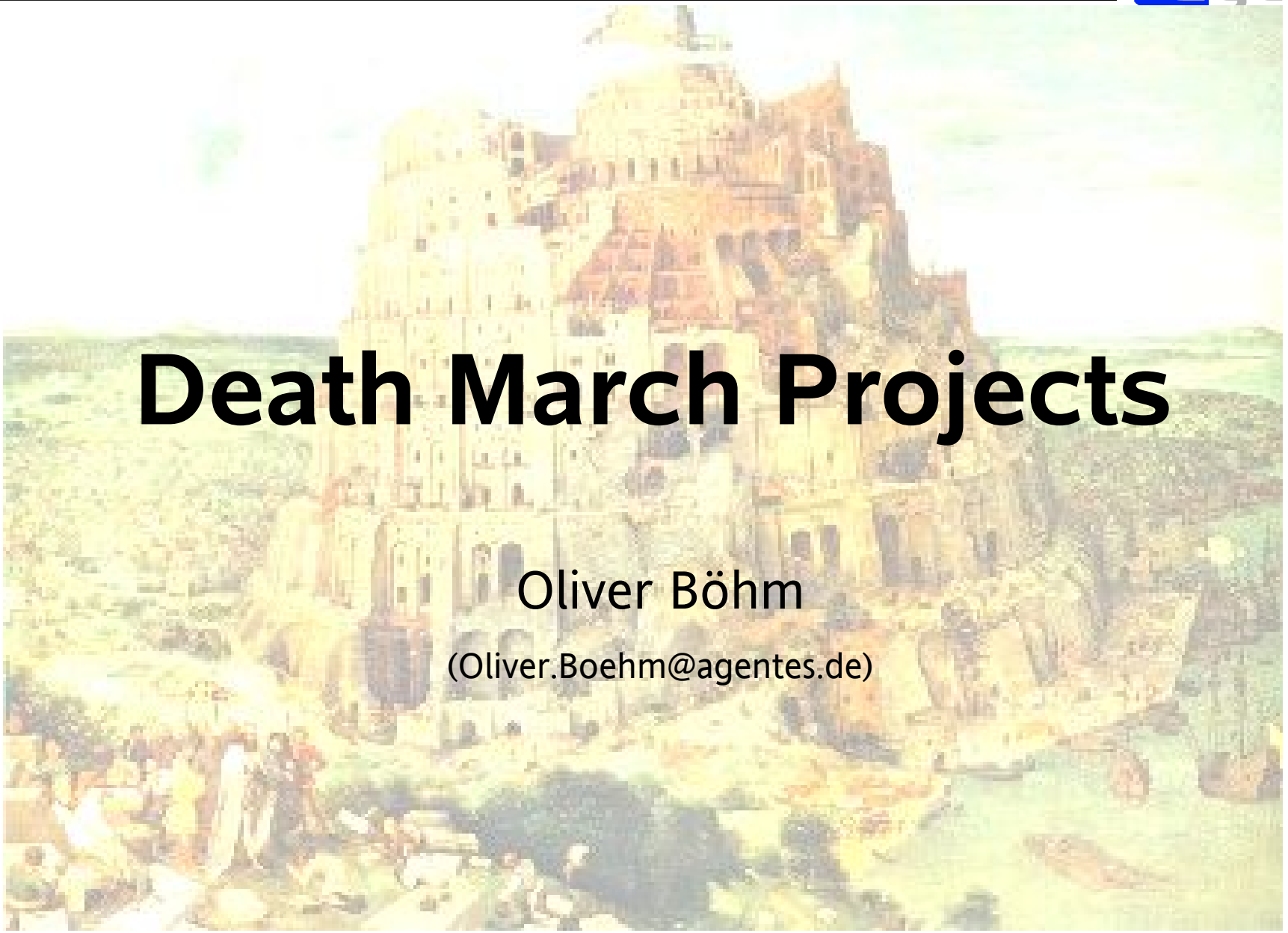


Death March Projects

Oliver Böhm

(Oliver.Boehm@agentes.de)

Symptome

- Zeitplan halbiert
- Mannschaft halbiert
- Budget / Resourcen halbiert
- doppelt so viele Features

Gründe

Politik, Politik, Politik

„Start-up“-Mentalität junger Unternehmen

Konkurrenzdruck durch neue Technologien

Naive Versprechungen

„Marine-Corps“-Mentalität („*Real* programmers don´t need sleep“)

unerwartete gesetzliche Regelungen

Fehlende Erfahrung („we can do it over the weekend“)

Konkurrenzdruck durch Globalisierung des Marktes

unerwartete / ungeplante Krisen

Beweggründe für die Teilnahme

Viel Feind, viel Ehr	Jugendliche Naivität und Optimismus	Schlechte Alternativen
Das „Mt. Everest“-Syndrom	Keine Alternativen	Flucht vor „normaler“ Bürokratie
Enthusiasmus	Investition in die Zukunft	Rache

What to do?

