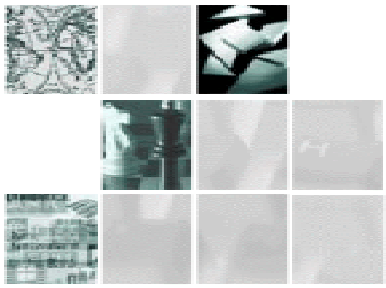
# Agile Entwicklung unter schwierigen Projektbedingungen



**oder:**
***Extreme Programming in Projekten unter extremen Bedingungen***

**Alexander van der Vekens**
Senior Consultant
avekens@logicline.de






## Überblick

- Was sind Projekte mit extremen Bedingungen?
  - Allgemein
  - Konkretes Beispiel
- Welche Techniken des XP wurden eingesetzt?
- Welche Techniken des XP wurden nicht eingesetzt?

## „extreme“ Bedingungen – allgemein (1)

- **Enger Zeitplan, Projektstart von heute auf morgen**
  - Endtermin steht lange fest, Entscheidung für das Projekt zieht sich in die Länge
- **Begrenztes, knappes Budget**
- **Sehr spezifische Funktionalitäten**
  - Nur deshalb wird das Projekt überhaupt durchgeführt
- **„angeblich“ ausreichend detailliertes Pflichtenheft**
  - Es muss deshalb „nur“ realisiert werden
- **Kunde stellt Ansprechpartner**
  - Es stellt sich jedoch heraus, dass er nur Vermittler zu den eigentlichen Anwendern ist







## „extreme“ Bedingungen – allgemein (2)

- **Architektur, Prototyp bereits vorgegeben**
  - Beides aber nicht brauchbar bzw. nicht ausreichend
- **Funktionsumfang steht unabänderlich fest**
- **Zu grobe Aufwandsabschätzungen**
  - Zu wenig Zeit
  - Keine ausreichende Einarbeitungszeit
  - Keine ausreichenden Fachkenntnisse

## Beispiel für ein Projekt mit extremen Bedingungen

- Offshoring andersherum – Kunde/Auftraggeber im Nahen Osten
  - Ab und zu Flüge in die Sonne ☺
  - Absprachen sonst nur per eMail und Telefon
- Keine übergreifende Projektinfrastruktur, sehr schwacher Internetzugang beim Kunden
- Eigenentwicklung des Auftraggebers gescheitert
  - Wahrung des „Gesichts“ des Auftraggebers gegenüber Endkunden







## Projekt Charakteristika (1)

- System zur Einwohnerverwaltung eines arabischen Landes (Registrierung von Geburt, Tod, Heirat, Scheidung, Einwanderung, Arbeitserlaubnis, Führerschein; Nutzung von Chipkarten und biometrischer Daten wie digitales Foto, eingescannte Unterschrift und Fingerabdrücke)
- Einführungstermin fix (Geburtstag des Staatsoberhaupts); Budget gedeckelt („Schätzpreis“)
- J2EE Anwendung mit Web Client
- Verteilte Entwicklung:
  - Erstellung des zentralen Verwaltungssystems (Geschäftslogik, Datenhaltung, Schnittstellen zu anderen Systemen) in Deutschland durch Firma LogicLine
  - GUI (JSPs, HTML) durch indische Entwickler vor Ort beim Kunden
  - andere Systeme (Chipkarten, Biometrie) in Frankreich durch Firma Gemplus

## Projekt Charakteristika (2)

- Team in Deutschland: 4-5 Leute (2-3 Entwickler, 1 Architekt, 1 Projektleiter); QS und Anforderungsmanagement durch Architekt und Projektleiter
- Laufzeit (des Teilprojekts): 4 Monate
- Tools/Techniken: J2EE (JSPs, Servlets, Stateless Session EJBs), Struts, WSAD, CVS, log4J, JUnit, WebServices, JDBC, Oracle DB







## Erfolgsfaktoren für ein Projekt unter extremen Bedingungen

- Kleines, kompetentes, motiviertes, eingespieltes Team
- Jeweils genau ein Ansprechpartner auf beiden Seiten (Auftraggeber/Auftragnehmer)
- Permanentes Fortschreiben und Abstimmen des Pflichtenhefts
- Fließende und wechselnde Rollen innerhalb des Teams
- Einfacher Entwicklungsprozess mit einfachen Tools
- ... und natürlich XP-Techniken!

