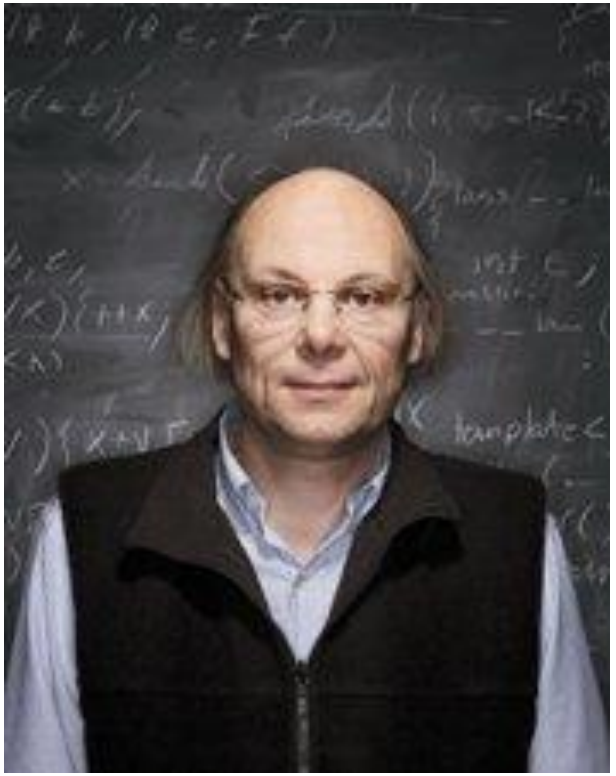
Club de Investigación Tecnológica

Desde 1988

50 años de Ingeniería del Software: Evolución y perspectivas

Ignacio Trejos Zelaya
Club de Investigación Tecnológica
TEC, Escuela de Computación
Universidad Cenfotec

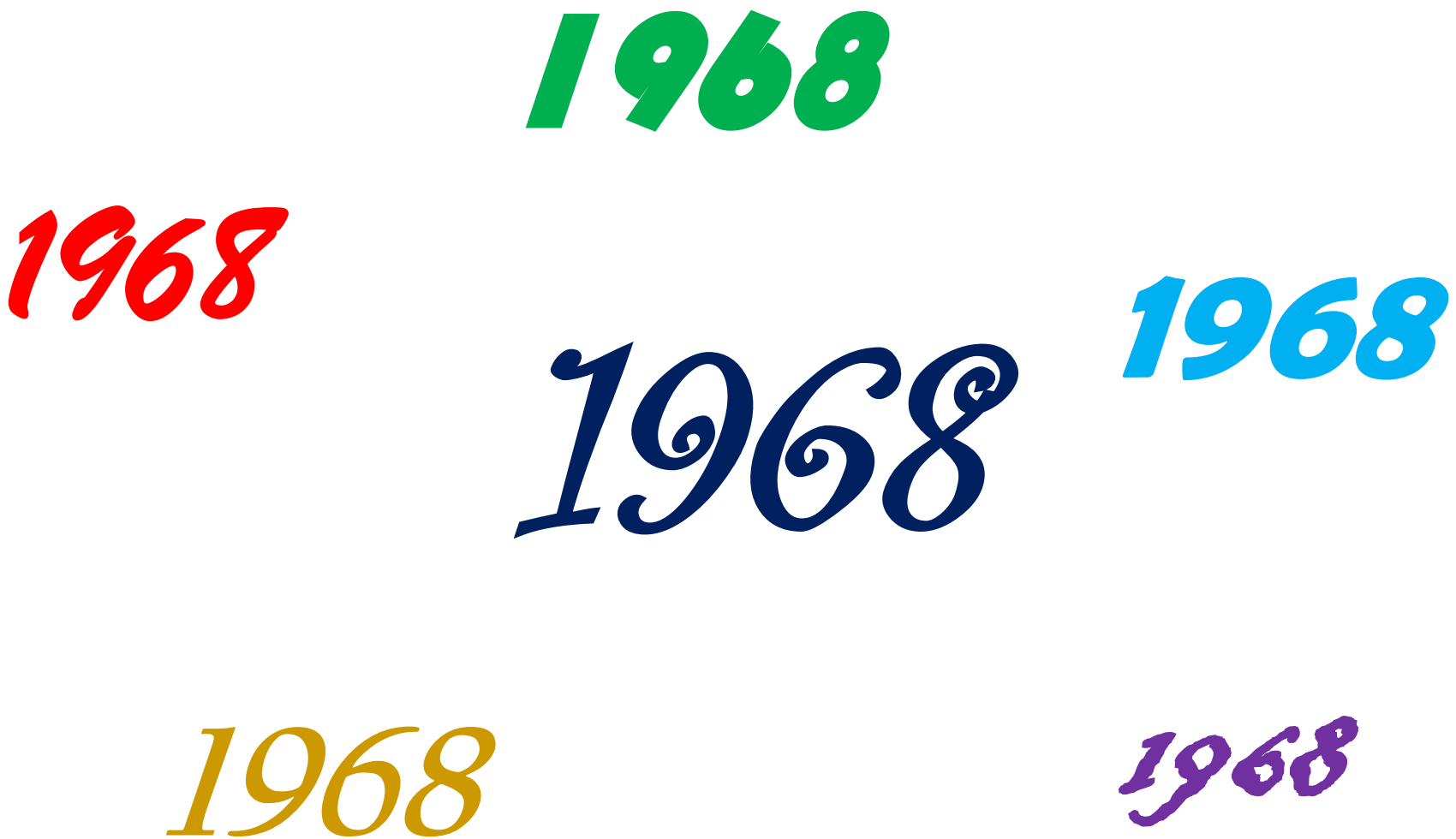


Our civilization runs on software.

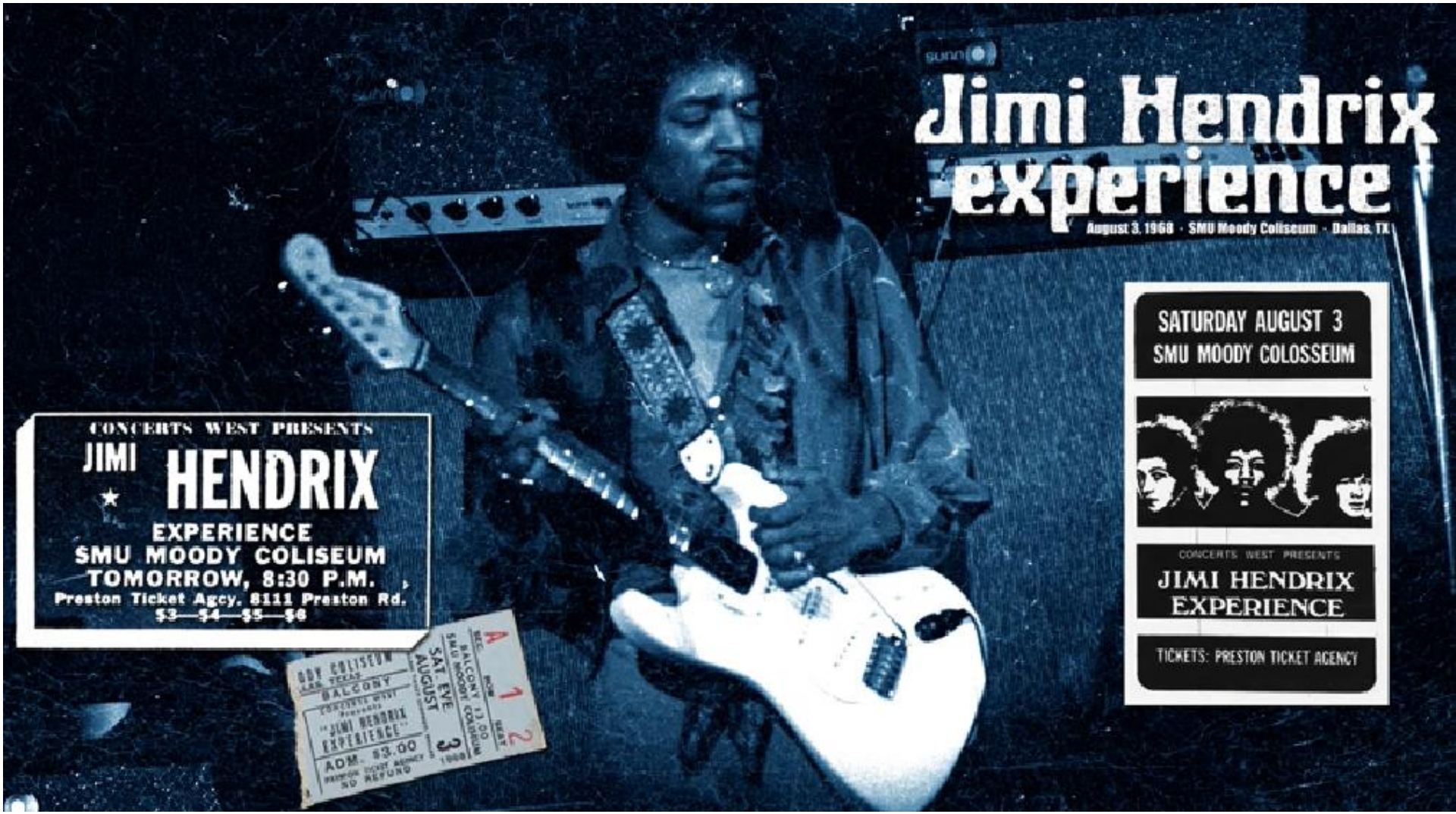
Bjarne Stroustrup

Software is eating the world.

Marc Andreessen



a, 2018..2019



Ignacio Trejos Zelaya, 2018..2019

Ignacio Trejos Zelaya, 2018..2019

Ignacio Trejos Zelaya, 2018..2019

Ignacio Trejos Zelaya, 2018..2019

Ignacio Trejos Zelaya, 2018..2019

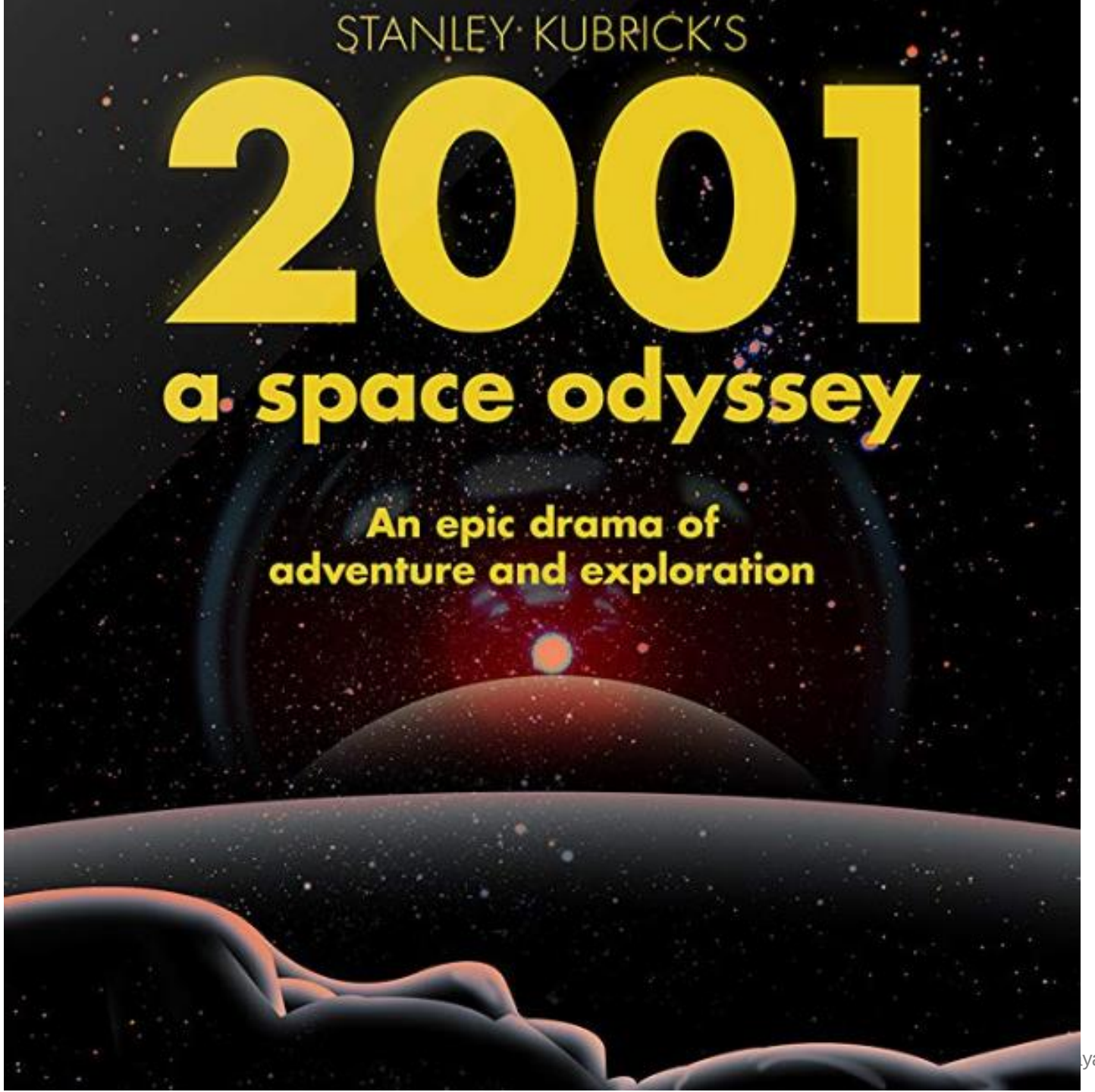
Ignacio Trejos Zelaya, 2018..2019

SPECIAL TIME EDITION

1968

The Year That Shaped a Generation

Ignacio Trejos Zelaya, 2018..2019


STANLEY KUBRICK'S
2001
a space odyssey
An epic drama of
adventure and exploration

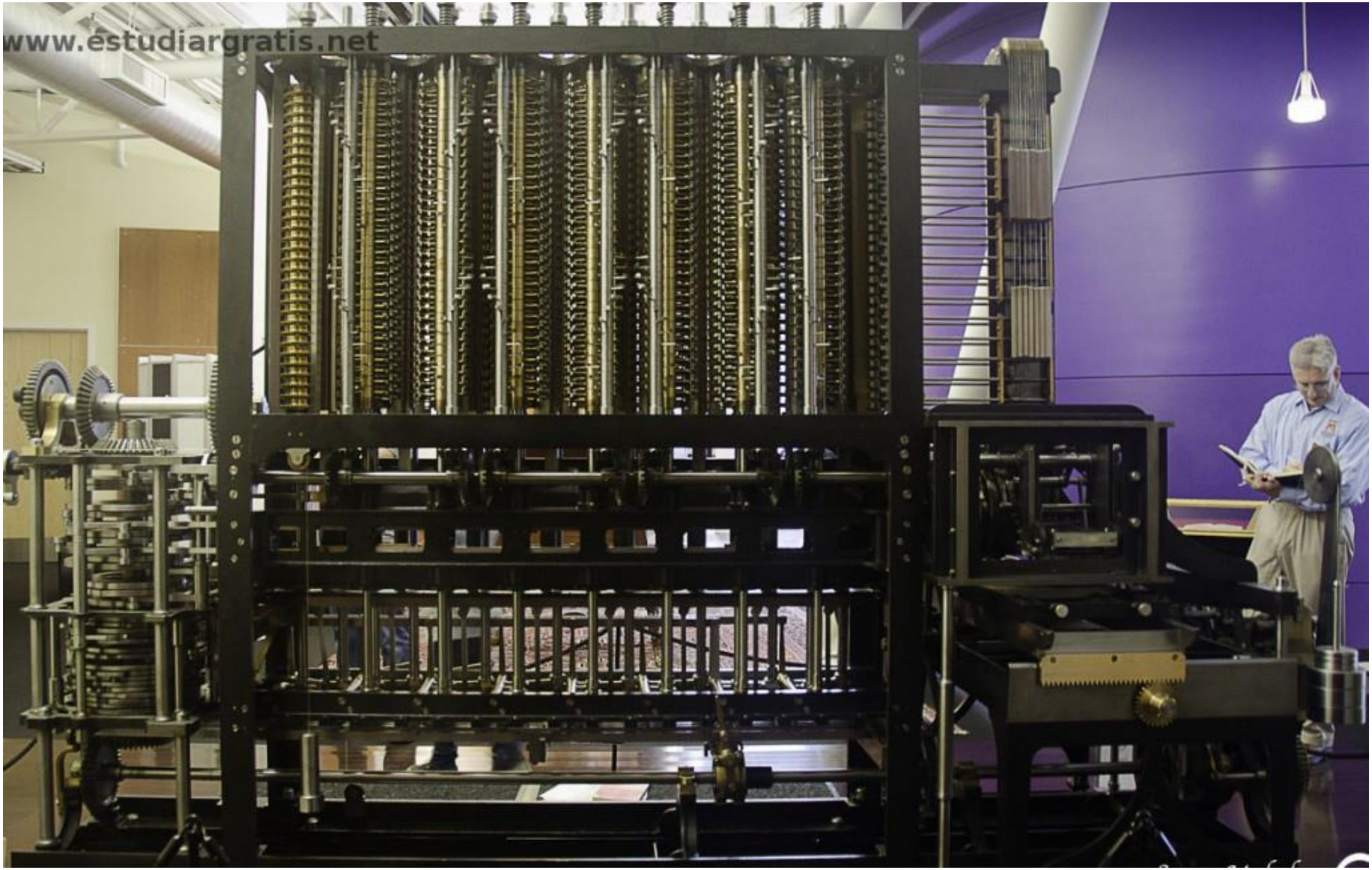
"***Aquellos que no pueden recordar el pasado están condenados a repetirlo***"

Jorge Santayana

Máquinas programables

Máquinas tabuladoras

Computadora mecánica: Babbage



Augusta Ada Byron, Condesa de Lovelace

La característica que distingue a la máquina analítica, es la inclusión en ella del principio que Jacquard concibió para regular la fabricación, mediante tarjetas perforadas, de los más complicados modelos de brocados.

