Podpora životního cyklu vývoje – sliby a realita

Michael Juřek
mjurek@microsoft.com
Software Architect
Microsoft s.r.o.

Microsoft Visual Studio 2005

Varování

- Obsah této prezentace není založen na žádném objektivním výzkumu
- Odráží zkušenosti autora při práci s množinou řádově desítkami dodavatelů SW
- Názor je nutně subjektivní
- Výběr firem není zcela reprezentativní (inklinují k MS technologiím)

Agenda

- Metodika
- Moje metodická doporučení
- Nástroje pro podporu životního cyklu vývoje
- Demo – TFS 2005

1. vlna – „vodopád“

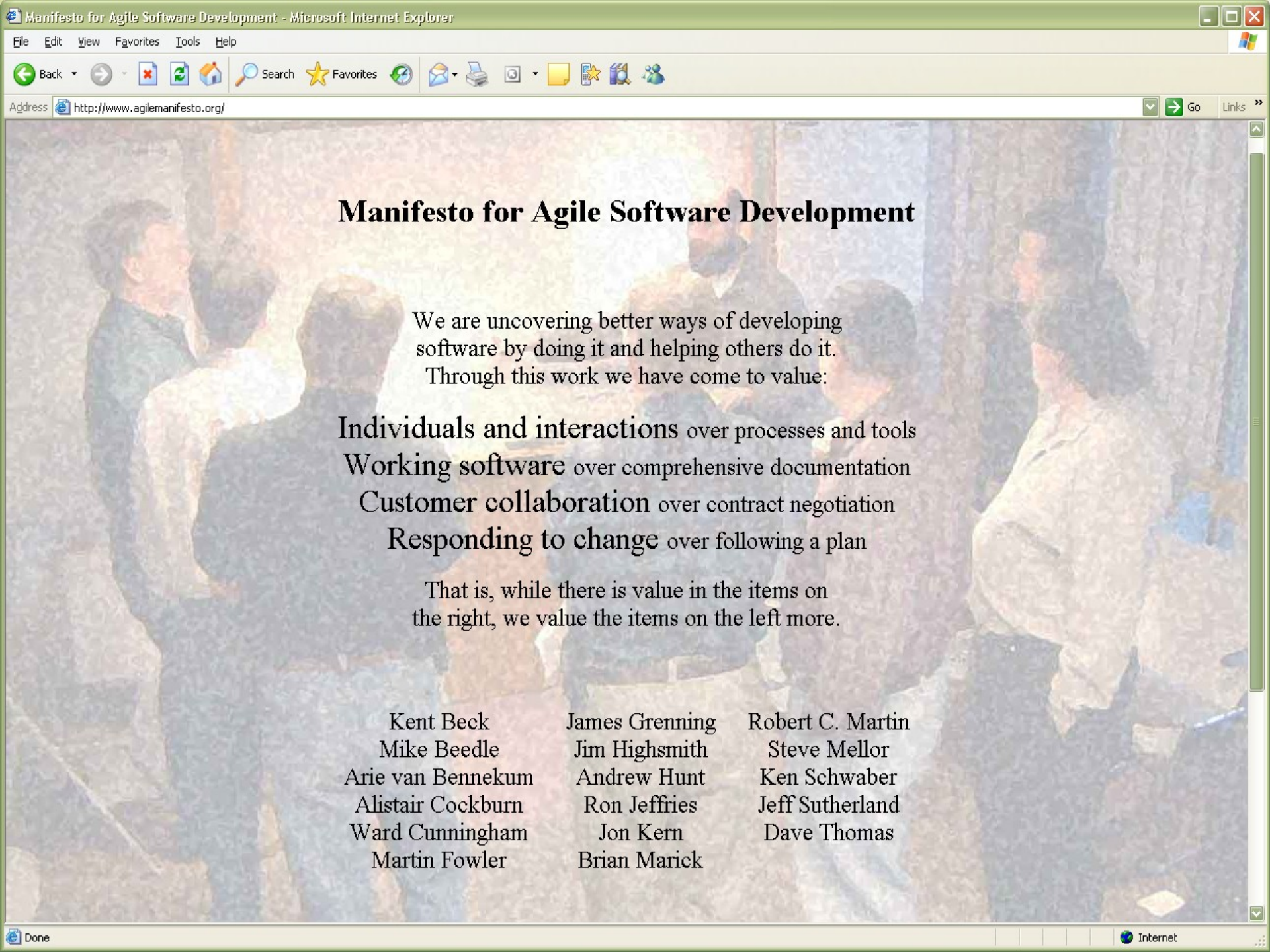
Budeme analýzu dělit na menší a menší části tak dlouho, až bude rozdělena na elementární kroky, ve kterých „nelze“ udělat chybu. Poté implementujeme.

