



Der Turmbau zu Babel

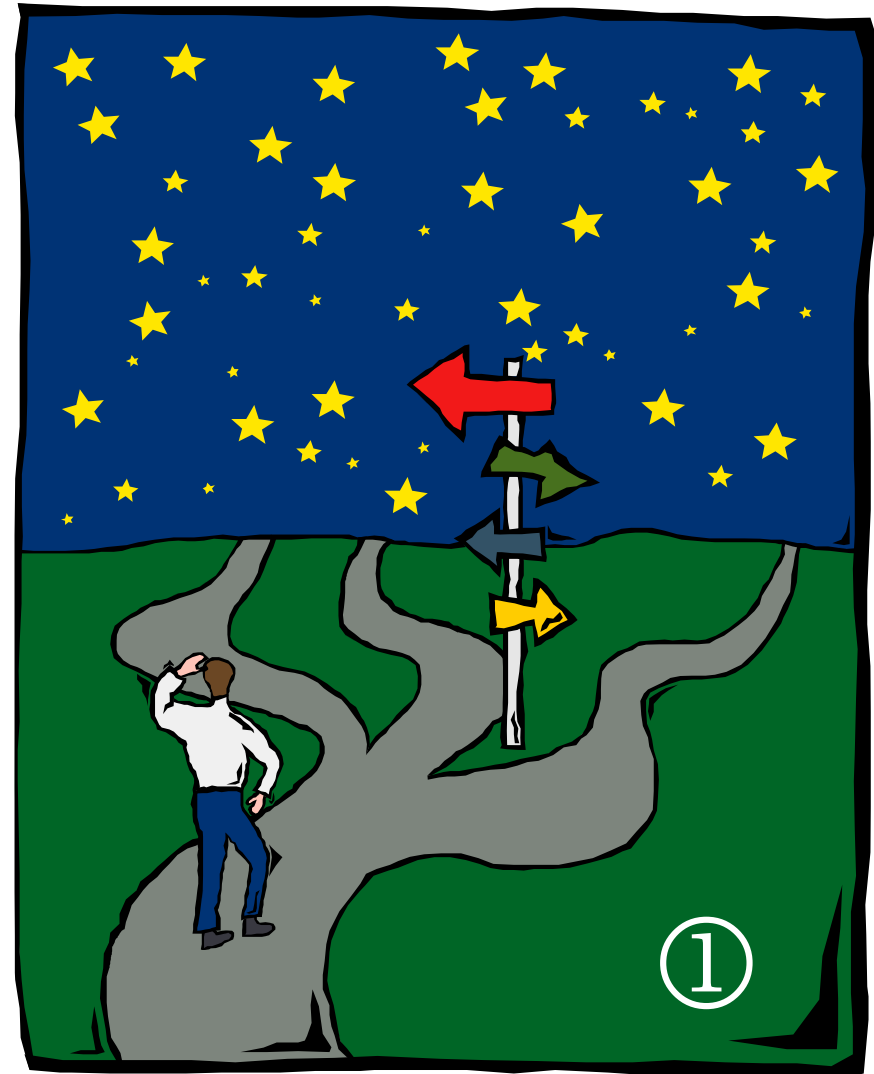
Grenzen der Software-Architektur

Prof. Heinz Züllighoven
Dr. Carola Lilienthal
Universität Hamburg
C1 WPS GmbH

Irritation: Die Situation wird komplizierter



- Die Unterstützung der Geschäftsprozesse durch IT nimmt stetig zu
 - Die Geschäftsprozesse in und zwischen Unternehmen werden umfangreicher
 - Software-Landschaften und IT-Projekte werden größer und umfassender
- **Wir können wir aus dieser Komplexitätsfalle herauskommen?**

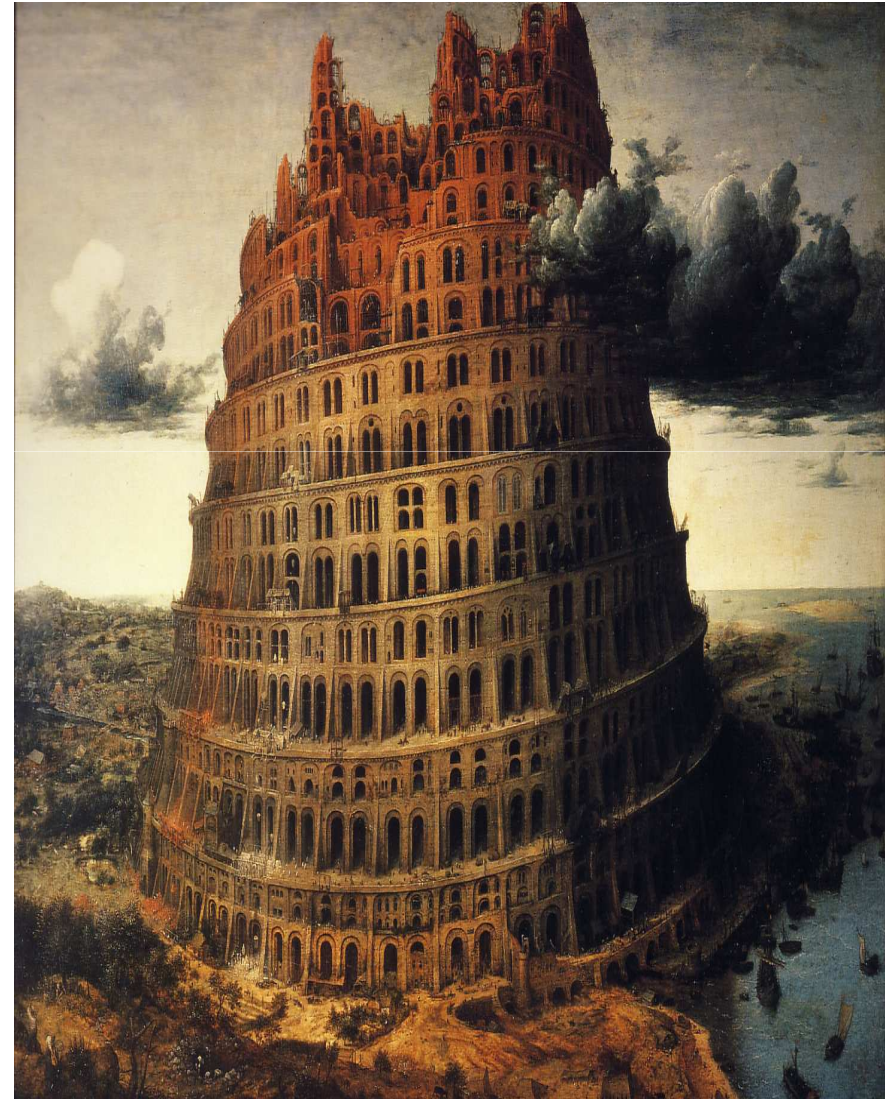


Hoffnung: Wir werden Software mit Hilfe der Theorie
in den Griff bekommen



Die Irritationen

- Korrektheit im Kleinen
Wie korrekt können wir programmieren?
- Vollständiges Testen
Wieviel können wir testen?
- Korrektheit im Großen
Lassen sich Programme formal fassen?





Axiome und Schlussregeln des Hoare Kalkül

Seien P, Q, R Zusicherungen

Leere Anweisung: Axiom

$$A_{\text{Skip}}: \quad \{P\} \text{Skip} \{P\}$$

Zuweisung: Axiomschema (Variable x, Ausdruck e)

$$A_{\text{Assignment}}: \quad \{Q [x \leftarrow e]\} x := e \{Q\}$$

Sequenz: Inferenzregel (Anweisungen a, b)

$$I_{\text{Compound}}: \quad \frac{\{P\} a \{Q\}, \{Q\} b \{R\}}{\{P\} a ; b \{R\}}$$

Alternative: Inferenzregel (Anweisungen a, b; Bedingung c)

$$I_{\text{Conditional}}: \quad \frac{\{P \text{ and } c\} a \{Q\}, \{P \text{ and not } c\} b \{Q\}}{\{P\} \text{if } c \text{ then } a \text{ else } b \{Q\}}$$

Wiederholung: Inferenzregel (Anweisung b, Bedingung c, Zusicherung: "Invariante" I)

$$I_{\text{Loop}}: \quad \frac{\{I \text{ and } c\} b \{I\}}{\{I\} \text{while } c \text{ do } b \{I \text{ and not } c\}}$$



Axiome und Schlussregeln des Hoare Kalkül

Seien P, Q, R Zusicherungen

Leere Anweisung: Axiom

$$A_{\text{Skip}}: \quad \{P\} \text{ Skip } \{P\}$$

Das pragmatische Argument (1):

Klassische Ansätze sind für reale Systeme nicht in realer Zeit einsetzbar
z.B. Hoare Kalkül auf Statement Ebene ist extrem aufwändig
Maschinelle Beweiser sind nur von Spezialisten mit hohem Zeitaufwand zu benutzen

Wiederholung: Inferenzregel (Anweisung b, Bedingung c, Zusicherung: "Invariante" I)

