

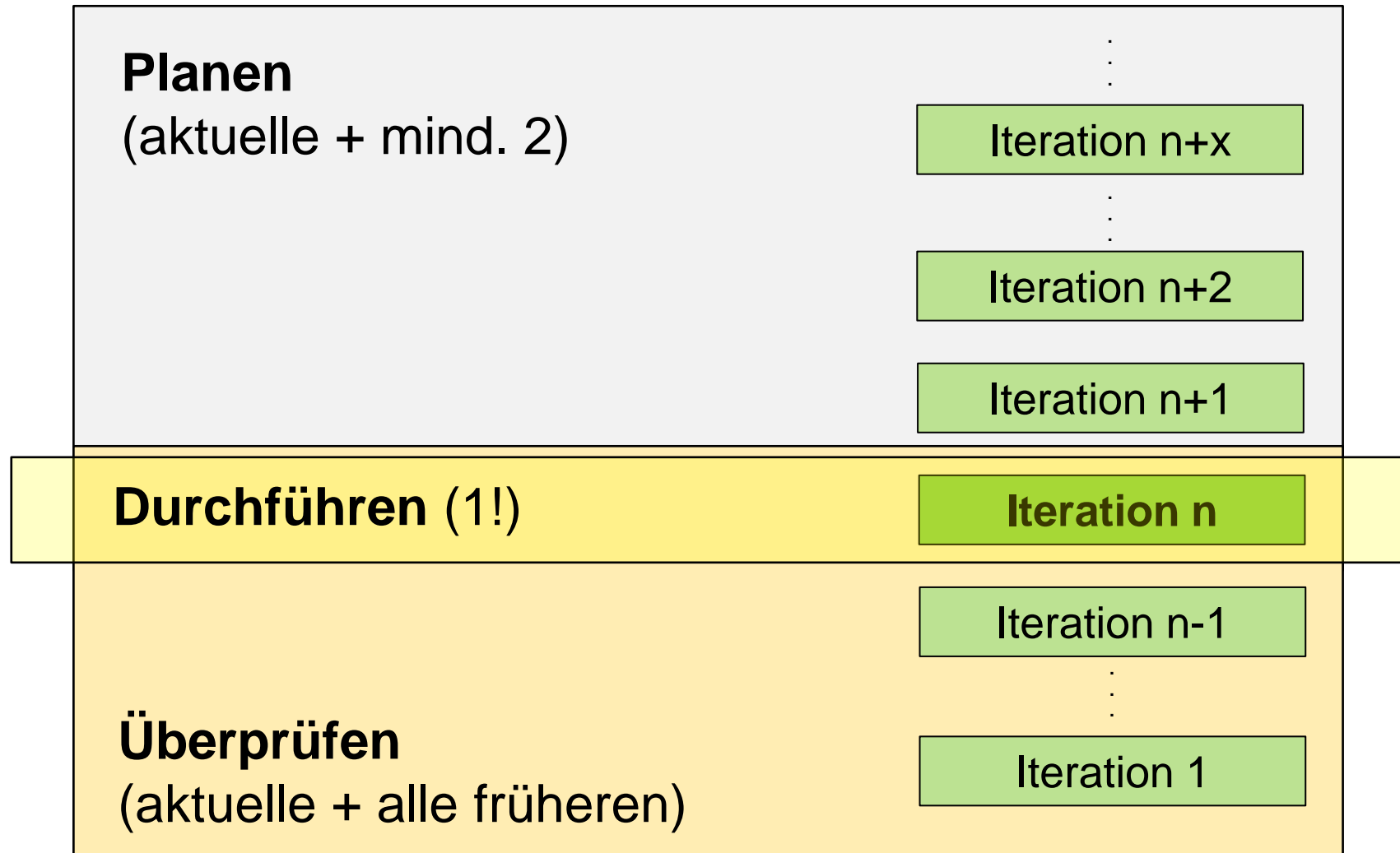
Tipps zur Agilen Softwareentwicklung

Herwig Mayr

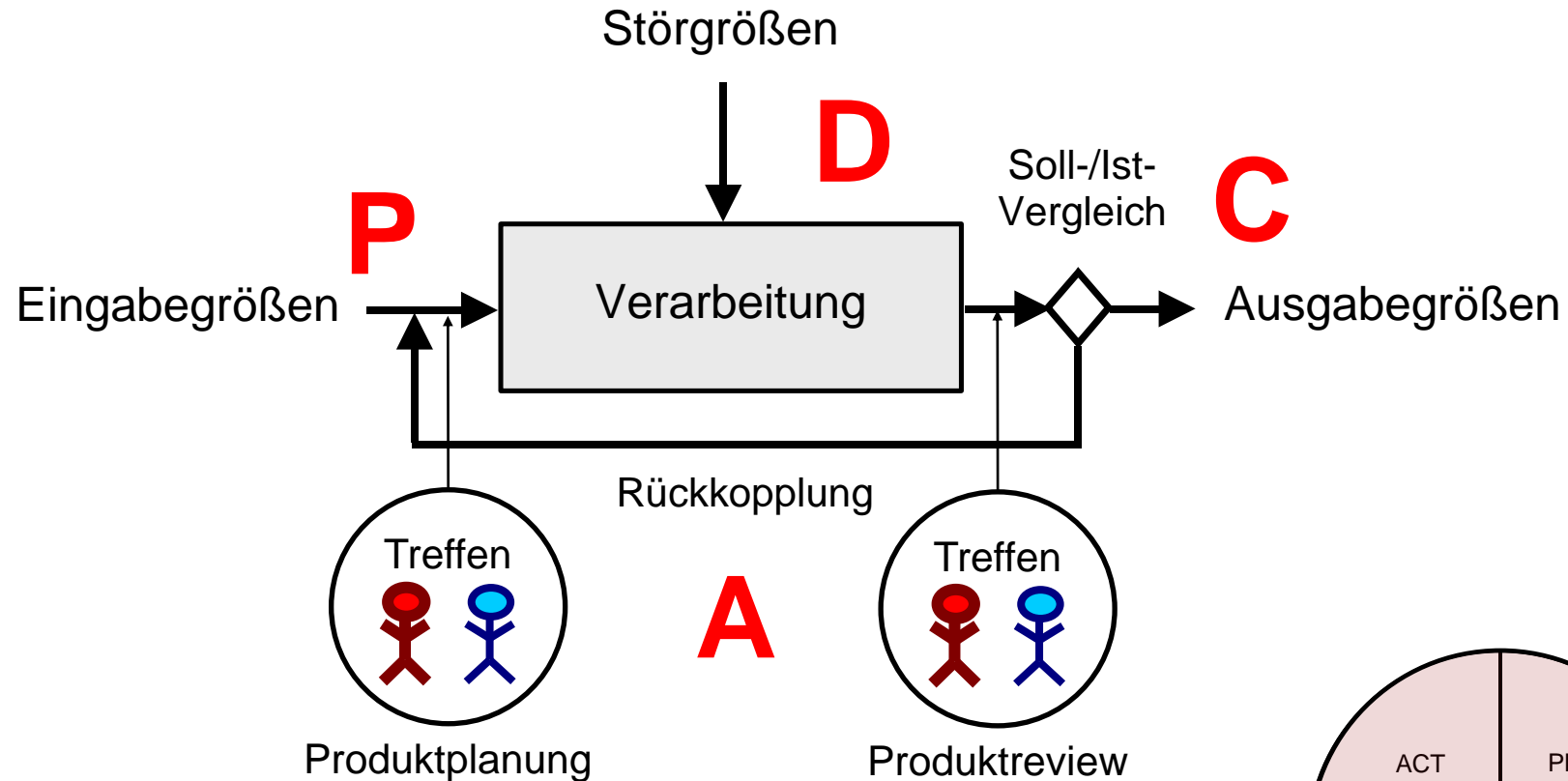
VL Software-Prozesse
Studiengang Software Engineering,
FH OÖ Campus Hagenberg



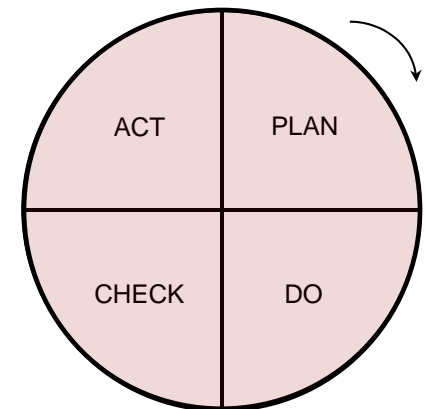
Konzept: Denken in Iterationen



Denken in Iterationen: Geregelter Prozess



Vorgehen entspricht **PDCA-Zyklus** (Deming)!

