

# Qualität UND Geschwindigkeit

Werkzeuge im agilen  
Entwicklungsansatz

# Einige Fakten aus traditionellen Projekten

- Releasezyklus: 1 Jahr oder mehr
- 2 bis 4 Wochen für eine Integration
- Bekannte Fehler im ausgelieferten Produkt: >20
- Anzahl Unit Tests: < 100
- Dauer für Anforderungsänderung: mehrere Tage oder Wochen

# Einige Fakten aus agilen Projekten

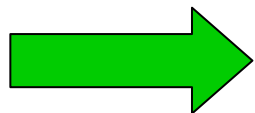
- Releasezyklus: minimal 30 Minuten, typisch 1 Monat
- Integrationen: 10 pro Tag oder mehr
- Bekannte Fehler im ausgelieferten Produkt: < 2
- Fehlerbehebungszeit: typisch < 4 h, max. 24 h.
- Anzahl Unit Tests: über 900
- Dauer für Anforderungsänderung: < 1 Tag
- Vollautomatischer Build inklusive Integration und Test
- Vollautomatisches Deployment
- Durchschnittliche LOC Klassen: < 150
- Durchschnittliche LOC Methoden: < 10

# Übersicht

- Determinanten der Softwareentwicklung
- Begriffe
- Gegenüberstellung Entwicklungsansätze
- Agile Entwicklungsansätze
- Agile Praktiken
- Organisation agiler Projekte
- Werkzeuge für den Agilen Ansatz
- (Rolle der Datenbank in Agilen Projekten)

# Determinanten der Softwareentwicklung

- Zeit
- Kosten
- Funktionsumfang (scope)
- Qualität



Drei legen den Vierten fest!

# Begriffe

- Qualität: „Quality is value to some person“ (Gerald Weinberg)
- Geschwindigkeit: Geschäftswert, der pro Zeiteinheit von einem SW-Entwicklungsteam bei geg. Qualität geliefert wird.
- Aber: Geschwindigkeit bedeutet auch: Fähigkeit, Änderungen zu berücksichtigen
  - Anforderungen
  - Randbedingungen

# Kundenfokus

- Kunden werden anspruchsvoller
  - Funktionalität
  - Verfügbarkeit
  - Preis
  - Qualität
- Mögliche Strategien
  - Verkürzung der Release-Zyklen
  - Outsourcing
  - Neuer Testansatz
  - Werkzeugeinsatz
  - Zusätzliche Entwickler

# Weitere Faktoren

---

- Managementsicht
- Unternehmensziele
- Branche
- Unternehmensgröße
- Problemdomäne

# Traditionelle Entwicklungsansätze

- „Monumentale Softwareentwicklung“ (Jim Highsmith)
- Langfristig
- Top-Down
- Analytisch, ingenieurmäßig

# Nachteile traditioneller Entwicklungsansätze

- Kunde bekommt, was er zu Beginn des Projekts wollte, nicht was er am Ende des Projekts braucht. -> Kunde unzufrieden.
- Geschäftsmodelle ändern sich schneller als es die Entwicklungsprozesse zulassen
- Zeithorizont für „strategische“ Entwicklungen verkürzt sich
- Abweichung vom Plan wird als Fehler verstanden

# Agile Entwicklungsansätze

- **Explorativ**
  - Abweichung vom Plan wird als Anpassung an neue Erkenntnisse begriffen
  - Strategie für Umgang mit Unsicherheit
- **Basieren auf**
  - Zusammenarbeit mit Kollegen, Kunden, Management, Lieferanten
  - Akzeptanz von Änderungen als Normalfall

# Agile Manifesto

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

# Übersicht Agile Entwicklungsansätze

- Capability Maturity Model (CMM)
- Unified Process (UP)
- Adaptive Software Development (ASD)
- Crystal
- Dynamic Systems Development Method (DSDM)
- Feature Drive Development (FDD)
- Lean Development (LD)
- Scrum
- Extreme Programming (XP)

