Seminários IC Unicamp

Paulo Meirelles e Leonardo Leite

{paulormm,leofl}@ime.usp.b

IME INSTITUTO DE MATEMÁTICA E ESTATÍSTICA UNIVERSIDADE DE SÃO PAULO

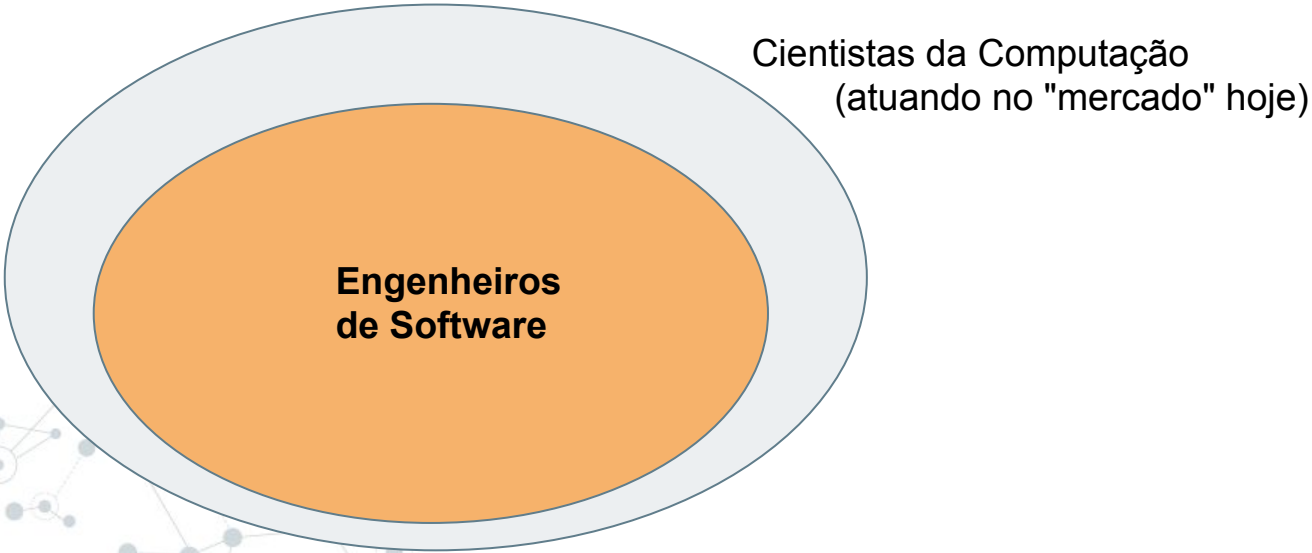
USP

Métodos Empíricos de Desenvolvimento de Software

Os Impactos do Software Livre e das Metodologias Ágeis na Engenharia de Software

Importância da Engenharia de Software

- Existe grande chance dos nossos alunos serem **Engenheiro de Software** (full stack developer, frontend developer, arquiteto, etc)



Conferência da OTAN (Alemanha, 1968)

- 1a vez que o termo Engenharia de Software foi usado

Comentário de participante da Conferência da OTAN

"Certos sistemas estão colocando demandas que estão além das nossas capacidades… Em algumas aplicações não existe uma crise... mas **estamos tendo dificuldades com grandes aplicações."**

Definição de Engenharia de Software

Área da Computação destinada a investigar os desafios e propor soluções que permitam desenvolver sistemas de software — principalmente aqueles mais complexos e de maior tamanho — de forma produtiva e com qualidade

O que se estuda em ES?

1. Engenharia de Requisitos
2. Projeto de Software
3. Construção de Software
4. Testes de Software
5. Manutenção de Software
6. Gerência de Configuração
7. Gerência de Projetos