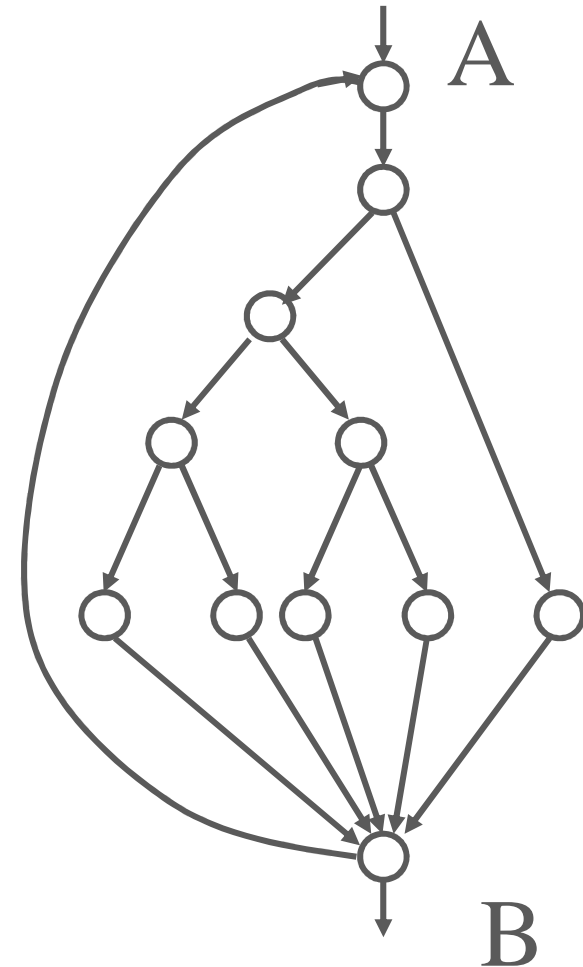
I_{Loop} :

$$\frac{\{I\} \text{ if } c \text{ then } a \text{ else } b \{Q\}}{\{I\} \text{ while } c \text{ do } b \{I \text{ and not } c\}}$$

Anspruch: Vollständig Testen?



- Ein einfaches Programm soll getestet werden, das aus vier Verzweigungen (IF-Anweisungen) und einer umfassenden Schleife besteht und somit fünf mögliche Wege im Schleifenrumpf enthält.
- Unter der Annahme, dass die Verzweigungen voneinander unabhängig sind und bei einer Beschränkung der Schleifendurchläufe auf maximal 20, ergibt sich folgende Rechnung:
$$5^1 + 5^2 + \dots + 5^{18} + 5^{19} + 5^{20}$$
- Wie lange dauert das Austesten bei 100.000 Tests pro Sekunde?



Wirklichkeit: Vollständig Testen?



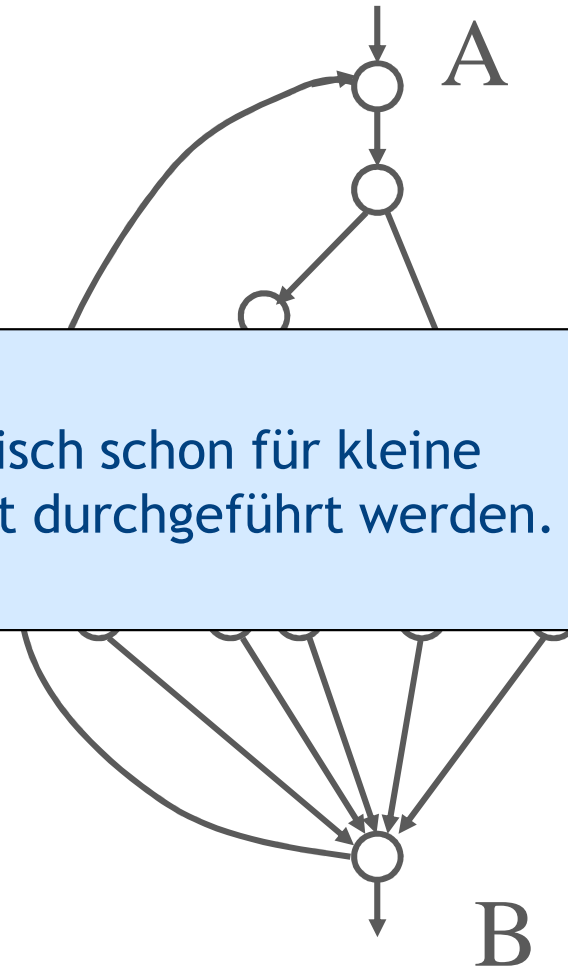
- Ein einfaches Programm soll getestet werden, das aus vier Verzweigungen (IF-Anweisungen) und einer umfassenden Schleife besteht und somit fünf mögliche Wege im Schleifenrumpf enthält.

- Das pragmatische Argument (2)

Vollständige Tests „explodieren“ kombinatorisch schon für kleine Software und können nicht in endlicher Zeit durchgeführt werden.

$$5^1 + 5^2 + \dots + 5^{10} + 5^{17} + 5^{20}$$

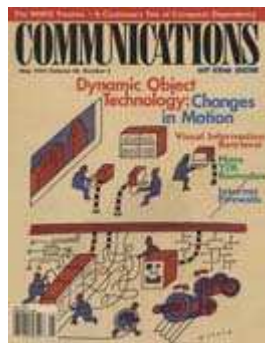
- Wie lange dauert das Austesten bei 100.000 Tests pro Sekunde?
- Es sind **119.209.289.550.780 Testfälle**
Dauer: ca. 38 Jahre



Das theoretische Argument von *Peter Wegner* Why interaction is more powerful than algorithms



- Turing machines extended by addition of input and output actions that support dynamic interaction with an external environment are called “interaction machines.”
- *Interaction-machine behavior is not reducible to Turing-machine behavior.*
- *Turing machines cannot handle the passage of time or interactive events that occur during the process of computation.*
- *Input streams of interaction machines are not expressible by finite tapes, since any finite representation can be dynamically extended by uncontrollable adversaries.*



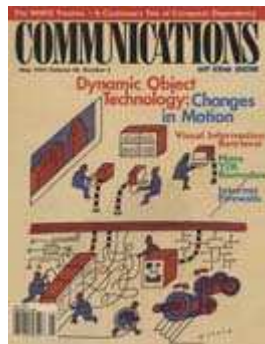
Das theoretische Argument von *Peter Wegner* Why interaction is more powerful than algorithms



- Turing machines extended by addition of input and output actions that support dynamic interaction with an external environment are called “interaction machines.”
- *Interaction-machine behavior is not reducible to Turing-machine behavior.*

■ Das theoretische Argument (1):

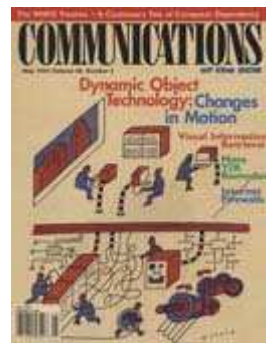
- Software, die mit ihrer Umgebung interagiert, ist mächtiger als die Turing Maschine (sie entspricht Turing Maschinen mit Orakel).
- Damit müssen die mathematischen Ansätze der Berechenbarkeit und Formalisierung revidiert werden.



Das theoretische Argument von *Peter Wegner* Why interaction is more powerful than algorithms



- A system satisfies its requirements if it supports specified modes of use, even though correct behavior for a given mode of use is not guaranteed and complete system behavior for all possible modes of use is unspecifiable.
- Though correctness of programs under carefully qualified conditions can still be proved, result checking is needed during execution to verify that results actually obtained are valid. Techniques for systematic online result checking will play an increasingly important practical and formal role as a supplement to off-line testing and verification.



Communications of the ACM
Volume 40 , Issue 5 (May 1997)
Pages: 80 - 91

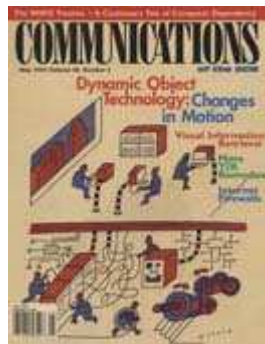
Das theoretische Argument von *Peter Wegner* Why interaction is more powerful than algorithms



- A system satisfies its requirements if it supports specified modes of use, even though correct behavior for a given mode of use is not guaranteed and complete system behavior for all possible modes of use is unspecifiable.
- Though correctness of programs under carefully qualified conditions can

Das theoretische Argument (2):

- Klassische Korrektheitsansätze sollten bei interaktiver Software durch die Überprüfung der Ergebnisse ersetzt (ergänzt) werden.
- Damit gewinnen Tests und die Prüfverfahren zur Laufzeit eine zunehmende Bedeutung.





Was können wir tun?

