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/

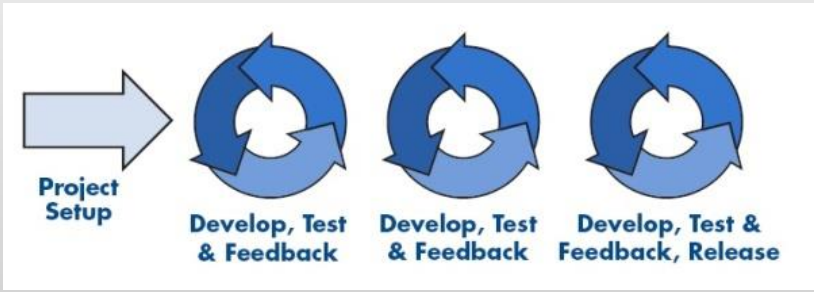
We didn't call those things by those names back then, but if you look at my first book (Computer Programming Fundamentals, Leeds & Weinberg, first edition 1961 —MB) and many others since, you'll see that was always the way we thought was the only logical way to do things. I learned it from Bernie Dimsdale,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

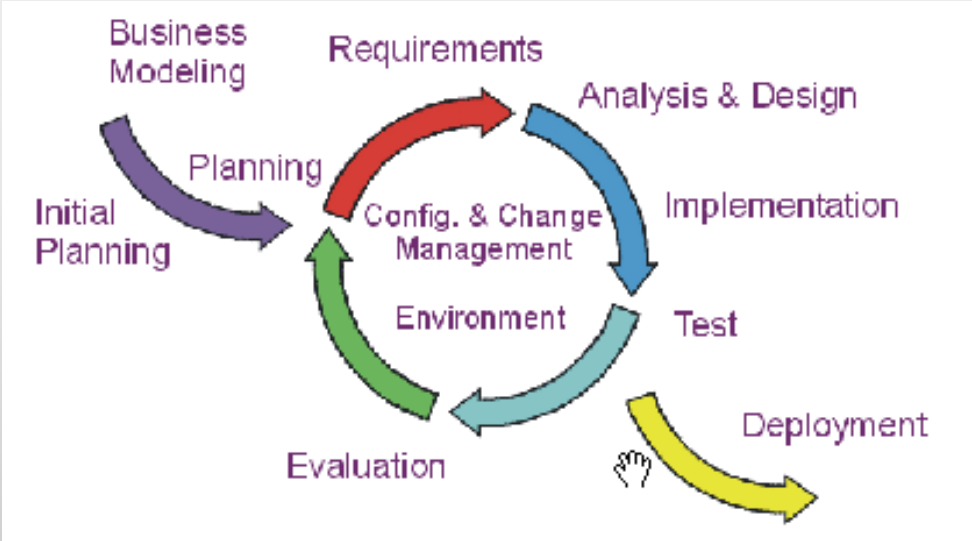


Průběh iterace

- 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

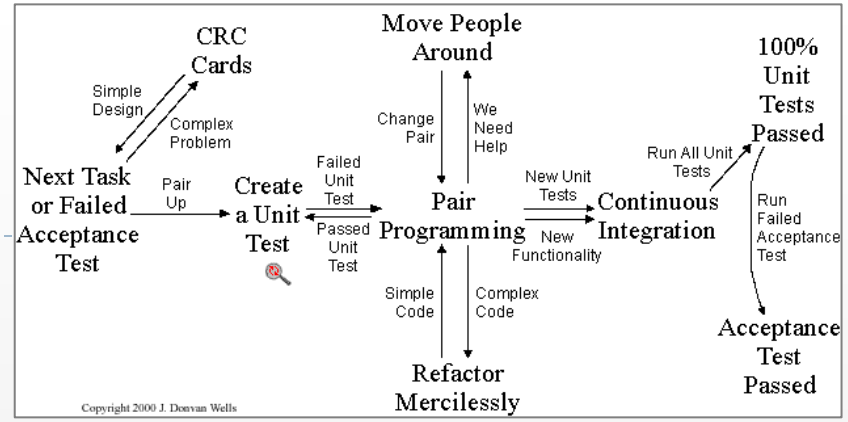
