Ingeniería de Software II

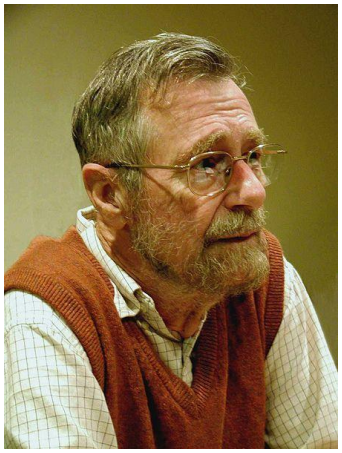
Ingeniería de Software II

Segundo Cuatrimestre de 2011

Clase 1a: Introducción a la Ingeniería de Software

Buenos Aires, 15 de Agosto de 2011

¿Existe la Ingeniería de Software?



- Ingeniería de Software según Dijkstra:
  - Así como la economía se conoce como "La ciencia miserable", la Ingeniería de Software debería ser conocida como "The Doomed Discipline".
  - La ingeniería de software, por supuesto, se presenta a si misma como otra causa meritoria, *pero eso es puro cuento*. Si uno lee cuidadosamente su literatura y analiza lo que sus devotos realmente hacen, descubre que la ingeniería de software ha aceptado como su carta de presentación "como programar si no puede".
  - Es realmente útil ver un programa como una fórmula. Primero, pone la tarea del programador en perspectiva: tiene que derivar esa fórmula... manipulación de símbolos usualmente llamada 'programación'.
  - Para llevarse a casa el mensaje que este curso es primariamente un curso en matemáticas formales, vemos que el lenguaje de programación en cuestión no ha sido implementado en la universidad. De esta forma, los estudiantes son protegidos de la tentación de "testear" sus programas.

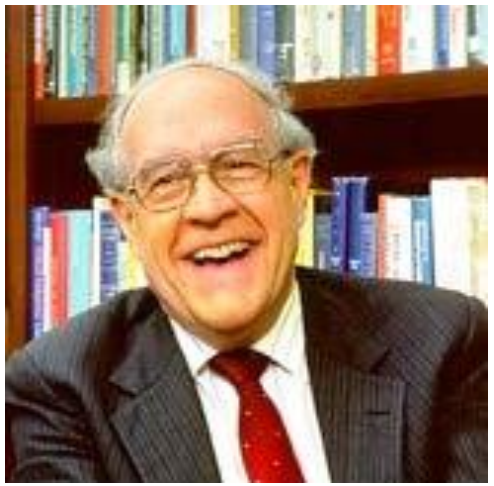
Fuente: ver http://www.cs.utexas.edu/users/EWD/transcriptions/EWD13xx/EWD1305.html

¿Existe la Ingeniería de Software? (cont.)

- Ingeniería de Software según Yourdon:
  - La construcción de sistemas de software que funcionen no es una "ciencia". Los así llamados "científicos en computación" nos tratan de convencer de que nuestros sistemas son en realidad grafos dirigidos, o n-tuplas de formas normalizadas, o autómatas de estados finitos... Sus pronunciamientos son mas relevantes a Zen que al difícil problema de construir sistemas y programas útiles y fáciles de mantener.
  - En lugar de aproximaciones académicas, lo que necesitamos es un conjunto de métodos prácticos que traten con la naturaleza propensa a errores del personal de proyectos, los usuarios, los encargados del mantenimiento, y el proceso de desarrollo. Ese conjunto de métodos es llamado "Ingeniería de Software".

Fuente: ver http://www.yourdon.com

Aportando al a confusión: ¿Existe la *ciencia de la computación*?



- Fred Brooks:
  - Un dicho académico es que "Todo lo que se hace llamar *ciencia,* no lo es".
  - "Física vs. *Ciencias* de la Computación"
  - *El científico construye para aprender*
  - *El ingeniero aprende para construir*
- Heinz Zemanek: "La ingeniería de objetos abstractos"

Fuente: Fred Brooks. The computer scientist as a Toolsmith II. Communications of the ACM, Marzo 1996.