→ Qualitätssicherung in den Fokus von IT-Projekte und Architektur stellen

→ Pragmatische Ansätze zur Qualitätssicherung:

- Automatisiertes Testen für sinnvoll erscheinende (Grenz-)fälle auf allen Ebenen (Klassen, Packages, Komponenten)
- Programmierrichtlinien und Messung der Code Qualität
- Architekturstile, Mustersprachen und Architekturanalyse

Architekturstile und Mustersprachen - Einführen und Entwickeln

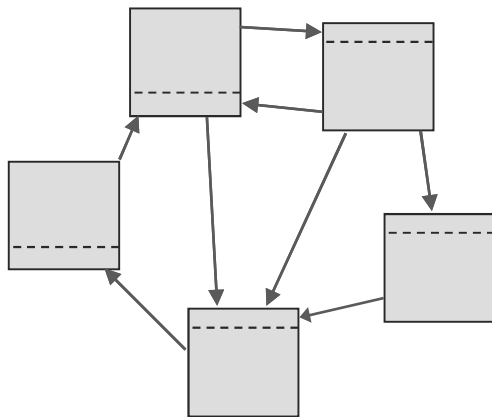


Definition

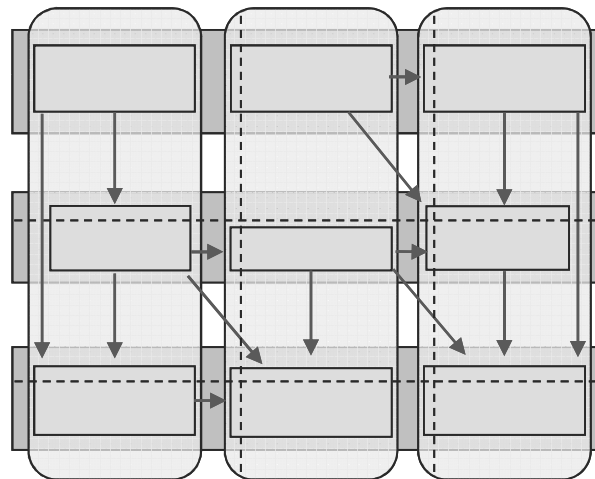
„Ein Architekturstil ist eine prinzipielle Lösungsstruktur, die für ein Softwaresystem durchgängig und unter weitgehendem Verzicht auf Ausnahmen angewandt werden sollte.“

[Reussner et al. 2006]

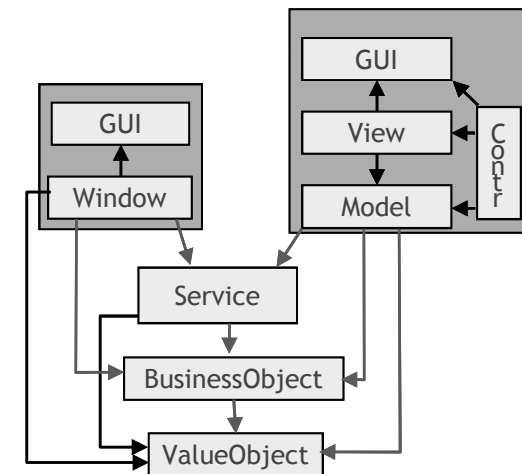
Komponentenarchitektur



Schichtenarchitektur

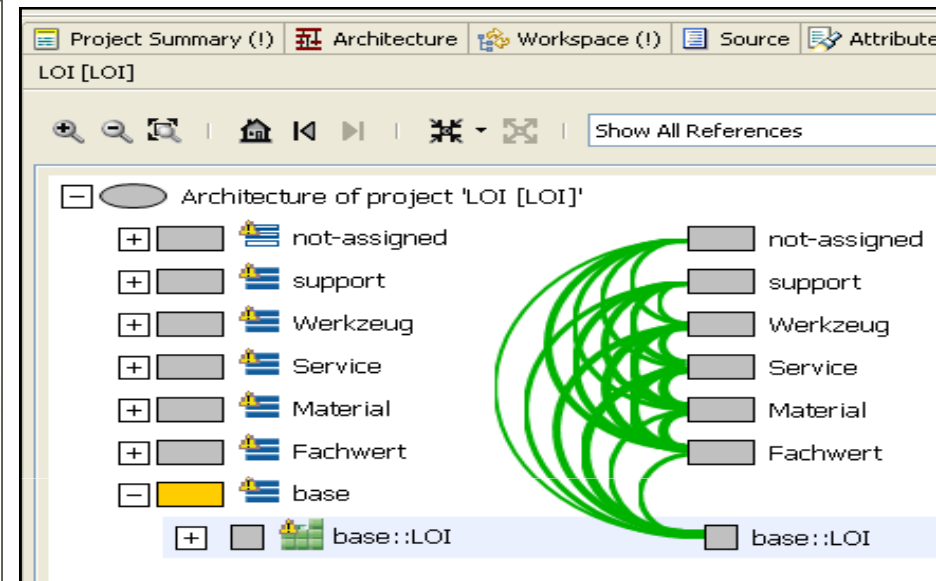
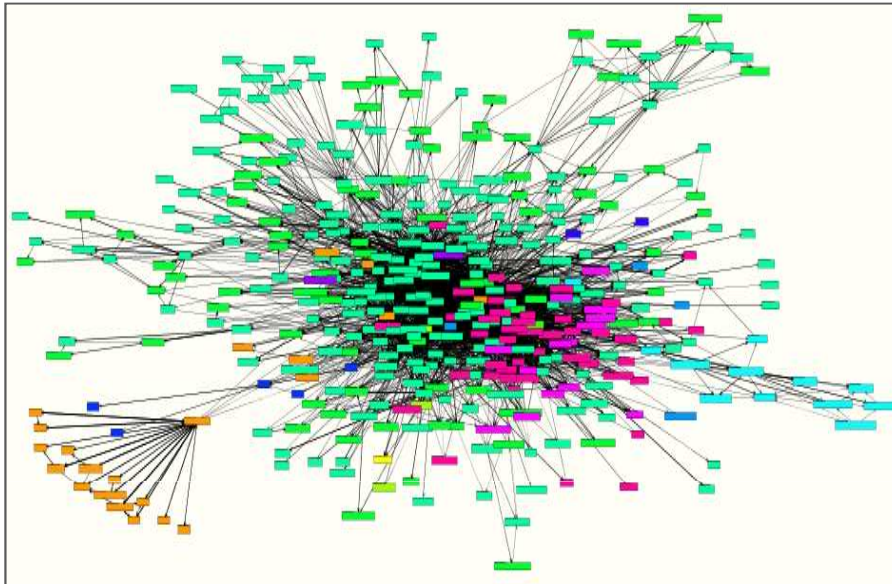


Mustersprachen



Fragestellung

- Welche Muster oder Architekturstile eignen sich für mein Softwaresystem?



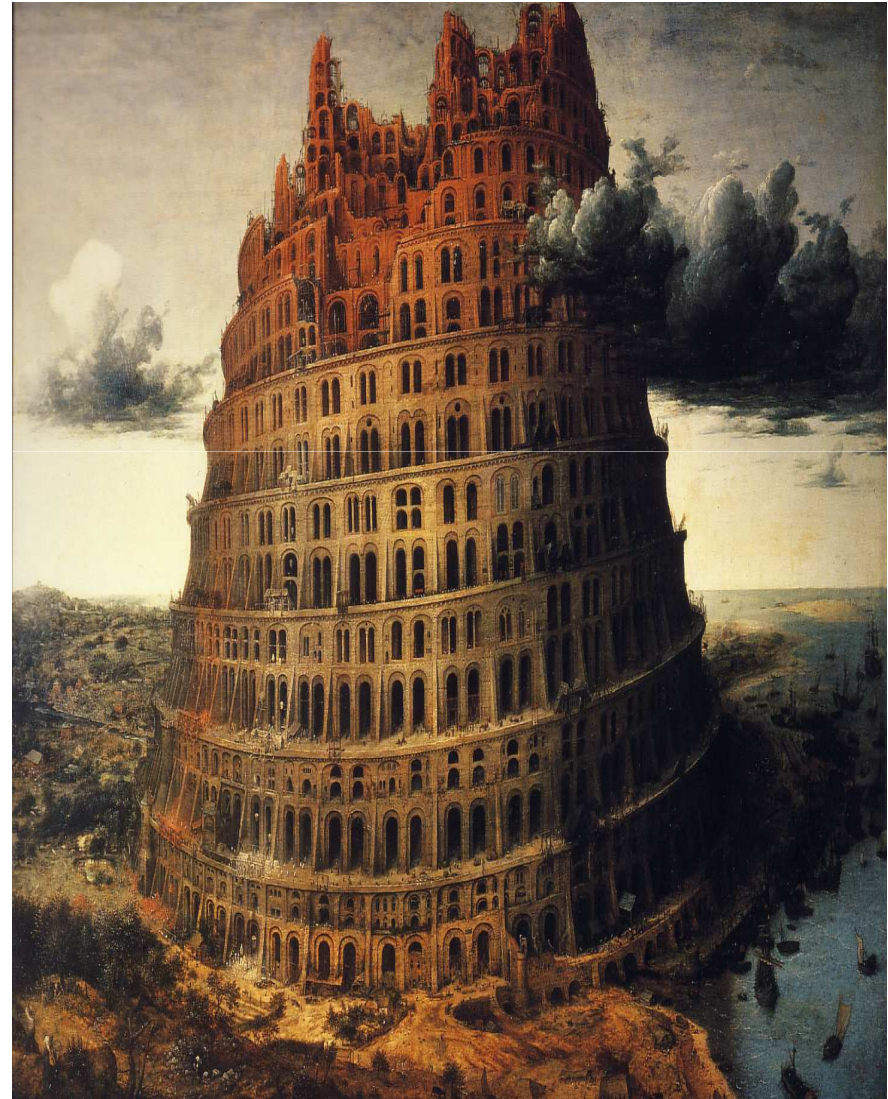
Fragestellung

- Welche Teile des Systems sind besonders groß, komplex und vernetzt?
- Welche Verletzungen von Schichten und Schnittstellen sind vorhanden?
Wie kann man sie auflösen?
- Welche Gefahren für die Qualität der Software und die zukünftige (Weiter)-Entwicklung gehen von den Verletzungen und Anomalitäten aus?