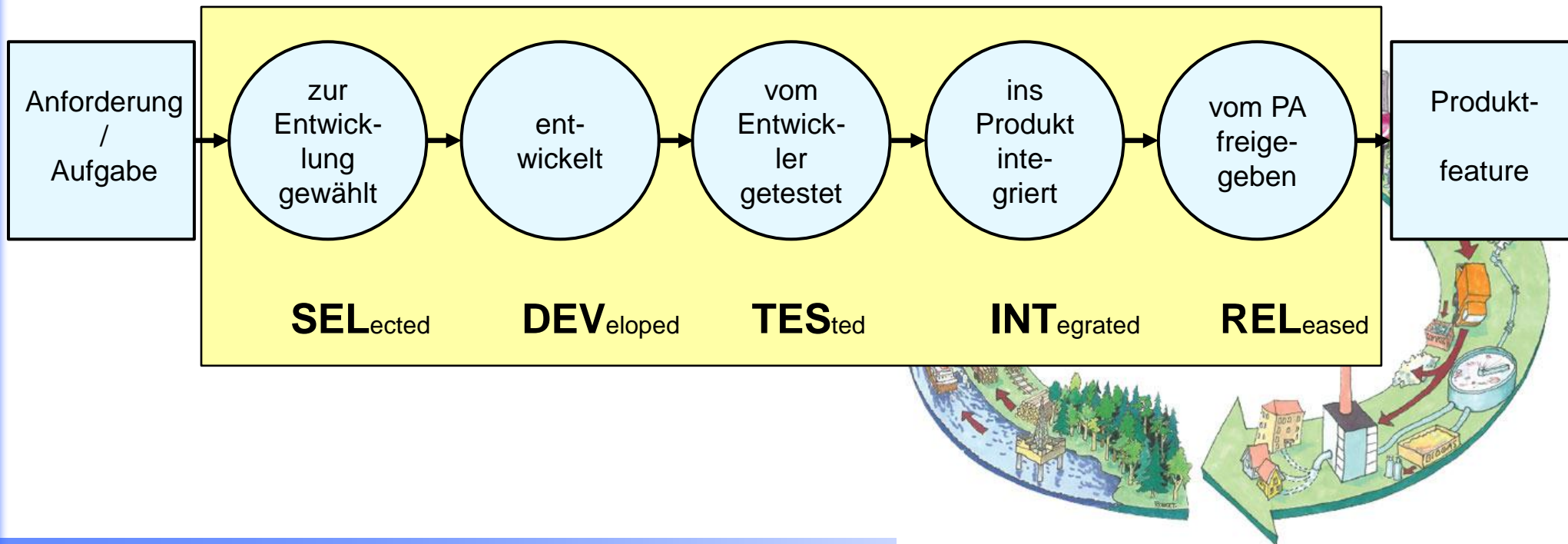


Konzept: Denken in Zuständen

Aufgaben befinden sich während ihrer Abarbeitung in definierten Zuständen.

Begriffe frei wählbar; Abfolge wichtig
(„**Zustandskette**“ bzw. „**Wertschöpfungskette**“)

Beispiel:



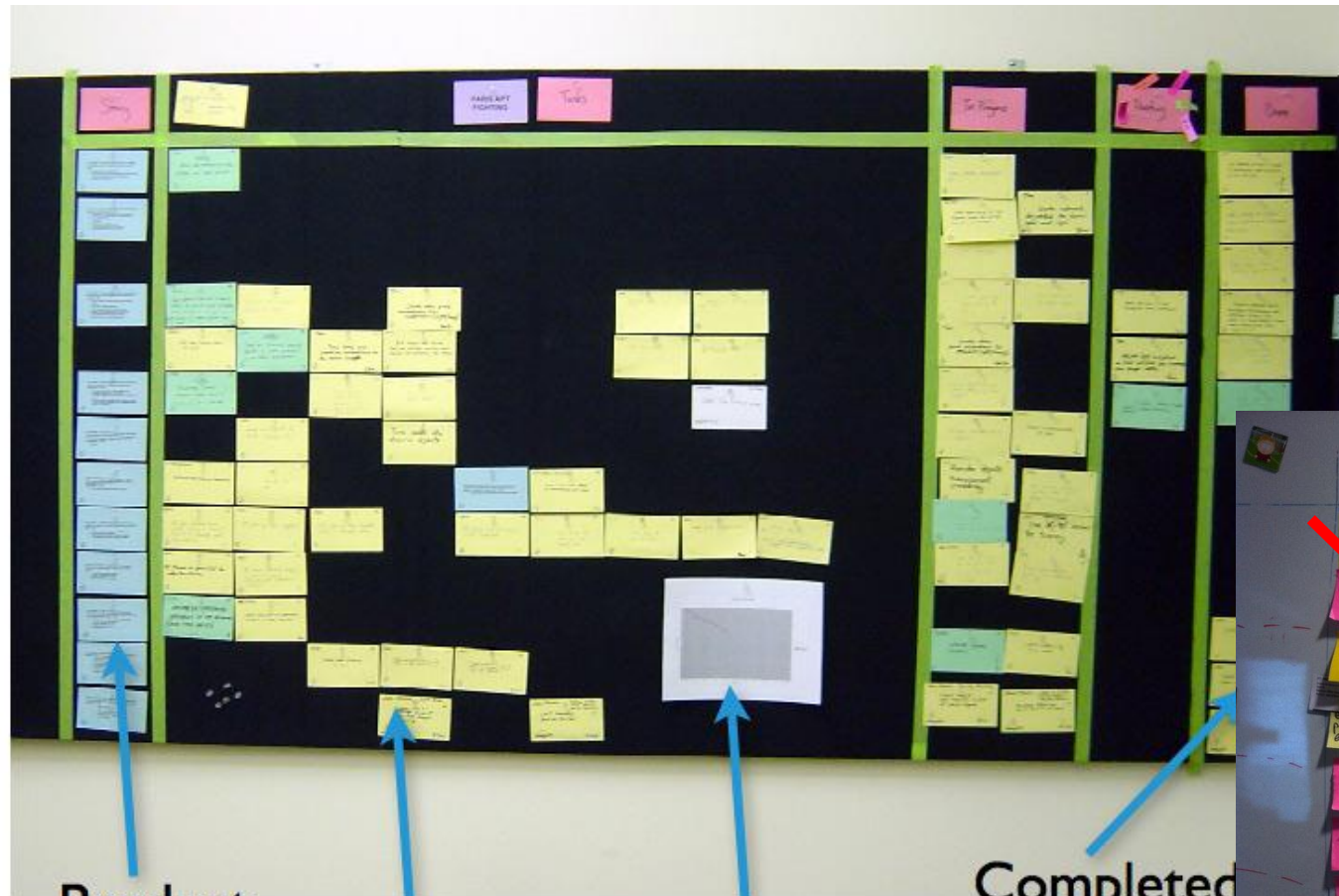
Aufgabenübersicht (Task Board)

Darstellung der **Wertschöpfungskette je Aufgabe** einer Iteration
(eine gemeinsame Übersicht pro Team!)

Anf./Aufg.	SEL	DEV	TES	INT	REL
Anforderung 2					
Aufgabe 2.1					
Aufgabe 2.3					
Aufgabe 2.4					
Anforderung 3					
Aufgabe 3.3					
Aufgabe 3.4					
Anforderung 5					
Aufgabe 5.1					
Aufgabe 5.2					
...					

Festgelegte Regeln für den Übergang von einer Spalte zur nächsten
sind wichtig („**Definition of Done**“)!

Beispiel: Scrum Task Board

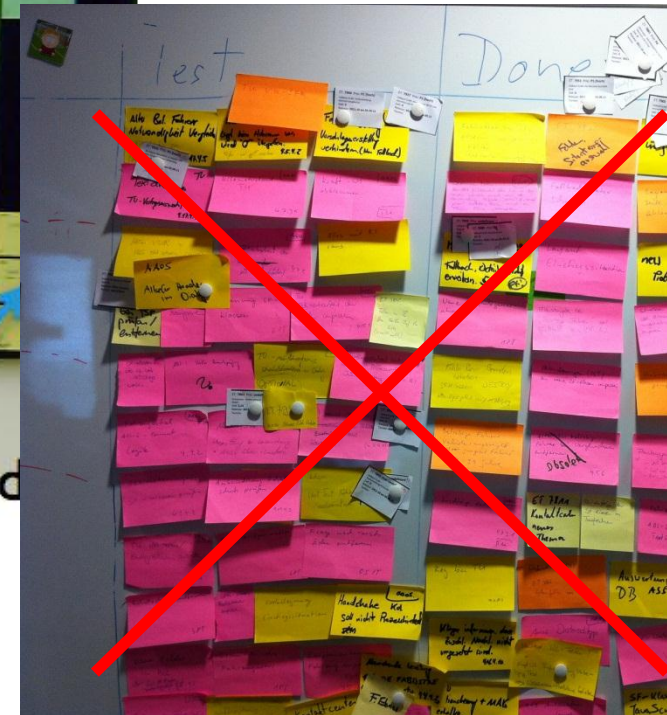


Product
backlog

Tasks
to do

Burndown
chart

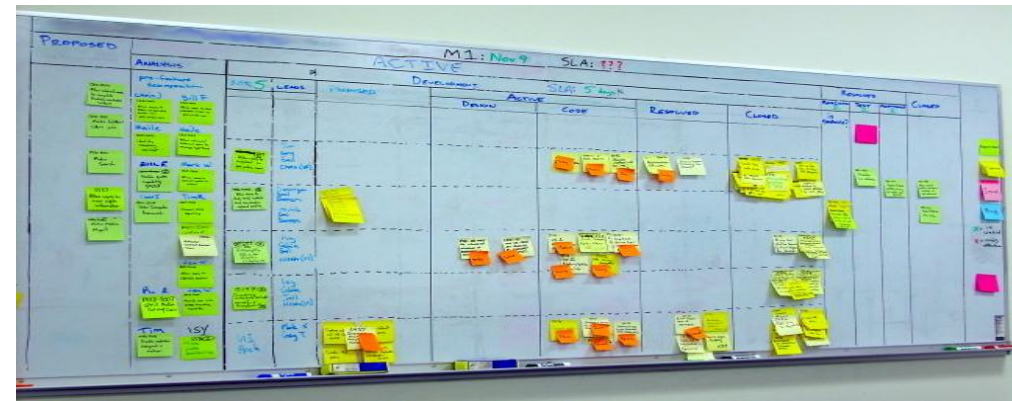
Completed
tasks



Abarbeiten von Zustandsketten: Kanban

Kanban: von jap. **Kan** = Signal; **Ban** = Karte

- nach dem 2. Weltkrieg in Japan erfunden (T. Ohno, Toyota)
- Produktablaufsteuerung zur Optimierung des Lagerstands
- betont das Pull-Prinzip (Zulieferung je nach Verbrauch)
- von D. Anderson für IT-Projekte adaptiert und um Konzepte der agilen Entwicklung erweitert (2007)



Kanban-Idee: Bsp. Essenszubereitung














- KEINE Vorgehensmethode, sondern Entwicklungstechnik! (aber in Vorgehensmethoden, z.B. Scrum, einsetzbar)
 - kein Requirements Management
 - keine Teamorganisation, keine Rollen
 - keine zeitliche Vorgaben / Durchlaufzeiten → keine Terminplanung
 - keine Iterationen (!), außer in Notfällen
 - aber inkrementell (falls sinnvoll)
 - Releasezyklen frei wählbar
 - Synchronisation der Aufgabenerledigung möglich
 - auch für mehrere Projekte zur selben Zeit!
- Gut einsetzbar für kleine, kurz getaktete, möglichst unabhängige Aufgaben (z.B. Wartungstätigkeit)!