La Ingeniería…

- Antes de ir a la ingeniería de software pensemos en la definición de ingeniería

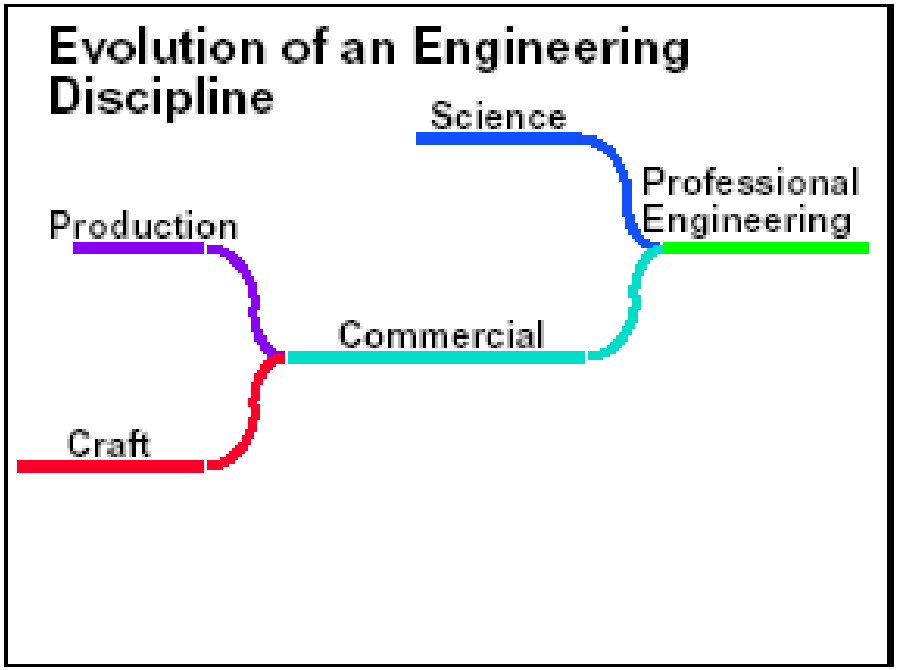
- Crear soluciones eficientes
  - A problemas prácticos
    - Aplicando conocimiento científico
      - Construyendo cosas
        - Al servicio de la humanidad

- *La ingeniería permite a la gente común hacer cosas que antes requerían virtuosos*

- *"Ingeniería es ciencia con un propósito"*

Fuente: Mary Shaw – Prospects for an Engineering Discipline of Software. IEEE Software. Noviembre 1990.

Evolución de una disciplina de ingeniería (Shaw)



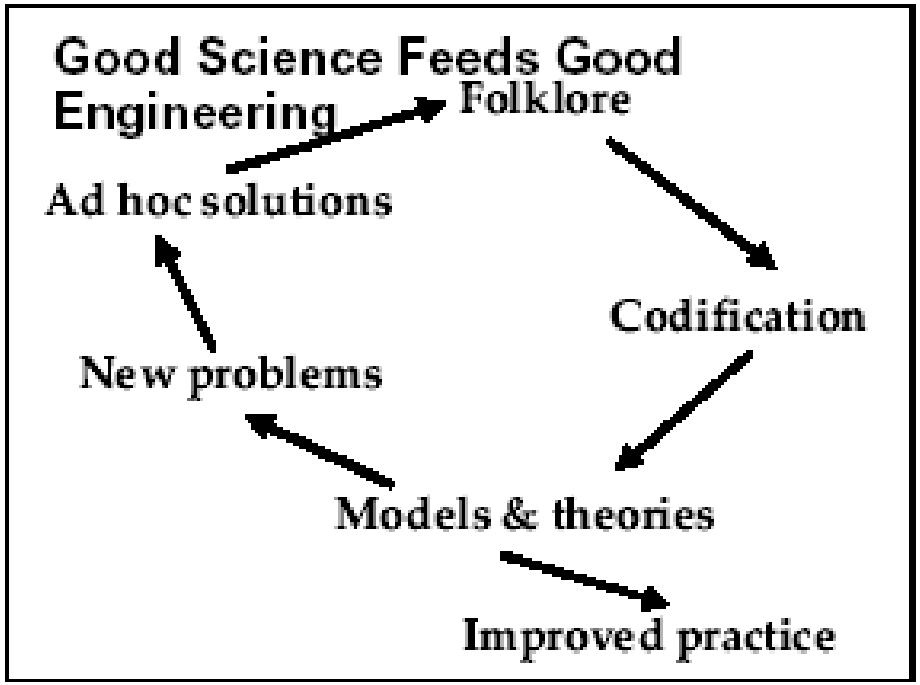
"Craft" ("habilidad"): virtuosos y amateurs. Intuición y fuerza bruta. Se construye para usar, no vender.