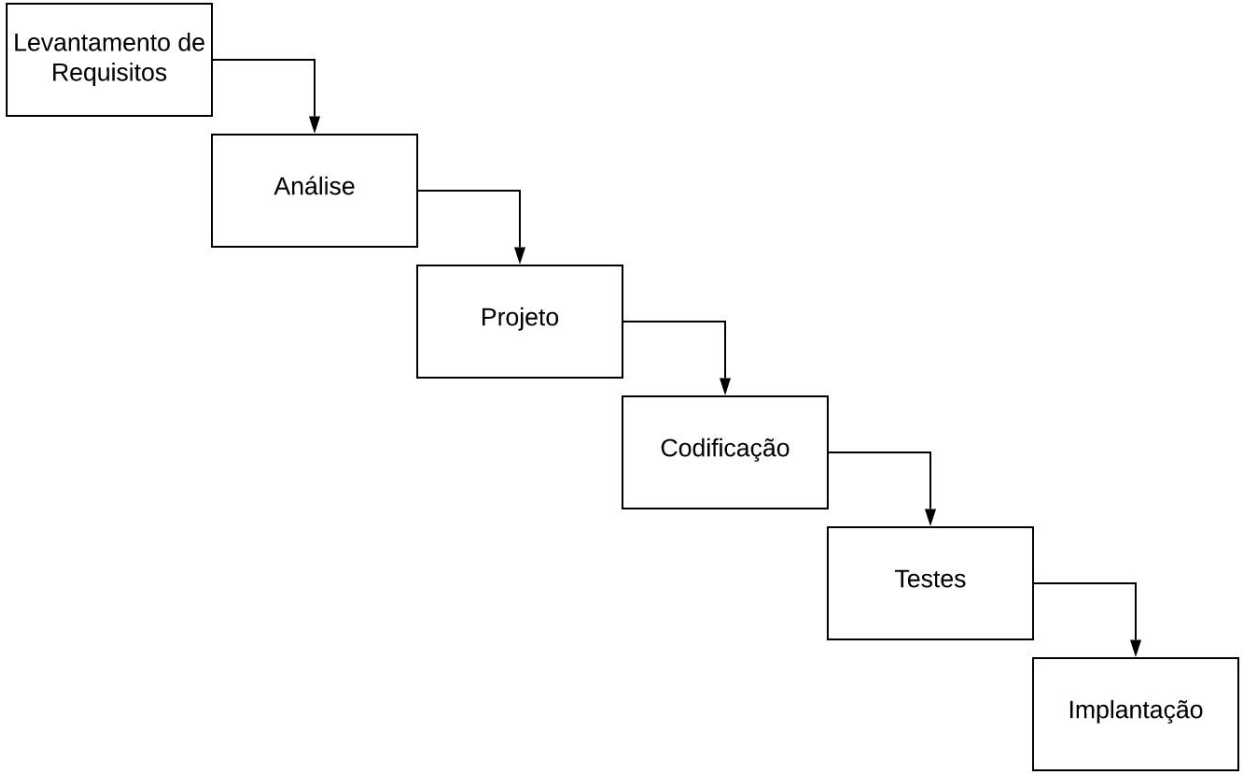
O que se estuda em ES?

8. Processos de Software
9. Modelos de Software
10. Qualidade de Software
11. Prática Profissional
12. Aspectos Econômicos

Engenharia Tradicional

- Civil, mecânica, elétrica, aviação, automobilística, etc
- Projeto com duas características:
  - Planejamento detalhado (*big upfront design*)
  - Sequencial
- Isto é: Waterfall
  - Há milhares de anos

Natural que ES começasse usando Waterfall



No entanto: Waterfall não funcionou com software!

Software é diferente

- Engenharia de Software ≠ Engenharia Tradicional
- Software ≠ (carro, ponte, casa, avião, celular, etc)
- Software ≠ (produtos físicos)
- Software é abstrato e "adaptável"

Dificuldade 1: Requisitos

- Clientes não sabem o que querem (em um software)
  - Funcionalidades são "infinitas" (difícil prever)
  - Mundo muda!
- Não dá mais para ficar 1 ano levantando requisitos, 1 ano projetando, 1 ano implementando, etc
- Quando o software ficar pronto,

ele estará obsoleto!

Dificuldade 2: Documentações Detalhadas

- Verbosas e pouco úteis
- Na prática, desconsideradas durante implementação
- *Plan-and-document* não funcionou com software

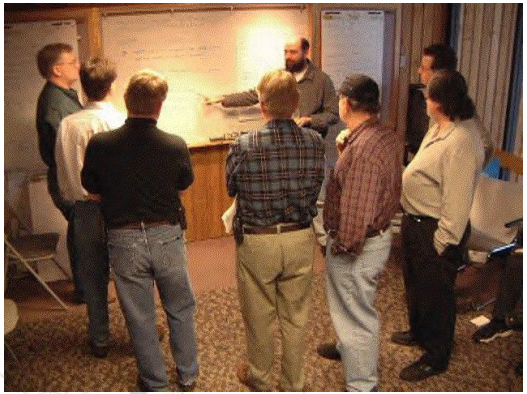
"A parte mais difícil da construção de um software é a definição do que se deve construir" -- Fred Brooks

