26 ottobre 2019

LDMI 2019

La «Unix Way» e i metodi per produrre software

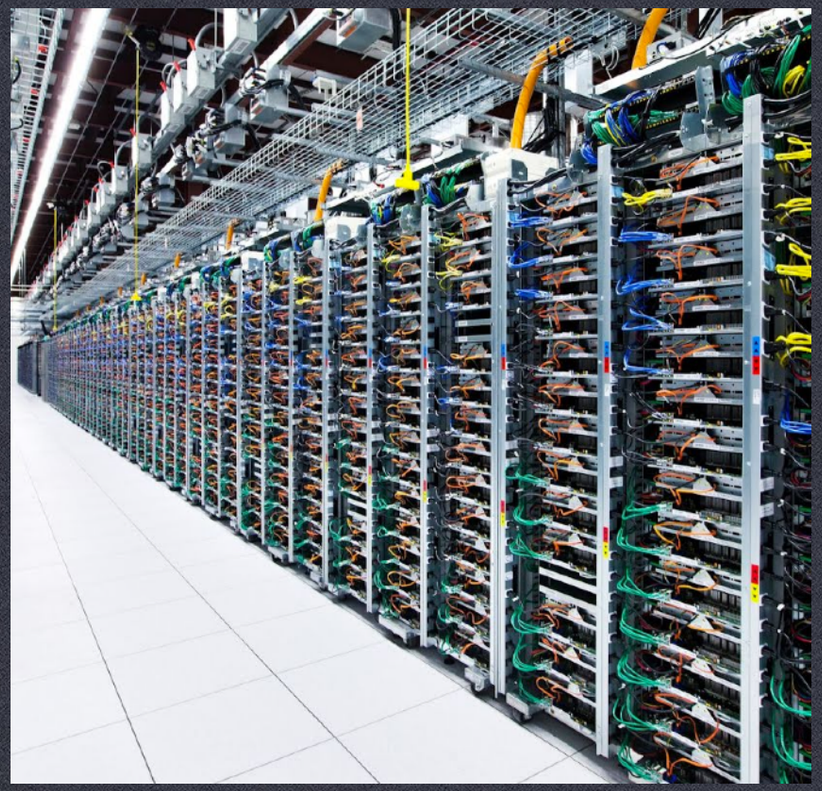
Fabio Mora

@fabiomora — fabiomora.com

Il contenuto è disponibile in licenza CC-BY-SA 4.0, salvo diversamente specificato. — Citazioni e [ref] a fine presentazione.

```
25 # persistent / runtime deps
26 RUN set -eux; \
27     apt-get update; \
28     apt-get install -y --no-install-recommends \
29         $PHPIZE_DEPS \
30         ca-certificates \
31         curl \
32         xz-utils \
33     ; \
34     rm -rf /var/lib/apt/lists/*
35
36 ENV PHP_INI_DIR /usr/local/etc/php
37 RUN set -eux; \
38     mkdir -p "$PHP_INI_DIR/conf.d"; \
39 # allow running as an arbitrary user (https://github.com/docker-library/php/issu
40     [ ! -d /var/www/html ]; \
41     mkdir -p /var/www/html; \
42     chown www-data:www-data /var/www/html; \
43     chmod 777 /var/www/html
44
45 ##<autogenerated>##
46 ##</autogenerated>##
47
48 # Apply stack smash protection to functions using local buffers and alloca()
49 # Make PHP's main executable position-independent (improves ASLR security mechan
50 # Enable optimization (-O2)
51 # Enable linker optimization (this sorts the hash buckets to improve cache local
52 # Adds GNU HASH segments to generated executables (this is used if present, and
53 # https://github.com/docker-library/php/issues/272
54 ENV PHP_CFLAGS="-fstack-protector-strong -fpic -fpie -O2"
55 ENV PHP_CPPFLAGS="$PHP_CFLAGS"
56 ENV PHP_LDFLAGS="-Wl,-O1 -Wl,--hash-style=both -pie"
NORMAL  +0 ~0 -0   Dockerfile-debian.template
"Dockerfile-debian.template" 225L, 7489C
```

il cloud!



2019

UN FRAMMENTO DI PROGRAMMA E LE INFRASTRUTTURE DI OGGI

Diagram for the computation by the Engine of the Numbers of Bernoulli.

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data. ${}^1V_1$ ○ 0 0 1 [1]	${}^1V_2$ ○ 0 0 2 [2]	${}^1V_3$ ○ 0 0 4 [n]	${}^0V_4$ ○ 0 0 0 [ ]	${}^0V_5$ ○ 0 0 0 [ ]	${}^0V_6$ ○ 0 0 0 [ ]	${}^0V_7$ ○ 0 0 0 [ ]	${}^0V_8$ ○ 0 0 0 [ ]	${}^0V_9$ ○ 0 0 0 [ ]	${}^0V_{10}$ ○ 0 0 0 [ ]
1	×	${}^1V_2 \times {}^1V_3$	${}^1V_4, {}^1V_5, {}^1V_6$	${}^1V_2 = {}^1V_2$, ${}^1V_3 = {}^1V_3$	$= 2n$	...	2	n	$2n$	$2n$	$2n$				
2	−	${}^1V_4 - {}^1V_1$	${}^2V_4$	${}^1V_4 = {}^2V_4$, ${}^1V_1 = {}^1V_1$	$= 2n - 1$	1	...	...	$2n-1$						
3	+	${}^1V_5 + {}^1V_1$	${}^2V_5$	${}^1V_5 = {}^2V_5$, ${}^1V_1 = {}^1V_1$	$= 2n + 1$	1	...	...	...	$2n+1$					
4	÷	${}^2V_5 \div {}^2V_4$	${}^1V_{11}$	${}^2V_5 = {}^0V_5$, ${}^2V_4 = {}^0V_4$	$= \frac{2n-1}{2n+1}$	...	...	...	0	0	...			...	...
5	÷	${}^1V_{11} \div {}^1V_2$	${}^2V_{11}$	${}^1V_{11} = {}^2V_{11}$, ${}^1V_2 = {}^1V_2$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$	...	2	...	...	...	...			...	...
6	−	${}^0V_{13} - {}^2V_{11}$	${}^1V_{13}$	${}^2V_{11} = {}^0V_{11}$, ${}^0V_{13} = {}^1V_{13}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$	...	...	...	...	...			...	...	...
7	−	${}^1V_3 - {}^1V_1$	${}^1V_{10}$	${}^1V_3 = {}^1V_3$, ${}^1V_1 = {}^1V_1$	$= n - 1 \ (= 3)$	1	...	n	...			...	...	...	$n-1$
8	+	${}^1V_2 + {}^0V_7$	${}^1V_7$	${}^1V_2 = {}^1V_2$, ${}^0V_7 = {}^1V_7$	$= 2 + 0 = 2$	...	2	...			...	2			
9	÷	${}^1V_6 \div {}^1V_7$	${}^3V_{11}$	${}^1V_6 = {}^1V_6$, ${}^0V_{11} = {}^3V_{11}$	$= \frac{2n}{2} = A_1$	...	...	...			$2n$	2	...	...	...
10	×	${}^1V_{21} \times {}^3V_{11}$	${}^1V_{12}$	${}^1V_{21} = {}^1V_{21}$, ${}^3V_{11} = {}^3V_{11}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...	...	...	...	...	...	...	...
11	+	${}^1V_{12} + {}^1V_{13}$	${}^2V_{13}$	${}^1V_{12} = {}^0V_{12}$, ${}^1V_{13} = {}^2V_{13}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$	...	...	...	...	...	...	...	...	...	...
12	−	${}^1V_{10} - {}^1V_1$	${}^2V_{10}$	${}^1V_{10} = {}^2V_{10}$, ${}^1V_1 = {}^1V_1$	$= n - 2 \ (= 2)$	1	...	...	...	...	...	...	...	...	$n-2$
13	−	${}^1V_6 - {}^1V_1$	${}^2V_6$	${}^1V_6 = {}^2V_6$, ${}^1V_1 = {}^1V_1$	$= 2n - 1$	1	...	...	...	...	$2n-1$				
14	+	${}^1V_1 + {}^1V_7$	${}^2V_7$	${}^1V_1 = {}^1V_1$, ${}^1V_7 = {}^2V_7$	$= 2 + 1 = 3$	1	...	...	...	...	...	3			
15	÷	${}^2V_6 \div {}^2V_7$	${}^1V_8$	${}^2V_6 = {}^2V_6$, ${}^2V_7 = {}^2V_7$	$= \frac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\frac{2n-1}{3}$		
16	×	${}^1V_8 \times {}^3V_{11}$	${}^4V_{11}$	${}^1V_8 = {}^0V_8$, ${}^3V_{11} = {}^4V_{11}$	$= \frac{2n}{2} \cdot \frac{2n}{}$	...	...	...	...	...	...	...	0	...	...
17	−	${}^2V_6 - {}^1V_1$	${}^3V_6$	${}^2V_6 = {}^3V_6$, ${}^1V_1 = {}^1V_1$	$= 2$	1	...	...	...	...	$2n-2$				
18	+	${}^1V_1 + {}^2V_7$	${}^3V_7$	${}^2V_7 = {}^3V_7$, ${}^1V_1 = {}^1V_1$		1	...	...	...	...	...	4			
19	÷	${}^3V_6 \div {}^3V_7$	${}^1V_9$	${}^3V_6 = {}^3V$, ${}^3V_7 = {}^3$		...	...	...	...	...	$2n-2$	4	...	$\frac{2n-2}{4}$	...
20	×	${}^1V_9 \times {}^4V_{11}$	${}^5V_{11}$	${}^1V_9$, ${}^4V$	$2 \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$	...	...	...	...	...	...	...	...	0	
21	×	${}^1V_{22} \times {}^5V_{11}$	${}^0V_{12}$		$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = B_3 A_3$	...	...	...	...	...	...	...	...	...	...
22	+	${}^2V_{12} + {}^2V_{13}$	${}^3V_{13}$	${}^2V_{13}$ ^{12} ${}^{V}_{13}$	$= A_0 + B_1 A_1 + B_3 A_3$	...	...	...	...	...	...	...	...	...	...
23	−	${}^2V$ ${}^{1}V$	${}^3V$	${}^2V_{10} = {}^3V_{10}$	$= n - 3 \ (= 1)$	1									$n-3$

