Sicherheit und Zuverlässigkeit in der Software-Entwicklung

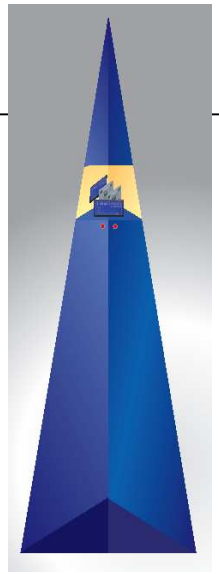
Sergio Montenegro

sergio@first.fhg.de

Holger Schlingloff

Holger.Schlingloff@first.fhg.de

Konzeption & Festlegung der Requirements



FIRST

Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

Requirements Festlegung

Nach der Konzeption...

Wie entwickele ich was der Kunde Braucht?

Vorsicht...

Es könnte schokierend sein!

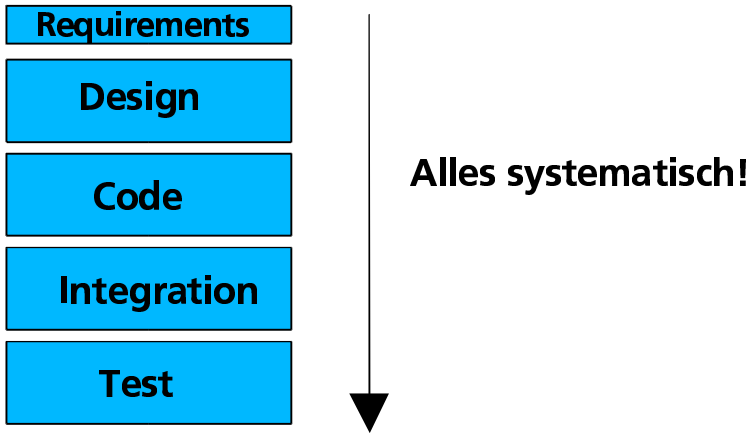
Requirements Festlegung?

Nach der Konzeption...

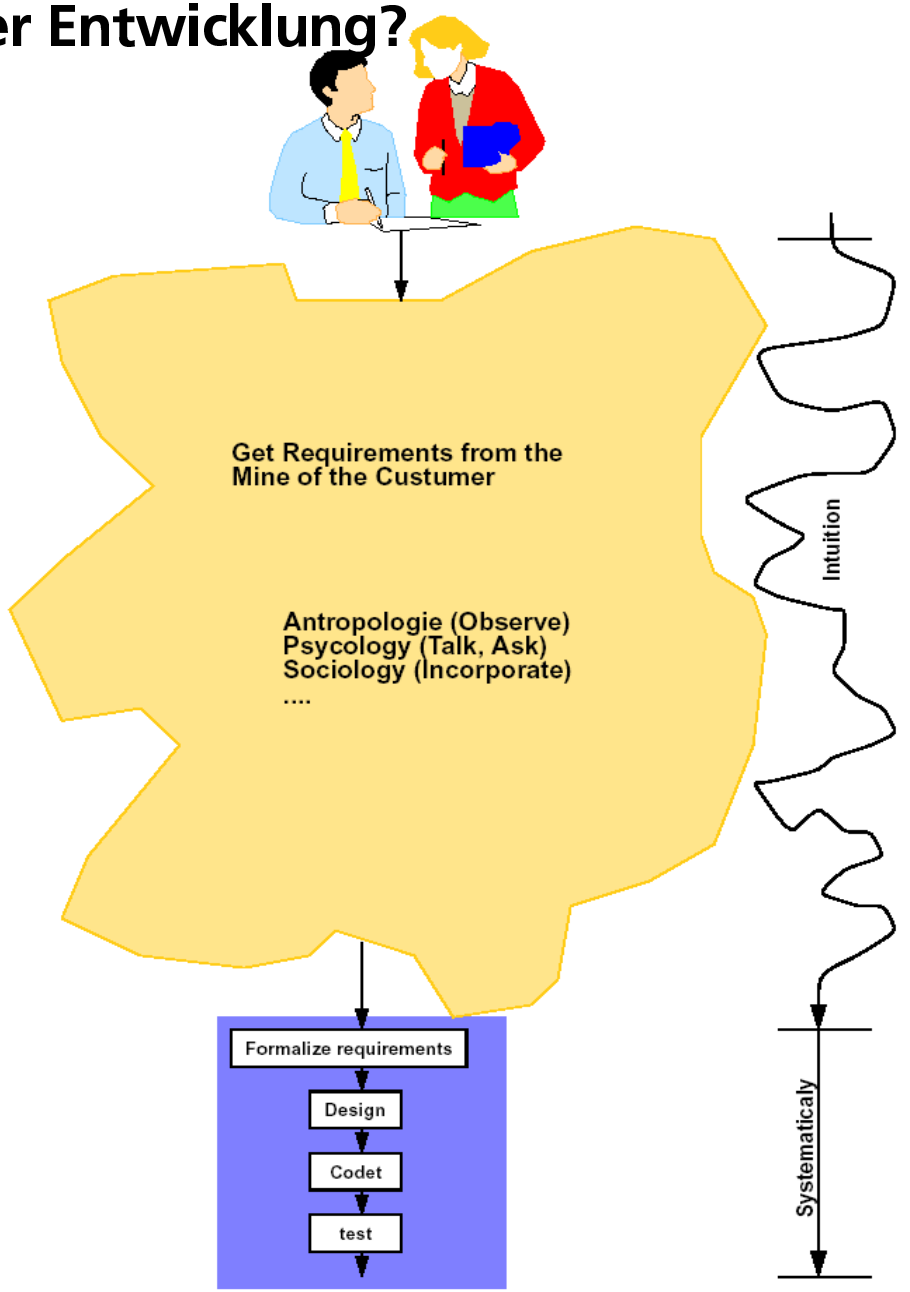
Wie entwickele ich was der Kunde Braucht?

.... Es geht nicht!