Problemas com Modelo Waterfall

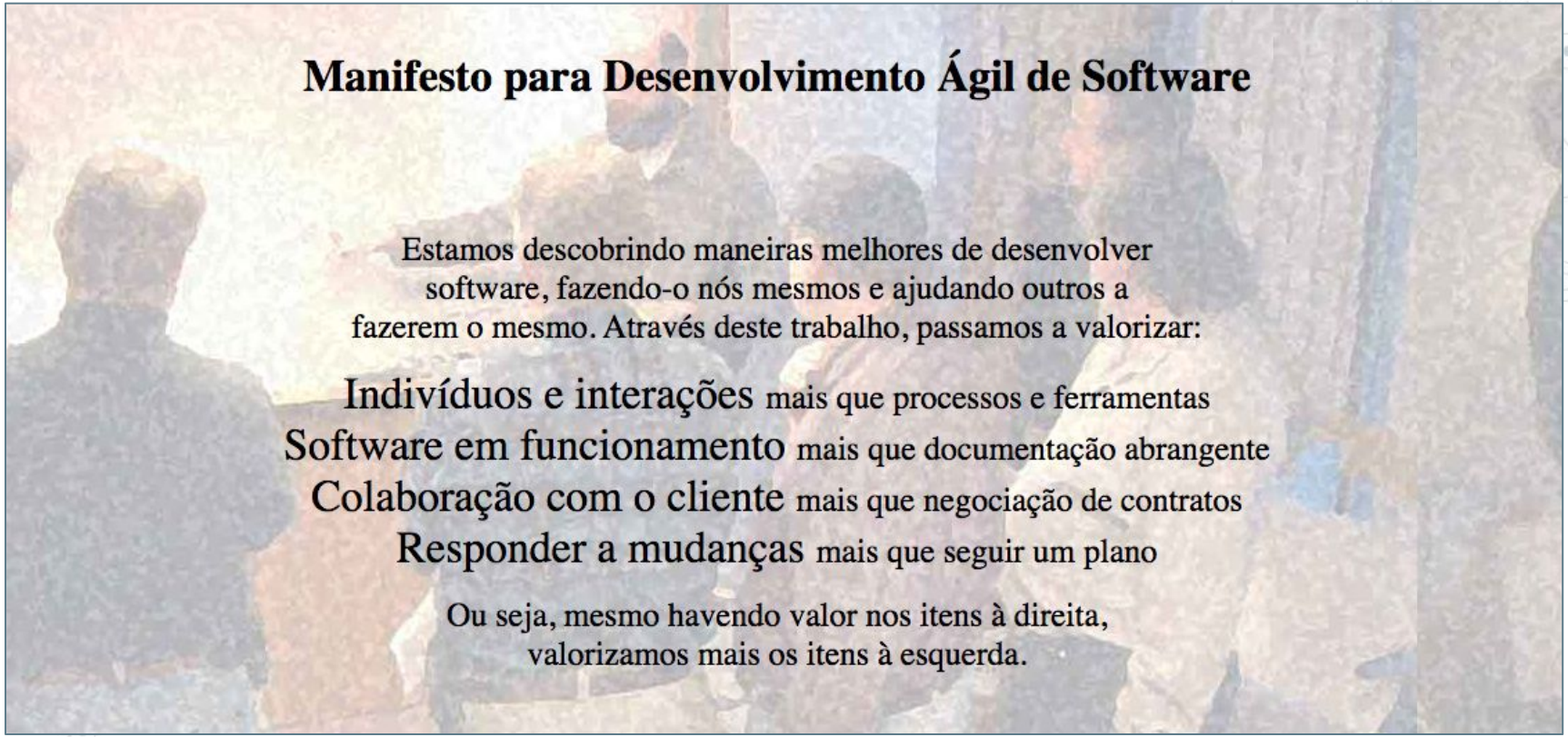
- Requisitos mudam com frequência
  - Levantamento completo de requisitos demanda tempo
  - Quando ficar pronto, o "mundo já mudou"
  - Clientes às vezes não sabem o que querem
- Documentações de software são verbosas
- E rapidamente se tornam obsoletas

Manifesto Ágil (2001)

- Encontro de 17 engenheiros de software em Utah
- Crítica a modelos sequenciais e pesados
- Novo modelo: incremental e iterativo



https://siamchamnankit.co.th/history-some-pictures-and-pdfs-of-the-agile-manifesto-meeting-on-2001-a33c40bcc2b