Al capacitar a los mecanismos para combinar entre sí símbolos generales en sucesiones de variedad y extensión ilimitadas, se establece un eslabón entre las operaciones materiales y los procesos mentales abstractos de la rama más teórica de la ciencia matemática.

Se desarrolla un lenguaje nuevo, amplio y poderoso, para su empleo futuro en el análisis, cuyas verdades se podrán manejar de modo que su aplicación sea más práctica y precisa para la humanidad de lo que hasta ahora han hecho las medidas a nuestro alcance...

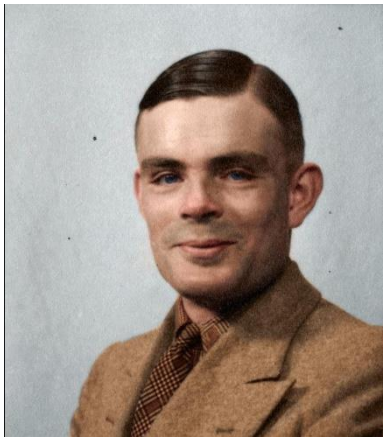
www.elintransigente.com/mundo/2012/12/10/maquina-analitica-ada-lovelace-160578.html

¿Cuándo comenzó el *software*?

- **Ada Lovelace** fue la primera persona en comprender que ***programar*** era algo nuevo, cuando escribía los primeros programas para la Máquina Analítica de Babbage. ¡1815 .. 1852!
- **Alan Turing** propuso el primer desarrollo de teórico de lo que hoy entendemos como crear código (instrucciones) para ***computar***.


Ada Lovelace


George Boole



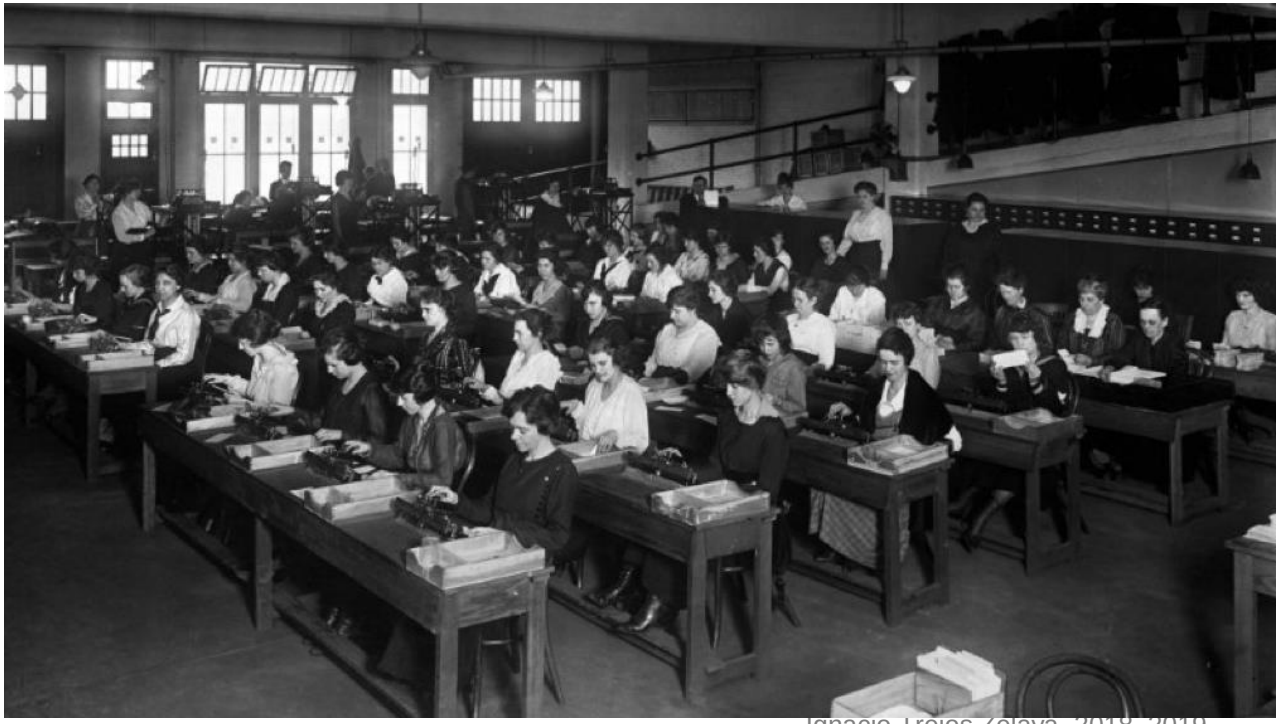
Las computadoras fueron humanas

Annie Cannon

Henrietta Leavitt

Frank&Lillian Gilbreth

Edith Clarke



PUNCHED CARD METHODS IN SCIENTIFIC COMPUTATION

By

W. J. ECKERT

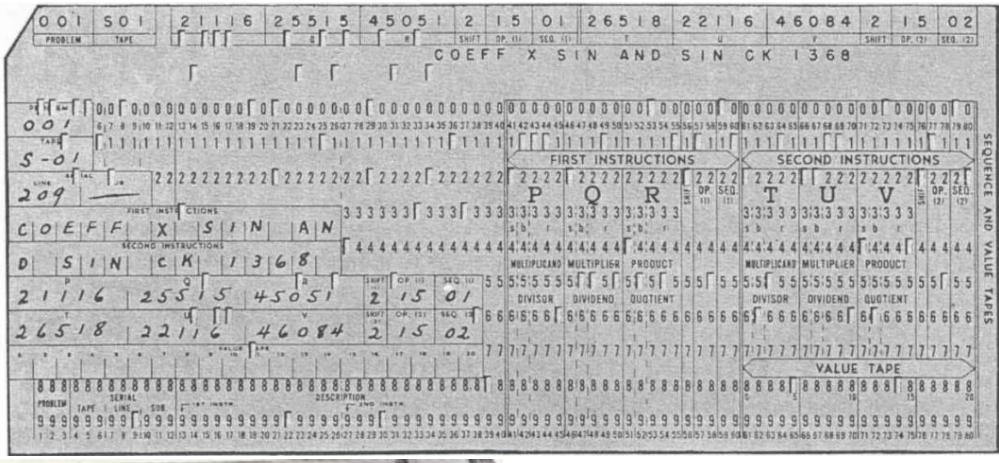
COLUMBIA UNIVERSITY

Published by

THE THOMAS J. WATSON ASTRONOMICAL COMPUTING BUREAU

COLUMBIA UNIVERSITY

January 1940



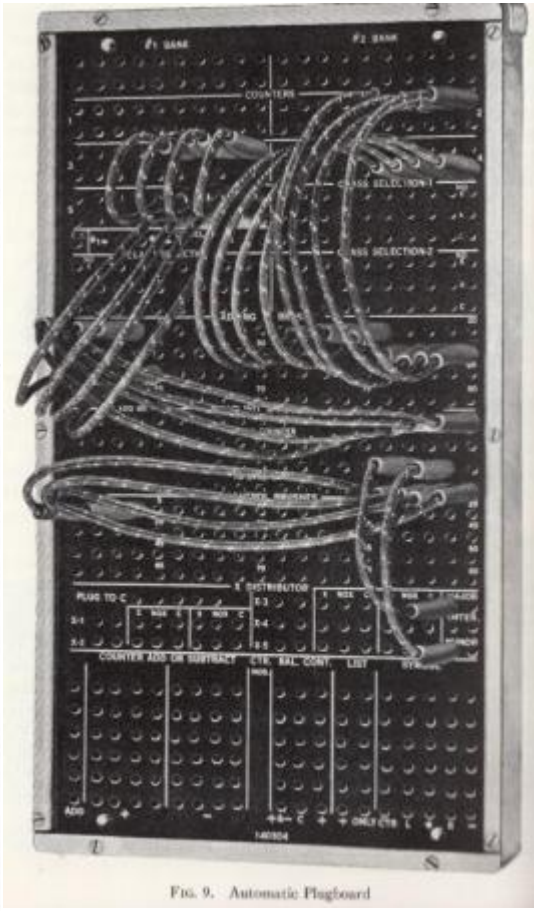


Fig. 9. Automatic Plugboard

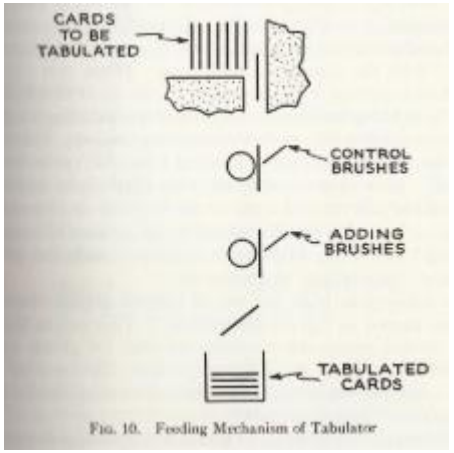
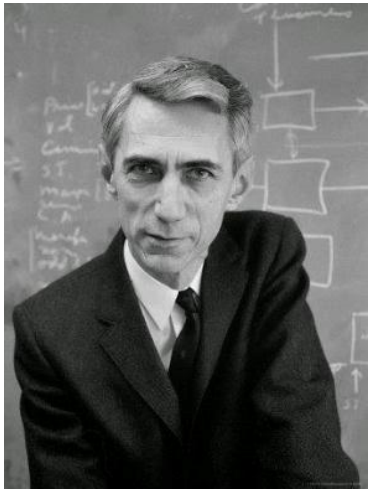
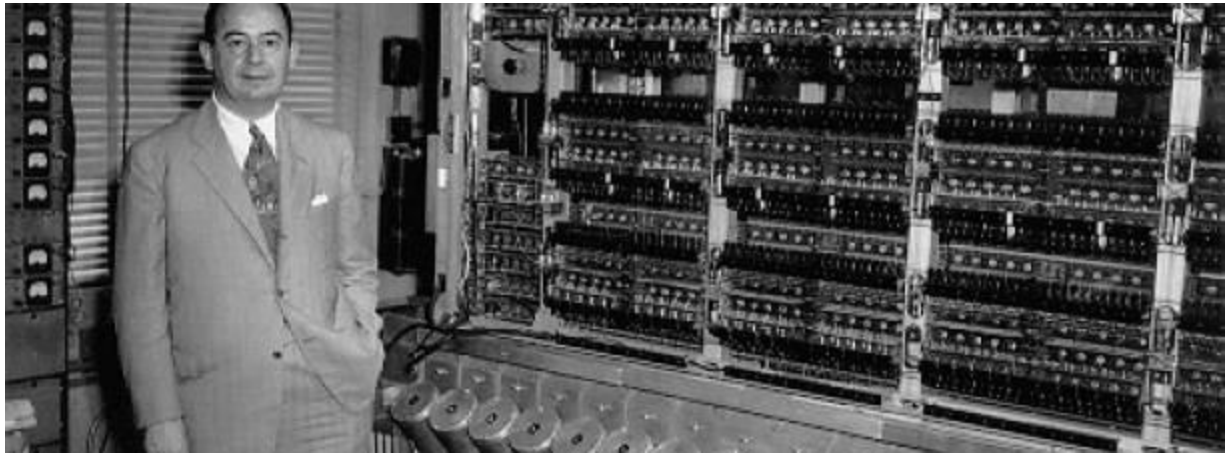


Fig. 10. Feeding Mechanism of Tabulator

¿Cuándo comenzó el *software*?

- **Claude Shannon** indicó cómo utilizar la lógica binaria (inspirada por George Boole) para ***codificar instrucciones*** para programar computadoras.
- **John von Neumann** propuso ***almacenar datos y programas en memoria*** de acceso directo.





¿Cuándo comenzó el *software*?

- **Tom Kilburn** escribió en 1948 el primer programa que pudo ser ***almacenado*** en una memoria electrónica.
- **Maurice Wilkes** publica en 1951 el primer libro sobre programación de computadoras, *The preparation of programs for an electronic digital computer: With special reference to the EDSAC and the use of a library of subroutines*. Él, David Wheeler y Stanley Gill inventaron las ***sub-rutinas*** como forma de abstracción. El libro incluía un capítulo denominado '*The Design of Programs For Electronic Computing Machines*'

Computadoras: grandes y aisladas


CRISIS
JUST AHEAD

Grace Murray Hopper

¿Cuándo comenzó el *software*?

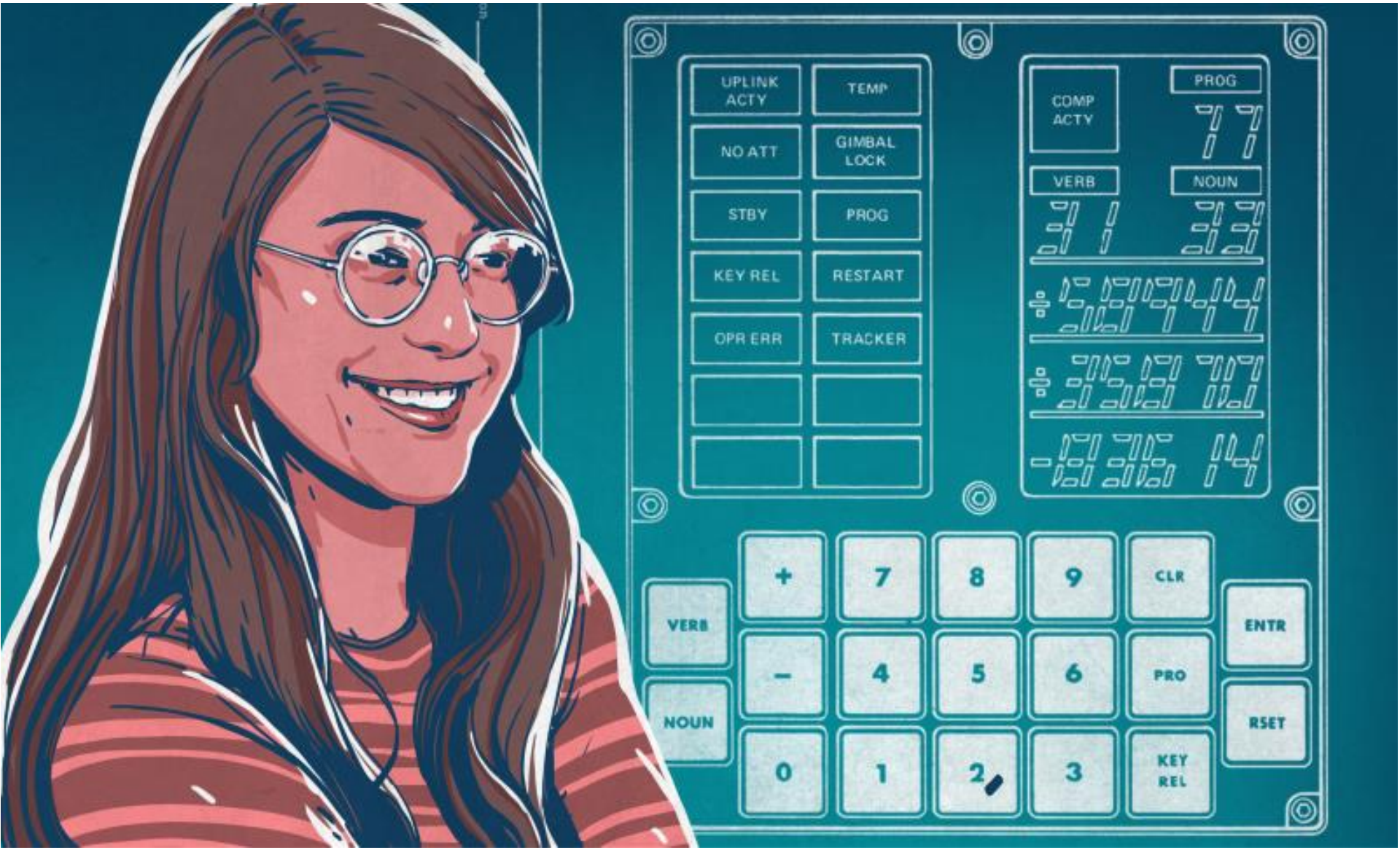
- **Grace Murray Hopper** inventó en 1952 los primeros ***compiladores*** para permitir la escritura de programas en lenguajes más cercanos a los humanos.
- El primer compilador de ***FORTRAN*** fue propuesto en 1953 por **John Backus** a IBM. Fue construido por él y sus colaboradores, para ser usado en abril de 1957; generaba código muy ***eficiente***.
- Hopper, Backus y colaboradores subieron el ***nivel de abstracción*** en la programación. Contribuyeron a aumentar la productividad de los programadores y la confiabilidad de sus programas.

¿Cuándo comenzó el *software*?

- Se atribuye al estadístico **John Tukey** el acuñar el término ***software*** en 1958. Sin embargo, el término había sido usado ya en 1953 por **Paul Niquette** y **Richard Carhart**.

