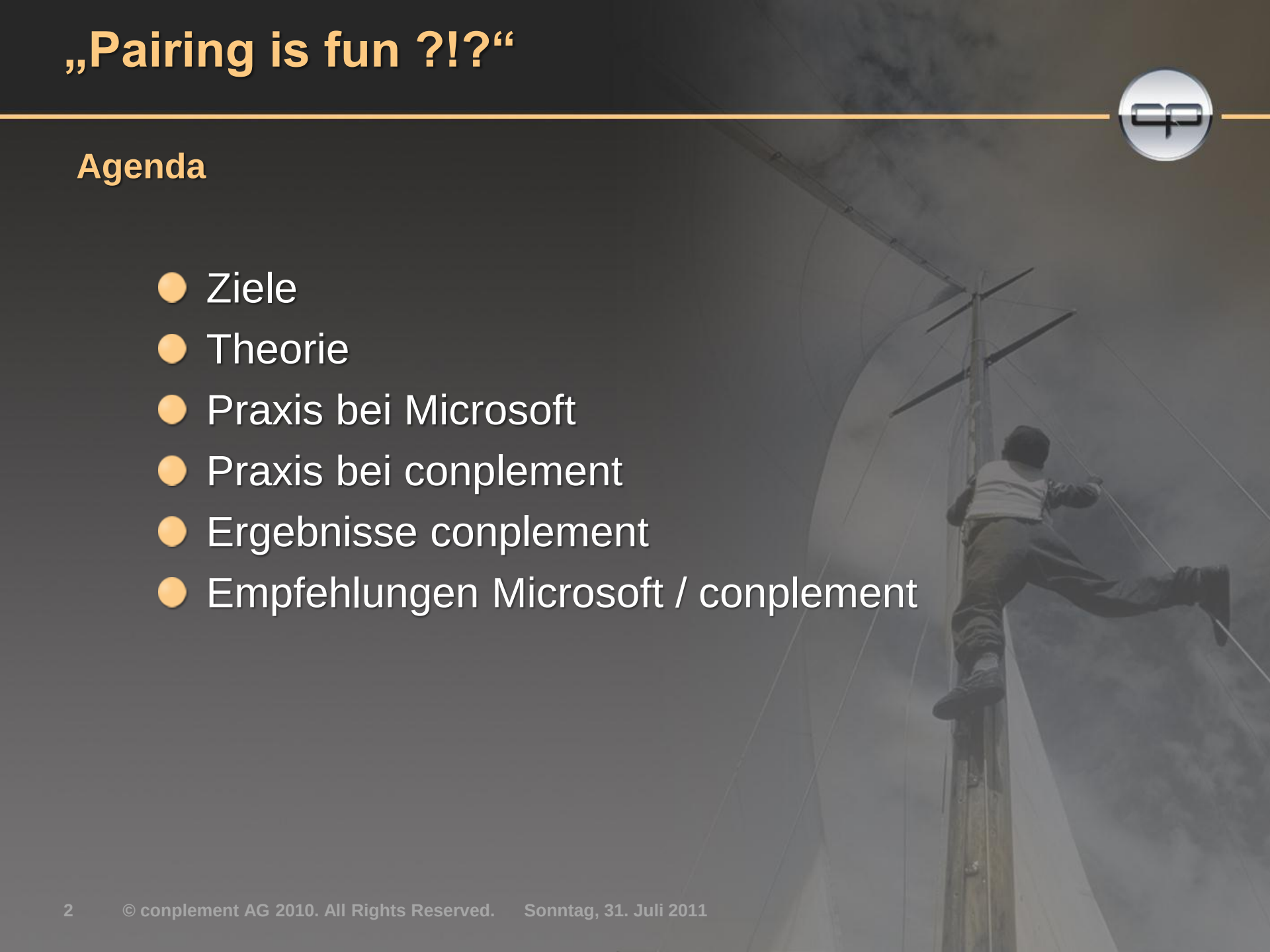

complement
getting ideas done

„Pairing is fun ?!?“

Agenda

- Ziele
- Theorie
- Praxis bei Microsoft
- Praxis bei conplement
- Ergebnisse conplement
- Empfehlungen Microsoft / conplement



