Softwarový proces

KIV/ASWI 2014/2015

Obsah

- Aspekty ovlivňující proces
- Motivace pro inženýrský přístup
- Pojmy
- Varianty procesu

Softwarový proces: statistiky

Standish Group „Chaos Report“ 1995

FAILURE RECORD

In the United States, we spend more than $250 billion each year on IT application development of approximately 175,000 projects. The average cost of a development project for a large company is $2,322,000; for a medium company, it is $1,331,000; and for a small company, it is

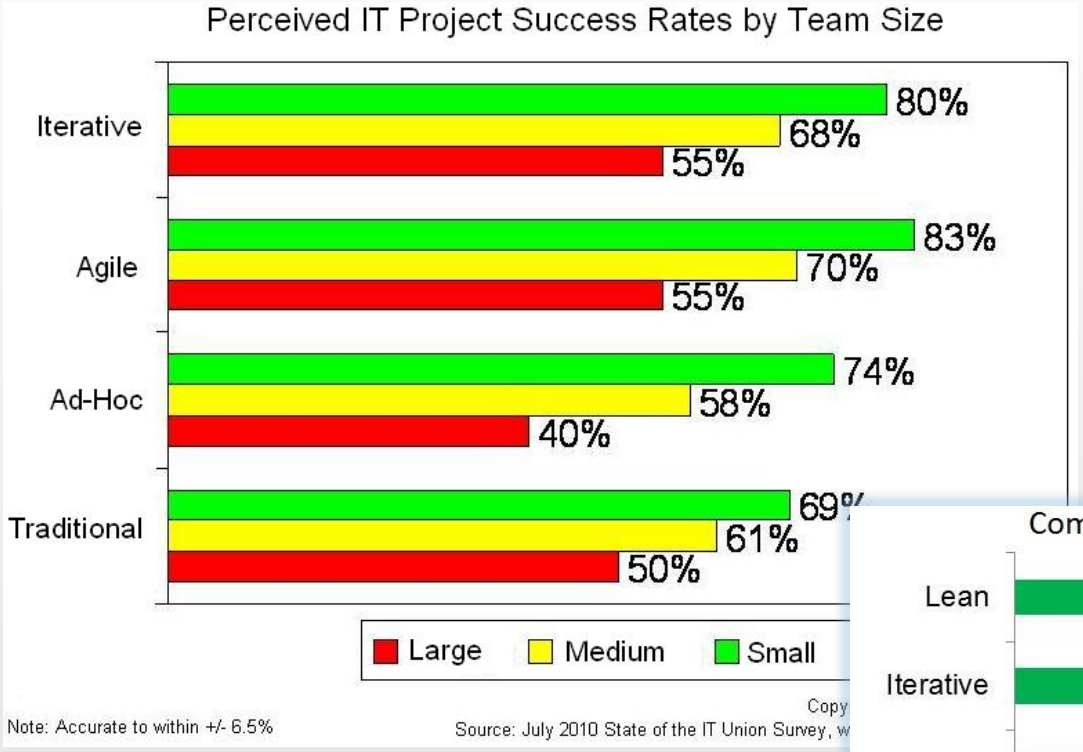
manufacturing, retail, wholesale, health care, insurance, services, and local, state, and federal organizations. The total sample size was 365 respondents and represented 8,380 applications. In addition, The Standish Group conducted four focus groups and numerous personal interviews to provide qualitative context for the survey results.

For purposes of the study, projects were classified into three resoluti

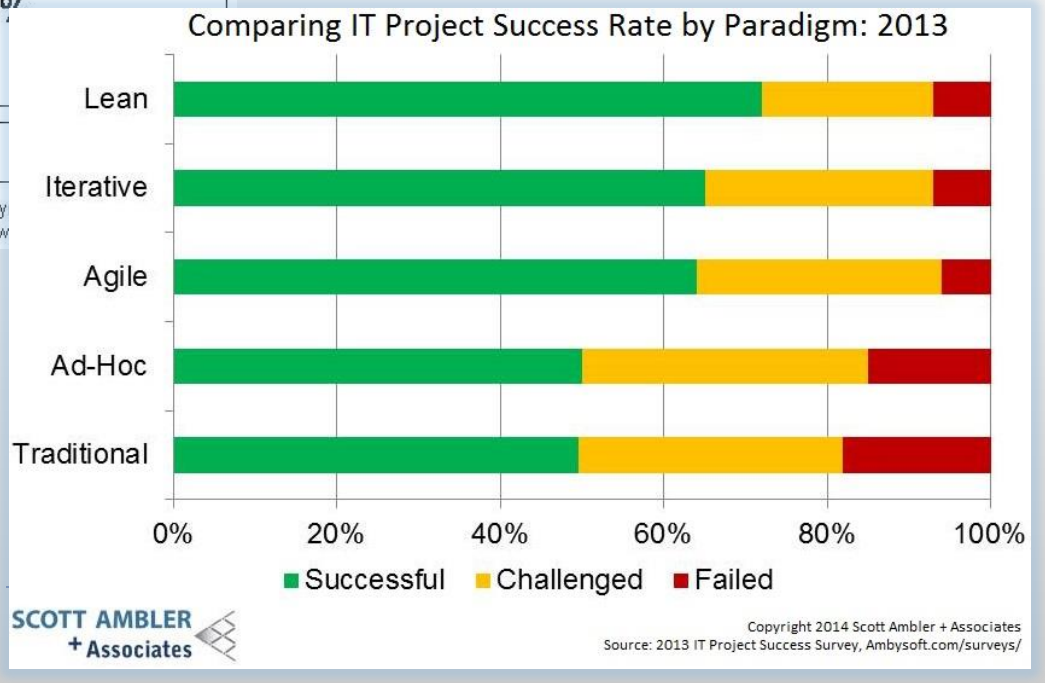
- Resolution Type 1, or project success: The project is completed or with all features and functions as initially specified.
- Resolution Type 2, or project challenged: The project is complete over-budget, over the time estimate, and offers fewer features originally specified.
- Resolution Type 3, or project impaired: The project is cancelled at development cycle.

Overall, the success rate was only 16.2%, while challenged projects accounted for 52.7%, and impaired (cancelled) for 31.1%.

Scott Ambler, 2010+



Some 20 years later …



How do we define software development success?

96% Meet the actual needs of stakeholders

90% Delivery high-

83% Provide the best

81% Deliver when the s

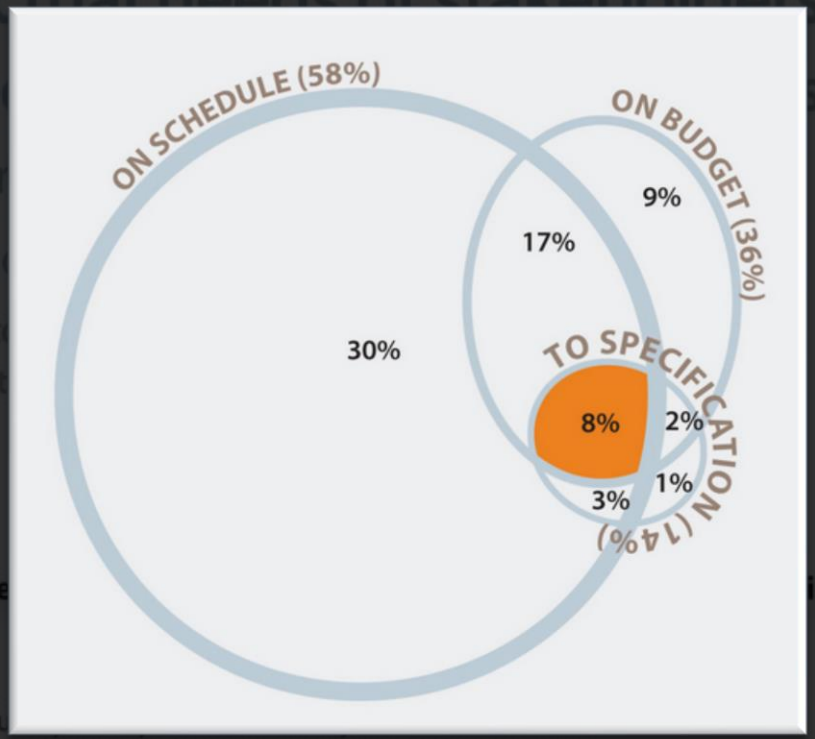
58% Deliver on time according t

44% Deliver on time and on budget

36% Deliver on or under budget

14% Build the system to specification

It is time to recognize that people **in** traditional terms.



Source: 2013 IT Project Success Rates Su

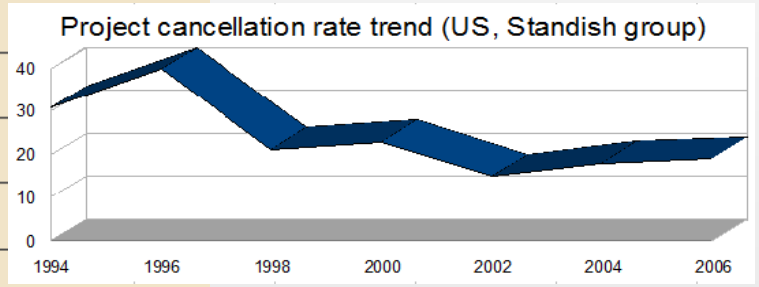
SCOTT AMBLER
+ Associates

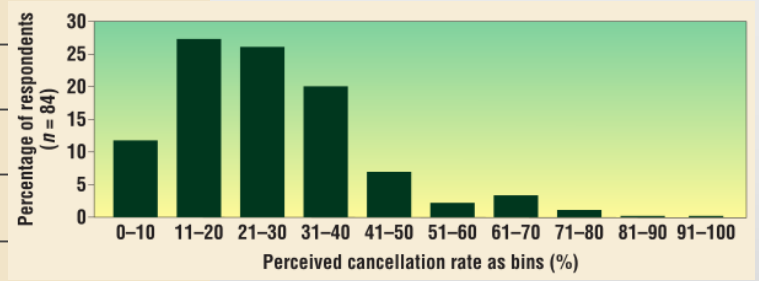
Realita stavu SWI

Data:
Emam, Koru: A Replicated Survey of IT Software Project Failures, IEEE Software 25(5), 2008

A summary of evidence on software project cancellation rates*

Study, year, and location	Cancellation/abandonment rate (%)
Standish Group, 1994, US	31
Standish Group, 1996, US	40
Standish Group, 1998, US	28
Jones,[8] 1998, US (systems projects)	14
Jones,[8] 1998, US (military projects)	19
Jones,[8] 1998, US (other projects)	> 24
Standish Group, 2000, US	23
Standish Group, 2002, US	15
Computer Weekly,[9] 2003, UK	9
UJ,[10] 2003, South Africa	22
Standish Group, 2004, US	18
Standish Group, 2006, US	19





Realita stavu SWI

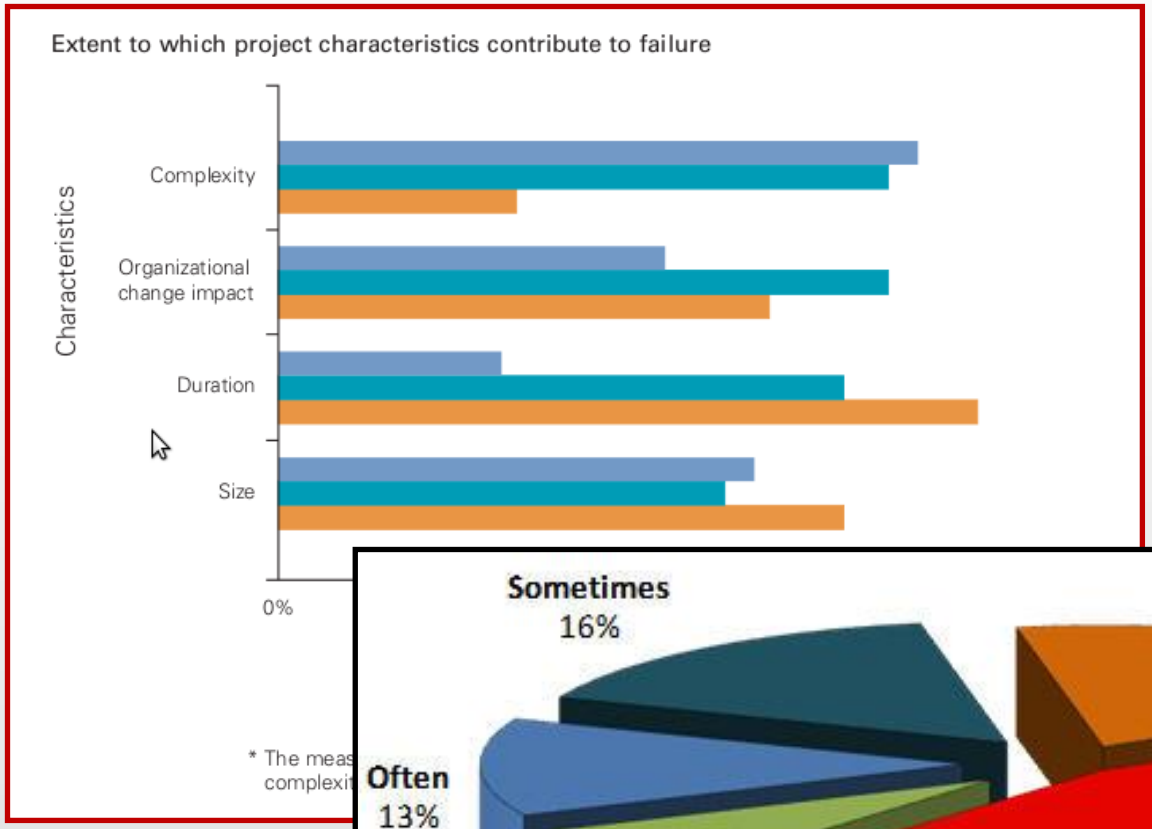
Data:
Standish Group: Chaos Report 1995
Emam, Koru: A Replicated Survey of IT Software Project Failures, IEEE Software 25(5), 2008

Project Challenged Factors	% of Responses
1. Lack of User Input	12.8%
2. Incomplete Requirements & Specifications	12.3%
3. Changing Requirements & Specifications	11.8%
4. Lack of Executive	
5. Technology Incom	
6. Lack of Resources	
7. Unrealistic Expecta	
8. Unclear Objectives	
9. Unrealistic Time Fr	
10. New Technology	
Other	

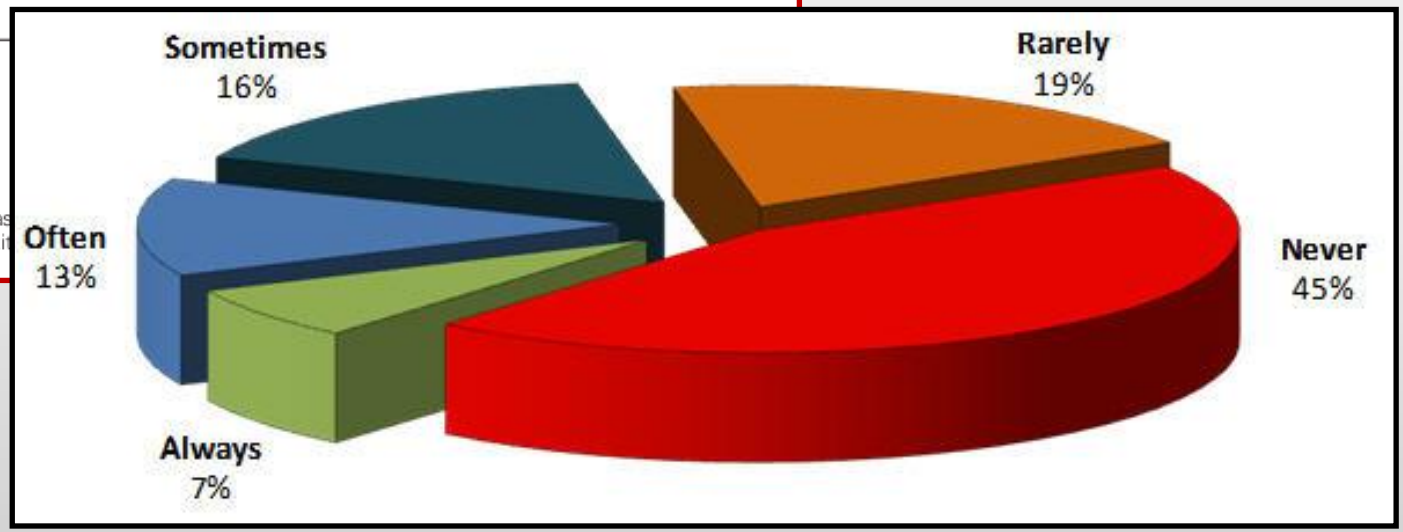
Reason for cancellation	Percentage of respondents (95% confidence interval)
Senior management not sufficiently involved	33 (13, 59)
Too many requirements and scope changes	33 (13, 59)
Lack of necessary management skills	28 (10, 54)
Over budget	28 (10, 54)
Lack of necessary technical skills	22 (6, 48)
No more need for the system to be developed	22 (6, 48)
Over schedule	17 (4, 41)
Technology too new; didn't work as expected	17 (4, 41)
Insufficient staff	11 (1, 35)
Critical quality problems with software	11 (1, 35)
End users not sufficiently involved	6 (0, 27)

Realita stavu SWI

Data:
Scott Ambler, 2011
Standish Group, 2002



The Standish data are NOT a good indicator of poor software development performance. However, they ARE an indicator of systemic failure of our planning and measurement processes.