John Tukey

Había *programadoras*



Ignacio Trejos Zelaya, 2018..2019

Margaret Hamilton

- "Estábamos creando un campo nuevo, no había ninguna institución que enseñara a programar. Cuando no podíamos hallar respuestas, debíamos inventarlas."
- "El mayor desafío con el software de los vuelos Apolo era que la vida de los astronautas dependía de que todo fuera ultraconfiable.“
- "Quería dar 'legitimidad' a nuestro *software* para que tanto el programa como quienes lo creaban gozaran del respeto debido"

Margaret Hamilton

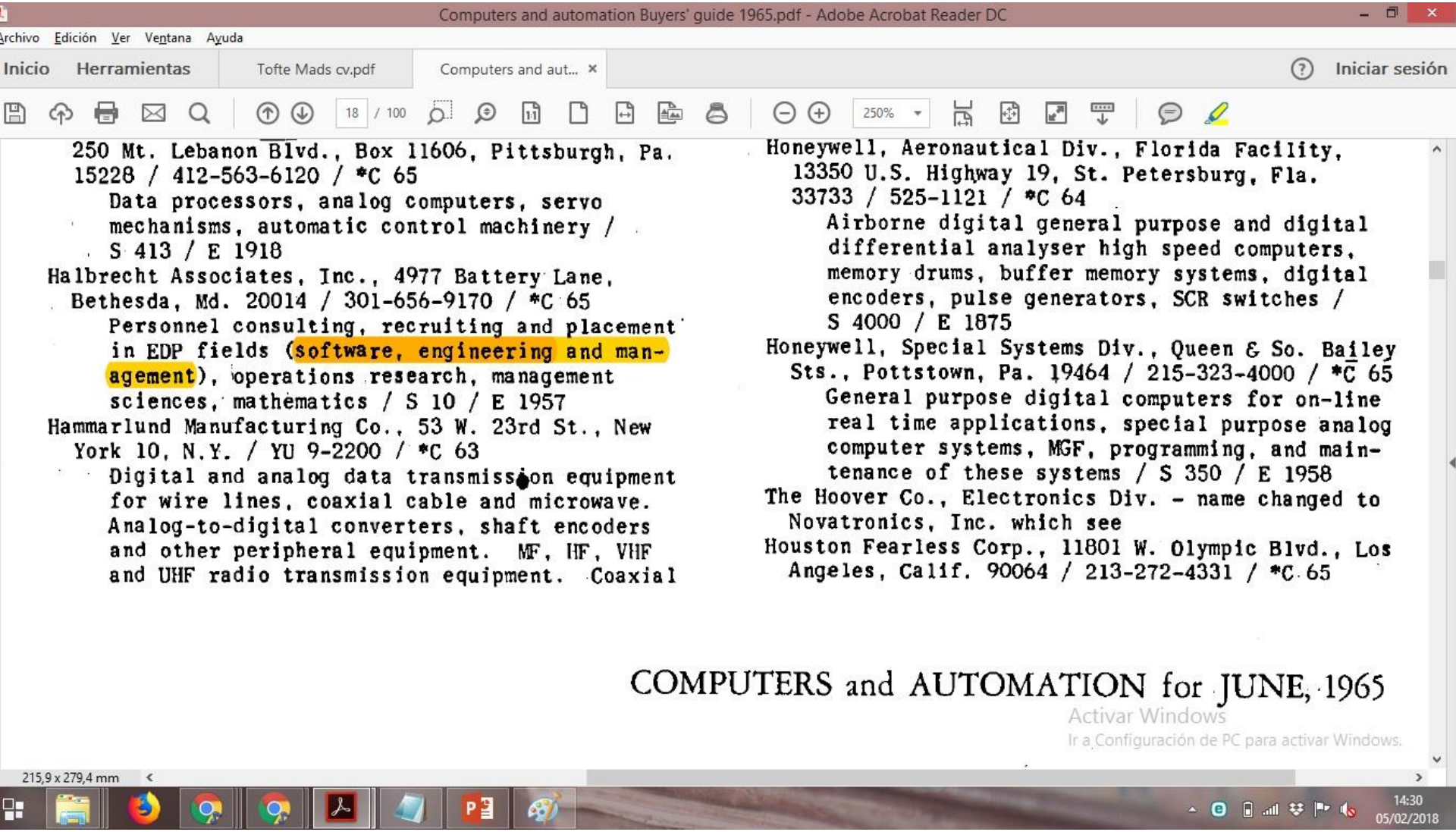
- "Comencé a llamar lo que hacíamos 'ingeniería de *software*' para que fuera reconocido como un tipo de ingeniería y al mismo tiempo como una disciplina propia"
- "Cuando se me ocurrió el término, **nadie lo había oído**. Por mucho tiempo fue motivo de bromas constantes. Para mí fue un día memorable cuando uno de los grandes gurúes del *hardware* dijo en una reunión que estaba de acuerdo conmigo, que crear *software* debería ser considerado un tipo de ingeniería"

Ignacio Trejos Zelaya, 2018..2019

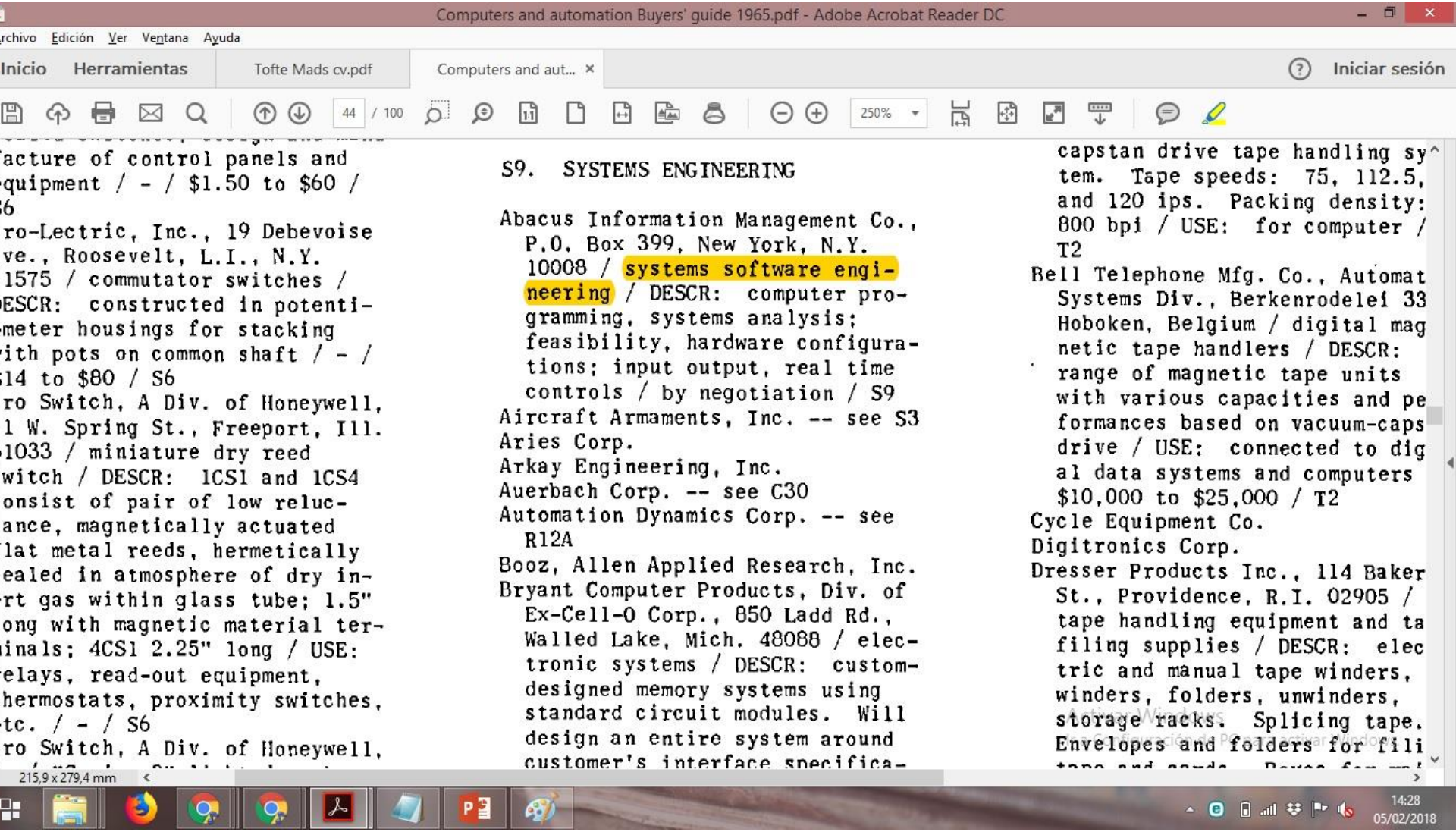
Margaret Hamilton

“When I first came up with the term, no one had heard of it before, at least in our world. It was an ongoing joke for a long time. They liked to kid me about my radical ideas. It was a memorable day when one of the most respected hardware gurus explained to everyone in a meeting that he agreed with me that the process of building software should also be considered an engineering discipline, just like with hardware. Not because of his acceptance of the new "term" per se, but because we had earned his respect and the acceptance of the others in the room as being in an engineering field in its own right.”

Computers and Automation, 1965.06, p.18



Computers and Automation, 1965.06, p.44



facture of control panels and equipment / - / $1.50 to $60 / S6
ro-Lectric, Inc., 19 Debevoise ve., Roosevelt, L.I., N.Y. 1575 / commutator switches / DESCR: constructed in potentiometer housings for stacking with pots on common shaft / - / $14 to $80 / S6
ro Switch, A Div. of Honeywell, 1 W. Spring St., Freeport, Ill. 1033 / miniature dry reed witch / DESCR: 1CS1 and 1CS4 onsist of pair of low reluctance, magnetically actuated lat metal reeds, hermetically ealed in atmosphere of dry inert gas within glass tube; 1.5" ong with magnetic material terinals; 4CS1 2.25" long / USE: elays, read-out equipment, hermostats, proximity switches, tc. / - / S6
ro Switch, A Div. of Honeywell,

S9. SYSTEMS ENGINEERING

Abacus Information Management Co., P.O. Box 399, New York, N.Y. 10008 / systems software engineering / DESCR: computer programming, systems analysis; feasibility, hardware configurations; input output, real time controls / by negotiation / S9
Aircraft Armaments, Inc. -- see S3
Aries Corp.
Arkay Engineering, Inc.
Auerbach Corp. -- see C30
Automation Dynamics Corp. -- see R12A
Booz, Allen Applied Research, Inc.
Bryant Computer Products, Div. of Ex-Cell-O Corp., 850 Ladd Rd., Walled Lake, Mich. 48088 / electronic systems / DESCR: custom-designed memory systems using standard circuit modules. Will design an entire system around customer's interface specifica-

capstan drive tape handling system. Tape speeds: 75, 112.5, and 120 ips. Packing density: 800 bpi / USE: for computer / T2
Bell Telephone Mfg. Co., Automat Systems Div., Berkenrodelei 33 Hoboken, Belgium / digital magnetic tape handlers / DESCR: range of magnetic tape units with various capacities and performances based on vacuum-caps drive / USE: connected to digal data systems and computers $10,000 to $25,000 / T2
Cycle Equipment Co.
Digitronics Corp.
Dresser Products Inc., 114 Baker St., Providence, R.I. 02905 / tape handling equipment and tafiling supplies / DESCR: electric and manual tape winders, winders, folders, unwinders, storage racks. Splicing tape. Envelopes and folders for fili

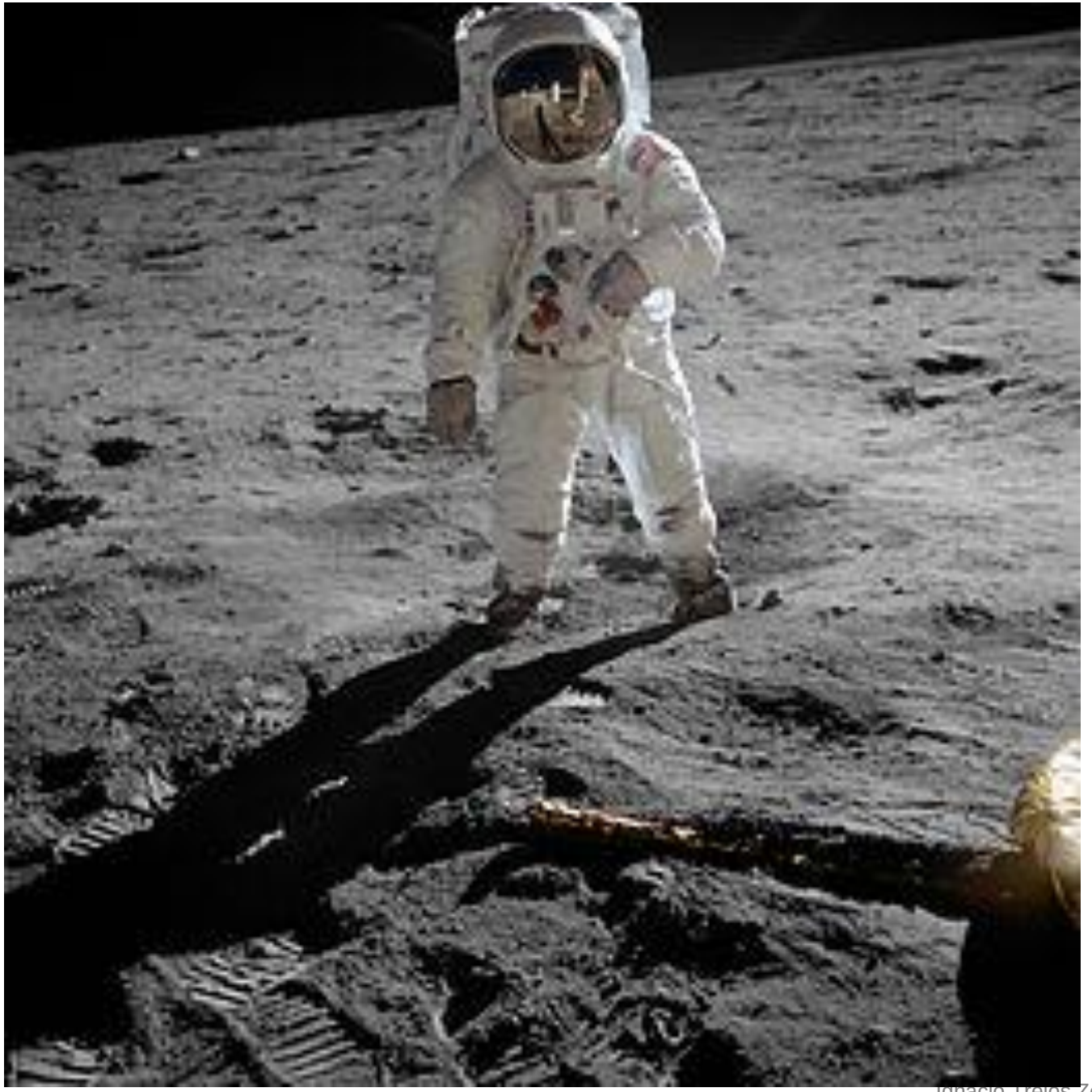
Anthony A. Oettinger, CACM, 1966.08

*A concern with the **science** of computing and information processing, while undeniably of the utmost importance and an historic root of our organization [the ACM] is, alone, too exclusive. While much of what we do is or has its root in not only computer and information science, but also many older and better defined sciences, even more is not at all scientific but of a professional and engineering nature. We must recognize ourselves – not necessarily all of us and not necessarily any one of us all the time – as members of an **engineering** profession, be it hardware engineering or **software engineering**, a profession without artificial and irrelevant boundaries like that between 'scientific' and 'business' applications*.

Ignacio Trejos Zelaya, 2018..2019

Margaret Hamilton

"El *software* no sólo tenía que funcionar, debía hacerlo de entrada. No había tiempo para pensar en ninguna otra cosa que no fuera hacer el trabajo a la perfección y hacerlo en el plazo casi imposible que había fijado el presidente Kennedy para llegar a la Luna"



THE SOFTWARE CRISIS

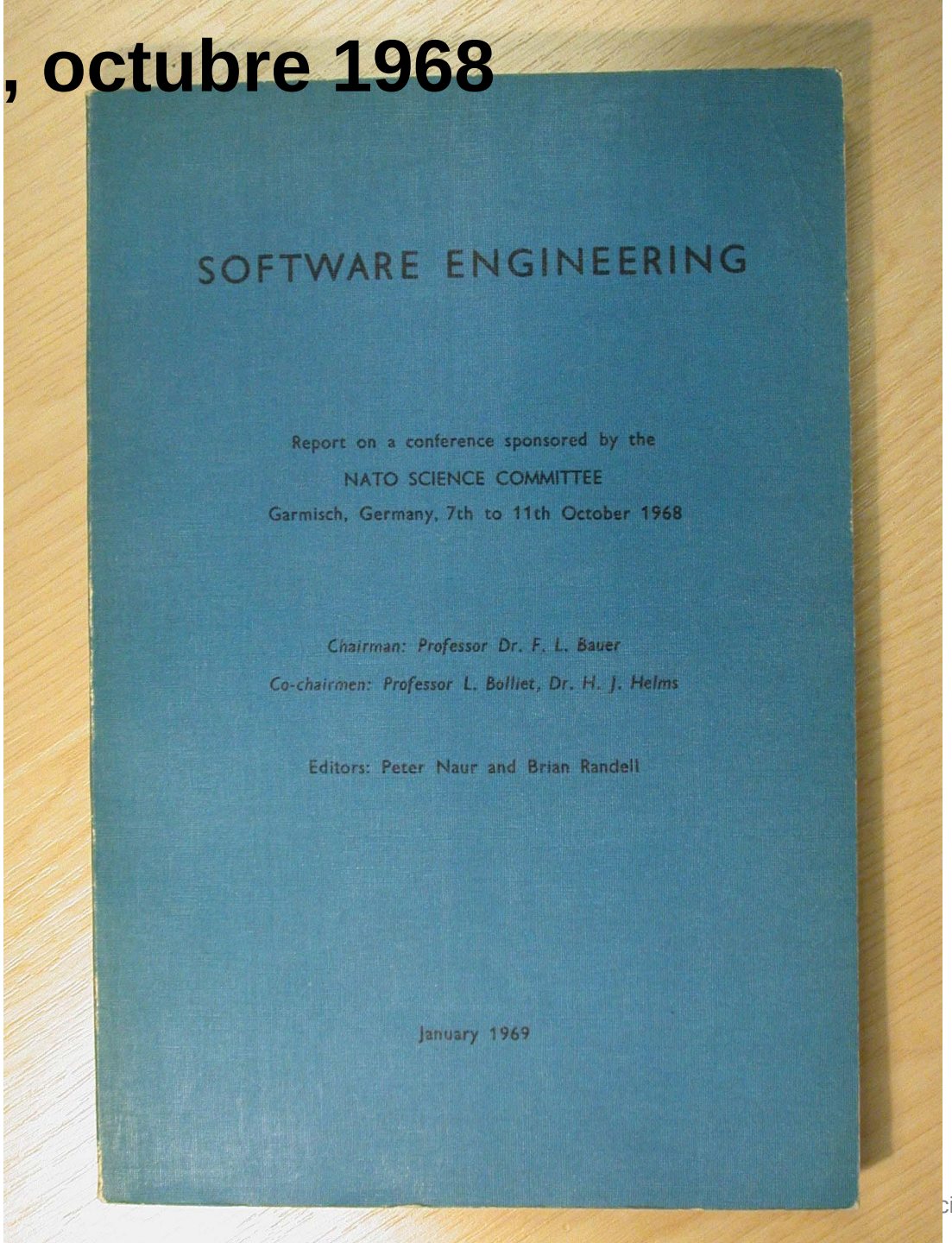
Fritz Bauer, 1967 .. 1968

- *Bauer was a colleague of the German Representative in the NATO Science Committee. In 1967, NATO had been discussing 'The Software Crisis' and Bauer had suggested the term 'Software Engineering' as a way to conceive of both the problem and the solution.*
- Brian Randell: *The idea for the first NATO* ***Software Engineering*** *Conference, and in particular that of adopting the then practically unknown term "**software engineering**" as its (deliberately provocative) title, I believe came originally from Professor Fritz Bauer*.