„Pairing is fun ?!?“

Ziel dieses Vortrags

- Erfahrungsbericht aus der Praxis
- Anreiz geben „Pairing selbst zu Versuchen und eigene Erkenntnisse gewinnen
- Konkrete Erkenntnisse aus Pair-Programming bei conplement

Theorie

Was ist Pair-Programming

- Arbeitstechnik bei agilen Vorgehensweisen zur Softwareentwicklung.
- Bestandteil von Extreme Programming (XP) und Feature Driven Development (FDD)
- Zwei Programmierer teilen sich einen Arbeitsplatz
- „Pilot“, „Copilot“ oder „Driver“, „Navigator“ Arbeitsteilung
  - „Pilot“ -> aktiver Part
  - „Copilot“ -> „Der kompetente Ansprechpartner im Hintergrund“

Theorie

Was ist Pair-Programming

- Durchführung
  - Pilot bedient Computer und programmiert aktiv
  - Erläutert ständig Vorgehensweise (denkt laut)
  - Copilot hat mehr Abstand (tippt nicht!)
    - Überdenkt, prüft und verbessert die angewandten Problemlösungen
  - Permanente Diskussion zwischen dem Paar
  - Wechsel Pilot / Copilot nach einem definiertem Zeitabschnitt (Regelfall 1 Stunde)
  - Wechsel Besetzung Paare nach Fertigstellung Workitem

Theorie

Das Lehrbuch verspricht

- Höhere Disziplin
- Besserer Code
- Belastbarer Flow
- Höhere Moral / Motivation
- Collective Code Ownership
- Mentoring (Junior – Senior; Senior – Senior)
- Teambildung
- Weniger Unterbrechungen

Theorie

Das Lehrbuch warnt

- Doppelte Arbeitszeit
- Räumliche Gegebenheiten (z. B. Geräuschkulisse)
- Technische Gegebenheiten (z. B. großer Bildschirm)
- Zeitliche Koordination
- „Pair-Findung“
- Autoritätsproblem
- Verantwortlichkeit

Praxis bei Microsoft

Pair-Programming bei MS [1]

- Einsatz Pair-Programming
  - „Last Minute Bugs“ und schwierige Fixes
- Zeitraum: Oktober 2006; Basis ca. 500 Entwickler
  - Arbeitsumfeld bei Microsoft

[1] Quellenangabe: Andrew Begel, Nachiappan Nagappan, „Pairprogramming: What‘s in it for Me?“;

Praxis bei Microsoft

Pair-Programming bei MS [1]

- Benefits

1. Fewer Bugs	66
2. Spreads Code Understanding	42
3. Higher Quality Code	48
4. Can Learn from Partner	42
5. Better Design	30
6. Constant Code Reviews	22
6. Two Heads are Better than One	22
8. Creativity and Brainstorming	17
9. Better Testing and Debugging	14
10. Improved Morale	13

- Probleme

1. Cost efficiency	79
2. Scheduling	31
3. Personality clash	25
4. Disagreements	24
5. Skill differences	22
6. Programming style differences	13
7. Hard to find a partner	12
8. Personal style differences	11
9. Distractions	10
10. Misanthropy	9
10. Bad Communication	9
10. Metrics/Hard to Reward Talent	9

[1] Quellenangabe: Andrew Begel, Nachiappan Nagappan, „Pairprogramming: What‘s in it for Me?“;

Praxis bei conplement

Warum Pair-Programming bei conplement

- Sicherstellung der hohen Qualität bei Test-Start auch in komplexen UseCases durch Pair-Programming
  - Ziel: Schon VOR dem eigentlichen Systemtest fast Bug-frei sein
- Wissen über komplexen Code von Anfang an auf mehrere Schultern verteilt
  - Ziel: Risiko minimieren, den Abgabetermin zu verfehlen