## Was ist „agil“? — Definition

- **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:

1. **Individuals and interactions over processes and tools**
2. **Working software over comprehensive documentation**
3. **Customer collaboration over contract negotiation**
4. **Responding to change over following a plan**







## Was ist „agil“? — Persönliche Sicht

- **Persönliche Sicht:**
  - (1) Vorgehen steht nicht von Anfang an fest, sondern entwickelt sich während des Projekts
  - (1) Schnelle und flexible Änderungen des Vorgehens bei Bedarf
  - (1) Minimale Vorgaben (an den Entwicklungsprozess und –tools)
  - (4) „Ad hoc“-Änderungen des Projektplans
    (z.B. Abänderung der an Mitarbeiter zugewiesenen Aufgaben)
  - (4) Flexible Arbeitszeiten, flexible Rollenverteilung
  - (3) Ständige Absprachen mit Kunden/Auftraggebern, Dokumentation der Ergebnisse
  - (2) Unabhängigkeit jedes Mitarbeiters
    (im Extremfall ist der Code die einzige zentrale Ressource!)
- ***... Aber: alles muss kontrolliert ablaufen, ansonsten entsteht Chaos***

## Vorzüge flexiblen Vorgehens

- Man kann sofort loslegen – kein Zeitverlust
- Keine hohen Kosten für Infrastruktur – schnelles ROI
- Korrektive Maßnahmen immer möglich – schnelle Reaktion auf Probleme
- Minimierung der Abhängigkeit von einzelnen Mitarbeitern – alle können alles
- Freiheiten des Mitarbeiters – Motivation, Bereitschaft zur Mehrarbeit

- **Oft ist ein agiles Vorgehen im strengen Sinne nicht möglich**
- **Trotzdem soll und kann nicht auf die Vorzüge agiler Verfahren verzichtet werden**







## Eingesetzte XP-Techniken (1)

***(Bezug zu „The Rules and Practices of Extreme Programming“)***

- **Pair Programming (*„All production code is pair programmed“*):**
  - Nützlich bei neuen, komplexen Sachverhalten und Fehlersuche („Pair Debugging“)
  - Hervorragend zur Einarbeitung neuer Teammitglieder
  - Wenig effizient bei Routinetätigkeiten
- **Gemeinsamer Code (*„Use collective code ownership“*):**
  - Absolut notwendig! Es müssen sofort Änderungen vorgenommen werden können, da keine Zeit für aufwändige Abstimmungsprozesse vorhanden ist
  - Trennung zwischen stabilem und in Arbeit befindlichem Code
  - (automatische) Vergleichbarkeit von Versionen unbedingt notwendig (Tool wie CVS, Codierrichtlinien) – wer hat wann (und insbesondere warum!) was geändert

## Eingesetzte XP-Techniken (2)

- **Unit Tests:**
  - Ebenfalls absolut notwendig, auch unter extremen Bedingungen:
    - ***„All code must have unit tests“***
    - ***„All code must pass all unit tests before it can be released“***
    - ***„When a bug is found tests are created“***
  - Sofortige Verifizierung der Implementierung (und besseres Verständnis der Anforderungen)
  - Durch regressionsfähige Tests werden unerwünschte Nebeneffekte früh entdeckt
  - Hilfe bei Fehlersuche (Erweiterung der Testfälle gemäß Randbedingungen des Fehlerfalls)
  - Ziel: Erreichung einer hohen Code-Abdeckung