# Einige XP-Praktiken

- Benutzergeschichten (User Stories)
- Planungsspiel
- Refaktorisierung
- Testgetriebene Entwicklung
- Paarweise Programmierung
- Kurze Iterationen
- Häufige Integrationen
- Schlichtes Design
- Kunde vor Ort

# Ein XP Projekt in der Praxis

- Charakteristika:
  - 5 Ingenieure
  - Verfügbarkeit 24x7x365
  - Weltweiter Einsatz
  - Web-Applikation (Customer Support Portal)
- Paarweises Programmieren
  - Testgetriebenes Programmieren
  - Refaktorisieren
  - Schlichtes Design
- Kurze Iterationen
- Häufige Integrationen
- Automatischer Build
- Automatisches Deployment

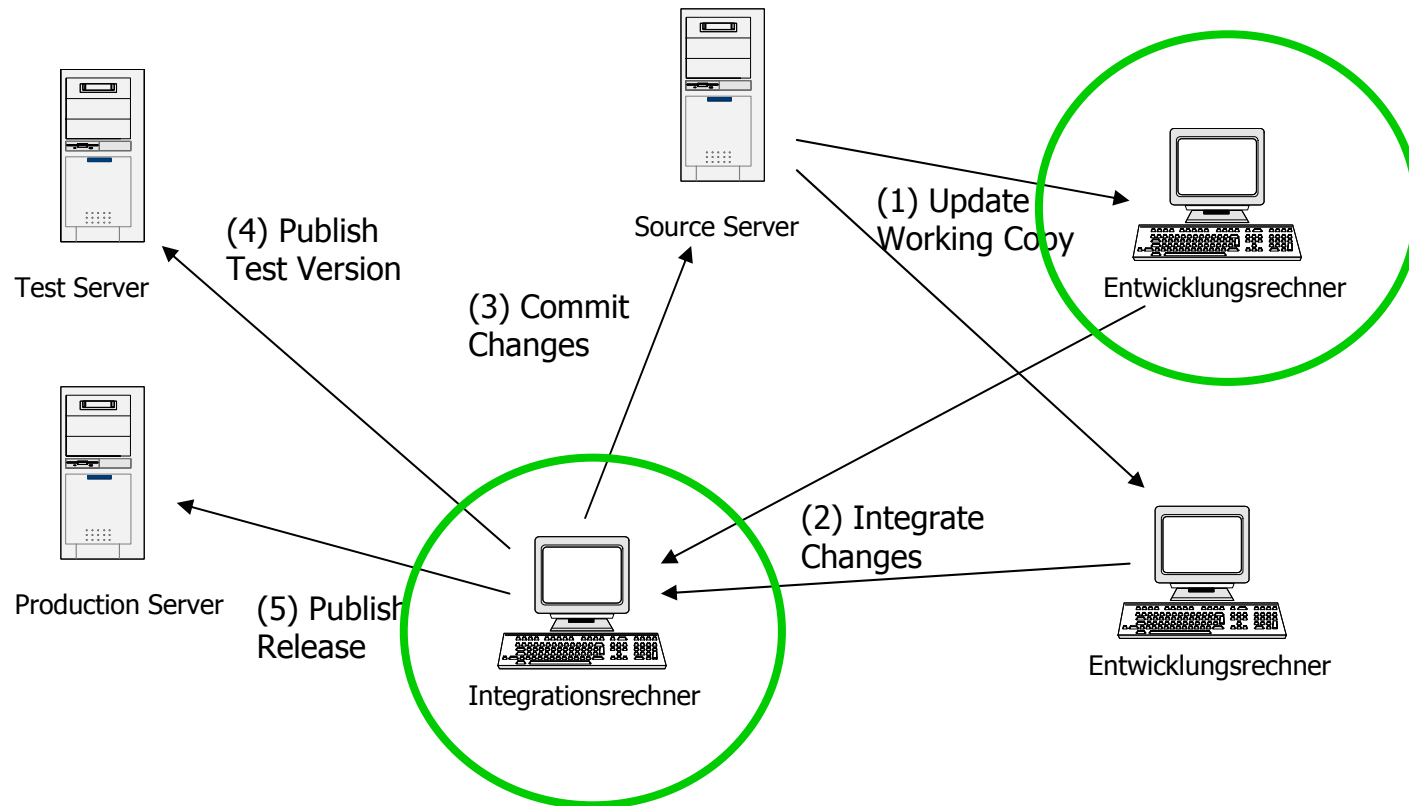
# XP: Zusammenarbeit mit dem Kunden

- Proxy User, z.B.
  - Program Manager
  - Customer Focus Group
- Applikation für User Story Management
  - Zugang für Customer Focus Group
  - User Stories sortiert nach Priorität
- Kommunikation
  - Face-To-Face
  - Telefonkonferenzen
  - Email
  - Instant Messenger