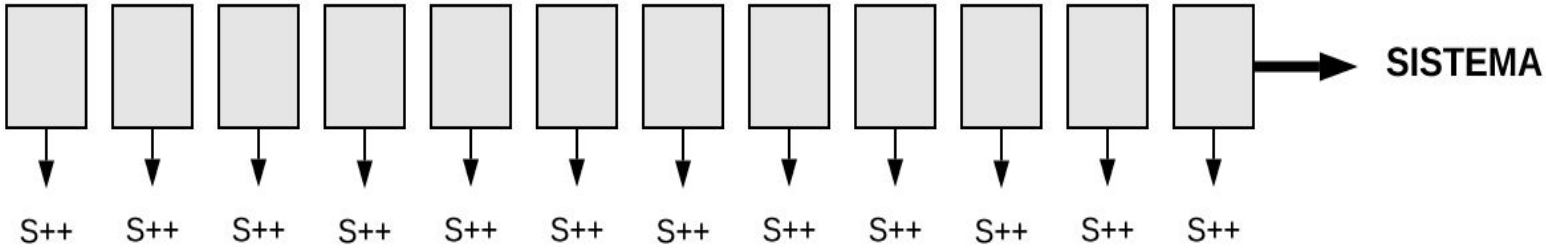


https://agilemanifesto.org/iso/ptbr/manifesto.html

Ideia central: desenvolvimento iterativo

Waterfall

Ágil

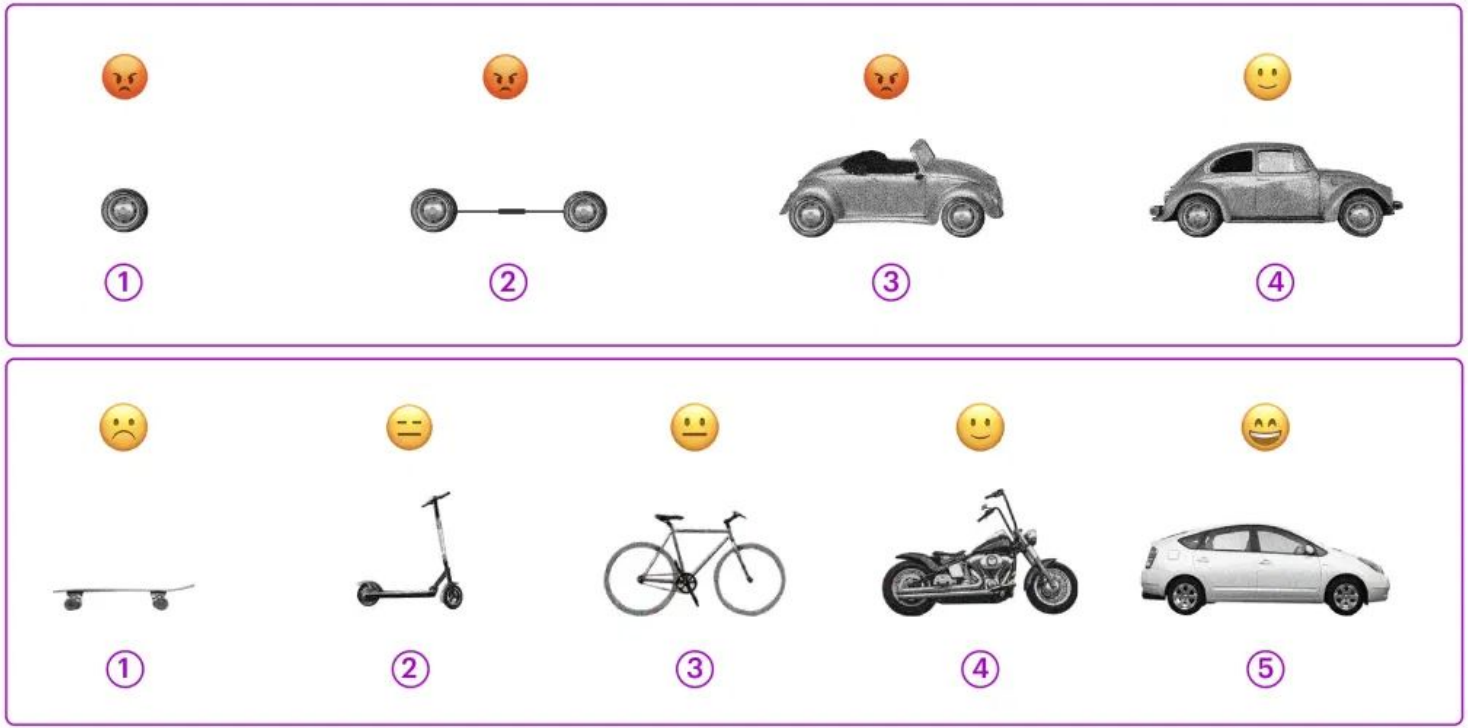


Desenvolvimento iterativo

- Suponha um sistema imenso, complexo etc
- Qual o menor "incremento de sistema" eu consigo implementar em 15 dias e validar com o usuário?
- Validar é muito importante!
- Cliente não sabe o que quer!

Reforçando: ágil = iterativo

Implementando um carro (Produto Mínimo Viável, MVP)



Fonte: https://blog.nubank.com.br/mvp-o-que-e-nubank/

Outros pontos importantes (1)

- Menor ênfase em documentação
- Menor ênfase em *big upfront design*
- Envolvimento constante do cliente (*product owner*)

Outros pontos importantes (2)

- Novas práticas de programação
  - Testes, refactoring, integração contínua, etc

Desenvolvimento Ágil

- Profundo impacto na indústria de software
- Hoje, tudo é ágil… Talvez adjetivo até desgastado

Março 2019

Maio 2020

Métodos Ágeis

Métodos Ágeis

- Dão mais consistência às ideias ágeis
  - Definem um processo, mesmo que leve
  - Workflow, eventos, papeis, práticas, princípios etc