(Kevin Huigens)

- Run!
- Keep your eyes open
- Make sure you can trust all of your co-workers
- Realize the developers aren´t your enemy, the managers are
- Communicate. Communicate. Communicate.
- Keep the team small
- Keep the team intact
- Review the design
- Make sure testing plan is ready when it's time to test
- Make sure you have a test plan.
- Don´t rush to code
- Keep documentation updated and current
- Everyone should have access to documentation
- Have regular weekly progress meetings
- All code works before you leave at the end of the day
- Make sure everyone understands what they´re doing

What *not* to do?

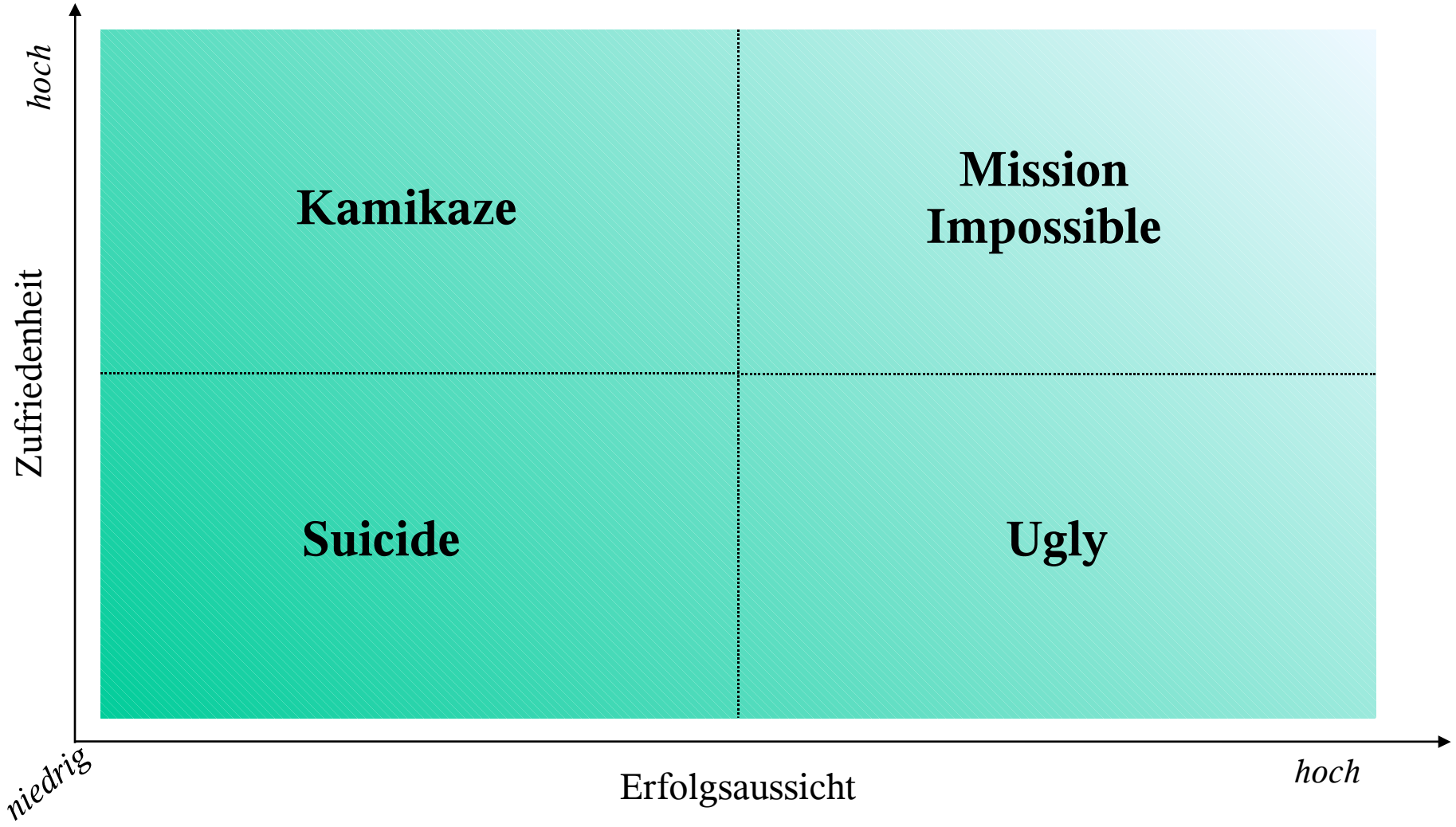
(Kevin Huigens)

