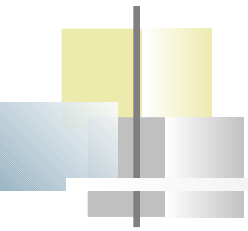
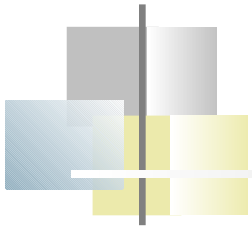


Softwarový proces
Iterativní vývoj software

KIV/ASWI 2008/2009



Vývoj software

… běžná aktivita v informační společnosti

- Na zakázku
  - komerční zákazník
  - státní sféra
- Interní projekt
- Krabicový software
- „Pro radost“

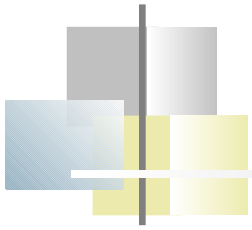
- Closed vs Open Source

Cíle zákazníka

- Typy zákazníků → důrazy
  - Telco, utility, banky
  - Výrobní sféra, NGO
  - Státní sektor

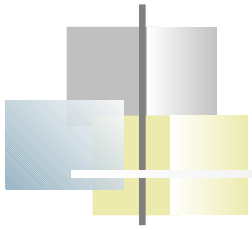
Cíle dodavatele

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



Cíle studenta

- Comments?



Životní cyklus, metodika

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

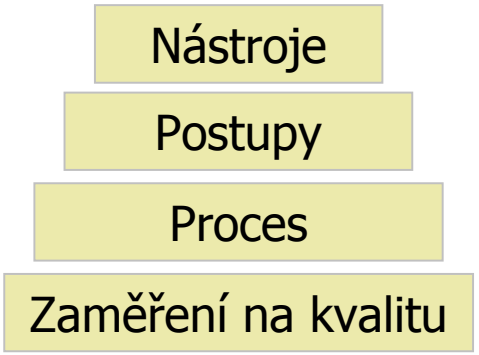
- **Metodika** = definovaný proces pro konkrétní účel, tj. fáze, aktivity, role, artefakty, milníky atd. jsou dobře popsány
  - » Booch method
  - » SSADM, Rational Unified Process, SCRUM
  - UML není metodika!

Softwarový proces

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

  [Random House Unabridged Dictionary, 2006]

- Softwarový
  - výsledek = kvalitní software
  - členění: fáze, aktivity; produkty
  - meta-proces, ŽC
    - » varianty uspořádání aktivit, produktů



Typické aktivity sw procesu

- **Technické**

- Komunikace
- Plánování
- Modelování
- Konstrukce
- Nasazení

- **Podpůrné**

- Řízení
- Kontrola kvality
- Správa konfigurace
- Dokumentace

Role lidí v procesu

- Technické
  - analytik (konzultant)
  - architekt, návrhář
  - vývojář
  - „buildovač" a správce konfigurace
  - tester
  - databázista
  - poradce, kouč
- Manažerské
  - team leader
  - technický vedoucí projektu
  - šéf vývojářů
  - šéf projektů
  - CEO (příp. CIO)
- Podpůrné
  - lektor
  - uživ. podpora
  - dokumentace

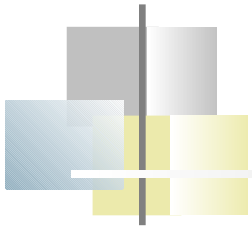
Artefakty a jejich role v procesu

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

- Účel
  - Popis - dokumentace
  - Kontrakt
- Vlastnictví
- Výsledek/vstup aktivity
  - » podpora – CASE

Varianty procesu

- Společná snaha = snížení rizika chaotického postupu
- Řízené plánem (jistotou)
  - » typicky sekvenční – vodopád, V-model
- Řízené riziky
  - » průzkumník/prototypování, spirála
- Řízené změnou
  - » iterativní, agilní



Sekvenční postup

- Hlavní technické aktivity lineárně po sobě
  - vztažené na celý produkt → „velký třesk“
  - naplánované pro celý projekt
  - oddělené meziprodukty
- Vodopádový model (v běžném podání)

Vodopádový model

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

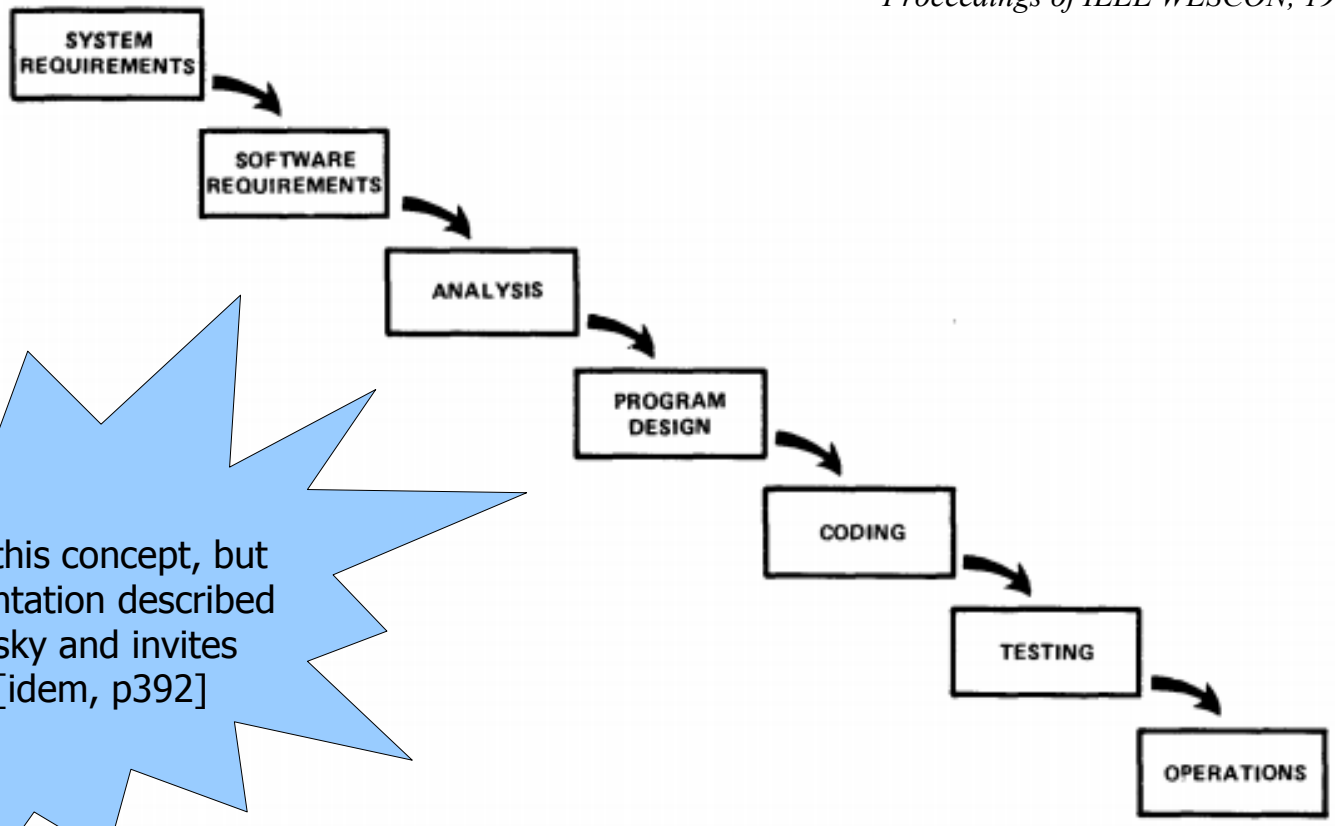


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

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

Problémy sekvenčního postupu

- Plán jako „zlaté tele“
  - » přehlednost a kontrolovatelnost
  - » vodopád, V-model
- Změna je součástí podnikání
  - » zákazník neví co chce
  - » dodavatel neví jak na to
- Dodávka celého systému najednou
  - » začátek: kompletní specifikace požadavků
- Napoprvé se to nepovede
  - » velký třesk
  - » všechna (špatná) překvapení na konci