l' infrastruttura

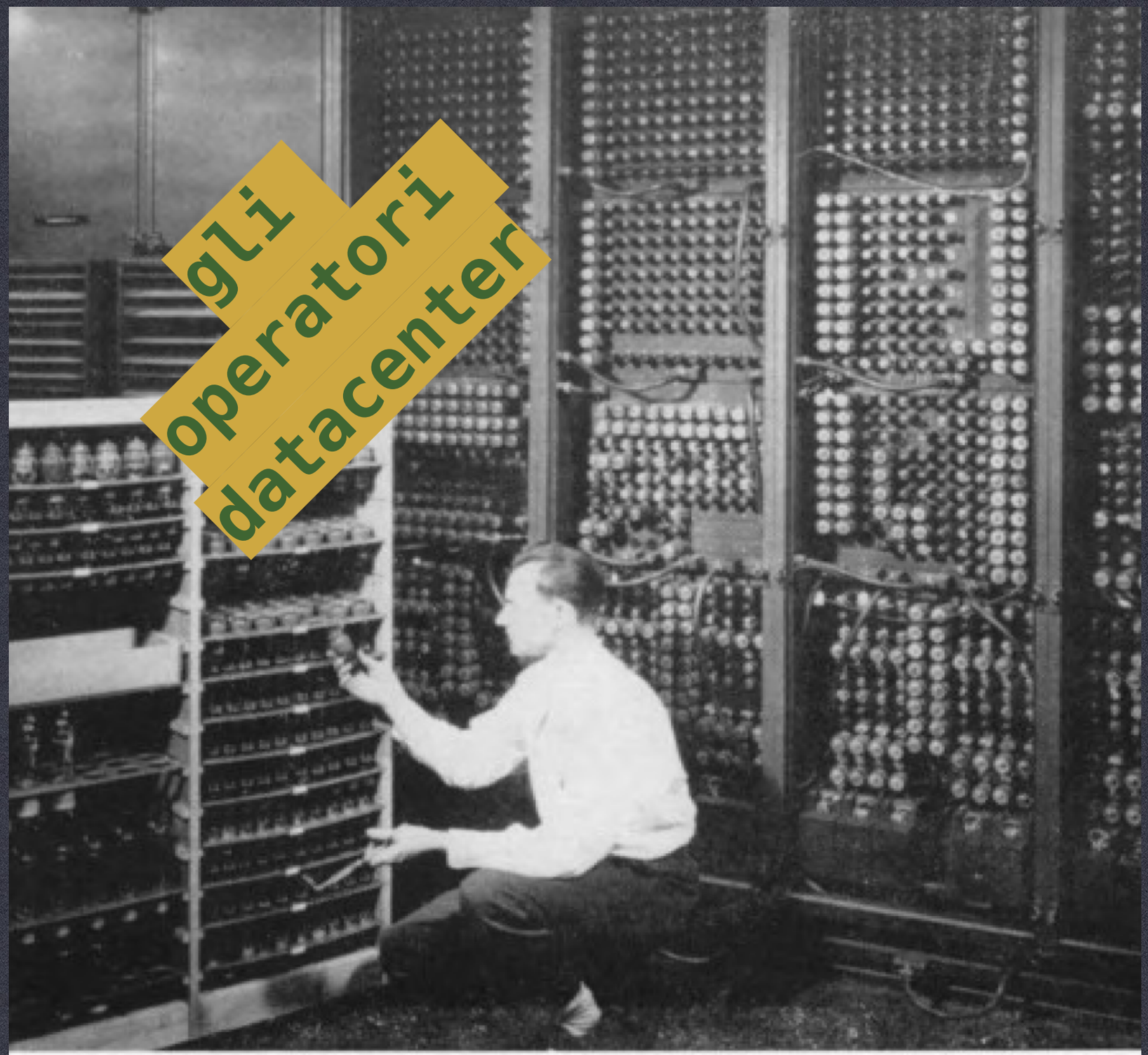
una SWE

il software

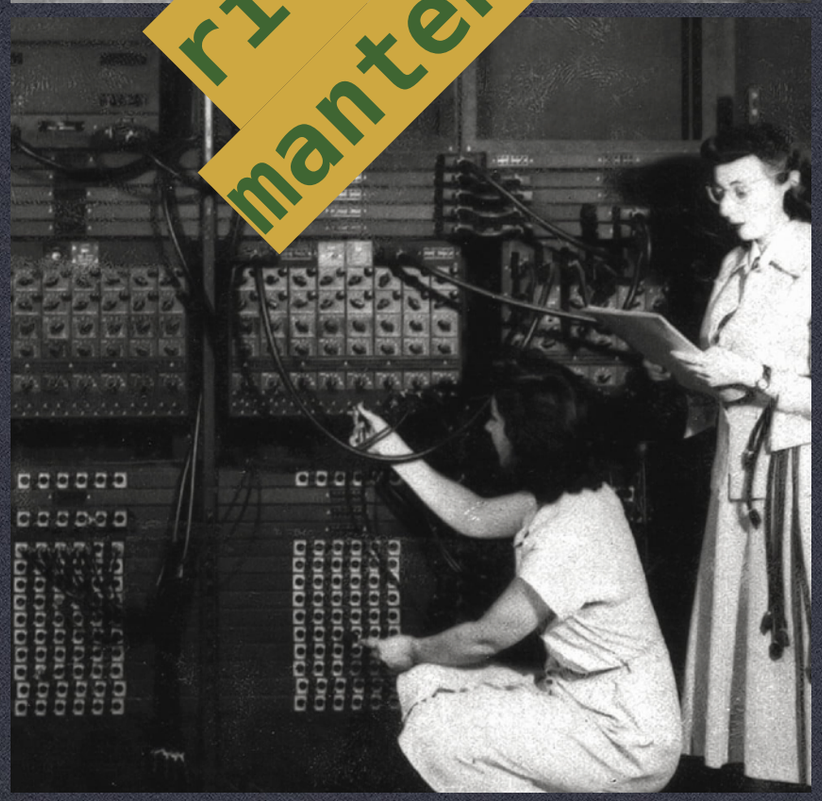
il "job"

1838

IL PRIMO PROGRAMMA DELLA STORIA E LA SUA INFRASTRUTTURA



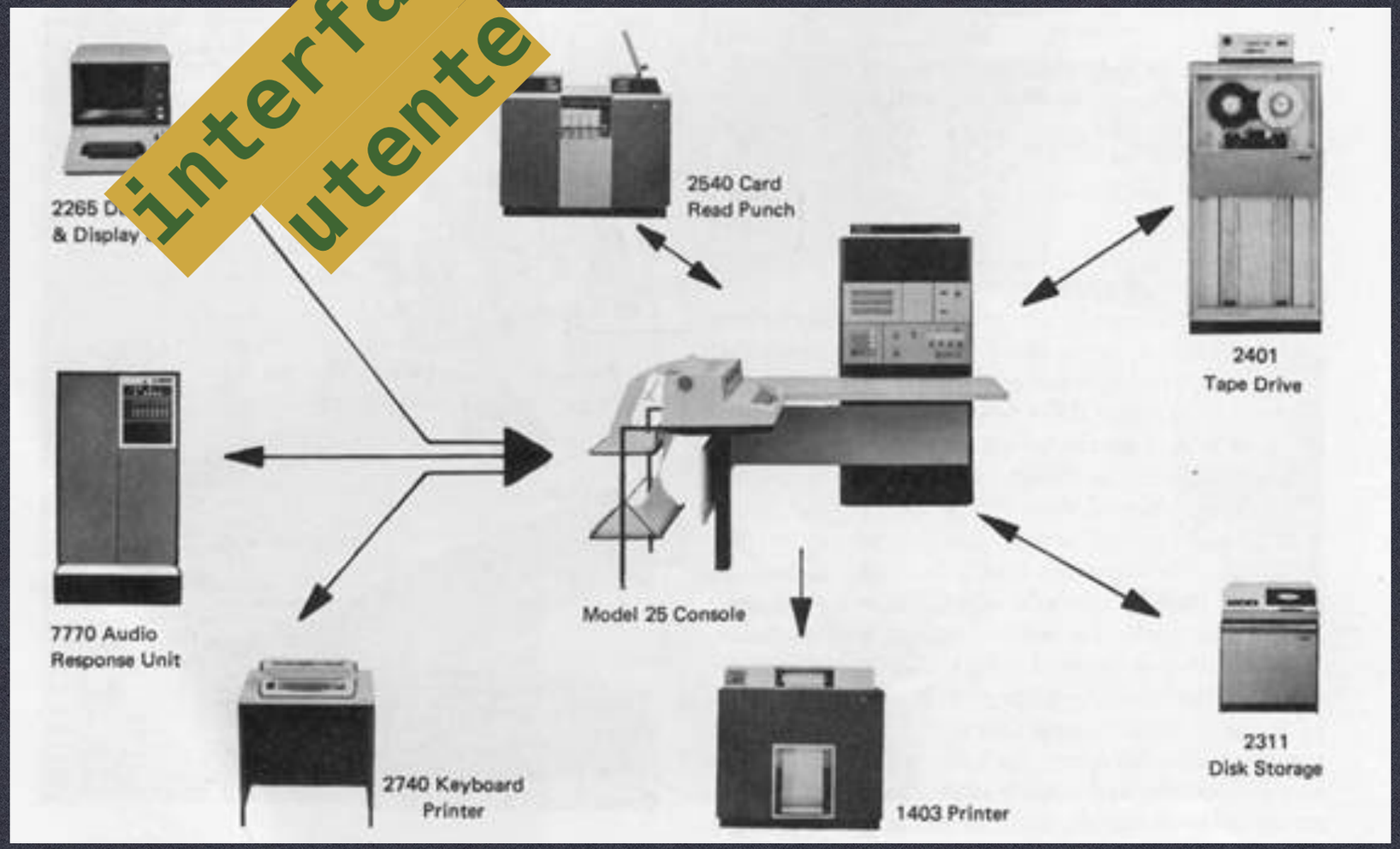
Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.