Comercial: se construye para vender, más habilidades, procesos establecidos.

Ingeniería profesional: Profesionales educados, el progreso basado en la ciencia, análisis y teoría, segmentación.

Fuente: Mary Shaw – Prospects for an Engineering Discipline of Software. IEEE Software. Noviembre 1990.

Evolución de una disciplina de ingeniería (Shaw) (cont.)



La aparición de modelos y teorías es clave para alimentar el ciclo que nos llevará a la verdadera disciplina de ingeniería.

Fuente: Mary Shaw – Prospects for an Engineering Discipline of Software. IEEE Software. Noviembre 1990.

Definición de Ingeniería de Software

- IEEE: "La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, mantenimiento y operación de sistemas de software"

- Otra definición interesante:
  - Bertrand Meyer: "The development of possibly large systems intended for use in production environments, over a possibly long period, worked on by possibly many people, and possibly undergoing many changes, where development includes management, maintenance, validation, documentation, and so forth."

Fuentes: IEEE Glossary for Software Engineering Terminology.
Bertrand Meyer, Software Engineering in the Academy, IEEE Computer, Mayo 2001.

La negación de la Ingeniería de Software

- Hay dos líneas de argumentos para negar la existencia de la Ingeniería de Software:
  - Falta formalidad en la "ciencia" que está por detrás, no hay consenso sobre cómo hacer las cosas… falta camino por recorrer. La Ingeniería de Software podrá existir algún día, pero todavía no merece ese nombre
  - Todo el enfoque "ingenieril" está equivocado. No es un tema de tiempo o madurez. Construir software se parece más a un arte que a una disciplina de Ingeniería. No tienen casi nada en común.
  - Otra menos común: "el software es abstracto y la ingeniería tiene que ver con cosas concretas. Búsquense otro nombre"

Las Diferencias: Ciencias de la Computación vs. Ingeniería De Software

- La ciencia de la computación se ocupa de las teorías y los fundamentos
- La Ingeniería de Software se ocupa de los aspectos prácticos de desarrollar y entregar software útil
- La Ingeniería de Software se nutre de las ciencias de la computación y otras ciencias
- Otra diferencia: con la Ingeniería de Sistemas (construcción de sistemas que combinan hardware y software)
- *La materia no intenta imponer una opinión de los alumnos sobre estos temas. Sólo intentamos que opinen con fundamentos, luego de leer a los mejores exponentes de cada teoría*

Algunos hitos en nuestra historia

- 1968 – Nace el uso del nombre, conferencia NATO.

SOFTWARE ENGINEERING

Report on a conference sponsored by the

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1968

Although much of the discussions were of a detailed technical nature, the report also contains sections reporting on discussions which will be of interest to a much wider audience. This holds for subjects like

- the problems of achieving sufficient reliability in the data systems which are becoming increasingly integrated into the central activities of modern society
- the difficulties of meeting schedules and specifications on large software projects
- the education of software (or data systems) engineers
- the highly controversial question of whether software should be priced separately from hardware.