Kanban: Empfehlungen

nach D. Anderson:

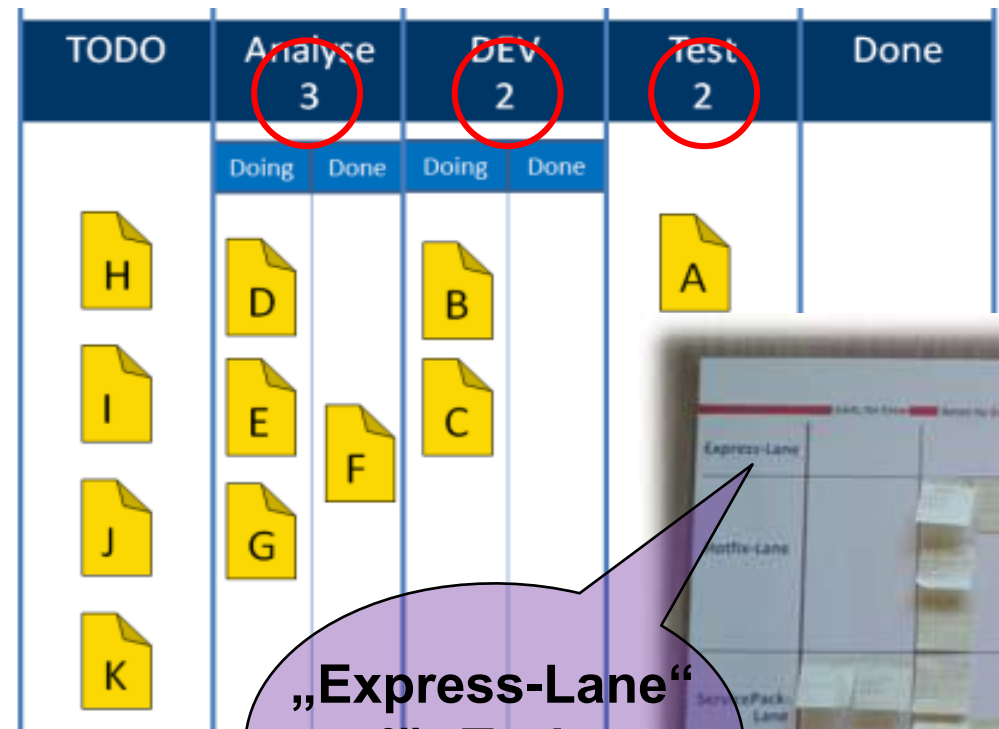
- **Visualisiere den Fluss der Arbeit.**
(Kanban-Board)
- **Begrenze die Menge angefangener Arbeit.**
(Ticket System, Pull-Prinzip)
- **Miss und steuere den Fluss.**
(Cycle Time Analysis, „Cost of Delay“)
- **Mache die Regeln für den Prozess explizit.**
(Definition of Done)
- **Verwende Modelle, um Chancen für kollaborative Verbesserungen zu erkennen.**
(Kaizen)

TODO	Analyse 3		DEV 2		Test 2	Done
	Doing	Done	Doing	Done		
						
						
						
						

Darstellung am Kanban-Board

Elemente *pro Spalte* ist begrenzt (außer Eingang und Ausgang)!

→ Zahl in Spaltenheader; Theory of Constraints



Mehrere Teams können
sich eine Tafel teilen!

„Swim Lanes“ (oder
versch.farbige Kärtchen)
für Teilprojekte

„Express-Lane“
für Tasks
vom Chef 😊



Bild: Fa. eurofunk Kappacher, St. Johann/P.

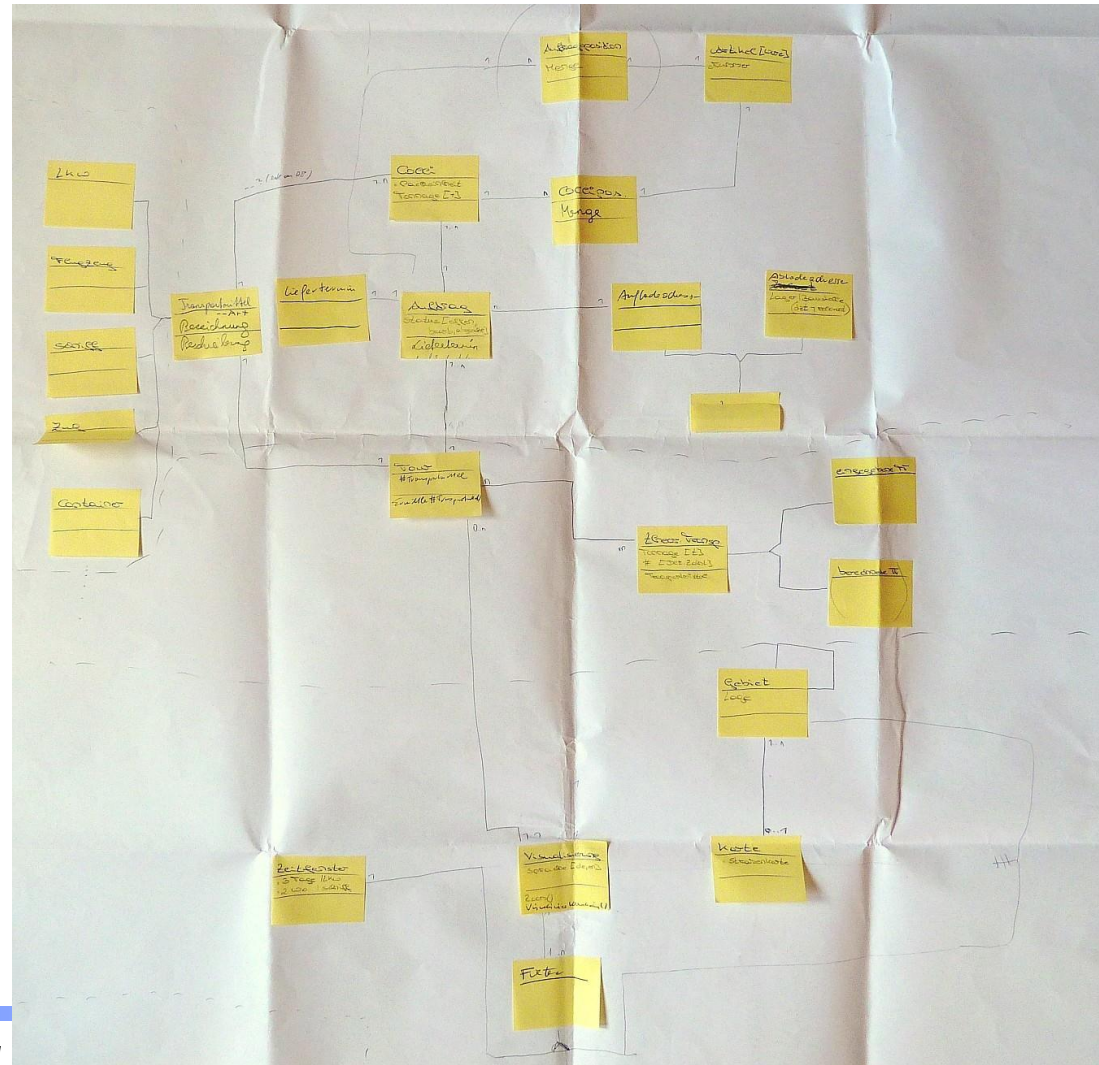
Zielerreichung durch Anzahl der Elemente (Aufgaben) in der rechten Spalte

Fortschritt durch „Cycle Time Analysis“

- **Zweck: gleichmäßige Arbeitsbelastung**
(vgl. „40-hour-week in XP“, „regular work load“ in Scrum)
- **Notwendigkeit: ~ gleich große Aufgaben**
(oft schwierig zu erreichen bzw. vorab schwer schätzbar)
- **Nur dann ist die Zykluszeitmessung aussagekräftig!**
(„Cycle Time“ beschreibt, wie lange Element x in Spalte y ist.)

