activities.**
  - **Establish the effective formal methods for SE activities, especially for SE activities of large, complex, high reliable and secure software systems.**
- **Automation and intellectualization of SE**
  - **Develop automated technologies for SE activities.**
  - **Develop intelligent technologies for SE activities.**

An Introduction to Computer Science

- **Computer Science: What Is It and Why Study It?**
- **Computation: What Is It and Why Study It?**
- **Computability**
- **Computational Complexity (CS101A class only)**
- **Algorithms**
- **Data, Information, and Knowledge, and Their Representations**
- **Data Storage**
- **Computer Architecture**
- **Data Manipulation in Computer Systems**
- **Programming Languages and Compilers**
- **Operating Systems**
- **System Software and Application Software**
- **Software Engineering (CS101A class only)**
- **Knowledge Engineering and Artificial Intelligence (CS101A class only)**
- **Information Security Engineering (CS101A class only)**

Knowledge Engineering: What Is It and Why Study It?

Knowledge Engineering: What Is It and Why Study It ?

Knowledge Is Power

"Knowledge is power."

- Francis Bacon, 1598

Knowledge Engineering: What Is It?

- Knowledge Engineering [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "The branch of artificial intelligence that is concerned with building expert systems."
- Knowledge-based system (KBS) [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "A computer system that uses a knowledge base to support reasoning processes in order to solve an application problem. Expert systems are examples, but knowledge-based systems can take many other forms and can be found in many areas of artificial intelligence."

Knowledge Engineering: What Is It?

- Knowledge base [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "A collection of knowledge, usually relevant to a particular application domain, that has been formalized in an appropriate scheme to support reasoning processes. Rule-based formalisms are often used but there are other methods of knowledge representation. Knowledge base are different from databases in that (a) they not only store data but facilitate modification, revision, and other forms of internal manipulation of the knowledge, (b) they are also able to handle knowledge that is incomplete, inconsistent, and uncertain, and (c) they may use imperative as well as declarative forms of knowledge."

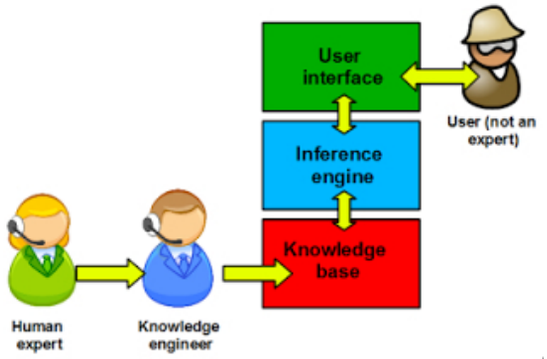
Knowledge Engineering: What Is It?

- Expert systems [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "Computer programs built for commercial application using the programming techniques of artificial intelligence, especially those techniques developed for problem solving. Expert systems have been built for a variety of purposes including medical diagnosis, electronic fault finding, mineral prospecting, and computer-system configuration."
- Knowledge base [IEEE Standard Computer Dictionary, 610, 1991]
  - "A collection of interrelated information, facts, or statements."

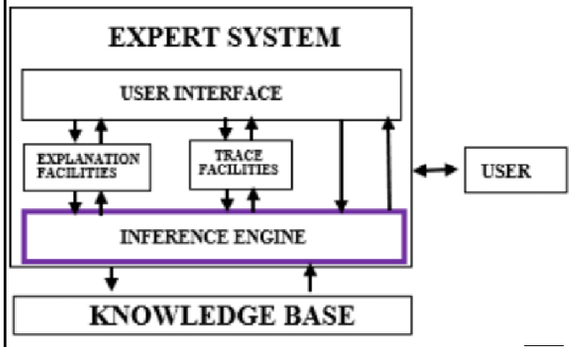
Expert System Examples

- Mycin and Dendral
  - Expert systems were introduced around 1965 by the Stanford Heuristic Programming Project led by Edward Feigenbaum ("father of expert systems").
  - The Stanford researchers tried to identify domains where expertise was highly valued and complex, such as diagnosing infectious diseases ("Mycin") and identifying unknown organic molecules ("Dendral").
- SID
  - The first expert system used in a design capacity for a large-scale product was the SID (Synthesis of Integral Design) system, developed (in LISP) in 1982.
  - SID generated 93% of the VAX 9000 CPU logic gates. Input to SID was a set of rules created by several expert logic designers.

Expert Systems (Knowledge-Based Systems)



Expert Systems (Knowledge-Based Systems)



Knowledge Engineering: Difficult Tasks

- Knowledge acquisition
  - Human experts often cannot describe their knowledge explicitly and formally.
  - The knowledge acquired from human experts may be inconsistent.
- Knowledge representation
  - Various knowledge in various application areas.
  - Various methods and/or tools of knowledge representation.
- Automated deduction/reasoning
  - Validity criteria for automated deduction/reasoning.
  - Efficiency of automated deduction/reasoning engines.

KR&R: Knowledge Representation and Reasoning

- KR&R is at the heart of the great challenge of AI
  - "Knowledge Representation and Reasoning is at the heart of the great challenge of Artificial Intelligence: to understand the nature of intelligence and cognition so well that computers can be made to exhibit human-like abilities."
    -- "Handbook of Knowledge Representation," Elsevier, 2008.
- General methods and/or tools in KR&R
  - Various logics, Various reasoning, Satisfiability (SAT) solvers, Constraint programming, Conceptual graphs, Answer sets, Belief revision, Qualitative modeling, Model-based problem solving, Bayesian networks, Situation calculus, Event calculus, ……

"Logic is the science of sciences, and the art of arts."

"Logic is the science of sciences, and the art of arts."

- John Duns Scotus, 13th century.

"Nothing can be more important than the art of formal reasoning according to true logic."

"Nothing can be more important than the art of formal reasoning according to true logic."

- Gottfried Wilhelm Leibniz

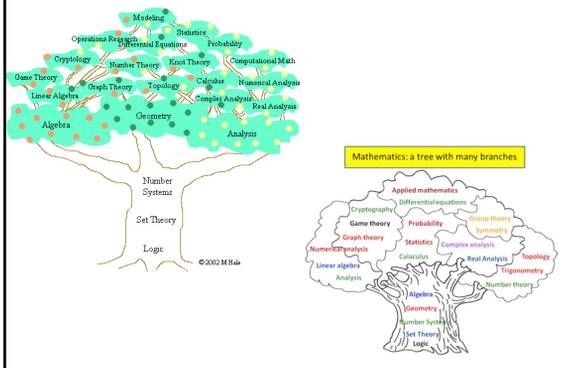
"Logic is the basis for all other sciences"

- "There is a special discipline, called logic, which is considered to be the basis for all other sciences."
  "Logic evolved into an independent science long ago, earlier even than arithmetic and geometry."
  - A. Tarski, 1941.
- "Mathematical Logic, it is a science prior to all others, which contains the ideas and principles underlying all sciences."
  - K. Gödel, 1944.

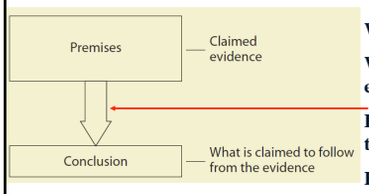
"Fields of Science and Technology" by UNISCO

- "Proposed International Standard Nomenclature for Fields of Science and Technology," UNESCO/ NS/ROU/257 rev.1, 1988.
- 11. Logic, 12. Mathematics
- 21. Astronomy and Astrophysics, 22. Physics, 23. Chemistry, 24. Life Sciences, 25. Earth and Space Science
- 31. Agricultural Sciences, 32. Medical Sciences, 33. Technological Sciences
- 1203. Computer Science
- 3304. Computer Technology

Logic as the Fundamental Basis for all Mathematics



Logic: What Is It?



[From P. J. Hurley, "A Concise Introduction to Logic"]

What entails what?

What follows from what?

Why? What are the evaluation criteria?

How to establish/define the evaluation criteria?

How to evaluate arguments/reasoning?

It is LOGIC to answer these fundamental questions.

Reasoning: An Example

♣ **Reasoning example**

(1) All rational numbers are expressible as a ratio of integers.