Fuente: Reporte de la Conferencia en *Software Engineering*, NATO, 1968

La crisis del Software

- Principio de los 80s
- Primeras "horror stories"
- Respuesta del DoD → Software Engineering Institute
- Famoso artículo de 1994 de Scientific American, caso del aeropuerto de Denver

Otro hito – No Silver Bullet

- Paper magistral de Brooks, IEEE Computer de Abril de 1987

- Dos tipos de problemas:
  - Esenciales
  - Accidentales

- Las 4 dificultades esenciales:
  - Complejidad (no lineal con el tamaño)
  - Conformidad (Arbitrariedad)
  - Facilidad de Cambios
  - Invisibilidad

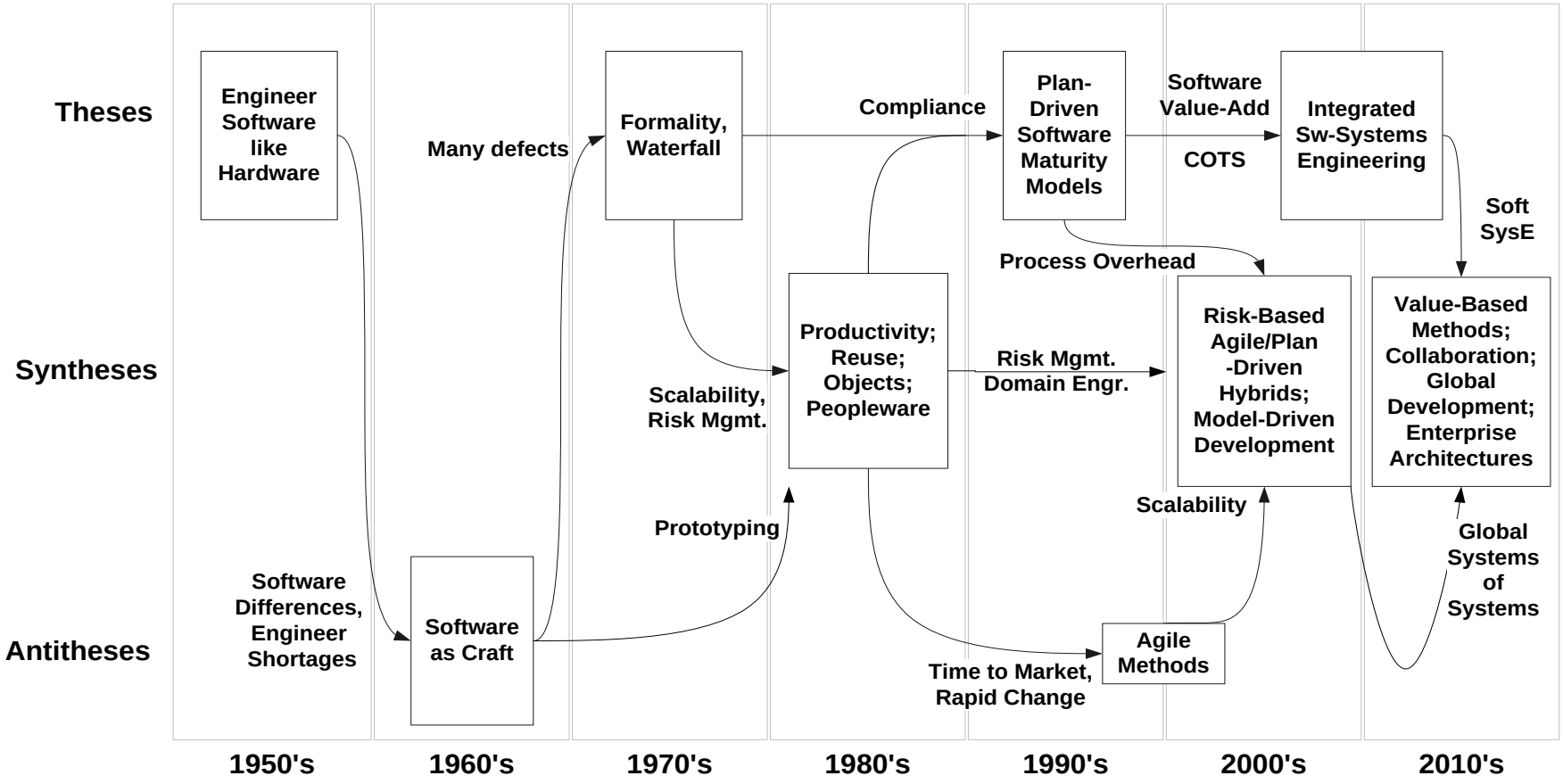
La importancia de "No Silver Bullet"