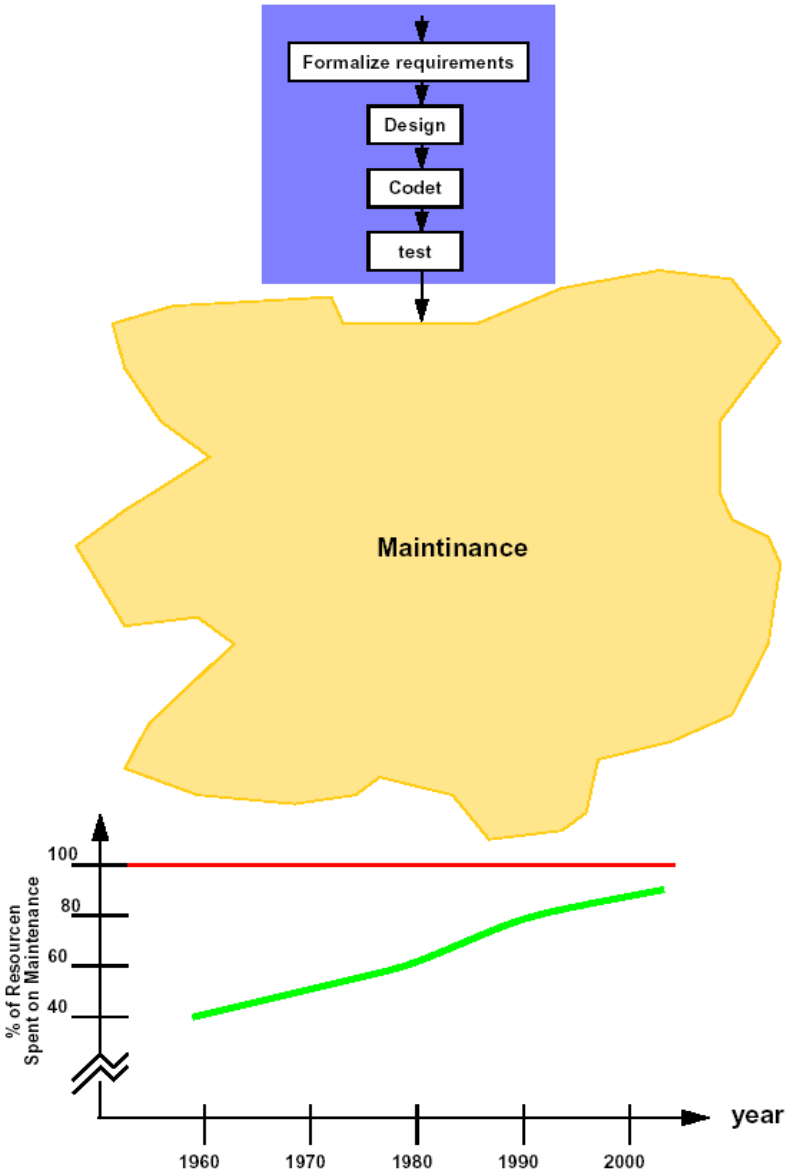
Entwicklung in 5 Lektionen... (oder 4 ½)

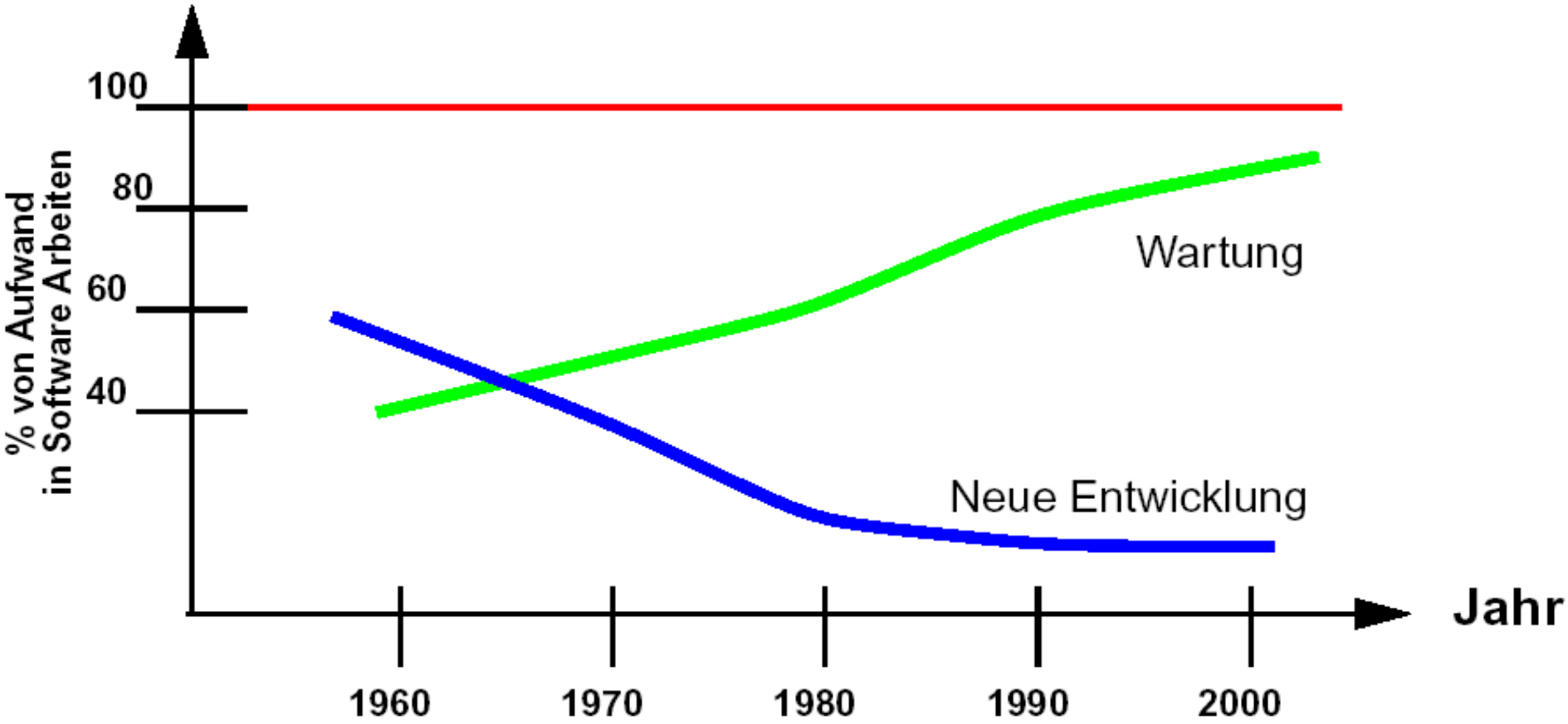


Requirements: Ein kleiner Teil in der Entwicklung?