(2) π is not expressible as a ratio of integers.

Therefore,

(3) π is not a rational number.

(4) π is a number.

Therefore,

(5) There exists at least one non-rational number.

The Importance of Reasoning Ability

♣ **Reasoning ability as the most fundamental attribute of human intelligence**

- **The great ability to reason, in particular, to reason conceptually, is the most fundamental attribute of human intelligence, and therefore, the most intrinsic difference between human being and animals.**

♣ **The importance of reasoning ability**

- **The ability to reason conceptually, is extremely important in our daily lives because it is the way of getting most of our knowledge.**
- **Most of our knowledge is inferential; it is gained not through direct observation, but by inferring one thing from another.**
- **The ability to reason conceptually, is our only way to predict various dangers for avoiding natural or man-made disasters.**

Reasoning: What Is It?

♣ **Reasoning as a process of drawing new conclusions**

- ***Reasoning* is the *process* of drawing *new conclusions* from given *premises*, which are already known facts or previously assumed hypotheses to provide some *evidence* for the conclusions.**

♣ **Notes**

- **"process", "new conclusions", "premises", "evidence"**

♣ **Reasoning as an ordered process**

- **In general, a reasoning consists of a number of *arguments* (*inferences*) in a certain order, i.e., a reasoning is an ordered process.**

♣ **Notes**

- **"arguments", "inferences", "order"**

Reasoning: What Is It?

♣ **Reasoning as a way to acquire new knowledge**

- **Reasoning is the process of going from what we do know or we assume (the premises) to what we previously did not know (the new conclusions).**

♣ **Reasoning as a way to expand our knowledge**

- **Reasoning is intrinsically *ampliative*, i.e., it has the function of enlarging or extending some things, or adding to what is already known or assumed.**

The Characteristics of Reasoning

- Evidential relation between premises and conclusions
  - The premises of a reasoning are supposed to present evidence for the conclusions of that reasoning.
  - Though the premises of a reasoning are intended to provide some evidence for the conclusions of that reasoning, they need not actually do so.
  - Note: Provided => good, Not provided => bad
- New conclusions
  - The conclusions of a reasoning are supposed to be new to the premises of that reasoning.
  - How to define the notion of 'new' formally and satisfactorily is still a difficult philosophical problem until now.

The Characteristics of Reasoning

- The correctness and/or validity of reasoning
  - Good and bad reasoning (of a reasoning)
  - Correct and incorrect reasoning (of a reasoning)
  - Valid and invalid reasoning (of a reasoning form)
- Fundamental problems about reasoning
  - What is a good, correct, valid reasoning and how we do it?
  - What is the criterion by which one can decide whether or not the premises of a reasoning really provide some evidence for the conclusions of that reasoning?
  - What is the criterion by which one can decide whether or not the conclusion of a reasoning is really new to the premises of that reasoning?
  - What is an argument (or inference)?

Reasoning and Logic

- Where can we find the solutions to the fundamental problems about reasoning?
  - It is logic that deals with the correctness and/or validity of reasoning in a general theory.
  - Note: validity, generality
- Reasoning and logic
  - Logic is primarily about inferring, about reasoning; in particular, it is the study of what constitutes correct reasoning.
  - Logic is the study of the methods and principles used to distinguish good (correct) from bad (incorrect) reasoning.
  - Formal logic deal with "formal reasoning" and/or "reasoning form"

Proving: What Is It?

- Definitions of 'proving' in dictionary [The OED, 2nd Edition]
  - "The action of showing to be true, genuine, or valid; demonstration."
  - "The action of establishing a claim."
- Proving
  - *Proving* is the process of finding a *justification* for an explicitly *specified statement* from given premises, which are already known facts or previously assumed hypotheses.
  - A *proof* is a *description* of a found justification.
- Notes
  - "process", "justification", "specified statement", "given premises"

Proving: What Is It?

- Proving and logic
  - *Classical mathematical logic* (CML) was established in order to provide formal languages for describing the structures with which mathematicians work, and the methods of proof available to them; its principal aim is a precise and adequate understanding of the notion of mathematical proof.
- Logically valid proving
  - A *logically valid proving* is a proving such that it is justified based on some logical validity criterion in order to obtain a correct proof.
  - Note: Any "correctness" must be depend on a certainly defined criterion.

Reasoning and Proving: Intrinsic Difference?

- Nature
  - The most intrinsic difference between reasoning and proving is that the former is intrinsically prescriptive and predictive while the latter is intrinsically descriptive and non-predictive.
- Aim
  - The purpose of reasoning is to find some new statement previously unknown or unrecognized, while the purpose of proving is to find a justification for some statement previously known or assumed.
- Goal (specified statement)
  - Proving has an explicitly specified target as its goal while reasoning does not.

Reasoning and Proving: Intrinsic Difference?

- Typical pattern of reasoning
  - From *A*, *B*, *C*, …, what we can say?
  - Before reasoning, we do not know what conclusion we can draw from the premises.
- Typical pattern of proving
  - From *A*, *B*, *C*, …, can we say *D*?
  - Before proving, we do know what statement we have to justify from the premises.

Big Challenges in Knowledge Engineering

Big Challenges in Knowledge Engineering

Big Challenges in Knowledge Engineering

- The fundamentals of KE
  - Define the primitive, explicit, measurable, consistent, and complete quality criteria of knowledge-based systems.
  - Establish the general logical validity criteria for deduction/ reasoning.
  - Establish the effective formal methods for KE activities.
- Automation of KE
  - Develop automated technologies for KE activities.
  - Develop autonomous evolutionary mechanisms for knowledge-based systems.

Artificial Intelligence: What Is It and Why Study It?

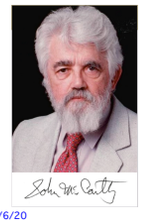
Artificial Intelligence: What Is It and Why Study It ?

Artificial Intelligence: What Is It?

- The seminal definition [John McCarthy, 1955]
  - "We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.
    The following are some aspects of the artificial intelligence problem: 1. Automatic Computers …… 2. How Can a Computer be Programmed to Use a Language …… 3. Neuron Nets …… 4. Theory of the Size of a Calculation …… 5. Self-Improvement …… 6. Abstractions …… 7. Randomness and Creativity …… "
    -- J. McCarthy, M.L. Minsky, N. Rochester, and C.E. Shannon, "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence," August 31, 1955.

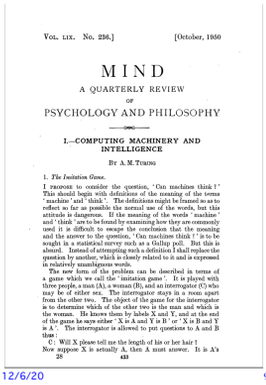
Artificial Intelligence: What Is It?

- The seminal definition [John McCarthy, 2007]
  - "It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable "



Professor John McCarthy

"Computer Machinery and Intelligence" (Turing Test)


Vol. LIX. No. 236.] [October, 1950

MIND

A QUARTERLY REVIEW

OF

PSYCHOLOGY AND PHILOSOPHY

I.—COMPUTING MACHINERY AND INTELLIGENCE

By A. M. Turing

1. *The Imitation Game.*

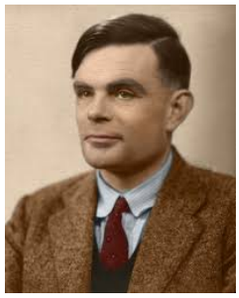
I propose to consider the question, 'Can machines think?' This should begin with definitions of the meaning of the terms 'machine' and 'think'. The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words 'machine' and 'think' are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, 'Can machines think?' is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words.

The new form of the problem can be described in terms of a game which we call the 'imitation game'. It is played with three people, a man (A), a woman (B), and an interrogator (C) who may be of either sex. The interrogator stays in a room apart from the other two. The object of the game for the interrogator is to determine which of the other two is the man and which is the woman. He knows them by labels X and Y, and at the end of the game he says either 'X is A and Y is B' or 'X is B and Y is A'. The interrogator is allowed to put questions to A and B thus:

C: Will X please tell me the length of his or her hair?

Now suppose X is actually A, then A must answer. It is A's object in the game to try and cause C to make the wrong identification. His answer might therefore be:

'My hair is shingled, and the longest strands are about nine inches long.'