- En parte debido a su inmadurez, la Ingeniería de Software tiene una peligrosa tendencia a "seguir modas"
  - Aparece una nueva tendencia, generalmente interesante y con aportes
  - Se genera un gran negocio
  - Los que están detrás del negocio presentan a esa tendencia como la "silver bullet"
  - En algún momento, la realidad le da la razón a Brooks
- Hay sólo dos grandes líneas que atacan las dificultadas esenciales:
  - Buy (o usar Open Source) vs. Build
  - Grow vs. Build

Más sobre la evolución de la Ingeniería de Software

- Es muy importante tener en cuenta que la construcción de software cambió drásticamente en las últimas décadas:
  - Jefe a Boehm en los 50: *"Escuchame atentamente, estoy pagando 600 dólares por hora por esta computadora y 2 dólares por hora por vos, así que espero que actúes de acuerdo a eso"*
  - Cada época dejó principios fundamentales
  - Ver Boehm: *"A View of 20th and 21st Century Software Engineering"*, ICSE 2006.
  - Una visión Hegeliana de la Ingeniería de Software: tesis, antítesis, síntesis

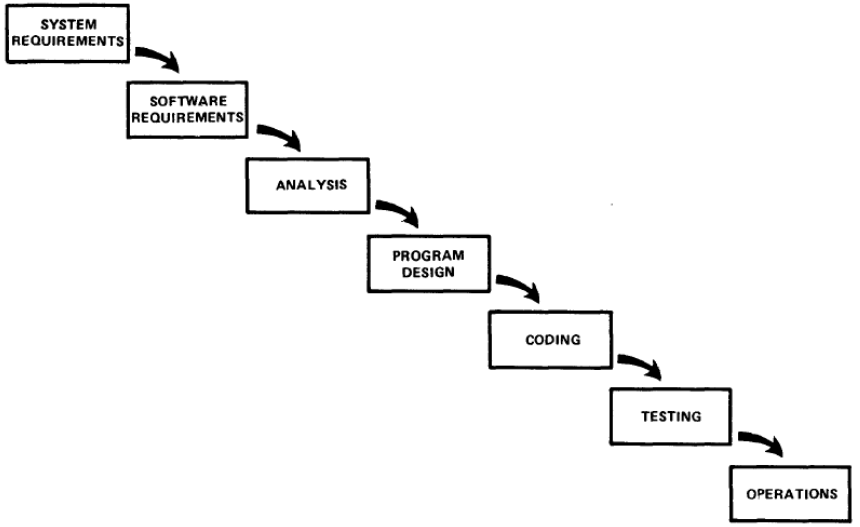
Boehm y su "visión Hegeliana"