Schwaber: SCRUM Development Process

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

DAILY SCRUM MEETING

24 HOURS

PRODUCT BACKLOG

SPRINT BACKLOG

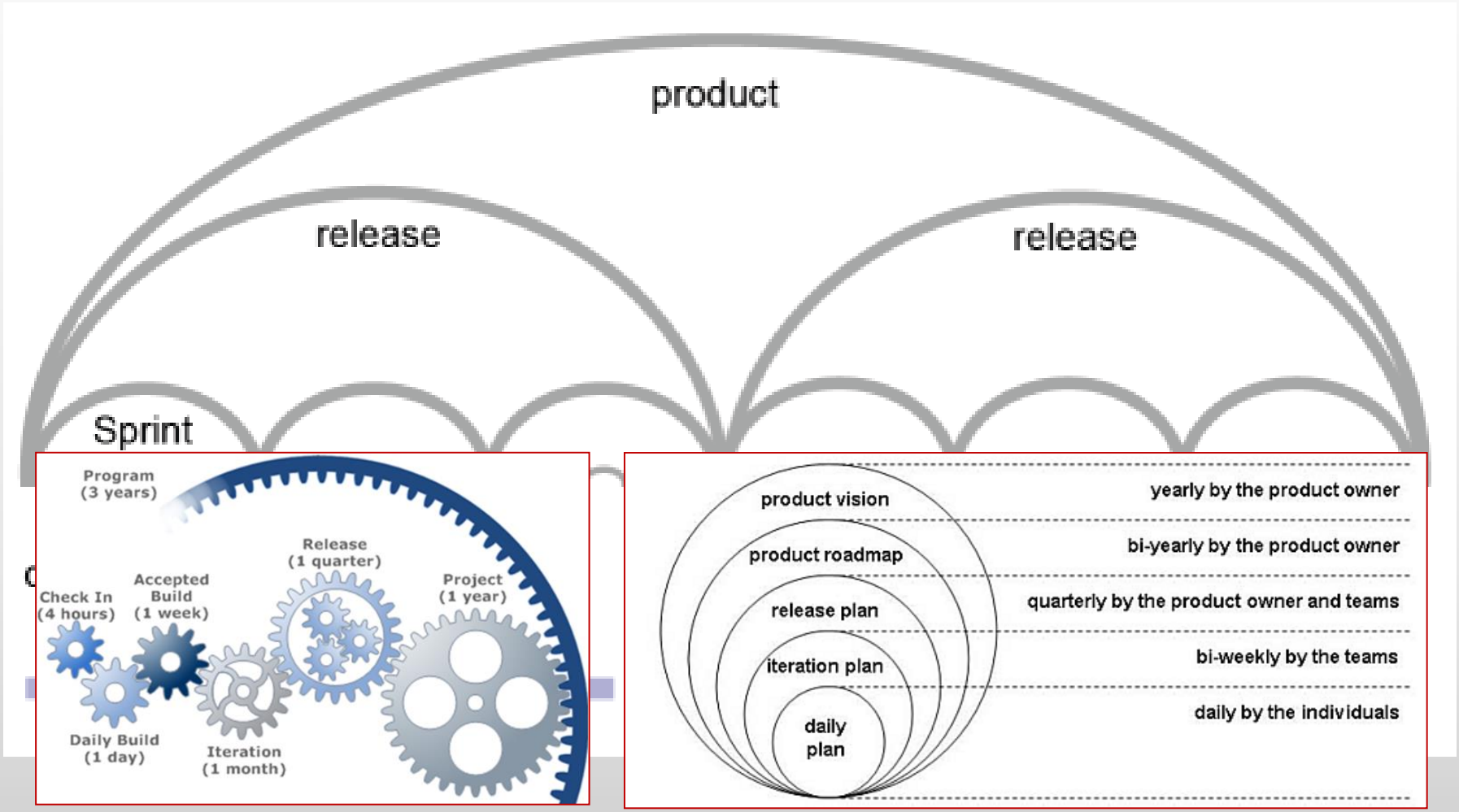
2-4 WEEKS

POTENTIALLY SHIPPABLE PRODUCT INCREMENT

Post-game

Iterace v kontextu

Kontext iterace v procesu vývoje



Globální řízení iterativního vývoje

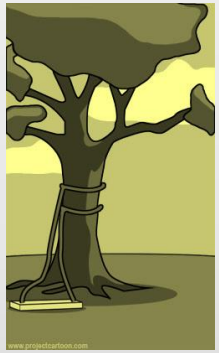
Problém: pro stromy nevidím les

Jak vysvětleno zákazníkem

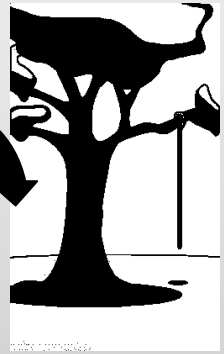
I-1 I-2 I-3 I-4 I-5

Co navrhl architekt

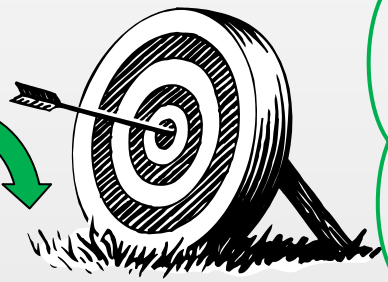
I-2 I-3 I-4 I-5