"Computer Machinery and Intelligence" (Turing Test)

♣ **The seminal paper [A. M. Turing, 1950]**

- **A. M. Turing, "Computer Machinery and Intelligence," Mind, Vol. LIX, No. 236, pp. 433-460, 1950.**

♣ **The original question [A. M. Turing, 1950]**

- **"I propose to consider the question, 'Can machines think?' This should begin with definitions of the meaning of the terms 'machine' and 'think.' The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous, If the meaning of the words 'machine' and 'think' are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, 'Can machines think?' is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words."**

"Computer Machinery and Intelligence" (Turing Test)

♣ **The imitation game [A. M. Turing, 1950]**

- **"The new form of the problem can be described in terms of a game which we call the 'imitation game.' It is played with three people, a man (A), a woman (B), and an interrogator (C) who may be of either sex. The interrogator stays in a room apart front the other two. The object of the game for the interrogator is to determine which of the other two is the man and which is the woman. He knows them by labels X and Y, and at the end of the game he says either 'X is A and Y is B' or 'X is B and Y is A.' The interrogator is allowed to put questions to A and B thus:**

C: Will X please tell me the length of his or her hair?

Now suppose X is actually A, then A must answer. It is A's object in the game to try and cause C to make the wrong identification. His answer might therefore be:

'My hair is shingled, and the longest strands are about nine inches long.'

In order that tones of voice may not help the interrogator the answers should be written, or better still, typewritten. The ideal arrangement is to have a teleprinter communicating between the two rooms. Alternatively the question and answers can be repeated by an intermediary. The object of the game for the third player (B) is to help the interrogator. The best strategy for her is probably to give truthful answers. She can add such things as 'I am the woman, don't listen to him!" to her answers, but it will avail nothing as the man can make similar remarks.

"Computer Machinery and Intelligence" (Turing Test)

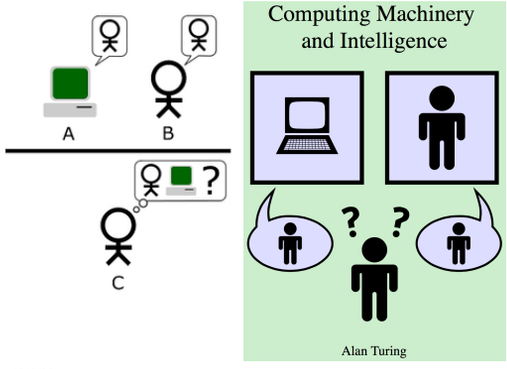
♣ **The "Turing test" question [A. M. Turing, 1950]**

- **"We now ask the question, 'What will happen when a machine takes the part of A in this game?' Will the interrogator decide wrongly as often when the game is played like this as he does when the game is played between a man and a woman? These questions replace our original, 'Can machines think?'"**
- **"We may now consider again the point raised at the end of §3. It was suggested tentatively that the question, 'Can machines think?' should be replaced by 'Are there imaginable digital computers which would do well in the imitation game?' If we wish we can make this superficially more general and ask 'Are there discrete-state machines which would do well?'"**

♣ **The second version of imitation game [A. M. Turing, 1950]**

- **"But in view of the universality property we see that either of these questions is equivalent to this, 'Let us fix our attention on one particular digital computer C. Is it true that by modifying this computer to have an adequate storage, suitably increasing its speed of action, and providing it with an appropriate programme, C can be made to play satisfactorily the part of A in the imitation game, the part of B being taken by a man?'"**

"Computer Machinery and Intelligence" (Turing Test)



"Computer Machinery and Intelligence" (Turing Test)

♣ The Turing's beliefs

- "It will simplify matters for the reader if I explain first my own beliefs in the matter. Consider first the more accurate form of the question. I believe that in about fifty years'time it will be possible, to programme computers, with a storage capacity of about 10^9, to make them play the imitation game so well that an average interrogator will not have more than 70 per cent chance of making the right identification after five minutes of questioning. The original question, "Can machines think?" I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted. I believe further that no useful purpose is served by concealing these beliefs. The popular view that scientists proceed inexorably from well-established fact to well-established fact, never being influenced by any improved conjecture, is quite mistaken. Provided it is made clear which are proved facts and which are conjectures, no harm can result. Conjectures are of great importance since they suggest useful lines of research."

"Computer Machinery and Intelligence" (Turing Test)

♣ The significance of Turing test

- The first proposal for a defined criterion to be used to measure "machine intelligence".

♣ Some fundamental questions

- Is "intelligent behavior" of programs really "machine's intelligence (thinking)"?
- Does Turing test really tested all aspects of human intelligence (thinking)?
- Does "five minutes", "70%" or "30%" really make sense?

♣ The Hao Wang's criticism [H. Wang, 1977, 1981]

- "honest dreams without marginality (诚实而不着边际的梦想)"

-- H. Wang, "Popular Lectures on Mathematical Logic," Science Press and Von Nostrand Reinhold, 1981; "数理逻辑通俗讲话", 科学出版社, 1981; "Popular Lectures on Mathematical Logic (added a postscript)," Dover Publications, 1993.

Artificial Intelligence: What Is It?

♣ Artificial Intelligence (AI) [A Dictionary of Computer Science (7th Edition), OUP, 2016]

- "A discipline concerned with the building of computer programs that perform tasks requiring intelligence when done by human. However, intelligent tasks for which a decision procedure is known (e.g. inverting matrices) are generally excluded, whereas perceptual tasks that might seem not to involve intelligence (e.g. seeing) are generally included. For this reason, AI is better defined by indicating its range. Examples of tasks tackled within AI are: game plying, automated reasoning, machine learning, natural-language understanding, planning, speech understanding, and theorem proving."

Artificial Intelligence: What Is It?

♣ Artificial Intelligence (AI) [A Dictionary of Computer Science (7th Edition), OUP, 2016]

- "Perceptual tasks (e.g. seeing and hearing) have been found to involve much more computation than is apparent from introspection. This computation is unconscious in humans, which has made it hard to simulate. AI has had relatively more success at intellectual tasks (e.g. game playing and theorem proving) than perceptual tasks. Sometimes these computer programs are intended to simulate human behaviour to assist psychologists and neuroscientists. Sometimes they are built to solve problems for technological application."

Artificial Intelligence: What Is It?

♣ Artificial Intelligence (AI) [A Dictionary of Computer Science (7th Edition), OUP, 2016]

- "Both theoretical and applied AI research have made very significant contributions to computer science. Computational techniques that originated from AI include augmented transition networks, means/ends analysis, production rule systems, resolution, semantic networks, and heuristic search."
- "Philosophers have long been interested in the question, 'can a computer think?' There are two schools of thought: weak AI, which is the proposition that computers can at least simulate thought and intelligence; and strong AI, which argues that can perform cognitive tasks is actually thinking. This is a complex topic that has received new interest with a focus on consciousness."

Strong AI vs. Weak AI [J. Searle, 1980]

♣ The seminal paper by J. Searle

- J. Searle, "Minds, Brains, and Programs," Behavioral and Brain Sciences, Volume 3, Issue 3, pp. 417-424, 1980.

♣ The abstract of the seminal paper

- "This article can be viewed as an attempt to explore the consequences of two propositions. (1) Intentionality in human beings (and animals) is a product of causal features of the brain. I assume this is an empirical fact about the actual causal relations between mental processes and brains. It says simply that certain brain processes are sufficient for intentionality. (2) Instantiating a computer program is never by itself a sufficient condition of intentionality."

Strong AI vs. Weak AI [J. Searle, 1980]

♣ The abstract of the seminal paper

- "The form of the argument is to show how a human agent could instantiate the program and still not have the relevant intentionality. These two propositions have the following consequences: (3) The explanation of how the brain produces intentionality cannot be that it does it by instantiating a computer program. This is a strict logical consequence of 1 and 2. (4) Any mechanism capable of producing intentionality must have causal powers equal to those of the brain. This is meant to be a trivial consequence of 1. (5) Any attempt literally to create intentionality artificially (strong AI) could not succeed just by designing programs but would have to duplicate the causal powers of the human brain. This follows from 2 and 4."

