Unit 1

軟體工程與管理簡介
Intro. to Software Engineering & Management

台科大資管系軟體工程與管理研究中心
黃世禎教授
Professor Sun-Jen Huang
Software Engineering and Management Lab.
Department of Information Management
National Taiwan University of Science & Technology

學習目標

- 瞭解軟體開發是一項專業且需要被管理
- 瞭解軟體開發的四項本質與困難
- 瞭解軟體開發常見的軟體疏失
- 能夠定義何謂「軟體工程」與知識領域
- 瞭解工程與軟體工程的原則
- 瞭解國際與產業軟體工程標準

何謂軟體

- 何謂「軟體」
  - 軟體是解決真實世界特定問題的系統
  - 軟體的是運作在電腦上所呈現之動態行為
  - 軟體系統與開發各階段之工作產出：需求規格、分析與設計文件、資料庫、程式碼、系統管理與使用者文件
- 不同類型的軟體系統
  - 系統軟體、應用軟體、嵌入式軟體、中介軟體
  - 異動處理軟體(TPS)、管理資訊系統(MIS)、決策支援系統(DSS)、高階主管資訊系統(EIS)
  - 企業資源規劃系統(ERP), 客戶關係系統(CRM), 供應鏈系統(SCM), 商業智慧(BI)
  - 套裝軟體Package、Web Application, APP, SaaS

軟體工程

- 鑑於軟體開發所遭遇到的困境，產生軟體危機，因此北大西洋公約組織於1968年舉辦了首次的軟體工程學術會議
- 軟體工程定義：研究軟體工程原則（principle）、技術、方法與工具，以提升流程效率、人員生產力與軟體品質
- 軟體工程目標：準時交付（如期）、符合預算（如本）、優異品質（如質）、客戶滿意
- 軟體公司主要競爭力面向：品質（物美）、成本（價廉）、時間（交付或上市時間）
- 影響軟體公司競爭力的主要因素：人員、技術、流程（制度）

軟體流程的主要階段或作業

- 需求定義
- 系統分析
- 系統設計
- 程式設計
- 系統測試
- 部署上線
- 系統導入教育訓練
- 系統維運

何謂優異的軟體系統？

- Functional requirement / 功能性需求
  - Accuracy / 正確
  - Completeness / 完整
- Non-Functional Requirements / 非功能性需求
  - Performance / 效能
  - Ease of Use / 易用性
  - Reliability / 可靠性
  - Maintainability / 維護性
  - Security / 安全性
  - Portability / 可移植性
  - Etc.