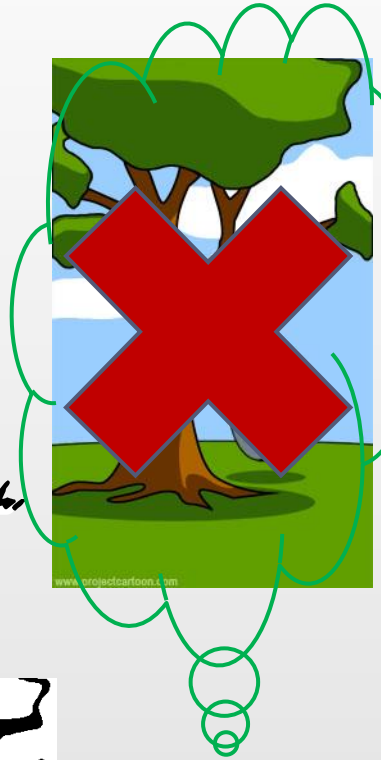


Implementace



Dodáno na konci





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

Globální plánování: milníky

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

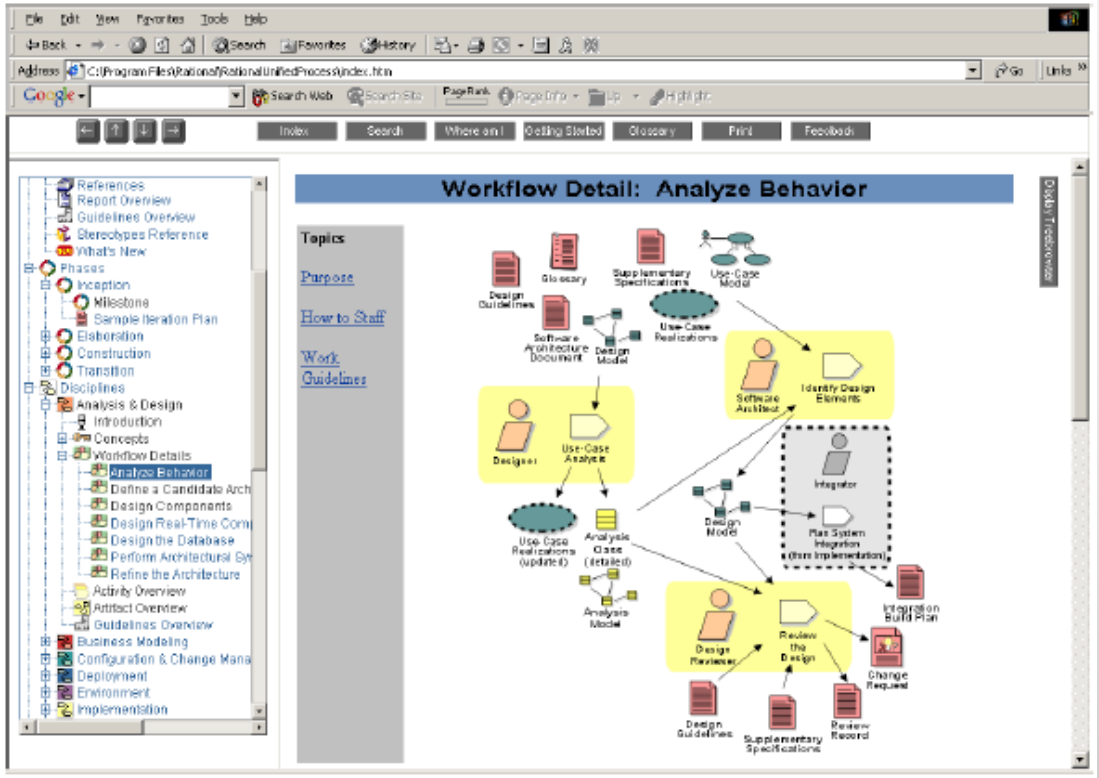
- **LCO** (Lifecycle Objectives)
  - definování terče – Vize produktu
- **LCA** (Lifecycle Architecture)
  - určení způsobu řešení – Architektura technického řešení
  - ověření – modely, technické prototypy, testy (executable)
- **IOC** (Initial Operational Capability)
  - schopnost efektivně „vyrobit“ 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