Cyklický postup

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

- Opakování technických aktivit
  - » obsah podle sekvenční fáze, znalosti detailů
- Produkt postupně „roste“
  - » znalost, funkcionalita, kvalita, …

Spirálový model

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

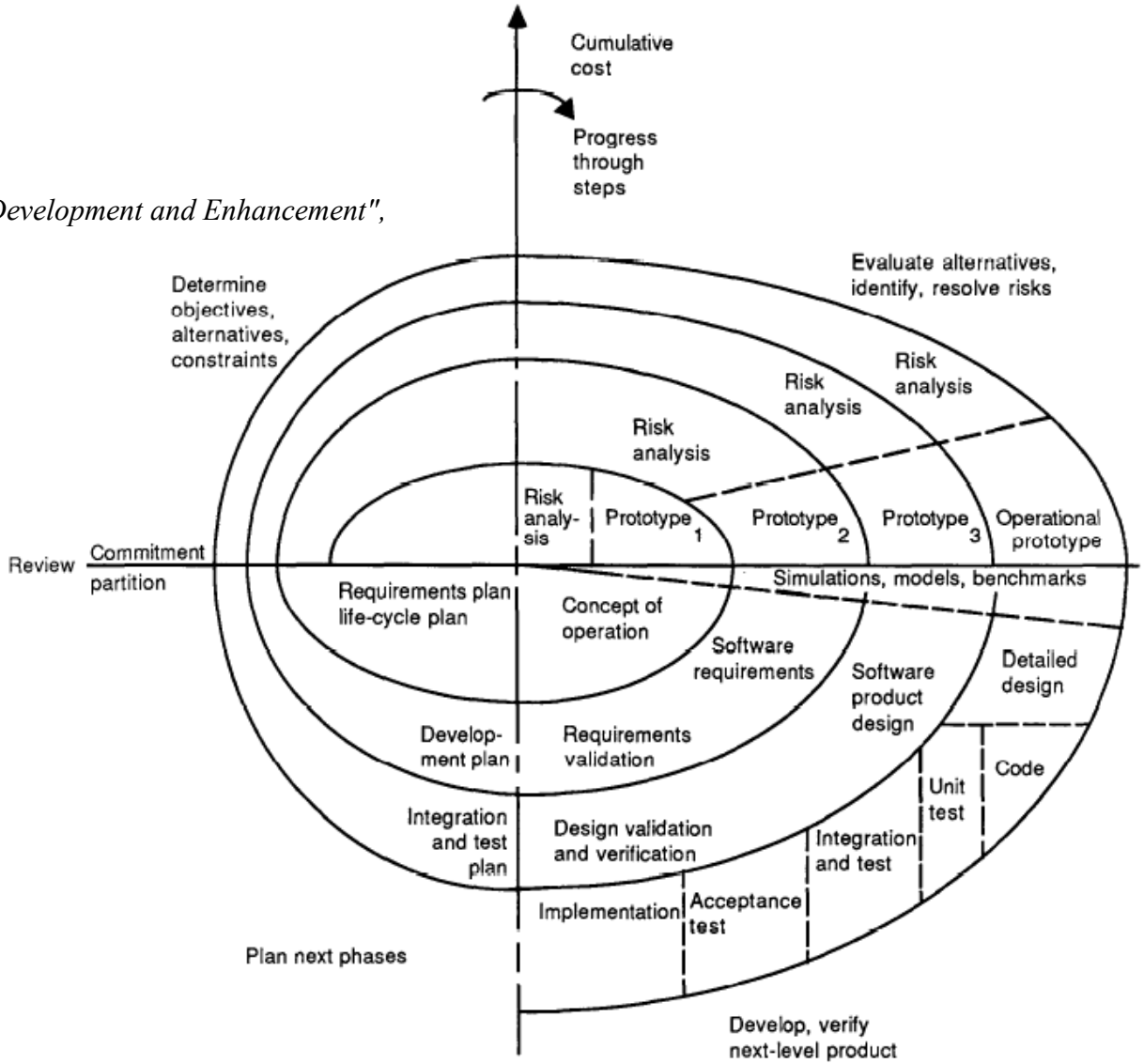
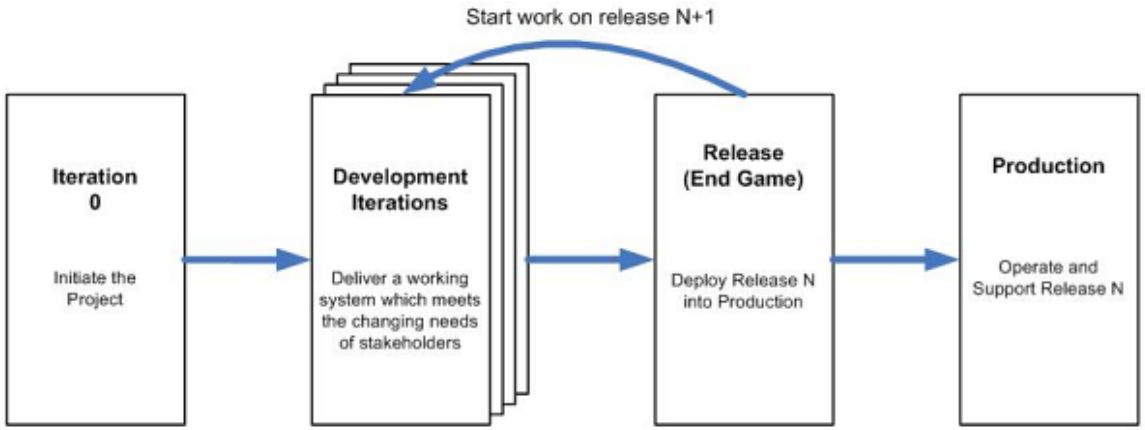


Figure 2. Spiral model of the software process.

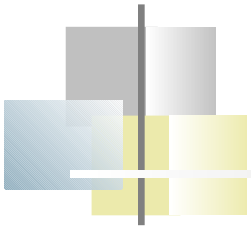
Alternativy dodávek funkčnosti

- Velký třesk
  - malé projekty, jasné požadavky
- Přírůstkově
  - » určení přírůstků -> plán -> postupné dodávky
  - zpětná vazba, ale úpravy projektu obtížné

+ iterativně ⇒ určování a plán průběžné, nutná



Jakou zvolit metodiku?



