P R O G R A M M I S M U S
NERDJOURNALISMUS
J O U R N A L I E R U N G

Der codende Journalist als Vorhut ?

Christoph Kappes, www.christophkappes.de

Vorweg, These 1:

Programmierenkönnen ist kein Muss, aber es hilft

- Warum niemand muss:
  - -, Lernaufwand sehr hoch bis „Können“, Veränderungstempo
  - -, Spezialisierung folgt der Ausdifferenzierung Gesellschaft
- Nutzen sind grundlegende Kompetenzen
  - Analyse (ἀνάλυσις analysis „Auflösung“)
  - Lösungsstrategien
  - Präzision, Fehlerakzeptanz
  - Konzepte: Objektorientierung, Schnittstellen, Rekursion
  - Ggf auch Dramaturgie, Story-Telling usw
- Gelebtes SW-Engineering nützt (dazu später)
- Thema ist 20 Jahre alt
  - „Entwicklungsredakteur 2013“ = „Konzepter 1993“
  - Gleiche Schmerzen beim Rollen-Umbau Texter -> Konzepter

Software-Engineering als Paradigma

- Journalismus & Programmieren, nicht viel Analogie

Informations-Verarbeitung	
Ungleich:	• Aussagen vs. Befehlsfolgen • „Zielgruppe“ Mensch vs. Maschine • Wortschatz groß vs. klein • Bedeutung komplex vs. einfach • Anschluss vs. Aufruf (?)
Gleich	• Formal & Kreativ (Kreativität, syntaktisch korrekt) • Dekontextualisierung • Anschluss (Programmierer an Programmierer!) • Produkt hat Referenzen auf andere Produkte (formal und inhaltlich) • Strukturen teils stabil, teils beweglich (Lexikon vs. Meldung)
Sinnverarbeitung (Begriffe als Ergebnis von Aufladungen durch Sprechakte) **vs. formale Symbolverarbeitung –** **Kein Platz für Analogie**	

Software-Engineering als Paradigma (2)

- Software als Treiber des Zeitalters (seit <1970)
  - Teilsysteme Wirtschaft Logistik, Automobile, Finanzen, Architektur
  - Ändert Prozesse, Werkzeuge, Kosten, Denken (KPI, SAP, ppt)
- Informationsverarbeitung ieS
  - „Rechnen“, „Kopieren“
  - Komplexe Algorithmen
  - Selbstlernende Algorithmen
  - Filter, Aggregation, Big Data ...

Software-Engineering befasst sich mit systematischer Konstruktion umfangreicher Softwaresysteme

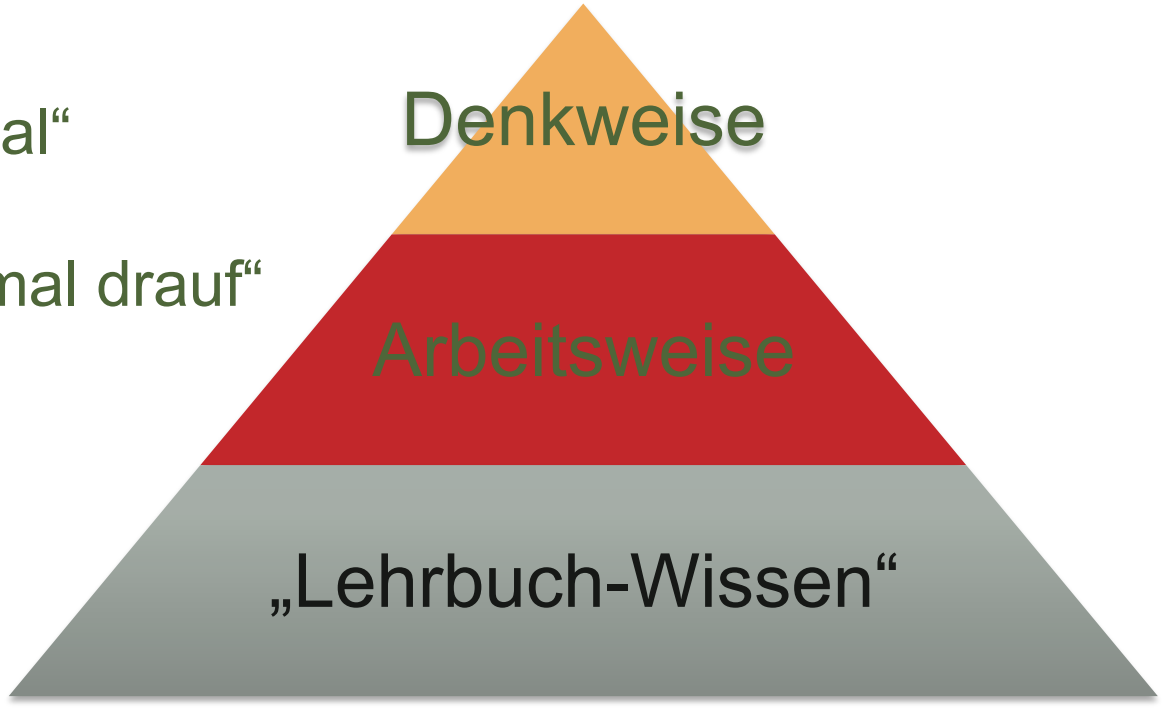
Kann Journalismus vom Software-Engineering lernen?