Irritationen

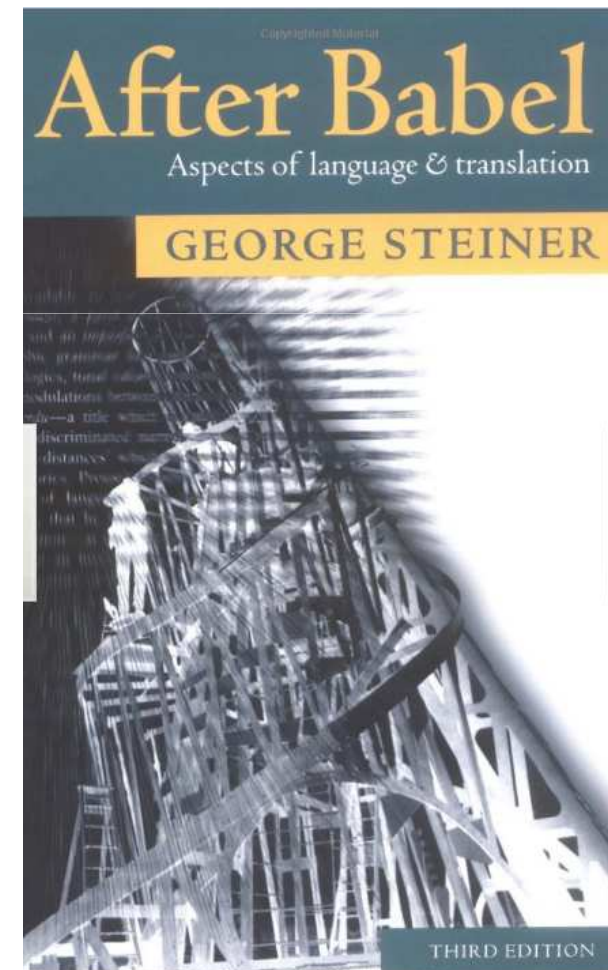
- Lassen sich Anforderungen eindeutig spezifizieren?
- Lassen sich Anforderungen aus Dokumenten extrahieren?
- Lassen sich Anforderungen maschinell erheben?



Wie eindeutig, formal und objektiv ist unsere Sprache?



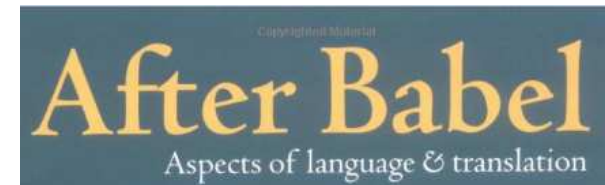
- Der Ansatz, aus gesprochener/geschriebener Sprache eindeutige Anforderungen zu extrahieren und in formal korrekte Aussagensysteme zu übertragen ist umstritten (nicht haltbar?)
- No set of rules, however complete, is sufficient to describe ... the utterances possible in any living language.
- *The great mass of ... words spoken and heard does not fall under the rubric of ‚faculty‘ and truth.*
- *Wir erdichten uns den größten Teil des Erlebnisses*
(Nietzsche, Jenseits von Gut und Böse).



Wie eindeutig, formal und objektiv ist unsere Sprache?



- Der Ansatz, aus gesprochener/geschriebener Sprache eindeutige Anforderungen zu extrahieren und in formal korrekte Aussagensysteme zu übertragen ist umstritten (nicht haltbar?)
- No set of rules, however complete, is sufficient to describe ... the utterances



Zentrale Aussagen:

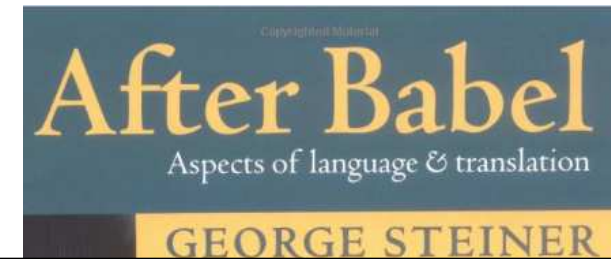
- Eine lebende Sprache kann nicht vollständig formal beschrieben werden.
- Die in einem Sprachraum verständlichen Sätze lassen sich nicht vollständig „generieren“.
- Mehrdeutigkeiten, Unklarheiten, „unkorrekte“ Wörter und Sätze sind konstituierende Merkmale einer Sprache und keine zu vernachlässigenden Randerscheinungen.



Wie eindeutig, formal und objektiv ist unsere Sprache?



- Der Ansatz, aus gesprochener/geschriebener Sprache eindeutige Anforderungen zu extrahieren und in formal korrekte Aussagensysteme zu übertragen ist umstritten (nicht haltbar?)
- No set of rules, however complete, is sufficient to describe ... the utterances possible in any living language.



Zu den Charakteristika menschlicher Sprache gehören:

- Menschen äußern sich nicht, um „Wahrheiten“ oder Fakten zu vermitteln.
- Menschen benutzen Sprache mit Hintergedanken, Vorstellungen, Hoffnungen, Illusionen.
- Menschen können Erfahrungen, Interpretationen und Einbildungen kaum trennen.





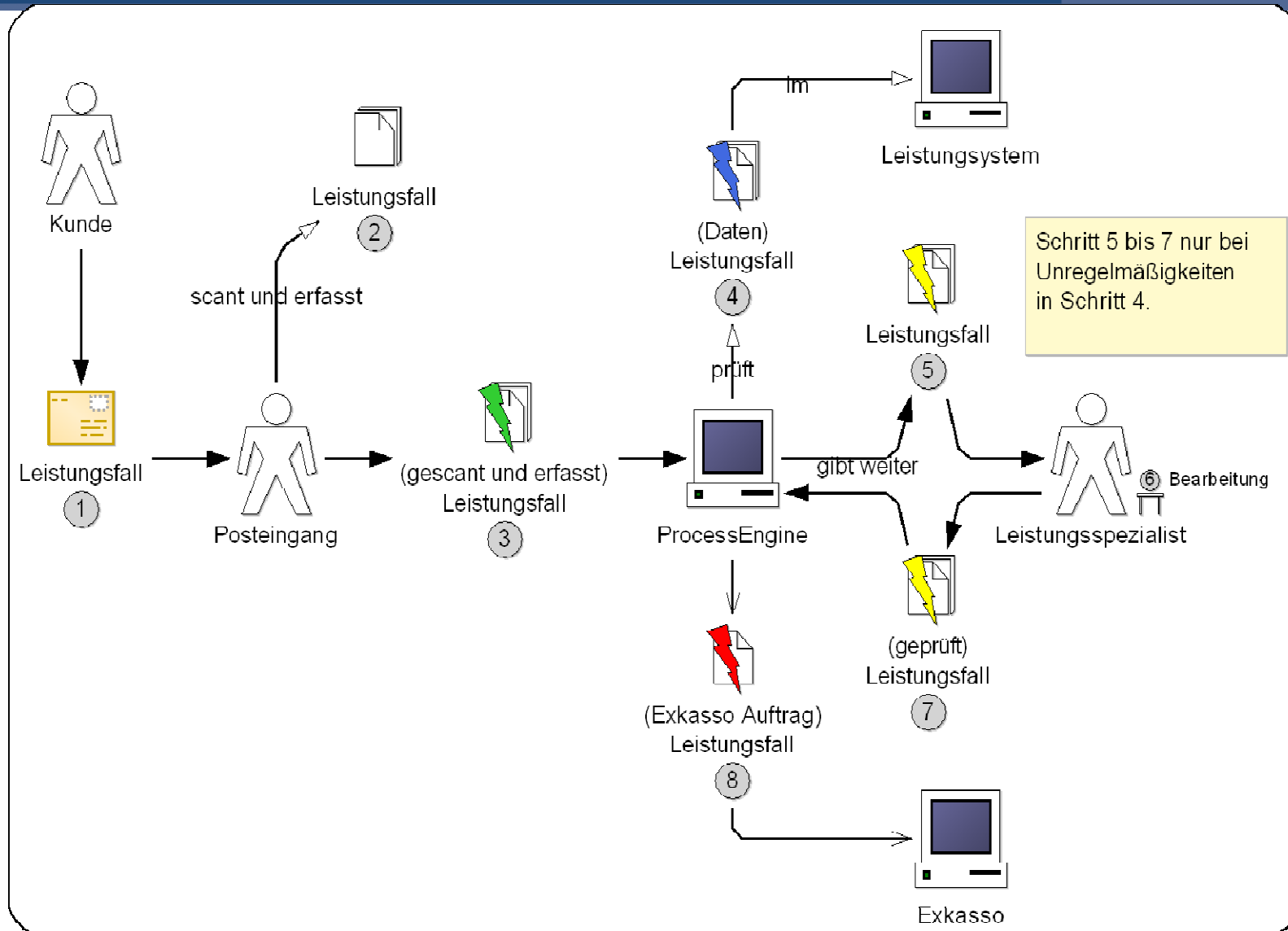
Was können wir tun?