- Don´t plan a wedding
- Don´t have unclear areas of responsibility
- Don´t allow design changes lightly
- Don´t become irritated or angry
- Don´t rely too heavily on 1 member of the team
- Don´t assume team members understand the entire project
- Don´t underestimate
- Don´t refrain from asking questions when you don´t understand
- Don´t start the project if haven´t got the money to finish
- Don´t commit to unreasonable dates
- Don´t let meetings last > 1.5 hours
- Don´t be afraid to let management know you need something
- Don´t be afraid to stand up to management
- Don´t accept gospel info from so-called experts
- Don´t forget that managent doesn´t understand how to develop software

Political Players

- Owner (Besitzer)
- Customer (Kunde)
- Shareholder (Aktionär)
- Stakeholder (Anteilseigner)
- Champion (Schutzherr)

Project Style Quadrant



Verhandlungs-Spiele

- Double and Add Some
- Reverse Doubling
- „Guess the number I´m thinking of“
- Double Dummy Split
- Spanish Inquistion
- Low Bid
- Gotcha
- Chinese Water Torture
- Smoke and Mirrors
- Hidden Variables of Maintainability / Quality

Teamzusammenstellung

Superstars anheuern

Mehr Leute auswählen und ihnen deutlich machen, auf was sie sich einlassen

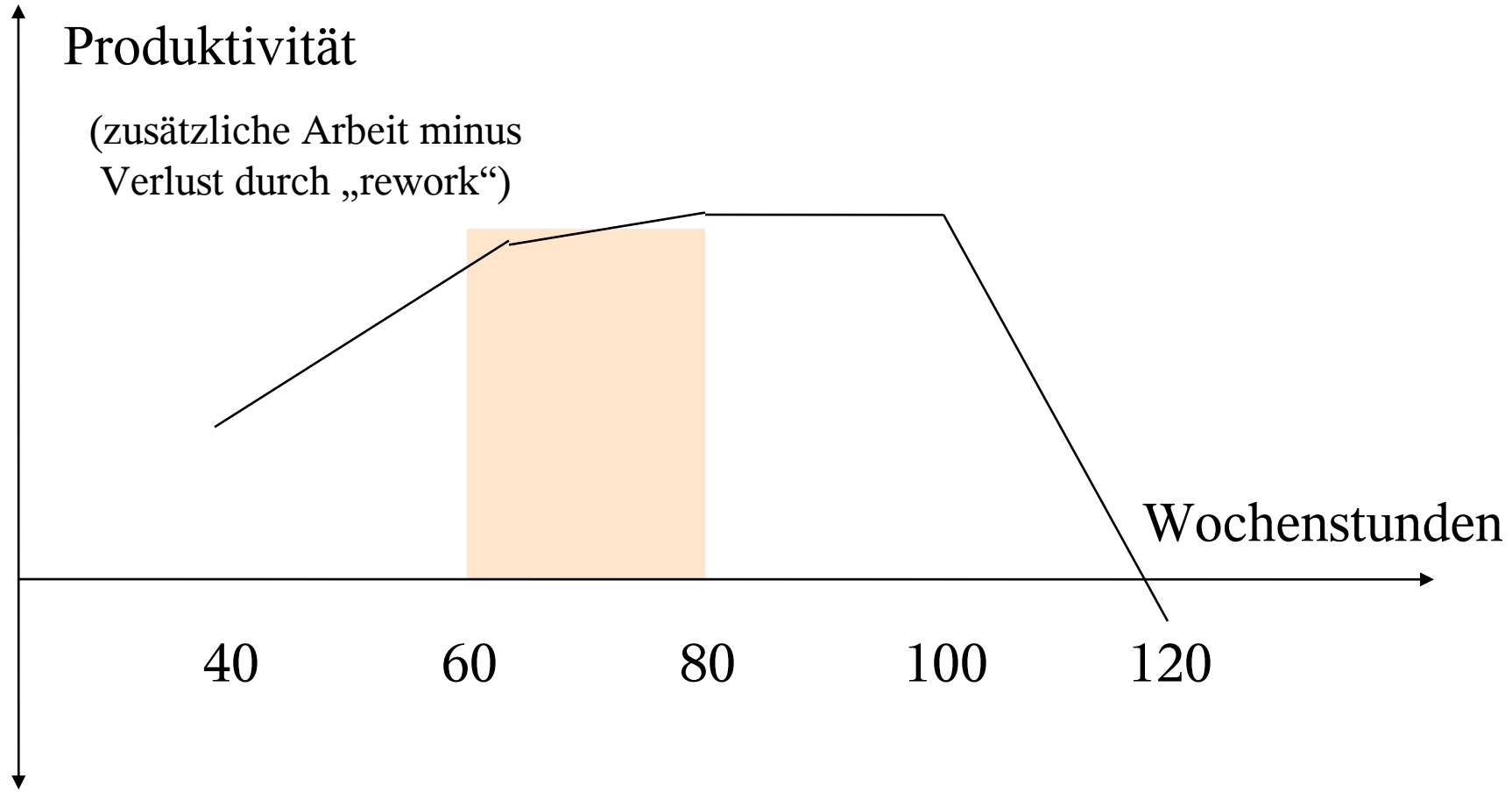
Übernahme eines „Mission Impossible“-Team, das gerade frei wird