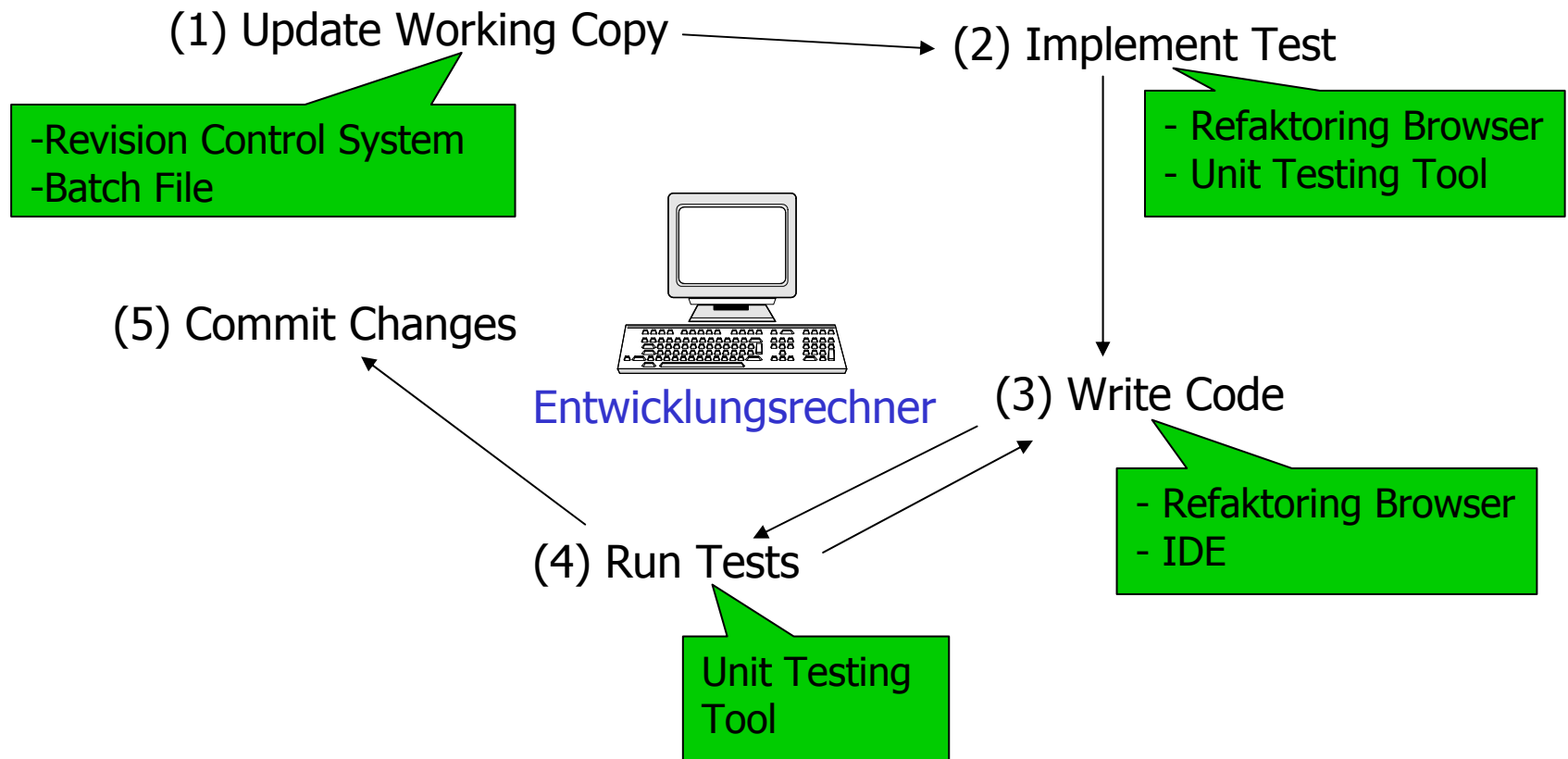
# XP: Zusammenarbeit mit dem Management

- Regelmäßig formloser Bericht via Email
  - pro Release, d.h. ca. 1x pro Monat
- Bericht enthält:
  - Abweichungen vom letzten Bericht
  - Neue Features in diesem Release
  - Geplante Features für nächstes Release
  - Geplantes Datum für nächstes Release
  - optional anstehende Probleme
  - Namentlicher Dank an alle Beteiligten
    - Tue Gutes und rede darüber!