- Přístup odráží vysokou cenu vytvoření verze v dřívějších dobách
- Stále má své zastánce, ale odumírá

2. vlna – zavedení iterací

Řešení budeme vytvářet v definovaných krocích (iteracích), abychom snížili rizika a měřili postup. Jednotlivé kroky budeme řídit striktním procesem.

- Iterace jsou umožněny nízkou cenou za vytvoření verze (důsledek komfortu vývojáře)
- Přiznává „právo dělat chyby“
- Vhodné pro projekty s velkou mírou rizika, ale dosti nákladné
- Představitelé:
  - RUP (Rational Unified Process) – nejznámnější
  - MSF (Microsoft Solution Framework)


Manifesto for Agile Software Development - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.agilemanifesto.org/
Go Links
Manifesto for Agile Software Development
We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:
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.
Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas
Done
Internet

3. vlna – agilní rebelie

Individuals and interactions *over processes and tools*

Working software *over comprehensive documentation*
Customer collaboration *over contract negotiation*
Responding to change *over following a plan*

- www.agilemanifesto.org
- Minimalizuje byrokracii a formálnost
- Snížení ceny je vyváženo zvýšením rizik
- Představitelé:
  - Fundamentalisté – Extreme Programming (XP), Continuous Integration (CI), Test-Driven Development (TDD), ...
  - Pragmatici – MSF for Agile Dev. (MSF4AD), Feature Driven Development (FDD), SCRUM, ...

Realita – co se v ČR používá?

- Dosud jsem neviděl nikoho, kdo by převzal a používal některou metodiku
- Kvalita vývoje nezávisí na velikosti firmy
  - Převládá ladění do pragmatického agilního tónu
  - Firmy používají směs:
    - Vlastních zkušeností
    - Inspirace z různých metodik
- Slabým článkem je zákazník:
  - Nezná realitu vývoje, nepožaduje kvalitu a proces, zajímá ho často pouze cena

Agenda

- Metodika
- Moje metodická doporučení
- Nástroje pro podporu životního cyklu vývoje
- Demo – TFS 2005

1. Definujte vizi

- Cca 3 věty říkající:
  - Co děláme?
  - Proč to děláme?
  - Co to přinese?
- Připravená odpověď na základní dotaz (v angličtině „elevator pitch“)
- Popisuje cíl projektu
- „Vytesejte do žuly a zasaďte nad dveře.“

2. Popište důležité osoby

- V každém projektu je řada důležitých osob (actor, persona, ...):
- Uživatel, manažer, finanční ředitel, administrátor, ...
- Pochopte, jak každý z nich funguje a co od projektu očekává
- Snažte se každého něčím „potěšit“, snižujete tím riziko problémů

3. Rozdělte na rozumné části

- Ani málo, ani mnoho – optimálně desítky
- Terminologie – scénář, use case, ...
- Nedělitelný funkční celek, který buď je anebo není implementován
  - Nic mezi tím nemá smysl
- Slouží k měření postupu a plánování do iterací, případně ke škrtům při nedostatku času

4. Zapojení zákazníka

- Zapojte zákazníka do celého procesu co nejdříve a co nejvíce
- Snižuje rizika nedorozumění, odstraňuje budoucí třecí plochy
- Pozor, nemusí to být vůbec snadné
- Oboustranná otevřenost a důvěra jsou dobrým předpokladem k úspěchu

5. Prioritizace

- Klasifikujte jednotlivé scénáře/use case:
  - Minimálně – pracnost, důležitost, rizika
  - Nepřehánějte stupnici – např. A / B / C
- Přiřaďte iteracím podle svých potřeb:
  - Chci snížit rizika – nejdříve vysoce riskantní, málo pracné scénáře
  - Chci rychle něco ukázat – nejdříve málo pracné, vysoce důležité, málo riskantní
  - ...