Jeden übernehmen, den man bekommen kann

Rewards

- Geld
  - aber: Geld ist „Hygiene-Faktor“ (keine zusätzliche Motivation)
- Extra Urlaub
- „Erholungs“-Projekt
  - Kurs
  - Beschäftigung mit neuen Technologien
  - ...
- Hardware
- Blumen für die Familie

Überstunden



Kommunikation

Keine Geheimnisse

Reger Gedankenaustausch

Stammtisch

Hindernisse, Todsünden

- Defensives Management (kein Vertrauen)
- Bürokratie
- Physikalische Trennung der Team-Mitglieder
- Zersplitterung der Arbeitszeit
- Verringerung der Qualität
- utopische Zeitvorgaben
- Umorganisation; Auflösung der Mannschaft nach Projekt-Ende

Arbeitsplatzumgebung

Die Produktivität in einer “ungestörten” Arbeitsumgebung ist 2,6 mal höher als in einer “normalen” Büro-Umgebung

Tom DeMarco and Tim Lister, *Peopleware* (Dorset House Publishing, 1987)

Verbesserung der Arbeitsumgebung

- Frontal attack
- “skunk works” mystique
- Squatter´s rights
- Telecommute
- Switch to the graveyard shift
- Barricades and buffers

Das “Triage”-Konzept

Requirements aufteilen in

- “must-do”
- “should-do”
- “could-do”

Requirements Management

- Requirements Management ist wichtig!
- Wichtig ist aber auch, Änderungen zu den Original-Anforderungen zu dokumentieren!

“Principal Best Practices” (Airlie Council)

- Formal Risk Management
- Agreement on Interfaces
- Peer Reviews
- Metric-based Scheduling and Management
- Binary Quality Gates at the “Inch-Pebble” Level
- Project-wide Visibility of Project Plan and Progress
- Defect Tracking against Quality Targets
- Configuration Management
- People-aware Management Accountability

“Breathalyzer Test” (Airlie Council)

- Do you have a current, credible activity network supported by a Work Breakdown Structure (WBS)?
- Do you have a current, credible schedule and budget?
- Do you know what software you are responsible for delivering?
- Can you list the top ten project risks?
- What is the estimated size of your SW deliverable? How was it derived?
- Does your staff have sufficient expertise in the project domain?