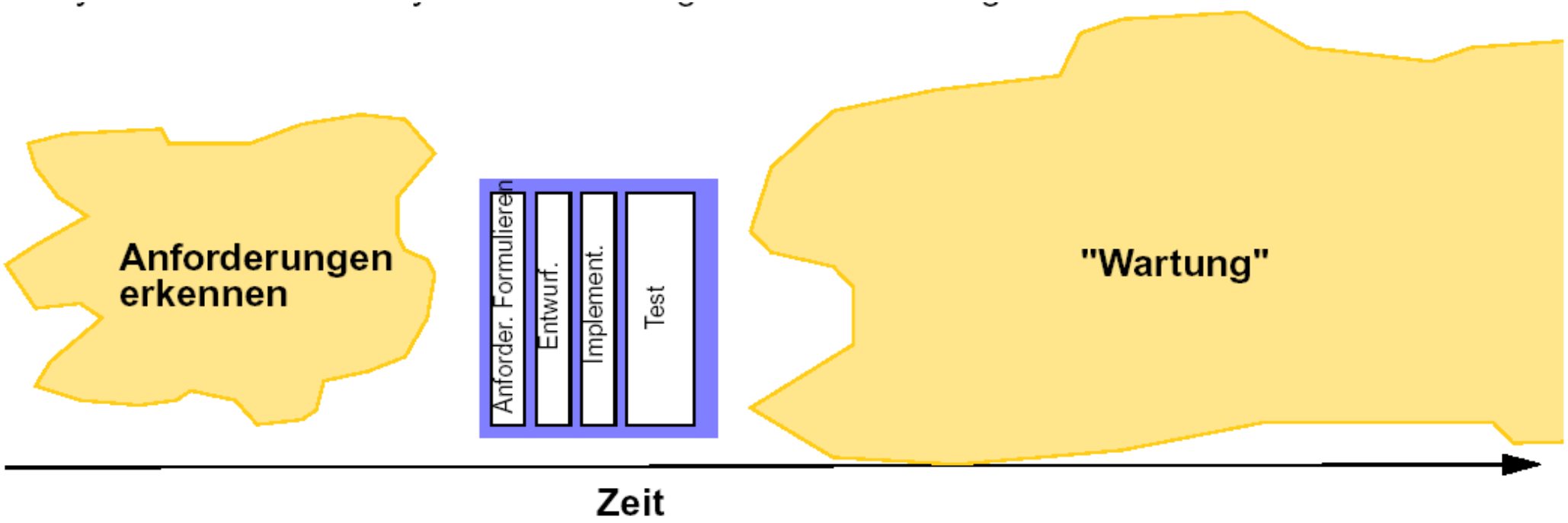


Test: Ende der Entwicklung?




100
80
60
40
% von Aufwand
in Software Arbeiten
Wartung
Neue Entwicklung
Jahr
1960
1970
1980
1990
2000

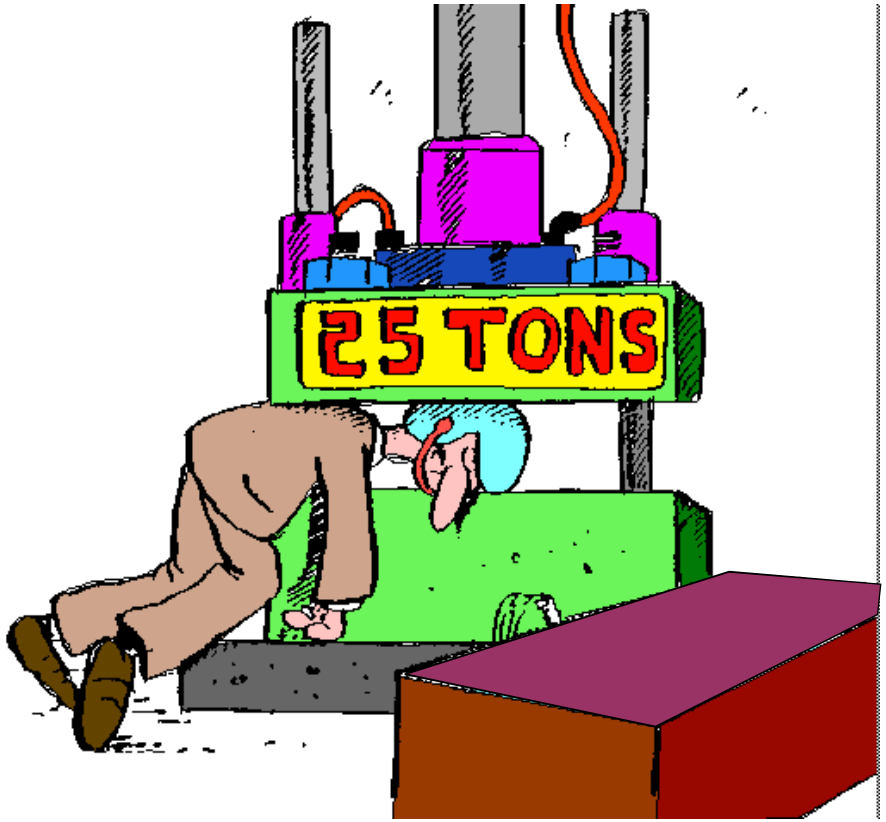
Entwicklung ist das kleinste!



Requirements: Wie Nehme ich die Requirements aus dem Kundenkopf..

Requirements: Wie Nehme ich die Requirements aus dem Kundenkopf..

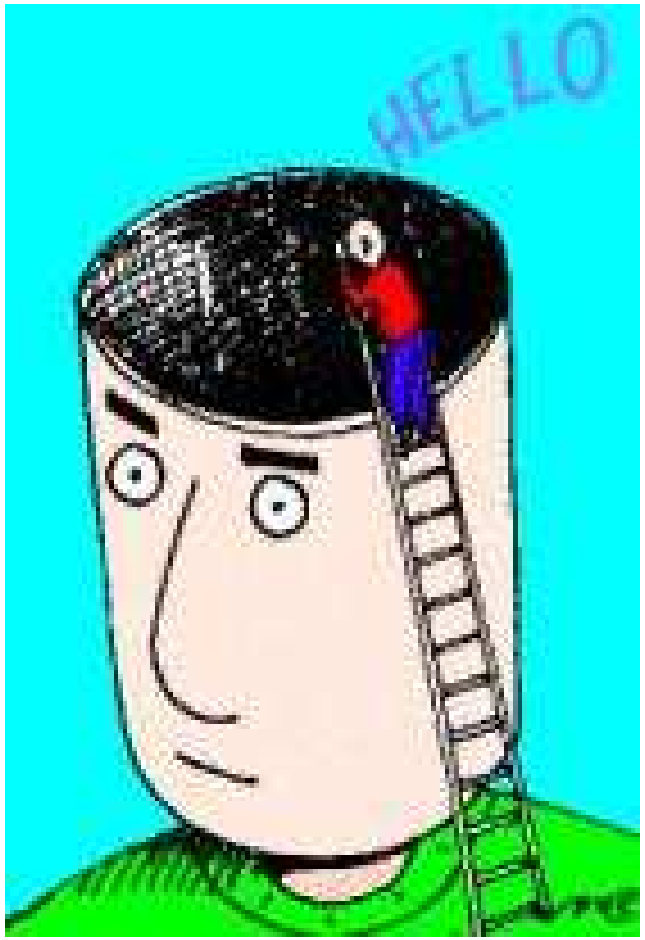
Ohne Ihm zu beschädigen...