6. Používání iterací

- Iterace = základní jednotka plánování
- Iterace umožňují korekci kurzu v definovaných okamžicích:
  - Stíháme termín?
  - Dodržíme rozpočet?
- Díky iteracím odhalíte rizika a problémy dříve
- Cca 1-3 iterace měsíčně
- Viz též „Zapojení zákazníka“

7. Důležité nástroje

- Minimálně byste měli mít:
  - Správu zdrojového kódu
    - Co kdo kdy změnil?
  - Správu pracovních položek („product backlog“)
    - Kolik je hotovo a kolik ještě chybí?
- Optimální je mít oba dva nástroje integrovány
  - Proč někdo tu změnu udělal?

8. Automatický build

- Prakticky ve všech moderních metodikách
  - Extrém – continuous integration (CI)
- Nenásilně zvyšuje disciplínu v týmu
  - Ukládám pouze kompilovatelný kód
- Lze využít i k dalším věcem:
  - Analýza kódu pomocí nástrojů
  - Spouštění testů (→ statistiky)
  - Měření změn (code churn)

9. Testování

- Z finančních důvodů často obětováno ☹☹☹☹☹
- Zvyšuje kvalitu software
- Brání chybám při pozdějších změnách (regression bugs)
- Pozor, testování nenahrazuje kvalitní vývojový proces !

10. Komunikace v týmu

- Metodiky neřeší
- E-mail je důležitý, ale není dostatečný
- Sdílejte informace, zvyšuje se tím pocit zodpovědnosti u členů týmu
- Konkrétní technologie závisí na vašich zvyklostech:
  - Portál – SharePoint Services, *Nuke apod.
  - Blog
  - WiKi
  - ...

Agenda

- Metodika
- Moje metodická doporučení
- Nástroje pro podporu životního cyklu vývoje
- Demo – TFS 2005

Modelování

- Používání CASE nástrojů je kontroverzní téma:
  - Fundamentalisté na ně nedají dopustit
  - Ale – kvalita generovaného kódu ☹
  - Nutnost úprav kazí čistotu i výhody
- Většina firem modeluje z důvodu analýzy, ne pro generování kódu
- Populární nástroje (za peníze):
  - IBM Rational SW Architect/Modeler (dříve Rose)
  - Enterprise Architect (Sparx Systems) – populární a levný
  - Borland Together

Správa požadavků

- Vyžadována pro velké projekty s důrazem na formální stránku procesu
- Většina firem / projektů nepoužívá
  - Nástroje jsou velmi drahé a hodnota je diskutabilní
- Nástroje:
  - CaliberRM (Borland)
  - Requisite Pro (IBM)

Správa zdrojového kódu

- Naprostá nezbytnost, využíváno drtivou většinou firem
- Enterprise nástroje:
  - ClearCase (IBM)
  - Team Foundation Server (Microsoft)
- Další nástroje:
  - SourceSafe (Microsoft), CVS, SVN, ...
- TIP – Používejte nástroje s úložištěm v relační databázi:
  - Jinak problémy s integritou

Pracovní položky

- Udržování seznamu scénářů, úloh, chyb, ... („product backlog“)
- Většina firem v nějaké formě používá
- První liga:
  - ClearQuest (IBM), Team Foundation Server (Microsoft), Star Team (Borland)
- Druhá liga:
  - Bugzilla, PVCS, ...
- Nouzová řešení:
  - Excel, Project, email, ...