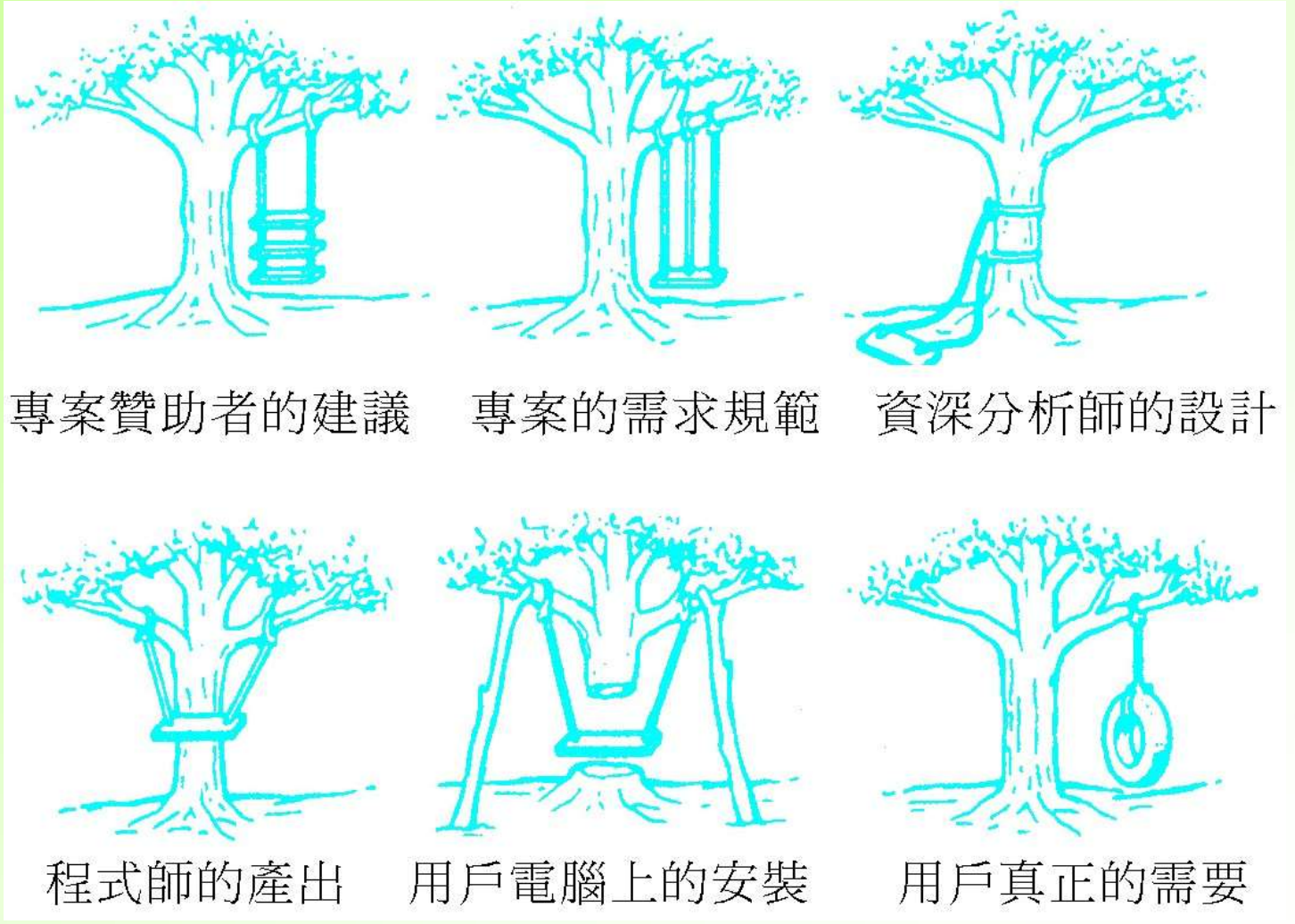
軟體開發的本質

- 從「問題」到「答案」
- 從「人腦」到「電腦」
- 從「程式」到「系統」
- 從「個人」到「團隊」

從「問題」到「答案」

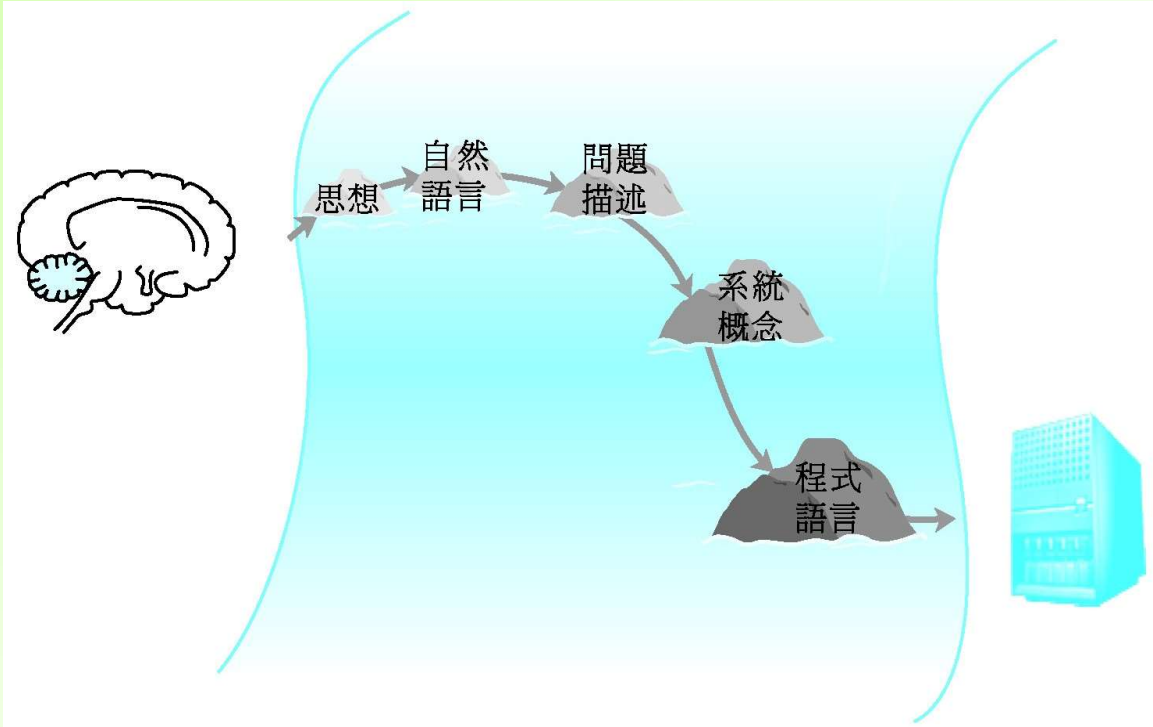
- 尋找解決特定的領域問題(Domain Problem)的答案，並將此解答以『軟體系統』呈現出，因此軟體開發具備以下的本質：
  - 複雜性(complexity)：軟體的目的是要解決問題，其中牽涉到尋找答案的步驟，並將答案（業務流程）以軟體系統建構出，通常是很複雜的。
  - 不可見性(invisibility)：軟體是執行在電腦硬體上，不像其他工程的產出是有實體的存在，因使軟體系統的規模估算、品質評估是較難的
  - 智能活動(Mental Activity)：尋找問題的答案與軟體系統的開發都是須要由人的心智活動來產出，因此是很容易會錯誤的現象
  - 變異性(changeability)：問題領域通常是不是很明確，而且在軟體系統開發過程很有可能會變動。
  - 專業差異性：問題領域或使用者的專業與軟體開發專業是不一樣的，因此在溝通上也常會造成問題。