Software-Engineering als Paradigma (3)

- Software-Engineering...
  - -Methoden: UML, Requirements, Dokumentation
  - -Prozesse: Kollaboration, Rollen, Testverfahren, Deployment...
  - -Ressourcen: Werkzeuge, Versionsmanagement...
- Lehrbuchwissen, Praxis, Denken -> Kulturtechnik

„Fehler sind normal“

„Schau Du noch mal drauf“



These 2: Prozess der Publikationen

- Bisher: Publikation als Resultat, immer wieder
- Neu: Prozess der Publikationen
  - Versionen
  - Alte Publikationen „sedimentieren“ kaum, Änderung wird sichtbarer
  - Wer Seiten entfernt, entfernt Adressen für andere Seiten
  - Stabile Formen: Lexikalisches Wissen, Ontologien, „long tail“, embedded eBooks
  - Wegfall fester Produktionsrhythmen
  - Modifizierte Kopie und Remix
  - Temporäre Formate wie Etherpads
- Software: Versionierung, Lebenszyklus, agile Methoden

Relevante Nachricht ist Differenz –
und die wird im Prozess erzeugt

These 3: Man & Machine verwachsen tiefer

- Bisher:
  - Buch: Menschen selektieren Information
  - Computer: Maschinen selektieren Information (Chartanalyse-Technik, Suchmaschinen-Relevanz, Facebook-Edge, Aggregatoren (!), Pingbacks als Auto-Anschlussverfahren)
- Neu: Hybride soziotechnische Einheit
  - „Mann und Maus“ bilden schon bei Excel eine Einheit, mit vorgefertigten Routinen
  - Nerd-Journalisten steuern mit Maschinen den Informationsstrom (RT!), verbinden Datenpools
  - Nutzer, die interagieren, sind Teil der soziotechnischen Einheit
- Editoren schreiben auf dem Computer ein Buch (Setzkasten), Nerd-Journalisten greifen hingegen in die Software ein
- Notwendige Entwicklung zur Komplexitätsreduktion

These 4: Vom Autor zum Team

- Bisher:
  - Einzelautor
  - Autorenteam
- Künftig (auch):
  - Rollendifferenzierung (z.B. „Tandem-Journalismus“, Impulsgeber, Verstärker, Mega-Hub, Connector, QS-Checker...)
  - Umgekehrt: „Flüssiges Schreiben“,Crowdwriting, siehe Google-Docs-Beobachtungen, „Tod des Individuums“
  - Organisation ist entgrenzt und wird durch Plattform geklammert
  - Neue Innovations- und Entwicklungsprozesse, neue QS-Prozesse
- Grund: Teams sind schneller, ausfallsicherer, korrigieren Fehler
- „Content-Repository als digitale Allmende“, Muster werden kommen

These 5: Die Quellen sind der Code

- Bisher: Output „das Werk“
- Künftig: Quellen sind Teil des Werkes
  - Quellen-Inhalte
  - Referenzen auf Quellentexte, Reputation von Quellen etc
  - Quellen-Personen
- Software:
  - Quellcode schafft Sicherheit (vgl. backdoors...)
  - Quellcode schafft Mehrwert (Vertiefung, Bearbeitung, Verstehen)
- Texte als Runtime /“Laufzeit-Texte“
  - Nur was gute Quellen hat, compiliert gut
  - Modifizierte Agenturmeldungen sind *Branches*

Journalist natürlich nicht auf Compiler reduzieren, Sprache ist nicht formaler Code, sondern komplex

These 6: Transparenz der Redaktion

- Bisher Tendenz zur Black Box (Redaktions-Blogs sind schon Indikation für ihre Notwendigkeit)
- Künftige Tendenz zur Öffnung
  - Personen und Details treten zunehmend hervor
  - Entgrenzung der Organisationen wird folgen, sobald im Informationsökosystem maschinelle Abrechnung möglich ist (bis hin zu RTB für Texte), Ad-Hoc-Teamkonfiguration
  - Transparenz kostet immer weniger, bringt aber dem Leser Nutzen
- Software: Open Source; dagegen: Knowhow-Bunker der IT-Firmen
- (Funktion der Massenmedien führt zu Öffnung, da Gegenreaktion auf immer mehr „Intrusion“)
- (Code ist ja im Web bereits offen und zieht immer mehr nach)

These 7: Fehlertoleranz

- Bisher: Gegenseitige Beobachtung führt zu ständigem Kritikmodus der Massenmedien, gesteigert durch Publikum mit Social Media
- Künftig:
  - Fehleranfälligkeit steigt, das wird bekannt (was messbar wäre mit einem Crawler), Fehlertoleranz steigt
  - Fehler werden behoben, weil sie normal sind