Praxis conplement

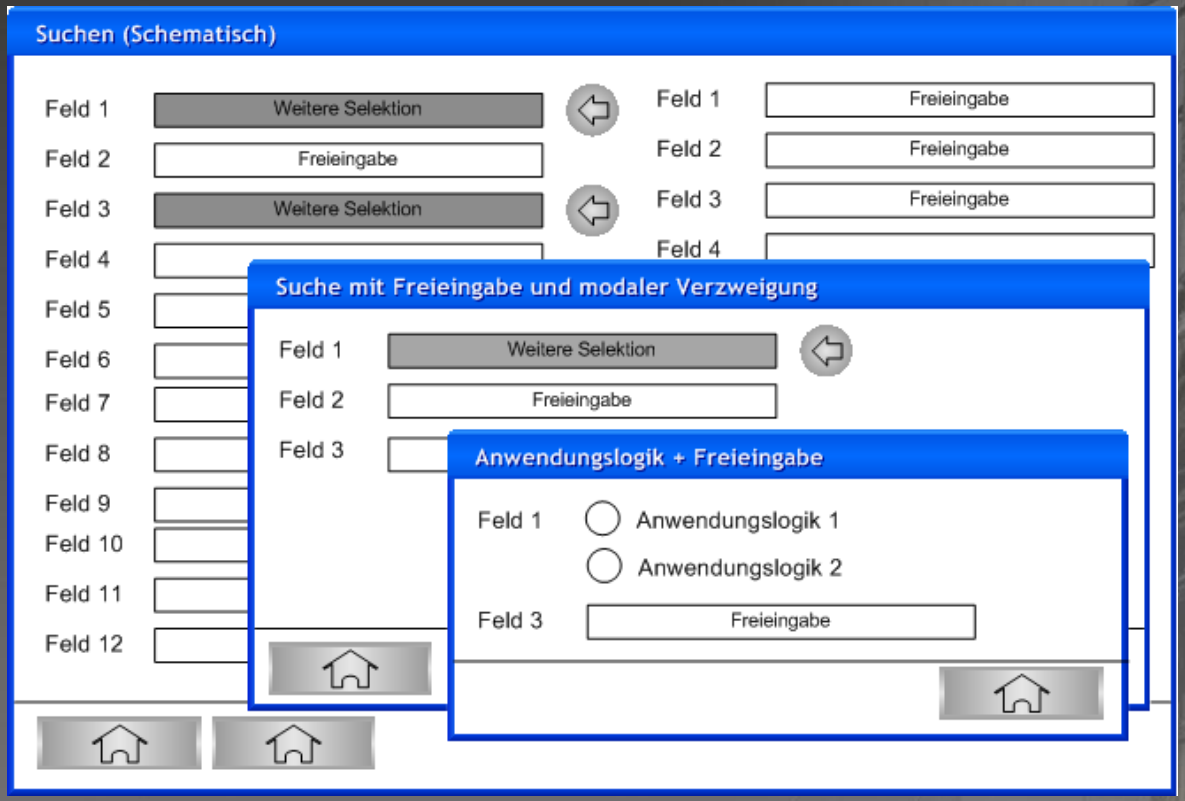
Aufgabenstellung

- Das Projekt
  - Migration einer Desktop-Applikation zu einer webbasierten Lösung
  - Umgebung: ASP.NET, AJAX, C#, Entity Framework
  - Gesamtaufwand: > 20 Personen-Jahre
  - Anzahl Dialoge: ca. 200
  - Teamgröße: bis zu 15 Entwickler(innen)
  - Agile Entwicklungsmethode mit ca. 15 Sprints
- Die Teilaufgabe
  - Migration von Suchfenster mit modalen Dialogen in eine gridbasierte Ergebnisanzeige mit Filterkriterien
  - Berücksichtigung von Applikationslogik bei Filterkriterien
  - Geschätzer Aufwand WorkItem: ~ 236 h