→ **Fachliche Darstellungsmittel** mit Anwendern iterativ ausarbeiten

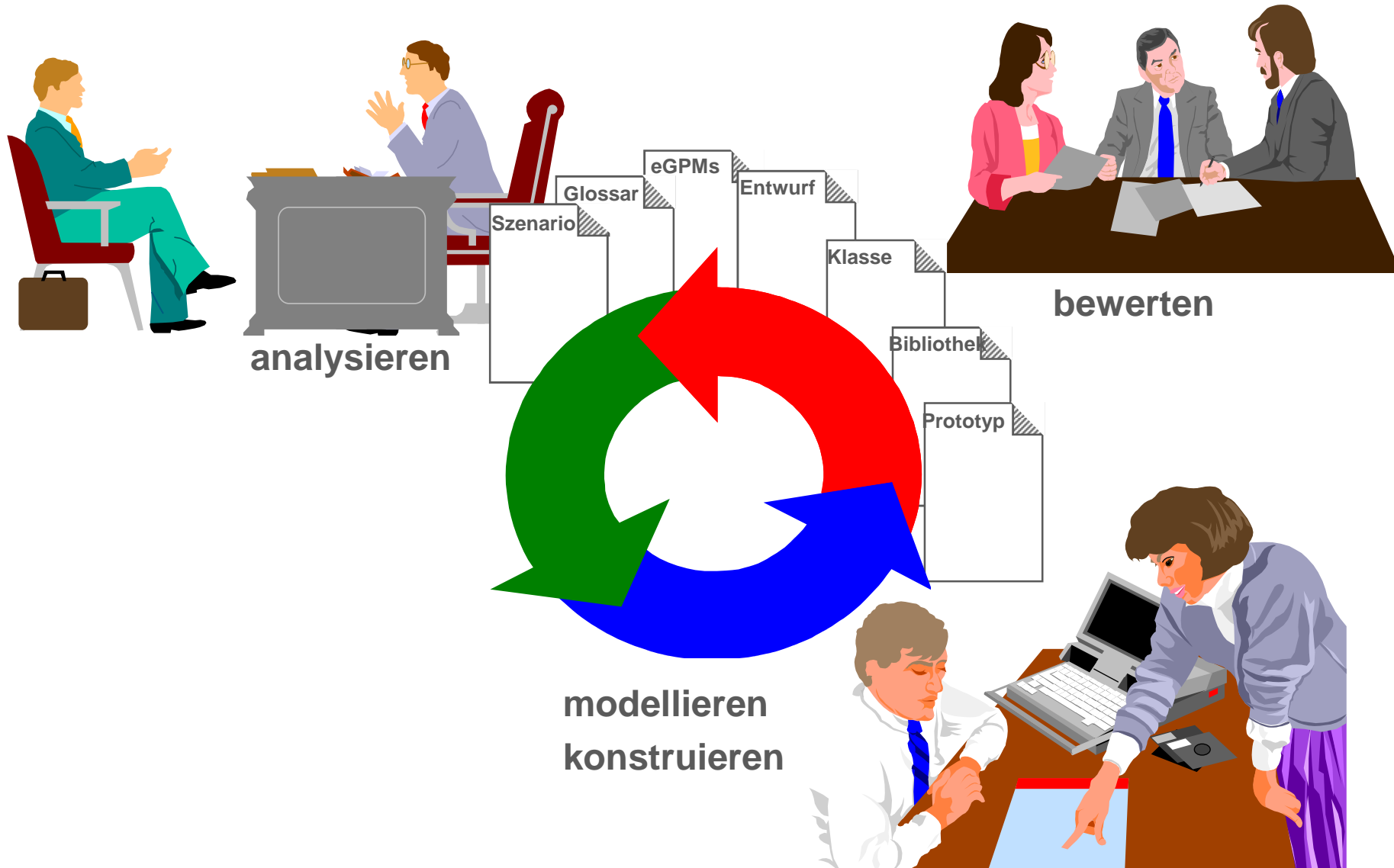
→ Pragmatische Ansätze zur Anforderungsermittlung und Geschäftsprozessmodellierung:

- Die Sprache des Anwenders als Ausgangspunkt
- Darstellungsmittel, die der fachlichen Denkweise entsprechen (im einfachsten Fall: Szenario-Techniken)
- Keine vollständige Vorab-Modellierung im Projekt, sondern iterativ nach Notwendigkeit und wachsender Einsicht
- Autor-Kritiker-Zyklen als Mittel der Verständigung

Fachliche Denkweise in den Darstellungsmittel



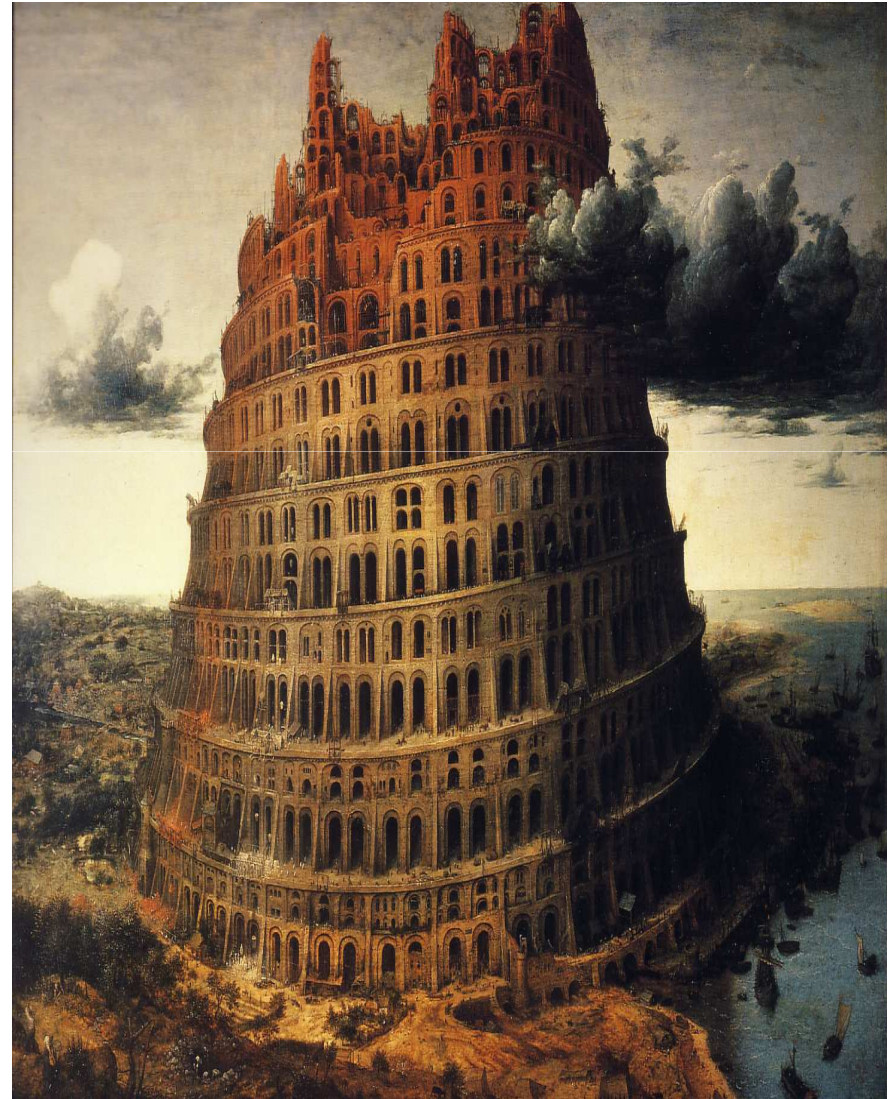
Autor-Kritiker-Zyklus als Mittel der Verständigung