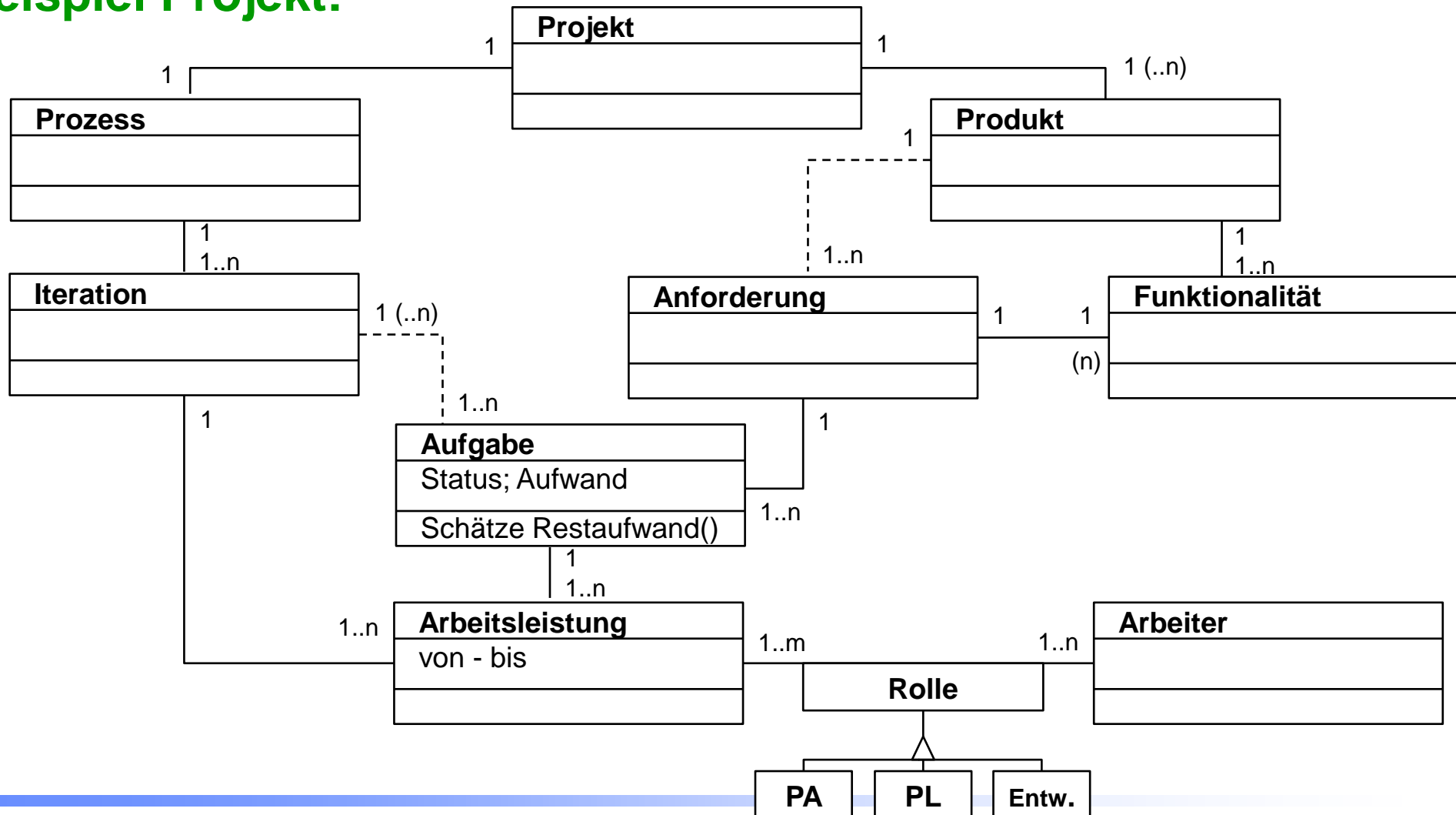
- Entwicklung: Struktur und Abläufe
- Ergebnis: Benutzung, Anwendungsfälle
- Prozess: (Um-)Planen aktuelle/zukünftige Iterationen

© Herwig Mayr, FH OÖ Campus Hagenberg



Planen: Statische Zusammenhänge (I)

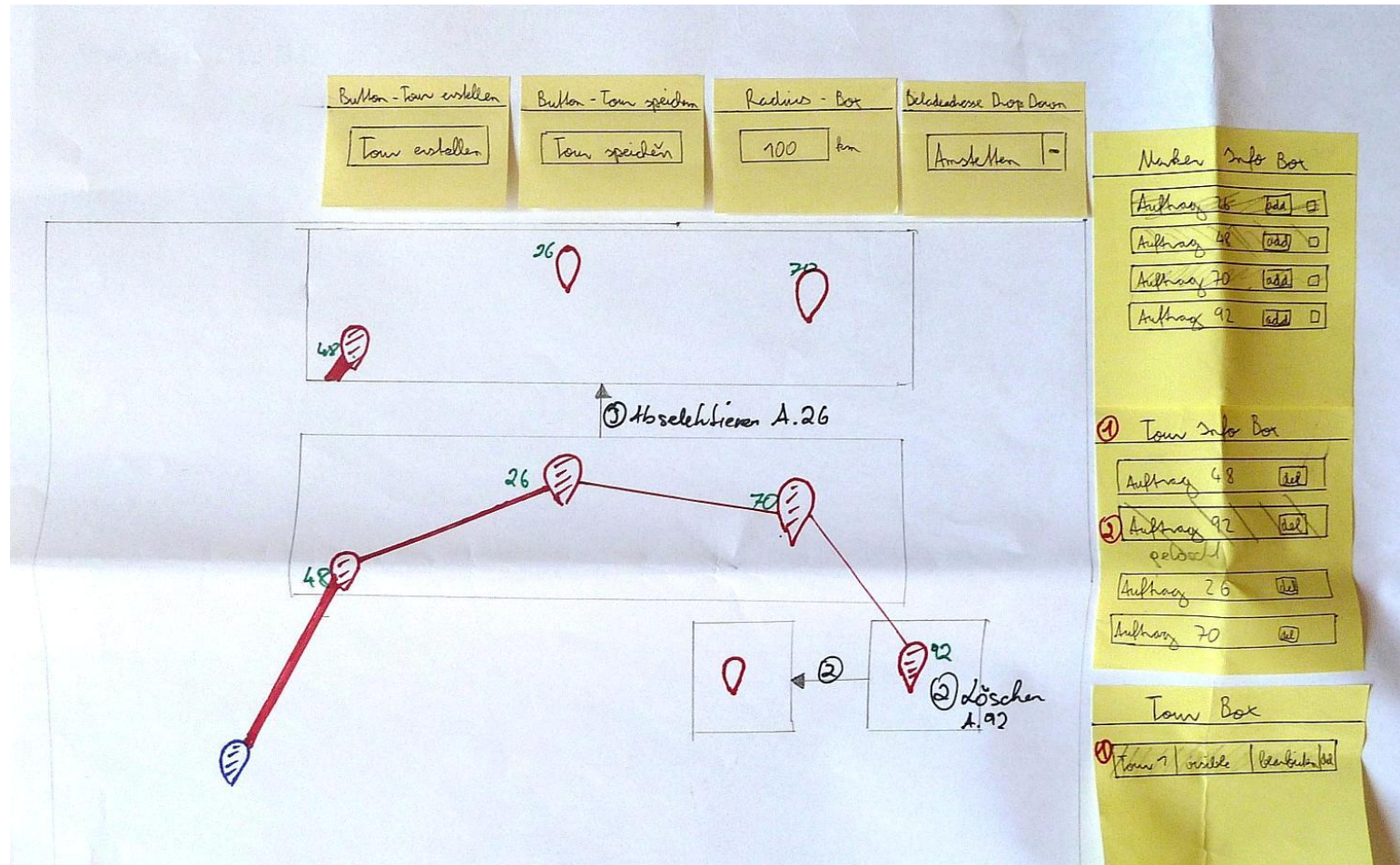
Klassendiagramm – Beispiel Projekt:



(„Swim Lane“ definiert Verantwortungsbereiche)