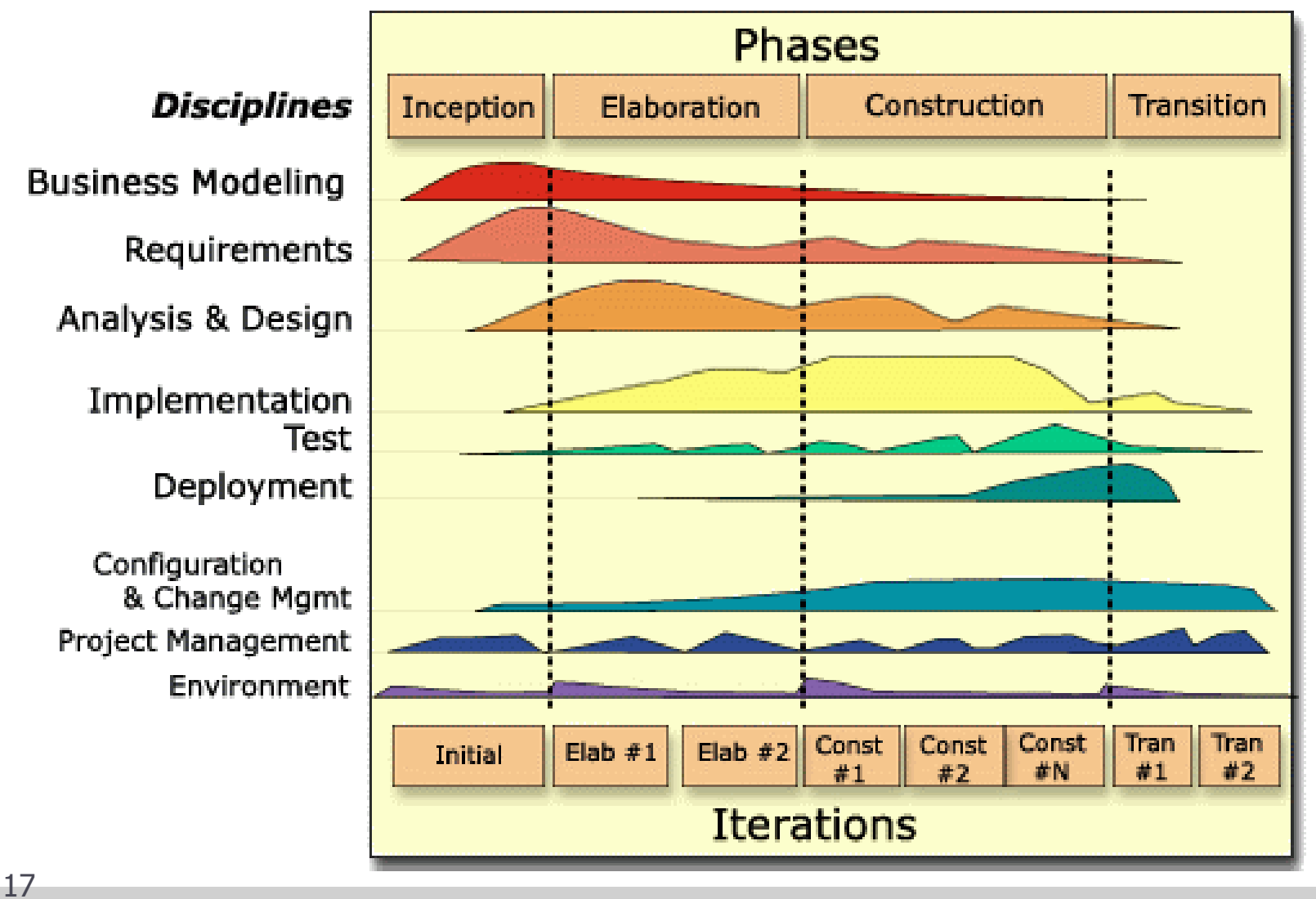


Rational Software White Paper
TP026B, Rev 11/01

RUP


Rational Unified Process
RUP
Proven. Practical. Flexible
BUSINESS MODELING
PHASES
ITERATIONS
IMPLEMENTATION
TEST
Analysis & Design
Implementation
Testing
Management
Environment
Configuration Management
Business Modeling
Planning
Evaluation
Initial Planning
Deployment
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