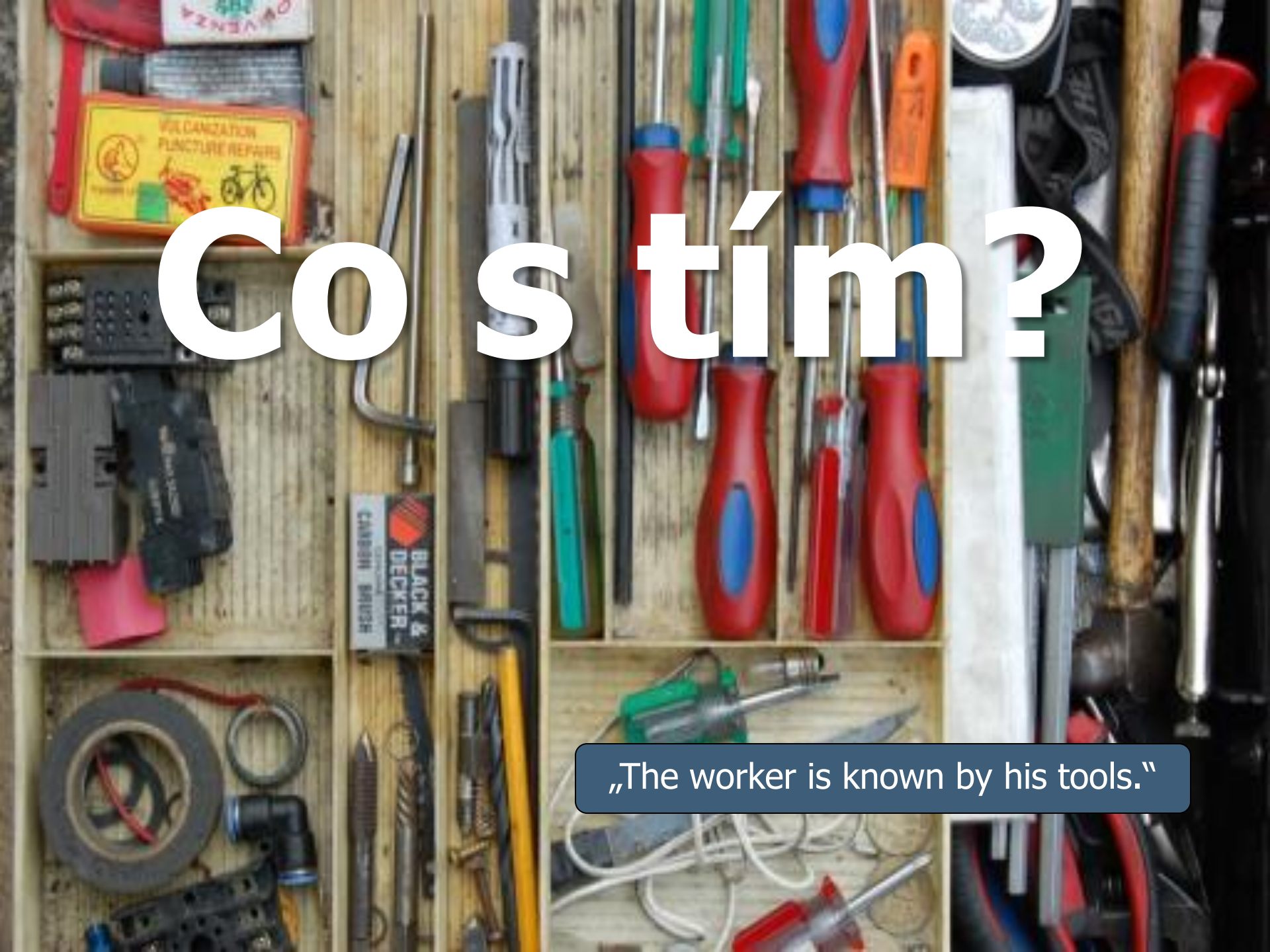


"One study [Jarzombek99] cited a 1995 DoD software project study (of over $37 billion USD worth of projects) showing that 46% of the systems so egregiously did not meet the real needs (although they met the specficications) that they were never successfully used, and another 20% required extensive rework to meet the true needs…"

– Larman: Agile and Iterative Development: A Manager's Guide, p.75

Kolik je $17 mld?

– tucet komerčních letů na Měsíc [google „project apollo cost"]
– … nebo dva tucty čtvrthodinek beztíže s Virgin Galactic [http://www.virgingalactic.com/booking/]
– 3x cena majority v ČTc
– výše dotace EU do zemědělství na jeden rok [google „14 miliard EUR"]
– 3/4 nákladů na přechod ČR od centrálně plánované ekonomiky na ekonomiku tržní [MFČR]
– cca 1/2 celkové ceny lunárního programu Apollo [google „project apollo cost"]
– dávky sociálního a důchodového pojištění ČR 2014 [http://www.ceskeinfografiky.cz/]



Co s tím?

„The worker is known by his tools.“

Softwarový proces: aspekty

Vývoj software má N rozměrů

- … běžná aktivita v informační společnosti

Na zakázku
Interní projekt
Krabicový software
„Pro radost“

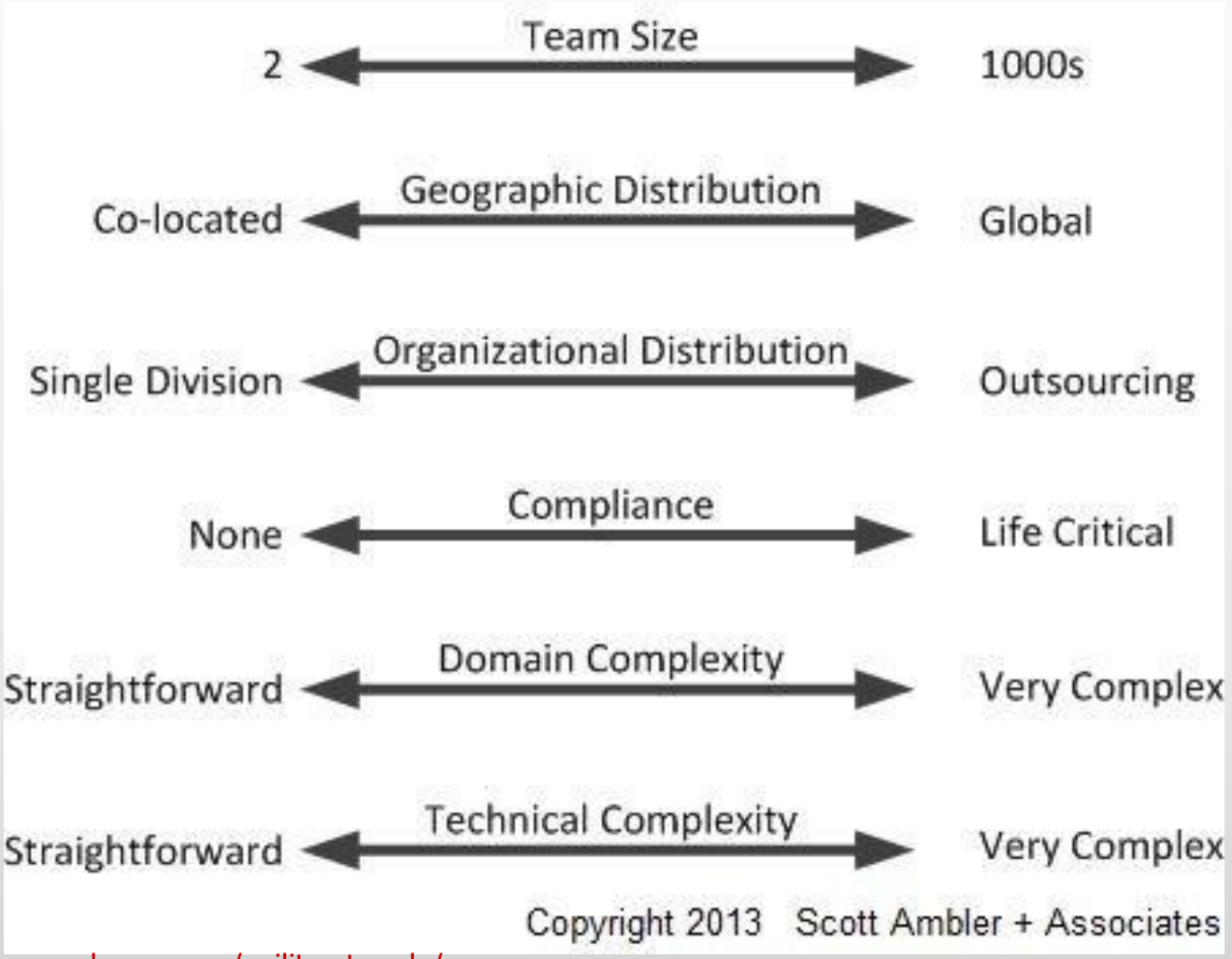
Closed source
Open source
(+ reuse)

Utilita
Systémová komponenta
Mission-critical software

Na zelené louce (green field)
Rozvoj existujícího produktu
Integrační projekt

Komerční zákazník
Státní sféra
Vertikály (utility, banky, telco, …)

Complexity factors of the Software Development Context Framework

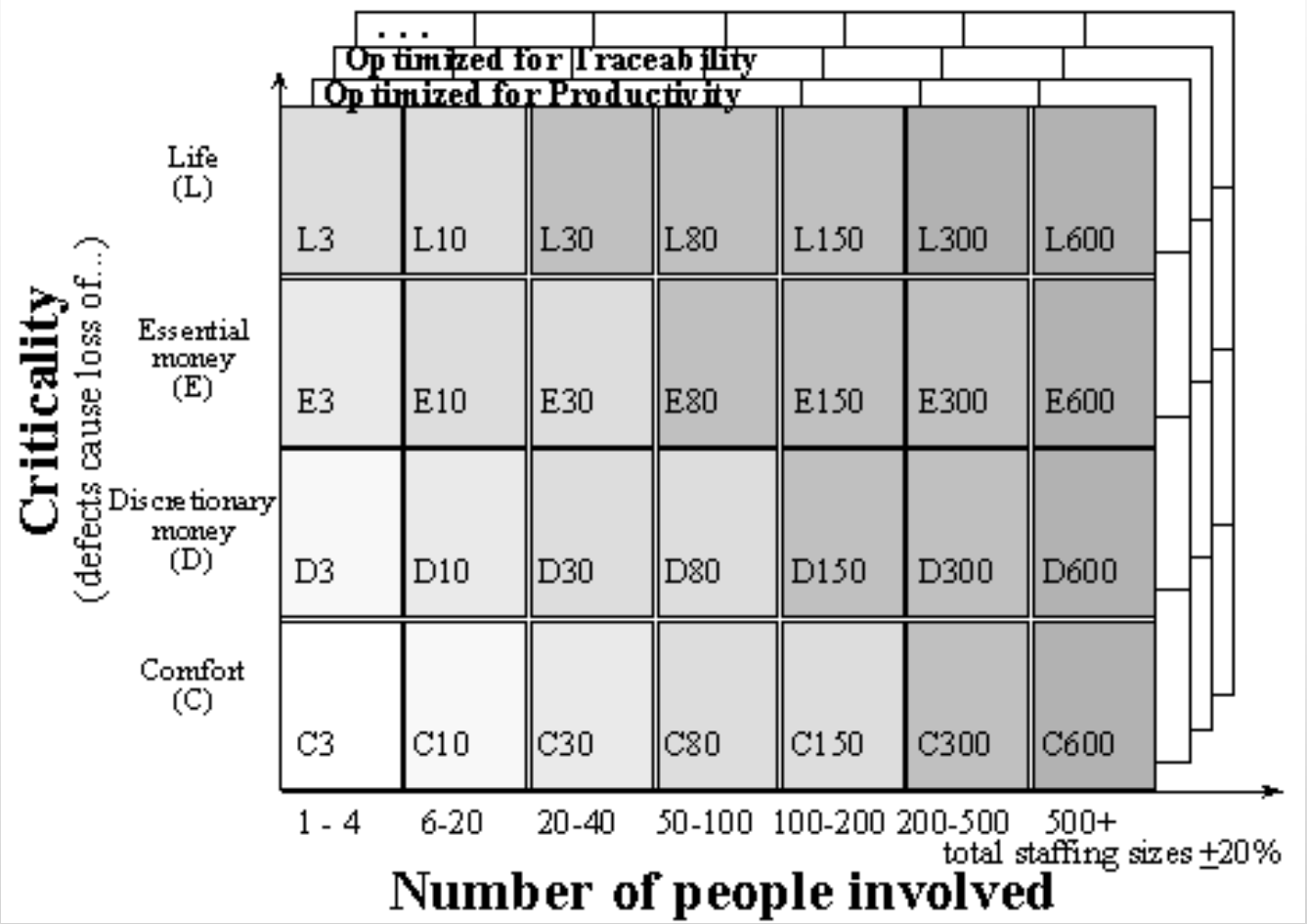


Doplňková četba:
https://disciplinedagiledelivery.wordpress.com/agility-at-scale/

Vývoj software má N rozměrů

- Rozdíl
  produkt
  x
  systém
- Faktor
  obtížnosti
  a rozsahu

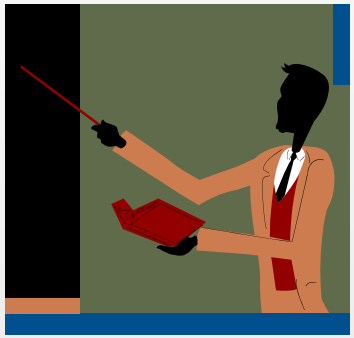
http://alistair.cockburn.us/Methodology+per+project



Stakeholders (a jejich cíle)

Zákazník

Dodavatel ~ vývojář



Učitel / mentor

Daňový poplatník

Student

Cíle dodavatele

- Vytvořit aplikaci co možná
  - nejefektivněji (zdroje)
  - nejrychleji
- Minimalizovat přepracování
  - zadání, re-use
- Snížit rizika plynoucí z neznámého
  - funkčnost, technologie

- Případně vůbec vejít se do omezujících podmínek
  - peníze vs čas vs funkčnost vs kvalita

Everyone

Everyone

Základní pojmy

Životní cyklus, metodika

- **ŽC** = proces od zahájení vývoje až po vyřazení z provozu

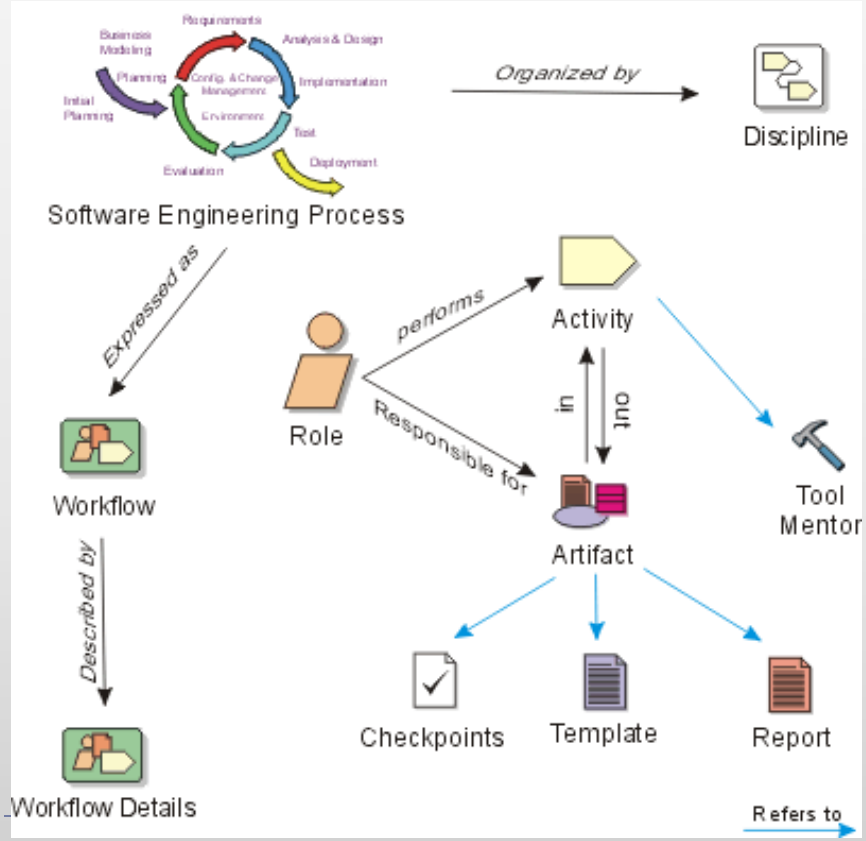
- **Metodika** = definovaný proces pro konkrétní účel, tj. fáze, aktivity, role, artefakty, milníky atd. jsou dobře popsány
  - Booch method
  - SSADM
  - Rational Unified Process
  - SCRUM
  - …
  - UML není metodika!