Anzeichen für “Troubles”

- Mitglieder verlassen das Team
- “Inverser Dilbert Correlation Faktor” steigt
- Galgen-Humor
- neuer Projekt-Namen, z.B. das Titanic-Projekt
- viel Aktivität, aber kein Vorankommen

“Daily Build”-Konzept

Komplettes Compilieren, Linken, Installieren und Testen des Codes jeden Tag (bzw. Nacht)

Einige Tips:

- Projekt-Manager sollte sein Büro zum Test-Team verlegen
- Bei einem Crash des Daily Builds ist der Schuldige bis zum nächsten “Crash” für den Daily Build verantwortlich
- Ob eine Daily Build erfolgreich war, wird durch eine grüne oder rote Flagge angezeigt (Verantwortlicher dafür benennen)

Risk-Management

Fehlende Prozesse

- Fehlendes CM: viel Zeit wurde vertrödelt mit
  - SW-Build
  - Nachforschen, wer welche Version benutzt
  - Untersuchen, warum die SW auf dem einen System läuft, auf einem anderen Nicht
- Fehlendes Features/Defect-Tracking:
  - kein Testplan konnte aufgestellt werden
  - es war unmöglich festzustellen, wie weit das Projekt ist
- Fehlendes Requirements Management:
  - unnötige Diskussionen

Die 3 wichtigsten Dinge für ein Projekt

- Leute
- Leute
- Leute

Mit den richtigen Entwicklern kann man mit einem Team mehr bewirken als mit einer dreimal so großen Gruppe

Minimal Toolset

- E-Mail, Groupware, Internet/Web-Tools
- Prototyping/RAD Tools
- Configuration Management
- Testing-, Debugging-Tools
- Project Management Tools
  (z.B. ESTIMACS, CHECKPOINT, SLIM, ...)
- Wiederverwendbare Komponenten
  (kommerz. Bibliotheken, In-House-Komponenten, ...)
- CASE-Tools

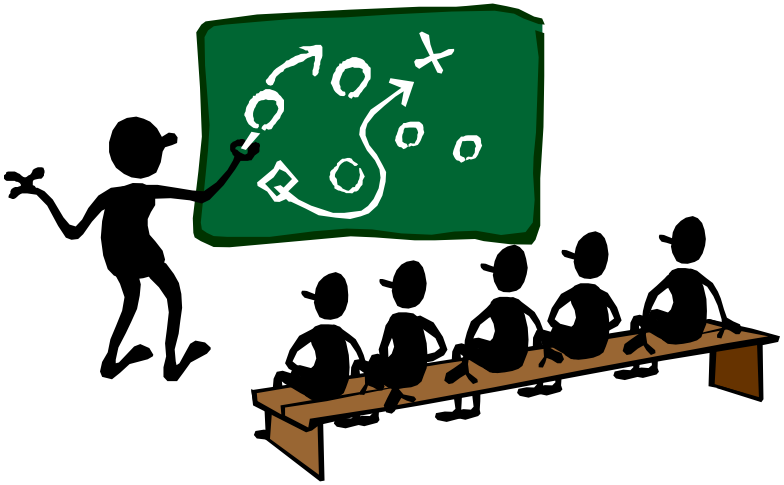
7 Stages of SW Engineering Mastery

(Page-Jones)

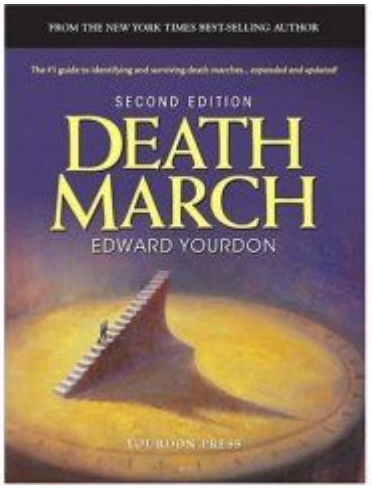
1. *Innocent* (hat noch nie was von ONT gehört)
2. *Aware* (hat schon mal was darüber gelesen)
3. *Apprentice* (hat einen 5-Tages-Workshop besucht)
4. *Practitioner* (bereit, ONT einzusetzen)
5. *Journeyman* (benutzt ONT on-the-Job)
6. *Master* (hat ONT verinnerlicht)
7. *Expert* (schreibt Bücher darüber)

Training

- Muss eingeplant werden bei
  - Einführung neuer Tools/Technologien
  - neuen Mitarbeitern
- bei fehlendem Training
  - längere Lernkurve
  - aus “Fehlern” wird gelernt
  - Gefahr von “falscher” Anwendung



Literatur



- Edward Yourdon: „DEATH MARCH“ Prentice Hall, ISBN 0-13-748310-4
- http://www.yourdon.com
- http://www.christine.com („Best Practice“)
- http://spmn.com (Airline Council´s Efforts)