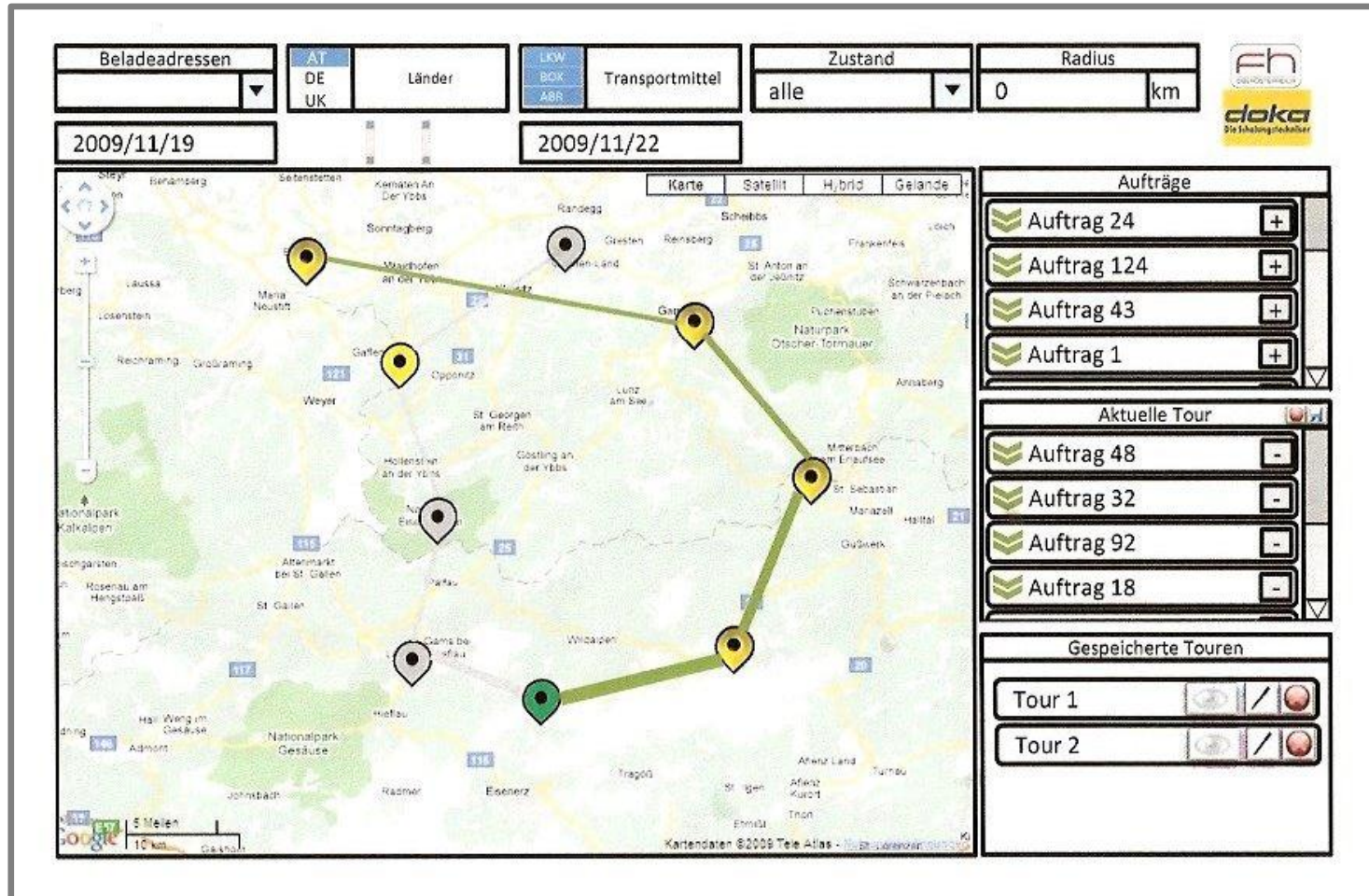


Mittels Benutzerschnittstellen-Prototyp: (am Anfang durchaus auf Papier!)



Planen: Erarbeiten der Ablauflogik (II)

Mittels Benutzerschnittstellen-Prototyp: (später als Software)



Planen: Erstellen typ. Anwendungsfälle (II)

① Auswahl der Beladeadresse

- a) Drop Down Menü
- b) Marker der Beladeadresse anklicken

Bei Auswahl der Beladeadresse bleiben InfoBoxen leer (Marker und Tour InfoBox)

② Auswahl des Radiusfüller

- a) Eingabe in Radiusbox
- b) Radius aufziehen mit Maus
Rechtsklick auf Marker → Gantt-Contentbox → Radiusfüller

③ Auswahl Abladeadresse im Kreis Radiusfüller (Linksklick)

- Marker InfoBox beinhaltet alle Aufträge des Markers (26,48,70)
- Strichierte Linie von Beladeadresse zu Marker Abladeadresse
- Linie unabhängig von Tourage, weil noch kein Auftrag selektiert

17 3+
(Kern)

④ Selektieren eines Auftrages in der Marker Info-Box

- a) Auftrag 48 → wird die Check box aktiviert (benötigt zum Anwenden Button "Anwenden")

b) Linksklick auf Button "Add" des Auftrags 48

- Auftrag 48 wird in der Tour Info Box hinzugefügt
- ⇒ Auftrag 48 wird in Marker Info Box ausgegraut

- Strichierte Linie zwischen Marker Beladeadresse und Marker Abladeadresse
→ wird zu durchgehender Linie, da Strichstärke abhängig von Tourage des Auftrags

⑤ Repeat Step ② - ④

5.2 Radiusfüller ändern

5.3 Auswahl Abladeadresse

→ Zusätzlich in Marker Info Box wird Auftrag 92 angezeigt

⑦ Darstellung gespeicherter Tour

- Beinhaltet Marker noch nicht zu einer Tour hinzugefügte Aufträge
→ Marker bleibt normal eichlbar, Verbindungsline ausgegraut
- Alle Aufträge eines Markers einer Tour hinzugefügt
→ Marker wird ausgegraut, Verbindungsline ausgegraut

⑧ Neue Tour erstellen

- Mit Klick auf Button "Neue Tour" wird eine neue Tour gestartet

Pilotanwender erstellt Anwendungstests gemäß des Ablaufs der Anwendungsfälle!

Festlegung der für den Auftraggeber sichtbaren Ergebnisse:

- **Externe Anforderungen/Aufgaben:** führen zu einem Ergebnis mit unmittelbarem Wert für den AG
- **Interne Anforderungen/Aufgaben:** dem AG nicht direkt „verkaufbar“, aus Projektsicht (AN) aber notwendig
- Einteilung über Projektverlauf einheitlich halten (Checkliste)
- Einheitlichkeit notwendig für Vergleiche zwischen Projekten (\Rightarrow Management-Overhead!)

Gründe:

- Mitarbeiter schließen Aufgaben früher als erwartet ab (priorisierte Liste weiterer Aufgaben vorhalten!)
- Aufgaben werden abgebrochen („Plan B“)
- Aufgaben werden obsolet (vom AG nicht länger benötigte Funktionalität)
- Störgrößen beeinflussen den Projektverlauf (Krankheit, Marktveränderungen, ...)

Planen: Vorplanen zukünftiger Iterationen

Teilbereiche:

Abgebrochene und nicht abgeschlossene Aufgaben	Interne Anforderun- gen	Bekannte externe Anforderun- gen	Neue Anforderungen des AG	Puffer (z.B. 20%)
---	-------------------------------	---	--	----------------------

Tipps zum Durchführen

- Arbeitszeit erfassen
- Restaufwand schätzen
- Aufgabenstatus aktualisieren


Prinzipien:

- Alle auf das Projekt zu buchende Arbeit wird erfasst (evtl. internes & externes Projektlogbuch).
- Jede Arbeit(seinheit) wird einer Aufgabe zugeordnet (ggf. **vorher** Aufgabe erstellen!).
- Jeder Mitarbeiter erfasst jeden Tag seine Arbeit.
- Eine sinnvolle kleinste Einheit ist zu definieren (z.B. 0,5 h).

Durchführen: Schätzen des Restaufwands

Jeden Tag erneut, kann auch mehr werden!

(mit oder ohne Vorgabe der Letztschätzung)

 Projekt - Durchführung - 3. Iteration - 11. Kriterienbewertung am PDA - **11.1. Sourcecode dokumentieren**

Geleistete Arbeitszeit eintragen

Aufgabe

11.1. Sourcecode dokumentieren

Verantwortlicher Pilotanwender

carmen

Mitarbeiter

thomas

Iterationstag

02.06.2005

letzte Schätzung

8

UE

wird später ev. nicht angezeigt

geleistete Arbeitseinheiten

UE

neue Schätzung

UE

Änderungen übernehmen

Änderungen verwerfen

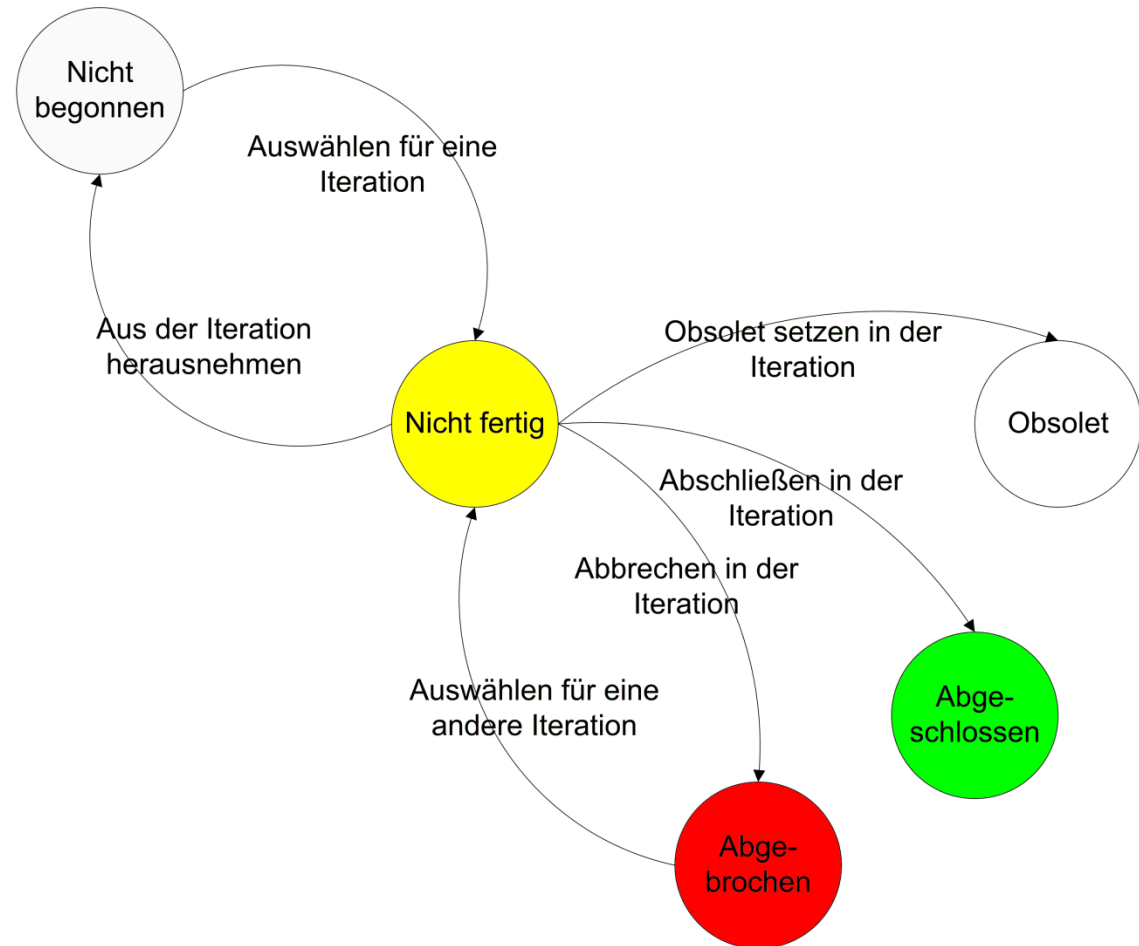
Aufgabe abschließen

Aufgabe abbrechen

Aufgabe obsolet setzen

Durchführen: Ändern des Aufgabenstatus

Möglichst mittels Tracking-System und Tickets:
(Ampel-Metapher!)