Softwarový proces

- Proces: *systematická série akcí vedoucí k určitému výsledku*

[Random House Unabridged Dictionary, 2006]

- Softwarový
  - výsledek = kvalitní software
  - členění: fáze, **aktivity**
  - mezivýsledky: **artefakty**
  - činitelé: **role**
- Meta-proces, ŽC
  - varianty uspořádání aktivit, produktů



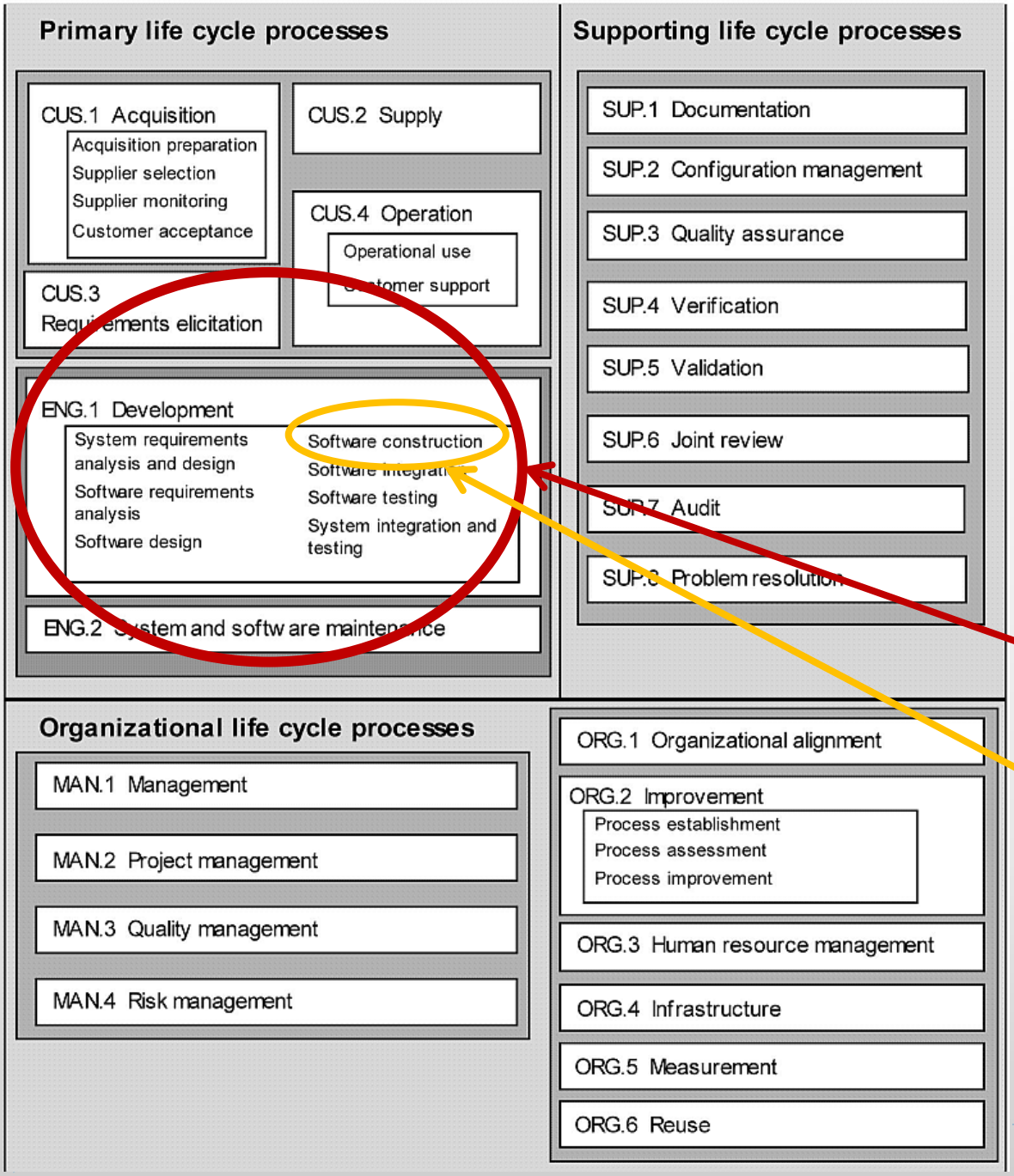
SW Proces: Typické aktivity

Technické

- Komunikace
- Plánování
- Modelování
- Konstrukce
- Nasazení

Podpůrné

- Řízení
- Kontrola kvality
- Správa konfigurace
- Dokumentace



Co vše souvisí s životním cyklem softwarového produktu

Zdroj: Standard ISO15504

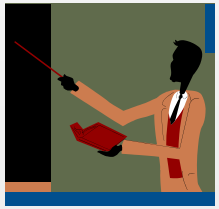
Pokud inženýr zná jenom tohle, asi to nebude úplně ono.

Pokud zná jenom tohle a jenom trochu, asi je něco úplně špatně…

SW Proces: Role, tj. lidi v procesu

- Technické
  - analytik (konzultant)
  - architekt, návrhář
  - vývojář
  - „buildovač“ a správce konfigurace
  - tester
  - databázista

- Manažerské
  - team leader
  - technický vedoucí projektu
  - šéf vývojářů
  - šéf projektů
  - CEO (příp. CIO)



- Podpůrné
  - poradce, kouč
  - lektor
  - uživ. podpora
  - dokumentace

SW Proces: Artefakty a jejich význam

- Technické
  - specifikace
  - dokumentace
  - kód, data
  - testy
  - modely
- Komunikační
  - specifikace
  - plán
- Obchodní
  - plán
  - rozpočet
  - produkt

- Účel
- Popis - dokumentace
- Kontrakt

- Vlastnictví
- Výsledek/vstup aktivity
  - definice – proces
  - podpora – nástroje

Varianty softwarového procesu

Společná snaha = snížení rizika chaotického postupu

Nulová varianta

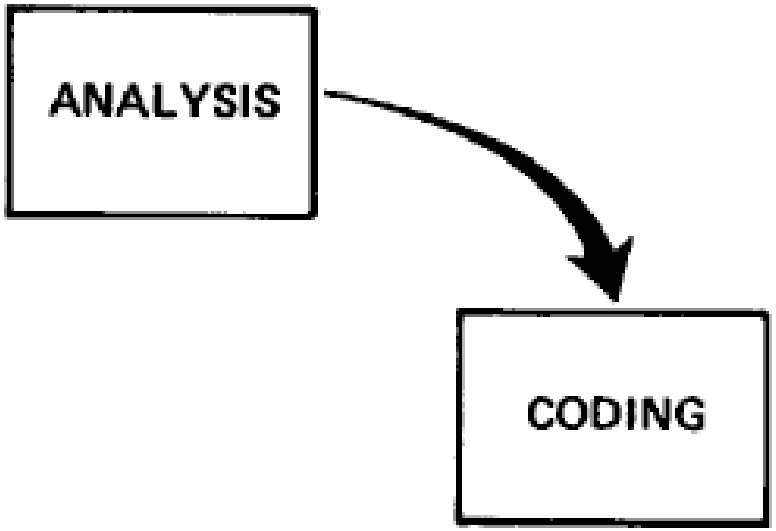


Figure 1. Implementation steps to deliver a small computer program for internal operations.

Sekvenční postup

- Hlavní technické aktivity lineárně po sobě
  - vztažené na celý produkt → „velký třesk“
  - naplánované pro celý projekt
  - oddělené meziprodukty
- Sledování plánu („hra na jistotu“)
  - kontext neměnný
  - zadání a technologie zřejmé, (nebo rozsah malý)

Vodopádový model

Winston Royce: Managing the Development of Large Software Systems. Proceedings of IEEE WESCON, 1970.

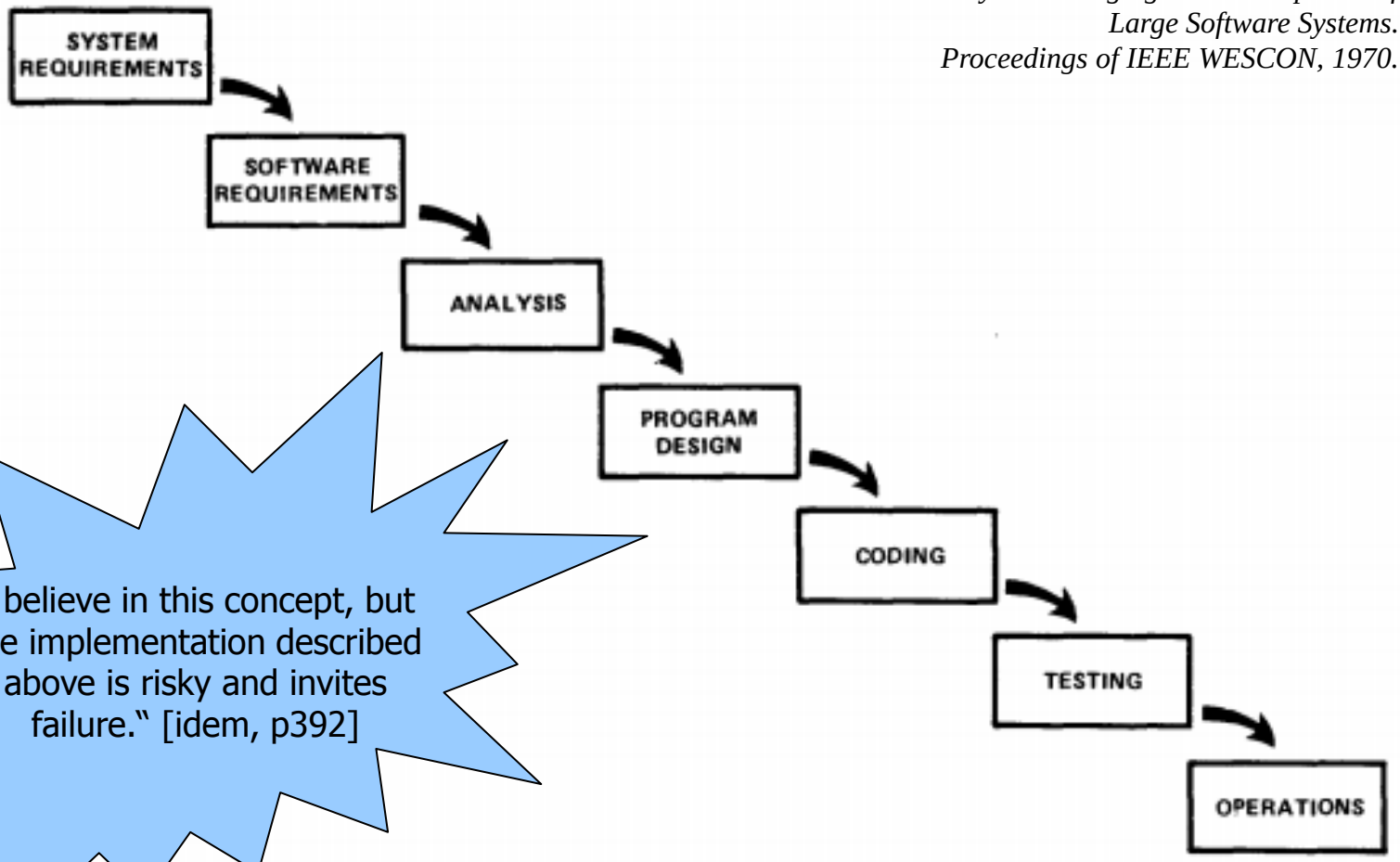
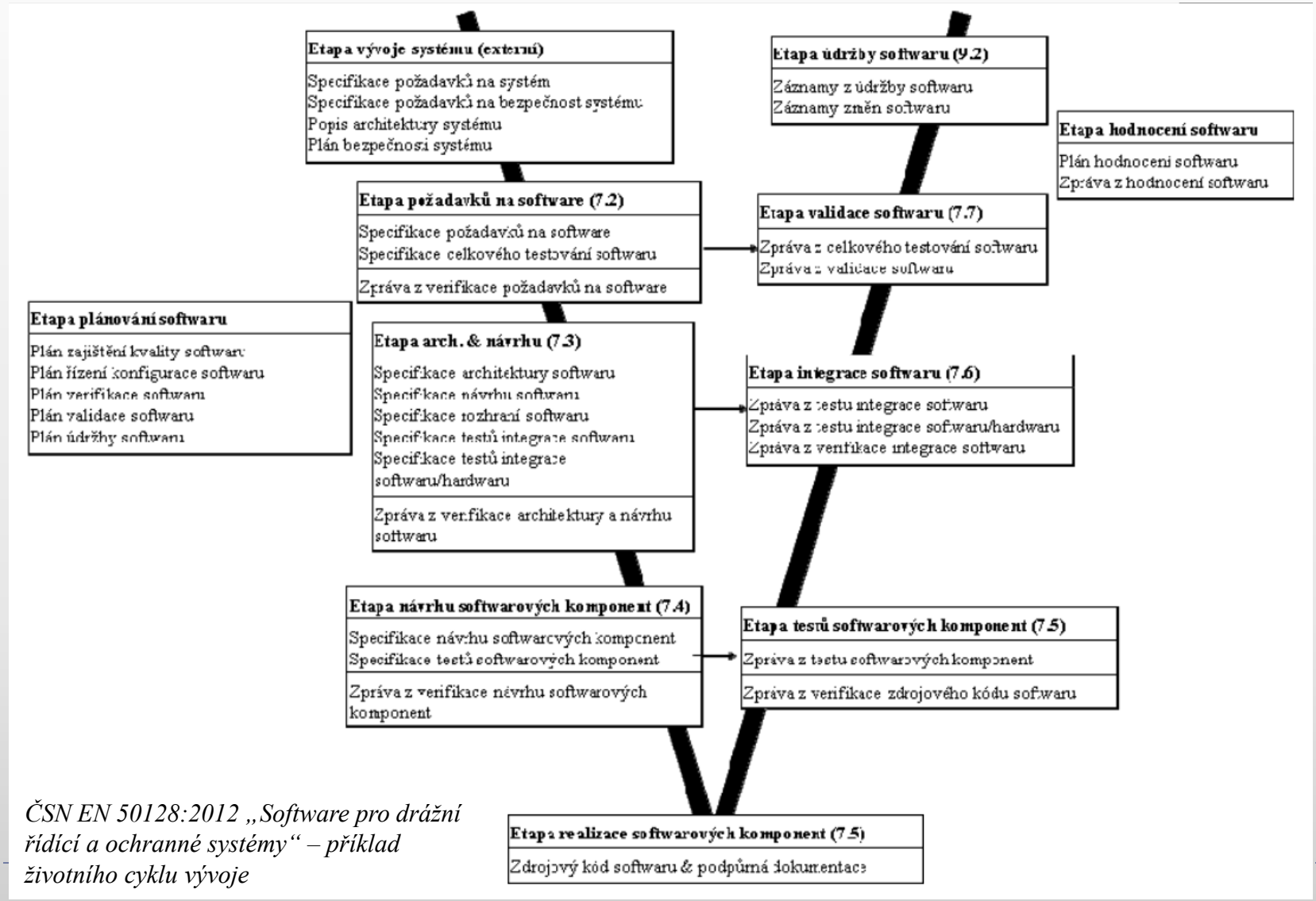