Strong AI vs. Weak AI [J. Searle, 1980]

♣ The abstract of the seminal paper

- "'Could a machine think?' On the argument advanced here only a machine could think, and only very special kinds of machines, namely brains and machines with internal causal powers equivalent to those of brains. And that is why strong AI has little to tell us about thinking, since it is not about machines but about programs, and no program by itself is sufficient for thinking."

Strong AI vs. Weak AI [J. Searle, 1980]

♣ J. Searle's definition

- "What psychological and philosophical significance should we attach to recent efforts at computer simulations of human cognitive capacities? In answering this question, I find it useful to distinguish what I will call 'strong' AI from 'weak' or 'cautious' AI (Artificial Intelligence). According to weak AI, the principal value of the computer in the study of the mind is that it gives us a very powerful tool. For example, it enables us to formulate and test hypotheses in a more rigorous and precise fashion. But according to strong AI, the computer is not merely a tool in the study of the mind; rather, the appropriately programmed computer really is a mind, in the sense that computers given the right programs can be literally said to understand and have other cognitive states. In strong AI, because the programmed computer has cognitive states, the programs are not mere tools that enable us to test psychological explanations; rather, the programs are themselves the explanations."

Strong AI vs. Weak AI [J. Searle, 1980]

♣ J. Searle's Chinese room thought experiment

- "One way to test any theory of the mind is to ask oneself what it would be like if my mind actually worked on the principles that the theory says all minds work on."
- "Suppose that I'm locked in a room and given a large batch of Chinese writing. Suppose furthermore (as is indeed the case) that I know no Chinese, either written or spoken, and that I'm not even confident that I could recognize Chinese writing as Chinese writing distinct from, say, Japanese writing or meaningless squiggles. To me, Chinese writing is just so many meaningless squiggles."

Strong AI vs. Weak AI [J. Searle, 1980]

♣ J. Searle's Chinese room thought experiment

- "Now suppose further that after this first batch of Chinese writing I am given a second batch of Chinese script together with a set of rules for correlating the second batch with the first batch. The rules are in English, and I understand these rules as well as any other native speaker of English. They enable me to correlate one set of formal symbols with another set of formal symbols, and all that 'formal' means here is that I can identify the symbols entirely by their shapes."
- "Now suppose also that I am given a third batch of Chinese symbols together with some instructions, again in English, that enable me to correlate elements of this third batch with the first two batches, and these rules instruct me how to give back certain Chinese symbols with certain sorts of shapes in response to certain sorts of shapes given me in the third batch."

Strong AI vs. Weak AI [J. Searle, 1980]

♣ J. Searle's Chinese room thought experiment

- "Unknown to me, the people who are giving me all of these symbols call the first batch 'a script,' they call the second batch a 'story.' and they call the third batch 'questions.' Furthermore, they call the symbols I give them back in response to the third batch 'answers to the questions.' and the set of rules in English that they gave me, they call 'the program.'"
- "Now just to complicate the story a little, imagine that these people also give me stories in English, which I understand, and they then ask me questions in English about these stories, and I give them back answers in English."

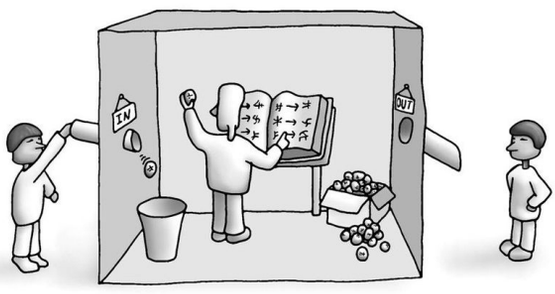
Strong AI vs. Weak AI [J. Searle, 1980]

- J. Searle's Chinese room thought experiment
  - "Suppose also that after a while I get so good at following the instructions for manipulating the Chinese symbols and the programmers get so good at writing the programs that from the external point of view that is, from the point of view of somebody outside the room in which I am locked -- my answers to the questions are absolutely indistinguishable from those of native Chinese speakers. Nobody just looking at my answers can tell that I don't speak a word of Chinese."
  - "Let us also suppose that my answers to the English questions are, as they no doubt would be, indistinguishable from those of other native English speakers, for the simple reason that I am a native English speaker."

Strong AI vs. Weak AI [J. Searle, 1980]

- J. Searle's Chinese room thought experiment
  - "From the external point of view -- from the point of view of someone reading my "answers" -- the answers to the Chinese questions and the English questions are equally good. But in the Chinese case, unlike the English case, I produce the answers by manipulating uninterpreted formal symbols. As far as the Chinese is concerned, I simply behave like a computer; I perform computational operations on formally specified elements. For the purposes of the Chinese, I am simply an instantiation of the computer program."
  - "Now the claims made by strong AI are that the programmed computer understands the stories and that the program in some sense explains human understanding."

J. Searle's Chinese Room Thought Experiment



J. Searle's Chinese Room Thought Experiment

Strong AI and Chinese Room [J. Searle, 1980]

- J. Searle's claims
  - Searle claims that there is no essential difference between the roles of a computer and himself in the Chinese room experiment.
  - Each simply follows a program, step-by-step, producing a behavior which is then interpreted by the user as demonstrating intelligent conversation.
  - However, Searle himself would not be able to understand the conversation. Therefore, he argues, it follows that the computer would not be able to understand the conversation either.
  - Searle claims that, without "understanding", we cannot describe what the machine is doing as "thinking". Therefore, he concludes that "strong AI" is impossible.

Strong AI vs. Weak AI

- Artificial Intelligence (AI) [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "Philosophers have long been interested in the question, 'can a computer think?' There are two schools of thought: weak AI, which is the proposition that computers can at least simulate thought and intelligence; and strong AI, which argues that can perform cognitive tasks is actually thinking. This is a complex topic that has received new interest with a focus on consciousness."
- Strong AI vs. Weak AI [S. Russell and P. Norvig, 2003]
  - **"Strong AI: Can machines really think?"**
    "Weak AI: Can machines act intelligently?"

Artificial Intelligence: Major Areas/Fields

- Artificial Neural Networks
- Automated Deduction/Reasoning
- Automated Planning
- Automated Theorem Proving/Finding
- Computer Games
- Decision Making
- Evolutionary Computing (Genetic Algorithms)
- Intelligent Agent Systems
- Knowledge Representation
- Knowledge-Based Systems

Artificial Intelligence: Major Areas/Fields

- Machine Learning
- Machine Vision
- Natural Language Understanding
- Pattern Recognition
- Robotics
- Speech Understanding

Intelligence Science: What Is It and Why Study It?

Intelligence Science: What Is It and Why Study It ?

Intelligence Science: What Is It and Why Study It?

♣ **Basic Fact 1: AI is still an engineering technology without a sound scientific foundation**

- The current AI has no widely accepted fundamental assumptions, first-principle(s), and unified fundamental theory.
- The current AI majorly focus on finding new technologies for applications but not on discovering new scientific facts, creating new concepts/notions, and proposing new principles.

Intelligence Science: What Is It and Why Study It?

♣ **Basic Fact 2: AI does not take "intelligence" as the central target of scientific researches and engineering practices**

- The current AI does not take "intelligence" as the central target of scientific researches and engineering practices, but just try to find those ways to achieve goals which are regarded as produced by "intelligence".
- Any engineering discipline must be goal-oriented, AI is not an exception (therefore, is still a engineering discipline).

♣ **Basic Fact 3: There is no science that takes "intelligence" as the central target of scientific researches**

- Not only the current AI but also any other science does not take "intelligence" as the central target of scientific researches.

Intelligence Science: What Is It and Why Study It?

♣ **Basic Fact 4: The scientific (logical) foundation of computer technology is computation science (mathematical logic) rather than electronics**

- It is computation science (mathematical logic) that established sound scientific foundation (fundamental assumptions, first-principle(s), and unified fundamental theory) of computer technology.
- In principle, computers are logical machines rather than electronic machines.
- Electronics is just the foundation of one of ways to implementing computers.

Intelligence Science: What Is It and Why Study It?