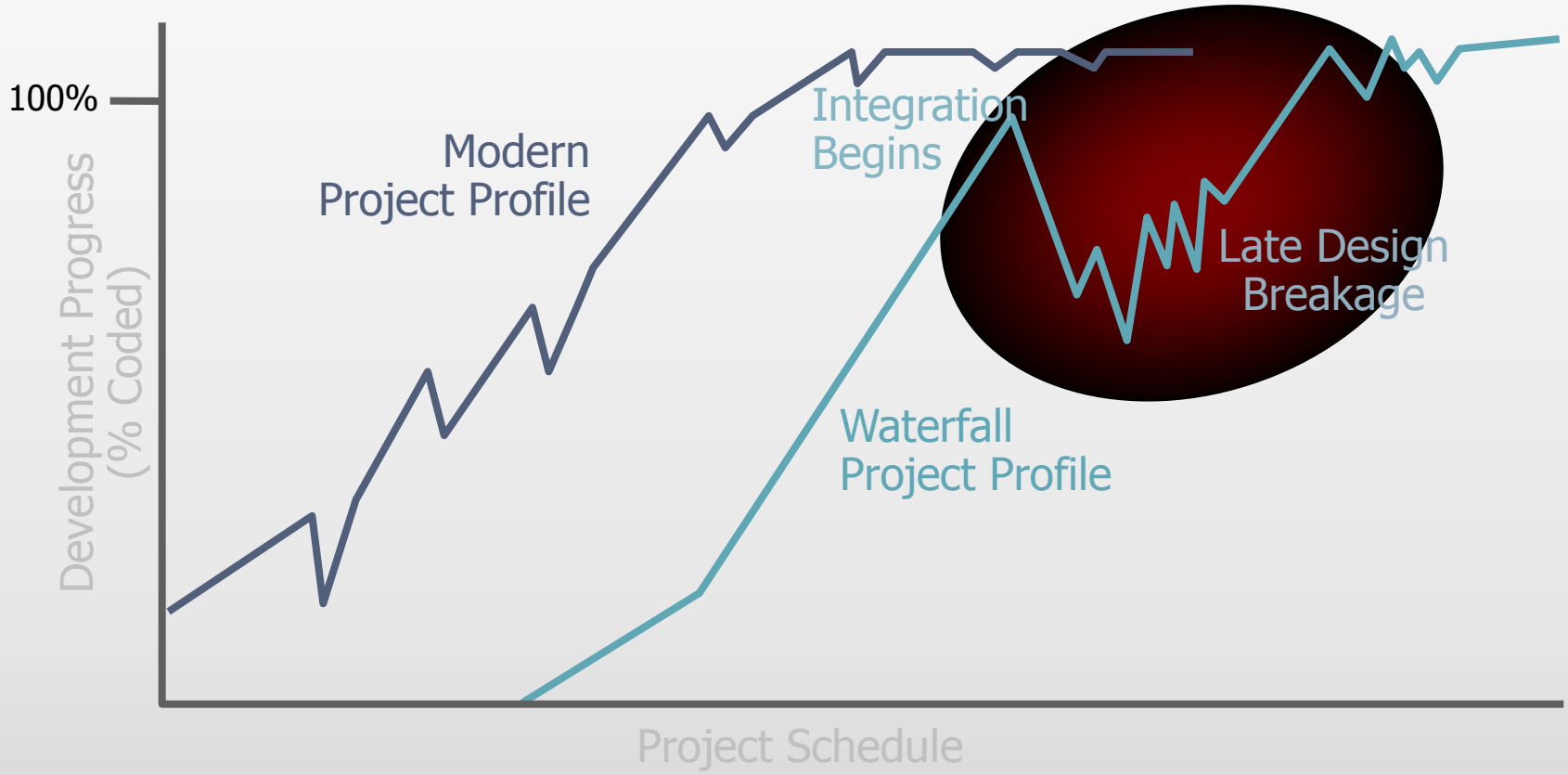
Evoluční a adaptivní charakter

… jeden z 4 nejčastějších faktorů úspěchu sw projektů

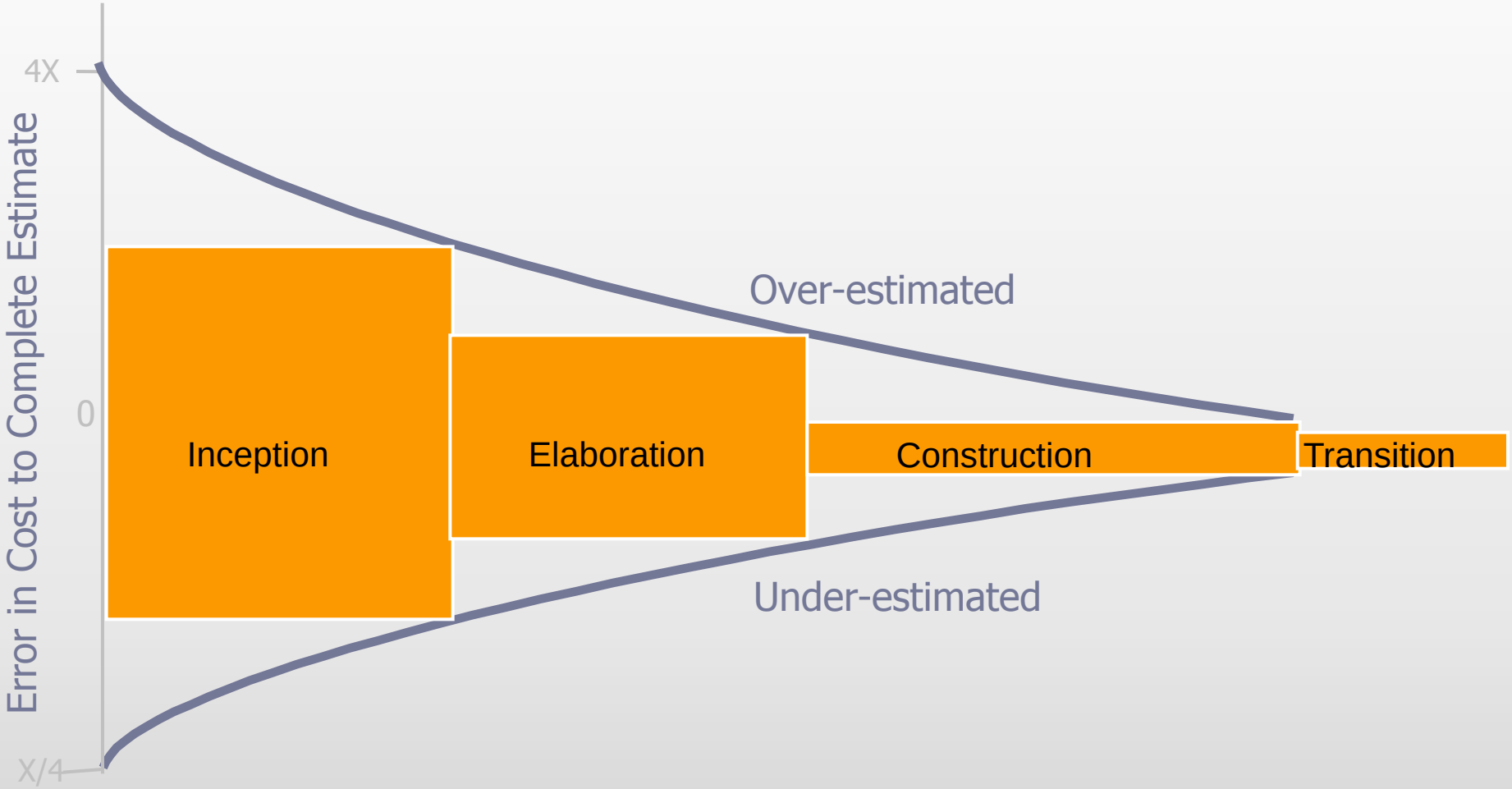
- Evoluční
  - znalosti o požadavcích, návrhu, odhadech a plánu se vyvíjejí a zpřesňují v průběhu projektu
    - vs kompletní, dále neměnné specifikace na začátku (20-80)
    - míra změny obvykle klesá s postupujícími iteracemi
  - „don't develop software, grow it"
- Adaptivní
  - zdůraznění procesu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