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

„I believe in this concept, but the implementation described above is risky and invites failure.“ [idem, p392]

V-model



ČSN EN 50128:2012 „Software pro drážní řídící a ochranné systémy“ – příklad životního cyklu vývoje

Cyklický postup

- Opakování technických aktivit
  - obsah podle sekvenční fáze, znalosti detailů
- Produkt postupně „roste“
  - znalost, funkcionalita, kvalita, …
- Omezování rizika
  - kontext zřejmý
  - zadání a/nebo technologie nejasné
- Model „průzkumník“

Spirálový model

Boehm B, "A Spiral Model of Software Development and Enhancement", IEEE Computer, 21(5):61-72, May 1988

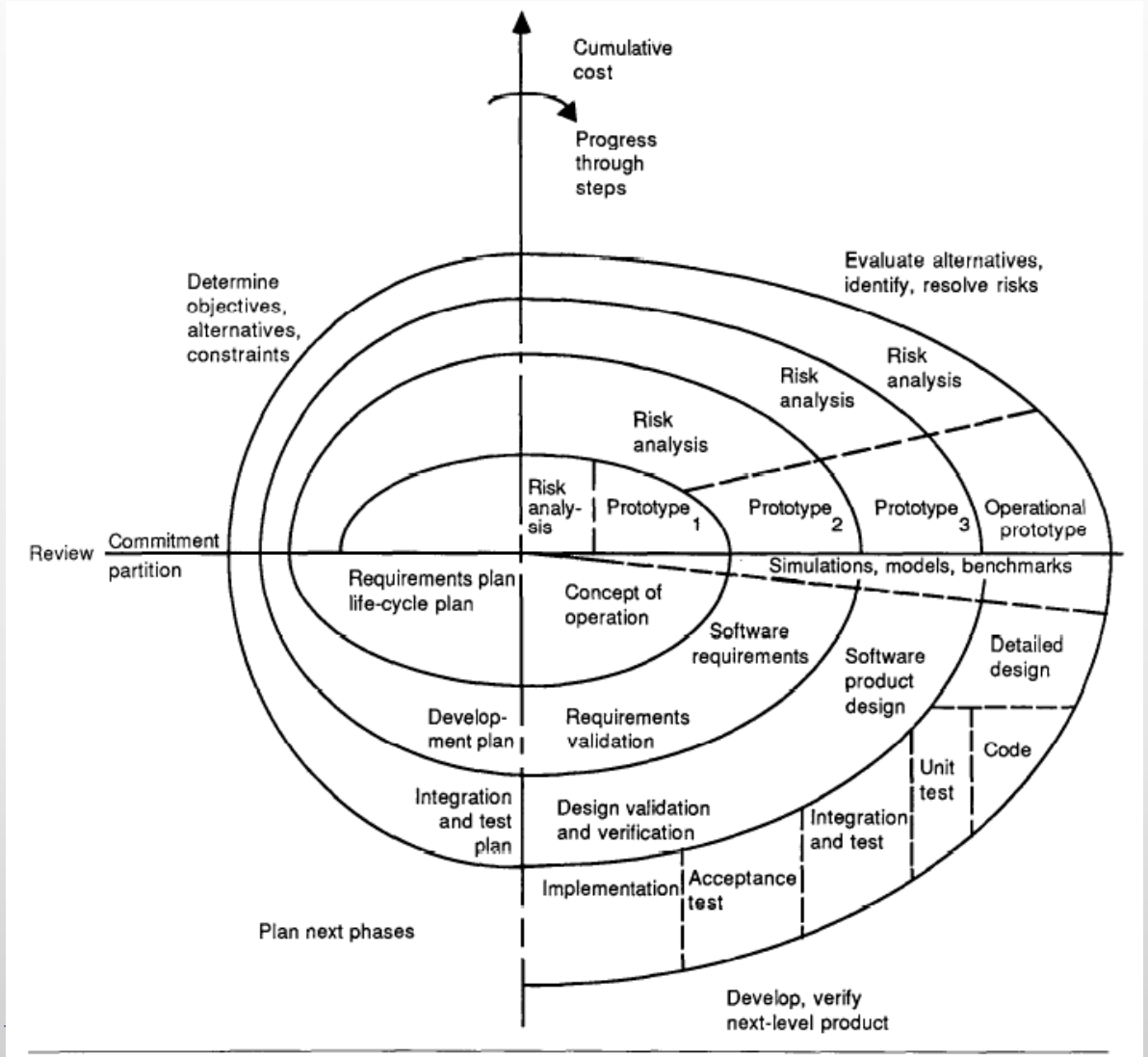
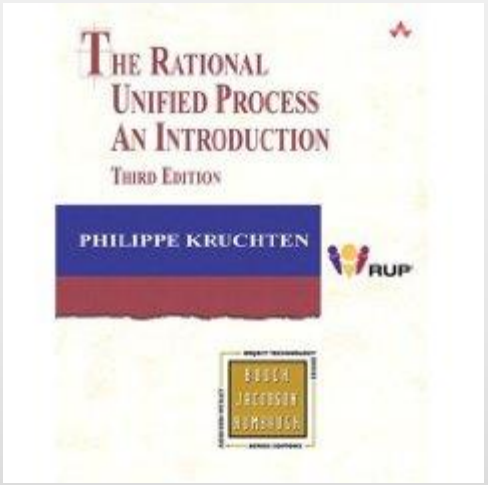
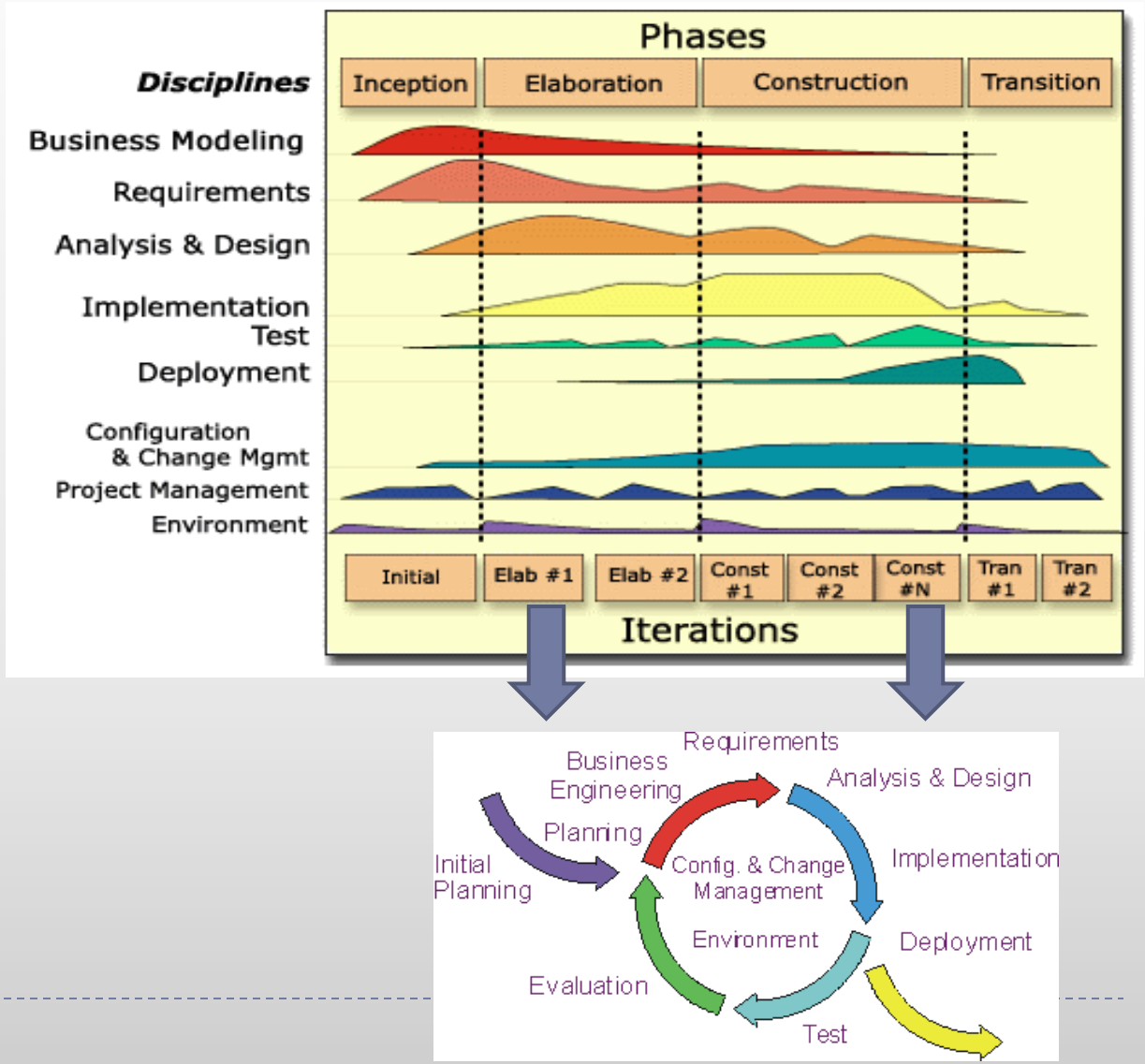


Figure 2. Spiral model of the software process.

(Rational) Unified Process



Kruchten, P. The Rational Unified Process: An Introduction. Addison-Wesley 2003



Agilní přístup

- Cyklický postup
  - krátké iterace
  - důraz na technické disciplíny a adaptaci „za pochodu“
- Adaptace na změnu
  - kontext proměnlivý
  - zadání a/nebo technologie nejasné

Agilní manifest, Lean principy

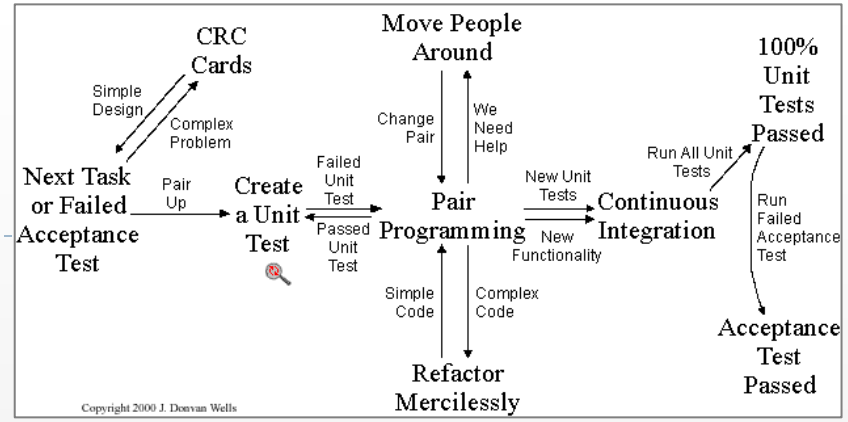
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

The Seven Principles of Lean Thinking:

- Eliminate Waste
- Amplify Learning
- Decide as Late as Possible
- Deliver as Fast as Possible
- Empower the Team
- Build Integrity In
- See the Whole

Scrum



Pre-game

DAILY SCRUM MEETING

24 HOURS

PRODUCT BACKLOG

SPRINT BACKLOG

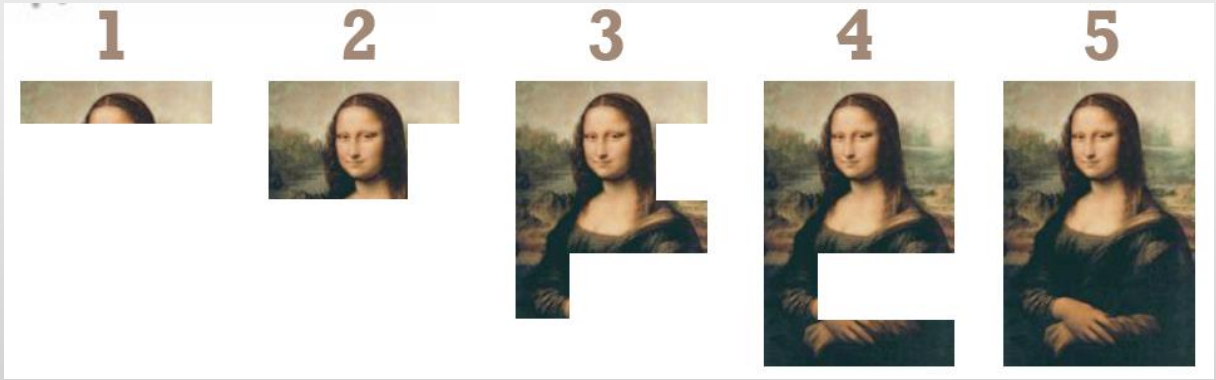
2-4 WEEKS

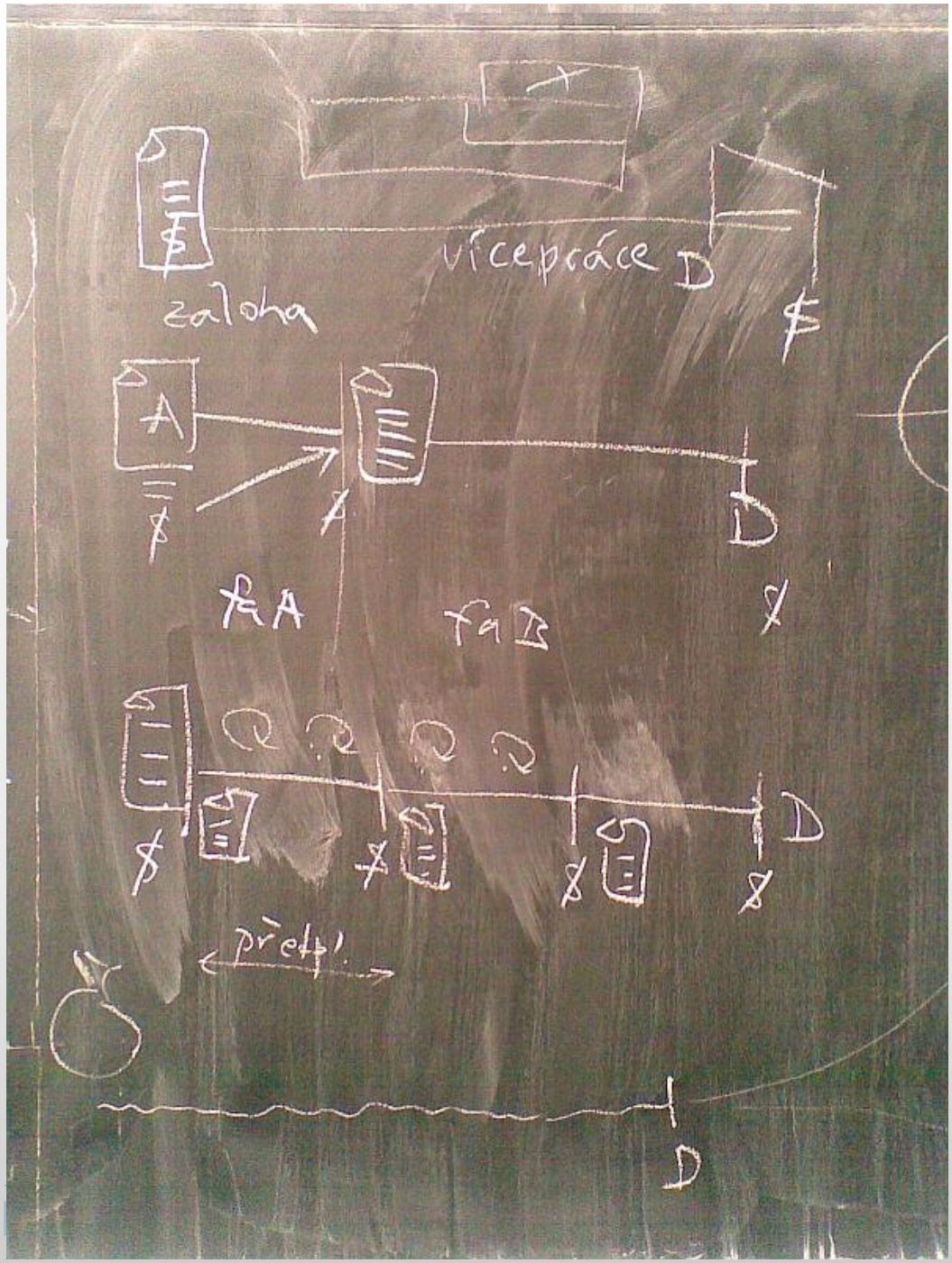
POTENTIALLY SHIPPABLE PRODUCT INCREMENT

Post-game

Alternativy dodávek funkčnosti

- Velký třesk
  - malé projekty, jasné požadavky
- Přírůstkově
  - určení přírůstků →plán → postupné dodávky
  - zpětná vazba, ale úpravy projektu obtížné
- Evolučně
  - cyklus: určení cíle → dodávka → zpřesnění („growing sw“)