Praxis conplement

Aufgabenstellung

- Schematische Übersicht Altsystem



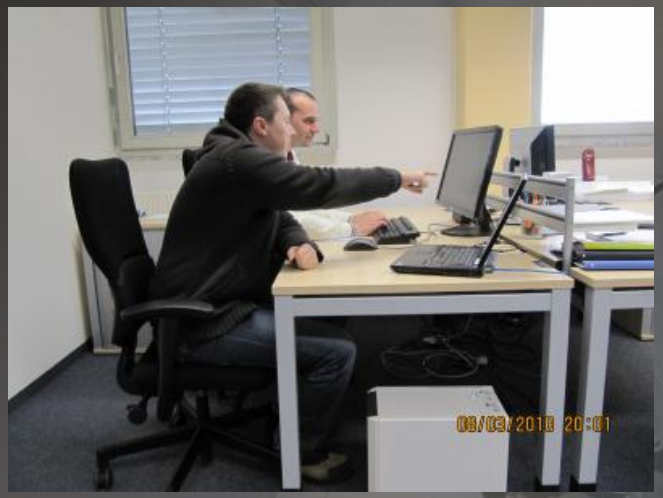
Praxis conplement

Aufgabenstellung / Details

- User Interface
  - 3 Selektionsseiten (extrem komplexe Business Logic)
  - 2 Dialoge zur erweiterten Suche
  - 1 Master / Detail Seite
  - 3 Dialoe zur Daten-Bearbeitung
- Business Logic
  - Ca. 25 einzelne komplexe Plausibilitätsprüfungen
- Unit Tests

Praxis conplement

Umfeld



- Normaler „Ein-Personen-Arbeitsplatz“
- Notebook mit ausreichend großem Monitor
- Kein Großraumbüro
- Kein Telefon

Praxis conplement

Durchführung

- Zeitliche Einteilung
  - Ca. 10:00 – 17:00 Uhr
  - Arbeitszeiten

    - Teammitglied 1: ca. 7:30 – 17:00 Uhr
    - Teammitglied 2: ca. 9:15 – 18:30 Uhr
    - Daily Scrum: 9:30 – 9:45 Uhr
  - Nur ca. 80% Pairing pro Tag -> 20 % Puffer für dringende Tasks
- Wechsel Pilot / Copilot im Stundentakt

Praxis conplement

Durchführung

Bilder sind nicht gestellt! Wurden mit Handy aufgezeichnet

Ergebnisse / Subjektive Einschätzungen

Thematische Einführung

Was	Holger	Robert
Fachliche und architekturelle Einführung. Diskussion über mögliche Umsetzungen	++	++

++ Sehr Effektiv; + Effektiv; 0 Neutral; - Weniger zielführend; - - Nicht zielführend

- UseCases sind i.d.R. „Neuland“ für die Entwicklung
- Einführug durch Business Analyst erfolgt im Pair.
- Gegenseitige Verständnisprüfung und Diskussion der peglanten Umsetzung bzw. der geplanten Algorithmen ist sehr effektiv

Ergebnisse / Subjektive Einschätzungen

Routinetätigkeiten

Was	Holger	Robert
Routinetätigkeiten: z. B. Überprüfung vorgenerierter Code, Anpassungen an Gridspalten, Übersetzungen, ...	0	-

++ Sehr Effektiv; + Effektiv; 0 Neutral; - Weniger zielführend; - - Nicht zielführend

- Bei Routinetätigkeiten ist der Benefit der gegenseitigen „Überwachung“ im Pair kaum spürbar.
- Copilot „langweilt“ sich schneller.
- ABER!! Flüchtigkeitsfehler werden vermieden

Ergebnisse / Subjektive Einschätzungen

Einfache Programmiertätigkeiten

Was	Holger	Robert
z. B. Programmierung Filteralgorithmen (DB View / DB SP, *.cs, *.ascx), Popup Window	+	++

++ Sehr Effektiv; + Effektiv; 0 Neutral; - Weniger zielführend; - - Nicht zielführend

- Bei einfachen, standardisierten Programmiertätigkeiten „Copy/Paste„ werden Flüchtigkeitsfehler durch den Copiloten vermieden
- Aufwendige Test-Zyklen entfallen
- Gegenseitige Überwachung greift sehr gut
- Hohe Anspannung / Belastung für den Copiloten

Ergebnisse / Subjektive Einschätzungen

Komplexe, neue Anforderungen (evtl. POC)

Was	Holger	Robert
Von der „Norm“ abweichende Algorithmen. Die Ausnahme von der Regel!	++	++

++ Sehr Effektiv; + Effektiv; 0 Neutral; - Weniger zielführend; - - Nicht zielführend

- Jede Applikation beinhaltet Spezialfälle die nicht nach dem „normalen“ Muster gelöst werden können. In diesem Fall ist Pairing extrem effektiv.
- Achtung! Gefahr!
  - Pair verläuft sich in technisch interessanten Umsetzungsvarianten. Versucht „goldene Wasserhähne“ zu bauen und ist damit nicht ergebnisorientiert.