Build

- Automatizace build procesu je používána velmi často
- Denní buildy rovněž běžnou praxí
- Nástroje:
  - Ant, NAnt (zdarma)
  - MSBuild (zdarma v .NET FX 2.0)
  - Team Build v Team Foundation Serveru (extenze MSBuild za peníze)

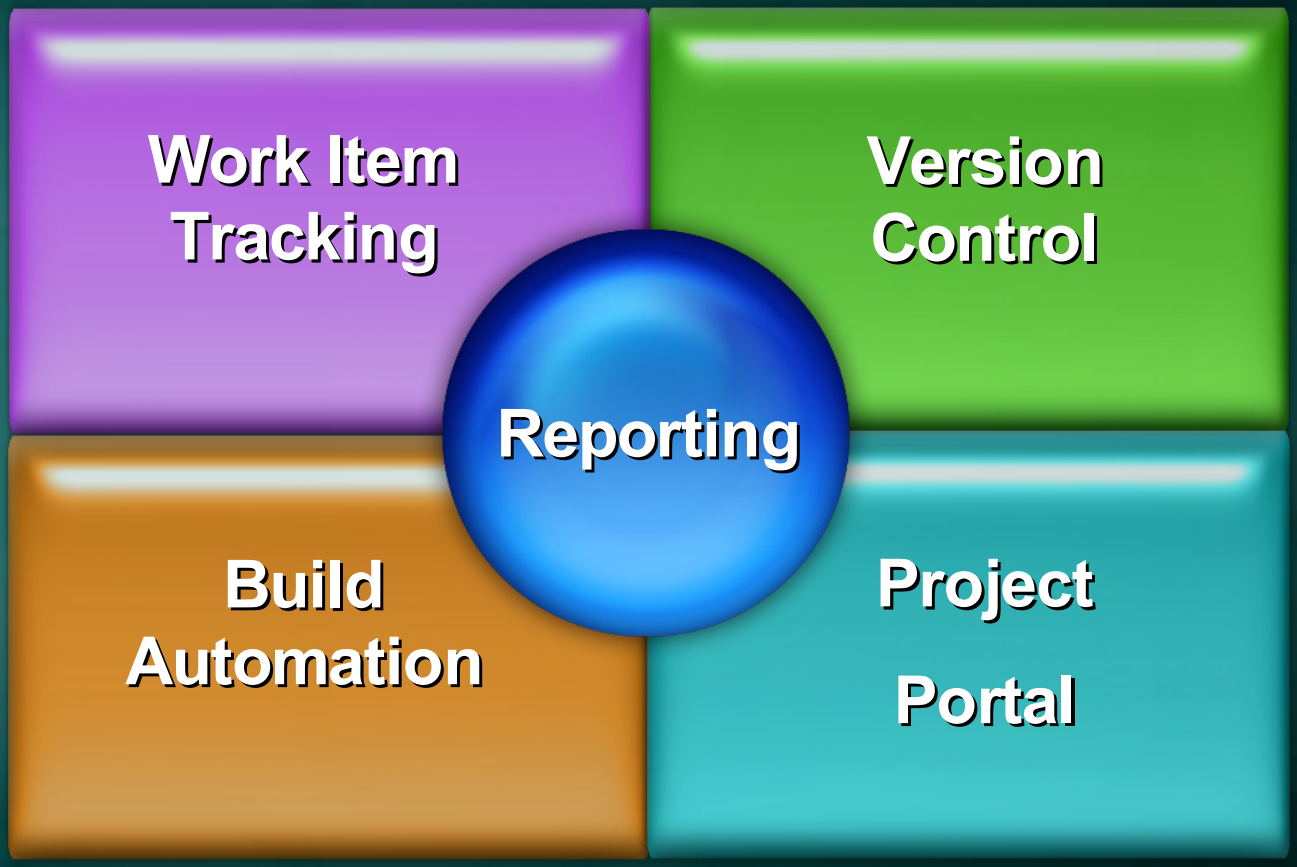
Testování

- Unit testy relativně časté, ostatní typy testů často opomíjeny kvůli vysoké ceně nástrojů a pracnosti
- Unit testy – kontrola funkce API
  - JUnit, NUnit, VS Team Dev/Test
- Funkční testy – simulace interakce uživatele:
  - VS Team Tester (Microsoft, pouze web), IBM/Rational (řada nástrojů), WinRunner+QuickTest, DevPartner+QACenter (Compuware)
- Výkonnostní/zátěžové testy:
  - VS Team Tester (Microsoft), LoadRunner (Mercury), IBM / Rational – řada nástrojů

Agenda

- Metodika
- Moje metodická doporučení
- Nástroje pro podporu životního cyklu vývoje
- Demo – TFS 2005