Fritz Bauer, definición

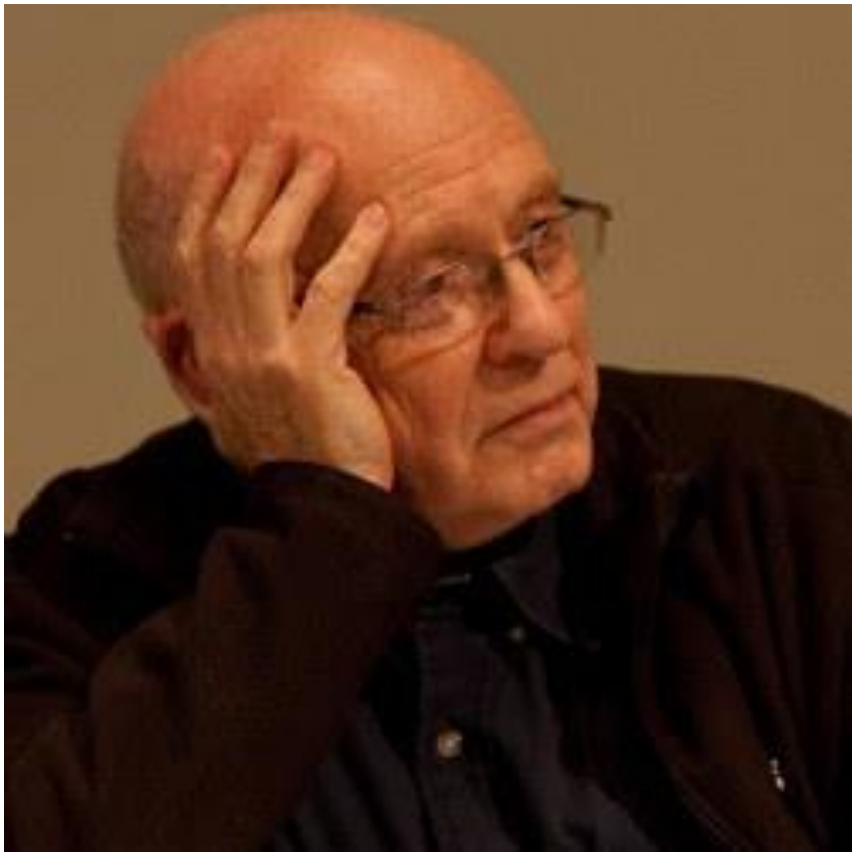
- F. Bauer: *Establishment and use of sound engineering principles to economically obtain software that is reliable and works on real machines efficiently*.

Garmisch, octubre 1968



Garmisch, octubre 1968

David L. Parnas



“Software engineering is often treated as a branch of computer science. This is akin to regarding chemical engineering as a branch of chemistry. We need both chemists and chemical engineers but they are very different. Chemists are scientists, chemical engineers are engineers. Software engineering and computer science have the same relationship.”

¿Qué es la ciencia?

- *La **ciencia** (del latín scientia, "conocimiento") es el conocimiento sistematizado elaborado mediante observaciones y razonamientos metódicamente organizados. La ciencia utiliza diferentes métodos y técnicas para la adquisición y organización de conocimientos sobre la estructura de un conjunto de hechos objetivos y accesibles a varios observadores. La aplicación de esos métodos y conocimientos conduce a la generación de más conocimiento objetivo en forma de predicciones concretas, cuantitativas y comprobables referidas a hechos observables pasados, presentes y futuros. Con frecuencia esas predicciones pueden formularse mediante razonamientos y estructurarse como reglas o leyes universales, que dan cuenta del comportamiento de un sistema y predicen cómo actuará dicho sistema en determinadas circunstancias.*

  Wikipedia (2008)

¿Qué es la tecnología?

- **Tecnología** es el conjunto de habilidades que permiten construir objetos y máquinas para adaptar el medio y satisfacer nuestras necesidades. Es una palabra de origen griego, τεχνολογος, formada por *tekne* (τεχνη, "arte, técnica u oficio") y *logos* (λογος, "conjunto de saberes").

  Wikipedia (2008)

- "*The application of science to the practical aims of human life or, as it is sometimes phrased, to the change and manipulation of the human environment. The term originally signified the study of or a discourse upon the arts, both fine and applied. By the late 20th. century it had come to mean the pursuit of results, especially useful results of scientific research, and it had become a global term connoting not only the tangible products of science but also the associated attitudes, processes, artifacts, and consequences*."

  Encyclopaedia Britannica (1990)

¿Qué es una ingeniería?

*La ingeniería nace tanto de la práctica (*fabrica*) como de la teoría (*ratiocinatio*). La práctica es una facilidad mecánica, desarrollada mediante el estudio y el ejercicio. La teoría es la capacidad de describir y explicar el objeto diseñado.*

Marcus Vitruvius Pollio, circa 25 AC

La Ingeniería es …

"*Crear soluciones eficaces respecto del costo …* La Ingeniería no es solamente acerca de la resolución de problemas; es acerca de resolver problemas con un uso económico de los recursos, incluyendo el dinero. *… a problemas prácticos …* La Ingeniería trata de problemas prácticos cuyas soluciones importan a gente fuera del dominio ingenieril – los clientes. *… mediante la aplicación de conocimiento científico …* La Ingeniería resuelve problemas de una manera particular: aplicando ciencias, matemática y análisis de diseños. *… a la construcción de cosas…* La Ingeniería hace énfasis en las soluciones, que usualmente son artefactos tangibles. *… al servicio de la Humanidad.* La Ingeniería sirve no solo al cliente inmediato, sino que desarrolla tecnología y pericia que apoya a la sociedad."

Mary Shaw, 1990

Los ingenieros

- Un ingeniero aplica conocimiento teórico y científico en el diseño y construcción de artefactos de valor económico para utilización práctica.
- El fundamento teórico permite verificar, mediante cálculos sistemáticos y *antes* de construir el artefacto, que el diseño satisfará las especificaciones, las restricciones técnicas y las limitaciones de recursos.
- Adquieren responsabilidad ante los clientes y el público.

Educación de científicos e ingenieros

Ciencia

(de la Computación)

versus

Ingeniería

(en Sistemas, en Informática, en Computación, del software, en Telemática)

Ciencia – David L. Parnas

- Los futuros científicos, quienes añadirán a nuestro cuerpo de conocimientos, deben aprender:
  - Lo que es cierto (cuerpo de conocimiento organizado acerca de los fenómenos de interés)
  - Cómo confirmar o refutar modelos del mundo
  - Cómo extender el conocimiento acerca de lo que es cierto en su campo

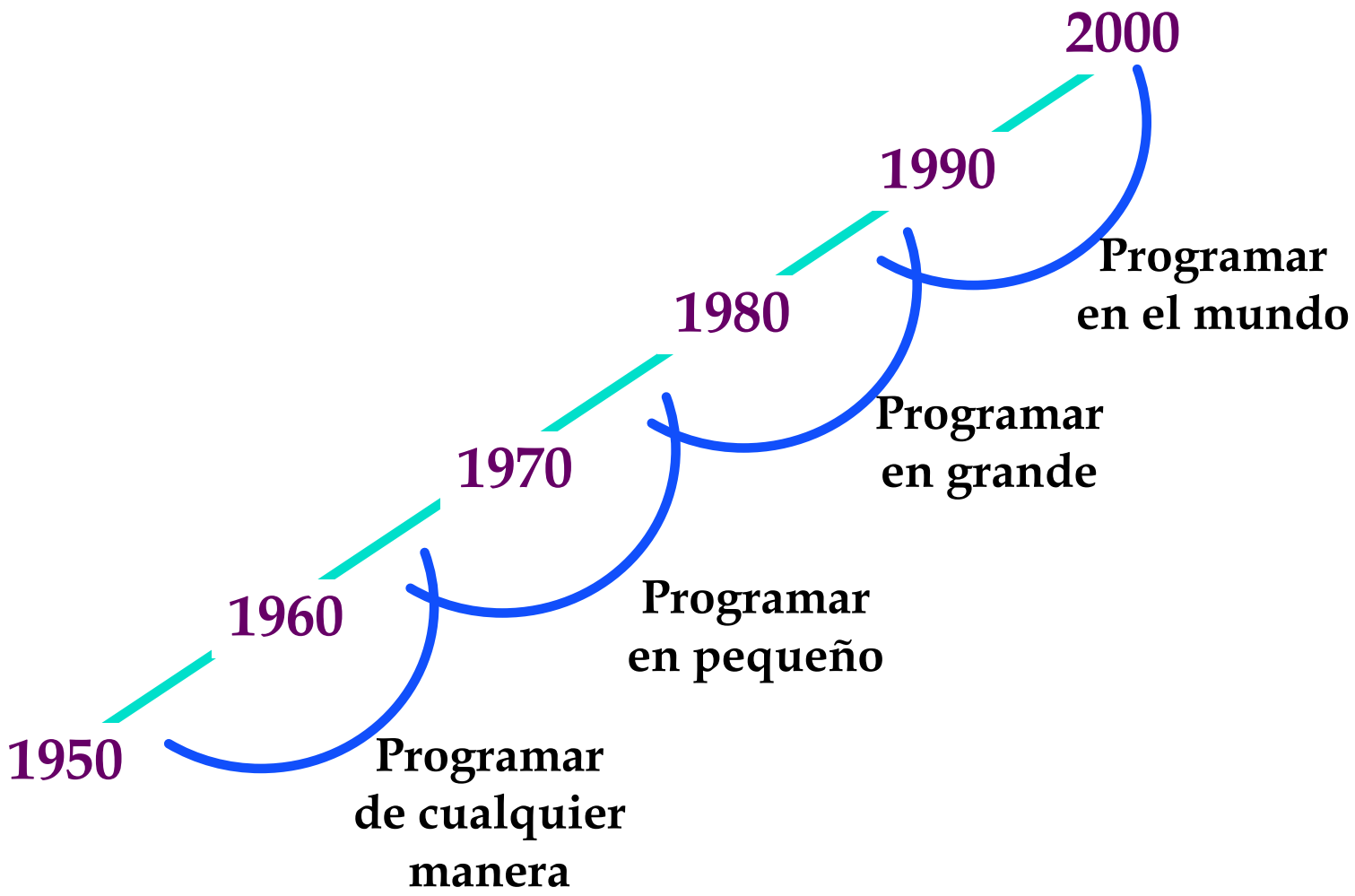
Ingeniería – David L. Parnas

- Los futuros ingenieros, quienes diseñarán productos dignos de confianza, deben aprender
  - Lo que es cierto y verdadero de su especialidad (el cuerpo organizado de conocimientos)
  - Cómo aplicar ese cuerpo de conocimientos
  - Cómo aplicar un área más amplia de conocimientos, necesaria para construir productos completos que funcionan en ambientes realistas
  - La disciplina de análisis y diseño que debe seguirse para cumplir con las responsabilidades atinentes a aquellos que construyen productos para otros

Finkelstein y Kramer, 2000

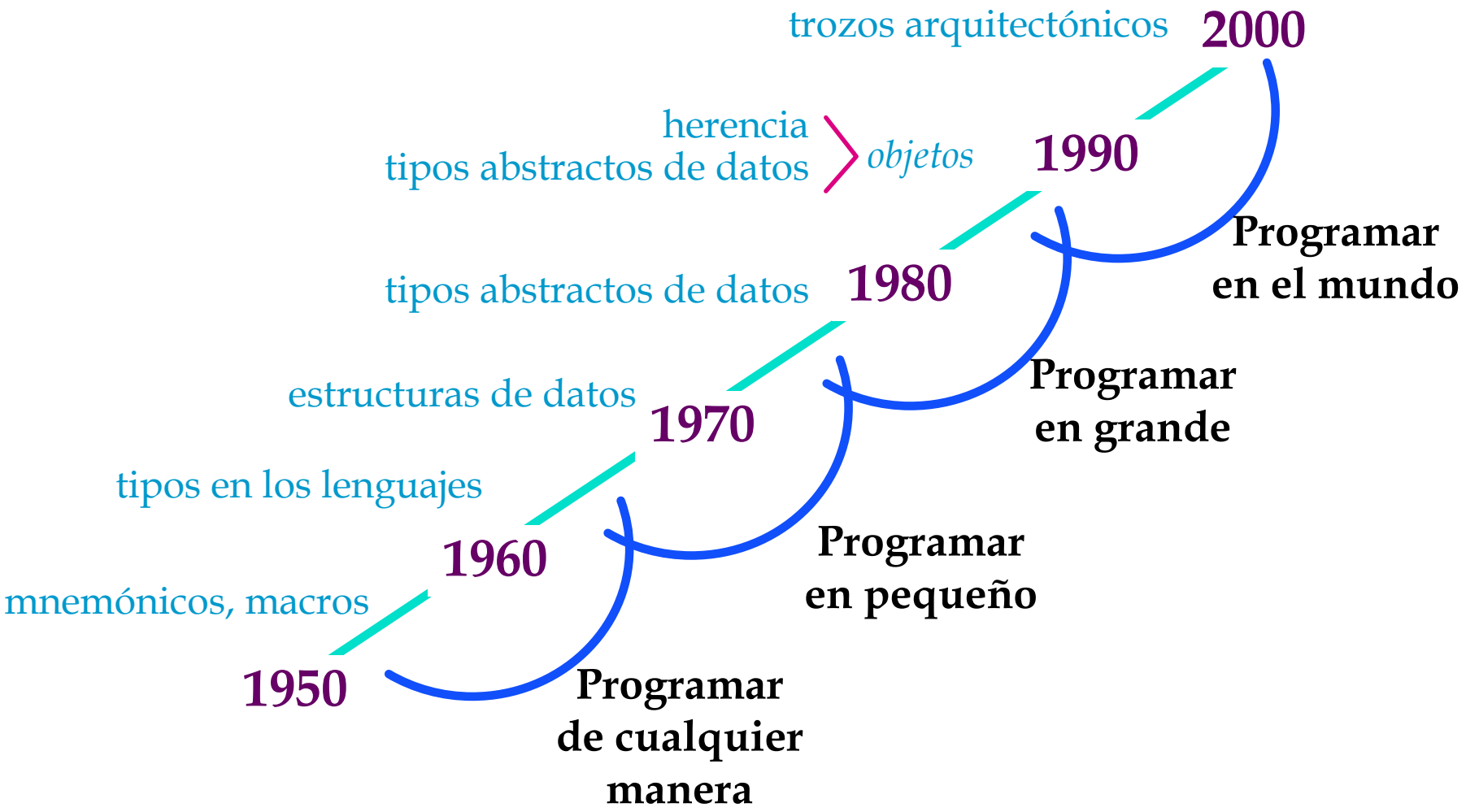
- “la rama de la Ingeniería de sistemas que trata del desarrollo de grandes y complejos sistemas intensivos en software. Se enfoca en:
  - las metas del mundo real de tales sistemas, así como los servicios que estos proveen y las restricciones sobre ellos
  - la especificación precisa del comportamiento y la estructura de los sistemas, y la implementación de estas especificaciones
  - las actividades requeridas a fin de desarrollar un aseguramiento de que se han satisfecho las especificaciones y las metas del mundo real
  - la evolución de tales sistemas en el tiempo y a través de familias de sistemas
  - los procesos, métodos y herramientas para el desarrollo de sistemas intensivos en software de una manera económica y oportuna.”