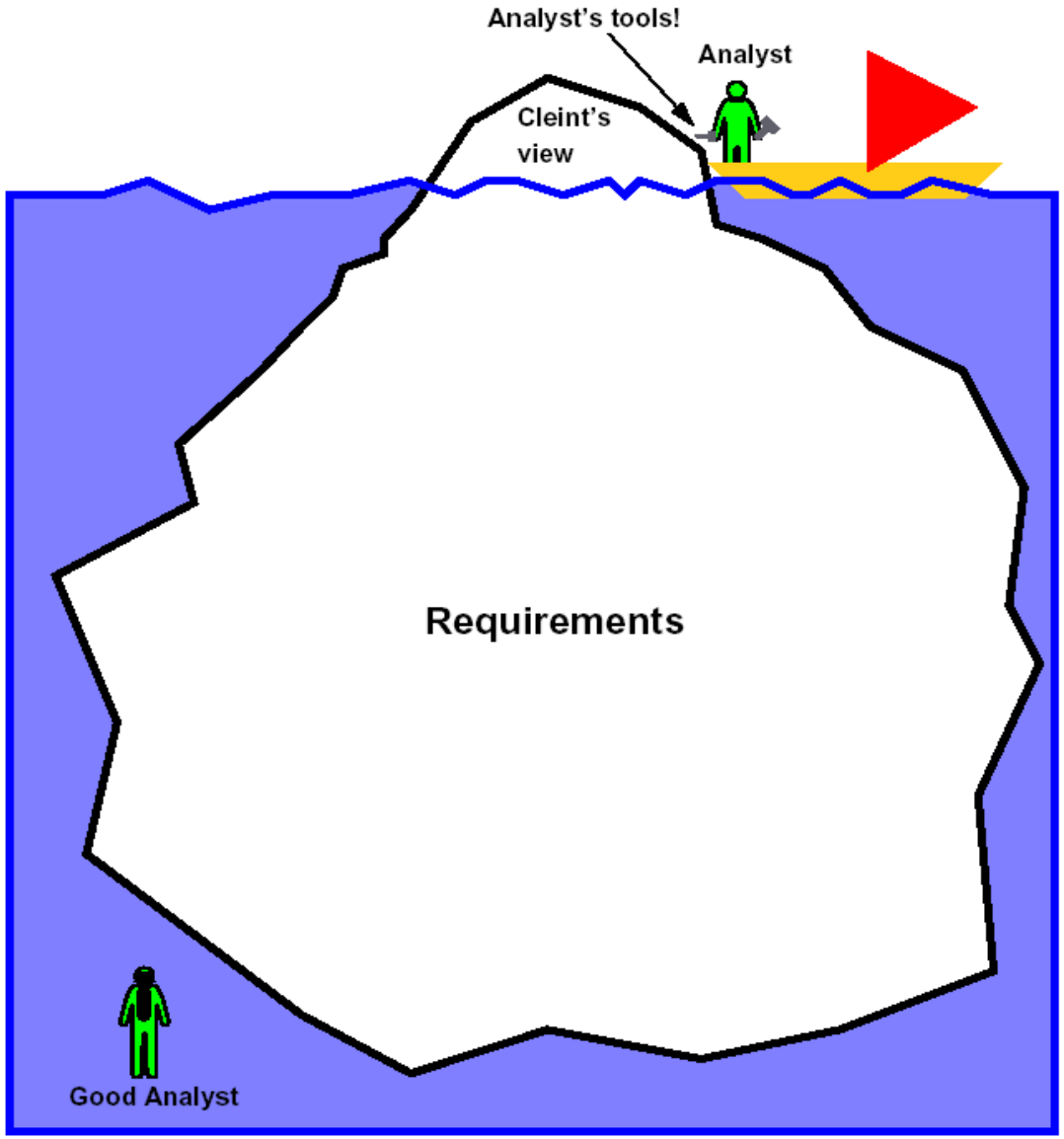
Requirements: Wie Nehme ich die Requirements aus dem Kundenkopf..

Ohne Ihm zu beschädigen...

Überhaupt nicht...
Die Requirements sind nicht da!



Requirements Analyse



Ein Experte für die Requirementsanalyse?

Wissender (Oberschlauer, Experte im Fachgebiet):

Er weiß bereits, er braucht nicht von Kunde zu hören

Er belehrt den Kunde darüber, was dieser braucht

Dieselben Worte mit verschiedener Bedeutung:

- **Keiner merkt es**
- **Merkt nicht die Annahmen (auch nicht die eigenen)**
- **Er kann nicht zeigen, dass er es nicht verstanden hat.**

Kein Experte? Dann ein Ignorant?

Ignorant (Unwissender, Inexperte im Fachgebit... aber Schlau):

Hat keine Annahmen -> muss sich informieren

Sogar minimal verschiedenen Bedeutungen eines Wortes interpretiert er als Inkonsistenz

Braucht nicht zu verheimlichen, dass er etwas nicht verstanden hat

Fragt alles, was nicht ganz klar ist

Dann ein Team Experte + Inexperte

Team:

Mehrere Fachgebiets-Experten und EIN Ignorant!

Experte: Haben Wissen

Ignorant soll sein:
Mutig, clever, Experte in SW und Logik,
ausgezeichnetes Gedächtnis, fähig Folgen zu durchblicken

Dann ein Team Experte + Inexperte **+ Kunde!**

Sprache des Kundes übernehmen (Kein Informatiker-Slang)
Verständlichkeit -> << Übersetzungsfehler

z.B.
Der HK der SBC erhält Entries aus der PDH und OC.
Die Items aus der UL werden in jedem Fall zu HK weitergeleitet.
Für dieses System haber wir das Paradigma einer DB angewandt.
Die Metapher mit der DB besteht im Heuristics beim Daten Suchen.