Ergebnisse / Subjektive Einschätzungen

Performance bei Dialogerstellung / „Serien-Fertigung“

Was	Holger	Robert
Integration wiederkehrender Schemata bei Dialogerstellung	++	++

++ Sehr Effektiv; + Effektiv; 0 Neutral; - Weniger zielführend; - - Nicht zielführend

- Integration von wiederkehrenden Programmabläufen (kein Copy/Paste) ist bezogen auf den Zeitbedarf sehr effektiv. Z. B. Ableitung von Dialogen im gleichen Use Case

Ergebnisse / Die konkreten Zahlen

Fakten

	Pair Aufgabe	Referenz Aufgabe
Gebuchte Stunden Implementierung	163 Std	170 Std
Durchführung Rütteltest	8 Std	8 Std
Anzahl Bugs (keine Schönheitsfehler)	1	5
Fix Bugs aus Rütteltest	16 Std	16 Std
Durchführung Testplan	6 Std	10 Std*
Anzahl Bugs aus Testplan	0	8/1*
Fix Bugs aus Testplan	0 Std.	20 Std.
Summe	~ 193 Std	~ 224 Std

* 2 Testplandurchläufe nötig.

Empfehlungen Microsoft

Pair Programming bei MS [1]

- Wünschenswerte Eigenschaften „Pairer“

Table 3: Attributes of good pair programming partners	
1. Complementary Skills	40
2. Flexibility	33
3. Good Communications	31
4. Smart	25
4. Personable	25
6. At Least the Same Skills as Me	21
7. Strong programmer	17
7. Better Skills than Me	17
7. Able to Focus	17
10. Knowledgeable	15

[1] Quellenangabe: Andrew Begel, Nachiappan Nagappan, „Pairprogramming: What‘s in it for Me?“;