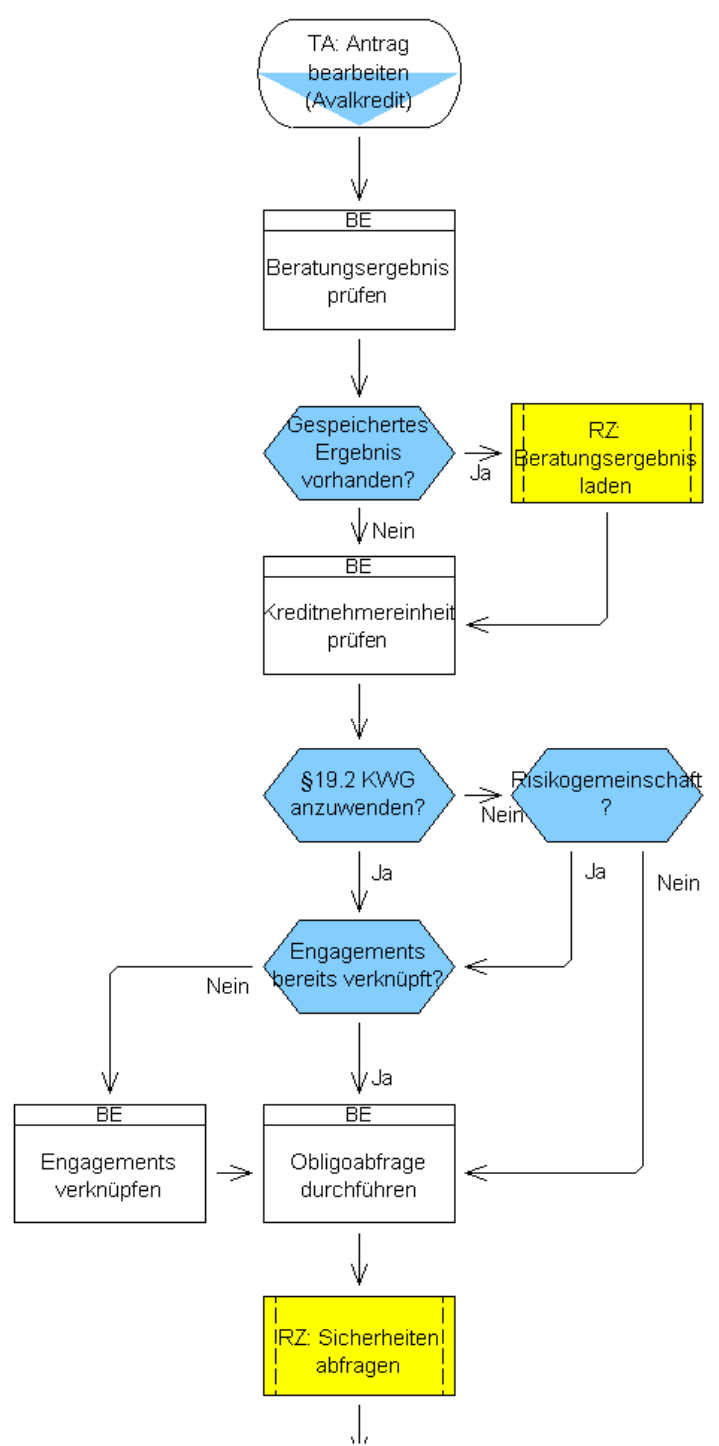
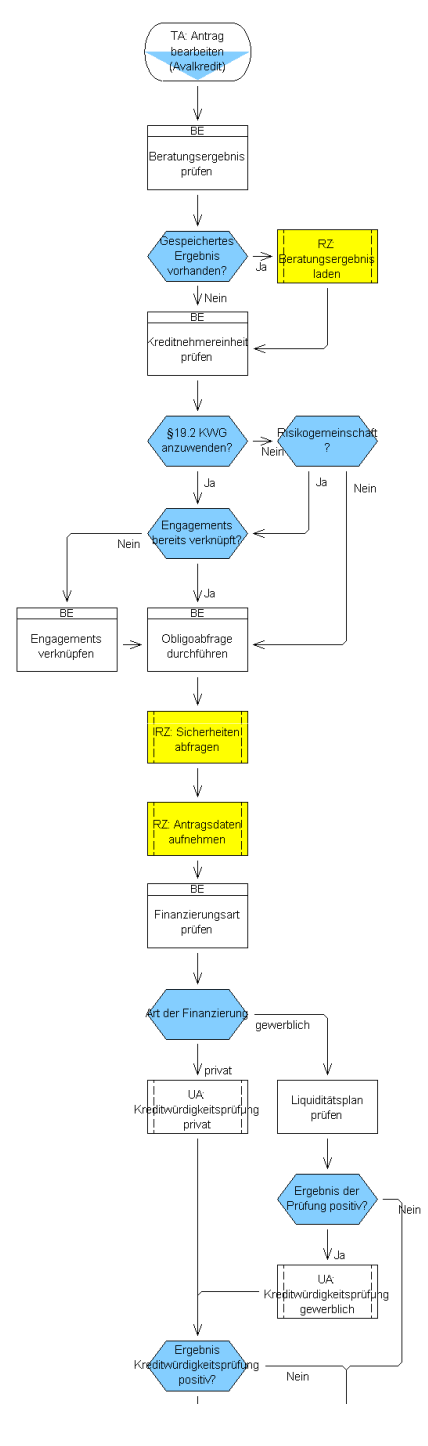
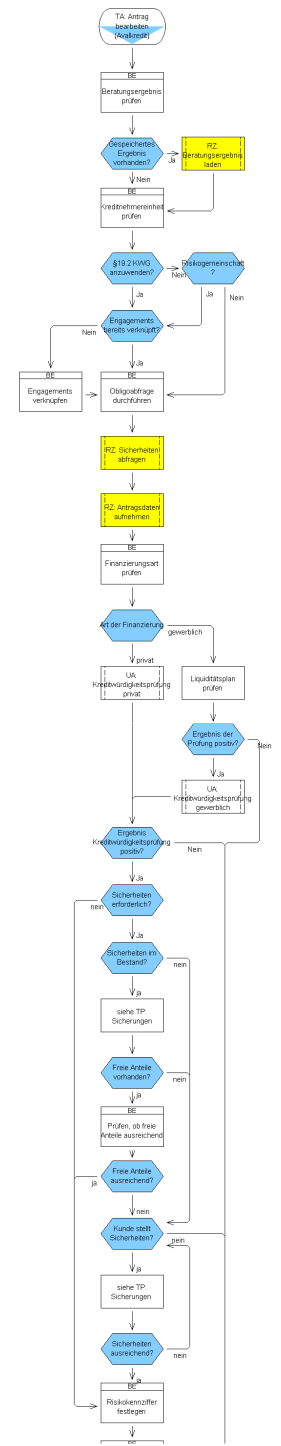
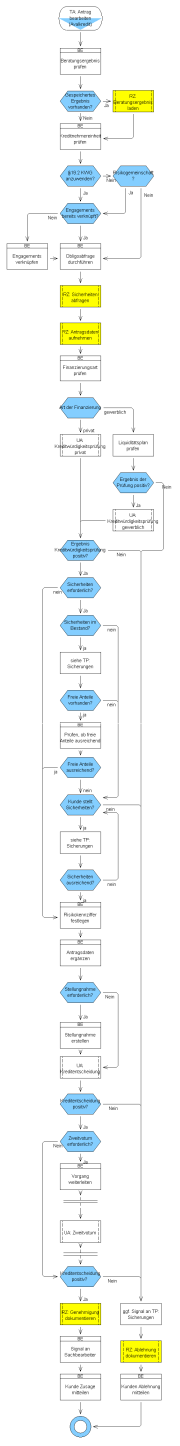


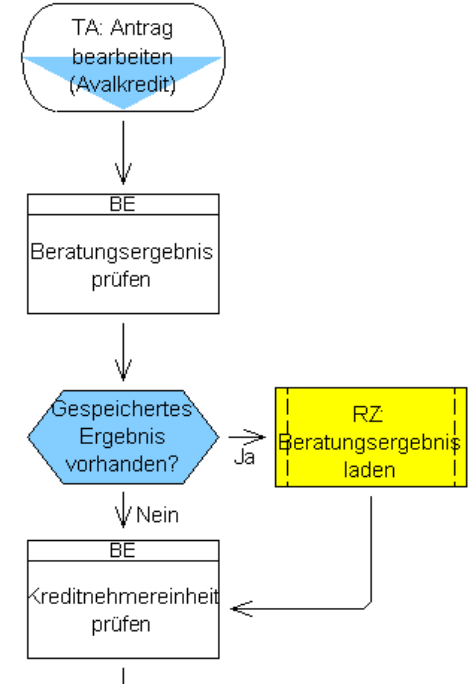
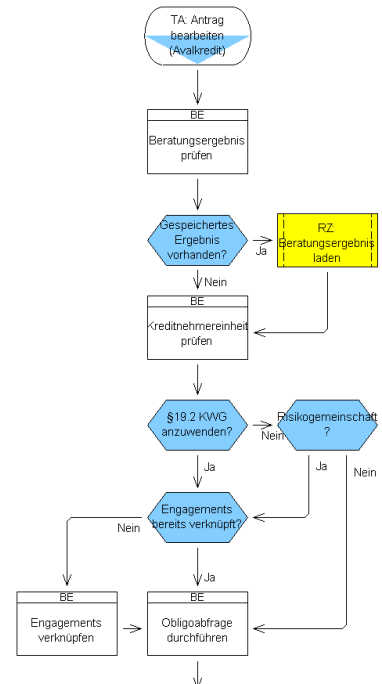
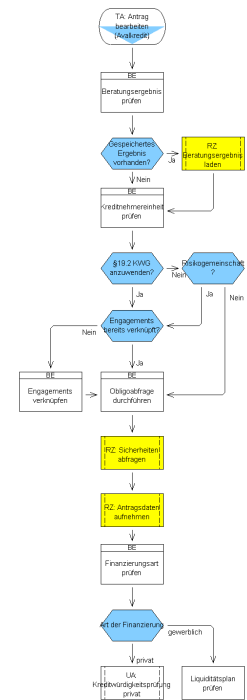
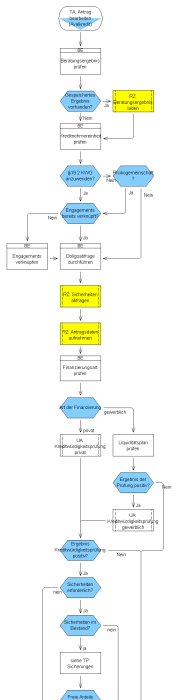