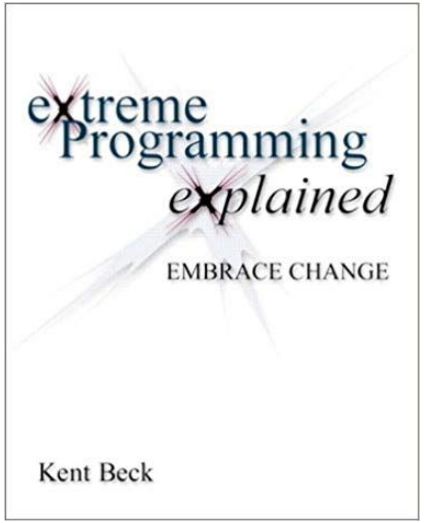
Antes de começar

- Nenhum processo é uma bala-de-prata
- Processo ajuda a não cometer certos "grandes erros"
- Processos não são adotados 100% igual ao manual
  - Bom senso é importante
  - Experimentação é importante

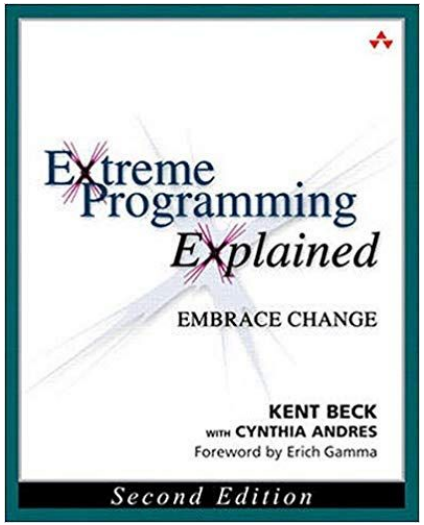
Extreme Programming (XP)

Extreme Programming

Kent Beck



1999



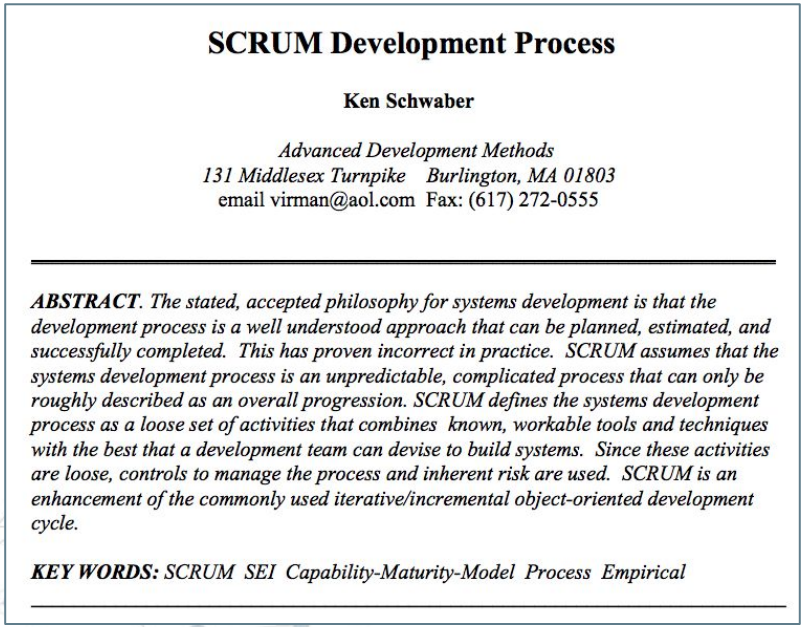
2004

XP = Valores + Princípios + Práticas

Scrum

Scrum

- Proposto por Jeffrey Sutherland e Ken Schwaber (OOPSLA 1995)



SCRUM Development Process

Ken Schwaber

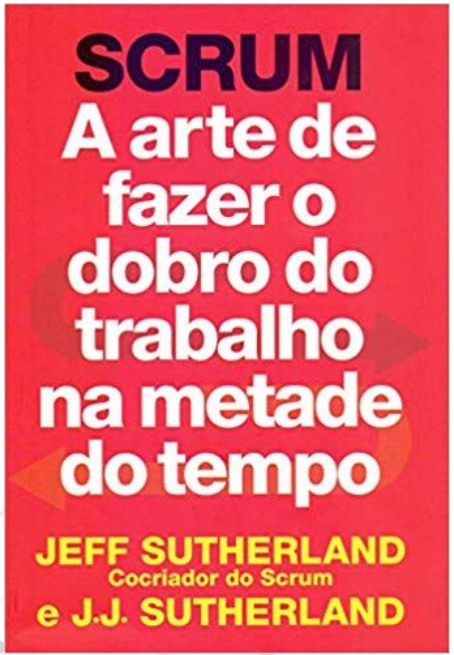
Advanced Development Methods
131 Middlesex Turnpike Burlington, MA 01803
email virman@aol.com Fax: (617) 272-0555