一個悲慘的軟體開發專案故事



從「人腦」到「電腦」

- 人腦與電腦的差異：人腦與電腦是不一樣的，因此須要思考如何將問題的解答（人腦）轉換並執行到軟體系統（電腦）上
- 目前的技術電腦還是無法取代所有人腦的思維活動，因此需問題的解答需能區分出人腦與電腦的負責範圍



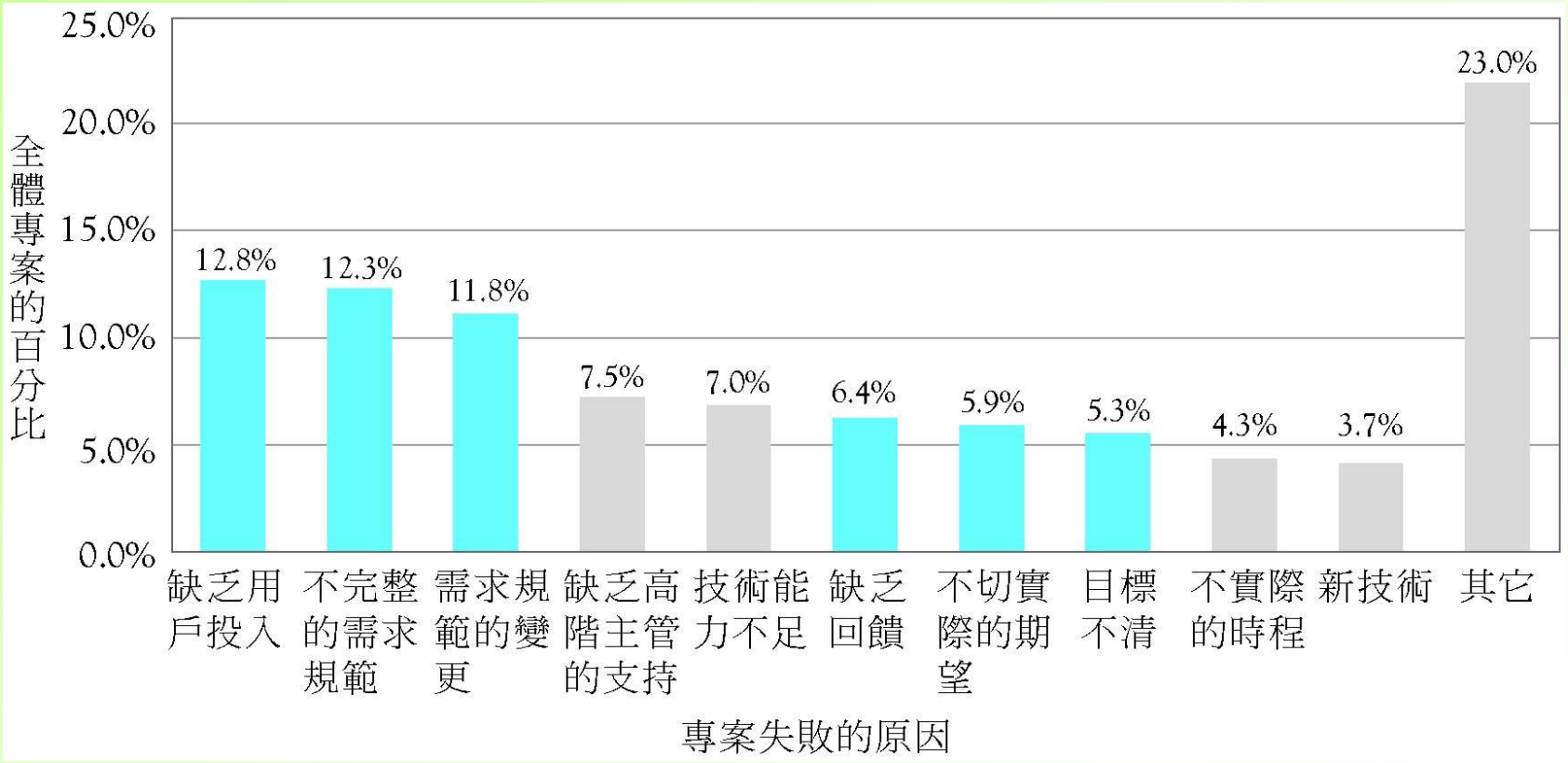
從「程式」到「系統」

- 軟體開發的另一個難題，是從單一程式到軟體系統過程中所造成複雜度的快速上升，其間需要包含不同的活動與技能，使得軟體開發必須面對多樣性的挑戰
- 系統一般是由許多子系統構成的，因此系統的架構與子系統間的協同作業一致性是很重要的
- 對程式與軟體系統的品質需求是有很大的差異
- 軟體生命週期包含軟體開發與維護，但通常軟體開發與維護不一定是同樣的人

從「個人」到「團隊專案」

- 複雜的軟體系統通常無法由一個人單獨建置完成，而是由一個團隊來負責開發完成。
- 商業性的軟體開發活動須要在預定的資源投入下與時程要求下開發完成，因此一般會成立一個特定的專案。
- 軟體的開發須要經由專案團隊的組織與分工、成員間的溝通、有效的專案計畫、執行與監控等，以確保專案如期、如本與如質目標的達成

軟體專案失敗的原因


25.0%
20.0%
15.0%
10.0%
5.0%
0.0%
全體專案的百分比
12.8%
12.3%
11.8%
7.5%
7.0%
6.4%
5.9%
5.3%
4.3%
3.7%
23.0%
缺乏用戶投入
不完整的需求規範
需求規範的變更
缺乏高階主管的支持
技術能力不足
缺乏回饋
不切實際的期望
目標不清
不實際的時程
新技術
其它
專案失敗的原因

工程原則(Engineering Principle) (1/2)

- 原則一：建立標準作業程序

  作業程序及步驟要定義的很明確與可行，並依照所制訂的程序來執行，並持續檢討與優化SOP。

- 原則二：分解

  將一個複雜的大需求或問題分解成階層式的小需求或小問題，方能找出其處理方法。

- 原則三：模組化

  將上述小問題的解決方案於以模組化，以利未來的重複使用，以達提昇生產力目標。

- 原則四：遞增法

  將上述階層式的分解結構圖依其重要性先完成較重要的部分，在確認沒問題後逐步遞增完成大系統的建置。

工程原則(Engineering Principle) (2/2)

- 原則五：預視改變

  在處理目前的問題時，需要想到未來問題與環境等是否會變動，亦即要有風險/不確定的意識，並能及時作好規劃

- 原則六：焦點分離

  將一個複雜的問題分為不同的層次或構面來處理，而不同的層次或構面需要專注與聚焦在各自要處理的問題上

- 原則七：通用性

  將一個特定問題的解決方法於以通用性，使其能適用在一般性問題的解決，達到再使用與擴大其效益

Software engineering fundamentals

- Some fundamental principles apply to all types of software system, irrespective of the development techniques used:
  - Systems should be developed using a managed and understood development process. Of course, different processes are used for different types of software.
  - Dependability and performance are important for all types of system.
  - Understanding and managing the software specification and requirements (what the software should do) are important.
  - Where appropriate, you should reuse software that has already been developed rather than write new software.