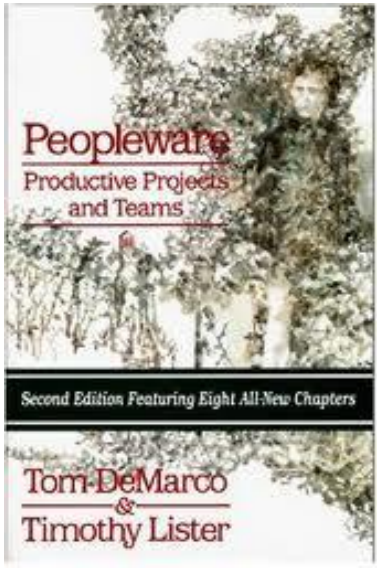


Trazando paralelos

Management en general

Desarrollo de Software

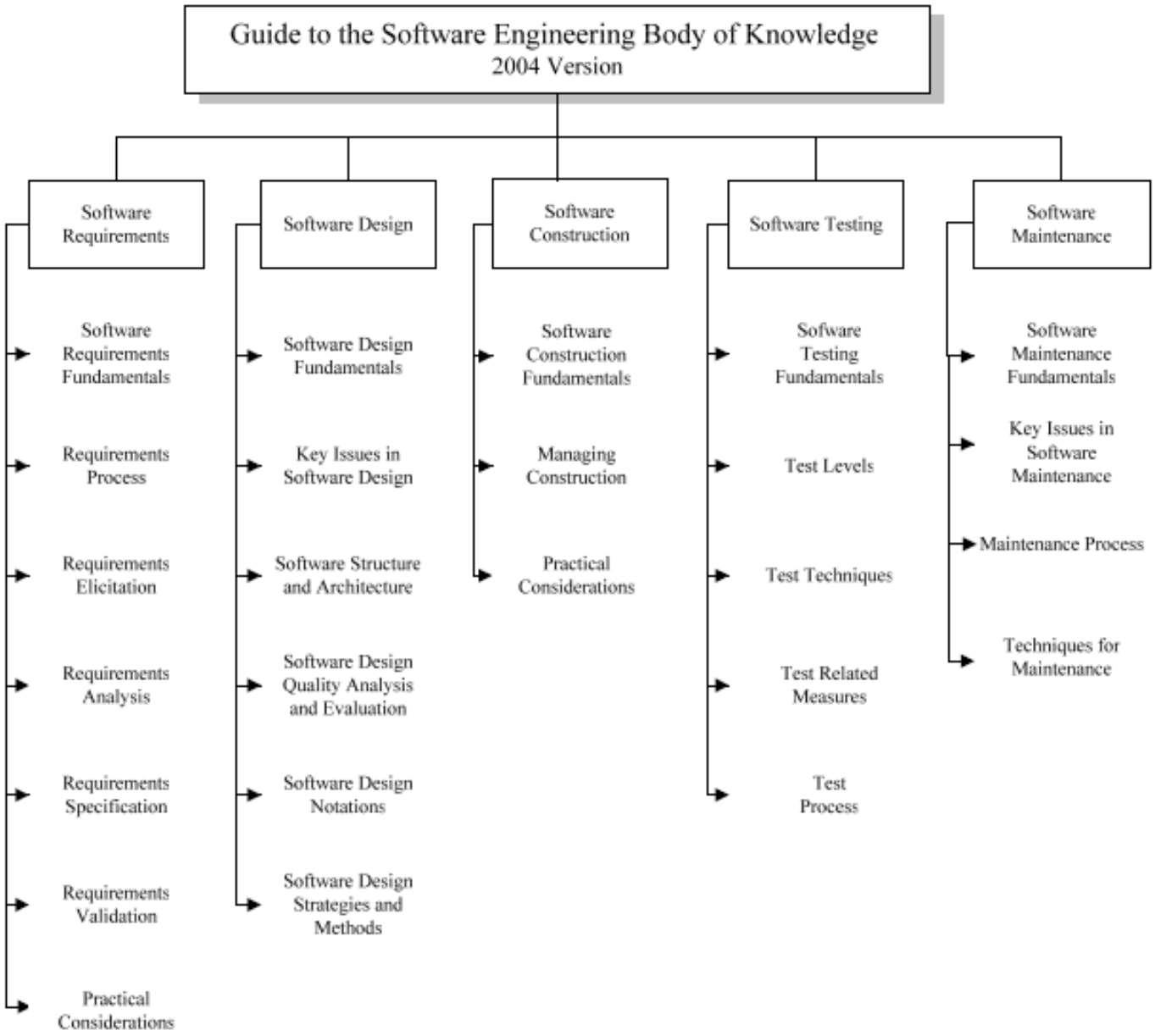




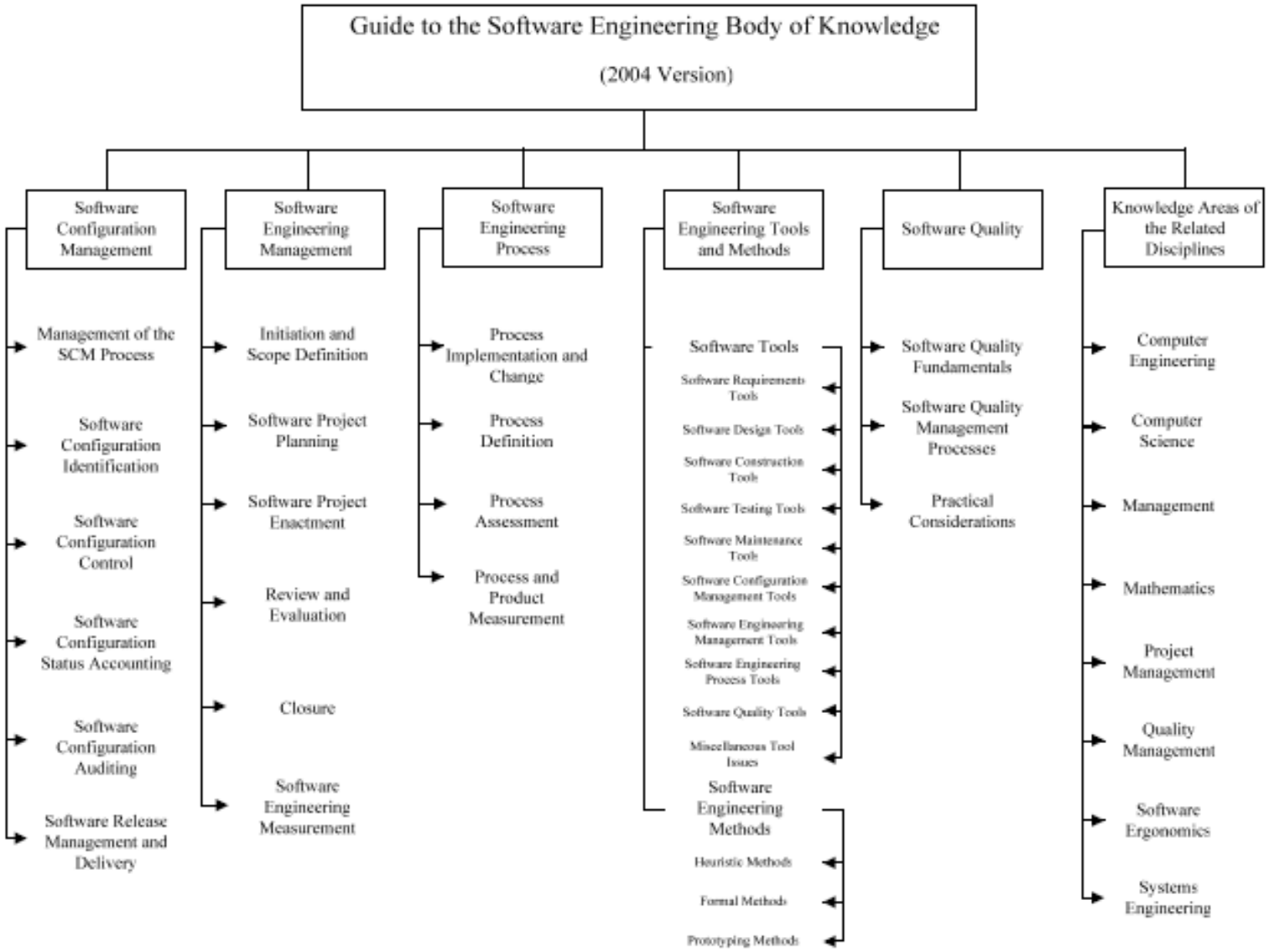
Un cambio de foco en la Ingeniería de Software

- Tom de Marco. *Software Engineering: An Idea who's time has come and gone?*
- "Controlling Software Projects" puso el foco donde no debía estar (en la gestión, y sobre todo el control). El foco debería haber estado siempre en la concepción de un sistema, que es una actividad experimental
- La consolidación de los modelos iterativos e incrementales y de las nuevas tecnologías está cambiando el foco de la Ingeniería de Software, hacia los aspectos más técnicos