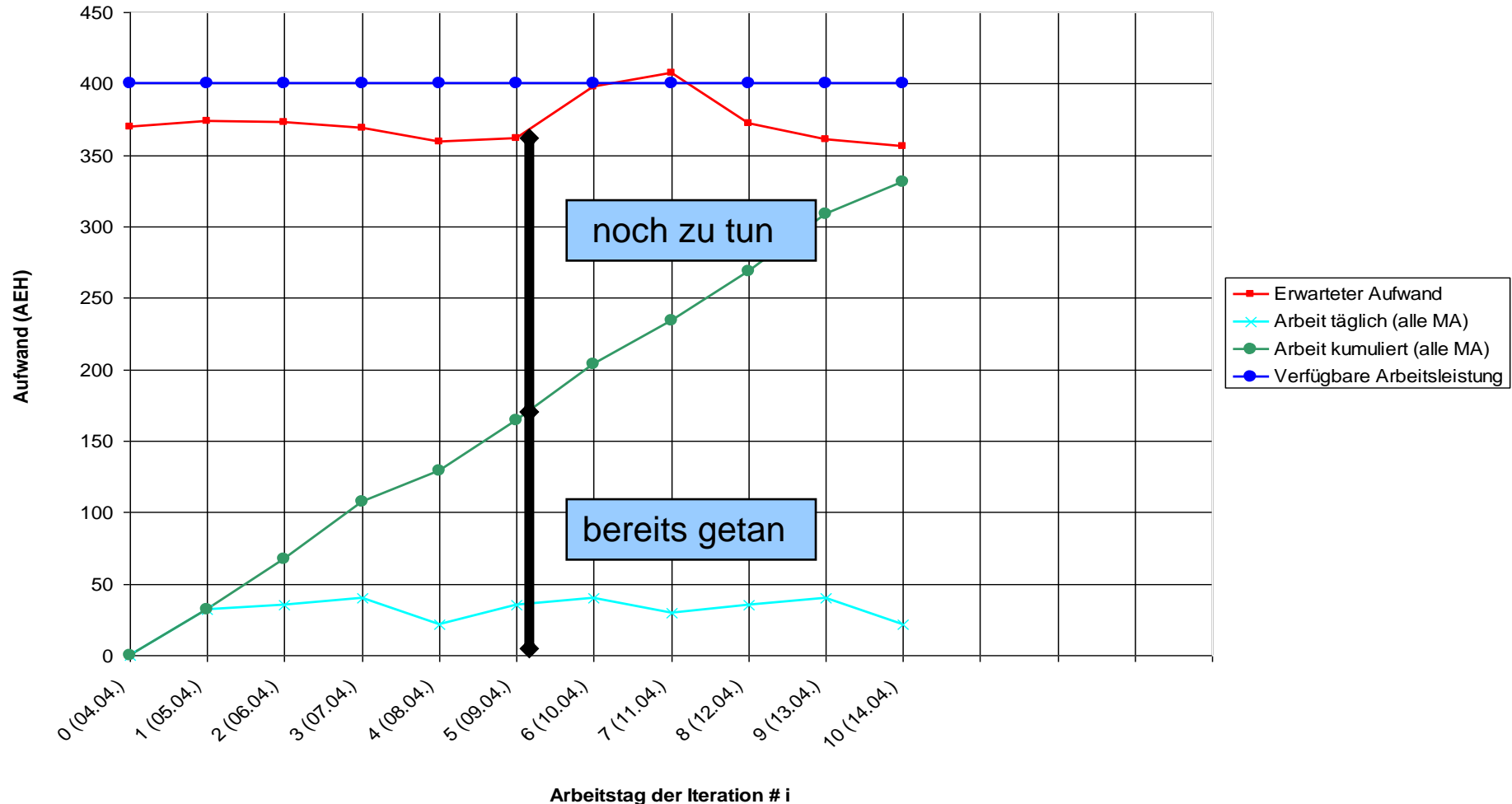
- Fortschritt und Zielerreichungsgrad feststellen
- Management-Overhead analysieren
- Mitarbeiterbelastung & Schätzqualität feststellen
- Prozess der Softwareentwicklung verbessern

Grafische Auswertung:

Fortschrittsdiagramm:
Aufgaben (über 1 Iteration)


FILTER:

















Aufgaben 1 / alle



Gliederung nach Anforderungen für AG-Sicht:

(mind. 1 Anforderung pro Iteration erfüllt, Ampel-Metapher!)

 **Projekt - Iterationen - 1. Iteration**

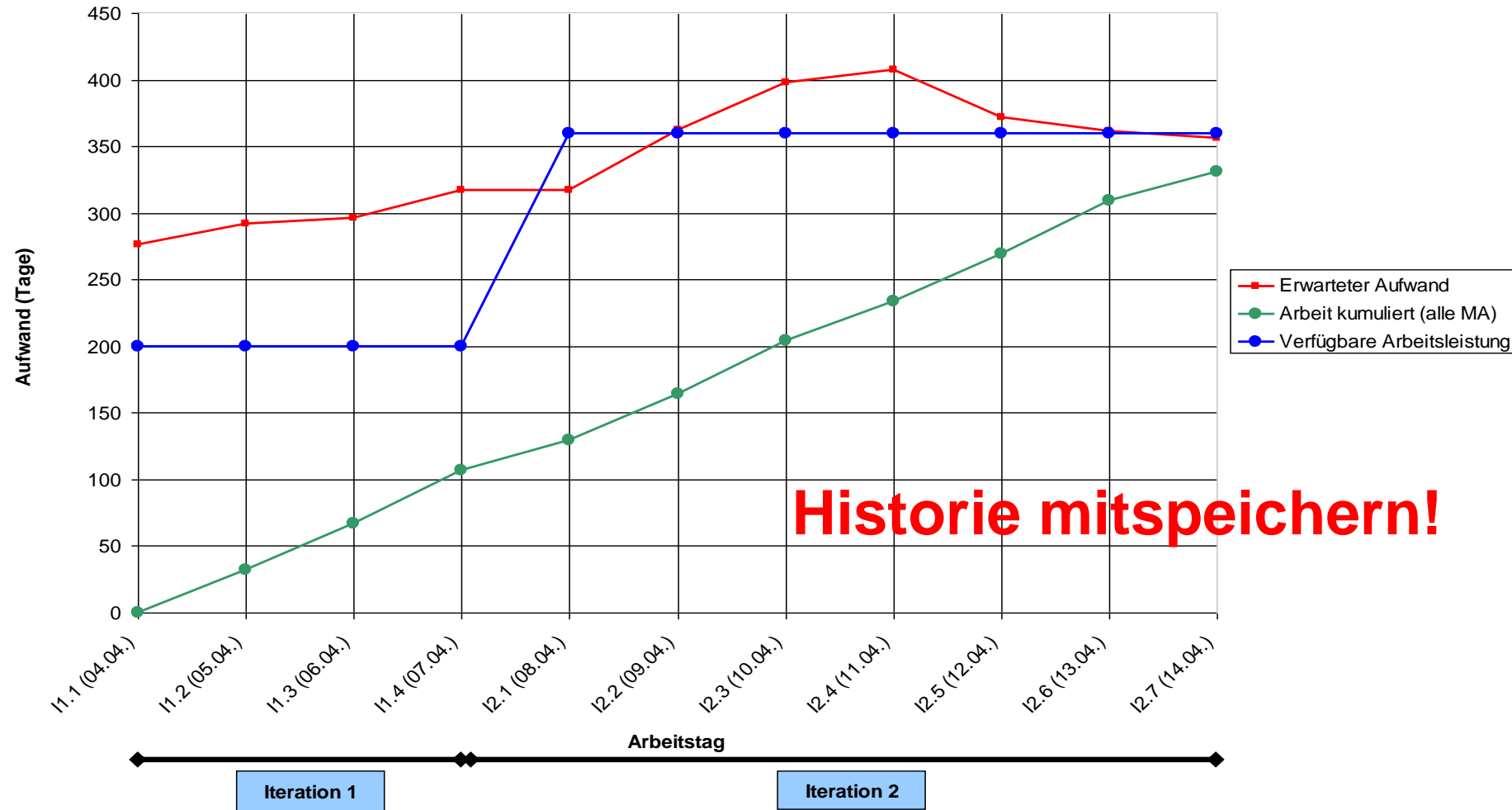
<input type="checkbox"/> Status	Nummer	<input type="text" value="1"/>
<input type="checkbox"/>  7 Erweiterung des Datenmodells	Arbeitstage gesamt	<input type="text" value="7"/>
 7.1 Vereinsnummer hinzufügen	Beginn	<input type="text" value="28.04.2005"/>
 7.2 Bezeichner für Ergebnisse in Datenmodell einfügen	Ende	<input type="text" value="13.05.2005"/>
 7.3 Datenmodell diskutieren	Voraussichtlicher Gesamtaufwand	<input type="text" value="211"/>
<input type="checkbox"/>  1 Aspektbewertung am PDA	- Bisheriger Aufwand	<input type="text" value="193"/>
 1.1 Kriterien- und Aspektbewertung mit Punktegenerator verbinden	Offener Restaufwand	<input type="text" value="18"/>
 1.2 Formular für die Aspekte erstellen	Verfügbare Arbeitszeit	<input type="text" value="168"/>
 1.3 Aspektformulardaten speichern	- Voraussichtlicher Gesamtaufwand	<input type="text" value="211"/>
 1.4 Aspektformulardaten auslesen	Reserve	<input type="text" value="-43"/>
<input type="checkbox"/>  4 Punktegenerator		
 4.1 Logik überlegen		
 4.2 Logik implementieren		
<input type="checkbox"/>  2 Generierung eines Gesamtberichts		
 2.1 Testdaten erstellen		
 2.2 Gesamtbericht testen		
 2.3 Vorhandenen Gesamtbericht bearbeiten		