1940

CIRCA 100 ANNI DOPO: UN PUNTO DI SVOLTA

interfacce utente



il "provisioning"

1965

LA TERZA GENERAZIONE DI INFRASTRUTTURE E SISTEMI OPERATIVI

"Futuro" del computing

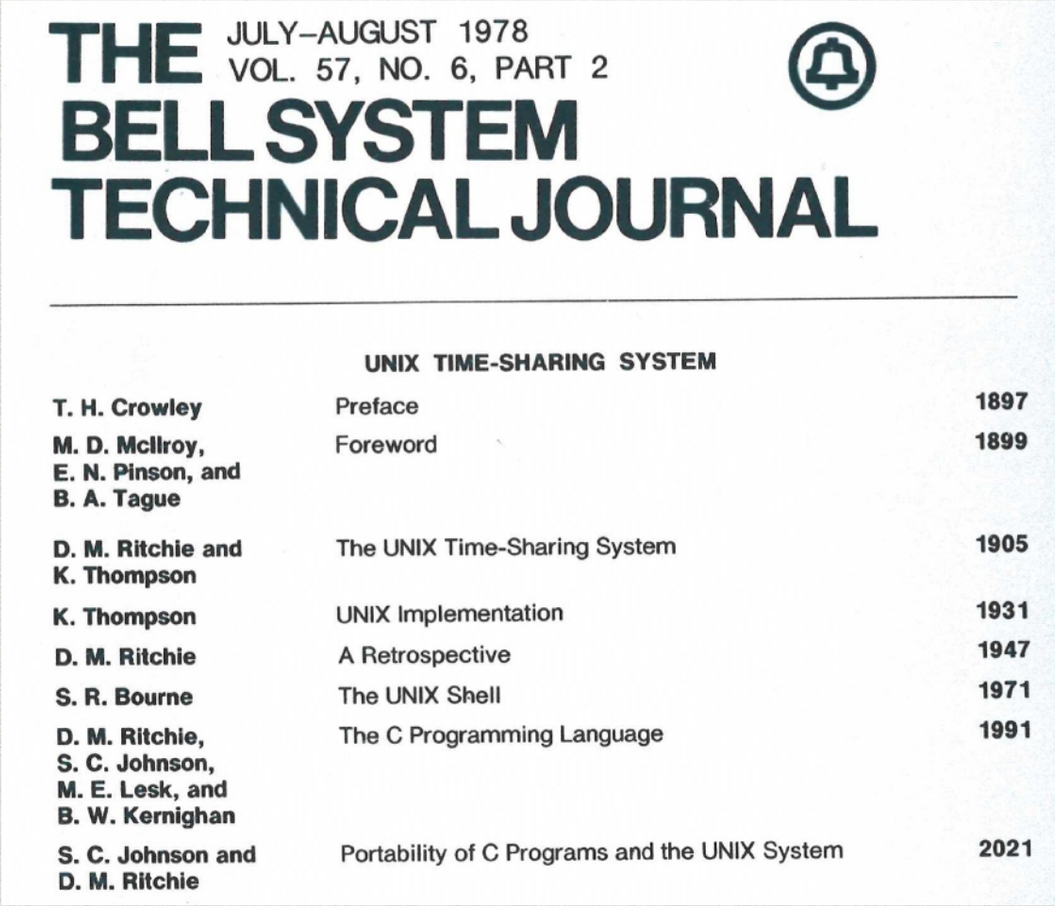
- **Memorie elettroniche** (RAM & dischi)
  - Consentono di caricare i programmi e dati [illegible] la fine delle schede perforate.
- **Multi-programmazione**
  - Un solo mainframe può poter gestire più *job* in contemporanea.
- **Time-sharing**
  - Il Sistema Operativo può mettere in pausa un job [illegible] seguire con l'esecuzione di un altro, salvando lo stato [illegible] ia.
- **Spooling**
  - Possibilità di comunicare con altri dispositi[illegible] stampanti, registratori magnetici) attraverso buffer o code.

"scheduling" automatico

sistemi operativi

Arriva Unix

- Nel 1969 **Ken Thompson** progettava un gioco in cui serviva calcolare la posizione dei pianeti
  - Desidera usare le potenzialità del *time-sharing*
- **Unix** nasce da un progetto fallito chiamato *Multics*
  - Era troppo **complesso**
  - Con **Dennis Ritchie** (C) e **Doug McIlroy** (pipe), Thompson da forma ai principi della ***Unix Way***



THE BELL SYSTEM TECHNICAL JOURNAL

JULY–AUGUST 1978
VOL. 57, NO. 6, PART 2

UNIX TIME-SHARING SYSTEM

T. H. Crowley	Preface	1897
M. D. McIlroy, E. N. Pinson, and B. A. Tague	Foreword	1899
D. M. Ritchie and K. Thompson	The UNIX Time-Sharing System	1905
K. Thompson	UNIX Implementation	1931
D. M. Ritchie	A Retrospective	1947
S. R. Bourne	The UNIX Shell	1971
D. M. Ritchie, S. C. Johnson, M. E. Lesk, and B. W. Kernighan	The C Programming Language	1991
S. C. Johnson and D. M. Ritchie	Portability of C Programs and the UNIX System	2021

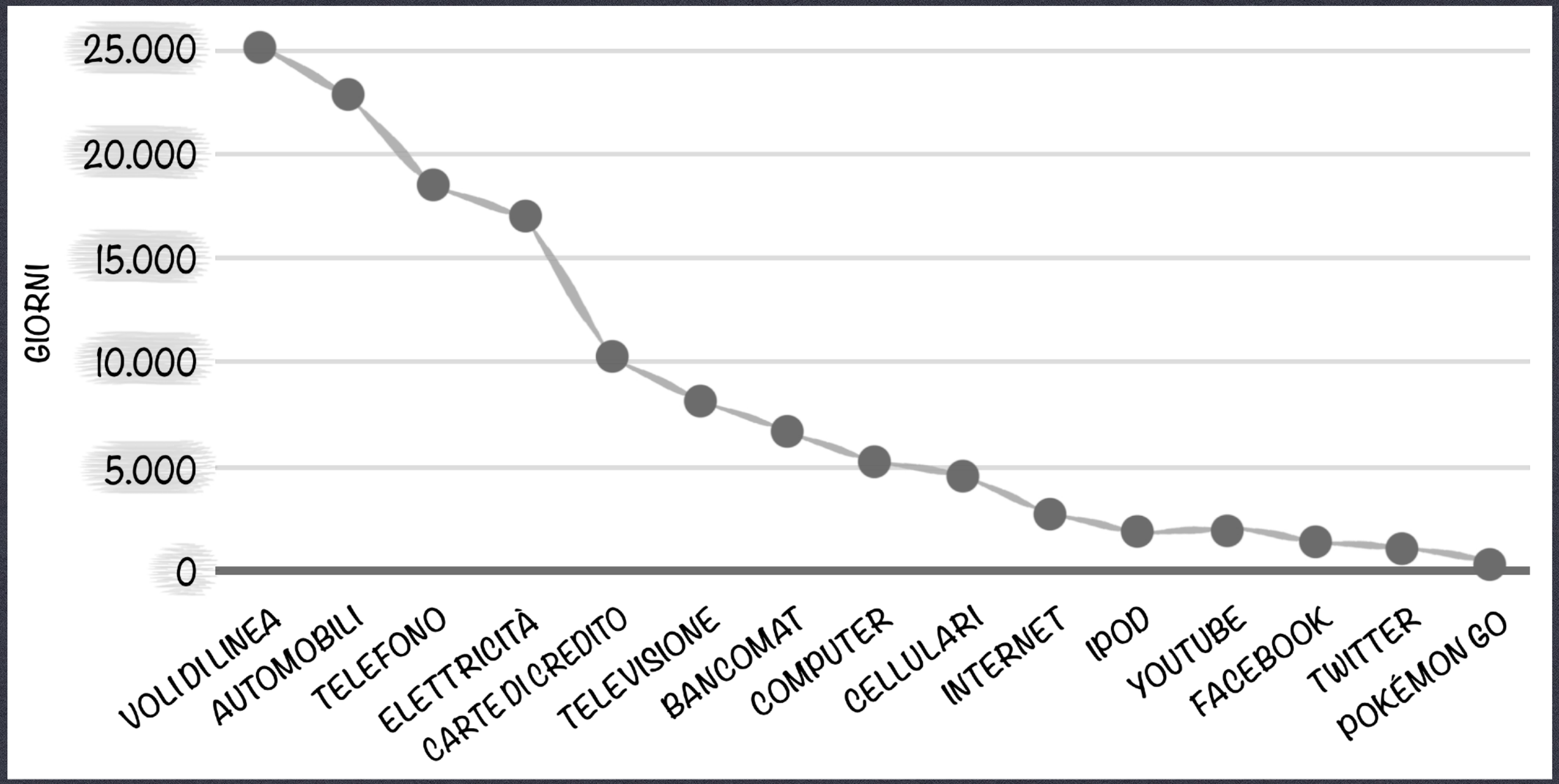
«Coloro che non capiscono Unix sono condannati a inventarlo di nuovo, con scarsi risultati»