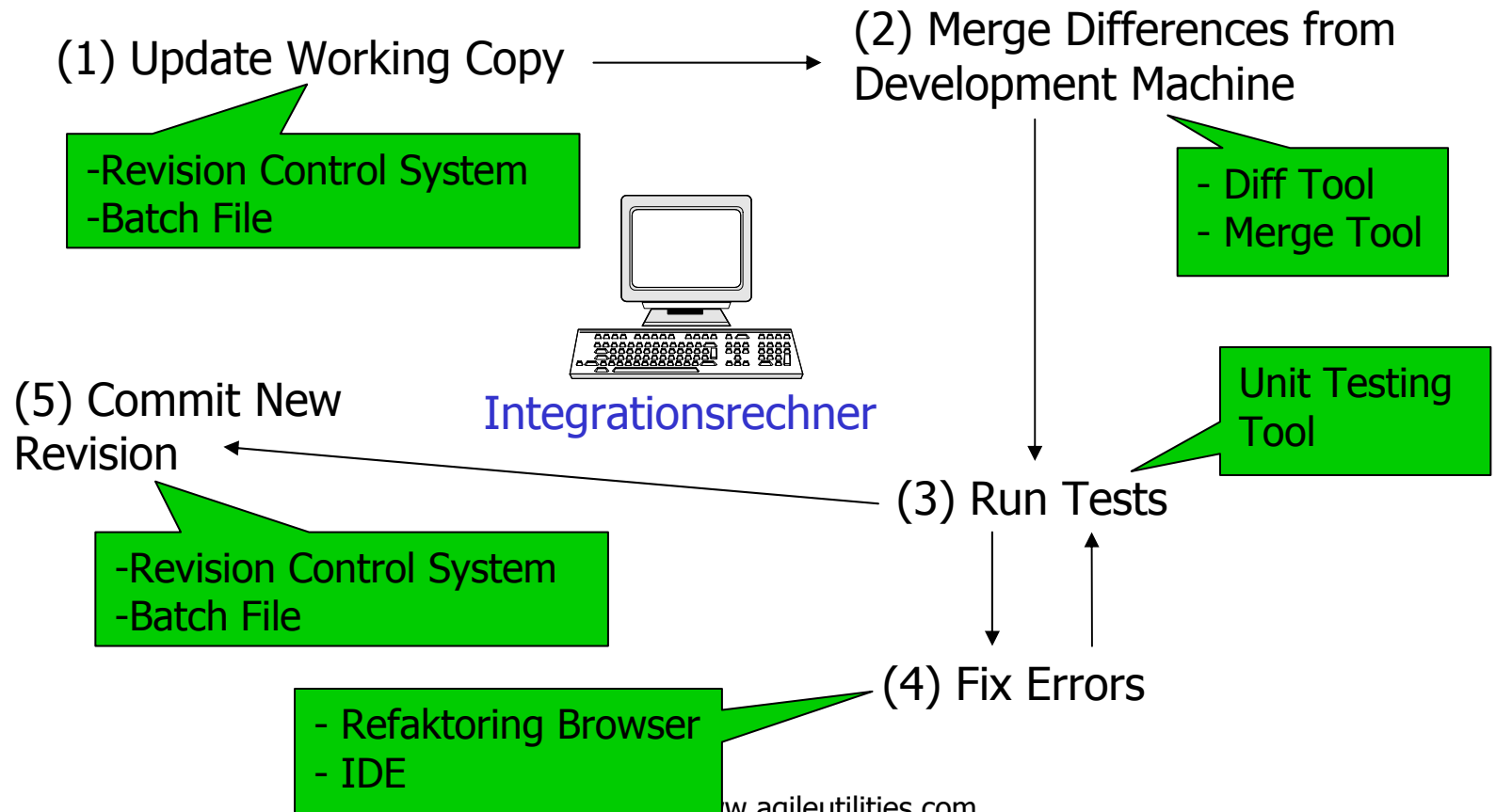
# XP: Überblick Programmierung



# Paarweises Programmieren



# Kurze Iterationen Häufige Integrationen



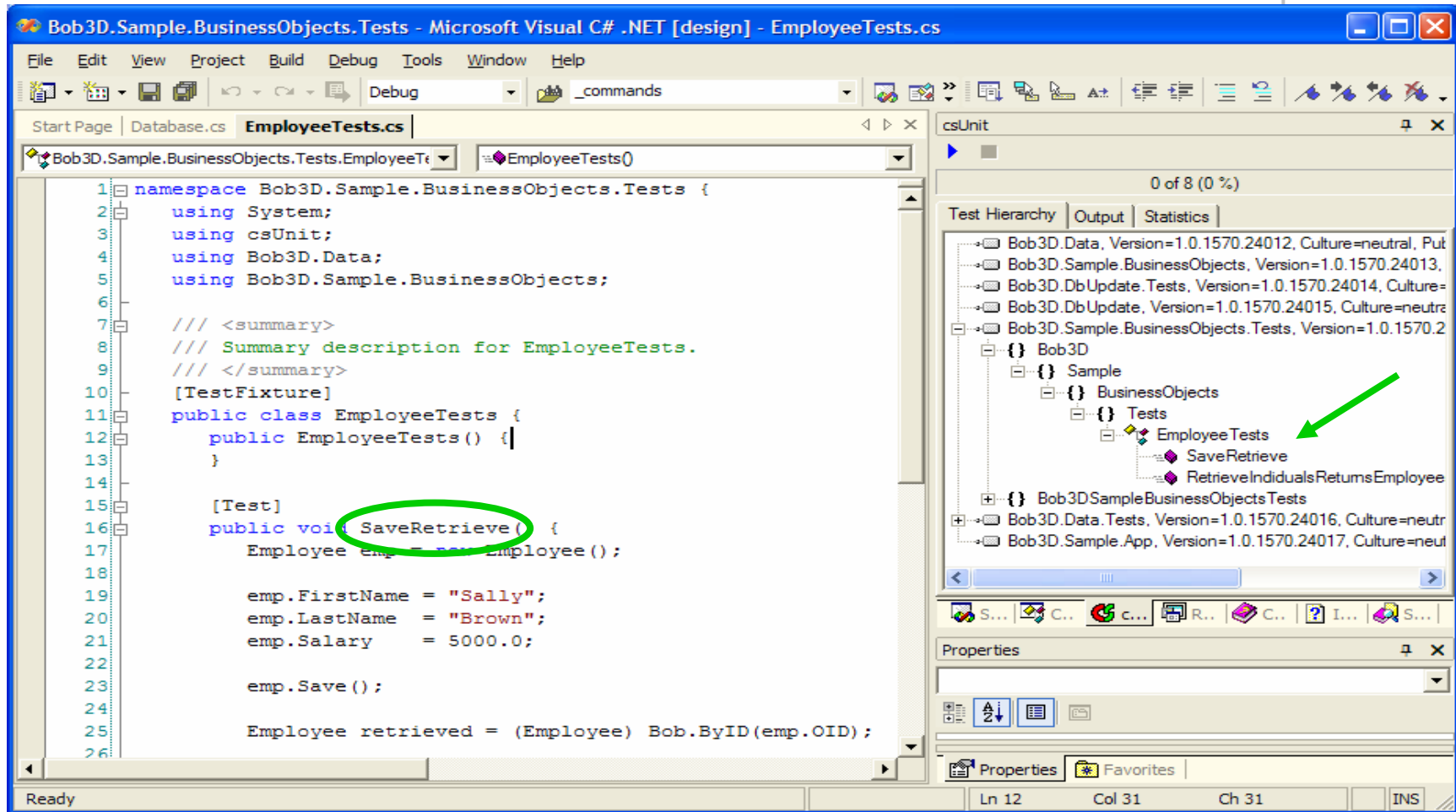
# Werkzeuge: IDE

- Integrierte Entwicklungsumgebung (IDE)
  - Editor
  - Debugger
  - Übersetzer (Compiler/Linker)
  - Browser
  - Refaktorisierungswerkzeug