19

Better Progress Profile

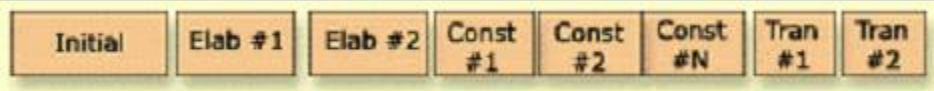


Cost Estimate Fidelity



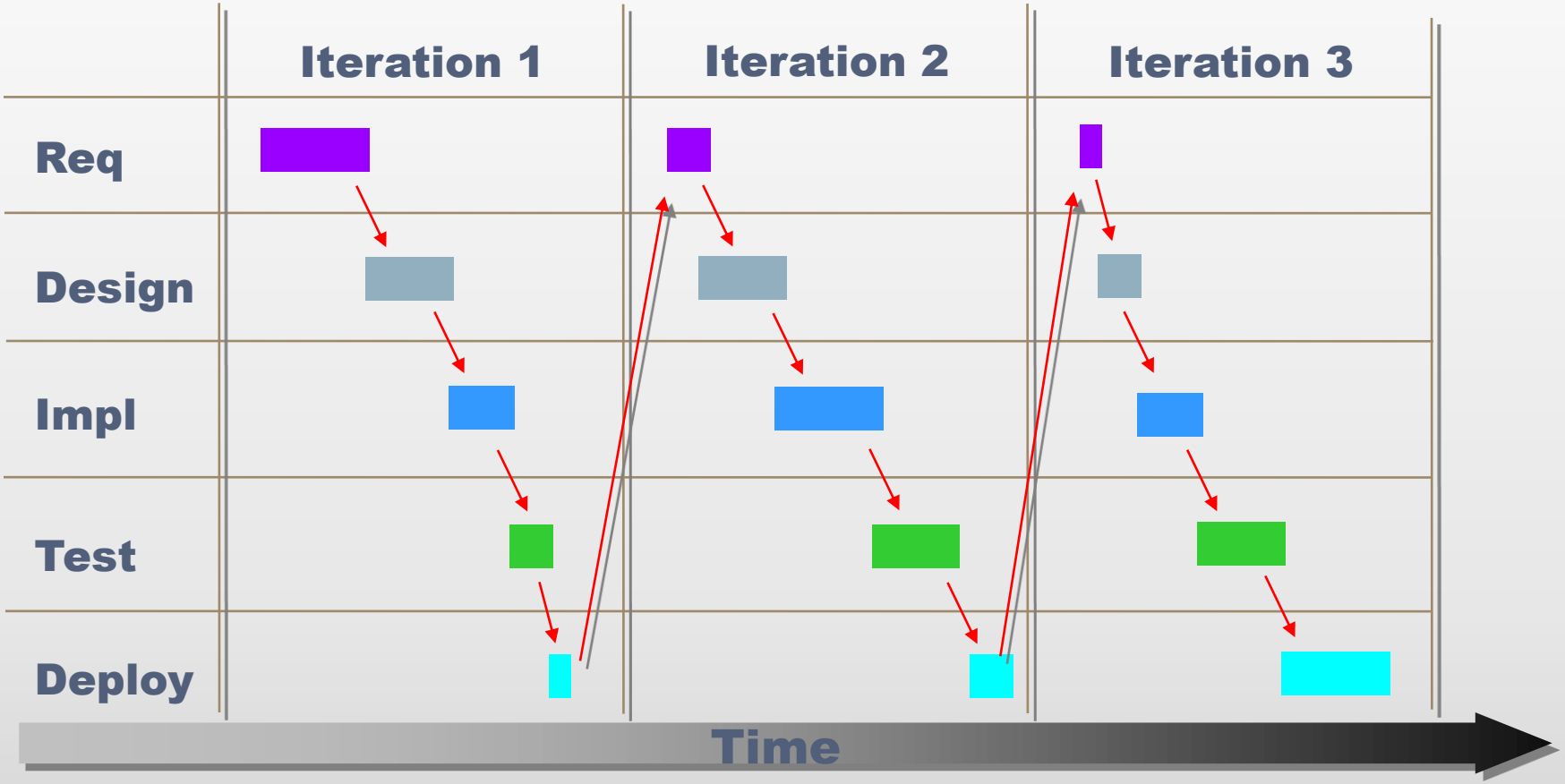
Charakter iterací dle fáze

- 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



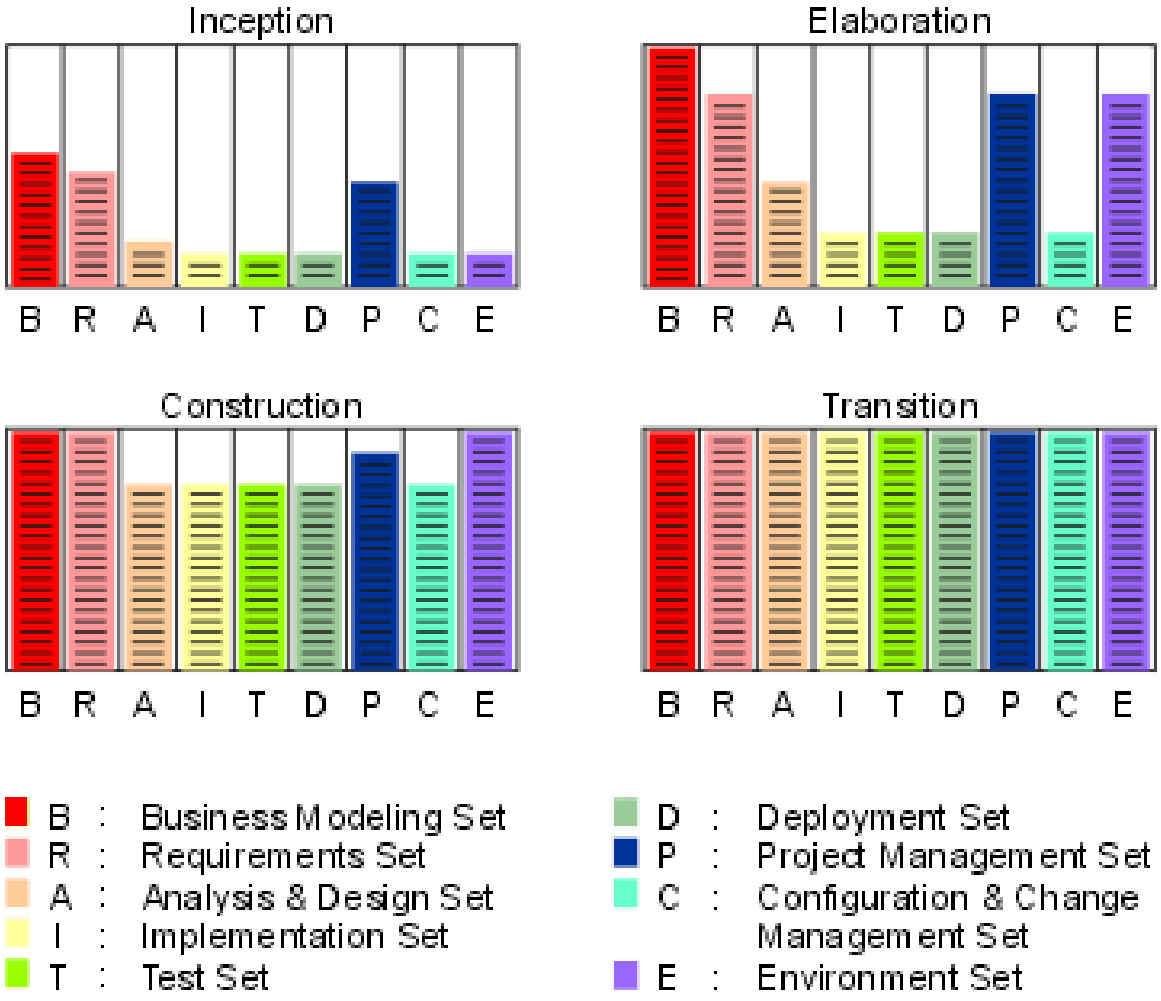
Example: RUP project profile: overview

Charakter iterací dle fáze: činnosti



Example: RUP project profile: details

Charakter iterací dle fáze: artefakty

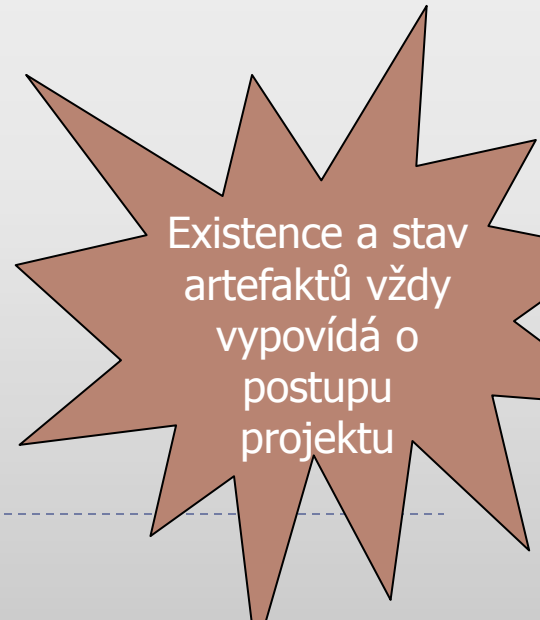