– Henry Spencer

ANTEFATTO

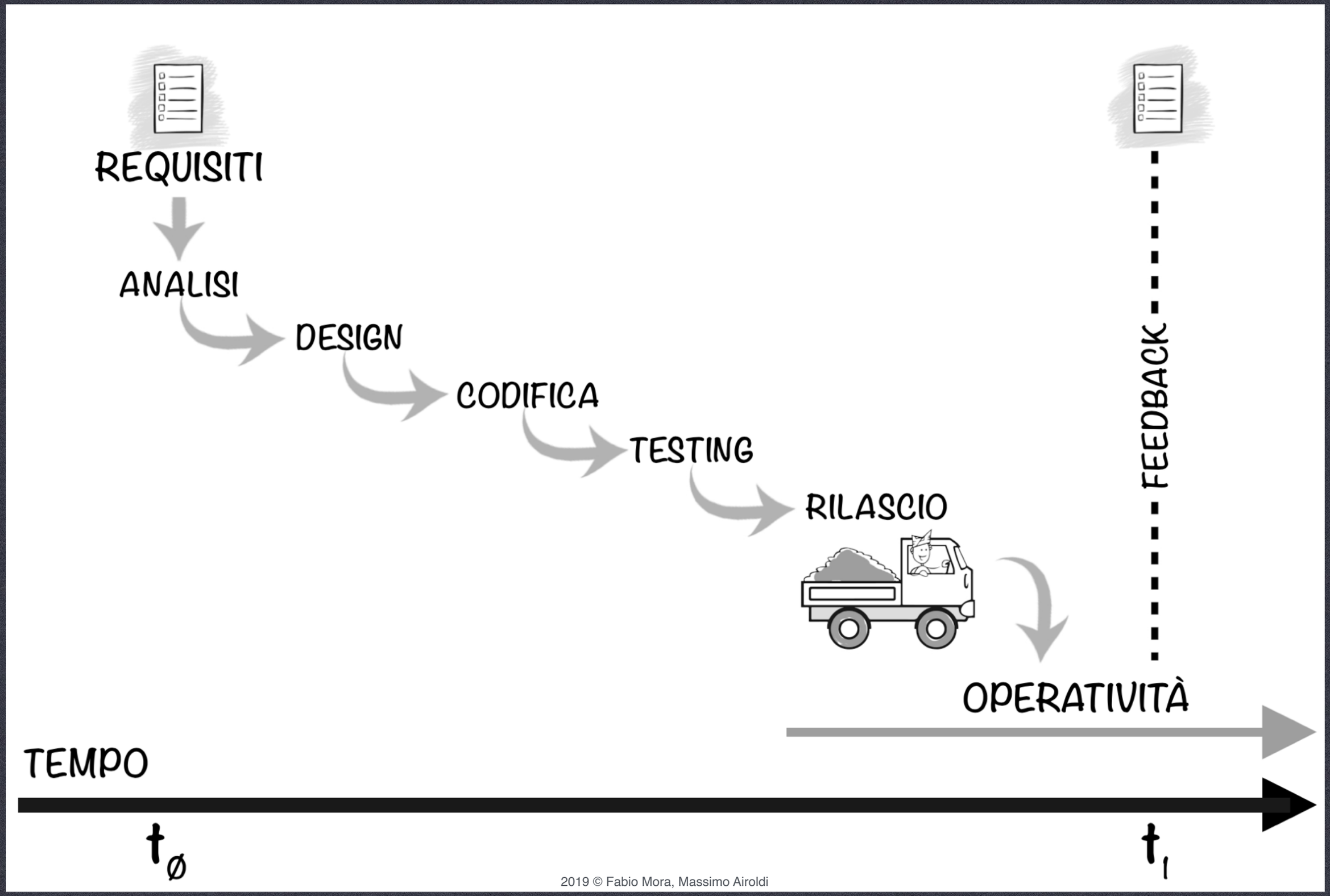
Evoluzione nei metodi per lo Sviluppo di Software

Dati: Jeff Desjardins, *How Long Does It Take to Hit 50 Million Users?*, Visual Capitalist, 2018 (https://www.visualcapitalist.com/how-long-does-it-take-to-hit-50-million-users/)
Illustrazione: 2019 © Fabio Mora, Massimo Airoldi



Giorni impiegati per raggiungere 50mln di utenti

Oltre l'ombra della guerra fredda, la corsa allo spazio e la crisi del software



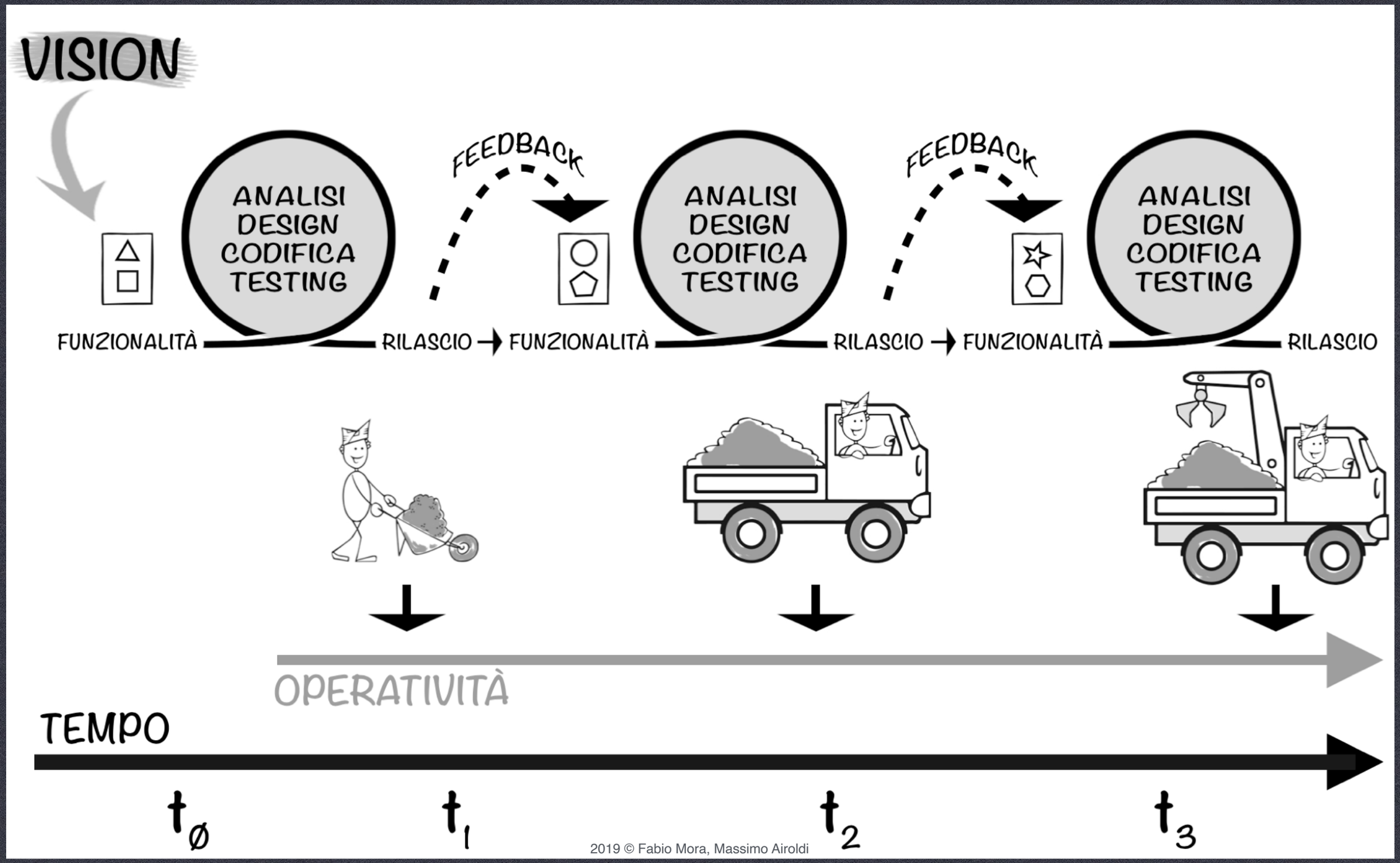
I METODI PREDITTIVI

UN METODO FUNZIONA BENE FINCHÉ…

«Il 95% della variazione di performance in un sistema è causata dal sistema stesso, solo il 5% dalle persone»