- Open Source:
  - Eclipse
  - SharpDevelop

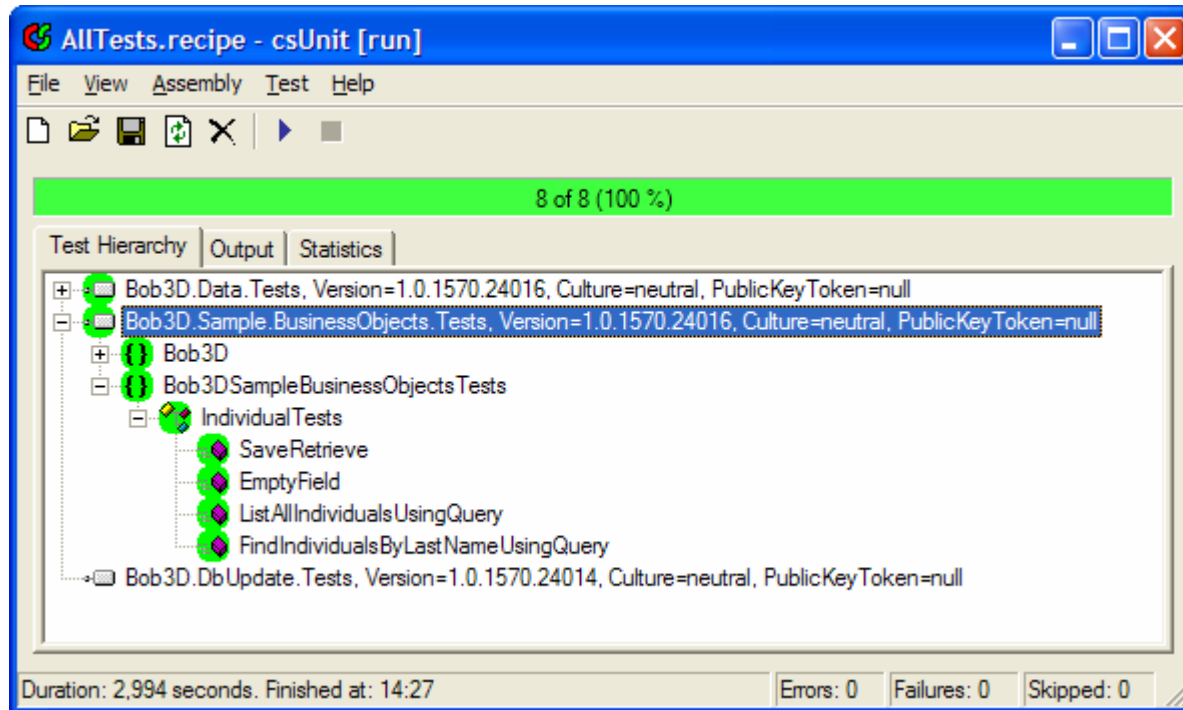
# Werkzeuge: Unit Testing

- **Wichtige Eigenschaften:**
  - Ausführung einzelner Tests
  - Information über Fehlerstelle
  - Einfachst zu bedienen
  - Integration in IDE
- **Open Source Produkte:**
  - JUnit ([www.junit.org](http://www.junit.org))
  - JWebUnit ([jwebunit.sourceforge.org](http://jwebunit.sourceforge.org))
  - FitNesse ([www.fitnesse.org](http://www.fitnesse.org))
  - csUnit (IDE-Integration in Beta, [www.csunit.org](http://www.csunit.org))
  - NUnit (derzeit keine IDE-Integration, [www.nunit.org](http://www.nunit.org))
  - Für weitere Sprachen: googeln nach xUnit wobei x für die Sprache steht.