Evolución de la programación, circa 1990



Mary Shaw

Evolución de la abstracción, circa 1990



Mary Shaw

Desarrollos desde 1990, Mary Shaw

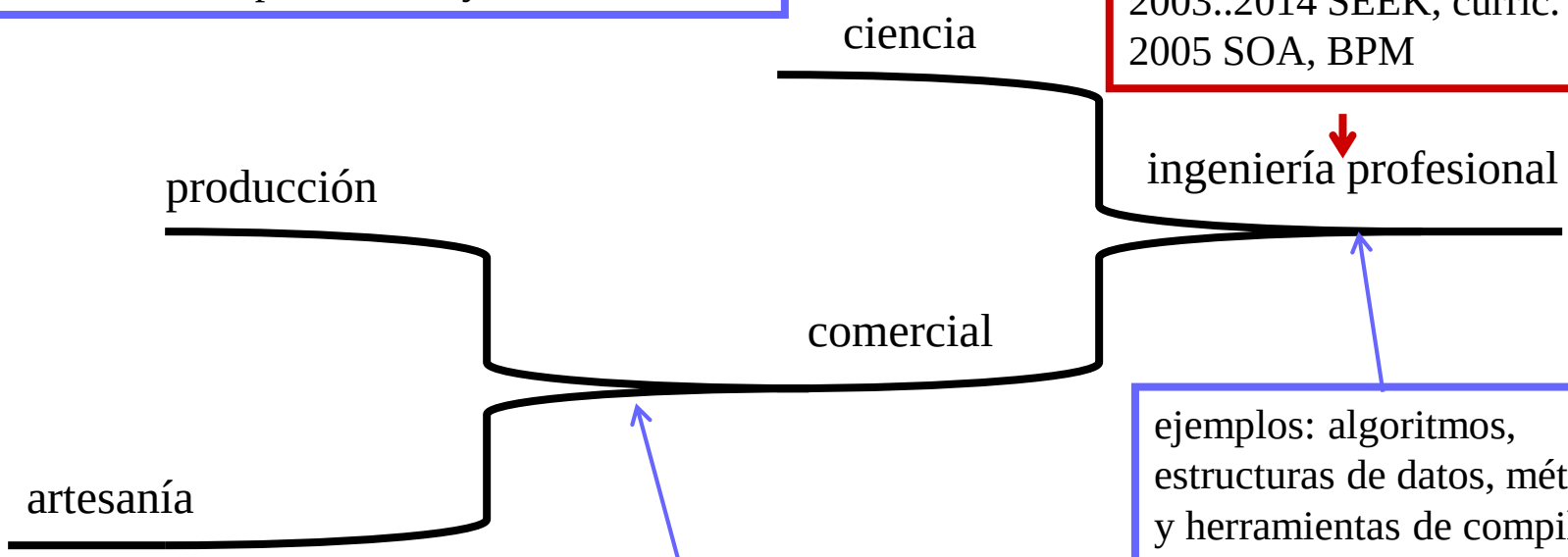
- 1990 ± 5: arquitecturas abstractas, estilos arquitectónicos, patrones de diseño, componentes y conectores.
- 2000 ± 5: Internet y Web como sustrato potenciador para software distribuido, interoperable
- 2010 ± 5: fronteras entre sistemas desaparecen, integración horizontal y vertical

Mary Shaw

Evolución del desarrollo de software

1965..1970: algoritmos y estructuras de datos
1960..1980: análisis sintáctico
1961..1995: semántica de lenguajes de programas
1970..1987: modelos de datos
1968..1993: verificación y cálculo de programas
1974.. : especificación y refinamiento formal
1976..: criptografía de clave pública
1978..: concurrencia, paralelismo y distribución

1987 CMM, ISO 9000
1993 arquitecturas
1994 patrones de diseño
1995 marcos de problema
1997 UML y OCL
1998..20014 SWEBOK
2002 MDA, MDD, MDSE
2003..2014 SEEK, curric.
2005 SOA, BPM



1965..1989: métodos estructurados de desarrollo de software

ejemplos: algoritmos, estructuras de datos, métodos y herramientas de compilación, diseño de bases de datos, protocolos, modelaje de rendimiento, automatización de pruebas

Mary Shaw (actualizado por ITZ)

Hitos en la Ingeniería del software

- Lenguajes y paradigmas de programación
- Sistemas administradores de bases de datos
- Sistemas operativos
- Virtualización
- Esconder información
- Modularidad
- Jerarquía

Hitos en la Ingeniería del software

- Programación estructurada
  - Diseño estructurado
    - Análisis estructurado
- Programación orientada a objetos
  - Diseño orientado a objetos
    - Análisis orientado a objetos
- Modelaje de software
- Métodos para análisis y diseño de sistemas de software

Evolución de los métodos

- Lenguaje y modelo computacional
- Técnicas de programación
- Técnicas y notaciones de diseño
- Técnicas y notaciones de análisis
- Proceso integrado análisis $\rightarrow$ diseño $\rightarrow$ programa
- Consultores
- Libros
- Catálogos de patrones
- Herramientas

Hitos en la Ingeniería del software

- Arquitecturas de software
- Patrones de diseño de software
- Componentes de software
- Reutilización de software
- Tecnologías Web
- Tecnologías móviles
- Tecnologías de Nube
- Arquitecturas orientadas a servicios
- Orquestación de servicios
- Microservicios
- Aplicaciones distribuidas

Hitos en la Ingeniería del software

- Ingeniería de requerimientos de software
- Gestión de requerimientos de software
- Gestión de configuraciones de software
- Procesos de software
- Gestión de riesgos en software
- Administración de proyectos de software
- Medición de software
- Modelos de estimación de esfuerzo, tamaño

Hitos en la Ingeniería del software

- Gestión de la calidad del software
- Gestión de la productividad en el desarrollo, la integración y el mantenimiento de software
- Procesos iterativos, incrementales, evolutivos, ágiles
- Modelos de madurez de capacidades
- Mejora de procesos de software

Hitos en la Ingeniería del software

- Software de código abierto
- Herramientas para colaboración distribuida
- Integración y despliegue continuos
- Máquinas virtuales
- Contenedores de software
- Tecnología de Nube
- *DevOps*

Hitos en la Ingeniería del software

- Especializaciones por ***dominios*** *de problema, de aplicación* o *de tecnología*
- Métodos formales para especificar software
- Modelaje formal de sistemas secuenciales, concurrentes, distribuidos y móviles
- Refinemiento de software
- Sistemas formales para razonamiento, verificación y desarrollo
- Modelaje cuantitativo de rendimiento, escalabilidad, confiabilidad, disponibilidad

Herramientas

- Modelaje gráfico (visual, diagramático)
- Análisis estático y de impacto de cambios
- Verificación formal
- Generadores de código
- Adaptadores de interfaces
- Apoyo al refinamiento y la integración

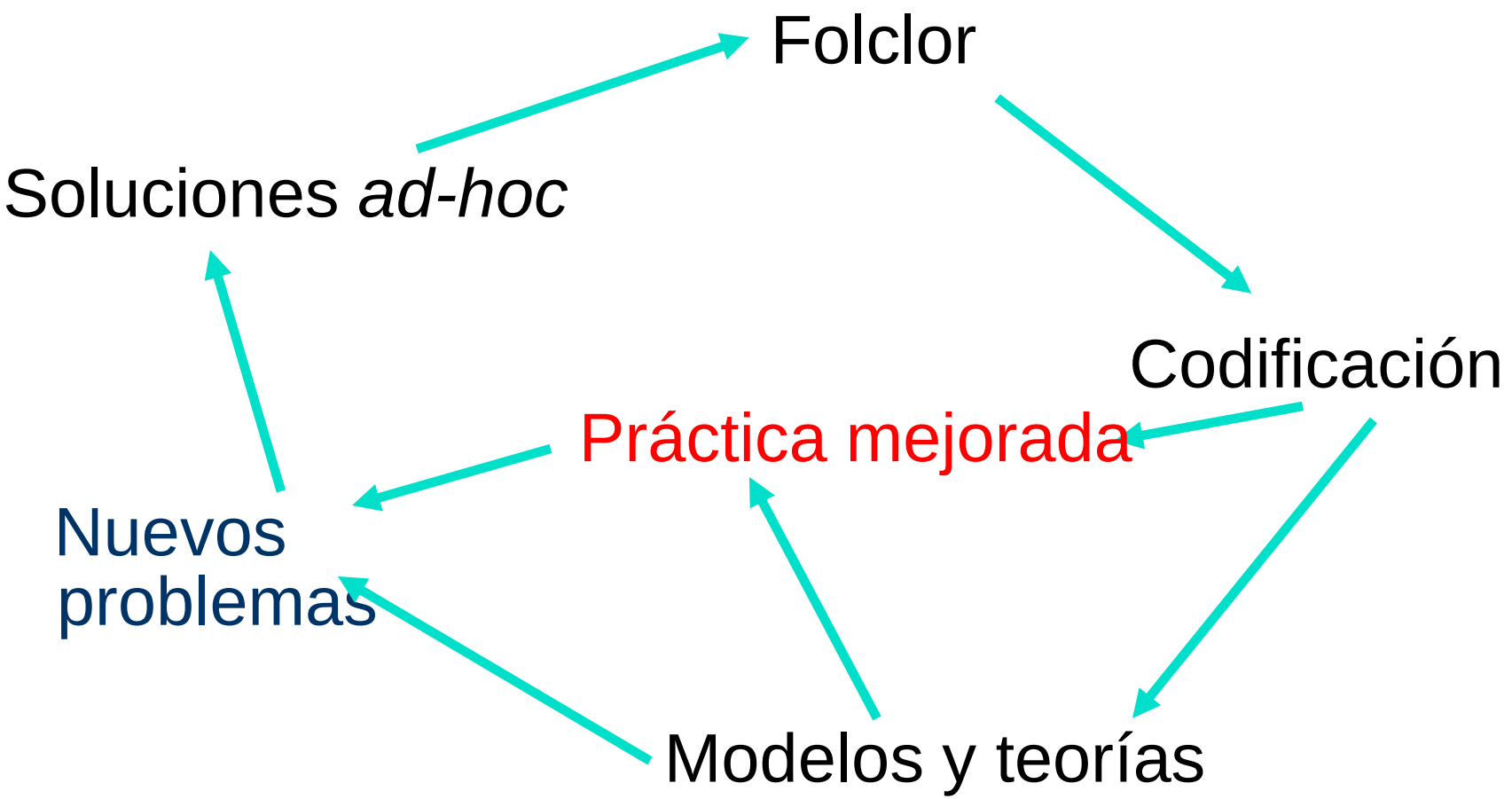
Herramientas

- Generación de casos de prueba
- Análisis de cobertura de pruebas
- Gestión de configuración, manejo de versiones
- Navegadores y buscadores de componentes
- Monitoreo y medición de rendimiento
- Generación de documentación

Principios

- Formular buenas abstracciones
- Diseñar claras estructuras y relaciones
- Mantener una clara separación de intereses
- Lograr una distribución balanceada de responsabilidades
- Buscar la simplicidad
- Cultivar un sistema mediante la liberación de arquitecturas ejecutables de manera iterativa, incremental y continua

Madurez: codificación progresiva

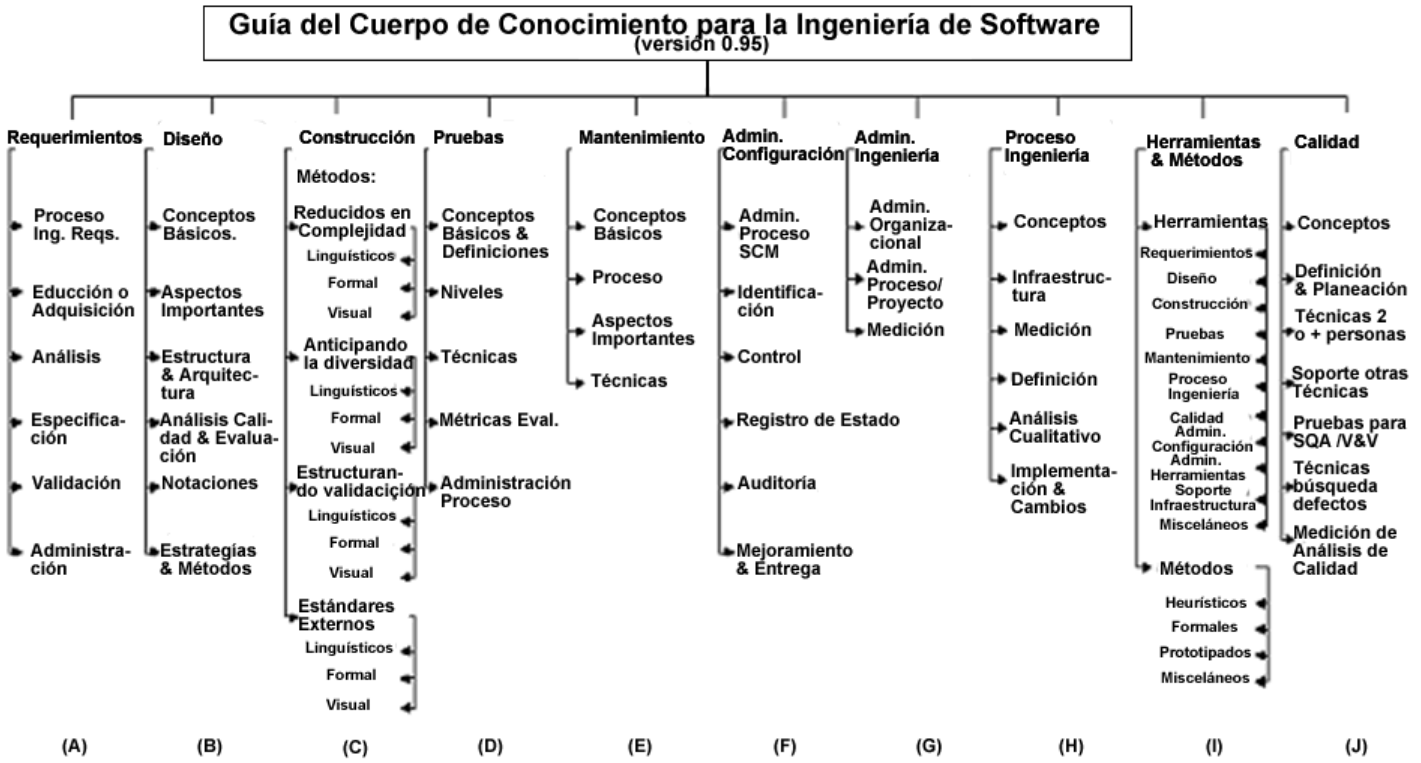


Shaw

Estado de madurez profesional

Inexistente (0)	*Ad-hoc* (1)	Específico (2)	Maduro (3)
		Cuerpo de conocimientos	
		Asociaciones profesionales	
	Código de ética		
	Sistema educativo inicial		
		Acreditación de programas de educación profesional	
	Desarrollo de habilidades profesionales		
	Educación profesional continua		
	Certificación de profesionales		
	Licenciamiento de profesionales		

SWEBOK 2004


Guía del Cuerpo de Conocimiento para la Ingeniería de Software
(versión 0.95)
Requerimientos
Proceso Ing. Reqs.
Educción o Adquisición
Análisis
Especifica-ción
Validación
Administra-ción
Diseño
Conceptos Básicos.
Aspectos Importantes
Estructura & Arquitec-tura
Análisis Cali-dad & Evalua-ción
Notaciones
Estrategias & Métodos
Construcción
Métodos:
Reducidos en Complejidad
Linguísticos
Formal
Visual
Anticipando la diversidad
Linguísticos
Formal
Visual
Estructuran-do validación
Linguísticos
Formal
Visual
Estándares Externos
Linguísticos
Formal
Visual
Pruebas
Conceptos Básicos & Definiciones
Niveles
Técnicas
Métricas Eval.
Administración Proceso
Mantenimiento
Conceptos Básicos
Proceso
Aspectos Importantes
Técnicas
Admin. Configuración
Admin. Proceso SCM
Identifica-ción
Control
Registro de Estado
Auditoría
Mejoramiento & Entrega
Admin. Ingeniería
Admin. Organiza-cional
Admin. Proceso/ Proyecto
Medición
Proceso Ingeniería
Conceptos
Infraestruc-tura
Medición
Definición
Análisis Cualitativo
Implementa-ción & Cambios
Herramientas & Métodos
Herramientas
Requerimientos
Diseño
Construcción
Pruebas
Mantenimiento
Proceso Ingeniería
Calidad Admin.
Configuración Admin.
Herramientas
Soporte Infraestructura
Misceláneos
Métodos
Heurísticos
Formales
Prototipados
Misceláneos
Calidad
Conceptos
Definición & Planeación
Técnicas 2 o + personas
Soporte otras Técnicas
Pruebas para SQA /V&V
Técnicas búsqueda defectos
Medición de Análisis de Calidad
(A)
(B)
(C)
(D)
(E)
(F)
(G)
(H)
(I)
(J)

SWEBOK v3 (2014) – KAs y relaciones

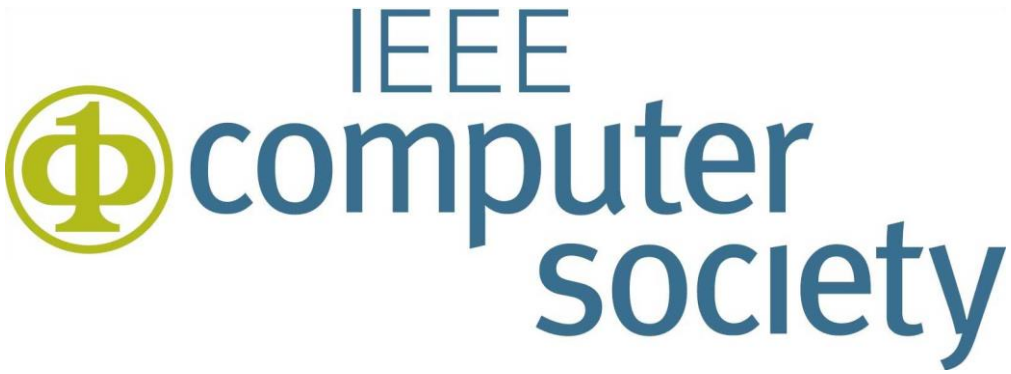
Table I.1. The 15 SWEBOK KAs
Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Models and Methods
Software Quality
Software Engineering Professional Practice
Software Engineering Economics
Computing Foundations
Mathematical Foundations
Engineering Foundations

Table I.2. Related Disciplines
Computer Engineering
Computer Science
General Management
Mathematics
Project Management
Quality Management
Systems Engineering