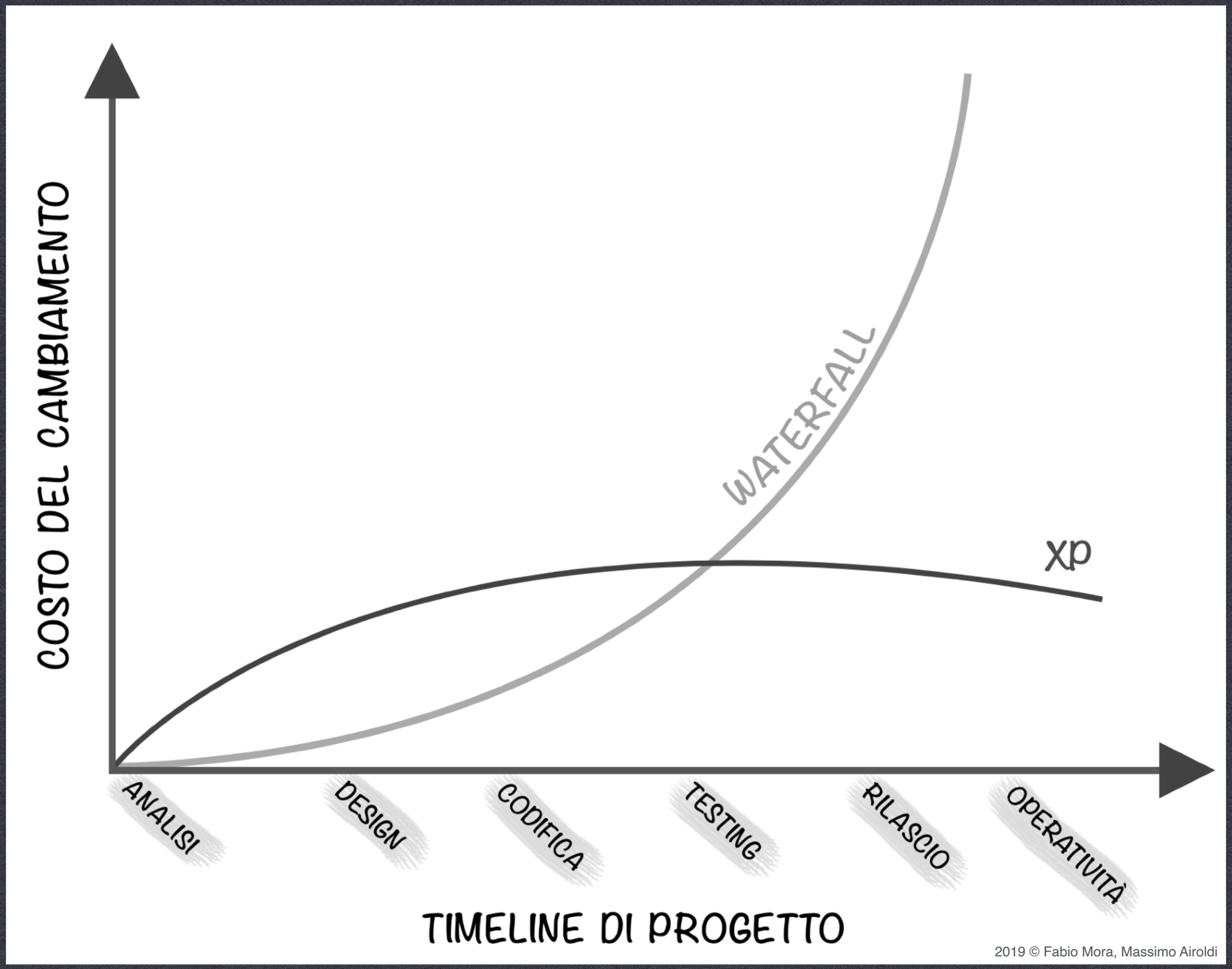
– W. E. Deming

https://www.youtube.com/watch?v=ckBfbvOXDvU



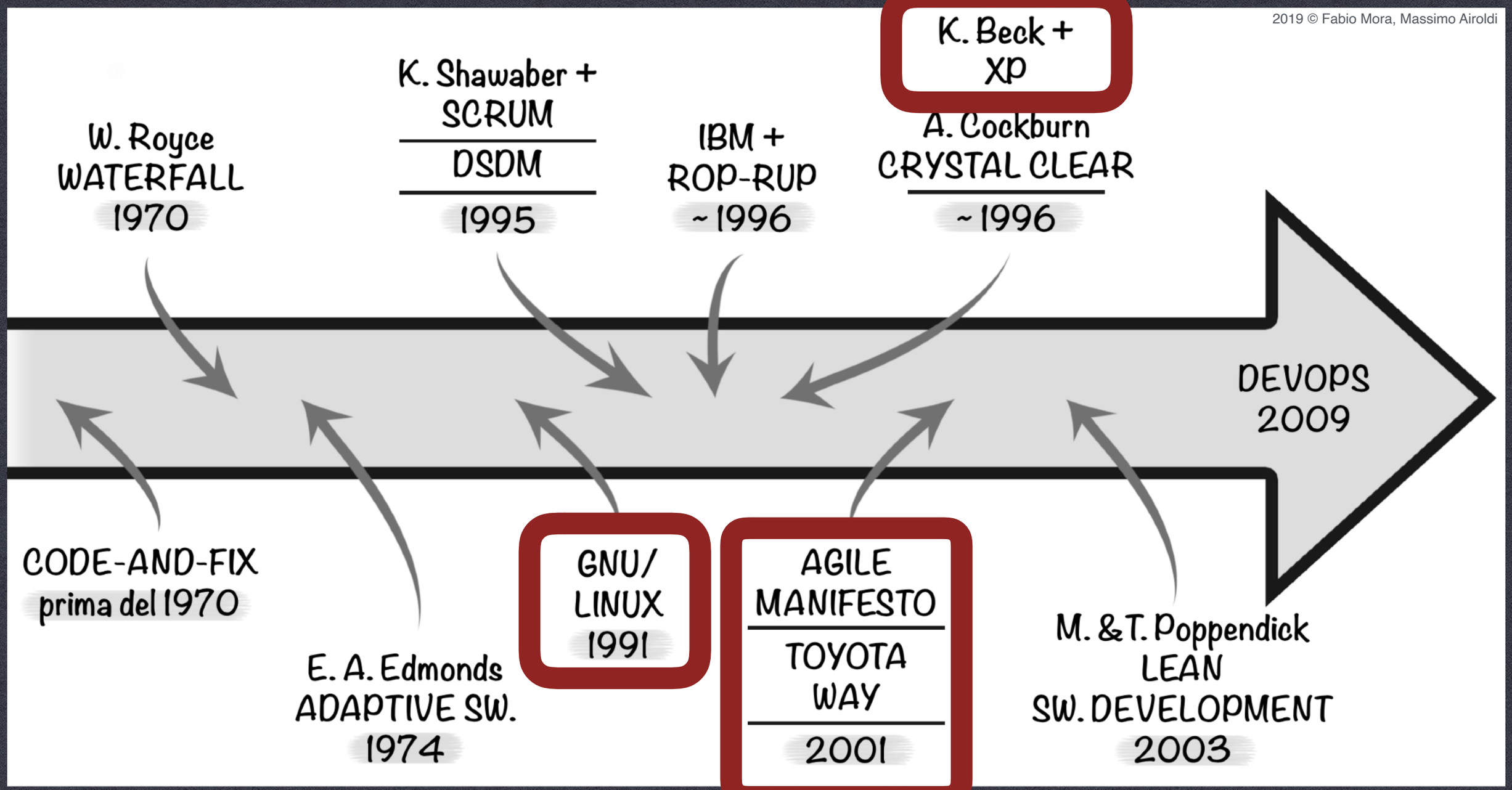
I METODI PREDITTIVI

UN METODO FUNZIONA BENE FINCHÉ…



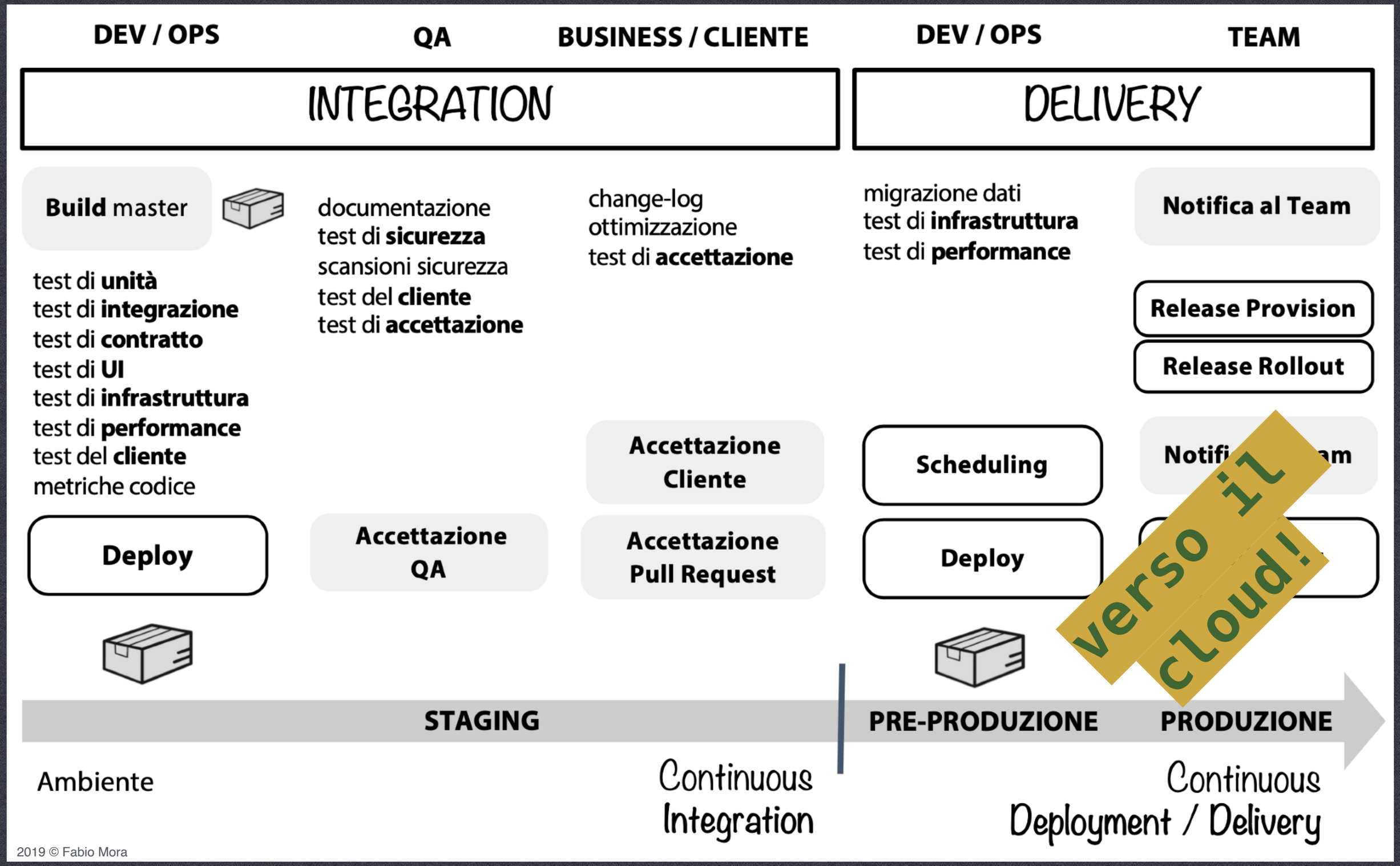
Costi contenuti + Time to Market ridotto

LA SFIDA DI DEVOPS (E XP) NEL SOFTWARE ENGINEERING PER IL MERCATO.