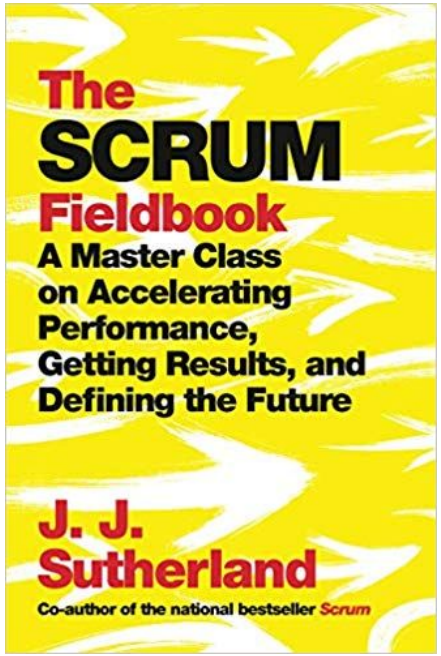
ABSTRACT. *The stated, accepted philosophy for systems development is that the development process is a well understood approach that can be planned, estimated, and successfully completed. This has proven incorrect in practice. SCRUM assumes that the systems development process is an unpredictable, complicated process that can only be roughly described as an overall progression. SCRUM defines the systems development process as a loose set of activities that combines known, workable tools and techniques with the best that a development team can devise to build systems. Since these activities are loose, controls to manage the process and inherent risk are used. SCRUM is an enhancement of the commonly used iterative/incremental object-oriented development cycle.*

KEY WORDS: *SCRUM SEI Capability-Maturity-Model Process Empirical*

Scrum

- Scrum é uma indústria: livros, consultoria, certificações, marketing





Scrum vs XP

- Scrum não é apenas para projetos de software
  - Logo, não define práticas de programação, como XP
- Scrum define um "processo" mais rígido que XP
  - Eventos, papéis e artefatos bem claros

Antes disso, um outro (contra) movimento também estava acontecendo ...

Software Livre
(Free/Libre/Open Source Software)

Linux kernel

63 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

The **Linux kernel** is a free and open-source,[12]:4 monolithic, modular, multitasking, Unix-like operating system kernel. It was originally written in 1991 by Linus Torvalds for his i386-based PC, and it was soon adopted as the kernel for the GNU operating system, which was written to be a free (libre) replacement for Unix.

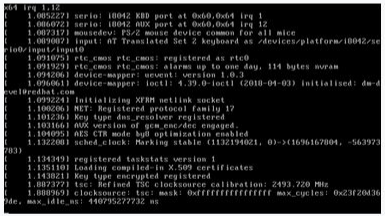
Linux is provided under the GNU General Public License version 2 only, but it contains files under other compatible licenses.[11] Since the late 1990s, it has been included as part of a large number of operating system distributions, many of which are commonly also called Linux.

Linux is deployed on a wide variety of computing systems, such as embedded devices, mobile devices (including its use in the Android operating system), personal computers, servers, mainframes, and supercomputers.[13] It can be tailored for specific architectures and for several usage scenarios using a family of simple commands (that is, without the need of manually editing its source code before compilation);[14][15][16] privileged users can also fine-tune kernel parameters at runtime.[17][18][19] Most of the Linux kernel code is written using the GNU extensions of GCC[12]:18[20] to the standard C programming language and with the use of architecture-specific instructions (ISA) in limited parts of the kernel. This produces a highly optimized executable (vmlinux) with respect to utilization of memory space and task execution times.[12]:379–380

Day-to-day development discussions take place on the Linux kernel mailing list (LKML). Changes are tracked using the version control system git, which was originally authored by Torvalds as a free software replacement for BitKeeper.

Linux kernel

Tux the penguin, mascot of Linux[1]



Linux kernel 3.0.0 booting

Original author(s)	Linus Torvalds
Developer(s)	Community contributors Linus Torvalds
Initial release	0.02 (5 October 1991; 31 years ago)
Stable release	6.4.12[2] / 23

Histórico do Software Livre

- 1976 - Bill Gates e sua “carta aberta aos hobbistas”
  - Software para microcomputadores tem grande potencial comercial
  - Não é possível financiar desenvolvimento fora do modelo de prateleira