Metapher: (Duden):
Eine Metapher ist ein sprachlicher Ausdruck, bei dem ein Wort oder eine Wortgruppe aus seinem eigentümlichen Bedeutungszusammenhang in einen anderen übertragen wird, ohne dass ein direkter Vergleich die Beziehung zwischen Beichnendem und Bezeichnetem verdeutlicht .

Hilfe für den Kunde: Informatiker-Slang Worterbuch (patent pending)

Informatiker	Normaler Mensch
System	Dingsda, Dingsbums (Sie wissen kein Name dafür)
Item	Ding
Paradigma	Beispiel
Metapher	Vergleich
Daten	Irgendwas in einem Rechner (Sie wissen es nicht genau)
90% Fertig	Es fehlen nur die anderen 90%
Es hängt davon ab...	Ja! / Nein!

FIRST
Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

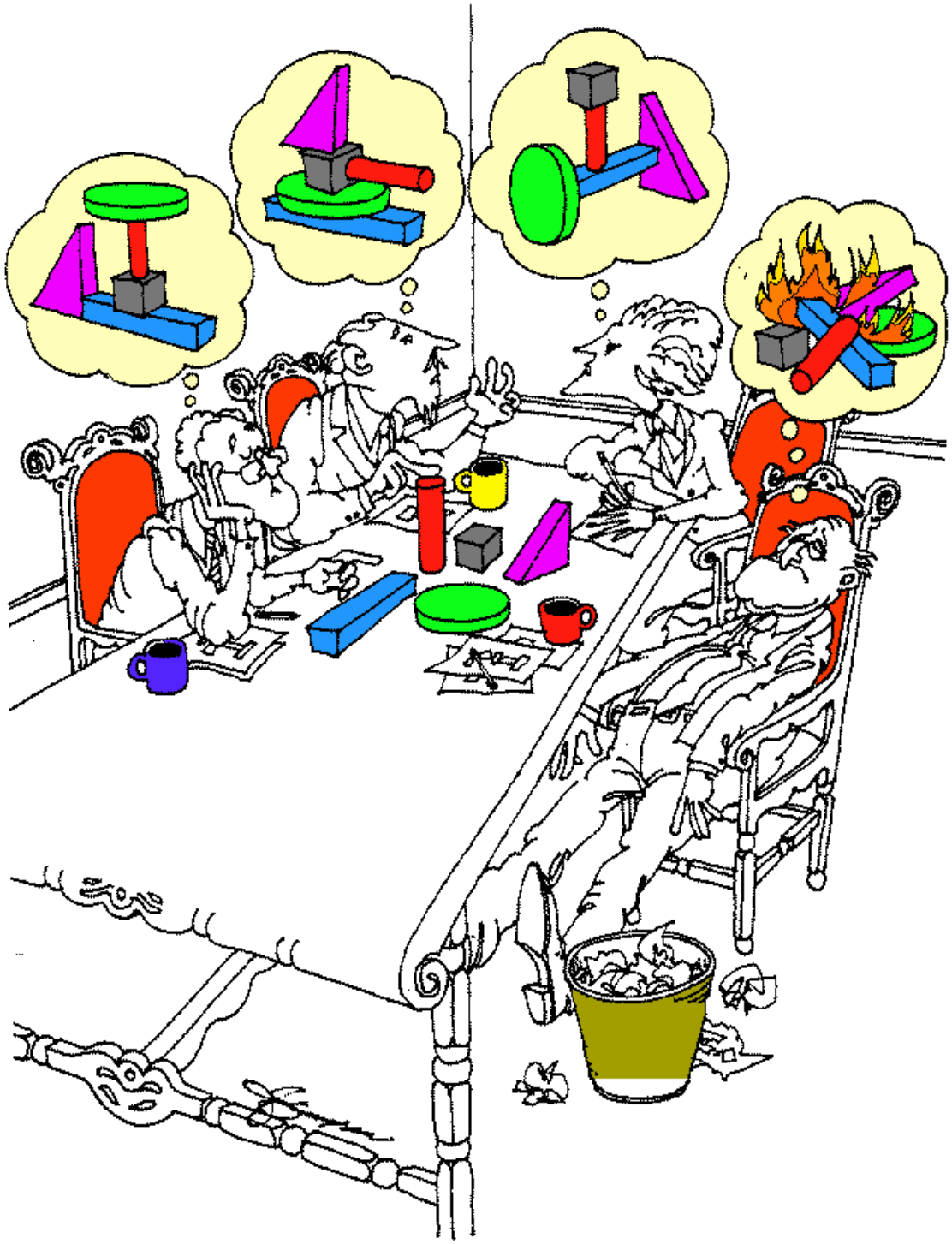
Dann ein Team Experte + Inexperte + Kunde!

Sprache des Kundes übernehmen (Kein Informatiker-Slang)
Verständlichkeit -> << Übersetzungsfehler