Software: Mýty vs Realita

- *Software není automobil*
- *Změna je život*
- *Dinosauři vyhynuli, myši nikoli*
- *Sjíždět vodopád je nebezpečné*

Pár čísel

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 resolution types: -

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

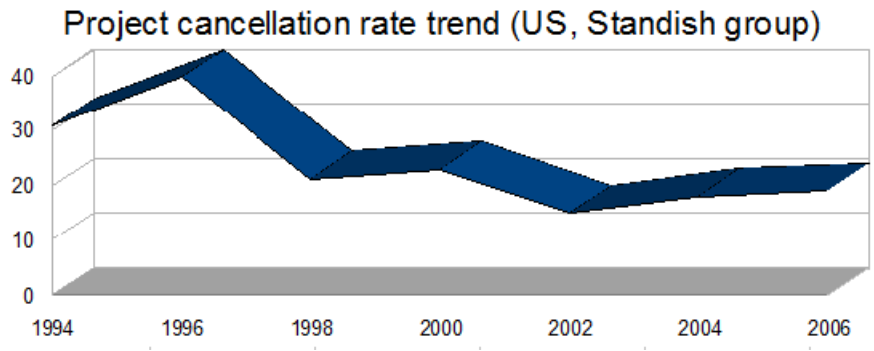
Overall, the success rate was only 16.2%, while challenged projects accounted for 52.7%, and impaired (cancelled) for 31.1%.

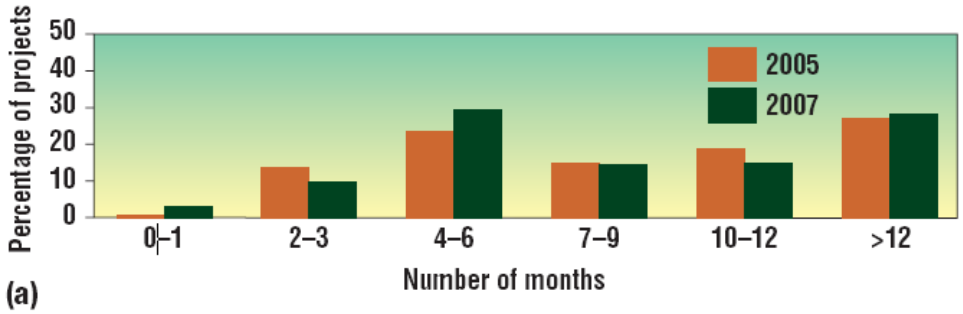
Data: Standish Group CHAOS Report 1995;
Emam, Koru: A Replicated Survey of IT Software Project Failures, IEEE Software 25(5), 2008

Realita stavu SWI

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





Kolik je $17 mld?

– tucet komerčních letů na Měsíc [google „project apollo cost"]
– 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"]

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

Discussion

- What are the symptoms of problems?
- To what root causes can they be traced?

THE STANDISH GROUP REPORT

© The Standish Group 1995. Reprinted here for sole academic purposes with written permission from The Standish Group.

CHAOS

Project Impaired Factors	% of Responses
1. Incomplete Requirements	13.1%
2. Lack of User Involvement	12.4%
3. Lack of Resources	10.6%
4. Unrealistic Expectations	9.9%
5. Lack of Executive Support	9.3%
6. Changing Requirements & Specif	8.7%
7. Lack of Planning	8.1%
8. Didn't Need It Any Longer	7.5%
9. Lack of IT Management	6.2%
10. Technology Illiteracy	4.3%
Other	9.9%

Reasons for project cancellation with percentages and 95% confidence intervals for the 2007 respondents (n = 18)*

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)

Mýty softwarových projektů

- Zákazník ví, co chce
  - pevné vlastnosti produktu
  - předem známý cílový stav
- Dodavatel ví, jak na to
  - predikovatelný postup, náklady, kvalita
  - lineární škálování složitosti projektu

- Tyto předpoklady – platné pro sériovou výrobu – používá příliš mnoho softwarových procesů (a manažerů a klientů)
  - založené na zjednodušeném, a idealizovaném, vodopádovém modelu

Tvorba SW není sériová výroba

Software není automobil

- Sériová výroba
  - CDčka, koloběžky, pračky, mobily, auta, paneláky
  - pevné a předem známé specifikace, známý cíl
  - známý výrobní postup, přesné odhady na začátku
  - malá míra variability a nutnost reakce na změny
  - problémem je logistika a ekonomie výroby kopií

- Tvorba software nemá (ve většině případů) charakter předvídatelného projektu a/nebo sériové výroby. Naopak: jde o vývoj nového (typu) produktu.
  - studie vozu, ekologický dům, raketoplán
  - produkt a projekt jedinečný, bez vzoru a modelu

Změny jsou pravidlem, ne výjimkou

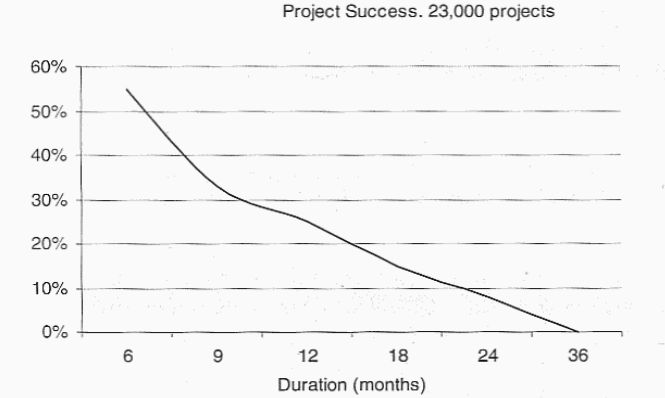
Změna je život

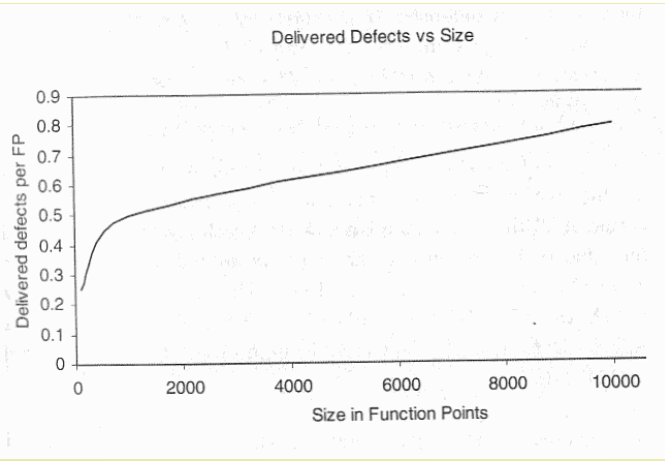
- Změny požadavků: spekulativní funkce/vlastnosti, fenomén IKIWISI, nedostatečná komunikace, ...
  - „Zákazník neví, co chce, neumí to říct, ale chce to“
  - typicky se změní zhruba 25% specifikovaných požadavků [Boehm88]
- Změny prostředí: legislativní rámec, akvizice firmy, upgrade systémů zákazníka, nové technologie, ...
- Změny postupu: fluktuace v týmu, chybná architektonická rozhodnutí, změny nástrojů, ...