軟體開發的重要原則

- 專案管理三角理論：範疇、時間、成本、品質

- 為了要提昇生產力與軟體品質，盡可能重複使用 (Reuse) 經過驗證安全無疑的既有軟體元件與模組
- 因為軟體開發有漣漪效果(Ripple Effect)，因此須要及早發現錯誤並修復之，避免流入後續階段發生更多的錯誤。

軟體開發的重要原則

- 瞭解軟體系統問題領域的業務流程與邏輯是影響軟體品質的重要關鍵因素之一
- 瞭解軟體系統各使用群組不同的需求（功能性及非功能性）、使用者構面之變因、系統驗收基準等亦是影響軟體品質的重要關鍵因素之一

SWEBOK 15知識領域KA

Guide to Software Engineering Body of Knowledge (SWEBOK) , V3.0 by IEEE Computer Society, 2014, include 15 knowledge Areas (KA)

- Software Requirement
- Software Design
- Software Construction
- Software Testing
- Software Maintenance
- Software Configuration Management
- Software Engineering Management
- Software Engineering Process

SWEBOK 15知識領域KA

- Software Engineering Models and Methods
- Software Quality
- Software Engineering Professional Practices
- Software Engineering Economic
- Computing Foundations
- Mathematical Foundations
- Engineering Foundations

The SWEBOK V4 is right now under the public review. If you are interested in providing comments, you can visit the website of SWEBOK.

Who is the ISO?

- International Organization for Standardization is the world's largest developer of International Standards
- ISO is a network of the national standards institutes of 162 countries, one member per country
- ISO is a non-governmental organization that forms a bridge between the public and private sectors
  - Many of its member institutes are part of the governmental structure of their countries, or are mandated by their government
  - Other members have their roots uniquely in the private sector, having been set up by national partnerships of industry associations
- This enables ISO to reach a consensus on solutions that meet both the requirements of business and the broader needs of society