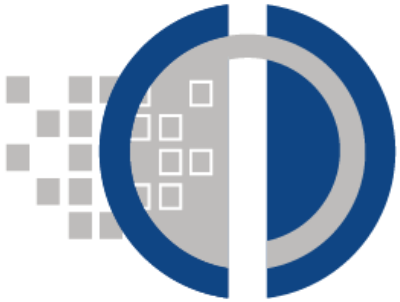
SEEK 2014

KA/KU	Title	Hours	KA/KU	Title	Hours
CMP	**Computing essentials**	**152**	**DES**	**Software design**	**48**
CMP.cf	Computer science foundations	120	DES.con	Design concepts	3
CMP.ct	Construction technologies	20	DES.str	Design strategies	6
CMP.tl	Construction tools	12	DES.ar	Architectural design	12
			DES.hci	Human-computer interaction design	10
			DES.dd	Detailed design	14
			DES.ev	Design evaluation	3
FND	**Mathematical and engineering fundamentals**	**80**	**VAV**	**Software verification and validation**	**37**
FND.mf	Mathematical foundations	50	VAV.fnd	V&V terminology and foundations	5
FND.ef	Engineering foundations for software	22	VAV.rev	Reviews and static analysis	9
FND.ec	Engineering economics for software	8	VAV.tst	Testing	18
			VAV.par	Problem analysis and reporting	5
PRF	**Professional practice**	**29**	**PRO**	**Software process**	**33**
PRF.psy	Group dynamics and psychology	8	PRO.con	Process concepts	3
PRF.com	Communications skills (specific to SE)	15	PRO.imp	Process implementation	8
PRF.pr	Professionalism	6	PRO.pp	Project planning and tracking	8
			PRO.cm	Software configuration management	6
			PRO.evo	Evolution processes and activities	8
MAA	**Software modeling and analysis**	**28**	**QUA**	**Software quality**	**10**
MAA.md	Modeling foundations	8	QUA.cc	Software quality concepts and culture	2
MAA.tm	Types of models	12	QUA.pca	Process assurance	4
MAA.af	Analysis fundamentals	8	QUA.pda	Product assurance	4
REQ	**Requirements analysis and specification**	**30**	**SEC**	**Security**	**20**
REQ.rfd	Requirements fundamentals	6	SEC.sfd	Security fundamentals	4
REQ.er	Eliciting requirements	10	SEC.net	Computer and network security	8
REQ.rsd	Requirements specification and documentation	10	SEC.dev	Developing secure software	8
REQ.rv	Requirements validation	4			

Asociaciones profesionales



Association for
Computing Machinery

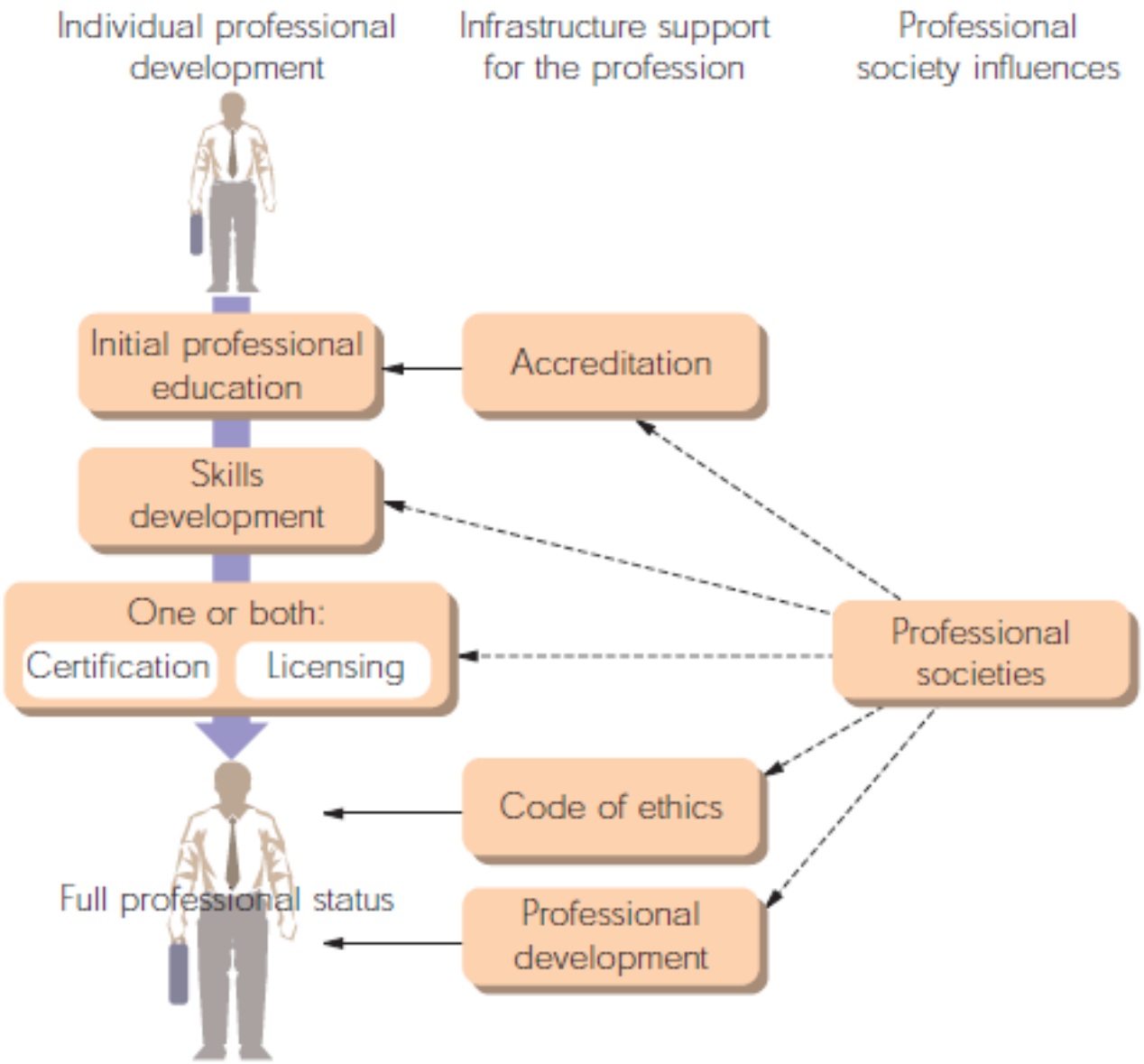


CPIC

COLEGIO DE PROFESIONALES
EN INFORMÁTICA Y COMPUTACIÓN

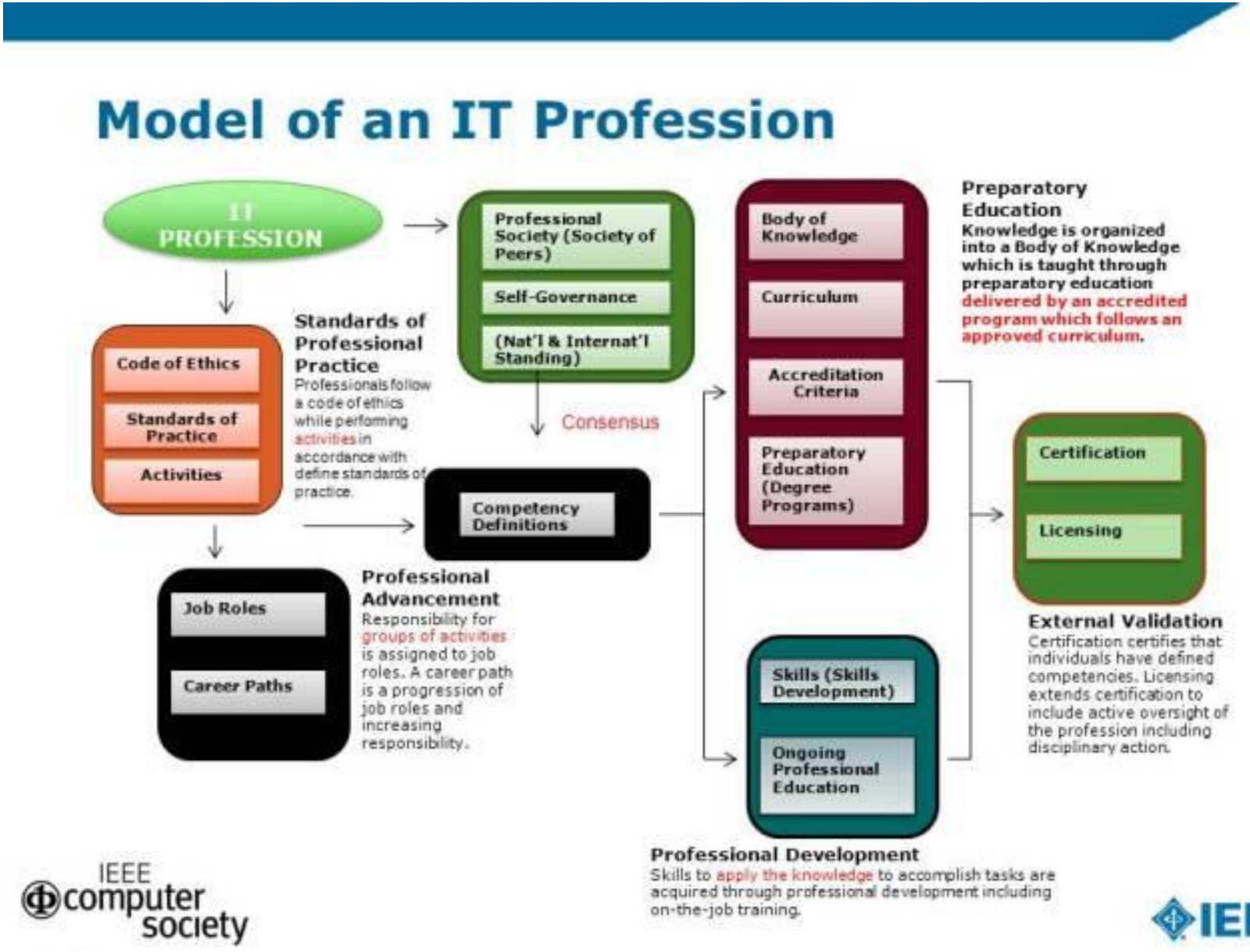
AUSTRALIAN
COMPUTER
SOCIETY

Desarrollo profesional



McConnell, 1999

Modelo de una profesión de TI



Código de ética

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

1 PUBLIC - Software engineers shall act consistently with the public interest.

2 CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.

3 PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

4 JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.

5 MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

6 PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

7 COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.

8 SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Certificaciones profesionales

Why IEEE certification?

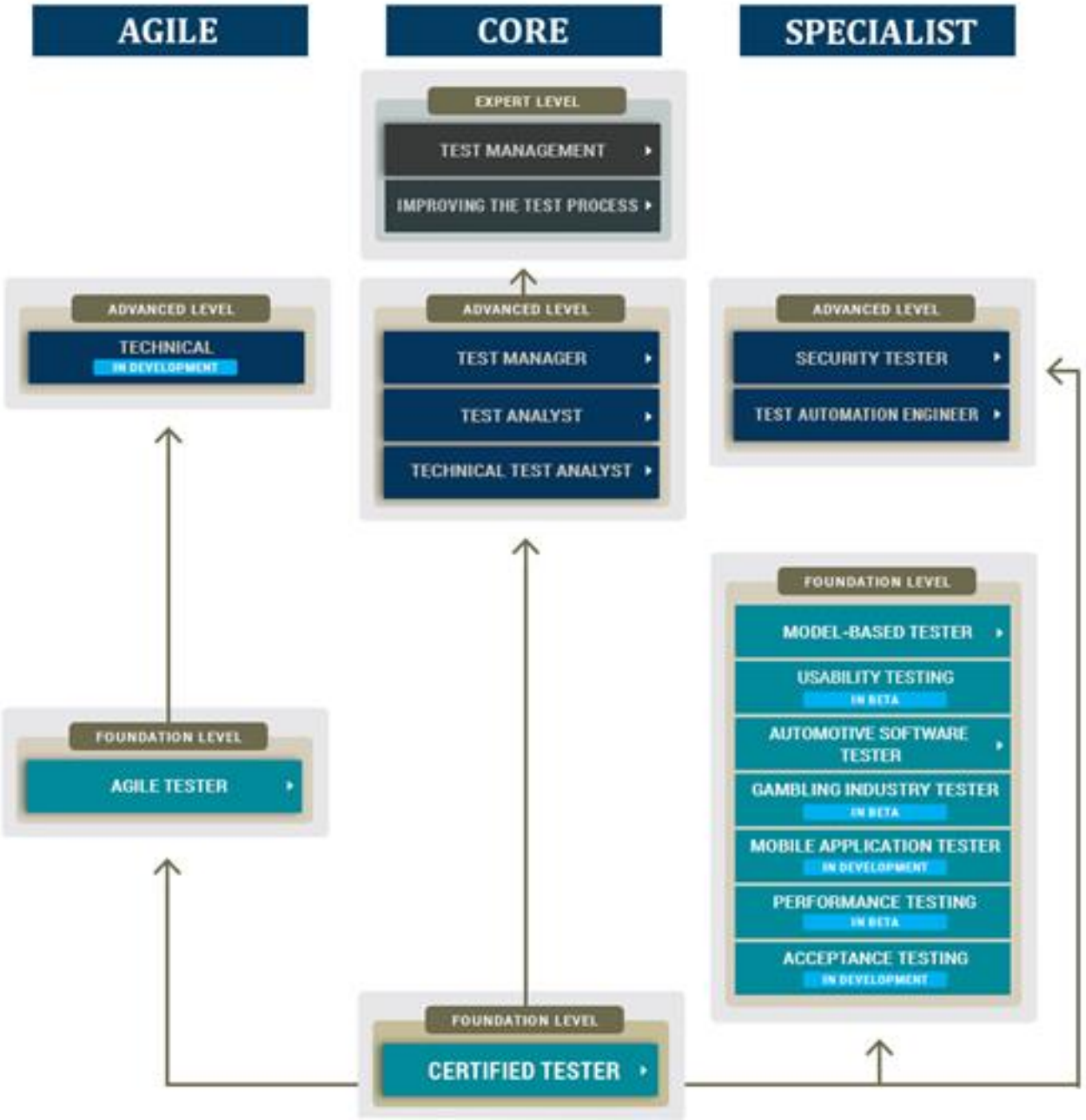
world's largest professional association

IEEE COMPUTER SOCIETY

creators of SWEBOK

vendor neutral

Certificaciones (pruebas de software)



Niels Bohr (1885 .. 1962)

Predecir es muy difícil, y sobre todo el futuro.

Niels Henrik David Bohr
Premio Nobel de Física, 1922

Tendencias

- (Invención [instalación] → Adopción [despliegue])*
- Innovación ágil de procesos, productos y servicios
- Complejidad => Especialización
- Ubicuidad → Movilidad → IoT
- ↑ automatización => ↑ demanda de personal
- ↑ conocimiento profundo de dominios
- ↑ modelaje de datos y dominios
- Esp. Inteligencia Artificial (o ‘sistemas inteligentes’)
- Esp. Sistemas ciberfísicos y empotrados
- Esp. Sistemas distribuidos a gran escala
- Esp. Analítica de datos

Retos

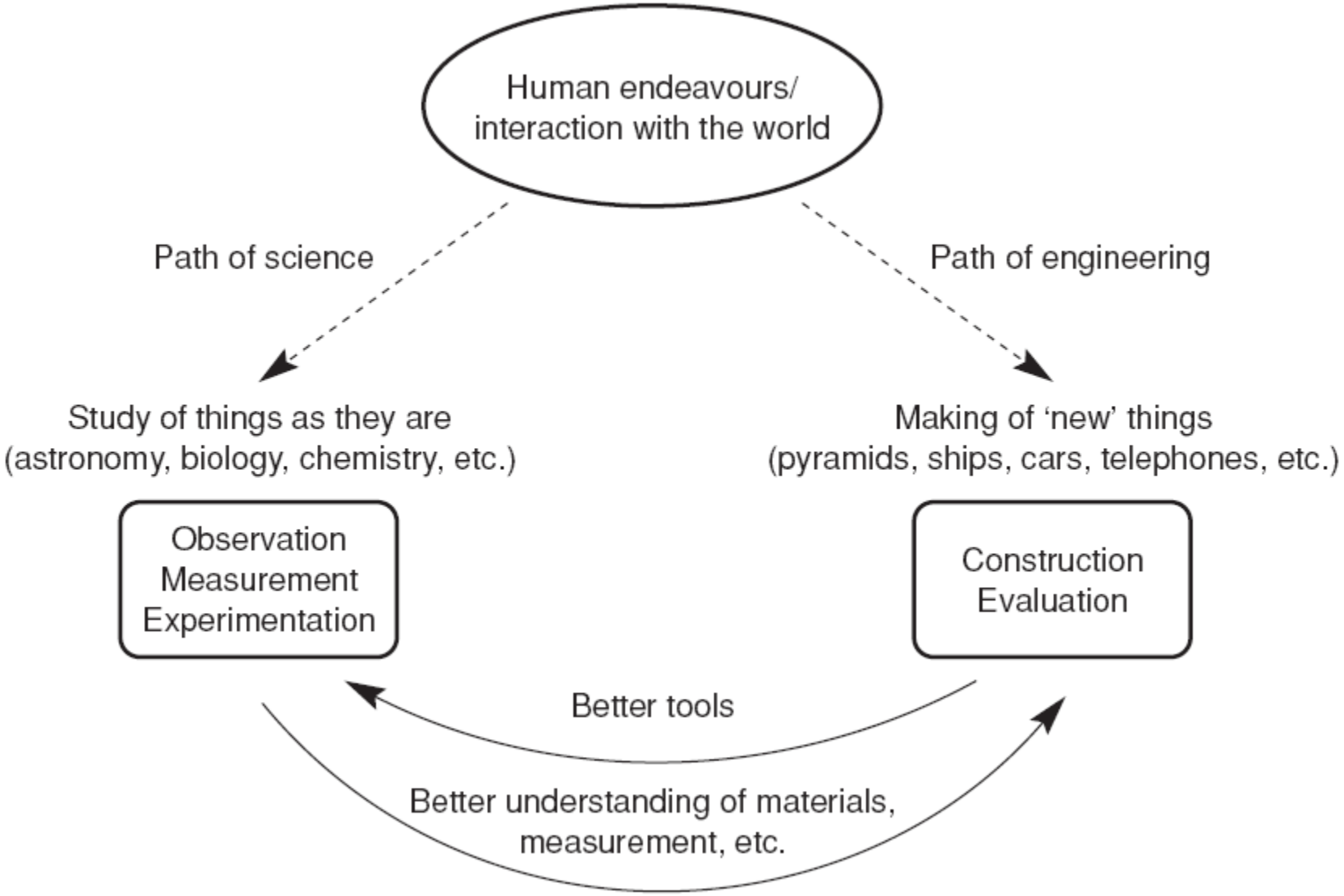
- Formación de profesionales
- Actualización continua de profesionales
- Interdisciplinariedad
- Colaboración con usuarios y especialistas no informáticos
- Asuntos éticos, sociales y profesionales
- Ordenar y codificar conocimiento tecnológico y profesional
- Gestión de personal y de relaciones
- Inteligencia artificial
- Análisis de datos masivos
- Sistemas ciberfísicos
- Sistemas distribuidos de escala mundial
- Seguridad (*security* y *safety*) *versus* Privacidad
- Confiabilidad, disponibilidad, escalabilidad