Grafische Auswertung:

Fortschrittsdiagramm: Anforderungen
(über gesamte Projektlaufzeit)

FILTER:

Anforderungen 1 / alle



Ermittlung des Management-Overhead:

Management-Overhead = Verhältnis zwischen *internen* und *externen* (bzw. *gesamten*) *Aufwänden* in Projekt

Zweck:

Darstellung des Management-Overhead als

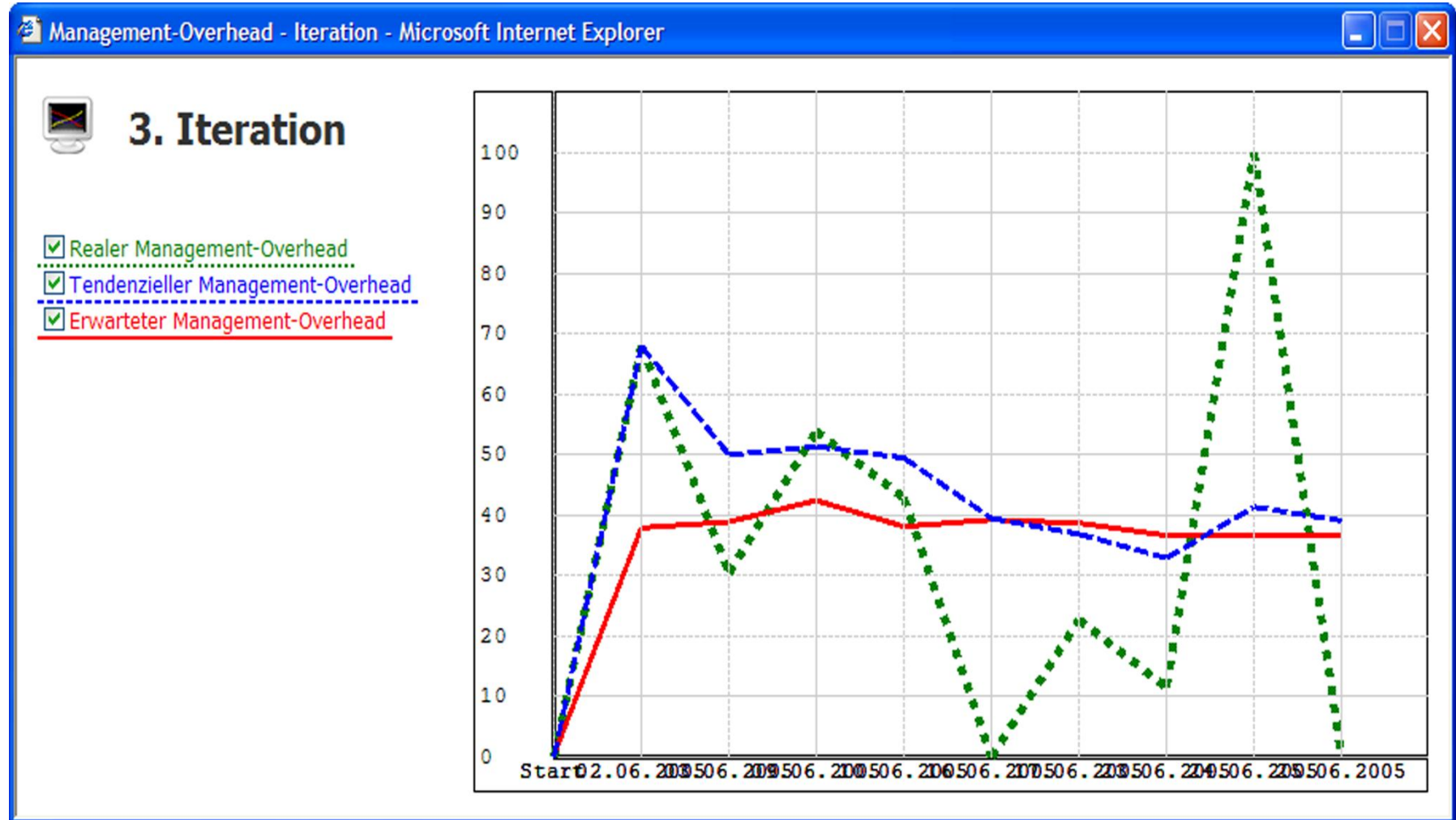
1. notwendig, 2. begründbar, 3. in der Höhe akzeptabel

Arten:

- *realer* Management-Overhead (Gegenwart)
- *tendenzieller* Management-Overhead (Vergangenheit)
- *erwarteter* Management-Overhead (Vergangenheit + Zukunft)

Überprüfen: Management-Overhead (II)

Beispiel:

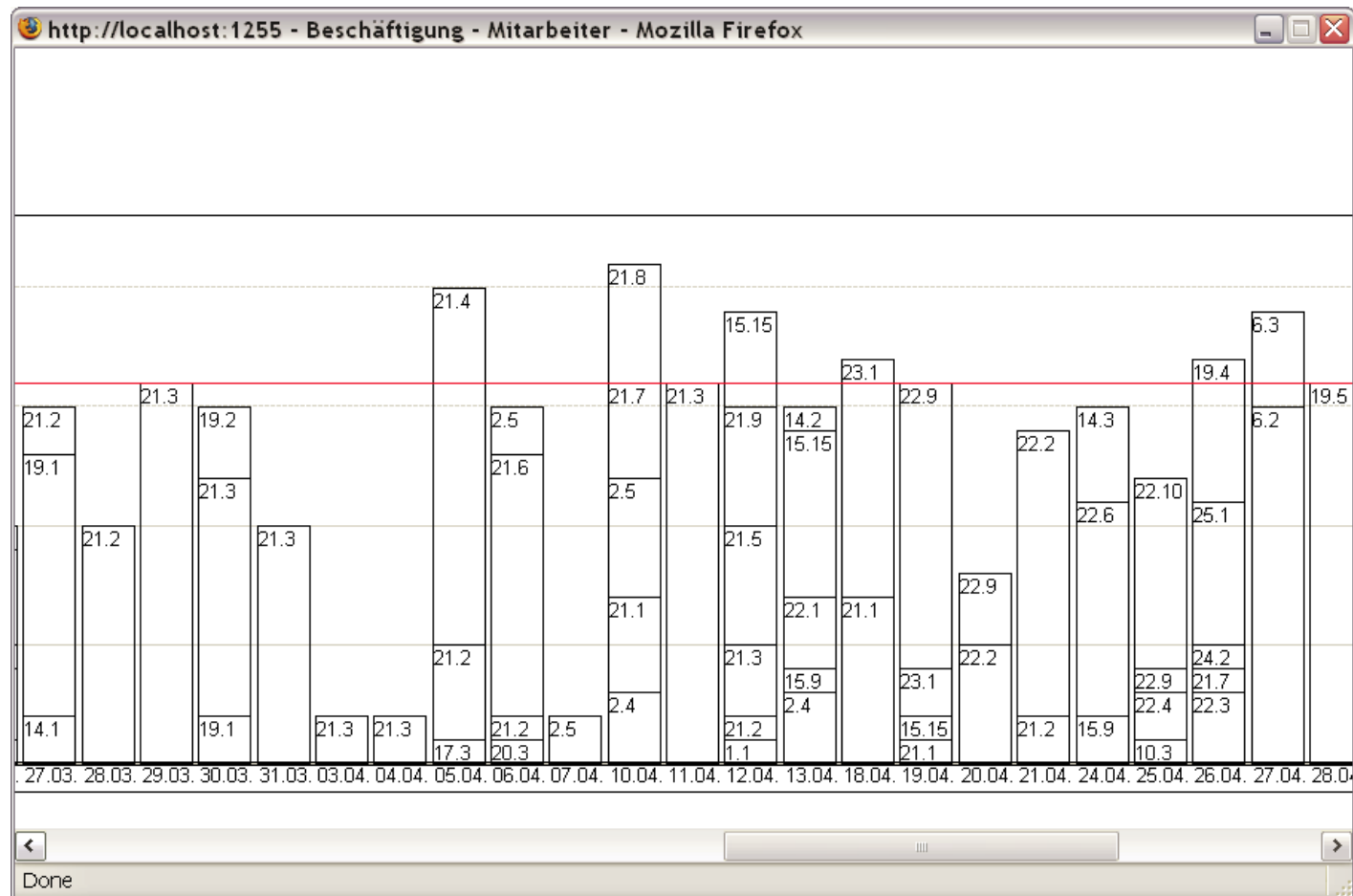


Vorteile eines explizit ausgewiesen Management-Overhead:

- leichteres Erkennen von „overmanaged“ Projekten
- präziser ermittelbare Aufschläge
- genauere Kontrolle aus Managementsicht
- bessere Reflexion vergangener Projektphasen
- besserer Vergleich von Projektphasen und Projekten

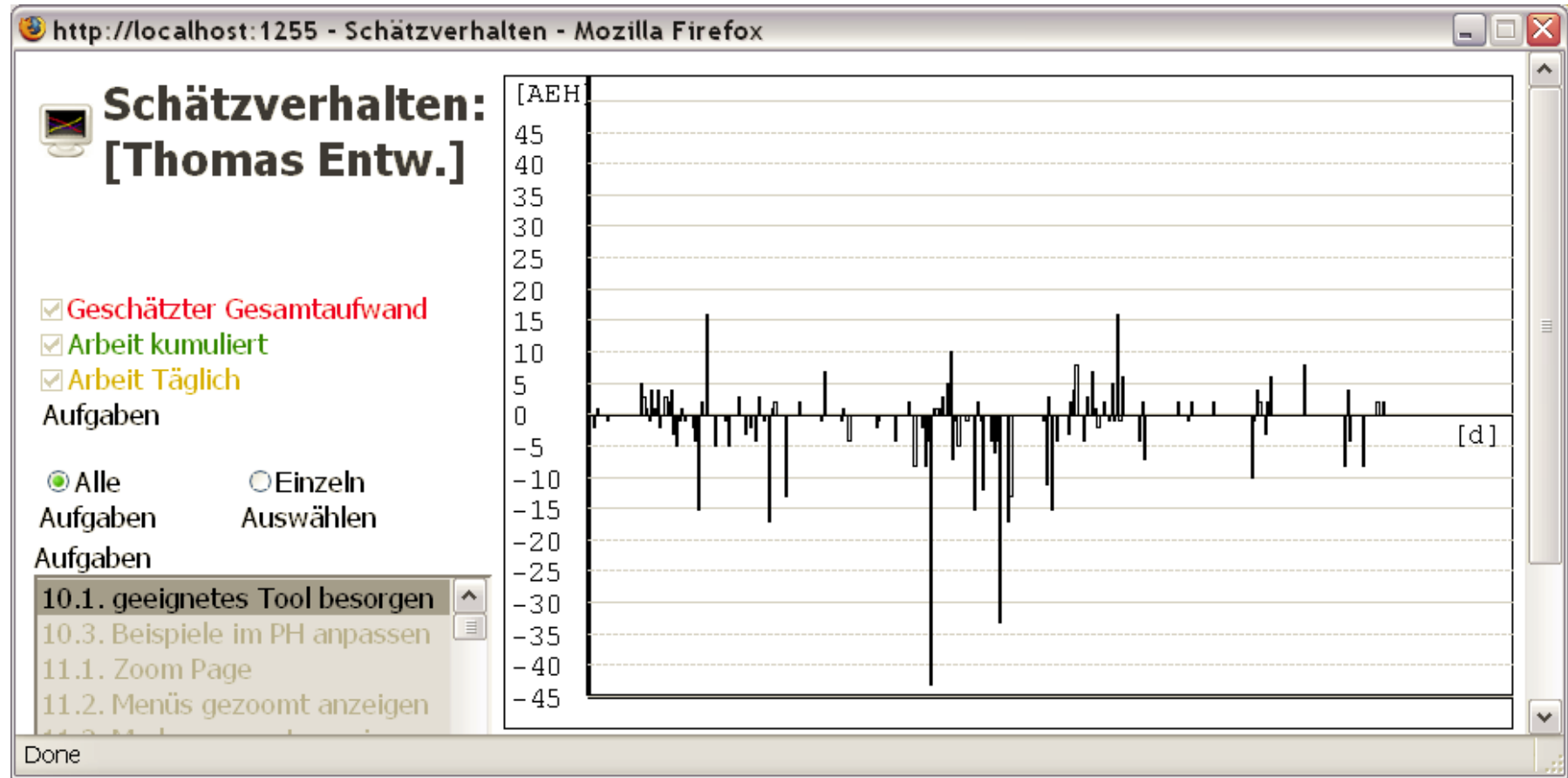
Überprüfen: Auslastung der Mitarbeiter

Sinnvoll inklusive Zuordnung zu den Aufgaben:



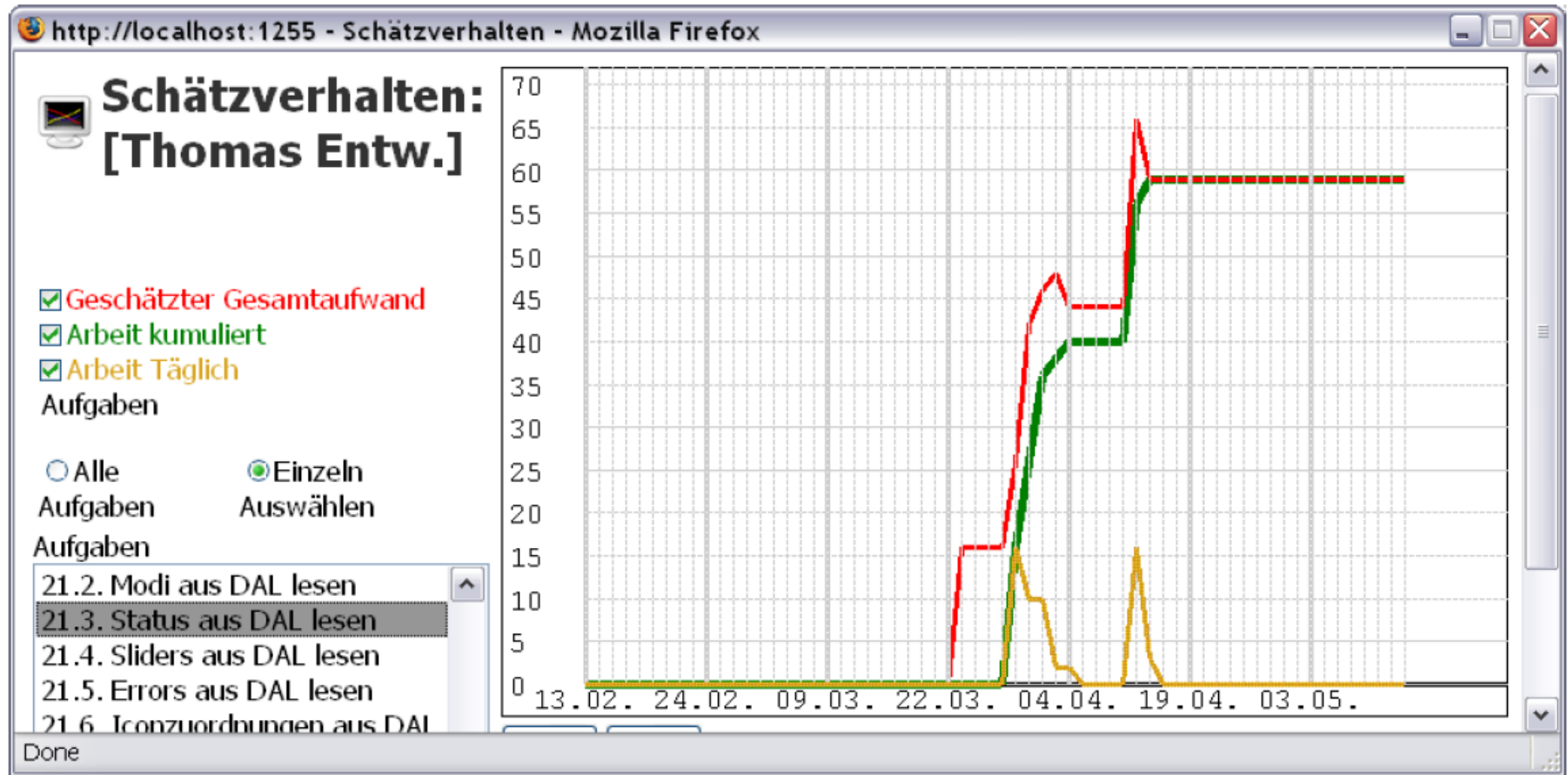
Überprüfen: Analyse d. Schätzvermögens (I)

Sinnvoll nur bei laufender Soll-Planung & Ist-Erfassung!



Überprüfen: Analyse d. Schätzvermögens (I)

Detaillierte Analyse pro Aufgabe:



Rückbetrachtung („Retrospektive“) nach jeder Iteration!

Z.B. in Scrum (erst ab 2004!):

1. Schaffe Sicherheit
2. Sammle die Fakten
3. Finde funktionierende Prozesse
4. Finde nicht funktionierende Prozesse
5. Finde die Kompetenz
6. Priorisiere die Verbesserungsmöglichkeiten