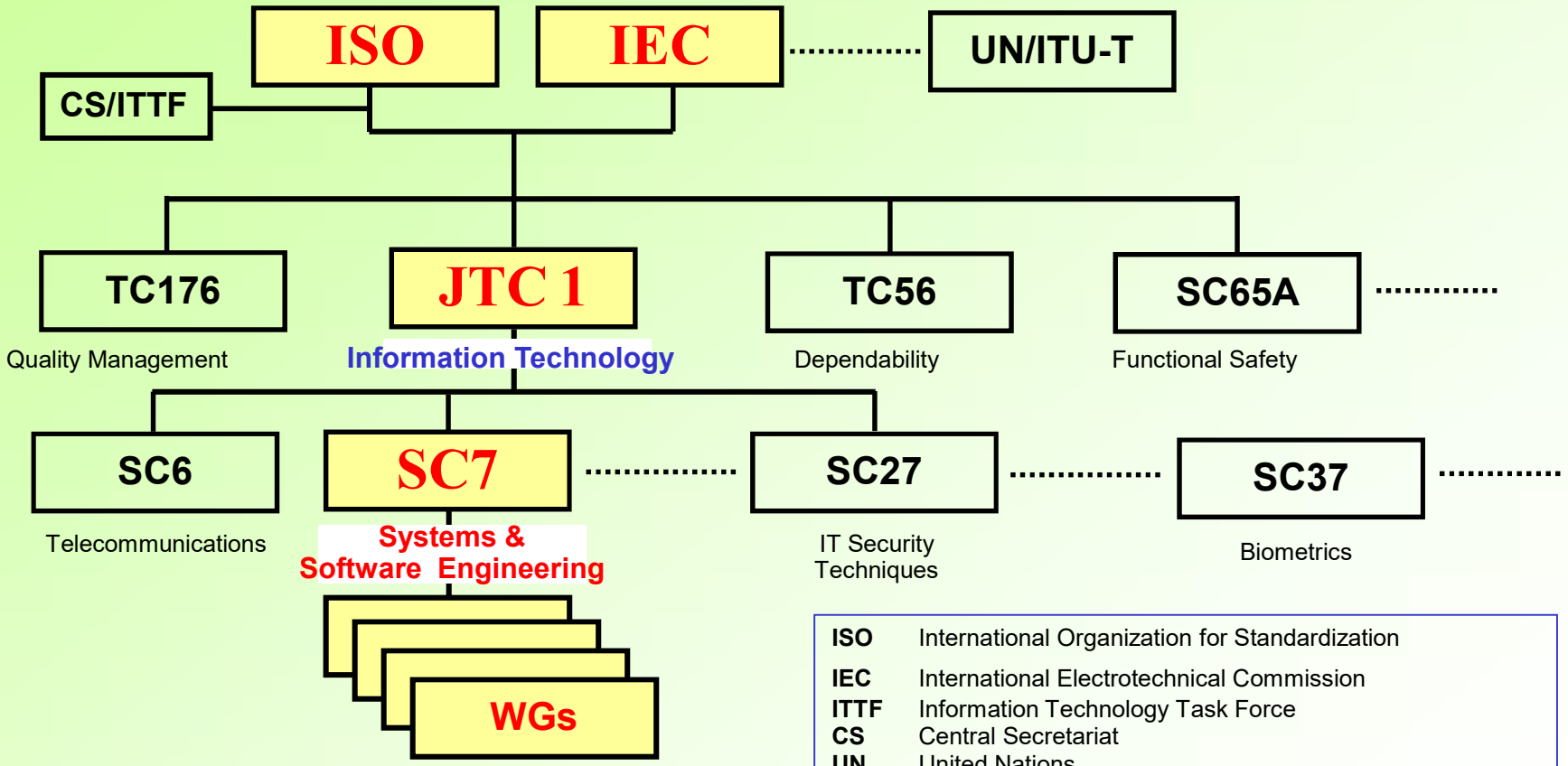
Who develops ISO standards

- ISO standards are developed by technical committees, (or subcommittees) comprising experts from the industrial, academic, technical and business sectors
- These experts may be joined by representatives of government agencies, consumer associations, non-governmental organizations and academic circles, etc.
- Experts participate as national delegations, chosen by the ISO national member body for the country concerned.

How ISO standards are developed

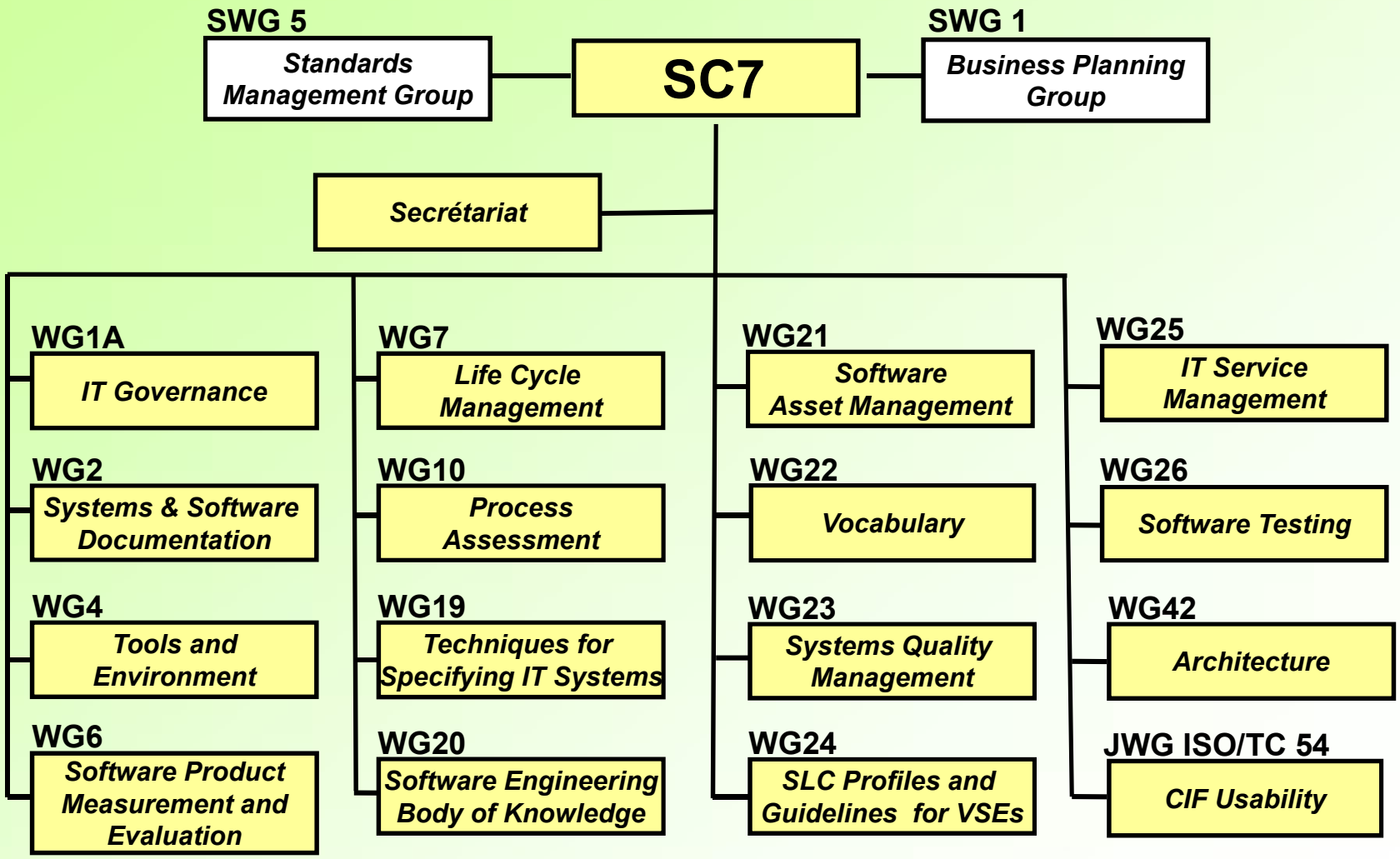
- The national delegations of experts of a committee meet to discuss, debate and argue until they **reach consensus** on a draft agreement
- The resulting document is circulated as a Draft International Standard (**DIS**) to all ISO's member bodies for voting and comment
- If the voting is in favor, the document, with eventual modifications, is circulated to the ISO members as a Final Draft International Standard (**FDIS**)
- The voting process is the key to consensus. If that's achieved then the draft is on its way to becoming an ISO standard. If agreement isn't reached then the draft will be modified further, and voted on again.
- From first proposal to final publication, developing a standard usually takes about 3 years.