## Eingesetzte XP-Techniken (3)

- **Häufige Integration (*„Integrate often“*):**
  - Am Ende einer Phase mehrmals täglich
  - Gute Toolunterstützung notwendig (automatische Builds/Deployments)
  - Nur sinnvoll, wenn auch ständig getestet wird
- **Iteratives Vorgehen (*„The project is divided into iterations“*)**
  - Konkrete Ziele, Erreichtes sichern, korrektive Maßnahmen möglich
  - Termine und Ziele müssen jedoch am Anfang fest vorgegeben sein
  - Dauer einer Iteration zwischen 1 und 3 Wochen realistisch (und notwendig)
- **weitere XP-Techniken:**
  - *„Make frequent small releases“*
  - *„Move people around“*
  - *„Code must be writen to agreed standards“*

## XP-Praktiken, die nicht eingesetzt wurden (1)

- **Ständige Verfügbarkeit des Kunden (*„The customer is always available“*)**
  - Realisierer sind „nur“ Subunternehmer
  - Auftraggeber konnte nicht vor Ort (der Realisierung) sein, da alle Ressourcen vor Ort beim Kunden benötigt wurden
  - Zeitverschiebung (4 Stunden): „Normale“ Arbeitszeit überlappte sich nur wenige Stunden
  - Muslimische Woche: Wochenende am Donnerstag und Freitag
- **User Stories (*„User stories are written“*)**
  - Das Pflichtenheft lag vor, enthielt bereits „Use Cases“ – ein Umschreiben hätte zu viel Zeit/Aufwand gekostet







## XP-Praktiken, die nicht eingesetzt wurden (2)

- **Häufiges Refaktorisieren (*„Refactor whenever and wherever possible“*)**
  - Reibungsverluste hoch, da evtl. am betroffenen Code von anderen gearbeitet wird
  - Schlechte Nachvollziehbarkeit von Änderungen
  - In kleinem Rahmen ja – in größerem Stil nur, wenn unbedingt nötig
- **Keine Überstunden (*„No overtime“*)**
  - Ha, ha, ha – ohne Pizzadienst und viel Kaffee wären pünktliche Auslieferungen (meistens um 23:55 eines Termins) nicht zu schaffen gewesen!
  - Nach einem „extremen“ Projekt sollte man Urlaub machen oder für einige Zeit einer weniger aufreibenden Tätigkeit nachgehen
- **Tägliche Meetings (*„A stand-up meeting starts each day“*)**
  - Zu aufwändig, unterschiedliche Arbeitszeiten
  - Anfangs einmal wöchentlich, am Ende einer Iteration zweimal wöchentlich, ansonsten ist Projektleiter der „Kommunikator“ (pendelt zwischen den Entwicklern)

## XP-Praktiken, die nicht eingesetzt wurden (3)

- **Flexible Release-Planung (*„Release planning creates the schedule“*)**
  - Reihenfolge vorgegeben durch Abhängigkeiten
  - Kunde hat keinen Nutzen aus Teilsystemen!
  - Keine echten Change Requests vorgesehen
    - Wurden anfangs zugesichert und hätten vergütet werden sollen
    - Änderungen an der Spezifikation waren später jedoch nur „Verfeinerungen“
- **Test First (*„Code the unit test first“*)**
  - Anwendung zu GUI- und DB-lastig
  - Aufbau einer geeigneten Infrastruktur in der Kürze der Zeit nicht möglich
  - Frühe Zwischenauslieferungen erwartet
  - Entwickler nicht daran gewöhnt
  - Stattdessen: „Code the unit test immediately after ...“







## Fazit

- Projekte unter extremen Bedingungen erfordern extreme Flexibilität
- Flexibilität bezieht sich auch auf den Einsatz agiler Methoden – es muss flexibel zwischen agilen und konventionellen (und teilweise auch chaotischen) Techniken gewechselt werden können
- Auch XP ist flexibel anpassungsfähig und sein Einsatz optimierbar (*„Fix XP when it breaks“*) – in unterschiedlichen Projekt-Kontexten
- Viele Kunden sind (noch) nicht reif für agile Prozesse:
  - Bestehen auf von vornherein festgelegte Funktionalitäten
  - (niedrige) Festpreise => Abstriche bei der Qualität
  - Zu spätes Einbeziehen des Auftragsnehmers in die Entscheidungsfindung