♣Science vs. technology

- Only scientific creation (i.e., new concepts/notions, new principles, new theories) and innovation (i.e., new ways, new methodologies) can bring about leap-forward technological progress.

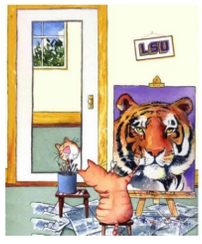
♣Human intelligence vs. animal intelligence

- Human intelligence is by far powerful than animal intelligence.
- There are so many unknown things about human intelligence even animal intelligence.
- Someone claims that computing machines can have "intelligence".

Intelligence Science: What Is It and Why Study It?

♣No science takes "intelligence" as the central target

- Is there a real tiger? Where is a real tiger?
- Tigers are not dragons!



Intelligence Science: What Is It and Why Study It?

♣So what?

- Find a real tiger!
- Investigate tigers at first!

An Introduction to Computer Science

- Computer Science: What Is It and Why Study It?
- Computation: What Is It and Why Study It?
- Computability
- Computational Complexity (CS101A class only)
- Algorithms
- Data, Information, and Knowledge, and Their Representations
- Data Storage
- Computer Architecture
- Data Manipulation in Computer Systems
- Programming Languages and Compilers
- Operating Systems
- System Software and Application Software
- Software Engineering (CS101A class only)
- Knowledge Engineering and Artificial Intelligence (CS101A class only)
- Information Security Engineering (CS101A class only)

Information Security Engineering: What Is It and Why Study It?

Information Security Engineering:

What Is It and Why Study It ?

知彼知己，知天知地

- 知彼知己，勝乃不殆； 知天知地，勝乃可全.

  – 孫子，孫子兵法，第十篇，約紀元前500年.

- 彼れを知りて己れを知れば，勝 乃ち殆うからず.
  地を知りて天を知れば，勝 乃ち全うすべし.

  – 孫子，孫子兵法，第十篇，紀元前500年頃.

- Know the enemy, know yourself,
  your victory will never be endangered;
  Know the ground, know the weather,
  your victory will then be total.

  - Sun Tzu, The Art of War, ch. 10, about 500 B.C.

先発制人，後発制于人

- 先発制人; 後発制于人.

  - 班彪, 班固, 班昭: 『漢書』, 約紀元82年.

- 先発すれば人を制し； 後発すれば人に制せられる.

  - 班彪, 班固, 班昭: 『漢書』, 紀年82年頃.

- Taking the anticipation, you will forestall your enemy; losing the initiative, you will be controlled by your enemy.

  - Ban Biao, Ban Gu, and Ban Zhao, "The Han Shu (The Book of Han)," about A.D. 82.

Information Security Engineering: What Is It?

- Security engineering: What is It? [A-SE2-08]
  - "Security engineering is about building systems to remain dependable in the face of malice, error and mischance. As a discipline, it focuses on the tools, processes and methods needed to design, implement and test complete systems, and to adapt existing systems as their environment evolves."
- Security engineering: Scope [A-SE2-08]
  - "Security engineering requires cross-disciplinary expertise, ranging from cryptography and computer security through hardware tamper-resistance and formal methods to a knowledge of economics, applied psychology, organizations and the law."
  - "System engineering skills, from business process analysis through software engineering to evaluation and testing, are also important; but they are not sufficient, as they deal only with error and mischance rather than malice."

Information Security

- Security [The Oxford English Dictionary, 2nd Edition, 1989]
  - The condition of being secure.
    - The condition of being protected from or not exposed to danger; safety.
    - The safety or safeguarding of (the interests of) a state, organization, person, etc., against danger, esp. from espionage or theft; the exercise of measures to this end; (the maintenance of) secrecy about military movements or diplomatic negotiations; in espionage, the maintenance of cover. Hence (with capital initial), a department (in government service, etc.) charged with ensuring this. (This sense tends towards 'the condition of making secure'.)
    - Freedom from doubt; confidence, assurance. Now chiefly, well-founded confidence, certainty.
    - Freedom from care, anxiety or apprehension; a feeling of safety or freedom from or absence of danger. Formerly often *spec.* (now only *contextually*) culpable absence of anxiety, carelessness.
    - The quality of being securely fixed or attached, stability, fixity.

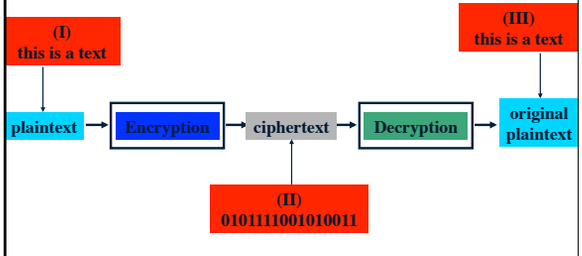
Information Security

- Security [The Oxford English Dictionary, 2nd Edition, 1989]
  - A means of being secure.
    - Something which secures or makes safe; a protection, guard, defence.
    - A means of securing or fixing in position.
    - Ground for regarding something as secure, safe, or certain; an assurance, guarantee.
    - Property deposited or made over, or bonds, recognizances, or the like entered into, by or on behalf of a person in order to secure his fulfilment of an obligation, and forfeitable in the event of non-fulfilment; a pledge, caution.
    - One who pledges himself (or is pledged) for another, a surety.
    - A document held by a creditor as guarantee of his right to payment. Hence, any particular kind of stock, shares, or other form of investment guaranteed by such documents.

Information Security Engineering

- Security [A Dictionary of Computer Science (7th Edition), OUP, 2016]
  - "Prevention of or protection against (a) access to information by unauthorized recipients or (b) intentional but unauthorized destruction or alteration of that information. Security may guard against both unintentional as well as deliberate attempts to access sensitive information, in various combinations according to circumstances. The concepts of security, integrity, and privacy are interlinked."
- Information Security Engineering (ISE)
  - To provide principles, methodologies, and tools for designing, developing, operating, and maintaining secure software systems.

Information and Data in Encryption and Decryption



Software Reliability Engineering vs. Information Security Engineering: What Are Differences ?

Software Reliability Eng. vs. Information Security Eng.

♣ **What is ensured?** [R. J. Anderson, 2001]

- SRE is about ensuring that certain things (i.e., functions required by users) happen.
- ISE is about ensuring that certain things (i.e., information leakage unaccepted by users) do NOT happen.

♣ **Ross John Anderson** (born 15 September 1956)

- He is a researcher, writer, and industry consultant in security engineering, and the author of Security Engineering, the first book on the topic in the world, published by Wiley in 2001.
- He is Professor of security engineering at the Computer Laboratory, University of Cambridge.
- He was elected a Fellow of the Royal Society (FRS) and a Fellow of the Royal Academy of Engineering (FREng) in 2009.

Software Reliability Eng. vs. Information Security Eng.

♣ **The presence of the enemy**

- SRE in general does not face to some enemies who intend to break the reliability of target systems.
- ISE always face to the enemies (active persons acting as assailants/crackers) who intend to break the security of target systems.

♣ **The intrinsic difference between SRE and ISE**

- The presence of the enemy is the most intrinsic difference between SRE and ISE.
- This intrinsic difference leads to many other differences between SRE and ISE.

Software Reliability Eng. vs. Information Security Eng.

♣ **Continuousness**

- The reliability of a target system is not necessarily an object in SRE that must be managed and/or controlled continuously (anytime, all the time).
- The security of a target system is necessarily an object in ISE that must be managed and/or controlled continuously (anytime, all the time).

♣ **Why continuousness?**

- Because assailants (crackers) are active persons who can get knowledge and skills day after day and then continuously attack the weakest parts and/or links in target systems always with the newest techniques, we have to improve the security of a target system continuously (anytime, all the time).

Software Reliability Eng. vs. Information Security Eng.

♣ **Additivity**

- The whole reliability of a target system is usually the sum total of the reliability of its all components.
- The whole security of a target system is not necessarily the sum total of the security of its all components but usually only as good and strong as the weakest security of some component or link between components in the system (**Cannikin Law**, **Cask Effect**, **Weakest Link Effect**).

Software Reliability Eng. vs. Information Security Eng.