Velikost pracuje proti nám

Dinosauři vyhynuli, myši nikoli

- Úspěšnost
  - velké plány, velká zklamání
    - přes 1/2 velkých zrušeno
  - potvrzováno teorií systémů
- Produktivita
  - nepřímá úměra k velikosti produktu
    - » větší pro malé přírůstky, týmy
- Četnost změn
  - 10% malé projekty (100 FP)
  - 35% velké (10000 FP)
- Kvalita
  - nepřímá úměra k velikosti produktu
  - prototyp se 40% → 20% funkčnosti ⇒ pokles chybovosti o 10/měs/MLOC



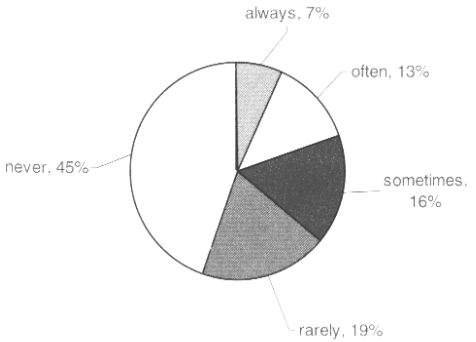


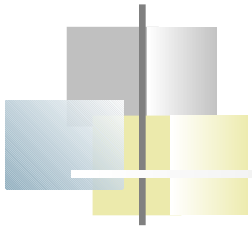
Zjednodušené modely nefungují

Sjíždět vodopád je nebezpečné

- „Pro každý složitý problém existuje řešení, které je jednoduché, elegantní, a špatné" [H.Mencken] – například vodopádový model v „klasickém" vydání ...
  - 4 z 5 faktorů neúspěchu projektů jsou spojeny s VM [Jones95]
  - použití VM nejvíce přispívá ke krachu projektů v 80% [Thomas01]
  - expertní doporučení je vyhnout se VM [Brooks87]
- Například
  - studie DoD 1995: ze systémů za celkem $37 mld, vyvíjených podle DOD-STD-2167, jich 46% nebylo nikdy nepoužito
  - US ATC projekt 1983-1994: vodopád, velký třesk, $2.6 mld, zrušeno
  - Johnson02: využití předem specifikovaných požadavků

Kolik je $17 mld?



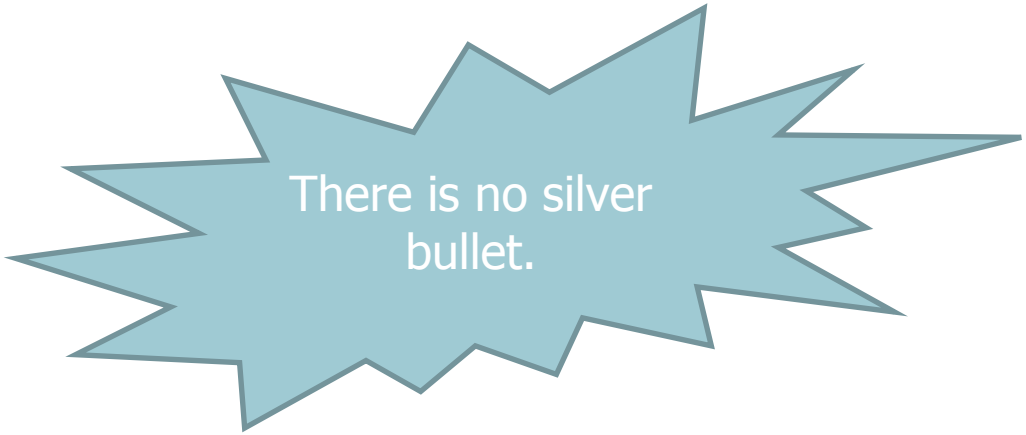


„Jestliže se čtení webu podobá prohlížení billboardů, pak navrhujte web tak, jako byste navrhovali billboard.“

Steve Krug

Čili: Jakou zvolit metodiku?

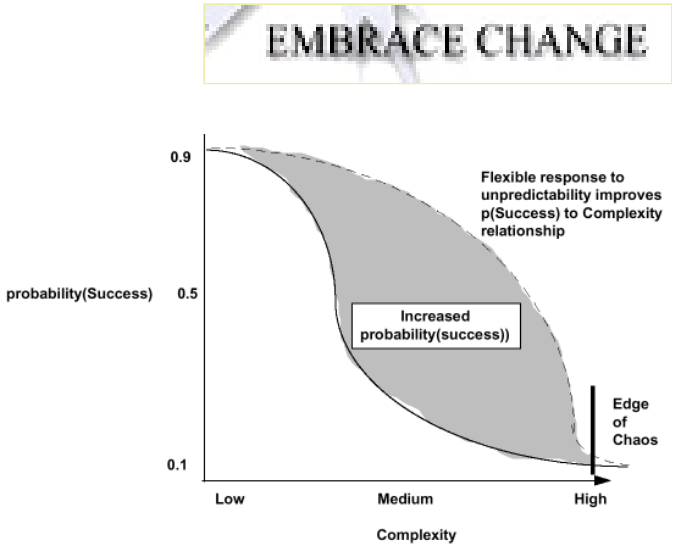
- Ad-hoc (neřízený proces)
- Sekvenční
- Iterativní
- Agilní

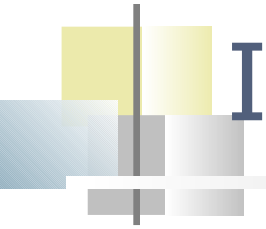


Řešení

- Přivítat změnu
  - » opustit to, co nefunguje – přístup vodopádu

- Iterativní a evoluční vývoj
- Adaptivní plánování
- Agilní přístup





Iterativní vývoj software

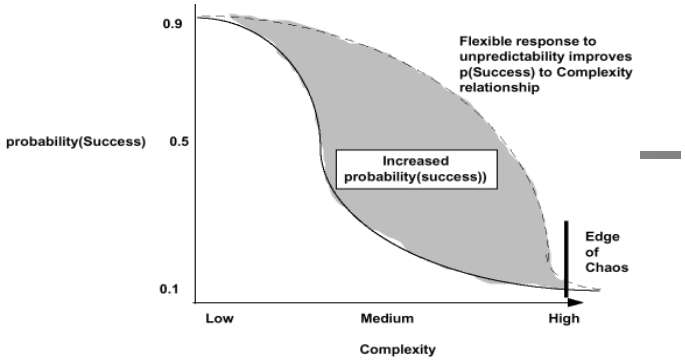
Q: Jaké můžeme v nejbližší době čekat nové, vzrušující a slibné myšlenky nebo techniky v oblasti software?

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

– David Parnas

EMBRACE CHANGE

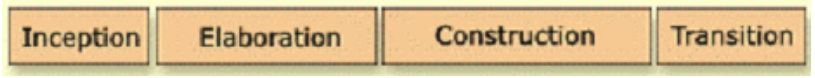
Přehled



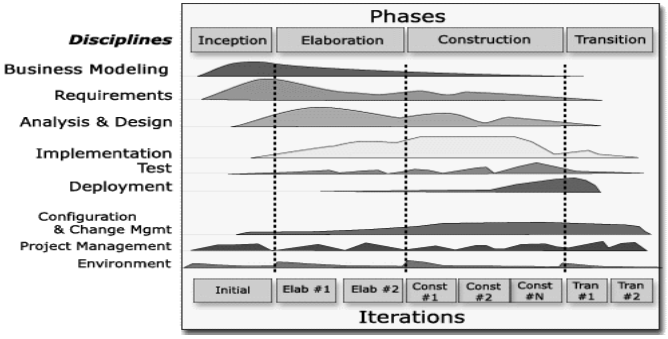
- Iterativní vývoj
  - včasná reakce na problémy při vývoji
- Evoluční (adaptivní) dodávky a plánování
  - podchycení změn požadavků
- Empirický proces
  - adaptace na změny týmu a postupu

Iterativní vývoj

- Rámcový plán životního cyklu
  - milníky – např. RUP

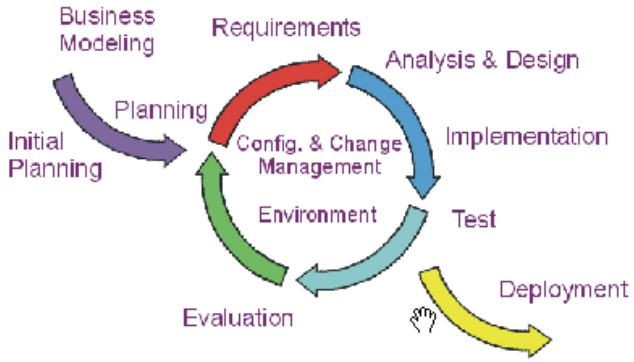


- Řetězec vývojových iterací
  - miniaturní úplný projekt – cca vodopádový model
  - cíl: iterační release (interní)
    - produkt funkčně neúplný
    - ale otestovaný a funkční
  - vede na přírůstkový vývoj