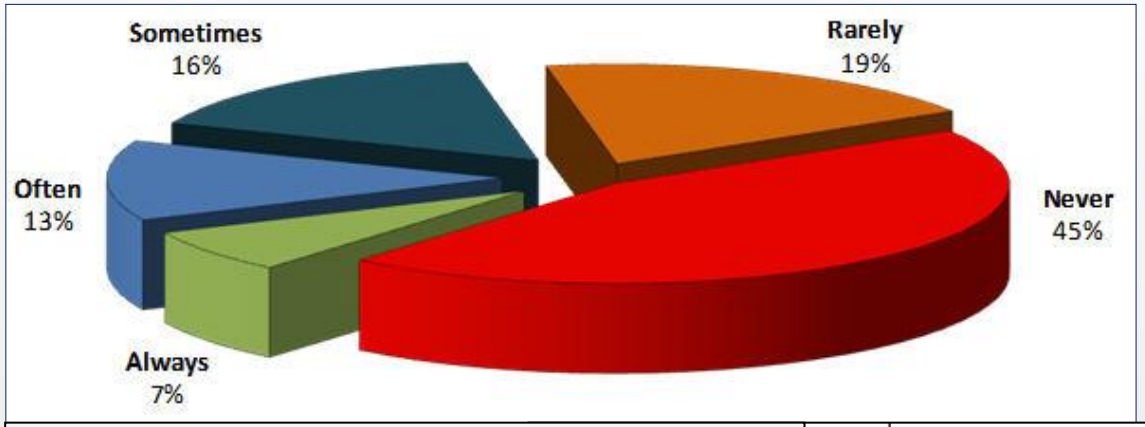


Vazba na obchodní model a smluvní podmínky

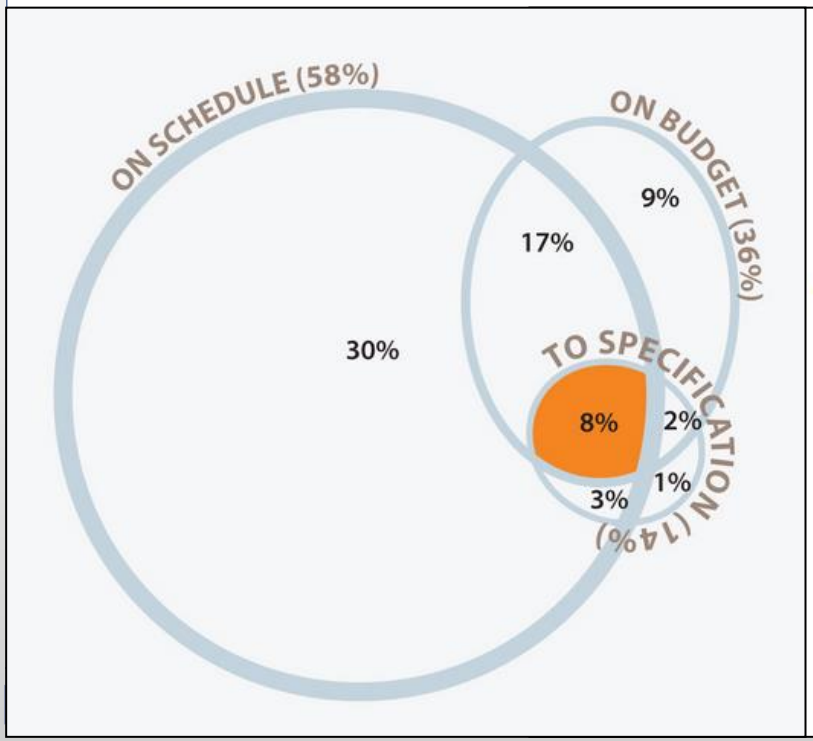
Různé varianty dle zvyklostí zákazníka a prostředí

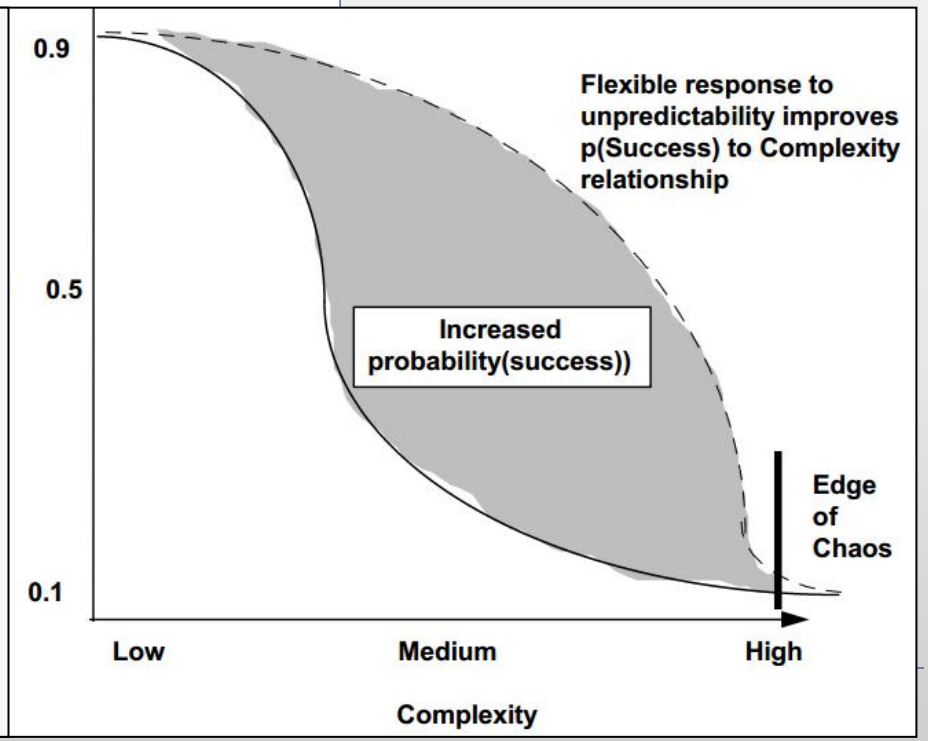
Jaký zvolit postup vývoje?

Driving Forces



I believe in this concept, but the implementation described above is risky and invites failure.

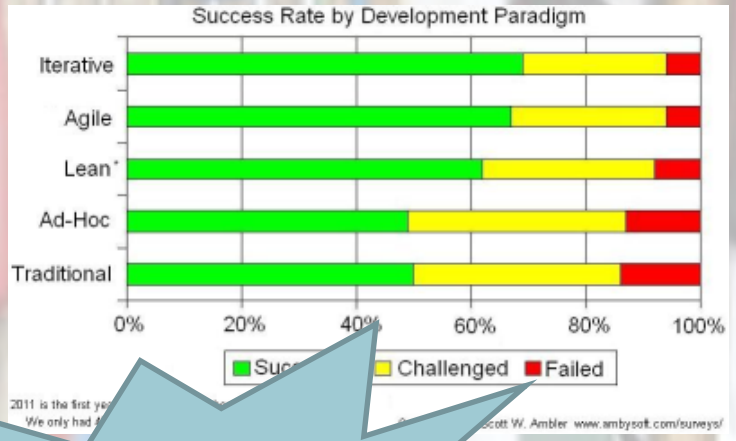





One size does not fit all

… vhodný pro daný účel!

- Typ projektu
- Velikost problému
- Složitost problému
- Charakter týmu
- …



There is no silver bullet.

ASWI: iterativní

SSADM Evo
RUP Lean
Yourdon MSF
Scrum MIL-
STD-498
Kanban XP
FDD Prince2
DSDM CMMI
OpenUP EUP
Cleanroom
BDUF SPICE
…

Shrnutí

Softwarový proces

- Systematická série **aktivit** + souvisejících **rolí** a **artefaktů** vedoucí k vyšší pravděpodobnosti úspěšného vytvoření potřebného software
  - sekvenční → velký třesk
  - cyklický → přírůstky / evoluce
  - agilní → evoluce

Iterativní vývoj software

KIV/ASWI 2014/2015

Obsah

- Iterativní vývoj
  - struktura a vlastnosti iterace
  - globální řízení
- Empirický proces

Q: Jaké můžeme v nejbližší době čekat nové, vzrušující a slibné myšlenky nebo techniky v oblasti software?

A: Myslím, že [nejslibnější myšlenky] jsou už léta známy, jen nejsou správně používány.

– David Parnas

- Kde jsou kořeny iterativního přístupu?

Although extreme programming itself is relatively new, many of its practices have been around for some time; the methodology, after all, takes "best practices" to extreme levels. For example, the "practice of test-first development, planning and writing tests before each micro-increment" was used as early as NASA's Project Mercury, in the early 1960s (Larman 2003). To shorten the total development time, some formal test documents (such as for acceptance testing) have been developed in parallel (or shortly before) the software is ready for testing.

http://en.wikipedia.org/wiki/Extreme_programming

http://arialdomartini.wordpress.com/2012/07/20/you-wont-believe-how-old-tdd-is/

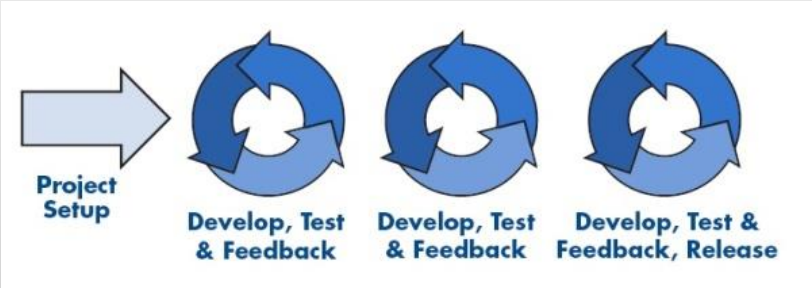
We didn't call those things by those names back then, but if you look at my first book (Computer Programming Fundamentals, Leeds & Weinberg, first edition 1961 —MB) and many others since, you'll see that was always the way we thought was the only logical way to do things. I learned it from Bernie Dimsdale, who learned it from von Neumann.

When I started in computing, I had nobody to teach me programming, so I read the manuals and taught myself. I thought I was pretty good, then I ran into Bernie (in 1957), who showed me how the really smart people did things. My ego was a bit shocked at first, but then I figured out that if von Neumann did things this way, I should.

Jak funguje iterativní vývoj

- *„Když sekvenční postup funguje pro malé projekty s malou mírou neznáma, proč nerozbít velký projekt do řady malých?“* – *P. Kruchten*

- Miniaturní úplný projekt
  - cca vodopádový model
  - prolínání aktivit
- Cíl: iterační release (interní či nasazený)
  - produkt funkčně neúplný
  - ale otestovaný a funkční
- Opakovaný postup
  - stále stejné aktivity (téměř)

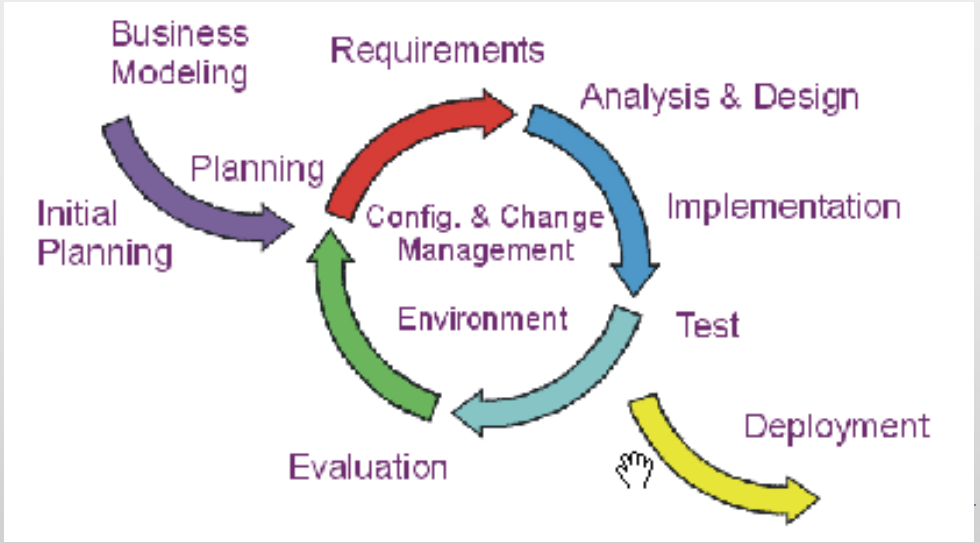


Průběh iterace

- Plánování cíle iterace
  - zejména funkčnost
- Doplnění / zpřesnění požadavků
  - základ: plán projektu, vize, předchozí feedback
- (Úprava návrhu)
- Implementace přírůstku funkčností
- Integrace přírůstku
  - ověření, otestování
- (Předání do provozu)
  - validace zákazníkem
- Zhodnocení

Simple way:
per-iteration-goal

Better way:
per-workitem



Charakter iterativního vývoje

Vize produktu …

… a její iterativní naplňování

cf přírůstková dodávka

Příklad metodiky: Scrum

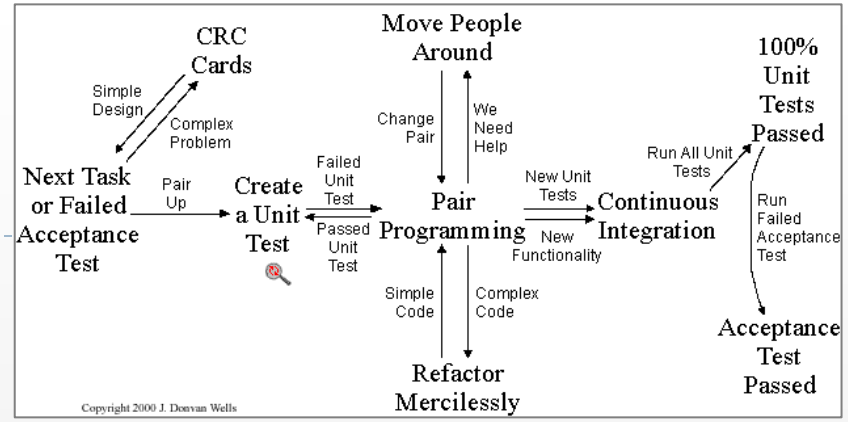
Schwaber: SCRUM Development Process

One can argue that current methodologies are better than nothing. Each improves on the other. The Spiral and Iterative approaches implant formal risk control mechanisms for dealing with unpredictable results. A framework for development is provided.

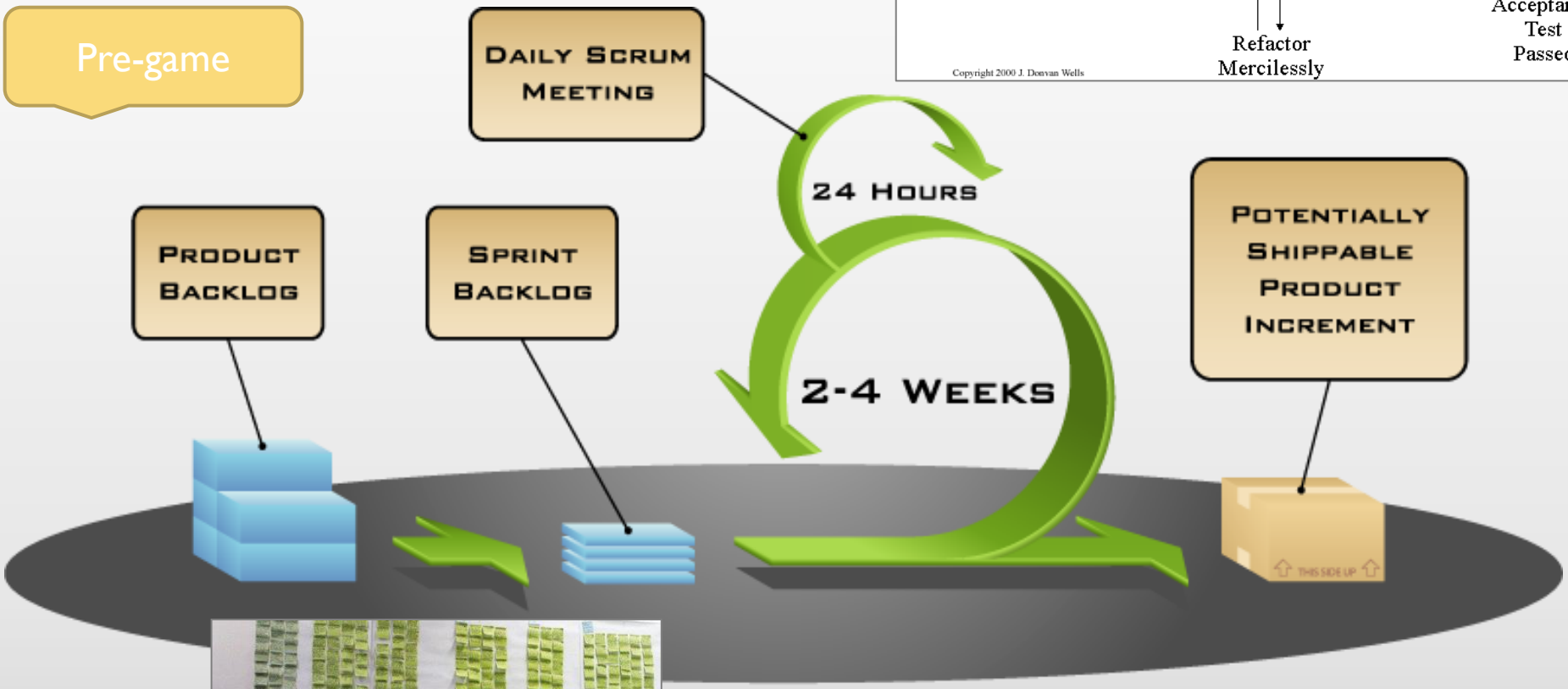
However, each rests on the fallacy that the development processes are defined, predictable processes. But unpredictable results occur throughout the projects. The rigor implied in the development processes stifles the flexibility needed to cope with the unpredictable results and respond to a complex environment.