Empfehlungen conplement

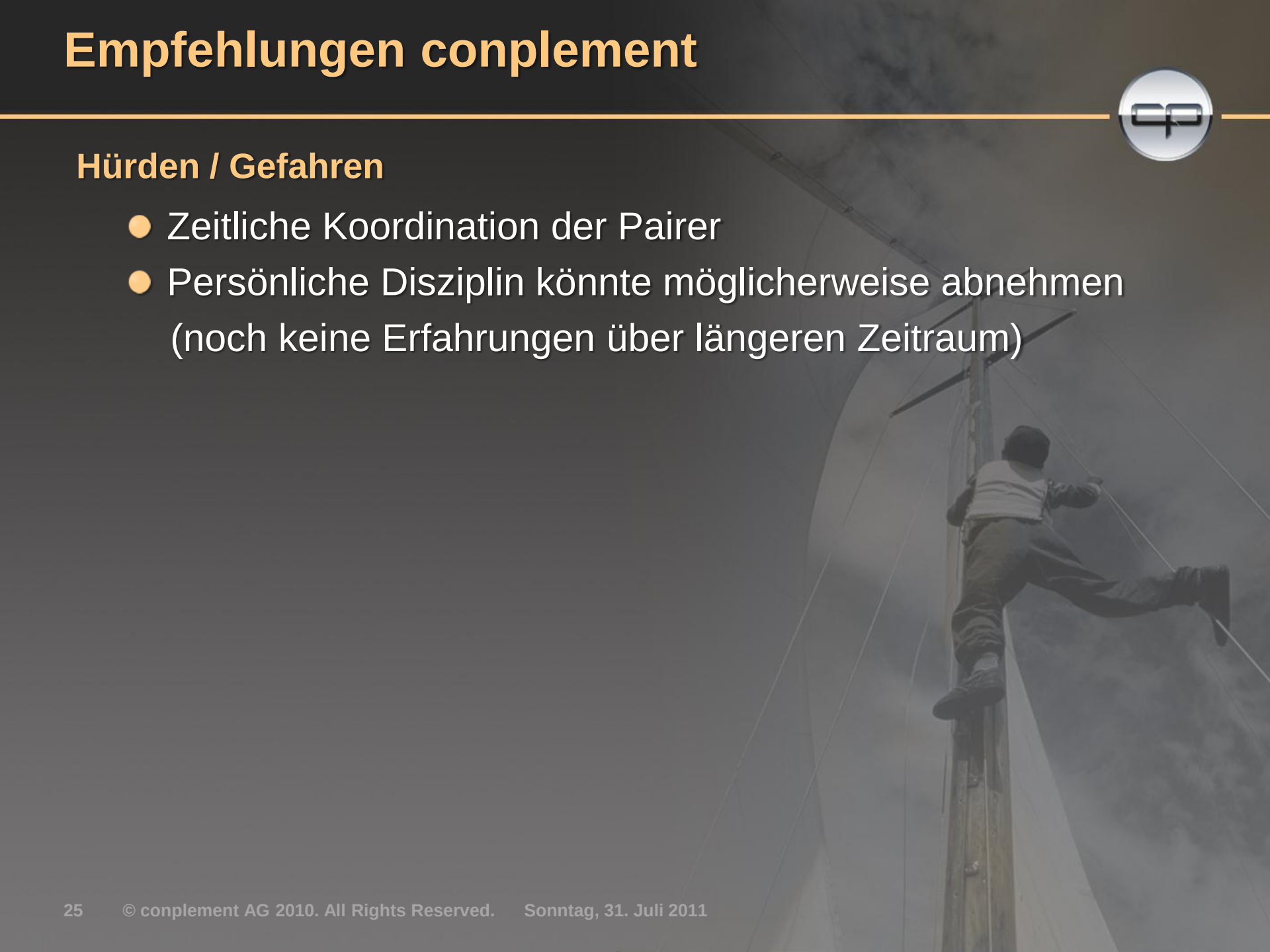
Empfehlung

- Entwicklersicht: Wir würden Pairing gerne bei allen Requirements nutzen
- Projektleitungs-Sicht: Pairing nur bei hochkomplexen Requirements
- Dem „Pair“ sollte nicht ein einzelnes WorkItem sondern ein kompletter UseCase zur Abarbeitung übergeben werden
- Das „Pair“ entscheidet selbst, welche davon alleine (Fleißarbeit, evtl. Routinetätigkeiten...) und welche im wirklichen „Pair-Programming“ umgesetzt werden.
- Auch bei „Solo-Arbeiten“ sollte das Pair nebeneinander sitzen.
- Fachliche Einführung in allen UseCases erfolgt immer im Pair.
- Wechsel der Pairpartner erst nach Abarbeitung eines UseCases und damit im Wochenbereich (nicht täglich) sinnvoll.
- Wechsel der Pair-Partner erscheint sinnvoll wenn der Mentoring Gedanke im Vordergrund steht.

Empfehlungen conplement

Hürden / Gefahren

- Zeitliche Koordination der Pairer
- Persönliche Disziplin könnte möglicherweise abnehmen (noch keine Erfahrungen über längeren Zeitraum)



Empfehlungen conplement

Etiquette

- Pilot
  - Keine privaten Mails / Messenger Kontakte
  - Keine Einstellungen am Computer ändern (Tastaturlayout, Sprache, Hintergrund ...)
  - Immer „Laut-Denken“
- Copilot
  - Nicht „Zurücklehnen und Entspannen“
  - Nicht „Mach nur, das passt schon“
  - Keine SMS / Handy Spielereien

Zusammenfassung

Fazit

- Pair-Programming mit Augenmaß
- Evtl. höherer Zeitaufwand wird kompensiert
- Für Kunden nicht erreichbar -> mit allen Konsequenzen
- Pairing Team kann negative Aspekte bei Routinearbeit kompensieren. Team benötigt Entscheidungsfreiraum
- Pairing is wesentlich anstrengender, macht aber deutlich mehr Spaß.

!! PAIRING IS FUN !!

Ich freue mich auf Ihre Fragen

Robert Eichenseer
robert.eichenseer@conplement.de
Tel: (09 11) 2550976-49
www.conplement.de