Information set evolution over the development phases.

Význam meziproduktů

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

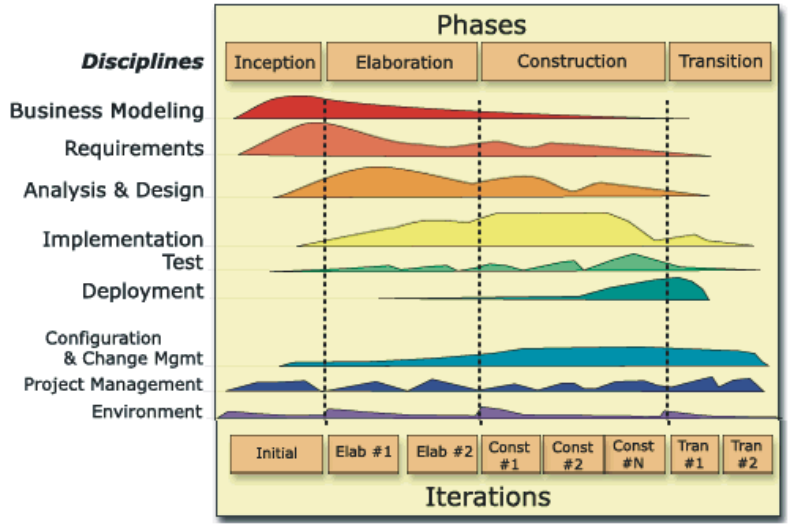
- Agilní přístup
  - artefakty jsou prostředkem
    (cíl = smysluplný stav/přírůstek produktu)
  - důsledky
    - forma, obsah artefaktů („dress code“):
      od zcela volné (XP) po vzory a šablony (RUP)
    - artefakty živé během projektu
    - výběr dle fáze/iterace



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