Sutherland, Jeffrey Victor; Schwaber, Ken (1995). Business object design and implementation: OOPSLA '95 workshop proceedings. The University of Michigan. p. 118. ISBN 3-540-76096-2.

Scrum



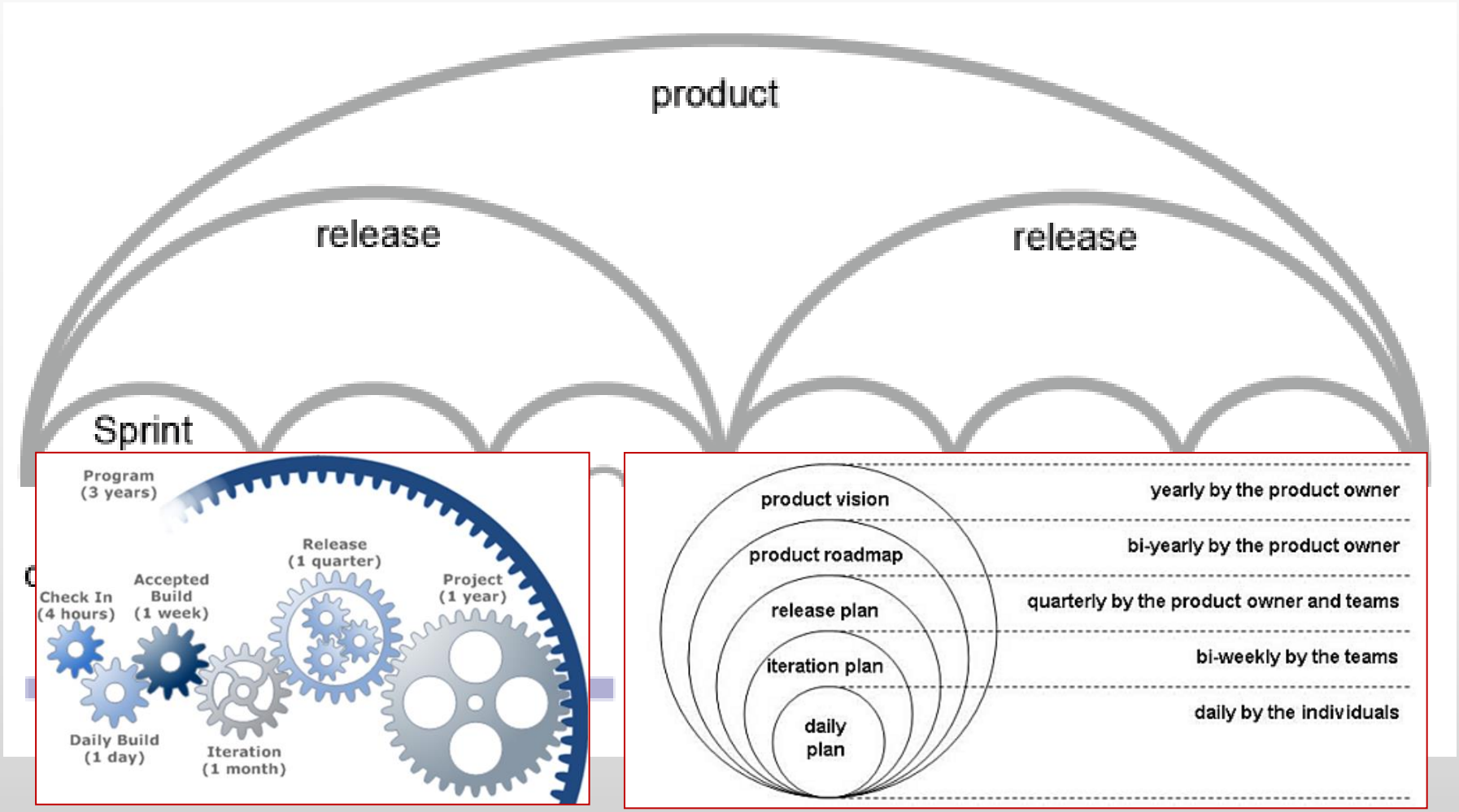
Pre-game



Post-game

Iterace v kontextu

Kontext iterace v procesu vývoje



Globální řízení iterativního vývoje

Problém: pro stromy nevidím les

Jak vysvětleno zákazníkem

I-1 I-2 I-3 I-4 I-5

Co navrhl architekt

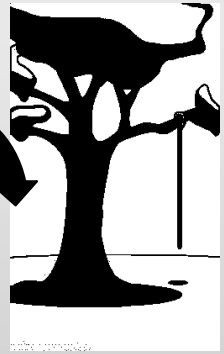
I-2

I-3

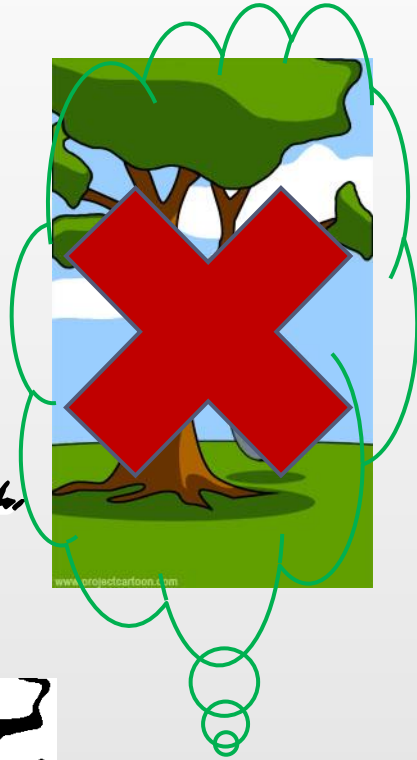
Implementace

I-4

I-5



Dodáno na konci



http://www.projectcartoon.com/cartoon/3

Globální plánování: milníky

- Cíl: eliminovat momentálně největší riziko
- Barry Boehm (1996): *Anchoring the Software Process*

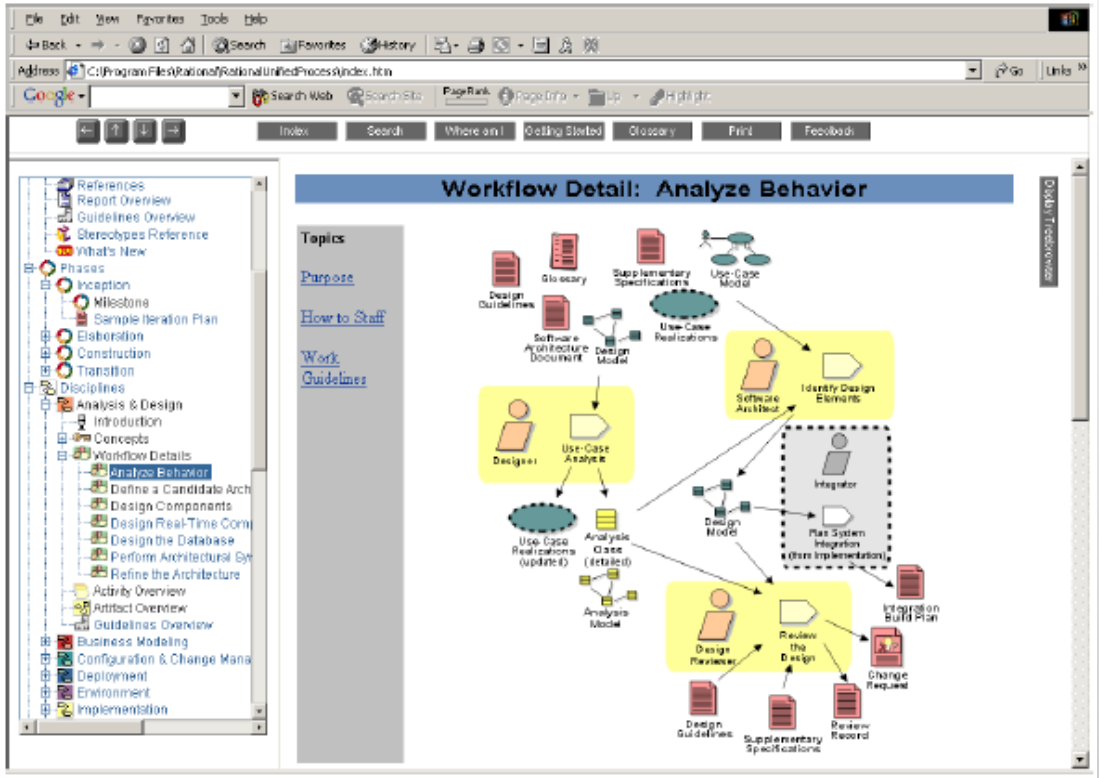
- **LCO** (Lifecycle Objectives)
  - definování terče – Vize produktu
- **LCA** (Lifecycle Architecture)
  - určení způsobu řešení – Architektura technického řešení
  - ověření – modely, technické prototypy, testy (executable)
- **IOC** (Initial Operational Capability)
  - schopnost efektivně „vyrobit“ řešení – beta verze, all features
  - unit a funkční testy
- **GA** (General Availability)
  - uvést produkt do rutinního provozu – „krabice“ s produktem, website launch, raut :-)
  - support team v provozu

Příklad metodiky: Rational Unified Process

Rational Unified Process Best Practices for Software Development Teams

The Rational Unified Process captures many of the ***best practices*** in modern software development in a form that is suitable for a wide range of projects and organizations. Deploying these best practices using the Rational Unified Process as your guide offers development teams a number of key advantages. In next section, we describe the six fundamental best practices of the Rational Unified Process.

1. Develop software iteratively
2. Manage requirements
3. Use component-based architectures
4. Visually model software
5. Verify software quality
6. Control changes to software

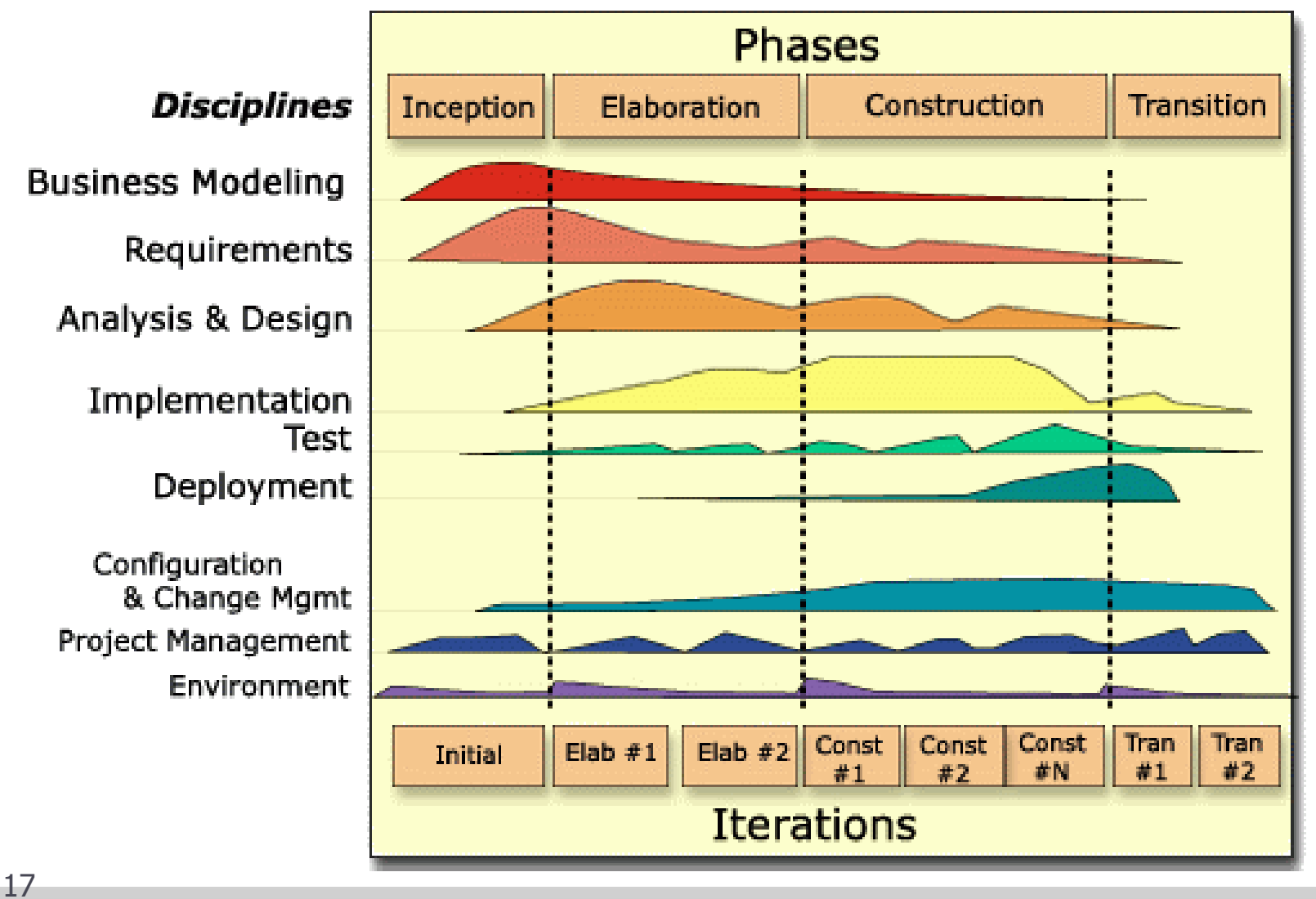


Rational Software White Paper
TP026B, Rev 11/01

RUP


Rational Unified Process
RUP
Proven. Practical. Flexible
BUSINESS MODELING
PHASES
ITERATIONS
IMPLEMENTATION
Analysis & Design
Implementation
Testing
Management
Environment
Configuration Management
Business Modeling
Planning
Evaluation
Initial Planning
Deployment
TEST
REQUIREMENTS
PROJECT MANAGEMENT
CONFIGURATION & CHANGE MANAGEMENT
ANALYSIS & DESIGN
ENVIRONMENT
DEPLOYMENT

RUP


Phases
Disciplines
Inception
Elaboration
Construction
Transition
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration
& Change Mgmt
Project Management
Environment
Initial
Elab #1
Elab #2
Const #1
Const #2
Const #N
Tran #1
Tran #2
Iterations