- Software: Bug-/CR-Management als permanenter Prozess, Fehler sind nur Abweichung von SOLL

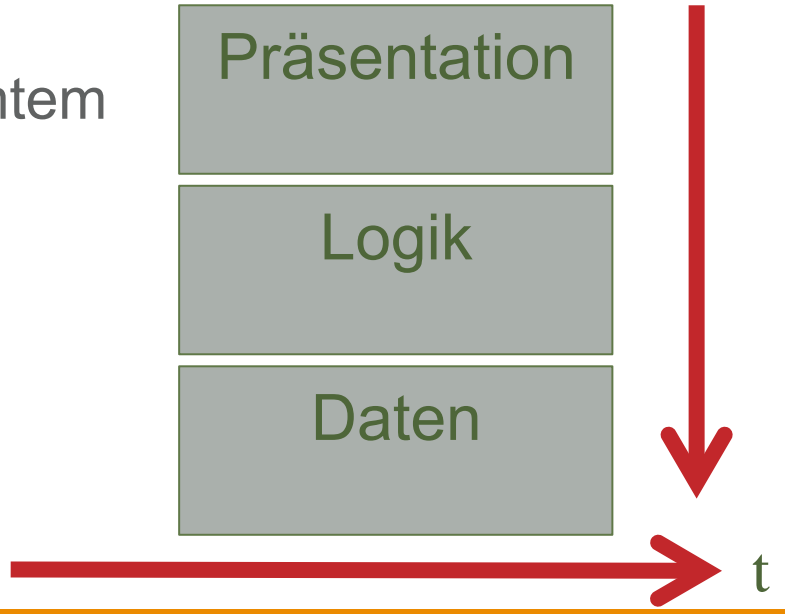
These 8: Journalismus messbar

- Bisher: Käufe (Abos, Einzelkäufe) der Vergangenheit
- Künftig:
  - Messung der Resonanz
  - Messung der Nachfrage (Suchaufkommen, Ad-Preise, Rohdaten wie Grippewelle)
  - Messung der Nutzung/Nicht-Nutzung (vgl. Bookanalytics)
  - Permanente Rückkopplung (zwischen z.B. SM und Search)
- Software: Messung Codequalität, Auswertung Incidents im Support, CRs sind Nachfrage bzw. Requirements

- Journalismus ist wohl letzte Wirtschaftseinheit, die messbarer wird (es verbleiben Kochen, Kunst und Kirche)
- Zahlen sind kein Indiz für kulturelle Relevanz; es zählt aber der zeitliche Vergleich vorher/nachher

These 9: Verschmelzung Werk und Werkzeug

- Bisher: Schreiben auf der Oberfläche („digitale Folie“)
- Künftig: Ausdehnung auf tiefere Schichten
  - Auch Logik und Daten unter der Folie haben Autoren, sind in Bewegung
  - Wechselwirkung zwischen Folie und Logik/Daten
- Backend-Funktion macht Frontend-ERFAHRUNG, ERLEBBARER
- Analog Musiker mit selbstgestimmtem Instrument, Maler mit Leinwand



Schlussworte

- Journalismus und Programmieren sind am entscheidenden Punkt unähnlich, Gegenstand (Wort ≠ Befehl) und Verfahren
- Software-Engineering befasst sich mit der Technik strukturierter Informationsverarbeitung und enthält gute Anregungen für leistungsfähigeren Journalismus (Methodik, Praxis, Denkweise)

- Eine Revitalisierung des Journalismus ist nötig und erhält von hier einen guten Impuls
- Daten sind nicht Öl, aber der empirische Ausgangspunkt für das Hinterfragen von Legenden, Umdeutungen und Aufladungen
- Programmieren hilft, aber Soziologie, Politologie, Wiwi etc. "first", weil der Gegenstand des Journalismus Gesellschaft, Poltiik, Wirtschaft sind

Danke.

Thesen 1-9:

1. **Programmierenkönnen ist kein Muss, aber es hilft**
2. **Von der Publikation als Resultat - zum Prozess der Publikationen** (und „fertig“ gibt es nicht, panta rhei)
3. **Mensch und Maschine verwachsen stärker**
4. **Vom Autor zum Team (Kollaboration)**
5. **Die Quellen sind der Code**
6. **Transparenz der Redaktion**
7. **Fehlertoleranz**
8. **Journalismus messbar**
9. **Wo der Druck Autor und Drucker trennte, wächst durch „Nerdjournalismus“ der Kreative wieder tief mit dem Werk zusammen**

hmmm

- Spiegel Online sieht nur aus wie eine Publikation
- Aus anderer Perspektive (ohne das Buch-Paradigma)
  - Suche ermöglicht nichtlinearen Zugriff
  - Links durchbrechen die Hierarchie der Inhalte
  - URL als ubiquitäre Adresse