# Beispiel: csUnit in IDE



# Beispiel: csUnit Stand-alone



# Werkzeuge: Revision Control Systeme

- **Wichtige Eigenschaften:**
  - Integration in IDE oder aber in File-Browser des Betriebssystems
  - Unterstützung konkurrierender Zugriffe
    - keine exklusiven Locks
- **Open Source Beispiele:**
  - CVS ([www.cvshome.org](http://www.cvshome.org))
    - TortoiseCVS (<http://sourceforge.net/projects/tortoise cvs/>)
  - Subversion ([subversion.tigris.org](http://subversion.tigris.org))
    - TortoiseSVN ([tortoisesvn.tigris.org](http://tortoisesvn.tigris.org))

# Werkzeuge: Architektur und Design

- Whiteboards (viele!)
- Microsoft Visio

# Werkzeuge: Anforderungen, Spezifikationen, Planung

- Karteikarten DIN A6, blanko
  - eventuell verschiedene Farben
- Pinnwände (viele!)
- Excel-Tabelle mit User Stories
  - sortiert nach Priorität
- Anwendung für User Story Verwaltung

# Werkzeuge: Generelle Anforderungen

- Preiswert, wenn möglich Open Source
- Werkzeuge gezielt einsetzen
- Manchmal genügt ein Batchfile!
- Kommandozeile für bessere Integration
  - Automatischer Build
  - Automatisches Deployment
  - Automatischer Test
  - Integration mit IDE
- Wichtig: „A fool with a tool is still a fool!“

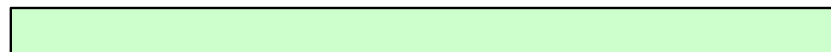
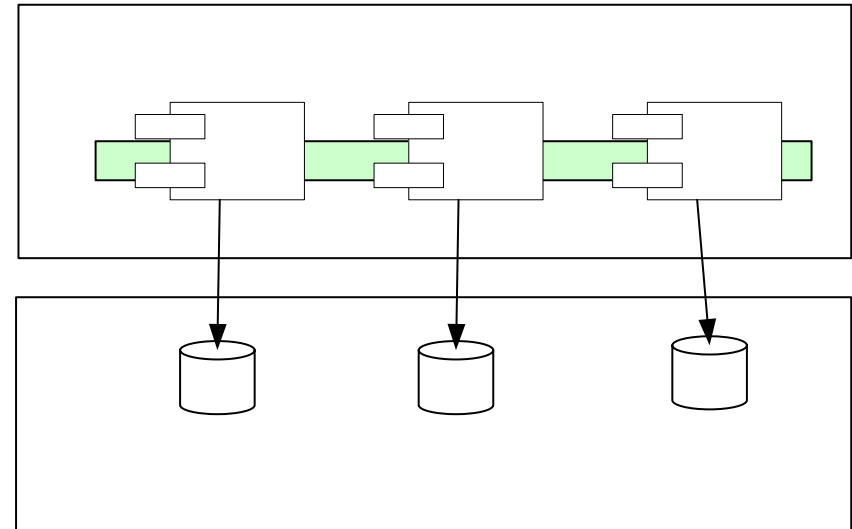
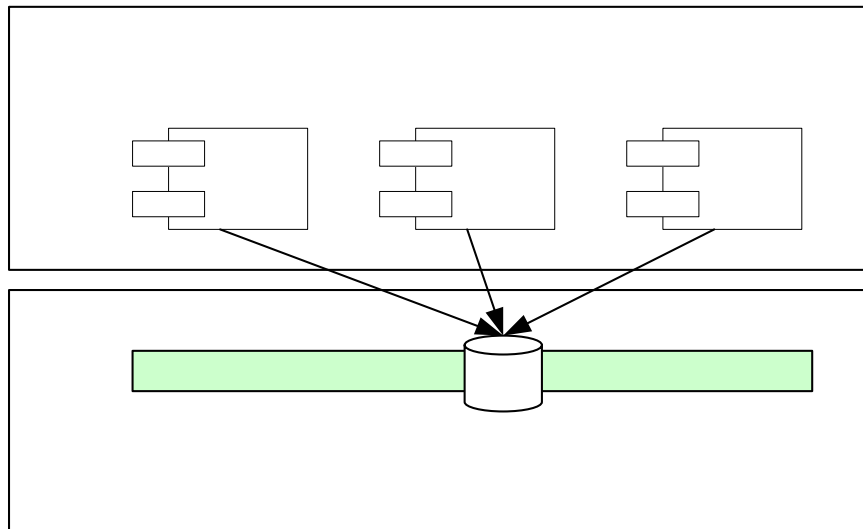
# Rolle der Datenbank in Agilen Projekten