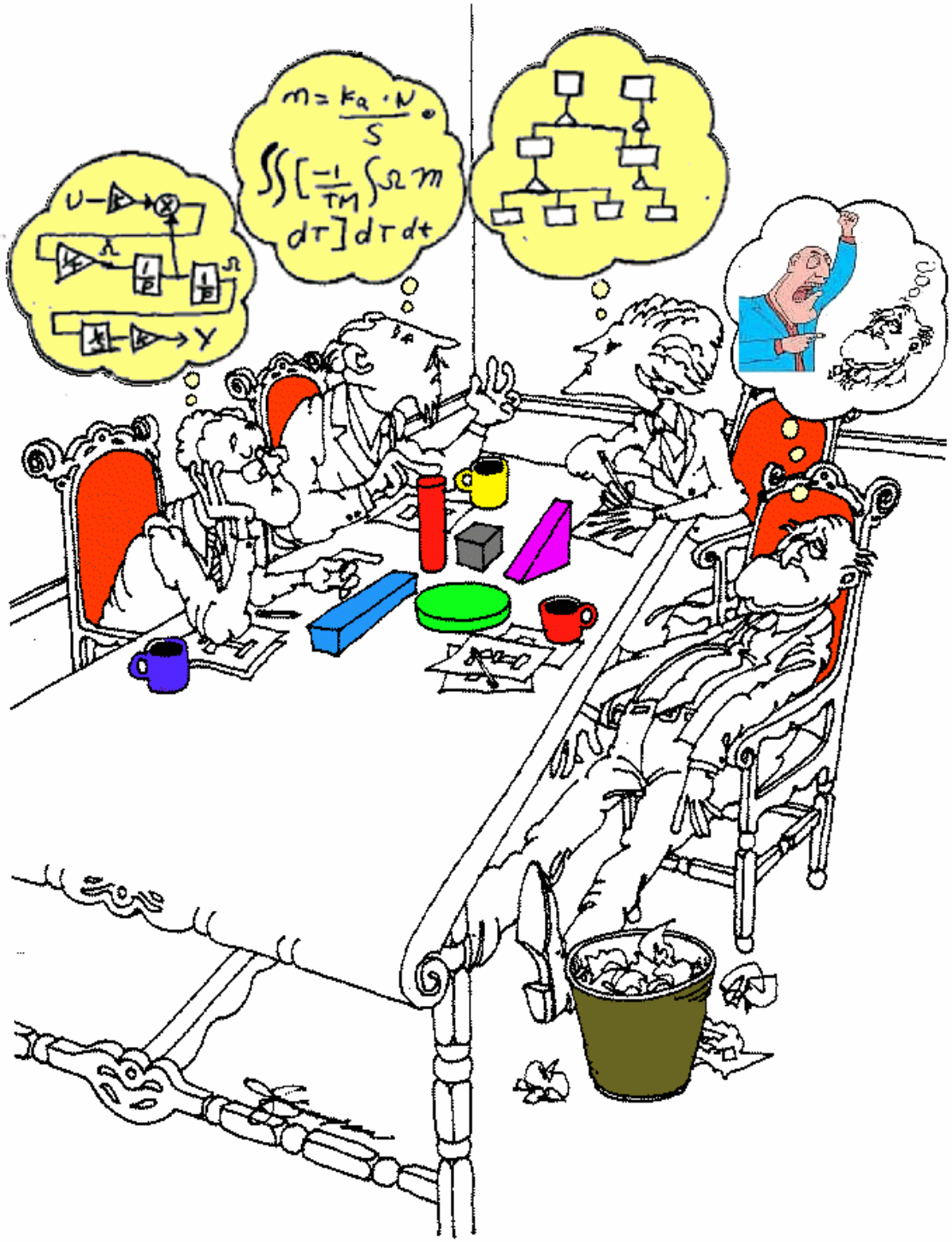
Gedanken des Kundes Übernehmen

Informatiker	Sequentiell,	Diskret
Elektrotechniker	Parallel,	Diskret
Analog Techniker	Parallel,	Kontinuierlich
Mathematiker	Sequentiell,	Diskret
Physiker	Parallel,	Kontinuierlich
Luft und Raumfahrt		
Mediziner		
Pyichologe		

Ein Team?



Ein Team?



Fra

Dann ein Team Experte + Inexperte + Kunde!

Sprache des Kundes übernehmen (Kein Informatiker-Slang)
Verständlichkeit -> << Übersetzungsfehler

Gedanken des Kundes Übernehmen

Informatiker	Sequentiell,	Diskret
Elektrotechniker	Parallel,	Diskret
Analog Techniker	Parallel,	Kontinuierlich
Mathematiker	Sequentiell,	Diskret
Physiker	Parallel,	Kontinuierlich
Luft und Raumfahrt		
Mediziner		
Pyichologe		

Dann ein Team Experte + Inexperte + Kunde!

Sprache des Kundes übernehmen (Kein Informatiker-Slang)
Verständlichkeit -> << Übersetzungsfehler

Einfach und Kurz (Gegenbeispiel ISO)

Bestimmung und Zweck des Systems (Was + Wofür)

System benennen (Name == System -> ich weiß nicht, was es ist)

Beobachten von innen

Nicht sagen was sei meinen: ZEIGEN!

Verschiedene Sichten suchen (Verschiedene Leute) (wenn alles Ihnen entgegen kommt, dann sind Sie auf der Falsche Spur)

Dann ein Team Experte + Inexperte + Kunde!

Die eigenen Verständnis überprüfen

Nicht vergessen: Sie bekommen nicht alle Informationen
-> Finden, was nicht da ist
-> Konsequenzen finden
-> Inkonsistenzen finden

System abgrenzen

Annahmen protokollieren
Wenn alles auf Annahmen basiert und sie falsch sind...
Solche Fehler werden nicht einmal beim Testen sichtbar!
(Tiefflliegern übers Totenmeer)

Requirements Analyse...

Noch mehr Information Herausholen ...
Informationen Filtern & Reduzieren
Nutzbare Information extrahieren
- **1000 seiten -> 10 Seiten**
- **mit allen nutzbaren Informationen!**
- **ohne unnütze Informationen und Redundanz**
- **Ohne Widersprüche**

Marin & Tsai, 1988 Train Control

10 Seiten Anforderungen

10x 4-Personen-Teams als Reviewer

Erwartung: 1 oder 2 Fehler

Gefunden: 92 Fehler (Inkonsistenzen, Lücken, etc.)

Jedes Team fand im Durchschnitt 35.5 Fehler (92 – 32 = 57!)

Viele Fehler wurden nur von einem Team gefunden
(Andere Fehler wurden nicht gefunden und sind noch da!)

Die schlimmsten Fehler wurden am seltensten gefunden

Analyst's Ilution (Dream):

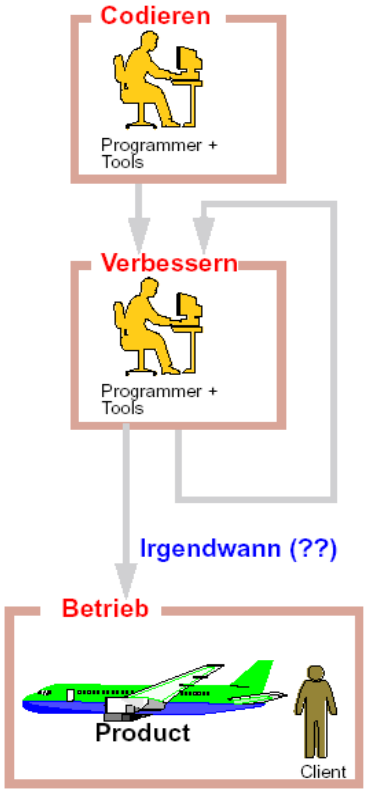
"You start coding. I'll go find out what they want"

"The Client knows what he wants"

"Requirements are sasy to obtain"

Where do requirements come from?

Not from the client's mind, They have to be invented!



Naiv!