Team Foundation Server

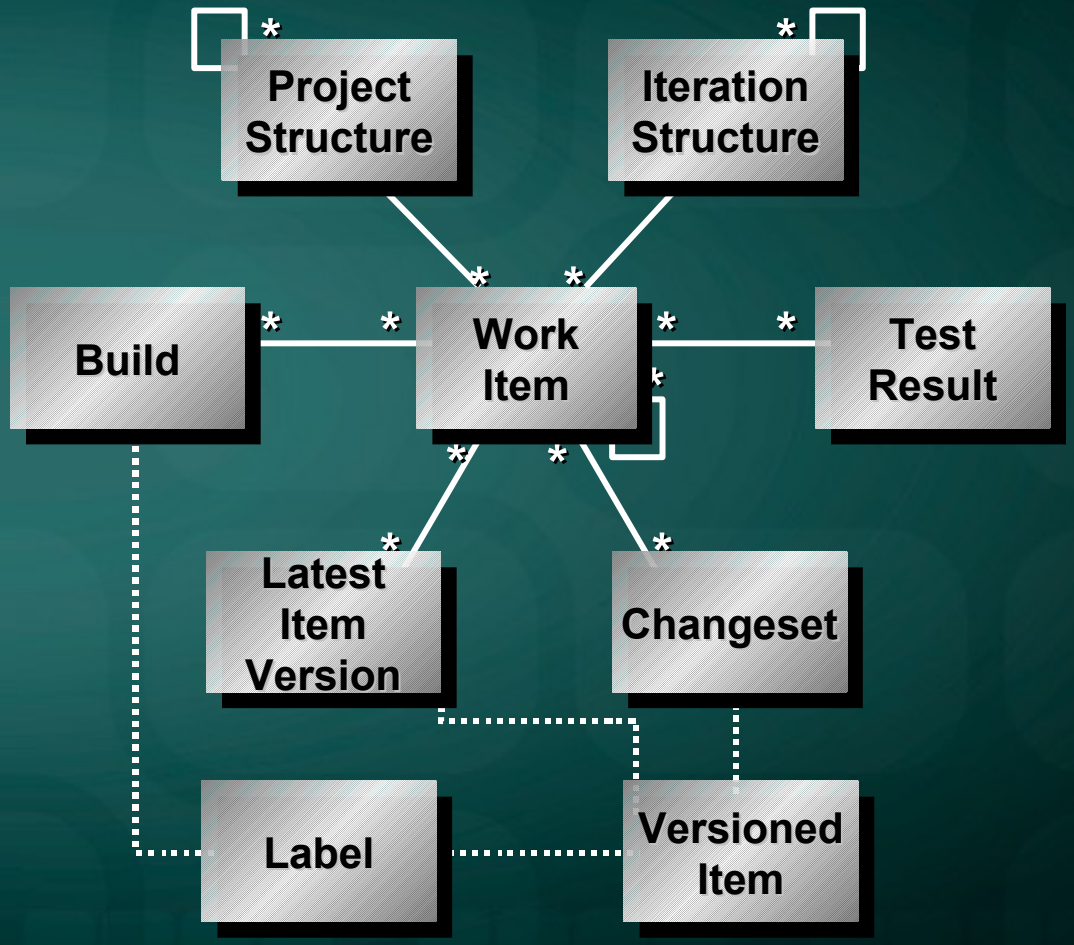


Microsoft Visual Studio 2005

Konkurenční výhody

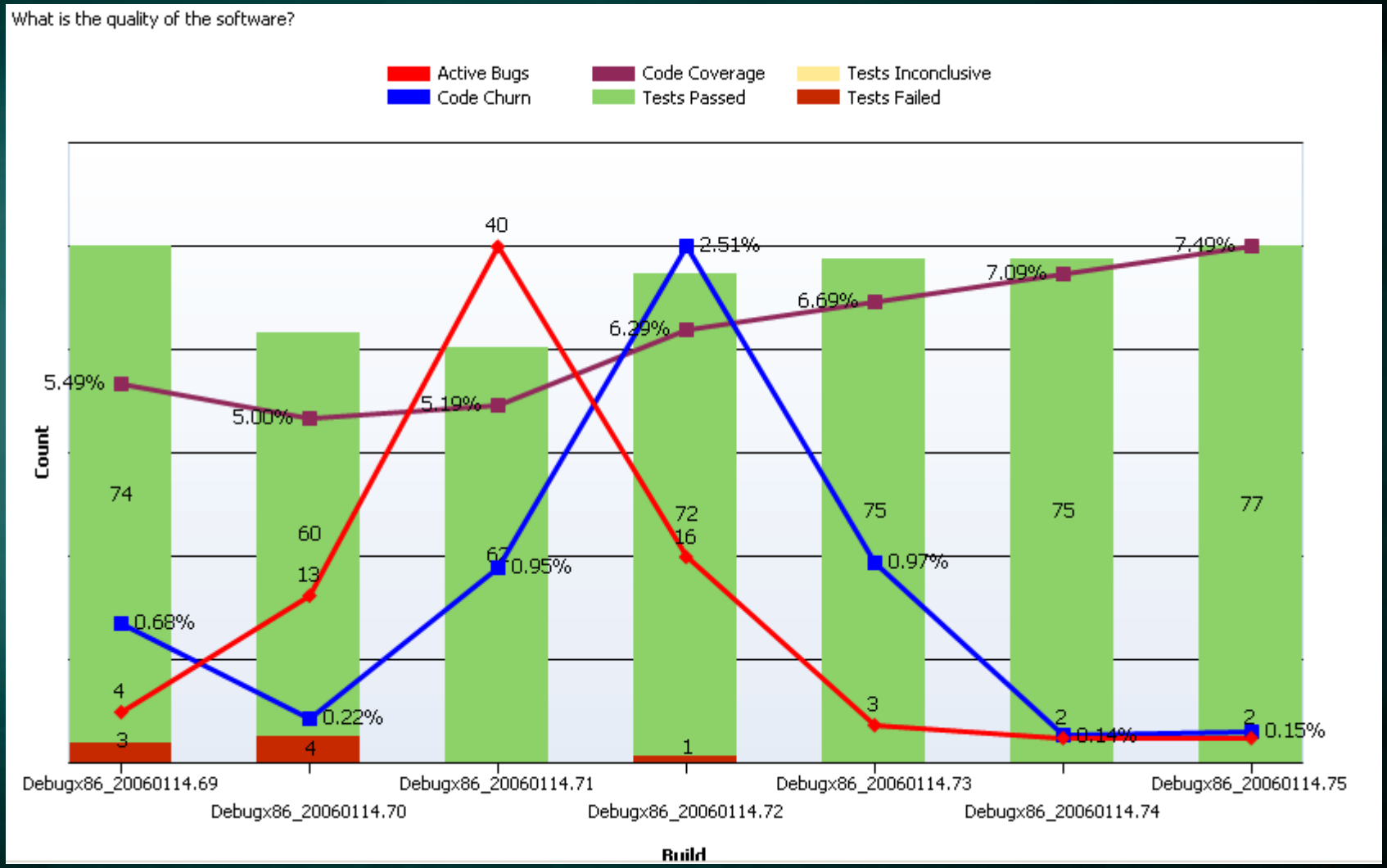
- Integrace:
  - Všechny části jsou vzájemně propojeny
  - Základ pro analýzu a reportování
  - 1 + 1 + 1 > 3
- Produktivita:
  - Vše přímo v prostředí Visual Studia
  - Minimalizace „obtěžování od práce“
  - „Žádné Alt-Tab“

Integrované reporty



Microsoft Visual Studio 2005

Integrované reporty



Microsoft Visual Studio 2005

Team Foundation Server

demo

- Automatický build
- Testy
- Work Items
- Správa kódu
- Reportování

Microsoft Visual Studio 2005

Závěrem

- Studujte dostupné metodologie, které jsou vám myšlenkově blízké a inspirujte se v nich
- Najděte rovnováhu mezi kreativitou lidí a formálností procesu akceptovatelnou vaší firemní kulturou
- Používejte integrované nástroje, ne jednotlivé části skládačky

mjurek@microsoft.com

Microsoft®

Your potential. Our passion.™