♣ **Stability**

- In general, the reliability of a target system is usually stable to its certainly defined requirements in the sense that once the requirements are certainly defined the reliability of the target system is determined by only its source programs that are logical products but has no physical loss dependent on time.
- The security of a target system is usually unstable to its certainly defined requirements in the sense that even if the requirements are certainly defined, the security of the target system is determined by not only its source programs but also the attack ability of assailants that may be growing with time.

Security Concepts and Relationships [ISO/IEC 15408 (CC)]

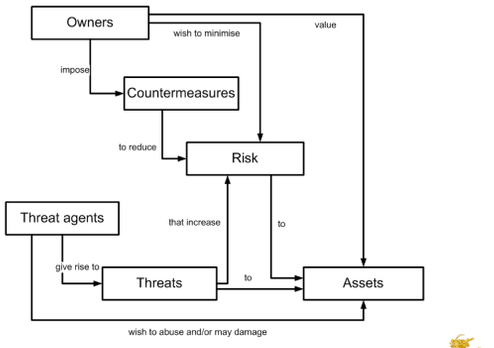


Figure 2 - Security concepts and relationships

Safety

- **Safety [The Oxford English Dictionary, 2nd Edition, 1989]**
  - **"The state of being safe; exemption from hurt or injury; freedom from danger."**
- **Safety [A Dictionary of Computer Science (7th Edition), OUP, 2016]**
  - **"Freedom from risk. The term is also used in the context of safety level to provide a quantitative measure of the level of safety."**
  - **"A safe system is one that will never do anything bad. The definition of what is "bad" is application-depended: the safety requirements for a system controlling an aircraft would obviously be more stringent than those for, say, a stock control system."**

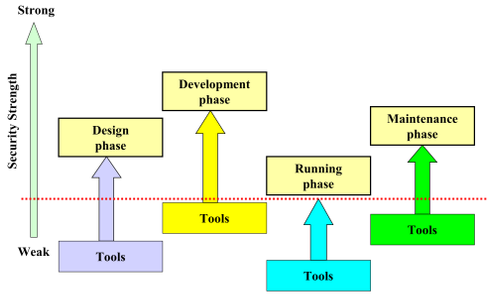
Availability

- **Availability [The Oxford English Dictionary, 2nd Edition, 1989]**
  - **"The quality of being available; capability of being employed or made use of."**
- **Availability [A Dictionary of Computer Science (7th Edition), OUP, 2016]**
  - **"1. The probability that a system will be capable of functioning according to specification at any point in time throughout a stated period of time. 2. The ratio of available time to total time for a system in a given period."**
- **Availability [IEEE Standard Computer Dictionary, 610, 1991]**
  - **"The degree to which a system or component is operational and accessible when required for use. Often expressed as a probability."**

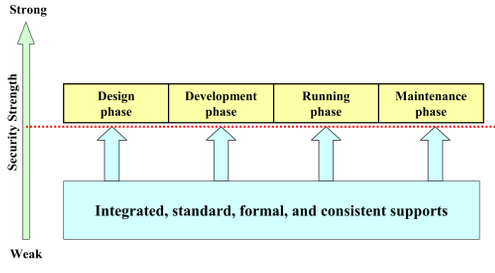
Dependability

- **Dependability [The Oxford English Dictionary, 2nd Edition, 1989]**
  - **"The quality of being dependable, reliability."**
- **Dependability = Reliability + Security [SE2-2008]**
  - **Reliability and security are often strongly correlated in practice**
  - **But malice is different from error!**
  - **Reliability: "Bob will be able to read this file"**
  - **Security: "The XXX Government won't be able to read this file"**
  - **Proving a negative can be much harder …**

Security in Life Cycle of Target Systems (Current)



Security in Life Cycle of Target Systems (Ideal, ISE)



Security Engineering Analysis Framework [A-SE2-08]

- ♣ Good security engineering
  - ◆ Good security engineering requires four things to come together: Policy; Mechanism; Assurance; Incentive.
  - ◆ All of these interact.
- ♣ Policy
  - ◆ What you're supposed to achieve.
- ♣ Mechanism
  - ◆ The ciphers, access controls, hardware tamper-resistance and other machinery that you assemble in order to implement the policy.

Security Engineering Analysis Framework [A-SE2-08]

- ♣ Good security engineering
  - ◆ Good security engineering requires four things to come together: Policy; Mechanism; Assurance; Incentive.
  - ◆ All of these interact.
- ♣ Assurance
  - ◆ The amount of reliance you can place on each particular mechanism.
- ♣ Incentive
  - ◆ The motive that the people guarding and maintaining the system have to do their job properly, and also the motive that the attackers have to try to defeat your policy.

Security Engineering Analysis Framework [A-SE2-08]

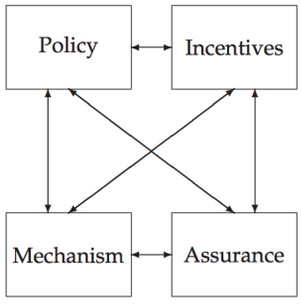
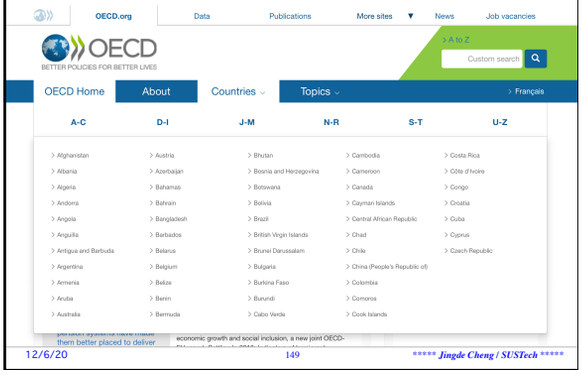


Figure 1.1: Security Engineering Analysis Framework

OECD Guidelines and Recommendations on Security

OECD Guidelines and Recommendations on Security

OECD: The Organisation for Economic Co-operation and Development



OECD: The Organisation for Economic Co-operation and Development

OECD: What is It?

- The Organisation for Economic Co-operation and Development (OECD)
  - OEED was established in 1948 to run the US-financed Marshall Plan for reconstruction of a continent ravaged by war.
  - By making individual governments recognise the interdependence of their economies, it paved the way for a new era of cooperation that was to change the face of Europe.
  - Encouraged by its success and the prospect of carrying its work forward on a global stage, Canada and the US joined OEEC members in signing the new OECD Convention on 14 December 1960.
  - OECD was officially born on 30 September 1961, when the Convention entered into force.

OECD: What is It?

CURRENT MEMBERSHIP

Australia	France	Korea	Portugal
Austria	Germany	Latvia	Slovak Republic
Belgium	Greece	Lithuania	Slovenia
Canada	Hungary	Luxembourg	Spain
Chile	Iceland	Mexico	Sweden
Czech Republic	Ireland	Netherlands	Switzerland
Denmark	Israël	New Zealand	Turkey
Estonia	Italy	Norway	United Kingdom
Finland	Japan	Poland	United States

In the Supplementary Protocol No. 1 to the Convention on the OECD of 14 December 1960, the signatories to the Convention agreed that the European Commission shall take part in the work of the OECD.

European Commission representatives participate alongside Members in discussions on the OECD's work programme, and are involved in the work of the entire Organisation and its different bodies.

While the European Commission's participation goes well beyond that of an observer, it does not have the right to vote and does not officially take part in the adoption of legal instruments submitted to the Council for adoption.

OECD Guidelines and Recommendations on Security

- Guidelines Governing the Protection of Privacy and Transborder Flows of Personal Data, 1980.
- Guidelines for the Security of Information Systems, 1992.
- Guidelines for the Security of Information Systems and Networks, 2002.
- Recommendation on Digital Security Risk Management for Economic and Social Prosperity, 2015.
- Guidelines for Cryptography Policy, 1997.

Guidelines for the Security of Information Systems and Networks, 2002

- Aims (These Guidelines aim to:)
  - − Promote a culture of security among all participants as a means of protecting information systems and networks.
  - − Raise awareness about the risk to information systems and networks; the policies, practices, measures and procedures available to address those risks; and the need for their adoption and implementation.
  - − Foster greater confidence among all participants in information systems and networks and the way in which they are provided and used.