Computer analyst to programmer:
"You start coding. I'll go find out what they want"

Codiere-und-Verbessere

Analyst's Ilution (Dream):

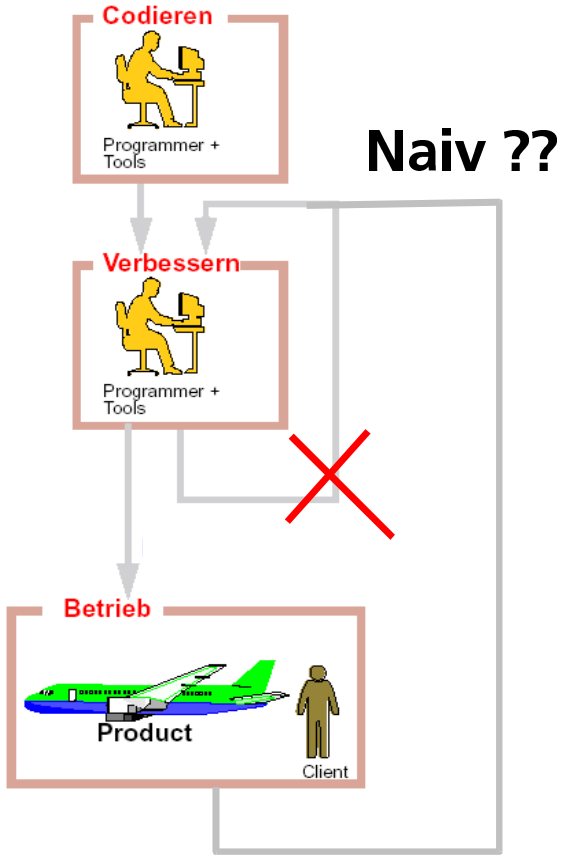
"You start coding. I'll go find out what they want"

"The Client knows what he wants"

"Requirements are sasy to obtain"

Where do requirements come from?

Not from the client's mind, They have to be invented!



Naiv ??