  - nevylučuje kompletní počáteční specifikaci požadavků

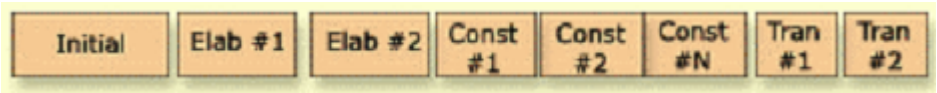
Průběh iterace



1. Plánování cíle iterace (funkčnost)
2. Doplnění / zpřesnění požadavků
3. Dotváření návrhu
4. Implementace přírůstku funkčností
5. Integrace přírůstku – ověření, otestování
6. Předání do provozu
    - » release interní / externí

… vodopád v malém

Počet a pravidla iterací



- Počet
  - charakter projektu (rozsah, velikost týmu)
  - fáze vývoje
  - obvykle alespoň 3 celkem
- Pevné datum ukončení
- Běžící iterace uzavřená změnám zvenčí
  - » nutné pro stabilitu projektu
  - » neakceptovat ani od šéfů (viz SCRUM)
  - nutnost dobrého změnového řízení
  - zdroje tlaku na změnu: čas, funkčnost, postup

Délka iterace

- Malá je lepší – blízký cíl, menší složitost/riziko, rychlá adaptace, vysoká produktivita (až 80 vs 25 FP/měs)
  - » 1-4 týdny pro malé, 3-6 týdnů velké projekty, zřídka měsíce
  - » psychologie: lidé si pamatují překročené termíny, ne opuštěné vlastnosti; nutí včas k těžkým rozhodnutím a kompromisům
- Vždy pevné datum ukončení
  - plánováno nejpozději na začátku iterace
- Timeboxované iterace = délka známa předem
  - omezení plánované funkčnosti možné
  - nelze: nehotový release, změna datumu, přesčasy

SCRUM: 30 dní
XP: 1-2 týdny

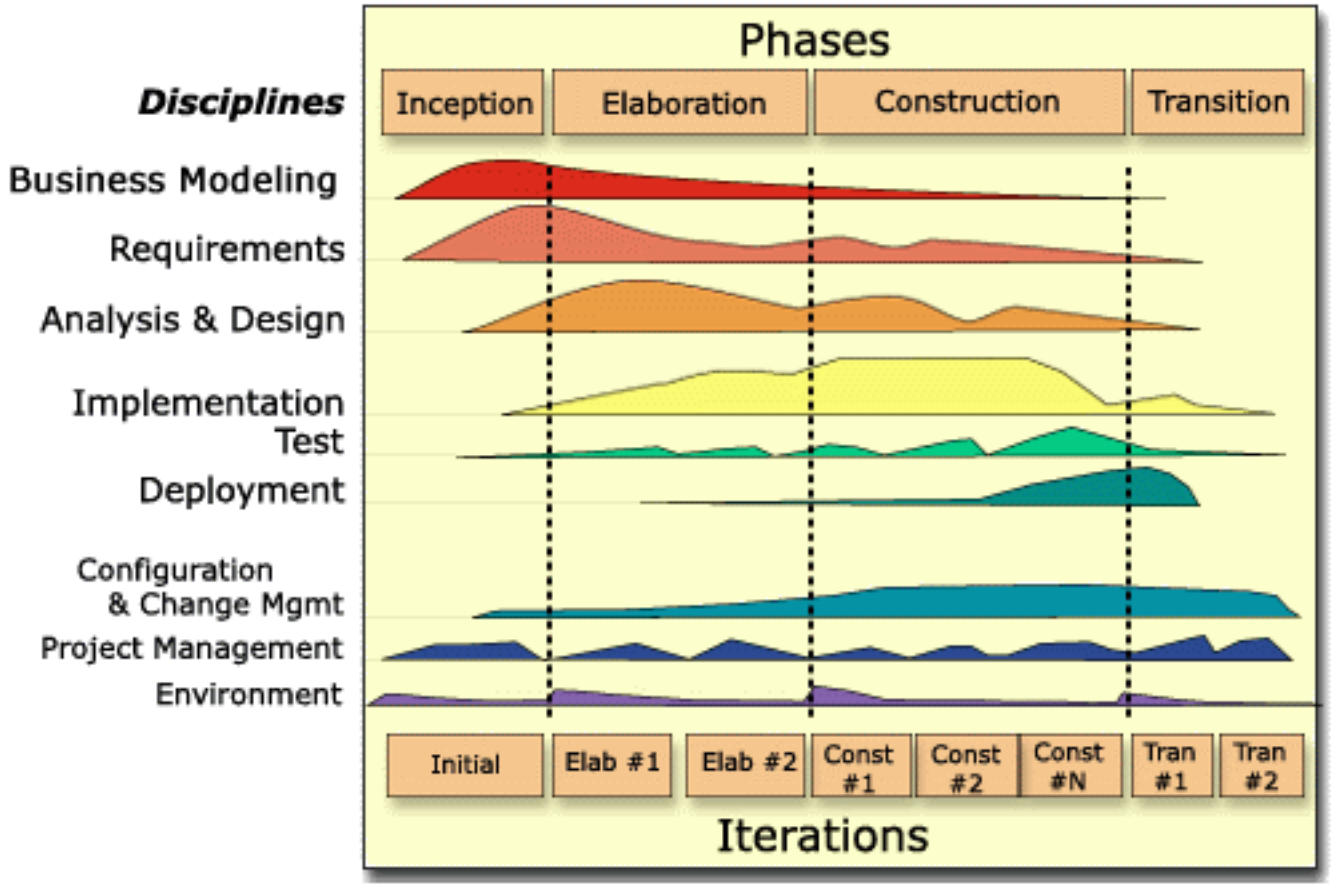
Globální řízení iterativního vývoje

- Problém: pro stromy nevidím les
- Oddělené sekvenční fáze
  - » analogie „klasických" inženýrských disciplin
  - » jasné rozdělení cílů a výsledků
  - milníky
    - » po stupních přesnosti, míře rizika
    - » vodopád: po činnostech 💣
- 1 fáze = 1..N iterací
- Inicializace projektu

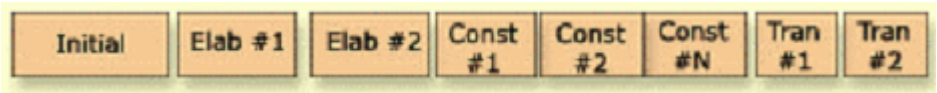
Barry Boehm (1995): *Anchoring the Software Process*

- LCO - Lifecycle Objectives
- LCA - Lifecycle Architecture
- IOC - Initial Operational Capability
- REL - Product Release

Fáze vývoje: příklad RUP



Charakter iterací dle fáze



- Základní schema iterace pevné
- Obsah, artefakty a počet iterací se mění
  - zahájení – analytické činnosti a produkty, validace vize zákazníkem; 1-2 iterace
  - projektování – analytické a designérské činnosti a produkty, ověřování prototypy; 2+ iterací
  - konstrukce – designérské a programátorské činnosti, změnové řízení, testování; N iterací
  - nasazení – integrační a konzultační činnosti, ověřování provozem; 1-2 iterace

Plánování, řízení a sledování iterativního vývoje

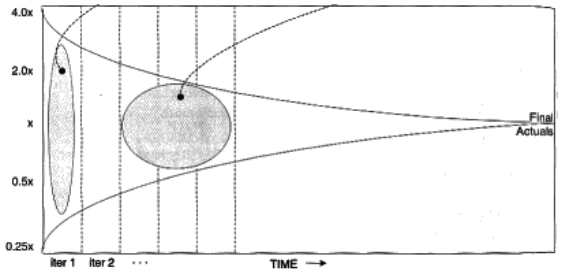
Cheap. Fast. Good. Choose any two.