Guidelines for the Security of Information Systems and Networks, 2002

- Aims (These Guidelines aim to:)
  - − Create a general frame of reference that will help participants understand security issues and respect ethical values in the development and implementation of coherent policies, practices, measures and procedures for the security of information systems and networks.
  - − Promote co-operation and information sharing, as appropriate, among all participants in the development and implementation of security policies, practices, measures and procedures.
  - − Promote the consideration of security as an important objective among all participants involved in the development or implementation of standards.

Guidelines for the Security of Information Systems and Networks, 2002

- Awareness Principle
  - "Participants should be aware of the need for security of information systems and networks and what they can do to enhance security."
- Explanation for Awareness Principle
  - Awareness of the risks and available safeguards is the first line of defence for the security of information systems and networks. Information systems and networks can be affected by both internal and external risks.
  - Participants should understand that security failures may significantly harm systems and networks under their control.
  - They should also be aware of the potential harm to others arising from interconnectivity and interdependency.
  - Participants should be aware of the configuration of, and available updates for, their system, its place within networks, good practices that they can implement to enhance security, and the needs of other participants.

Guidelines for the Security of Information Systems and Networks, 2002

♣ **Responsibility Principle**

- "All participants are responsible for the security of information systems and networks."

♣ **Explanation for Responsibility Principle**

- Participants depend upon interconnected local and global information systems and networks and should understand their responsibility for the security of those information systems and networks.
- They should be accountable in a manner appropriate to their individual roles.
- Participants should review their own policies, practices, measures, and procedures regularly and assess whether these are appropriate to their environment.
- Those who develop, design and supply products and services should address system and network security and distribute appropriate information including updates in a timely manner so that users are better able to understand the security functionality of products and services and their responsibilities related to security.

Guidelines for the Security of Information Systems and Networks, 2002

♣ **Response Principle**

- "Participants should act in a timely and co-operative manner to prevent, detect and respond to security incidents."

♣ **Explanation for Response Principle**

- Recognising the interconnectivity of information systems and networks and the potential for rapid and widespread damage, participants should act in a timely and co-operative manner to address security incidents.
- They should share information about threats and vulnerabilities, as appropriate, and implement procedures for rapid and effective co-operation to prevent, detect and respond to security incidents.
- Where permissible, this may involve cross-border information sharing and co-operation.

Guidelines for the Security of Information Systems and Networks, 2002

♣ **Ethics Principle**

- "Participants should respect the legitimate interests of others."

♣ **Explanation for Ethics Principle**

- Given the pervasiveness of information systems and networks in our societies, participants need to recognise that their action or inaction may harm others.
- Ethical conduct is therefore crucial and participants should strive to develop and adopt best practices and to promote conduct that recognises security needs and respects the legitimate interests of others.

Guidelines for the Security of Information Systems and Networks, 2002

♣ **Democracy Principle**

- "The security of information systems and networks should be compatible with essential values of a democratic society."

♣ **Explanation for Democracy Principle**

- Security should be implemented in a manner consistent with the values recognised by democratic societies including the freedom to exchange thoughts and ideas, the free flow of information, the confidentiality of information and communication, the appropriate protection of personal information, openness and transparency.

Guidelines for the Security of Information Systems and Networks, 2002

♣ **Risk Assessment Principle**

- "Participants should conduct risk assessments."

♣ **Explanation for Risk Assessment Principle**

- Risk assessment identifies threats and vulnerabilities and should be sufficiently broad-based to encompass key internal and external factors, such as technology, physical and human factors, policies and third-party services with security implications.
- Risk assessment will allow determination of the acceptable level of risk and assist the selection of appropriate controls to manage the risk of potential harm to information systems and networks in light of the nature and importance of the information to be protected.
- Because of the growing interconnectivity of information systems, risk assessment should include consideration of the potential harm that may originate from others or be caused to others.

Guidelines for the Security of Information Systems and Networks, 2002

♣ **Security Design and Implementation Principle**

- "Participants should incorporate security as an essential element of information systems and networks."

♣ **Explanation for Security Design and Implementation Principle**

- Systems, networks and policies need to be properly designed, implemented and co-ordinated to optimise security.
- A major, but not exclusive, focus of this effort is the design and adoption of appropriate safeguards and solutions to avoid or limit potential harm from identified threats and vulnerabilities.
- Both technical and non-technical safeguards and solutions are required and should be proportionate to the value of the information on the organisation's systems and networks.
- Security should be a fundamental element of all products, services, systems and networks, and an integral part of system design and architecture. For end users, security design and implementation consists largely of selecting and configuring products and services for their system.

Guidelines for the Security of Information Systems and Networks, 2002

- **Security Management Principle**
  - "Participants should adopt a comprehensive approach to security management."
- **Explanation for Security Management Principle**
  - Security management should be based on risk assessment and should be dynamic, encompassing all levels of participants' activities and all aspects of their operations.
  - It should include forward-looking responses to emerging threats and address prevention, detection and response to incidents, systems recovery, ongoing maintenance, review and audit.
  - Information system and network security policies, practices, measures and procedures should be co-ordinated and integrated to create a coherent system of security.
  - The requirements of security management depend upon the level of involvement, the role of the participant, the risk involved and system requirements.

Guidelines for the Security of Information Systems and Networks, 2002

- **Reassessment Principle**
  - "Participants should review and reassess the security of information systems and networks, and make appropriate modifications to security policies, practices, measures and procedures."
- **Explanation for Reassessment Principle**
  - New and changing threats and vulnerabilities are continuously discovered.
  - Participants should continually review, reassess and modify all aspects of security to deal with these evolving risks.

ISO/IEC 15408 (Common Criteria): What Is It?

ISO/IEC 15408 (Common Criteria): What Is It?

ISO/IEC 15408 (Common Criteria): What Is It?

- **Common Criteria (CC)**
  - The Common Criteria for Information Technology Security Evaluation (CC) .
  - The CC and the companion Common Methodology for Information Technology Security Evaluation (CEM) are the technical basis for an international agreement, the Common Criteria Recognition Arrangement (CCRA).
- **ISO/IEC 15408 (CC)**
  - Information technology -- Security techniques -- Evaluation criteria for IT security
- **ISO/IEC 18045 (CEM)**
  - Information technology -- Security techniques -- Methodology for IT security evaluation

ISO/IEC 15408 (Common Criteria): What Is It?

- **Common Criteria (CC) is a framework**
  - CC is a framework in which *end-users* can specify their security functional and assurance requirements (SFRs and SARs respectively) through the use of Protection Profiles (PPs), *vendors* can then implement and/or make claims about the security attributes of their products, and *testing laboratories* can evaluate the products to determine if they actually meet the claims.
- **Common Criteria (CC) provides assurance**
  - CC provides assurance that the process of specification, implementation and evaluation of a computer security product has been conducted in a rigorous and standard and repeatable manner at a level that is commensurate with the target environment for use.

ISO/IEC 15408 (Common Criteria): What Is It?

- **Evaluations based on Common Criteria (CC)**
  - Evaluations based on CC are performed on computer security products and systems.
  - The evaluation serves to validate claims made about the target.
  - To be of practical use, the evaluation must verify the target's security features. This is done through PP, ST, SFRs.
  - The evaluation process also tries to establish the level of confidence that may be placed in the product's security features through quality assurance processes. This is done through SARs and EAL.

ISO/IEC 15408 (Common Criteria): What Is It?

- Evaluations based on Common Criteria (CC)
  - Common Criteria is very generic; it does not directly provide a list of product security requirements or features for specific (classes of) products.
  - CC certification cannot guarantee security, but it can ensure that claims about the security attributes of the evaluated product were independently verified.
  - Products evaluated against a CC standard exhibit a clear chain of evidence that the process of specification, implementation, and evaluation has been conducted in a rigorous and standard manner.