Ethos: Frederick P. Brooks

- Computer Scientist as Toolsmith II (CACM 39(3), 1996, pp. 61..68):

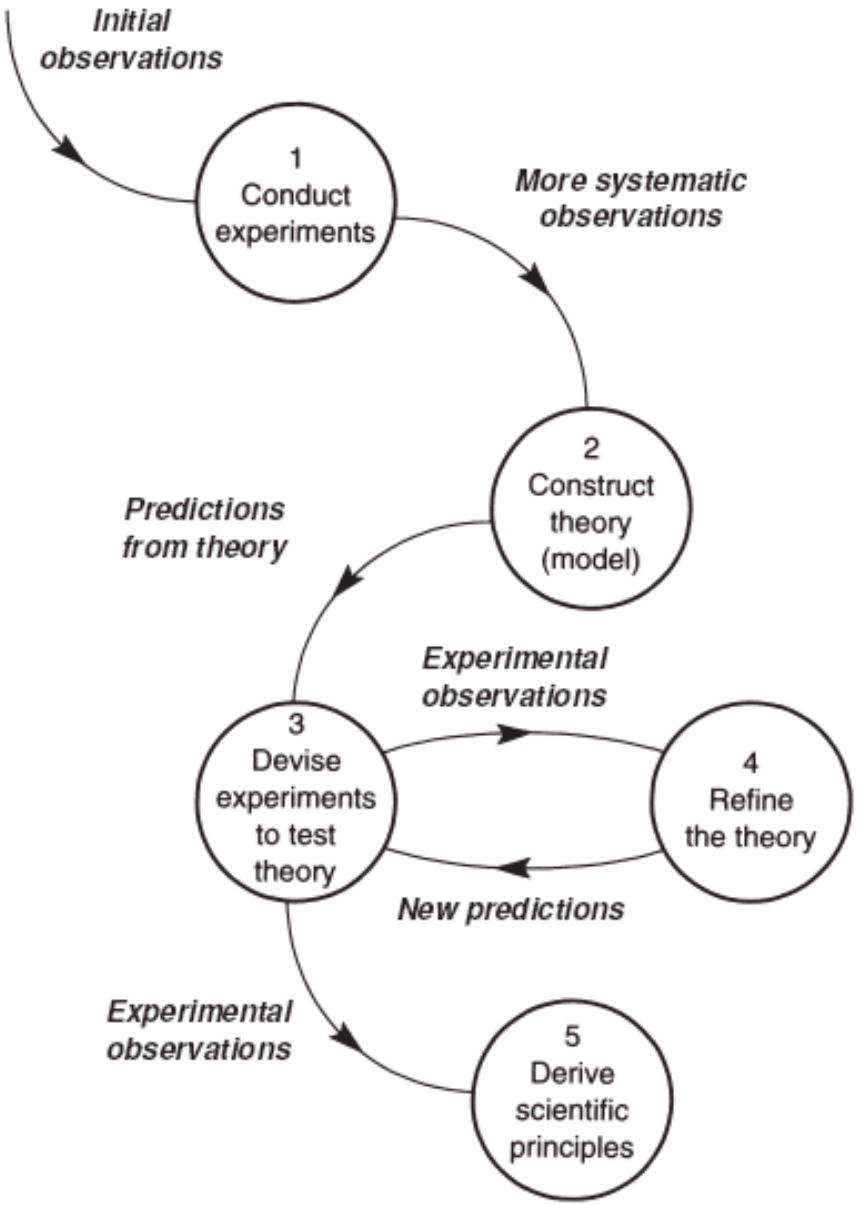
> **The scientist** *builds in order to study;*
> **the engineer** *studies in order to build.*

Investigación



Budgen

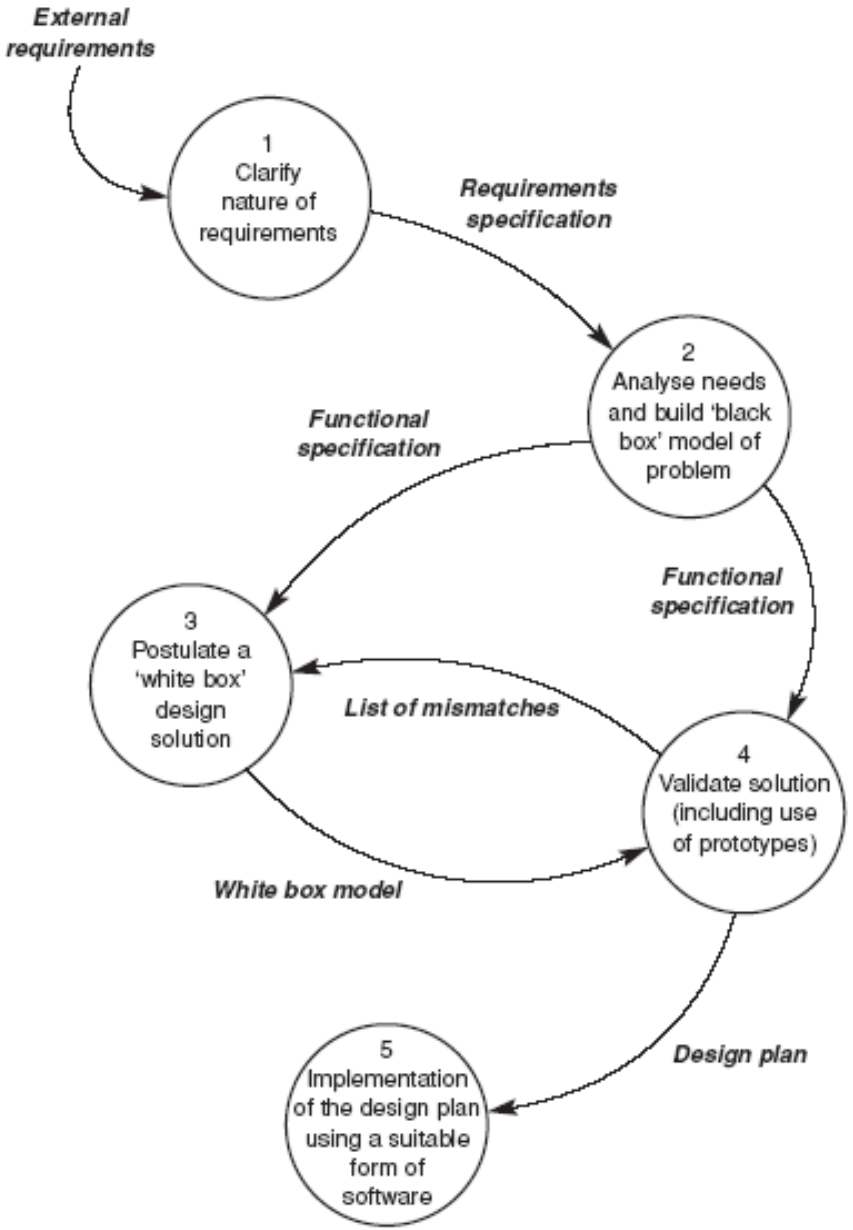
Investigación – estilo científico



Budgen

The nature of scientific analysis.

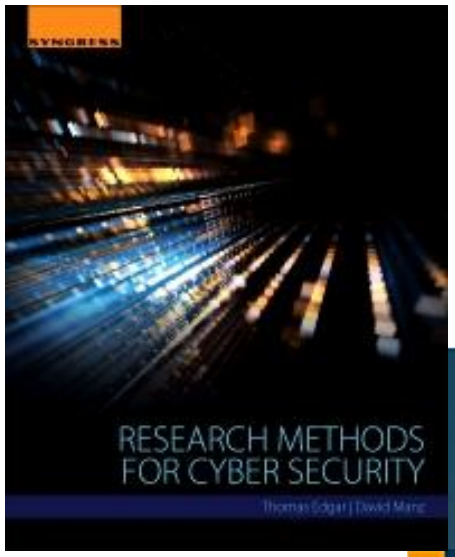
Investigación – estilo ingenieril

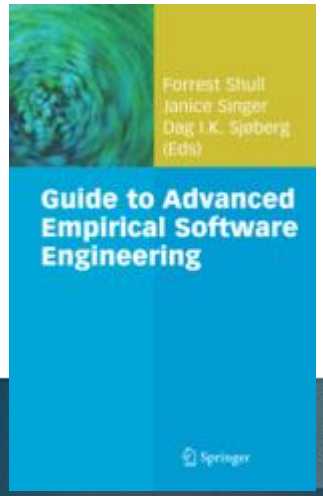


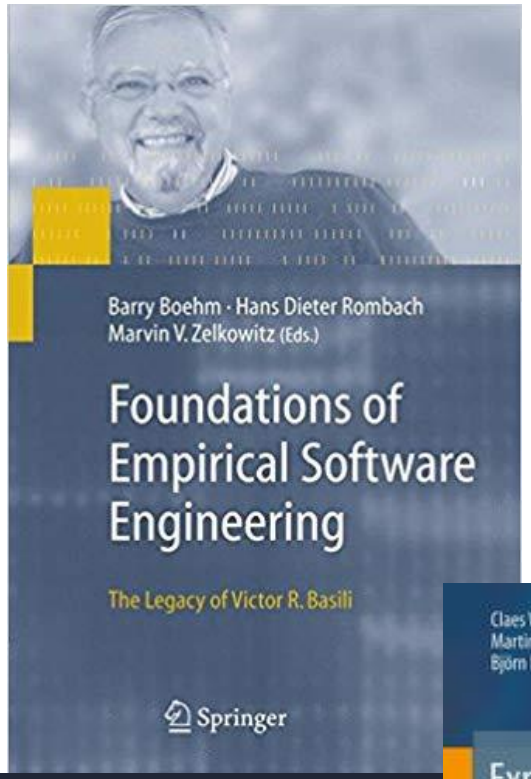
A model of the design process.