Evoluční a adaptivní vývoj

… jeden z 4 nejčastějších faktorů úspěchu sw projektů

- Evoluční vývoj
  - dotažení iterativního přístupu
  - znalosti o požadavcích, návrhu, odhadech a plánu se vyvíjejí/zpřesňují v průběhu projektu
    - » žádné kompletní, dále neměnné specifikace na začátku (20-80)
    - » míra změny obvykle klesá s postupujícími iteracemi
  - „don't develop software, grow it"
- Adaptivní
  - zdůraznění zpětné vazby v evolučním vývoji
    - » analogie řízení auta
  - zejména evoluční dodávky – zpětná vazba od uživatelů

Adaptivní plánování

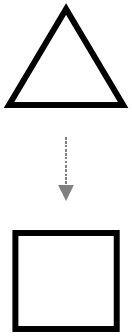


- Prediktivní plánování: velká míra nejistoty
  - » „plan work, work plan“
  - neznalost odhadů v době, kdy jsou potřeba
  - měnící se požadavky ⇒ rozsah projektu
- Řešení
  - přesnější odhady a plán až po několika iteracích
  - detailně plánovat jen na co máme rozumně přesná data
    - » obvykle nejvýše pro následující iteraci
    - » plus hrubé milníky (dodávky zákazníkovi)

Stupně volnosti při plánování

Cheap. Fast. Good.
Choose any two.

- Klasicky: čas, zdroje (cena), kvalita
  - obtížně měnitelné, odhadované
  - kvalita obtížně řiditelná
    - » typický požadavek: „bude to v termínu, s daným rozpočtem, a v bezchybné kvalitě jako vždy“ … „you get crappy SW late“
- Agilně: +funkčnost
  - nejlepší faktor pro řízení projektu
    - » první tři pevné, funkčnost nejsnáze měnitelná
  - vhodná granularita ⇒ snadné a přesné odhady



Iteration Plan

- Shows timeframes and resources by discipline

Outline of an Iteration Plan

1. Objectives
2. Scope
3. References
4. Plan
   - 4.1 Iteration Activities
   - 4.2 Iteration Schedule
   - 4.3 Iteration Deliverables
5. Resources
   - 5.1 Staffing Resources
   - 5.2 Financial Resources
   - 5.3 Equipment & Facilities Resources
6. Use Cases
7. Evaluation Criteria

Iteration Schedule section for Requirements discipline

Requirements	40 days	Tue 12/1/98	Mon 1/25/99	
Develop Vision	25 days	Tue 12/1/98	Mon 1/4/99	System Analyst
Elicit Stakeholder Requests	4 days	Tue 1/5/99	Fri 1/8/99	System Analyst
Manage Dependencies	26 days	Tue 12/1/98	Tue 1/5/99	System Analyst
Capture a Common Vocabulary	10 days	Wed 12/23/98	Tue 1/5/99	System Analyst

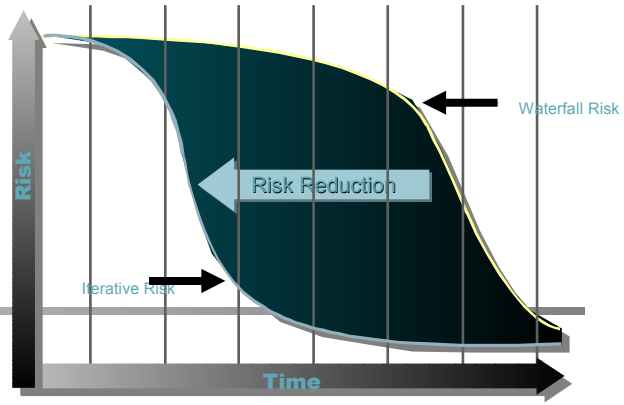
Plánování podle rizik a/nebo priorit klienta

Již z dob spirálového modelu (1986)

» Kontext: plán iterace (výběr funkčnosti)

- Řízení riziky
  - vyhodnotit rizikové faktory projektu
    - » designová/architektonická rizika, obchodní, legislativní, neznámá funkčnost, použitelnost, …
  - začít s částmi funkčnosti/designu s největší mírou rizika
- Řízení prioritami klienta
  - výběr funkčnosti je na zákaznícovi
    - » množství funkcí omezeno délkou iterace
  - umožňuje pružně reagovat na aktuální potřeby

Risk Terms



- **Direct risk** - the project has a large degree of control
- **Indirect risk** - the project has little or no control

- **Risk Magnitude** is used for ranking risks. It is a combination of:
  - Probability of occurrence
  - Impact on the project (severity) e.g. project delays

- Týmové rozhodování: dot voting

Výsledek: Empirický proces

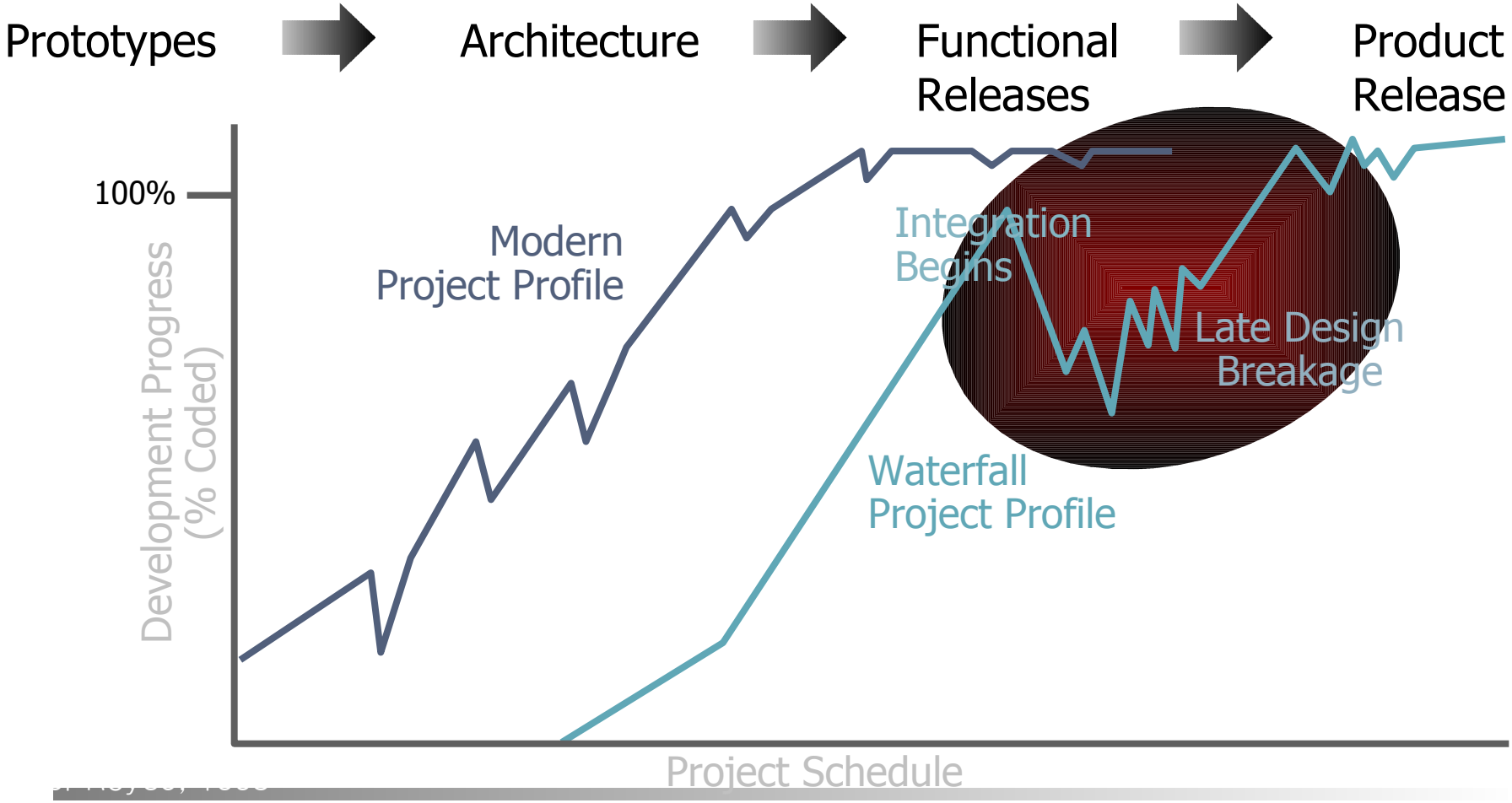
- Kontrast: pevně definovaný proces („rule-based“)
  - předem známé/dané aktivity, jejich návaznost
  - PERT diagram