ISO/IEC 15408 (CC): Security Concepts and Relationships

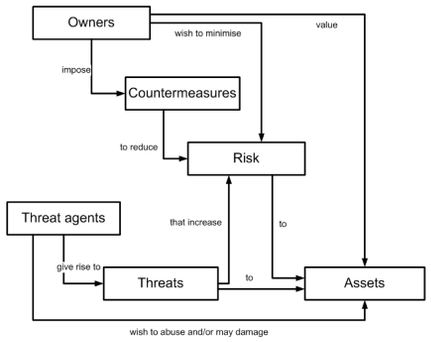


Figure 2 - Security concepts and relationships

ISO/IEC 15408 (CC): Evaluation Concepts and Relationships

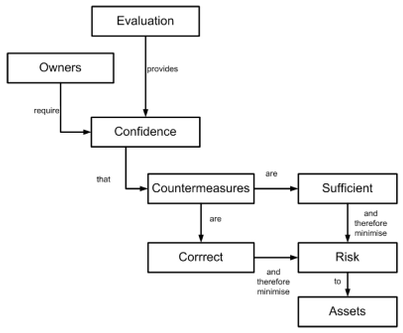


Figure 3 - Evaluation concepts and relationships

ISO/IEC 15408 (CC): Relationships between PP, ST, and TOE

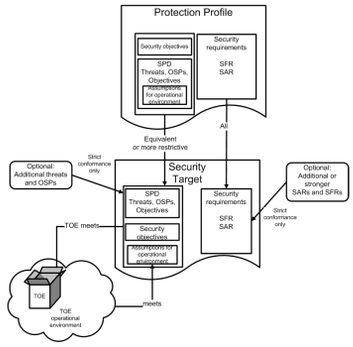


Figure 4 — Relationships between PP, ST and TOE contents

ISO/IEC 15408 (CC): Evaluation Results

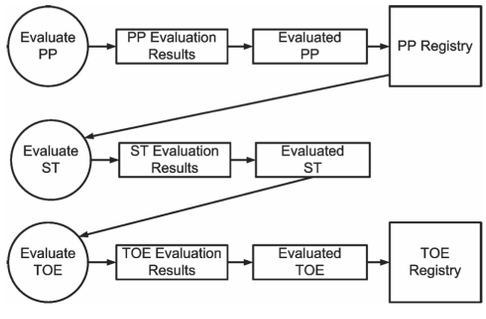


Figure 5 — Evaluation results

History of ISO/IEC 15408 (CC)

- TCSEC
  - Trusted Computer System Evaluation Criteria [DoD(NSA), August 1983]
- CTCPEC
  - Canadian Trusted Computer Product Evaluation Criteria [1988]
- ITSEC
  - Information Technology Security Evaluation Criteria [Germany, France, UK, Holland, June 1991]
- CC project (1993)
  - Common Criteria for Information Technology Security Evaluation [C, F, G, H, UK, US, Ver. 1.0, January 1996]

History of ISO/IEC 15408 (CC)

- ISO/IEC 15408: 1999 (Ver. 1.0)
  - Evaluation Criteria for Information Technology Security [ISO, December 1999]
- ISO/IEC 15408: 2005 (Ver. 2.3)
  - ISO/IEC 15408: 2005 Information Technology - Security Techniques - Evaluation Criteria for IT Security [ISO, August 2005]

The Current Version of ISO/IEC 15408 (CC)

- ISO/IEC 15408: 2009 (Ver. 3.0)
  - ISO/IEC 15408: 2009 Information Technology - Security Techniques - Evaluation Criteria for IT Security [ISO, 15 December, 2009]
    - Part1: Introduction and general model (72 pages)
    - Part2: Security functional requirements (240 pages)
    - Part3: Security assurance requirements (184 pages)
- CC project
  - CC, Version 3.1, Revision 5, April 2017
    - Part1: Introduction and general model (106 pages)
    - Part2: Security functional requirements (323 pages)
    - Part3: Security assurance requirements (247 pages)

CCRA (CC Recognition Arrangement): What it Ensures?

- Products can be evaluated by competent and independent licensed laboratories so as to determine the fulfilment of particular security properties, to a certain extent or assurance;
- Supporting documents, are used within the Common Criteria certification process to define how the criteria and evaluation methods are applied when certifying specific technologies;
- The certification of the security properties of an evaluated product can be issued by a number of Certificate Authorizing Schemes, with this certification being based on the result of their evaluation;
- These certificates are recognized by all the signatories of the CCRA.

Members of CCRA (June, 2017)

ISO/IEC 15408 (CC): Composition

- Part1: Introduction and general model (72 pages)
  - Part1 establishes the general concepts and principles of IT security evaluation and specifies the general model of evaluation given by various parts of ISO/IEC 15408 which in its entirety is meant to be used as the basis for evaluation of security properties of IT products.
- Part2: Security functional requirements (240 pages)
  - Part2 defines the content and presentation of the security functional requirements to be assessed in a security evaluation using ISO/IEC 15408.
- Part3: Security assurance requirements (184 pages)
  - Part3 defines the assurance requirements of the evaluation criteria. It includes the evaluation assurance levels that define a scale for measuring assurance for component targets of evaluation (TOEs), the composed assurance packages that define a scale for measuring assurance for composed TOEs, the individual assurance components from which the assurance levels and packages are composed, and the criteria for evaluation of protection profiles and security targets.

ISO/IEC 15408 (CC): Composition

The following table presents, for the three key target audience groupings, how the parts of ISO/IEC 15408 will be of interest.

	Consumers	Developers	Evaluators
Part 1	Use for background information and are obliged to use for reference purposes. Guidance structure for PPs.	Use for background information and reference purposes. Are obliged to use for the development of security specifications for TOEs.	Are obliged to use for reference purposes and for guidance in the structure for PPs and STs.
Part 2	Use for guidance and reference when formulating statements of requirements for a TOE.	Are obliged to use for reference when interpreting statements of functional requirements and formulating functional specifications for TOEs.	Are obliged to use for reference when interpreting statements of functional requirements.
Part 3	Use for guidance when determining required levels of assurance.	Use for reference when interpreting statements of assurance requirements and determining assurance approaches of TOEs.	Use for reference when interpreting statements of assurance requirements.

Table 1 — Road map to the "Evaluation criteria for IT security"

Big Challenges in Information Security Engineering

- ♣ The fundamentals of ISE
  - ◆ Define the primitive, explicit, measurable, consistent, and complete security criteria of information systems.
  - ◆ Establish the logical/mathematical foundation of information systems and ISE activities.
  - ◆ Establish the effective formal methods for ISE activities, especially for ISE activities of large, complex, high secure information systems.
- ♣ Automation and intellectualization of ISE
  - ◆ Develop automated technologies for ISE activities.
  - ◆ Develop intelligent technologies for ISE activities.

- ♣ Major characteristics of the current ISE
  - ◆ The enemy is the active people: In any case, it is the active people who has creation, learning, growth ability to damage information security of systems.
  - ◆ Passive defense: The current situation is "*passive defense*", "losing the initiative, you will be controlled by your enemy" (后发制于人), and "seek medical advice only after illness", "mend the fold after the sheep have been stolen" (亡羊补牢).
- ♣ The fundamental question about the future ISE
  - ◆ How can we make *active defense*?
  - ◆ 先发制人, 防患于未然?

- ♣ Automated ISE
  - ◆ The problem to be solved: How can we perform ISE actions/ tasks consistently, continuously, formally, rapidly?
  - ◆ *Automation*: Engineering activities supported by automated and semi-automated tools based on ISO/IEC security standards.
- ♣ Intelligent ISE
  - ◆ The problem to be solved: How can we identify and find, by discovery and prediction based on reasoning, possible security holes, leaks by inside enemies, and attacks from outside enemies, ahead of the enemies?
  - ◆ *Intelligence*: Discovery and prediction supported by forward reasoning based on strong relevant logic.

- ◆ Computer Science: What Is It and Why Study It?
- ◆ Computation: What Is It and Why Study It?
- ◆ Computability
- ◆ Computational Complexity (CS101A class only)
- ◆ Algorithms
- ◆ Data, Information, and Knowledge, and Their Representations
- ◆ Data Storage
- ◆ Computer Architecture
- ◆ Data Manipulation in Computer Systems
- ◆ Programming Languages and Compilers
- ◆ Operating Systems
- ◆ System Software and Application Software
- ◆ Software Engineering (CS101A class only)
- ◆ Knowledge Engineering and Artificial Intelligence (CS101A class only)
- ◆ Information Security Engineering (CS101A class only)