Alcune tappe nell'evoluzione dei metodi di Sviluppo Software

Dal metodi predittivi all'iterativo / incrementale

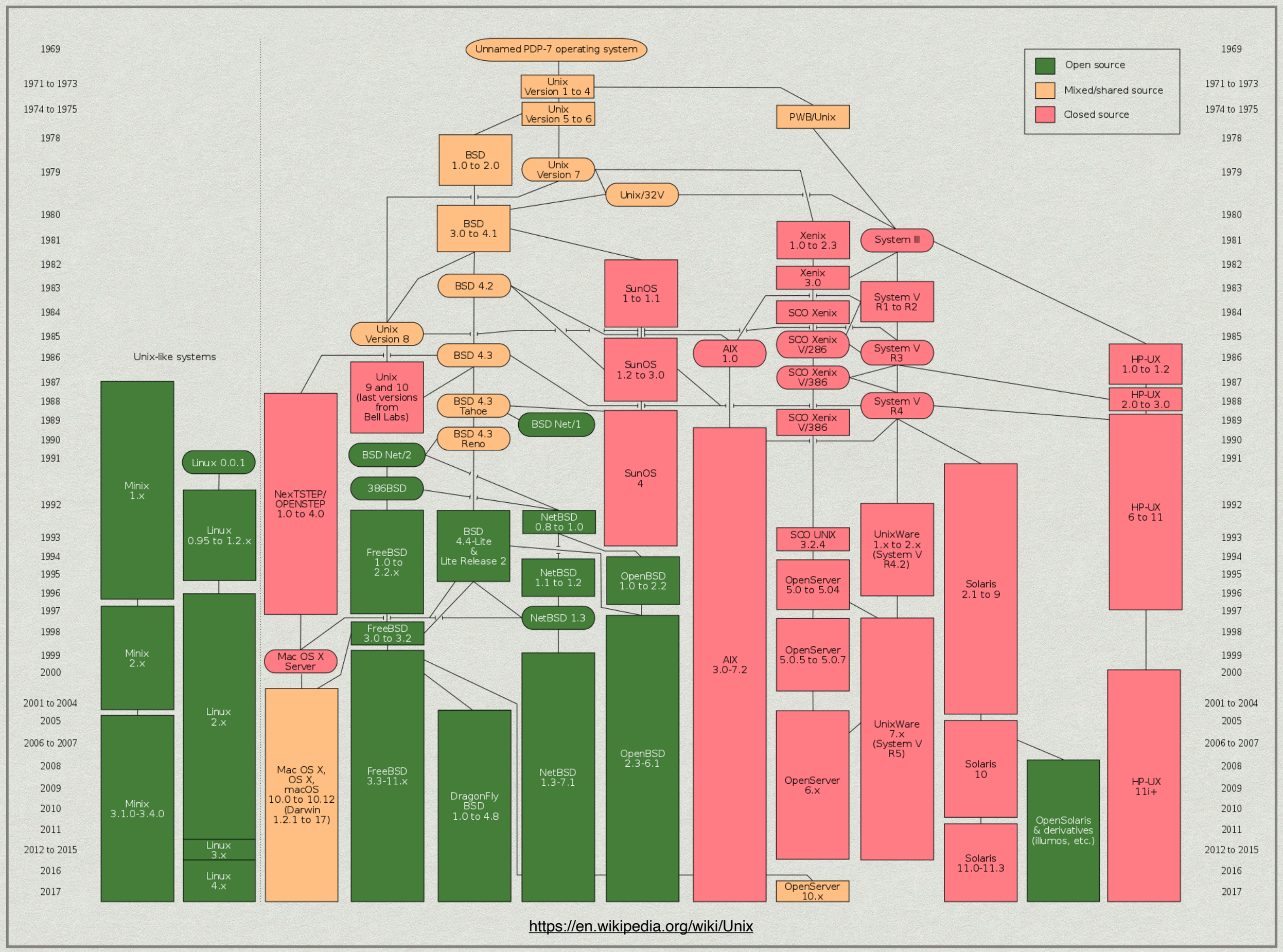


UN FLUSSO DI PRODUZIONE DEL SOFTWARE NEL 2019

LA UNIX WAY

Keep It Simple, Stupid.

– Douglas McIlroy



https://en.wikipedia.org/wiki/Unix

LA UNIX WAY

- Uno stile **modulare** di strutturazione dei sistemi.
- Si procede per mattoncini piccoli, **semplici**, efficienti e riutilizzabili.
- Ci permette di gestire la **complessità** dei problemi grazie all'uso di regole e principi **infrastrutturali**.
- La formulazione più "antica" include **4 principi**

Principio #1

- **Ogni programma deve fare una sola cosa e bene.**

  Meglio fare un nuovo programma che prenderne uno vecchio e aggiungere funzionalità.

```
fabio@Fabio-M15: ~ (zsh)
➜  ~ head -n 2 /var/log/system.log
Oct 21 20:46:41 Fabio-M15 LuLu[363]: LuLu(363): name (offset: c): www
Oct 21 20:46:41 Fabio-M15 LuLu[363]: LuLu(363): address type: ANAME [
➜  ~ tail -n 2 /var/log/system.log
Oct 21 21:57:07 Fabio-M15 LuLu[363]: LuLu(363): process start: /bin/z
Oct 21 21:57:07 Fabio-M15 LuLu[363]: LuLu(363): process start: /usr/b
```

https://github.com/coreutils/coreutils/blob/master/src/tail.c
https://github.com/coreutils/coreutils/blob/master/src/head.c

Principio #2

- **Ci aspettiamo che l'output (risultato) di tutti programmi possa diventare l'input (dati in ingresso) di altri programmi sconosciuti.**

  Meglio evitare formati tabulari o binari per rappresentare i dati, così come l'input interattivo.

```
fabio@Fabio-M15: ~ (zsh)
➜  ~ curl -s https://api.github.com/users/morafabio | jq '.name' | cat
"Fabio Mora"
```

Principio #3

- **Disegnate e costruite il software, inclusi i sistemi operativi, perché vengano provati frequentemente (settimane).**

  Non esitate a buttare via parti goffe e rifarle da capo.

Principio #4

- **Preferite gli strumenti adatti anche ai tecnici meno esperti e alle applicazioni più leggere.**

  Anche se avete bisogno di modificare la rotta per buttare via quelli vecchi.

```
fabio@Fabio-M15: ~ (zsh)                                    ⌥⌘3
➜  ~ make --version
GNU Make 3.81
Copyright (C) 2006  Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
```

COSA HO IMPARATO #1

L'Open Source, Cattedrali, Bazaar

#philosophy

#economics

GNU &...

- Un sistema operativo è utile se possiede un **kernel** e **programmi** di base
  - **Richard Stallman** nel 1977 inizia a lavorare a un clone di Unix con licenza libera, per motivi filosofici. Nasce GNU

  - **GNU/Hurd** è il futuro **kernel-space**, ispirato al PDP-10 di Digital
  - La suite GNU di programmi sarà lo **user-space**, ma è da scrivere da zero. Il lavoro verrà completato solo nel 1981

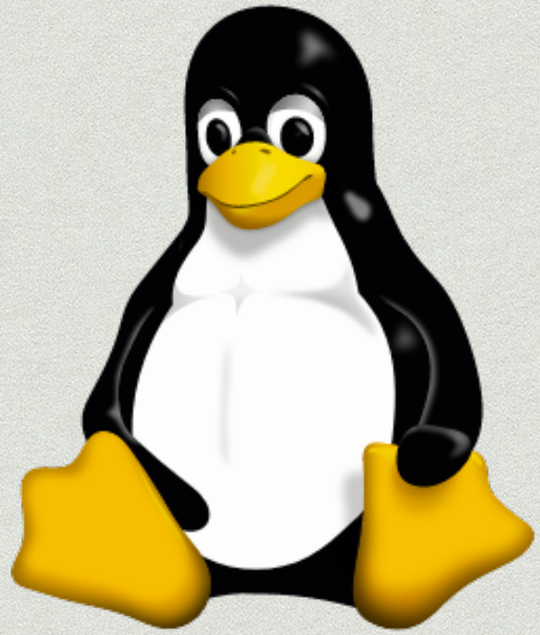
«Ciao a tutti quelli che usano Minix. Sto realizzando un **sistema operativo** free (è solo per hobby, non sarà una cosa grossa e professionale come gnu) per cloni 386 (486) AT. [...].

Mi piacerebbe avere la vostra opinione. [...] sapere quali sono le caratteristiche che interessano di più alla gente. Tutti i suggerimenti sono bene accetti :-)

– Linus Torvalds, 1991

Arriva GNU/Linux

- **Linus Torvalds** nel 1991 inizia un kernel giocattolo a partire da *Minix.*
  Linux presto sottrae il primato a Hurd
- Nel 1999 Eric Raymond pubblica *La cattedrale e il Bazaar*. Le **licenze** e la **conoscenza aperta** giocano un ruolo fondamentale.
- Un **team** non è un team se non collabora alla tastiera, e quindi al **codice sorgente**.



COSA HO IMPARATO #2

La storia di GIT, le piattaforme SaaS

#tech #method

«Quando sei in dubbio, fai il contrario di ciò che fa CVS»