ISO/IEC outline Structure



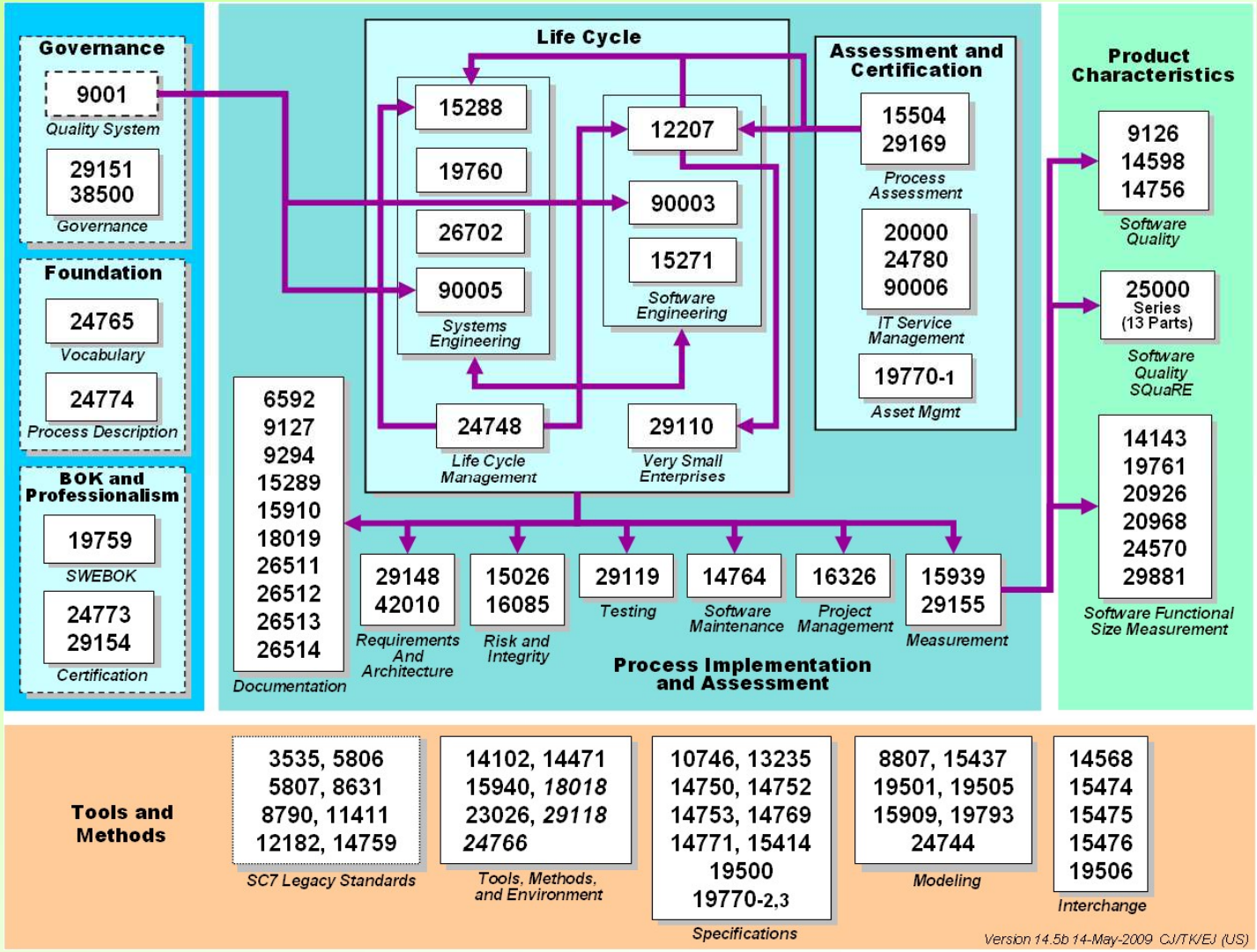
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
ITTF	Information Technology Task Force
CS	Central Secretariat
UN	United Nations
ITU-T	International Telecommunications Union
TC	Technical Committee
SC	Sub Committee
JTC	Joint Technical Committee
WG	Working Group