- Empirický („principle-based“)
  - » uznání, že vývoj software není sériová výroba
  - sada jednoduchých aktivit
  - časté měření procesu a zpětná vazba
  - dynamická adaptace na změny a události
    - » emergent behaviour, samoorganizující tým
  - přizpůsobení typu (závažnosti) projektu

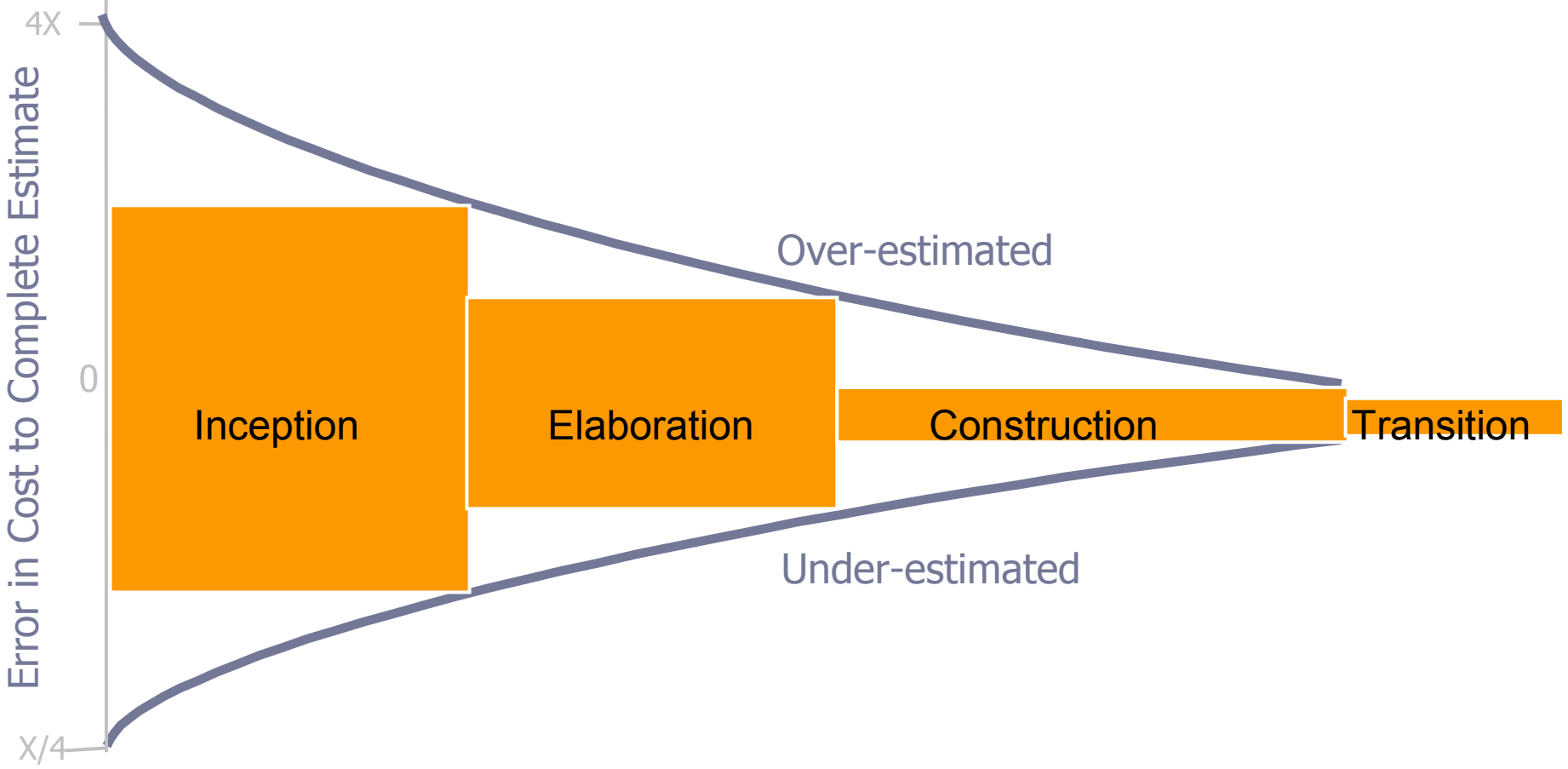
SCRUM: denní setkání týmu, „empowered team“