Histórico do Software Livre

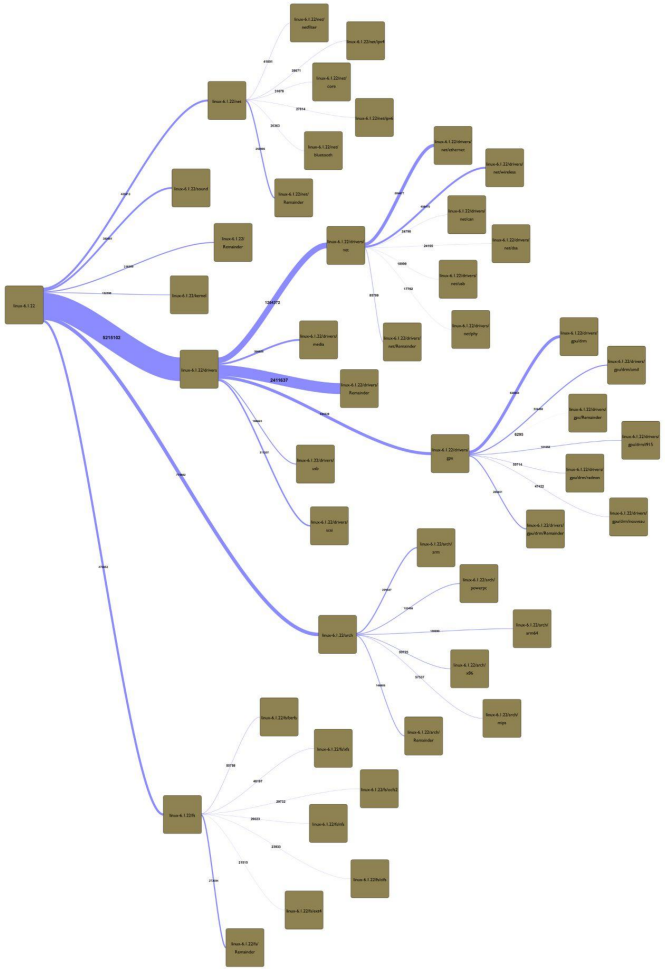
- 1984 - Richard Stallman lança o projeto GNU
  - uso de software restrito não é ético
- 1985 - A Free Software Foundation é fundada por Richard Stallman
  - dedica-se a eliminação de restrições sobre a cópia, estudo e modificação de programas de computadores bandeiras do movimento do software livre, em essência

Histórico do Software Livre

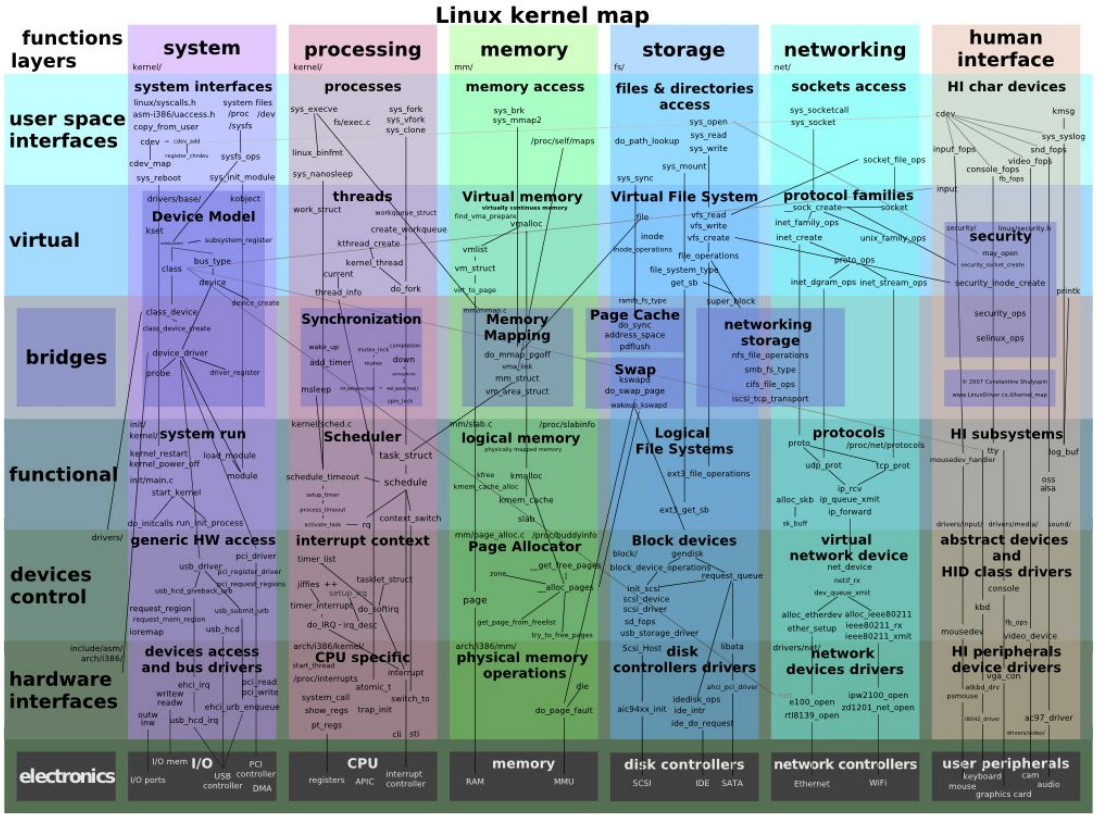
- 1991 - É lançamento/surgimento do Linux
  - Linux refere-se aos sistemas operacionais que utilizam o Kernel Linux
  - O núcleo foi desenvolvido por Linus Torvalds, inspirado no sistema Minix, como projeto informal, apenas para estudo

Histórico do Software Livre

- 1995 – Boom da Internet
  - tem início a comercialização em escala do Linux, por exemplo, o Red Hat Linux.