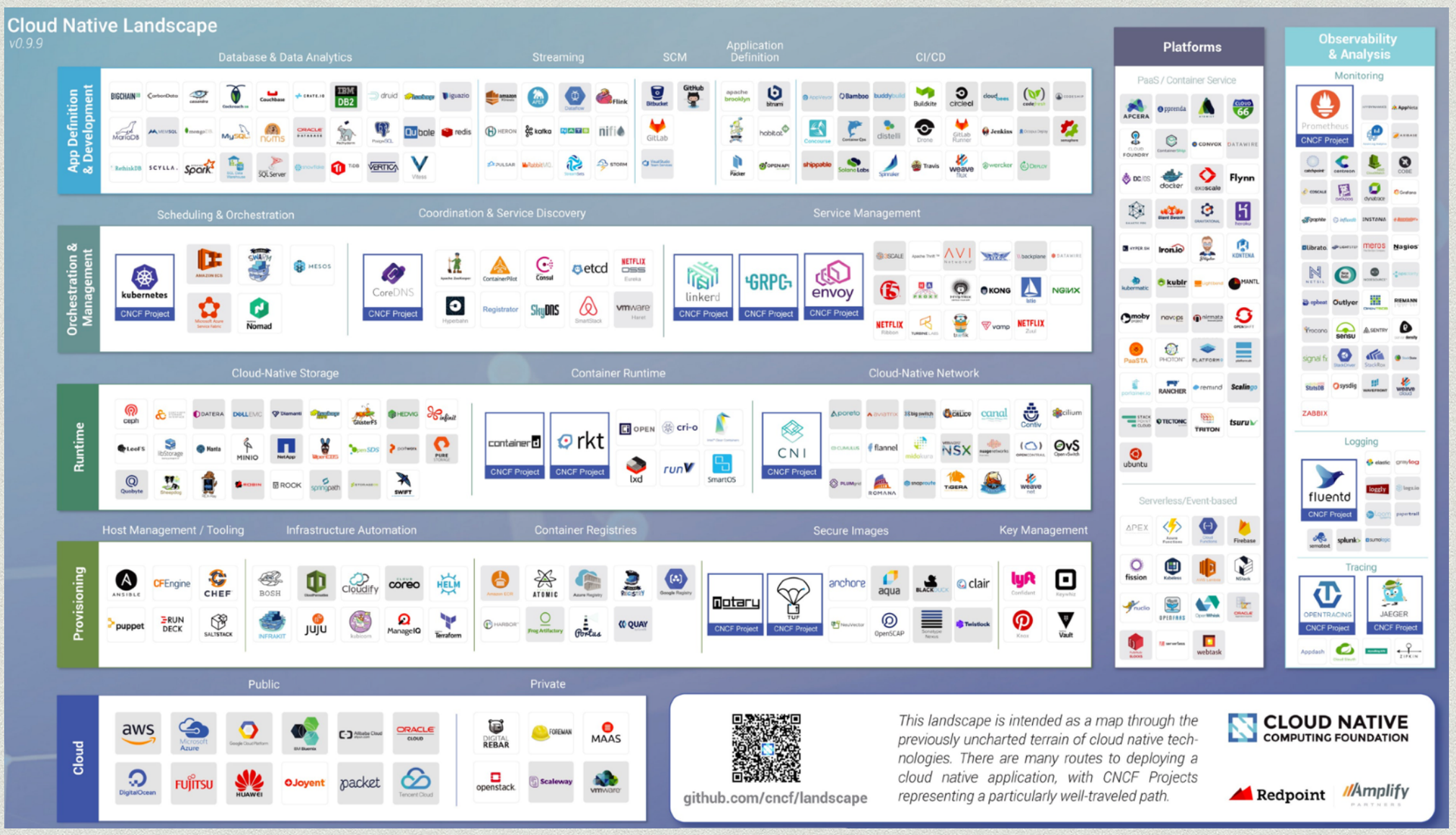
– Linus Torvalds

https://www.youtube.com/watch?v=4XpnKHJAok8

PER CONCLUDERE

Una Sfida della Unix Way oggi

- **Dalle risorse finite al cloud** è una sfida aperta
  - La complessità non sparisce, piuttosto *si sposta* dalle applicazioni all'infrastruttura
  - Esiste uno **standard** per i container (OCI), ma non per le *infrastrutture*. Una tentativo è iniziato con *Google Anthos*



«Il Sistema Operativo è tuo amico.
Usalo a tuo vantaggio!»

– *Matteo Vaccari*

/proc
/dev

TCP/IP

POSIX signals

copy on-write

permissions

descriptors

everything is a file

coreutils

netfilter

sockets
socat

netlink

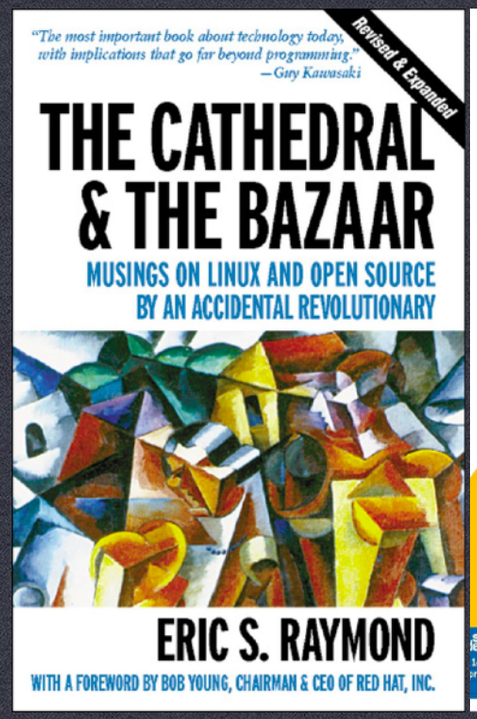
iNodes

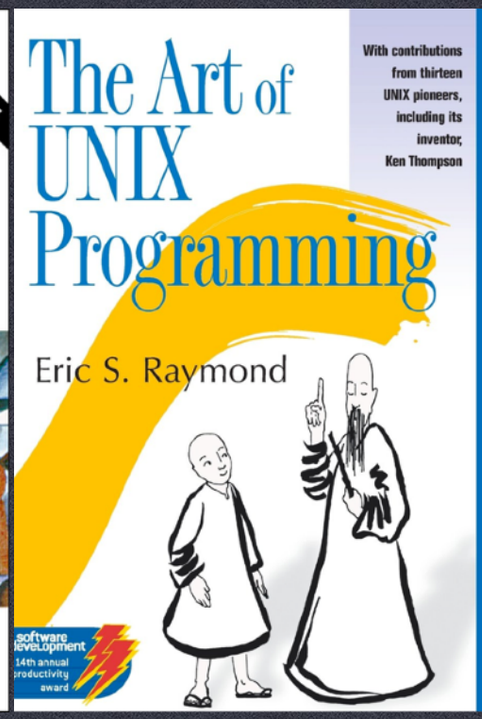
chown
cgroups

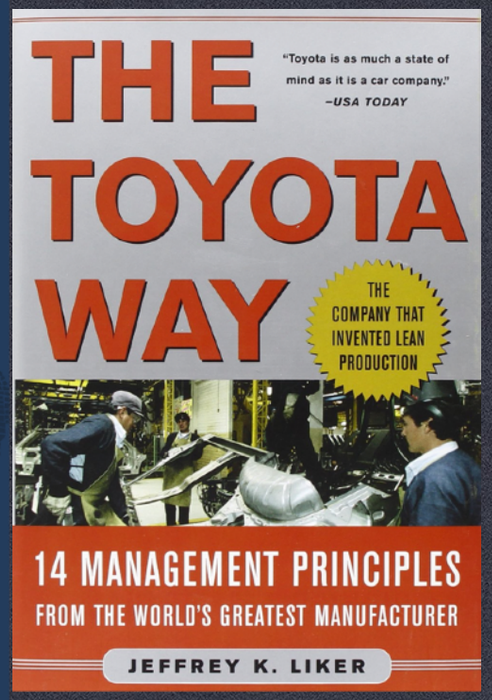
capabilities

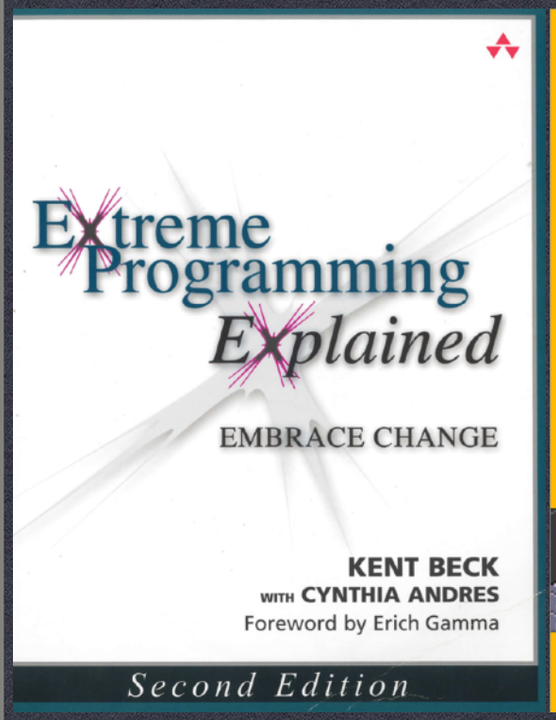
LXC
libvirt

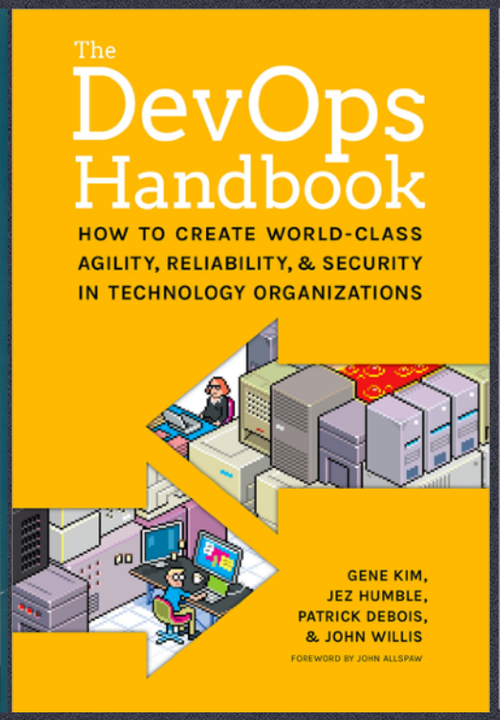
namespaces

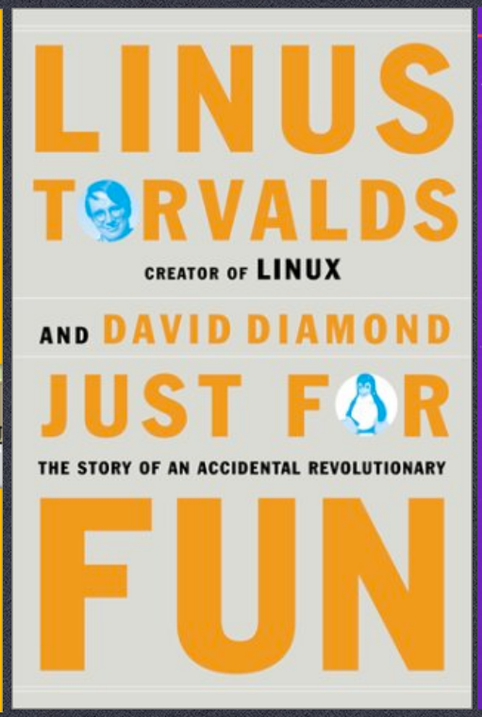


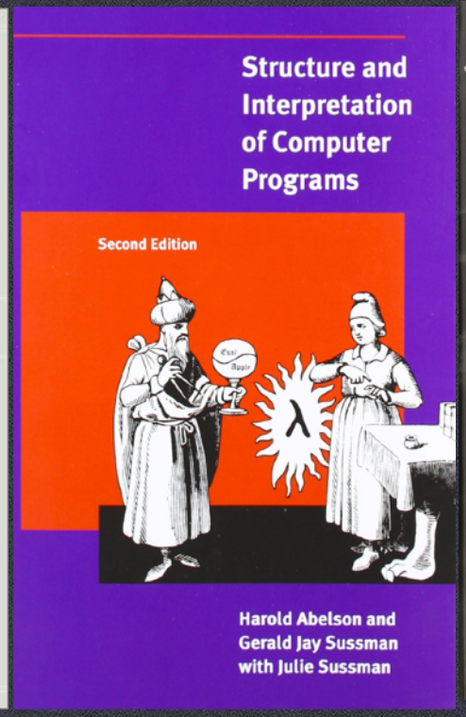


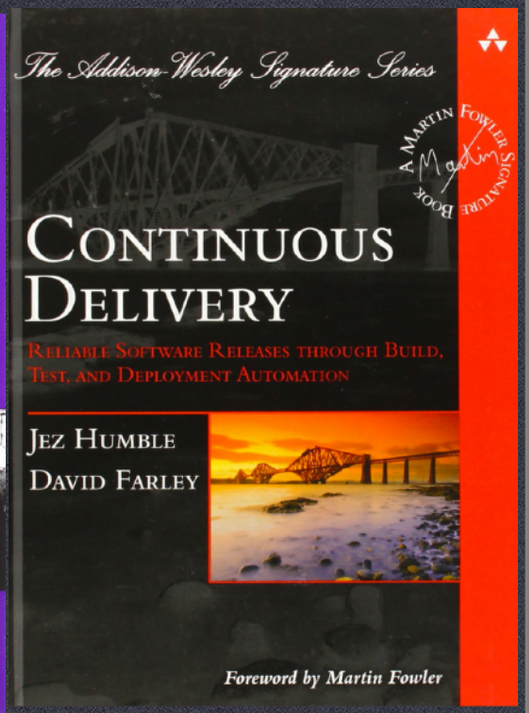












Per approfondire…

“DevOps coinvolge tutta l’azienda, rivoluzionandola e rendendola più dinamica.”