SC7 Structure



‡ Adapted from Prof. M. Azuma

JTC 1 SC7 Standards Collection



ISO Software Engineering Standards

- www.iso.org
- Standards by ISO/IEC JTC 1 Information technology
- ISO/IEC JTC 1/SC 7 Software and systems engineering
- How many published standards under ISO/IEC JTC 1/SC 7? How many standards under development?
- How many published standards under ISO by ICS 35.080 Software (including development, documentation and use)?

IEEE Software Engineering Standards

- https://library.ntust.edu.tw/home.php
- 常用資料庫 IEEE (IEL)
- https://ieeexplore-ieee-org.ezproxy.lib.ntust.edu.tw/Xplore/home.jsp
- Browse Standards
- By collection => Information Technology => Software
  How many active standards?
  What are the newest standards?
- By ICS code
  How many active standards of ICS code 35.08 Software ?
  What is the first standard sorted by standard number?

小結 (1/2)

- 軟體開發是一項高複雜、高難度、高風險的活動，由於高失敗率，故有「軟體危機」一詞
- 軟體工程發展至今，依然存在著許多問題，尚未能有標準的成功法則，所以學習軟體工程這門科學，不能一知半解，否則或許差之毫釐，卻失之千里。
- 不只要學習**Know-what**與**Know-how**，還必須要瞭解**Know-why**。應透過思辨的過程，深入瞭解各種方法背後的假設與邏輯，以避免掉入各種軟體開發陷阱。
- 軟體開發採用的技術、方法與管理需要考慮到不同的軟體類型，沒有最好的技術、方法，只有最適合的。
- 工程師們須培養多樣化的技能，尤其是瞭解與分析用戶需求的能力。此外，在團隊合作的過程中，溝通與管理的技能也不可或缺的；在可預見的未來，人的素質依然決定軟體產品的品質。

小結(2/2)

- 用整體觀點看待軟體開發：許多專案之所以遭遇到困難，是因為消耗太多資源在專案的某一方面，忽略了其它應注意而未注意的事項，以致於後續發生問題時，再回頭來彌補或救火，卻事倍功半甚至無法達到目的。
- 建立工程與管理的基礎：知道理論不表示就能夠做好事情，除非具備基本的實務訓練，否則更多作為最後可能只是空談。對於軟體工程的各項實務，應設法學習、瞭解，並導入軟體開發專案之中。
- 避免典型疏失：軟體專案牽涉到非常多的決策；成功的專案，意味著沒有在開發之路上，犯下一些基本的錯誤。軟體開發的過程中藏有許多陷阱，而大部分的疏失都起因於一些直覺上看起來合理的理由。認識這些疏失並儘量避開，是軟體專案開發成功的先決條件。