XP: denní setkání týmu, role „tracker“, plánovací hra

Better Progress Profile



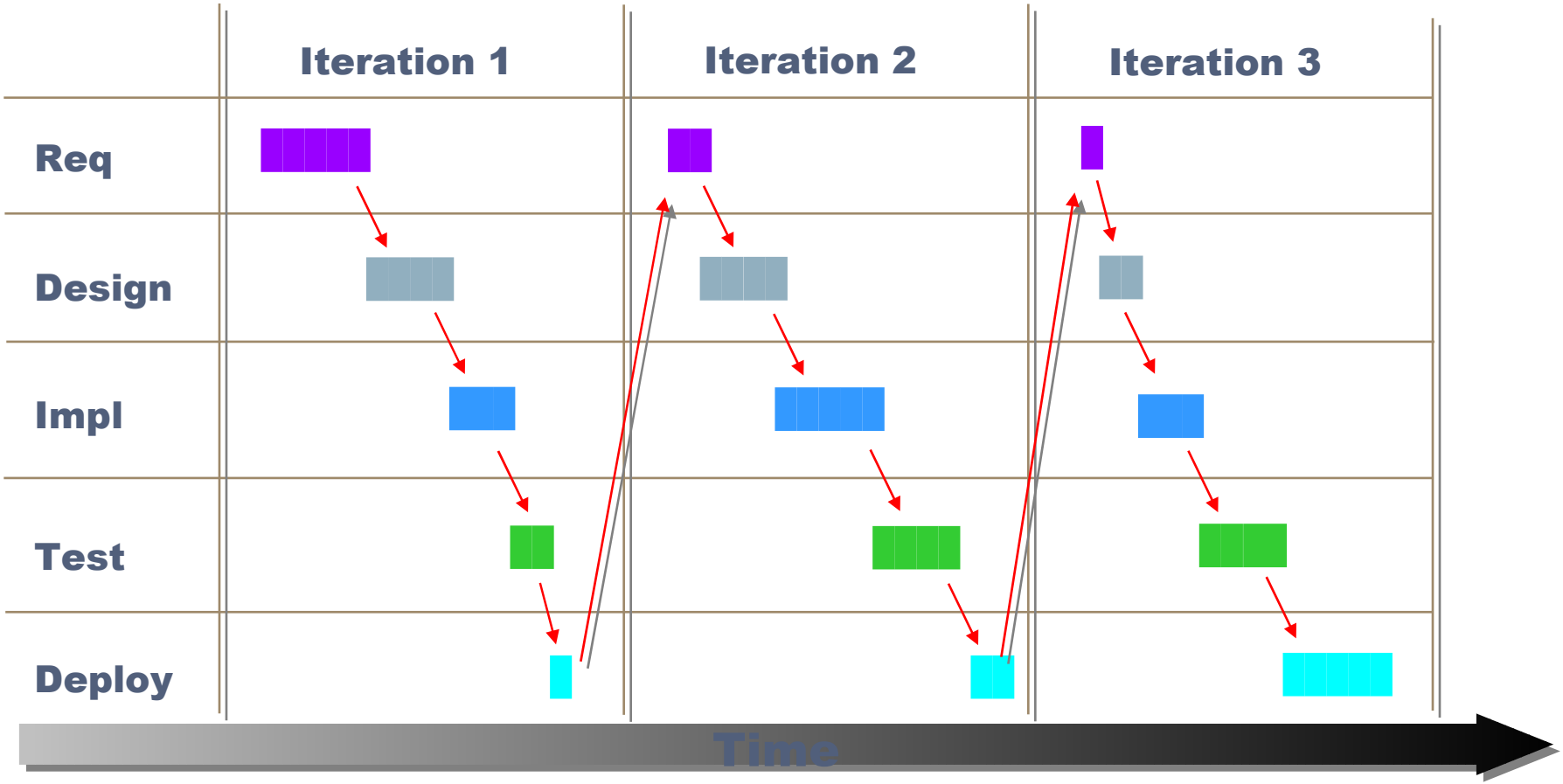
Cost Estimate Fidelity



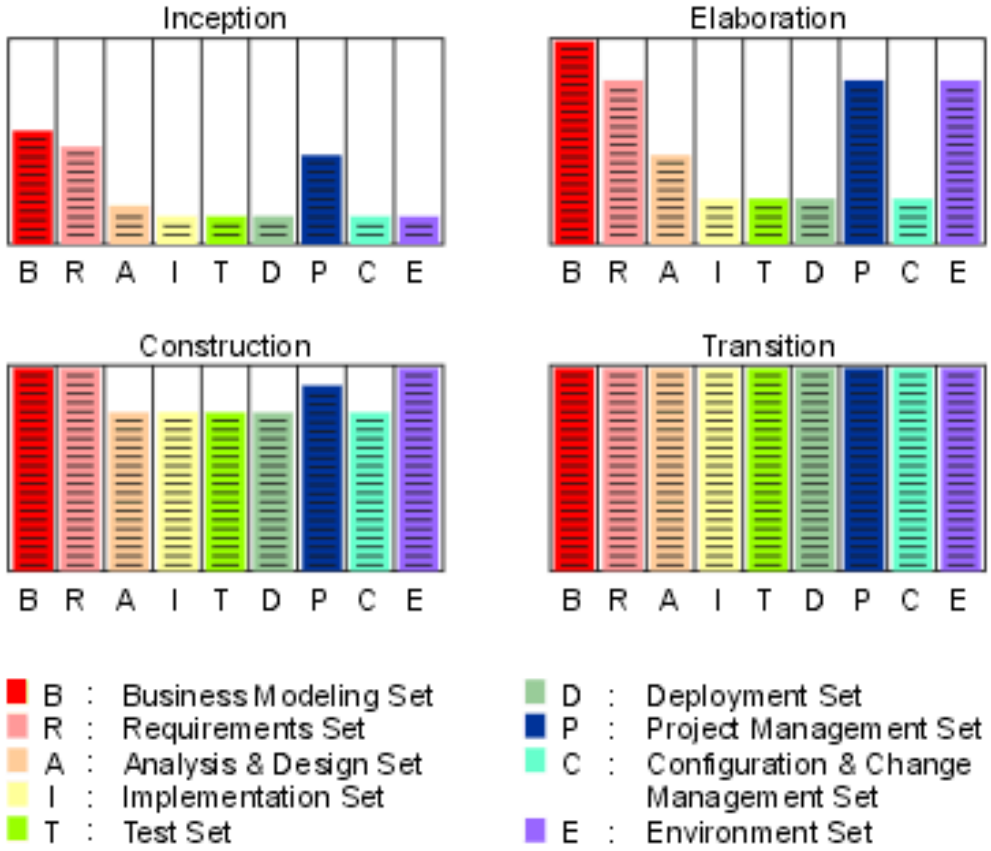
Význam meziproduktů v iterativním vývoji

- Kontrast: postup řízený jistotou – meziprodukty (artefakty) jsou cílem a indikátorem dosažení cíle
  - důsledek: review → podpis → změnové řízení
- Agilní přístup – cílem je funkční software
  - artefakty prostředkem k dosažení (cíl = test smysluplnosti meziproduktu)
  - forma, obsah artefaktů („dress code“): od zcela volné (XP) po vzory a šablony (RUP)
  - artefakty živé během projektu, výběr dle fáze/iterace

Changing Focus Over Time



Lifecycle Evolution of Artifacts

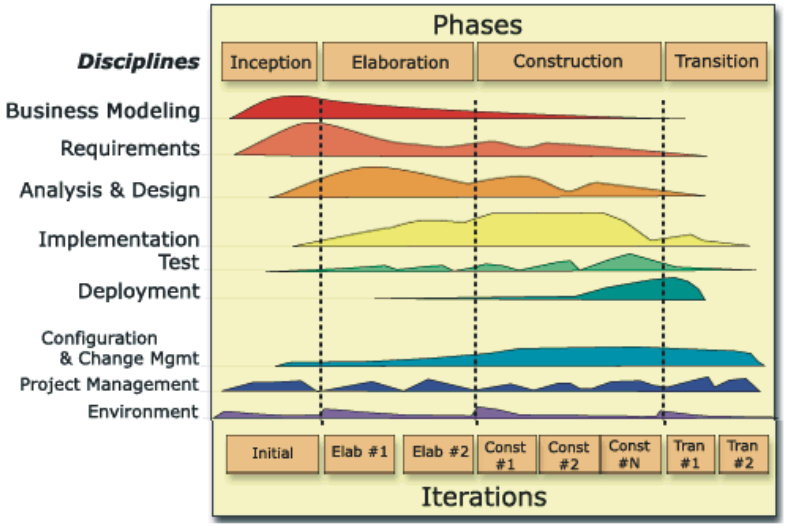


Information set evolution over the development phases.

- Artifact sets mature over time.

Shnutí

Iterativní vývoj



Achieving efficiency:

Through a Process

- Iterative approach preferred
- Process focus on architecture

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

© The Standish Group 1995. Reprinted here for sole academic purposes with written permission from The Standish Group.

CHAOS