- DB-Schemata sind häufig schwer zu ändern
  - Mehrere Applikationen müssen zeitgleich geändert werden
- Übernehmen viele Rollen:
  - Integration von Applikationen
  - Synchronisation von Applikationen
  - Kommunikation zwischen Applikationen

# Eine neue Rolle für die DB

- Fokussieren auf
  - Datenhaltung
  - Datenintegrität
- Andere Aufgaben auf andere Layer übertragen
  - Transaktionales Verhalten ist z.B. Eigenschaft des Business Objects

# Design für DB in neuer Rolle



# Vorteile

- Datenbankschema automatisch aus Businessobjektlayer generiert
- Reduzierte Dauer für Implementierung von Business Objekten
  - ca. 45 Minuten bis wenige Stunden inkl. Test, Integration
- Entwickler benutzt *eine* Programmiersprache
  - SQL hat untergeordnete Bedeutung
  - Keine Programmlogik mehr in Stored Procedures
- Entspricht dem Web Services Modell besser
- Kann erweitert werden um:
  - automatische Schemamigration für bereits benutzte Datenbanken

# Offene Fragen

- Reporting
  - Kann mit Read-Only-Account direkt auf Tabellen erfolgen
  - neue Reporting-Werkzeuge für verbundene Web Services
- Generierter SQL-Code möglicherweise nicht optimal
  - für 80% aller Anwendungen ausreichend?

# Einige Fakten aus traditionellen Projekten

- Releasezyklus: 1 bis 3 Jahre
- 2 bis 4 Wochen für eine Integration
- Bekannte Fehler im ausgelieferten Produkt: >20
- Anzahl Unit Tests: < 100
- Dauer für Anforderungsänderung: mehrere Tage oder Wochen

# Einige Fakten aus agilen Projekten

- Releasezyklus: minimal 30 Minuten möglich, typisch 1 Monat
- Integrationen pro Tag: 10
- Bekannte Fehler im ausgelieferten Produkt: < 2
- Fehlerbehebungszeit: < 4 Stunden
- Anzahl Unit Tests: über 900
- Dauer für Anforderungsänderung: < 1 Tag
- Vollautomatischer Build inklusive Integration und Test
- Vollautomatisches Deployment
- Durchschnittliche LOC Klassen: < 150
- Durchschnittliche LOC Methoden: < 10

# Literatureempfehlungen

- Jim Highsmith: „Agile Software Development Ecosystems“
- Gerald Weinberg: „Quality Software Management“
- Kent Beck: „Extreme Programming Explained“
- Martin Fowler: „Refactoring“
- Ken Schwaber, Mike Beedle: „Agile Software Development with Scrum“
- Alistair Cockburn: „Agile Software Development“
- Mike Cohn: „User Stories Applied For Agile Software Development“

# Vielen Dank!

- Fragen
- Kontakt
  - [ml@agileutilities.com](mailto:ml@agileutilities.com)
  - <http://www.agileutilities.com>
  - Manfred Lange  
Rechbergweg 24  
71131 Jettingen