La tecnologia rappresenta una grande opportunità per la competitività delle imprese ma coniugarne lo sviluppo all’agilità e all’affidabilità dei processi non è facile. Una risposta arriva da DevOps – contrazione di *development*, “sviluppo”, e *operations*, “messa in produzione” – una metodologia di sviluppo software pensata per aiutare le aziende a realizzare prodotti e servizi in modo rapido ed efficiente. Google, Amazon, Facebook e Netflix sono solo alcune delle realtà che la utilizzano per realizzare e mantenere applicazioni di vario tipo.

Questo manuale vuole fornire una panoramica chiara e completa del movimento DevOps, mettendo in luce metodi, regole, principi e valori che sono alla base della creazione del software in modo iterativo e incrementale. Quindi si passa alla pratica, mostrando passo dopo passo tutti gli elementi che compongono un ciclo di rilascio continuo, dalla modifica sul computer del programmatore fino all’arrivo in produzione.

Una guida dedicata sia allo sviluppatore alle prime armi sia al sistemista esperto e, in spirito Agile, anche a coloro che non scrivono codice: manager, analisti, coach, consulenti, imprenditori e appassionati.

ARGOMENTI TRATTATI

- **Affrontare lo sviluppo software nel contesto di mercato moderno**
- **Scoprire le caratteristiche di un’organizzazione DevOps**
- **Capire il concetto di valore in relazione ai modelli Lean e Flow**
- **Conoscere l’evoluzione dei metodi di produzione verso Extreme Programming e DevOps**
- **Valorizzare le infrastrutture a base Unix e Linux**
- **Utilizzare Git per il mantenimento del software**
- **Comprendere il valore del testing e della Continuous Integration**
- **Realizzare container con Docker**
- **Implementare flussi di Continuous Delivery**
- **Gestire la produzione con Kanban**

Scarica i file degli esempi
https://bit.ly/apo-devops

L’AUTORE

Fabio Mora è un programmatore e Agile coach freelance entusiasta di Extreme Programming e Linux. Appassionato di open source, di economia e di tutto quello che riguarda la matematica e la scienza dei dati, ha prima fondato una web agency e poi lavorato in eBay come Senior Software Engineer. Ama la musica, l’ingegneria del suono e la divulgazione scientifica.

anche in ebook

9 788850 334506

www.apogeonline.com

apogeonline @apogeonline apogeo

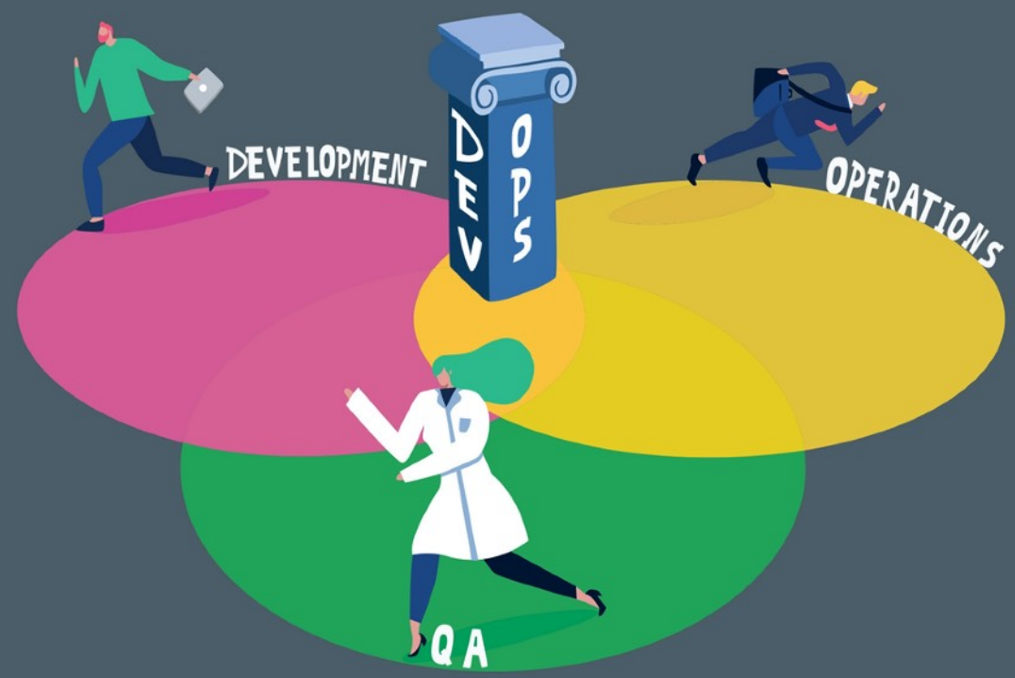
FABIO MORA

DevOps

APOGEO

FABIO MORA

DevOps



Guida per integrare Development e Operations e produrre software di qualità

APOGEO

Ho scritto questo. Da settembre è in tutte le librerie ☺️ E qui: http://bit.ly/2pBuiQh

Fonti e bibliografia

Pubblicazioni

- [Beck05] Kent Beck, Cynthia Andres, *Extreme Programming Explained – Embrace Change*, Addison-Wesley, Boston 2005.
- Eric S. Raymond, *The art of UNIX programming*, Pearson Education, 2004.
- Peter H. Salus, *A Quarter Century of UNIX*, Addison-Wesley Professional, 1994.
- Robert L. Glass, *Facts and Fallacies of Software Engineering*, Addison-Wesley, 2013.
- Craig Larman, *Agile & Iterative Development: A Manager's Guide*, Addison-Wesley Professional, 2004.
- Luigi Federico Menabrea, *Sketch of The Analytical Engine Invented by Charles Babbage*, Officer of the Military Engineers, Torino 1842, No. 82.
- Linus Torvalds, David Diamond, *Rivoluzionario per caso*, Garzanti, Milano 2011.
- Gerald M. Weinberg, *How Software Is Built*, Leanpub, 2014 (https://leanpub.com/howsoftwareisbuilt).
- James P. Womack, Daniel T. Jones, *Lean Thinking*, Productivity Press, 1996.
- Tim Wu, *The Master Switch: The Rise and Fall of Information Empires*, Atlantic Books, London 2010.
- Betsy Beyer et alii, *Site Reliability Engineering*, Google, 2016.

Fonti e bibliografia

Articoli, paper e talk

- [Vac14] Matteo Vaccari, *The UNIX Way vs. the Java Enterprise Edition Way*, 2014(https://www.slideshare.net/xpmatteo/the-unix-way-vs-the-java-enterprise-edition-way).
- Matteo Vaccari, Extreme Programming, *Agile For Innovation*, Milano, 2014 (slideshare.net/xpmatteo/extreme-programming-32324409)
- Unix Time-Sharing System, *Bell System Technical Journal*, 1978 (http://emulator.pdp-11.org.ru/misc/1978.07_-_Bell_System_Technical_Journal.pdf).
- John D. Cook, Where the Unix philosophy breaks down, 30 giugno 2010 (https://www.johndcook.com/blog/2010/06/30/where-the-unix-philosophy-breaks-down/).
- Patrick Debois, *Agile Infrastructure & Operations*, Agile 2008 Conference (http://www.jedi.be/presentations/agile-infrastructure-agile-2008.pdf).
- Dennis M. Ritchie, Ken Thompson, *The UNIX Time Sharing System, Bell Laboratories*, 1974 (https://people.eecs.berkeley.edu/~brewer/cs262/unix.pdf).
- Chris Tozzi, *Want to understand DevOps? Look to open source's history* (https://devopsagenda.techtarget.com/opinion/Want-to-understand-DevOps-Look-to-open-sources-history).
- Janet Kornblum, Netscape sets source code free, 31 marzo 1998 (https://www.cnet.com/news/netscapesets-source-code-free/).
- Linus Torvalds, *Tech Talk: Linus Torvalds on git* (https://www.youtube.com/watch?v=4XpnKHJAok8).
- https://www.cnet.com/news/test-version-of-new-linux-kernel-available/
- http://linux.sys-con.com/node/32707

Fonti e bibliografia

Immagini

- https://www.google.com/about/datacenters/
- https://upload.wikimedia.org/wikipedia/commons/c/cf/Diagram_for_the_computation_of_Bernoulli_numbers.jpg
- https://www.scienceandsociety.co.uk (#10303273, Punched cards for Babbage's Analytical Engine)
- https://www.ibm.com/ibm/history/ibm100/us/en/icons/system360/transform/
- https://it.wikipedia.org/wiki/ENIAC#/media/File:Eniac.jpg
- https://it.wikipedia.org/wiki/Robert_Cecil_Martin#/media/File:Robert_Cecil_Martin.png
- Massimo Ubertini, Architettura degli elaboratori (http://www.ubertini.it/pagine/archi-elab.html)
- https://www.worldcat.org
- https://www.zdnet.com/article/after-linux-civil-war-ubuntu-to-adopt-systemd/

@fabiomora — fabiomora.com

Grazie!

Domande?
Suggerimenti?