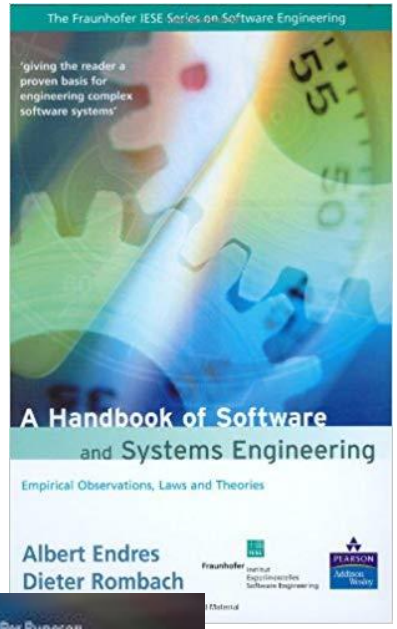
Budgen

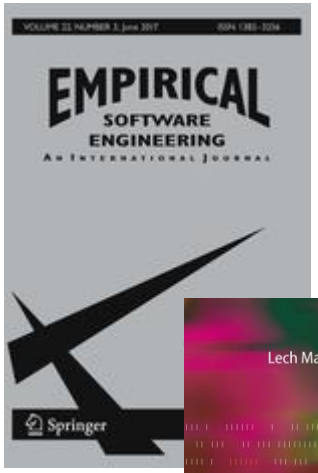
Investigación

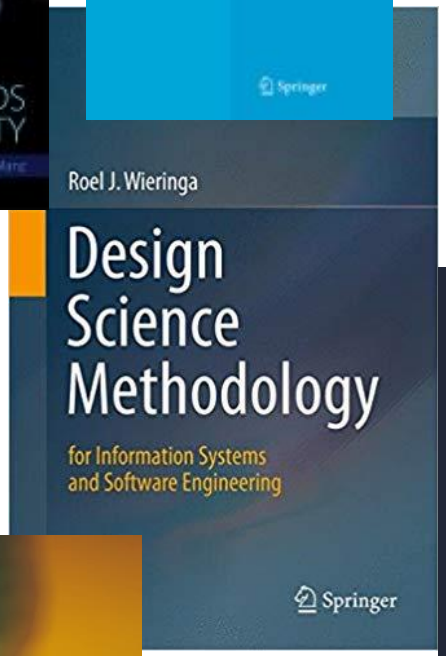


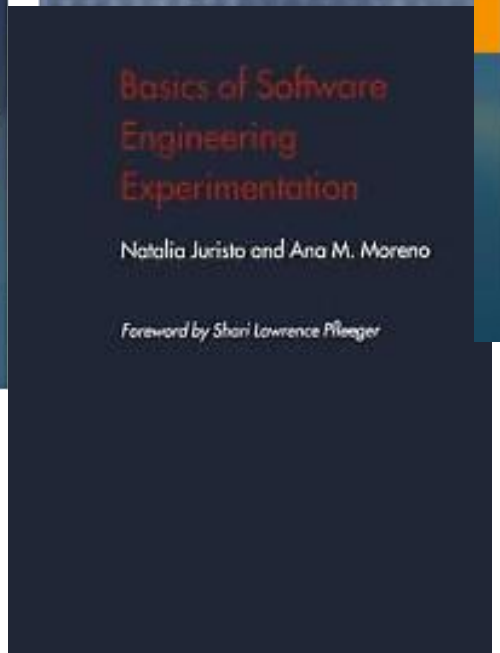


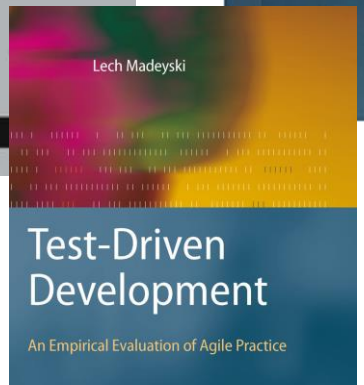


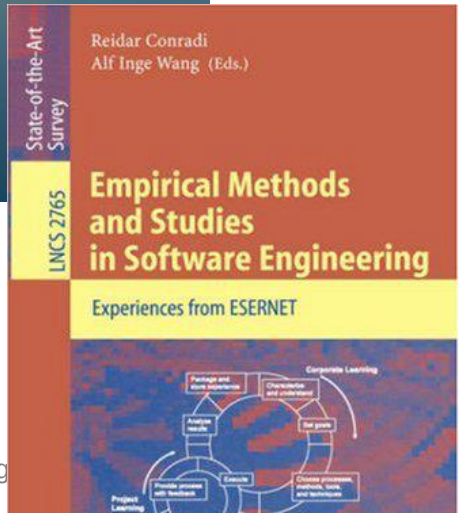




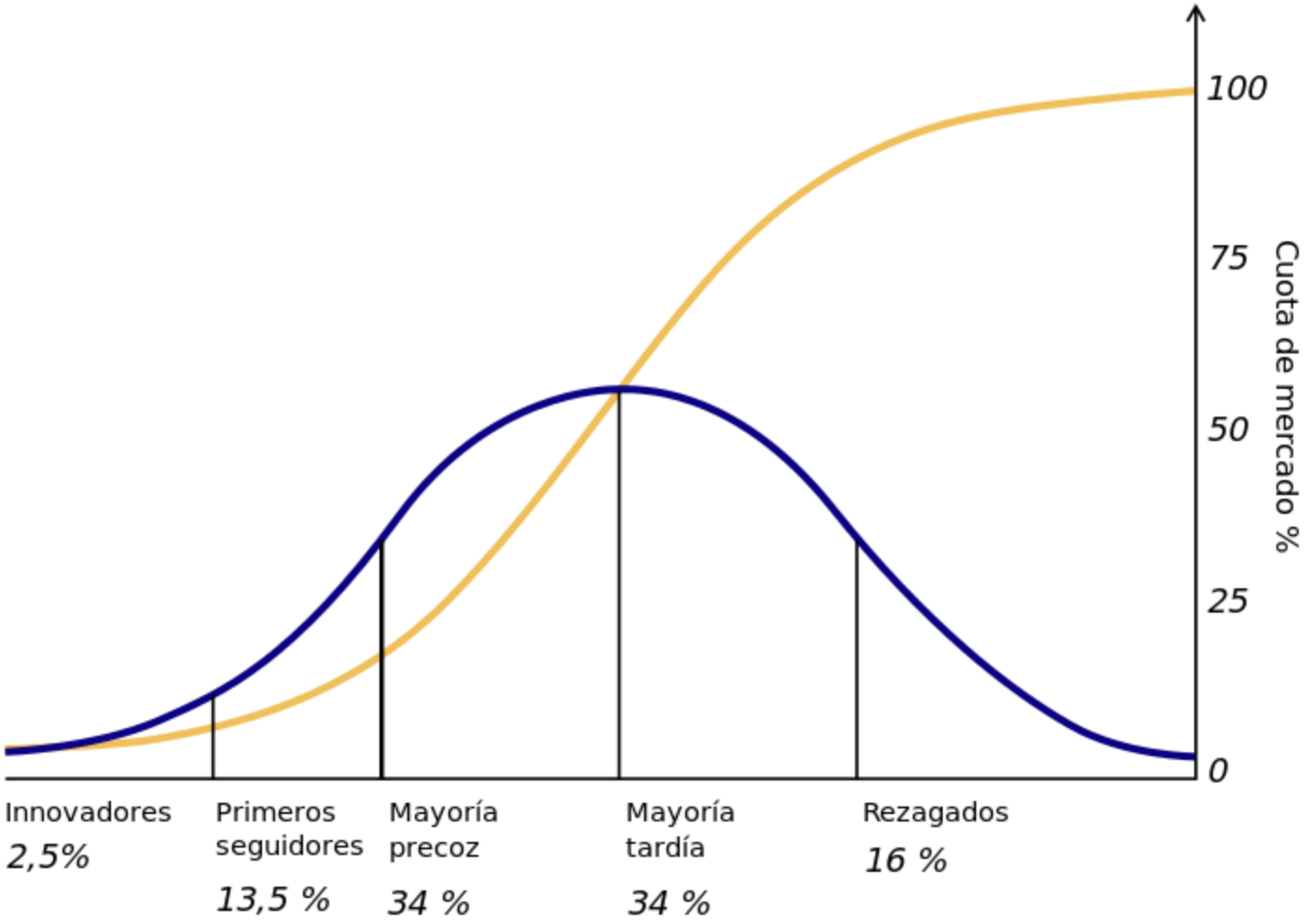




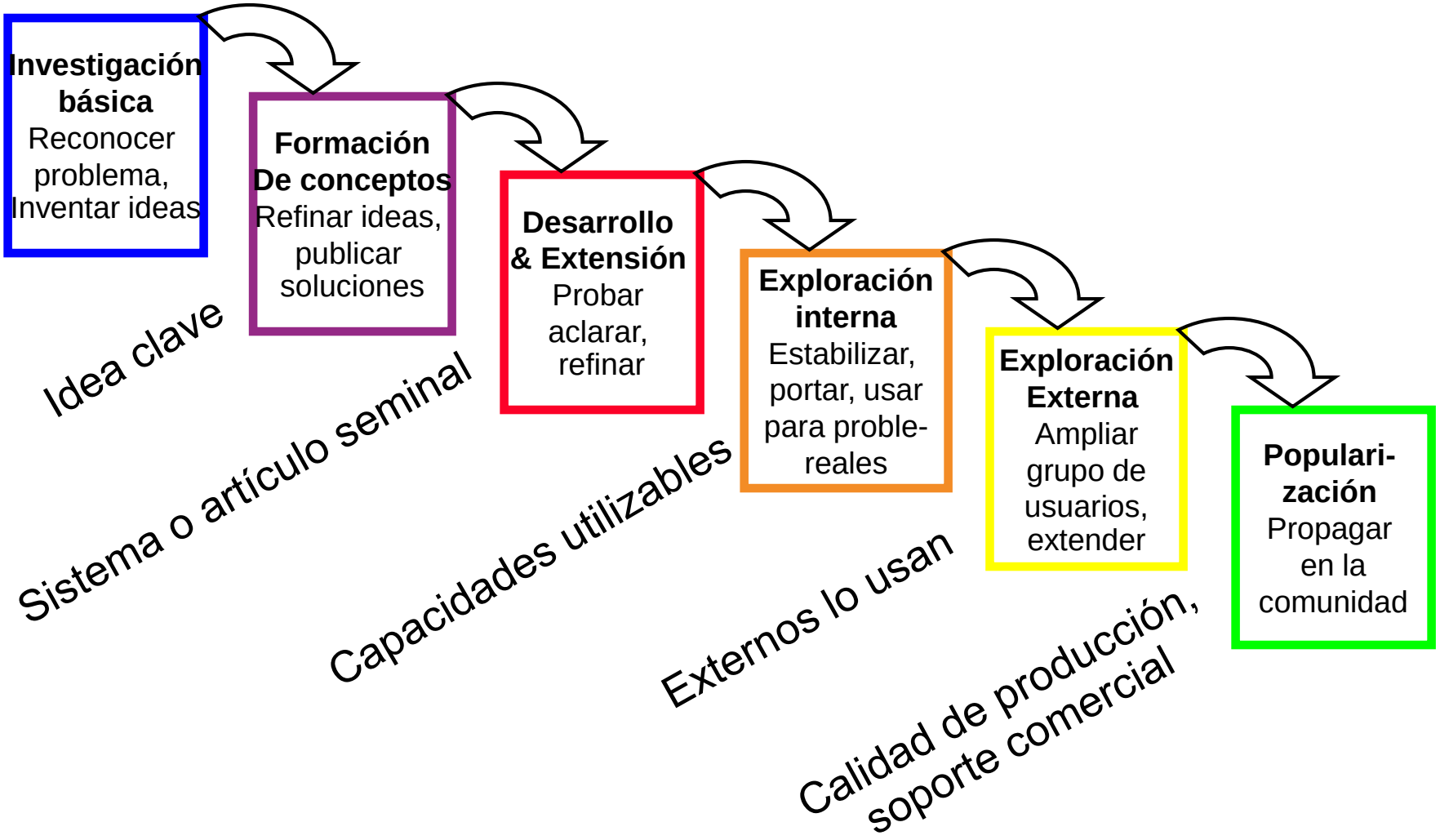




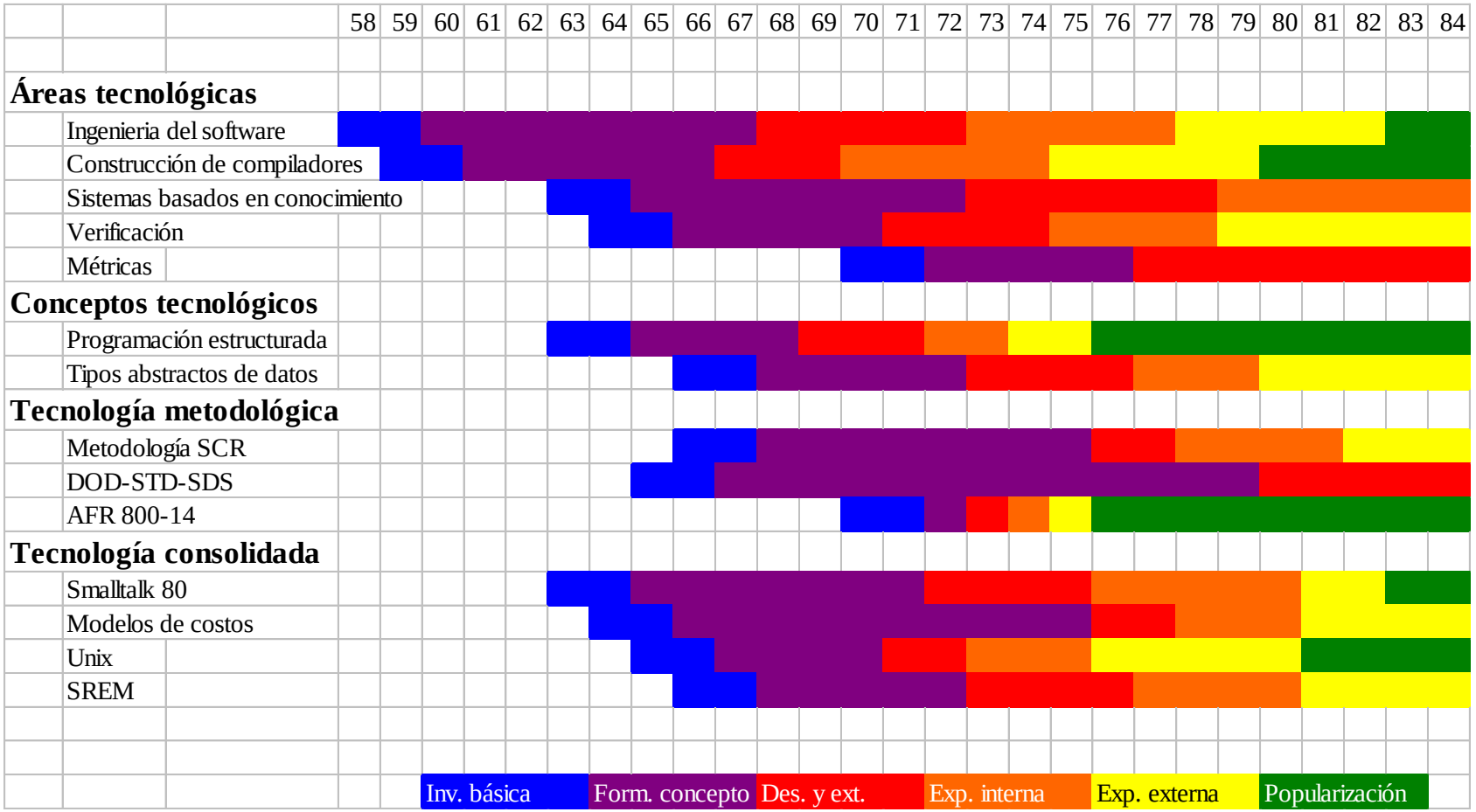
Difusión de innovaciones



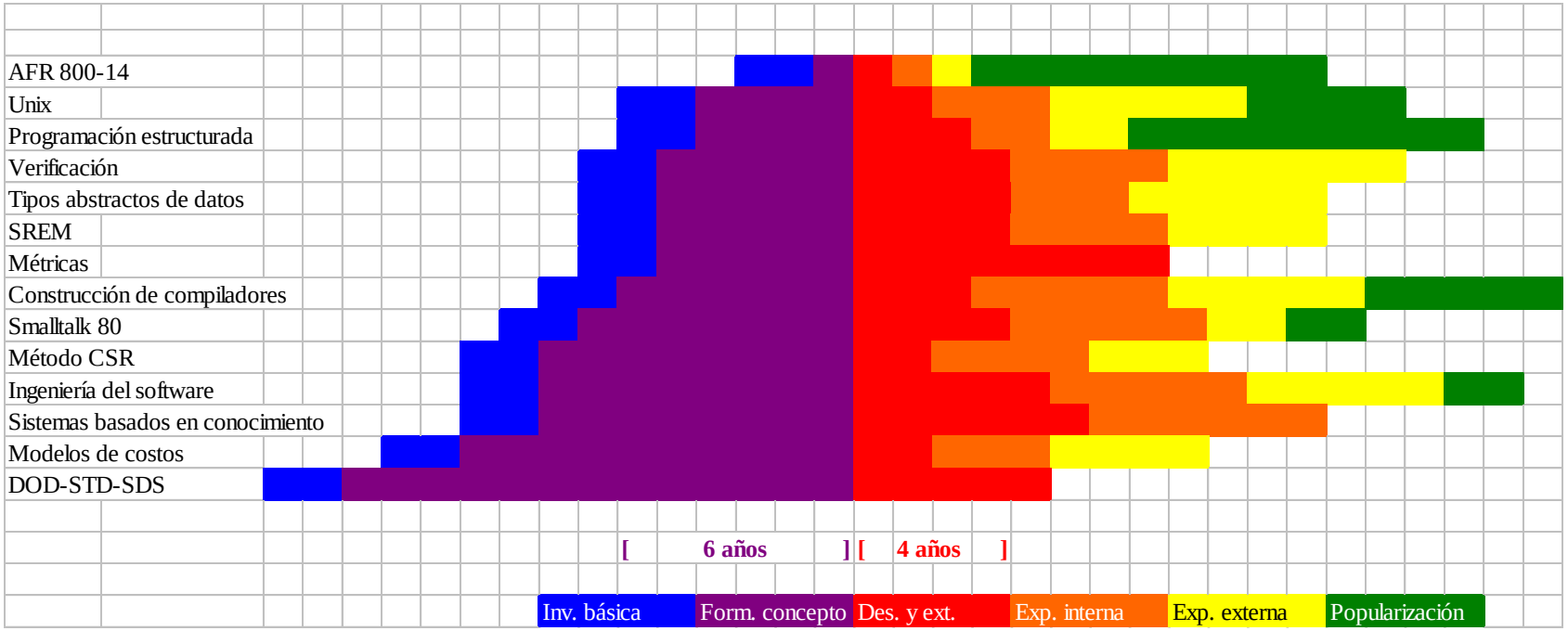
Modelo de Redwine y Riddle



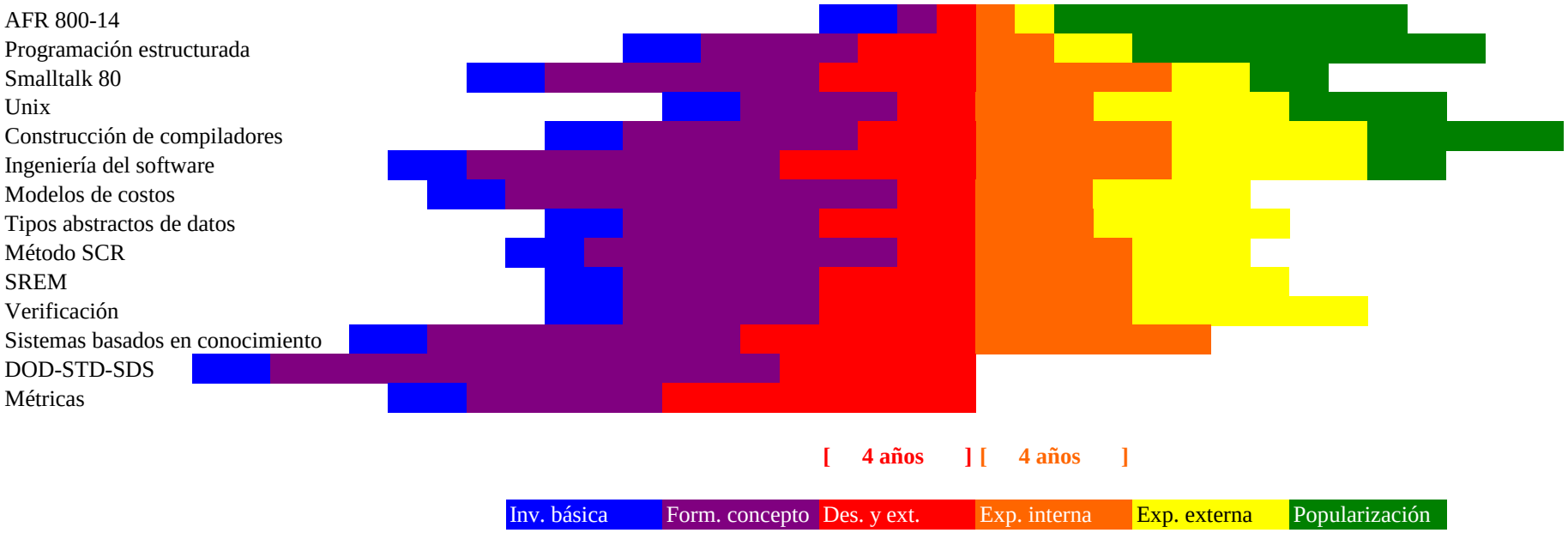
Puntos de maduración



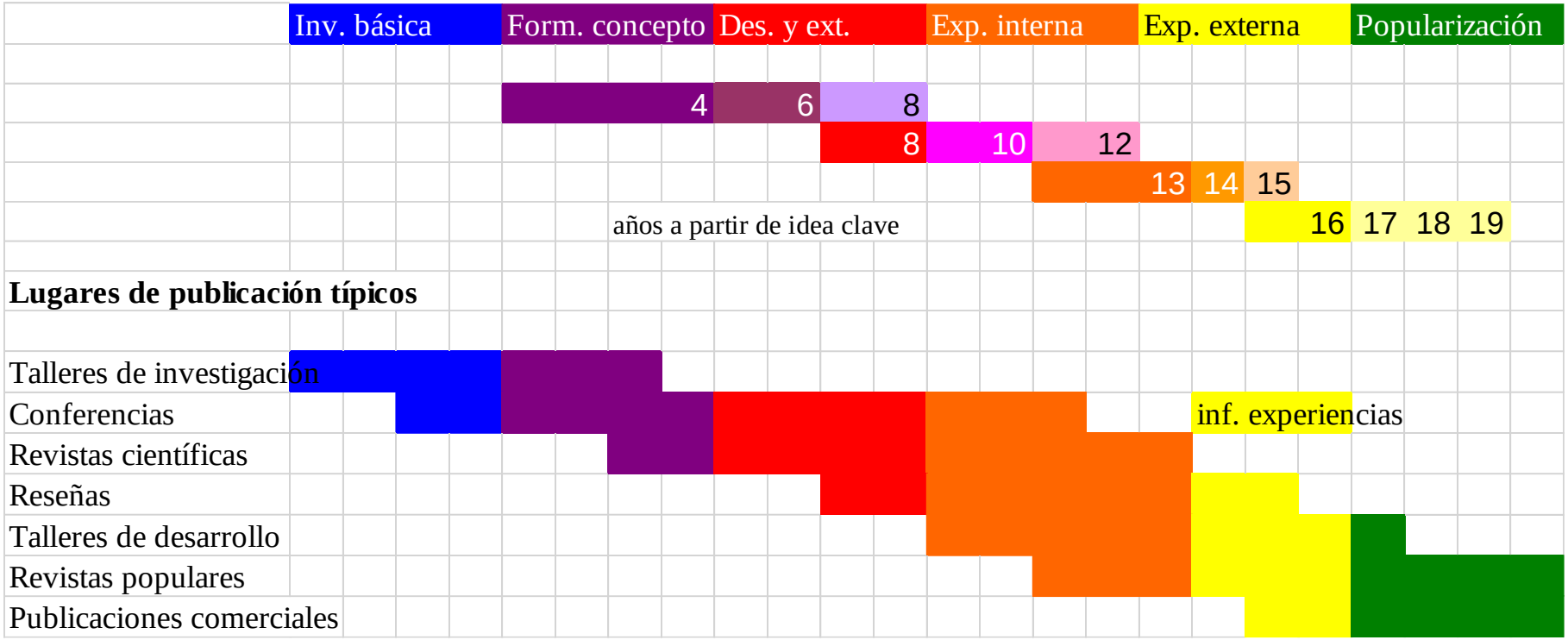
Aparición de solución clara



Capacidades utilizables

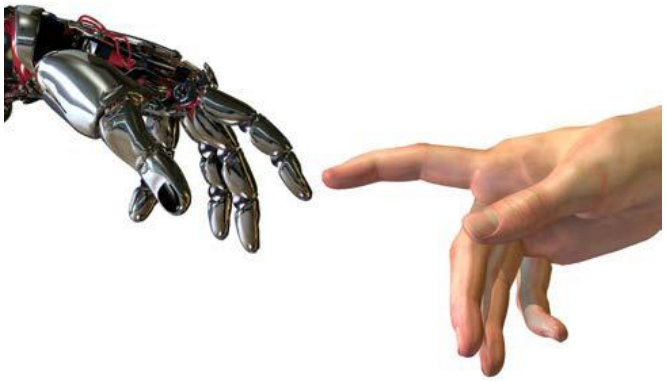


Tiempos de fases y publicaciones



Envoi (Booch)

Software is the invisible writing that whispers the stories of possibility to our hardware…



*…and you are the **storytellers**.*

Grady Booch