Charakteristiky iterativního vývoje

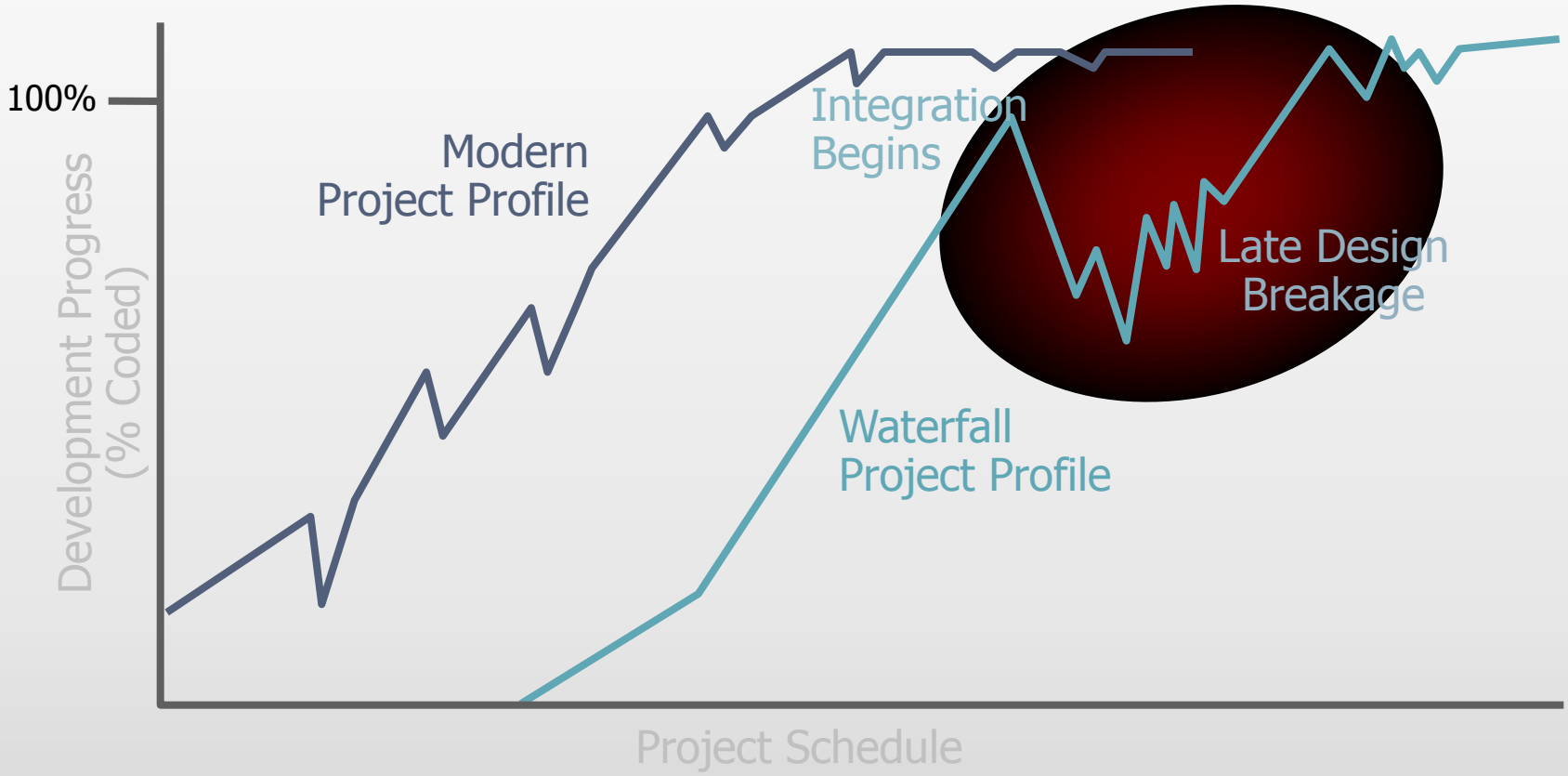
Evoluční a adaptivní charakter

… jeden z 4 nejčastějších faktorů úspěchu sw projektů

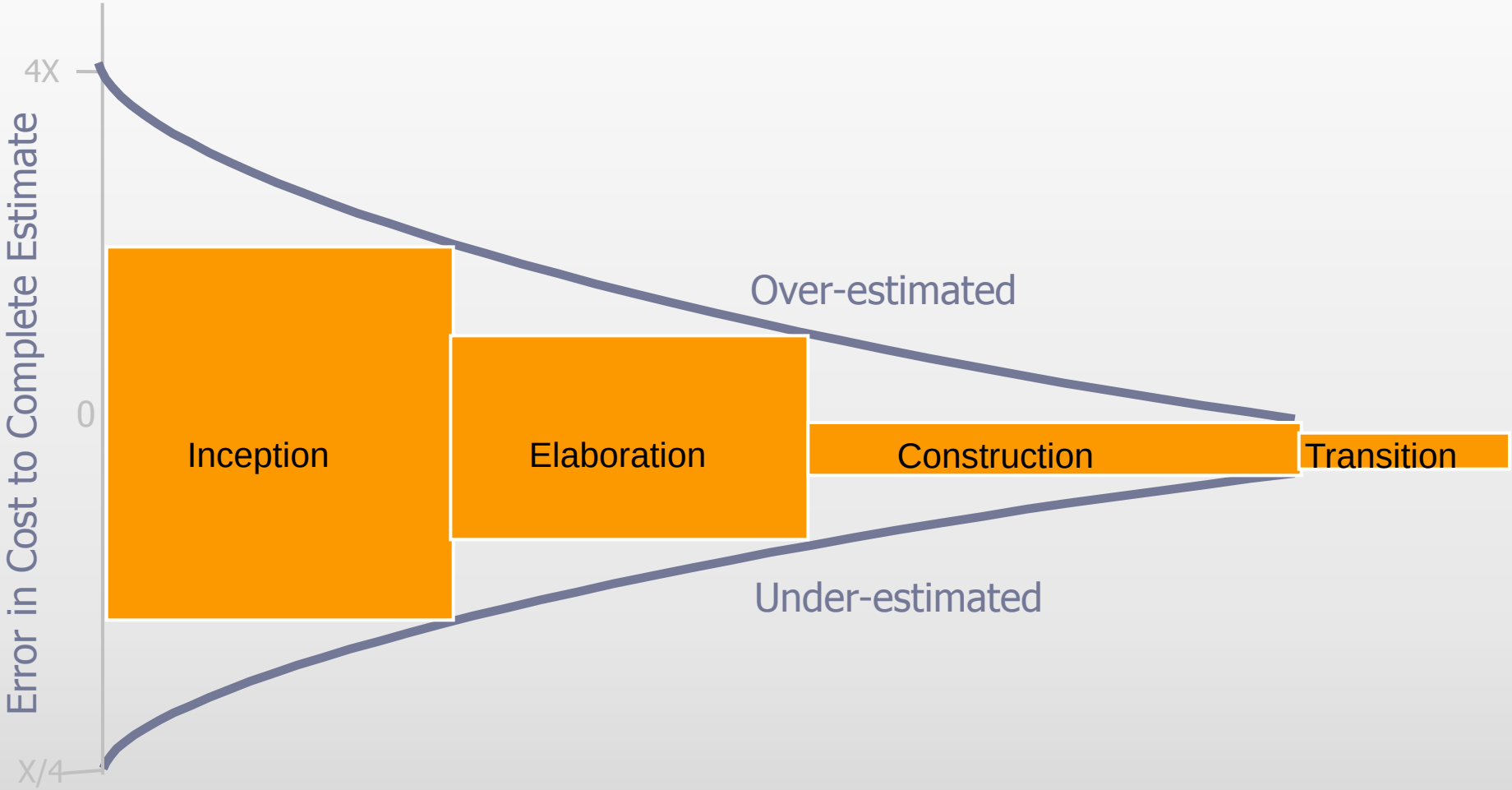
- Evoluční
  - znalosti o požadavcích, návrhu, odhadech a plánu se vyvíjejí a zpřesňují v průběhu projektu
    - vs kompletní, dále neměnné specifikace na začátku (20-80)
    - míra změny obvykle klesá s postupujícími iteracemi
  - „don't develop software, grow it"
- Adaptivní
  - zdůraznění procesu učení
  - zpětná vazba od uživatelů
  - empirický proces

19

Better Progress Profile

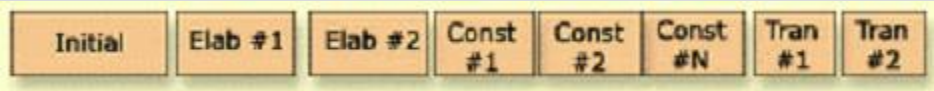


Cost Estimate Fidelity



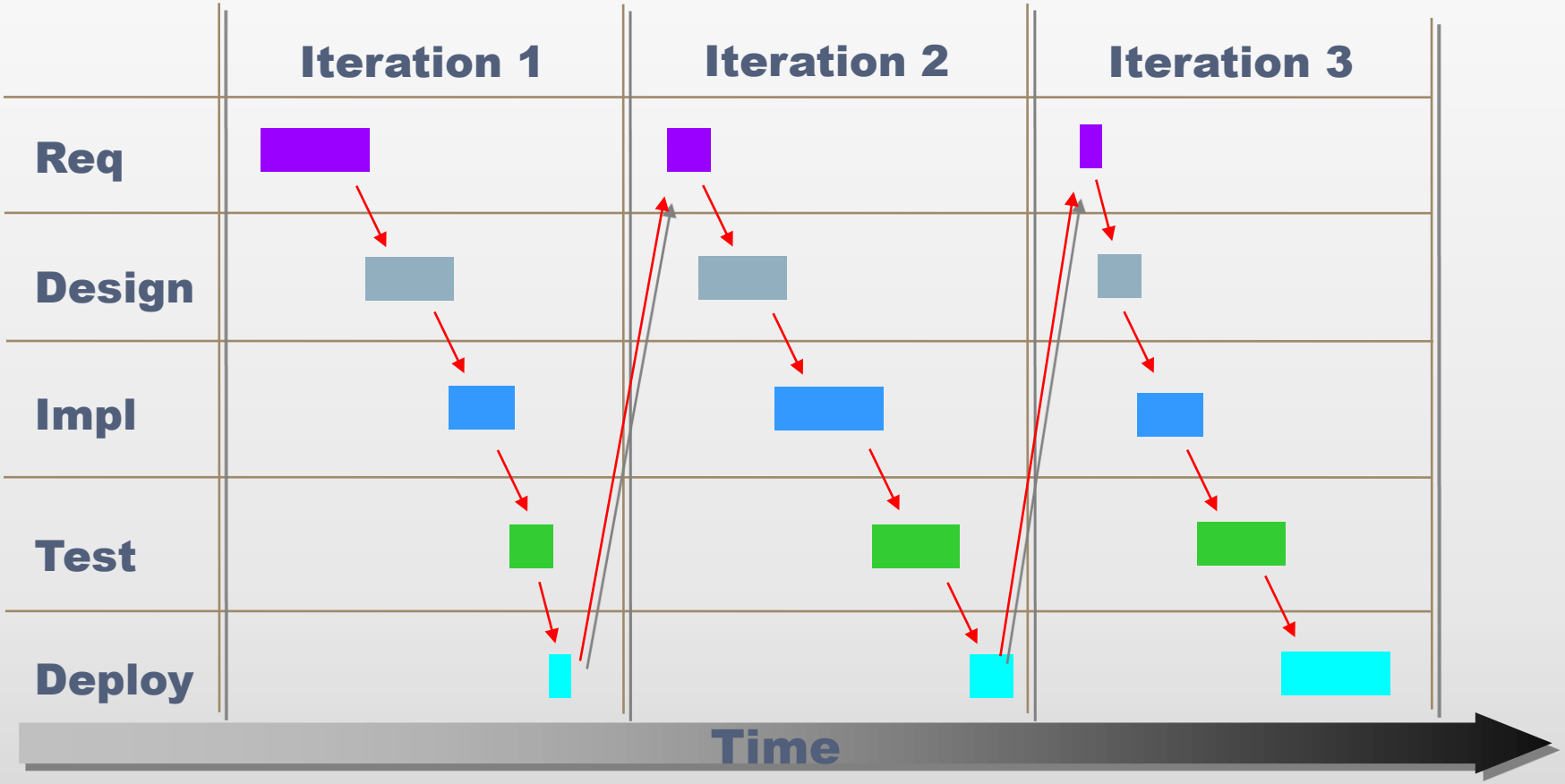
Charakter iterací dle fáze

- Základní schema pevné, mění se činnosti a artefakty
- **Zahájení** – analytické činnosti, validace vize zákazníkem
  - 1-2 iterace
- **Projektování** – analytické a designérské činnosti, ověřování prototypy, implementace
  - 2+ iterací
- **Konstrukce** – designérské a programátorské činnosti, změnové řízení, testování a ověřování
  - N iterací
- **Nasazení** – integrační a konzultační činnosti, ověřování provozem, náběh uživatelské podpory
  - 1-2 iterace



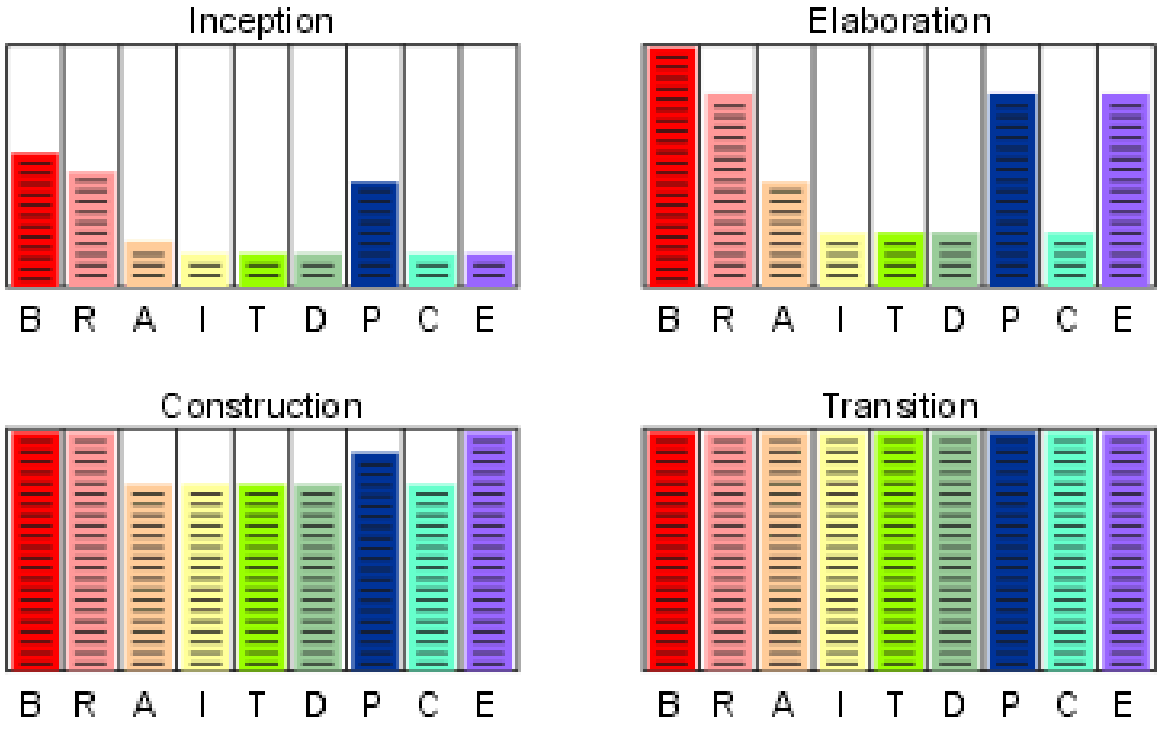
Example: RUP project profile: overview

Charakter iterací dle fáze: činnosti



Example: RUP project profile: details

Charakter iterací dle fáze: artefakty

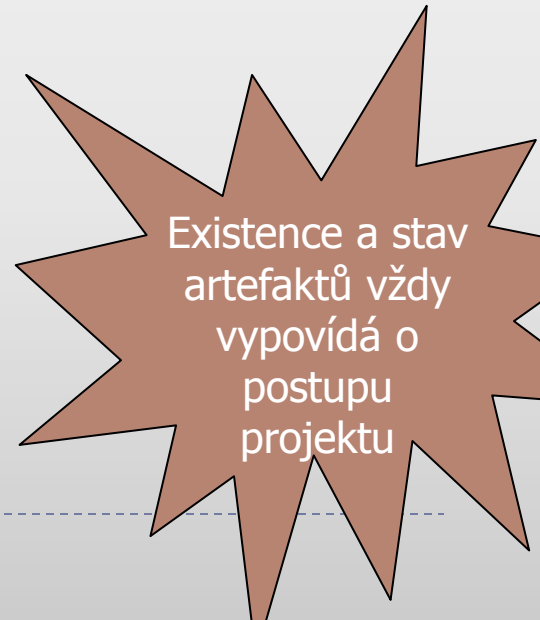


Information set evolution over the development phases.