CC BY-SA 4.0: Linux Kernel 6.1.22 Sankey Flow Diagram

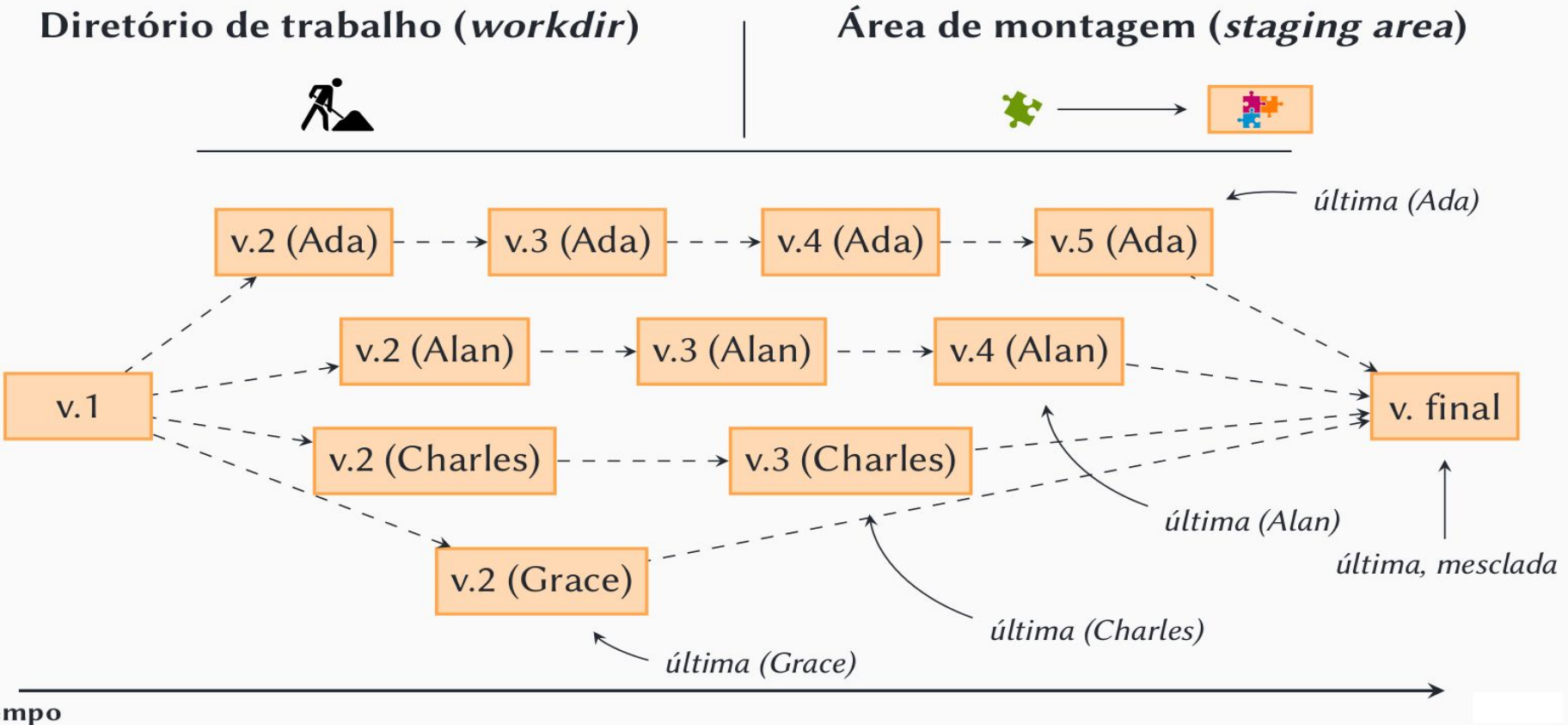


CC-BY 3.0: http://www.makelinux.net/kernel_map

Fonte: https://en.wikipedia.org/wiki/Linux_kernel

“Cada um faz sua parte e depois a gente junta tudo”

- Cada um vê uma linha do tempo diferente!
- Cada um vê uma “última” versão (o “presente”) diferente!



https://www.casadocodigo.com.br/products/livro-como-se-faz-devops

Obrigado!

Seminários IC Unicamp

Paulo Meirelles e Leonardo Leite

{paulormm,leofl}@ime.usp.b

IME INSTITUTO DE MATEMÁTICA E ESTATÍSTICA UNIVERSIDADE DE SÃO PAULO

USP