Irritationen

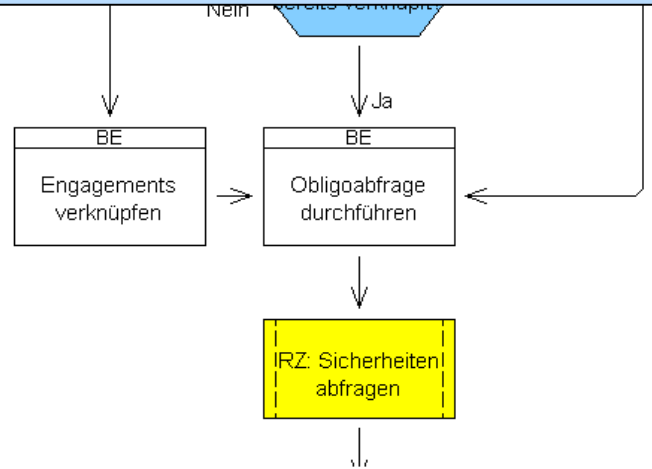
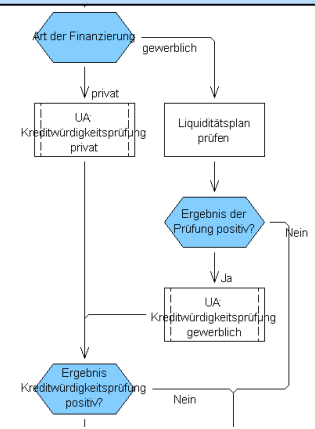
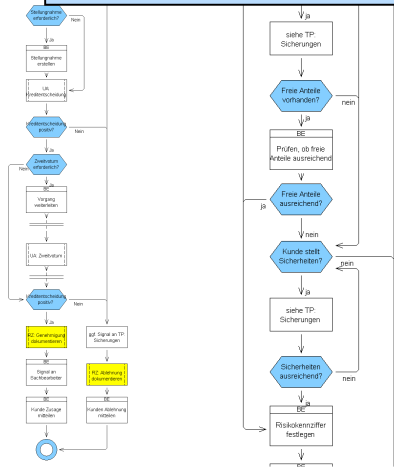
- Software-Landschaften werden größer und sind stärker integriert.
- Die Anforderungen an Software verändern sich ständig.
- Anforderungen an IT aus Prozesssicht steigen
- Die unterschiedlichsten Technologien ko-existieren







• Schon für einzelne Produkte (hier Kreditwesen in einer Retailbank) und Services ist die Menge der einzelnen Prozesse kaum überschaubar.
 • Die Analyse eines Prozesse zeigt, wie viele Varianten ein Prozess hat.





- Die „Heisenberg‘sche Unschärferelation“ der Informatik - Teil 1:
 - Die vorhandenen Softwaresysteme einer IT-Landschaft lassen sich nur vollständig erfassen, wenn das Unternehmen „stillsteht“.
 - Werden vorhandene Softwaresysteme im „laufenden Betrieb“ erfasst, verändert sich die IT-Landschaft während des Erfassungsprozesses substantiell.

- Die „Heisenberg‘sche Unschärferelation“ der Informatik - Teil 2:
 - Die vorhandenen Prozesse eines Unternehmens lassen sich nur vollständig erfassen, wenn das Unternehmen „stillsteht“.
 - Werden vorhandene Prozesse im „laufenden Betrieb“ erfasst, verändert sich die Abläufe und ihre Randbedingungen während des Erfassungsprozesses substantiell.



- Zur „Unschärferelation“ kommt erschwerend hinzu:
 - Es gibt kein zentrales, „objektives“ Wissen über Prozesse
 - Es gibt keine Rolle und keinen Akteur, die wissen, wie Prozesse „wirklich“ ablaufen
 - Es gibt eine substantielle Differenz zwischen beschriebenen Abläufen und tatsächlichen Abläufen
 - Prozesse müssen sich an bestehende Software anpassen und sind dadurch teilweise kompliziert und langwierig



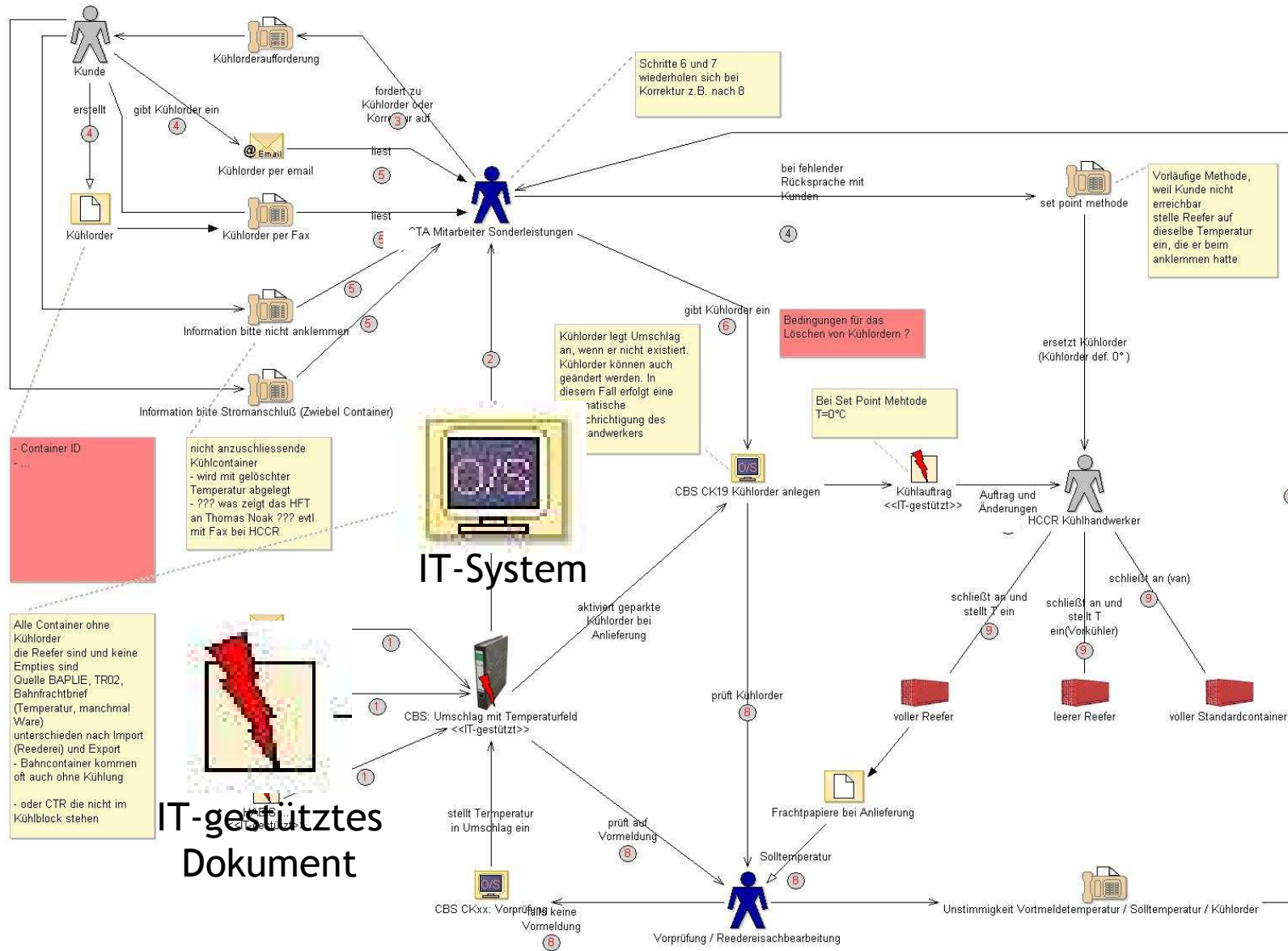
Was können wir tun?