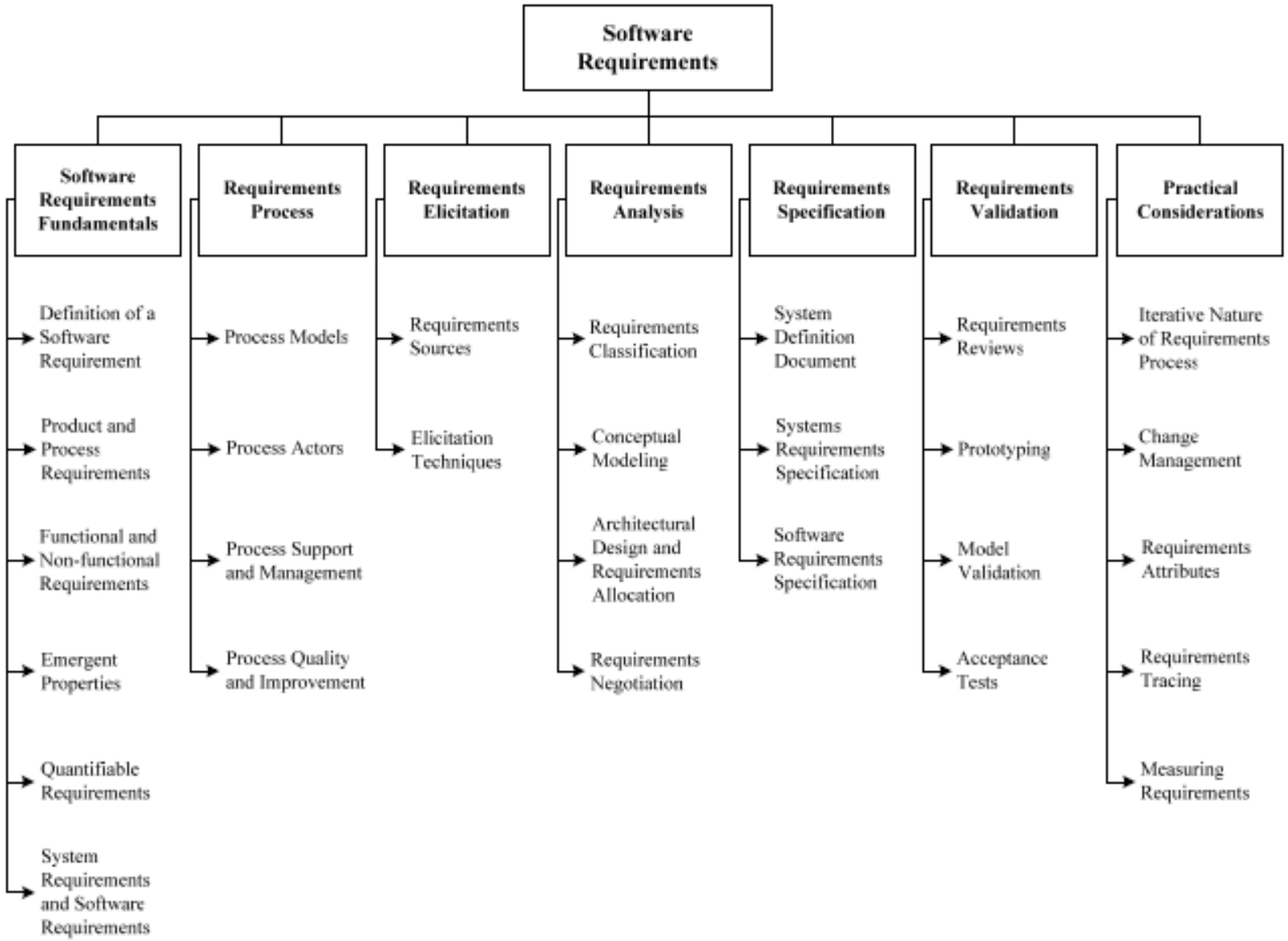
Si quisiéramos saber todo…



SWEBOK – Cont.



Ojo que cada una se expande... Por ejemplo



En resumen:

- Trabajamos con algo muy complejo
- Que trata con una problemática muy amplia
- Que tiene dificultades esenciales
- Y que además está evolucionando
- Hay muchos temas por cubrir
- *Nos vamos a concentrar en aquellos que más afectan la construcción de grandes sistemas de software*