Význam meziproduktů

- Preskriptivní metodiky
  - artefakty jsou cílem (výsledkem) fáze procesu
  - důsledek: review → podpis → změnové řízení

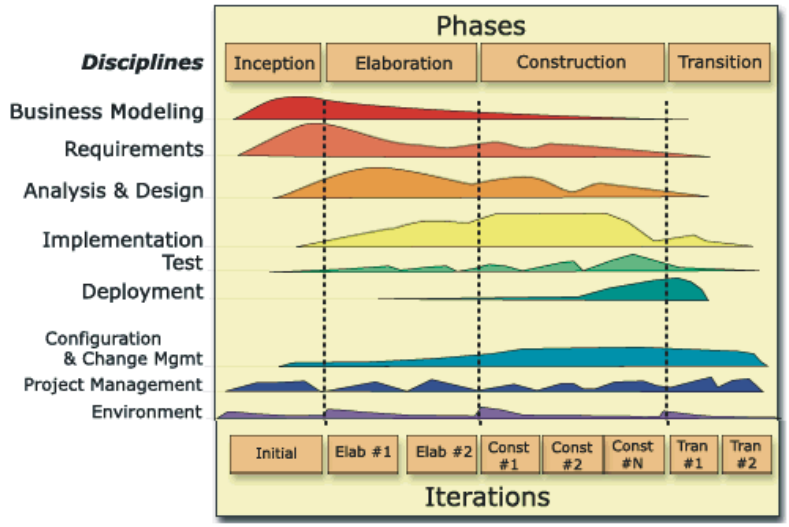
- Agilní přístup
  - artefakty jsou prostředkem
    (cíl = smysluplný stav/přírůstek produktu)
  - důsledky
    - forma, obsah artefaktů („dress code“):
      od zcela volné (XP) po vzory a šablony (RUP)
    - artefakty živé během projektu
    - výběr dle fáze/iterace



Shrnutí

Iterativní vývoj

- Risk and user-priority driven
- Process focus on architecture



… but waterfall is not dead

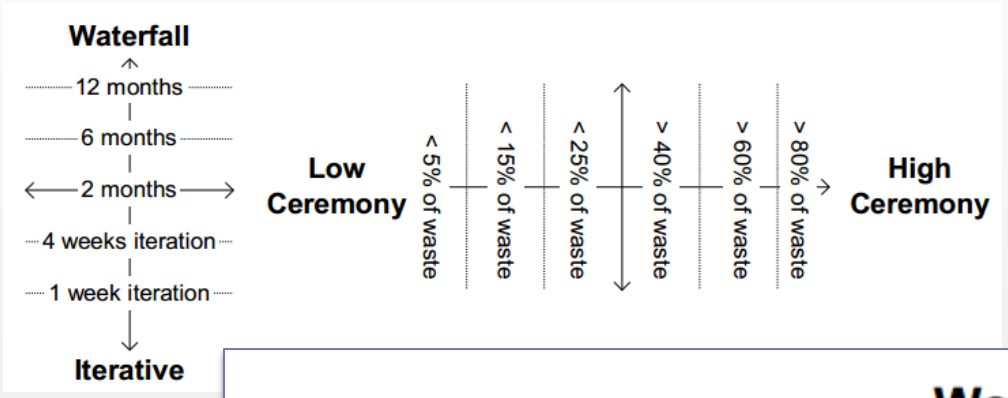
- Requirements drive design and implementation
- Models abstract the system
- Guidance for activities and artifacts

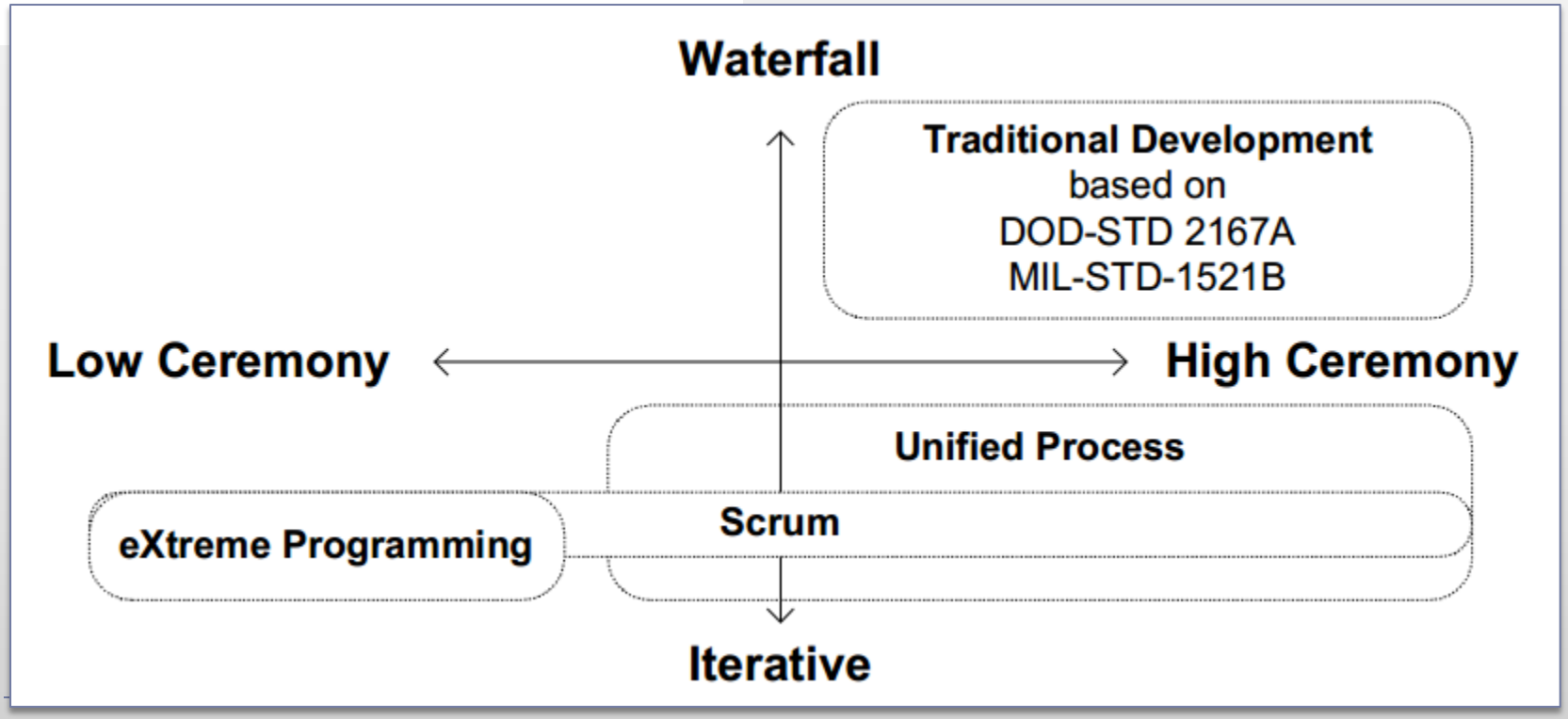
Research at The Standish Group also indicates that smaller time frames, with delivery of software components early and often, will increase the success rate. Shorter time frames result in an iterative process of design, prototype, develop, test, and deploy small elements. This process is known as "growing" software, as opposed to the old concept of "developing" software. Growing software engages the user earlier, each component has an owner or a small set of owners, and expectations are realistically set. In addition, each software component has a clear and precise statement and set of objectives to be less complex. Making the projects simpler causes only confusion and increased cost.

THE STANDISH GROUP REPORT

CHAOS

Varianty dle velikosti projektu





Zahájení projektu (fáze Inception)

KIV/ASWI 2014/2015

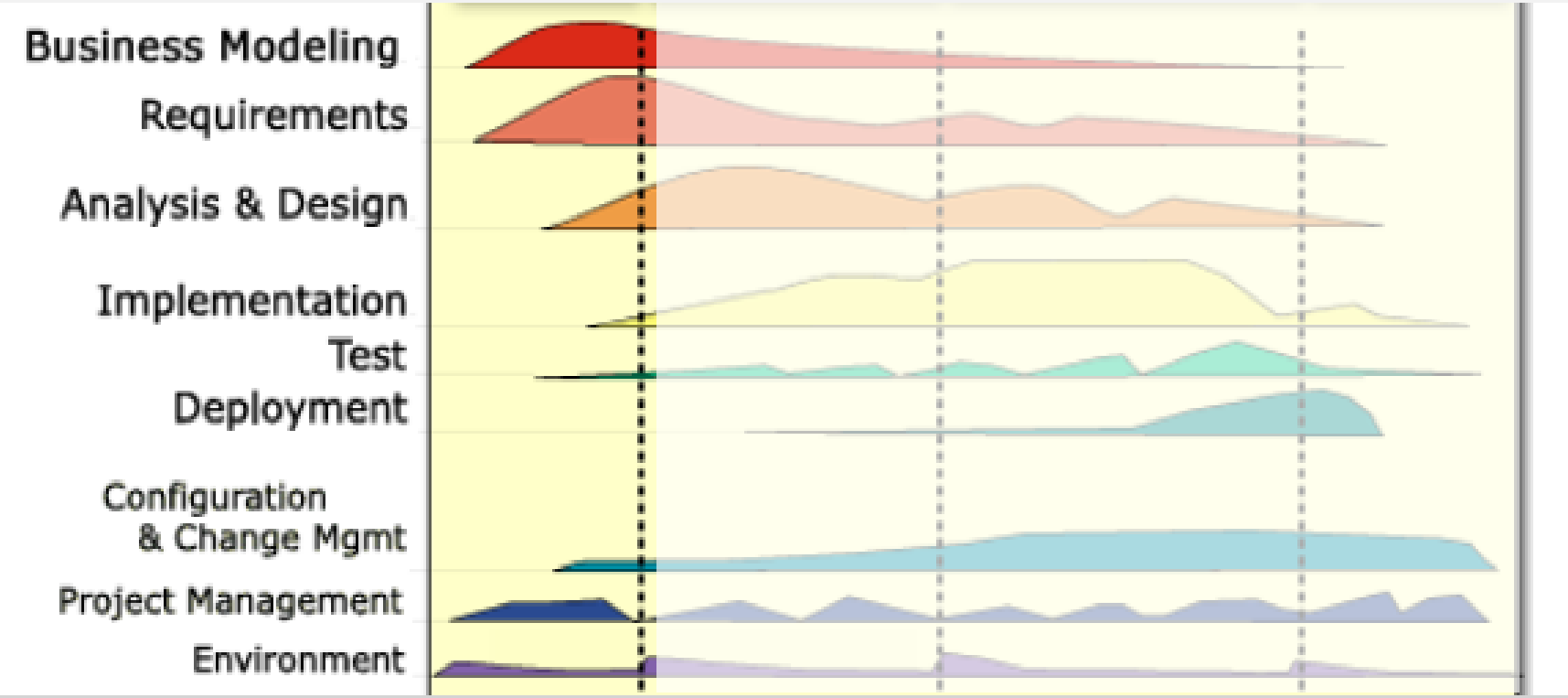
Přehled a cíle

- Nápad → poptávka → nabídka
  → zformování vize || kontrakt → zahájení projektu

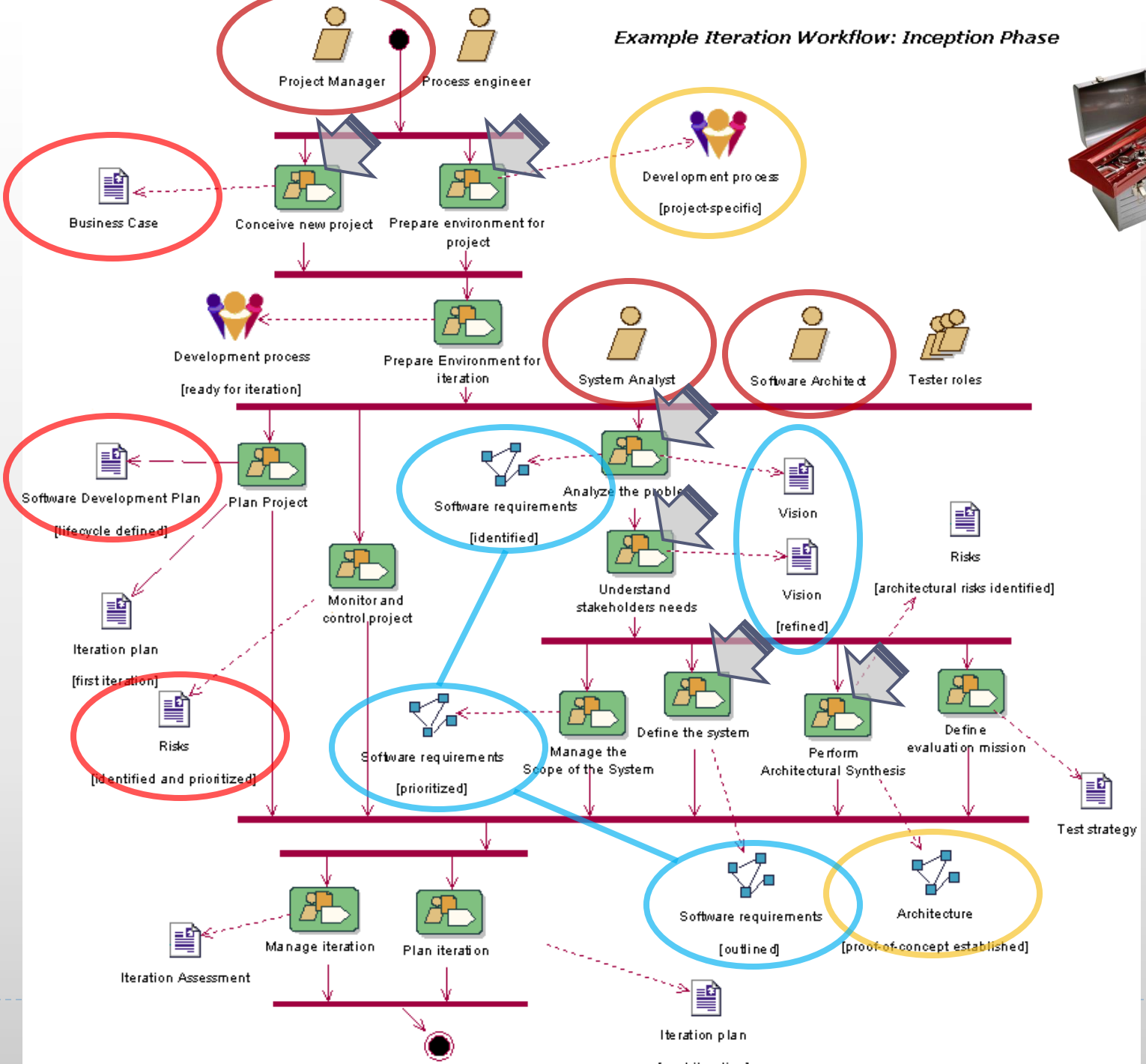
- Cíl fáze:
  „To achieve concurrence among all stakeholders on the lifecycle objectives for the project“
  - vize produktu – co se má vytvořit
  - business case – zdůvodnění, že se to vyplatí
  - technický koncept – ověření proveditelnosti

Klíčové disciplíny a aktivity

rozsah a hranice projektu / návrh ceny a harmonogramu / acceptance criteria / klíčové funkcionality (use cases) a omezující podmínky / koncept technické architektury / ověření proveditelnosti / určení rizik / příprava prostředí projektu



Example Iteration Workflow: Inception Phase



Milník = kdy jsou cíle fáze dosaženy

- LCO
  - srozumění s rozsahem, cenou, harmonogramem
  - viz Boehm: Anchoring the Software Process

- Artefakty
  - Vize produktu, Business case
  - Seznam rizik a strategie jejich řešení
  - Slovník pojmů a přehled klíčových požadavků
  - Koncept technického řešení (architektura + prototypy)
  - Plán projektu
  - Popis procesu a infrastruktury

Viz RUP Milestone: Lifecycle Objectives

Charakteristiky fáze

- Důležitá zejména pro nově zahajované projekty
  - pokračující / well-defined projekty: význam pro znovu-ověření předpokladů a odhadů
- Počáteční „chaos“ => důsledky pro plánování iterací
- Význam „měkkých“ disciplín
  - business modelování
  - komunikace

Dále podrobnosti o …

- Vize produktu
- Sběr a popis klíčových požadavků
- Plánování iterací

- Samostudium: koncept technického řešení
  - OpenUP: Task > Architecture > Envision the Architecture
  - Architectural Proof-of-Concept
    - viz též později v přednáškách