→ **Landschaft und Geschäftsprozesse** iterativ und fachlich aufarbeiten

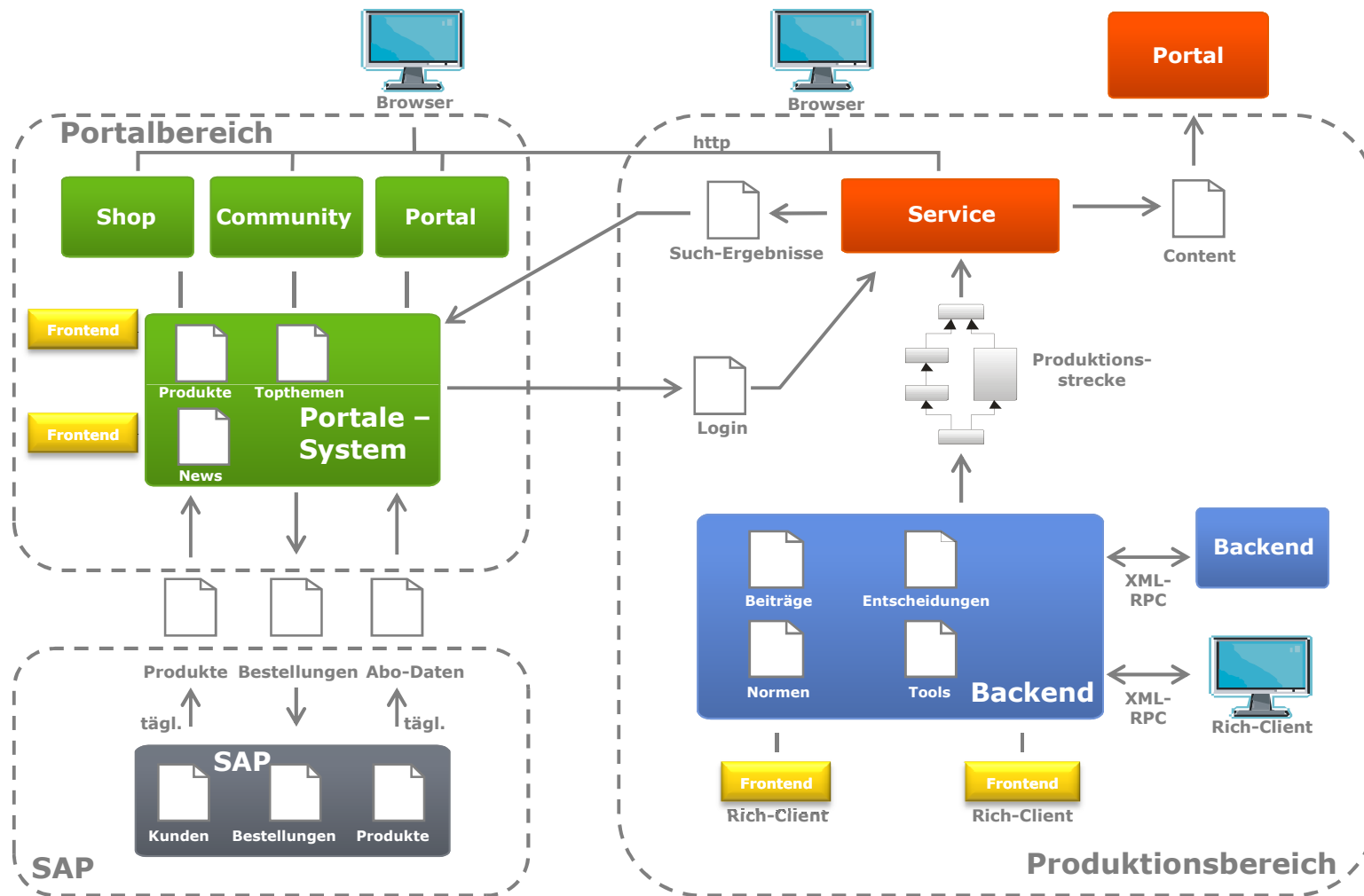
→ Pragmatische Ansätze zur Landschaftspflege:

- Das Prinzip der Vollständigkeit aufgeben und exemplarische Modellierung entlang priorisierter Fragen/Probleme vornehmen
- Den technik-zentrierten Ansatz aufgeben und fachliche Software-Landschaften aus den Geschäftsprozessen ableiten
- Das Interessen-/Sichtenproblem durch die Einführung einer IT-Bauaufsicht behandelbar machen

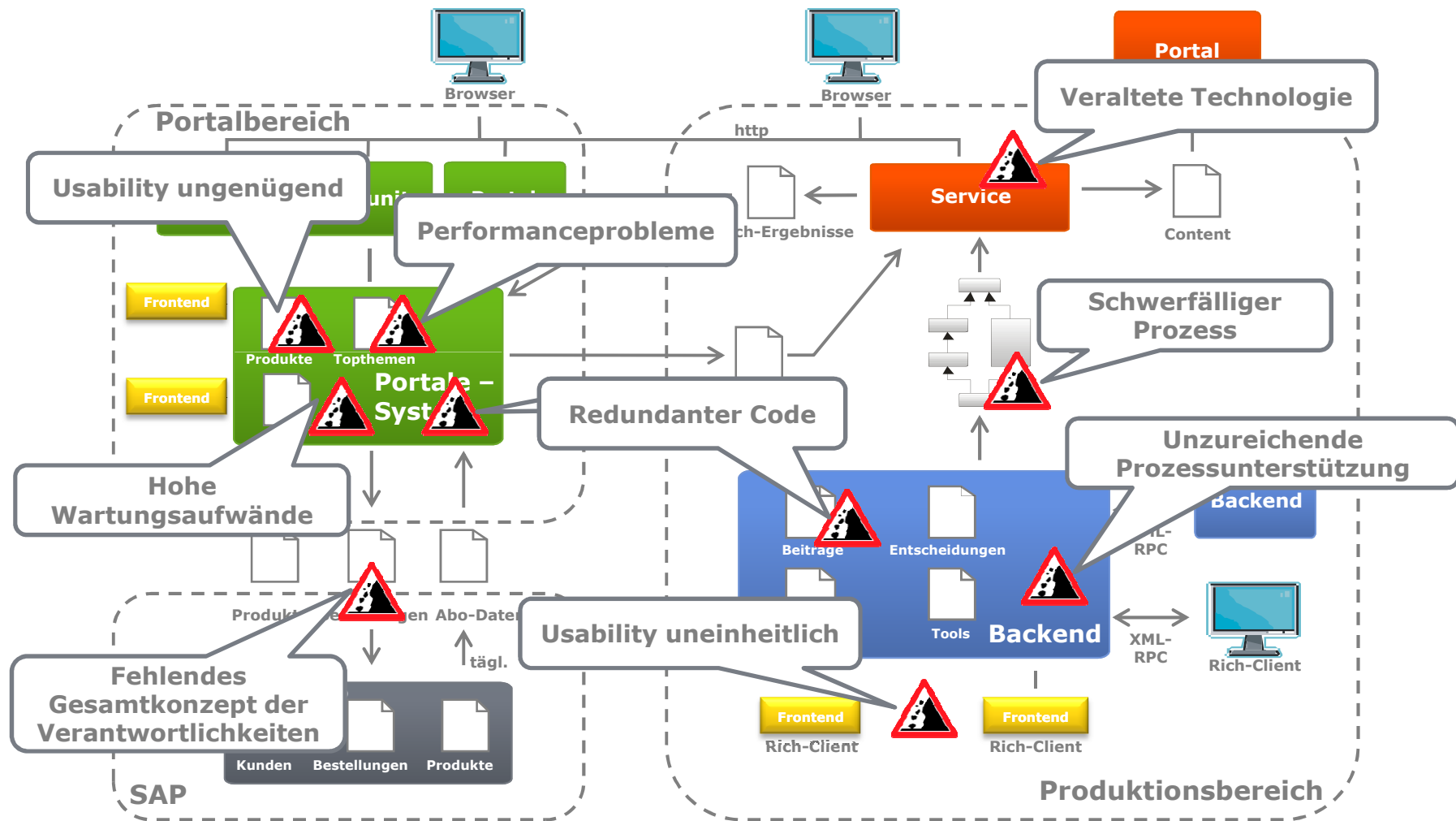
Fachliche Denkweise in den Darstellungsmitteln



Ein Blick aus 30.000 Fuß Höhe



Aktuelle Probleme und Risiken





Aufgaben der Bauaufsicht:

- Übereinstimmung geplanter und realisierter Architektur prüfen
- Einhaltung des Bebauungsplanes prüfen
- Einhaltung grundlegender softwaretechnischer Qualitätsrichtlinien prüfen
- Reporting an den Auftraggeber über den Projektfortschritt
- Abstimmung und Moderation zwischen den Beteiligten bei Planabweichung



Methoden:

- Code und Projekt-Reviews
- Werkzeuggestützte Analyse (Metriken, Architekturprüfung)
- Interviews, Workshops, Briefings



Was können wir tun?

- **Qualitätssicherung** in den Fokus von IT-Projekten und Architektur stellen
- **Fachliche Darstellungsmittel** mit Anwendern iterativ ausarbeiten
- **Landschaft und Geschäftsprozesse** iterativ und fachlich aufarbeiten
- **Neutrale Bauaufsicht** für IT-Landschaften und Migrationen einführen