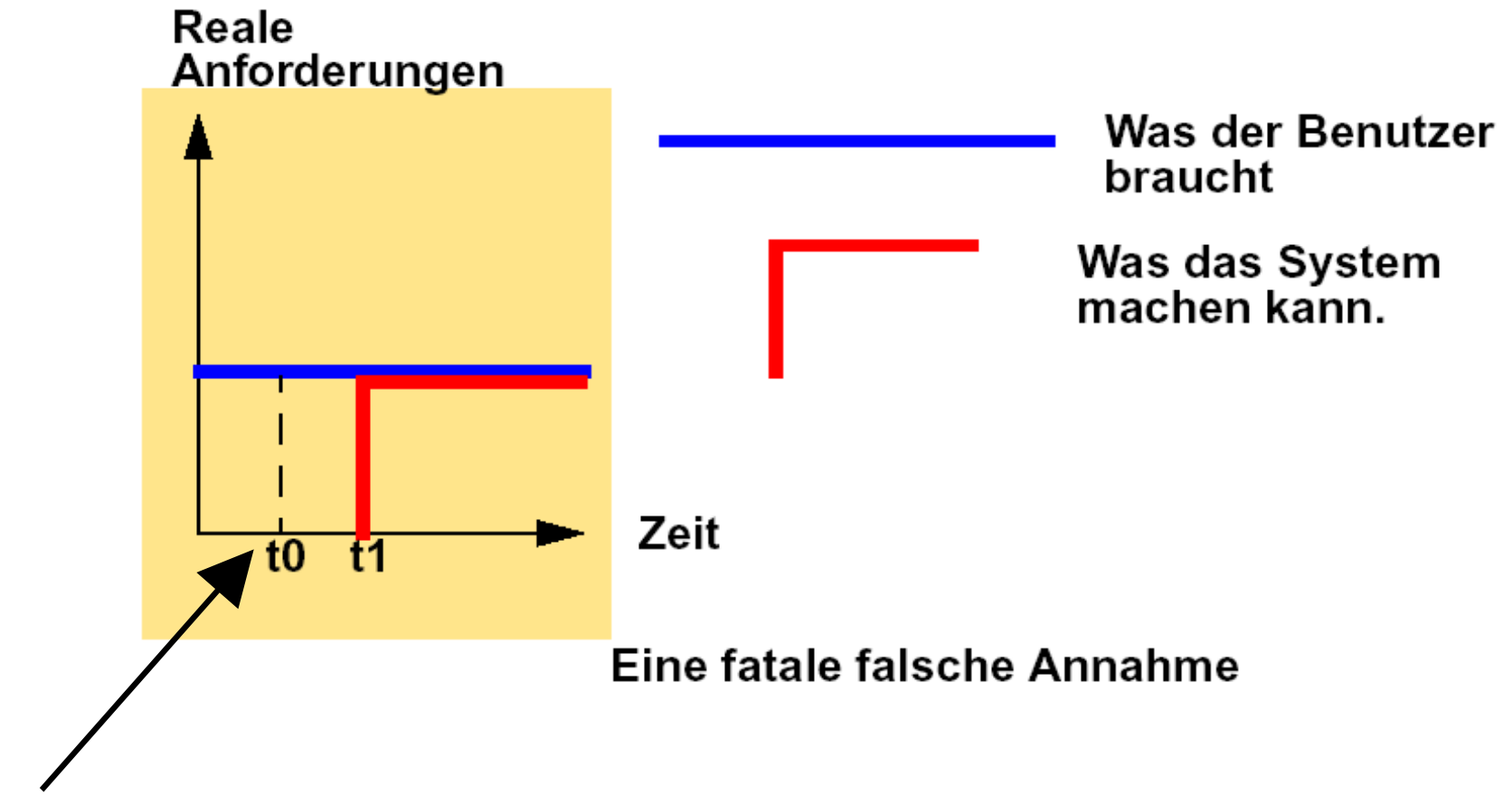
Codiere-und-Verbessere

1. Schritt: Requirements festlegen und einfrieren

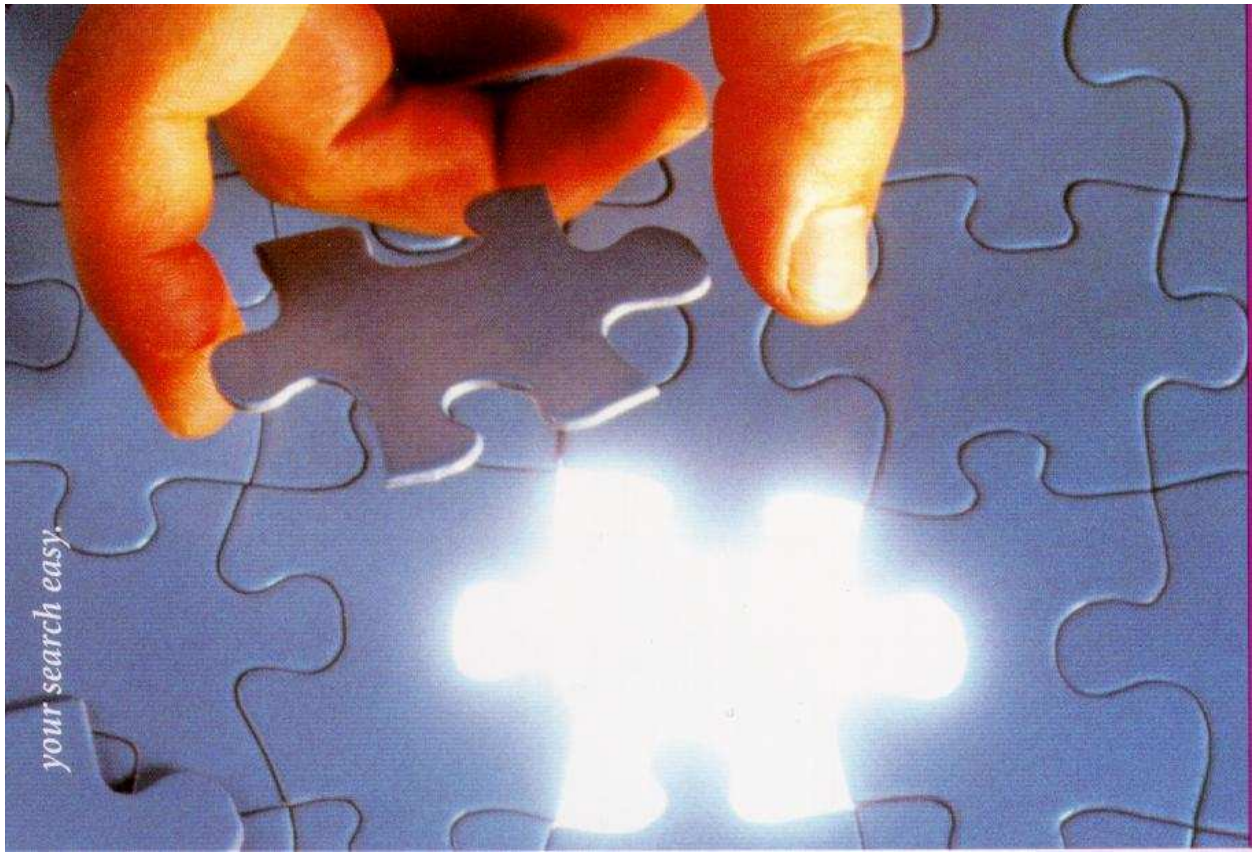
1. Schritt: Requirements festlegen und einfrieren

1. Schritt in der falsche Richtung
1. Fehler!
Sie Können den Kunde nicht einfrieren

1. Schritt: Requirements festlegen und einfrieren



Das Computer System als letze Puzzleteil des Gesamtssystems



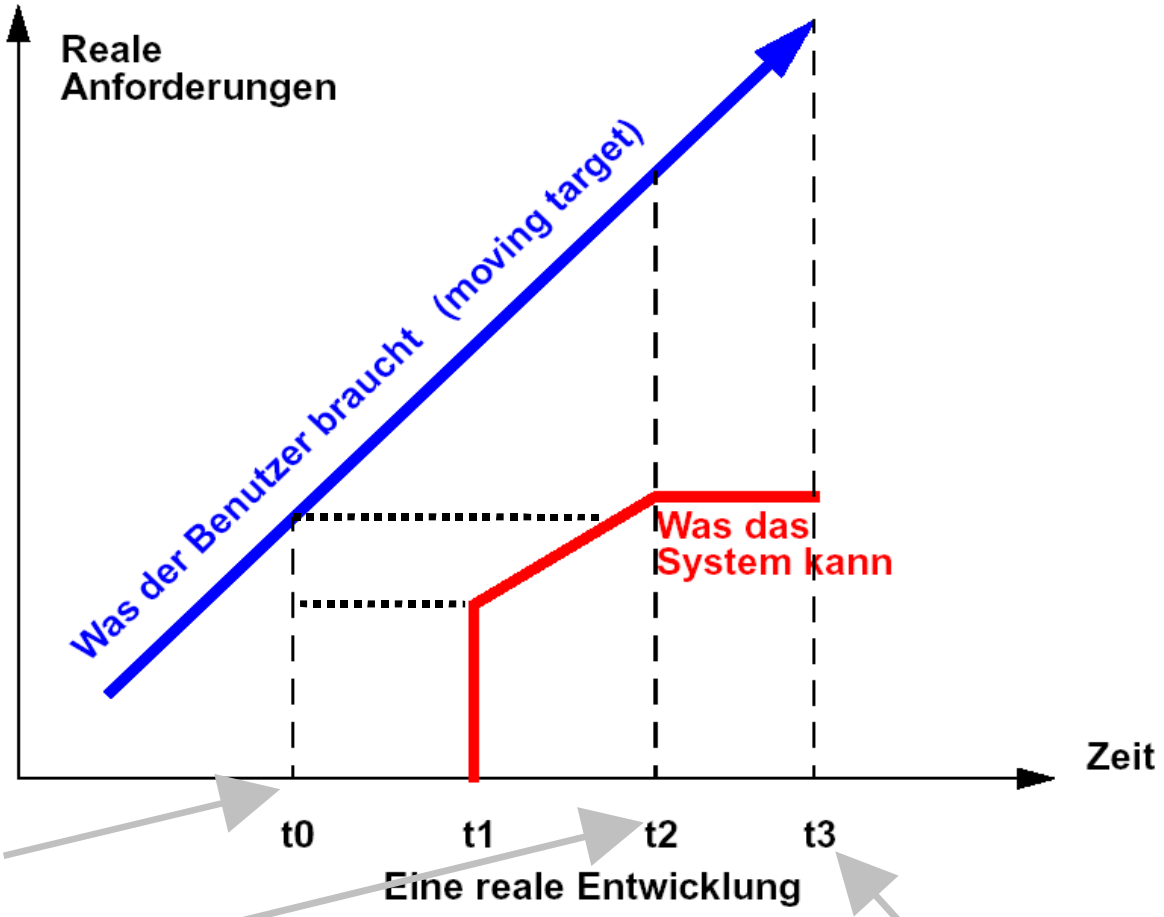
Aber....

**Wenn die Umgebung des Systems sich ändert,
dann muss sich das System auch ändern
(sonst pass es nicht mehr)**

**Sobald das System eingebaut wird, ändert sich
die Umgebung.... !**

**Die richtige Anforderungen kennt man erst wenn es richtig läuft
davor sind alles nur Vermutungen (z.B. ein Neues Haus)**

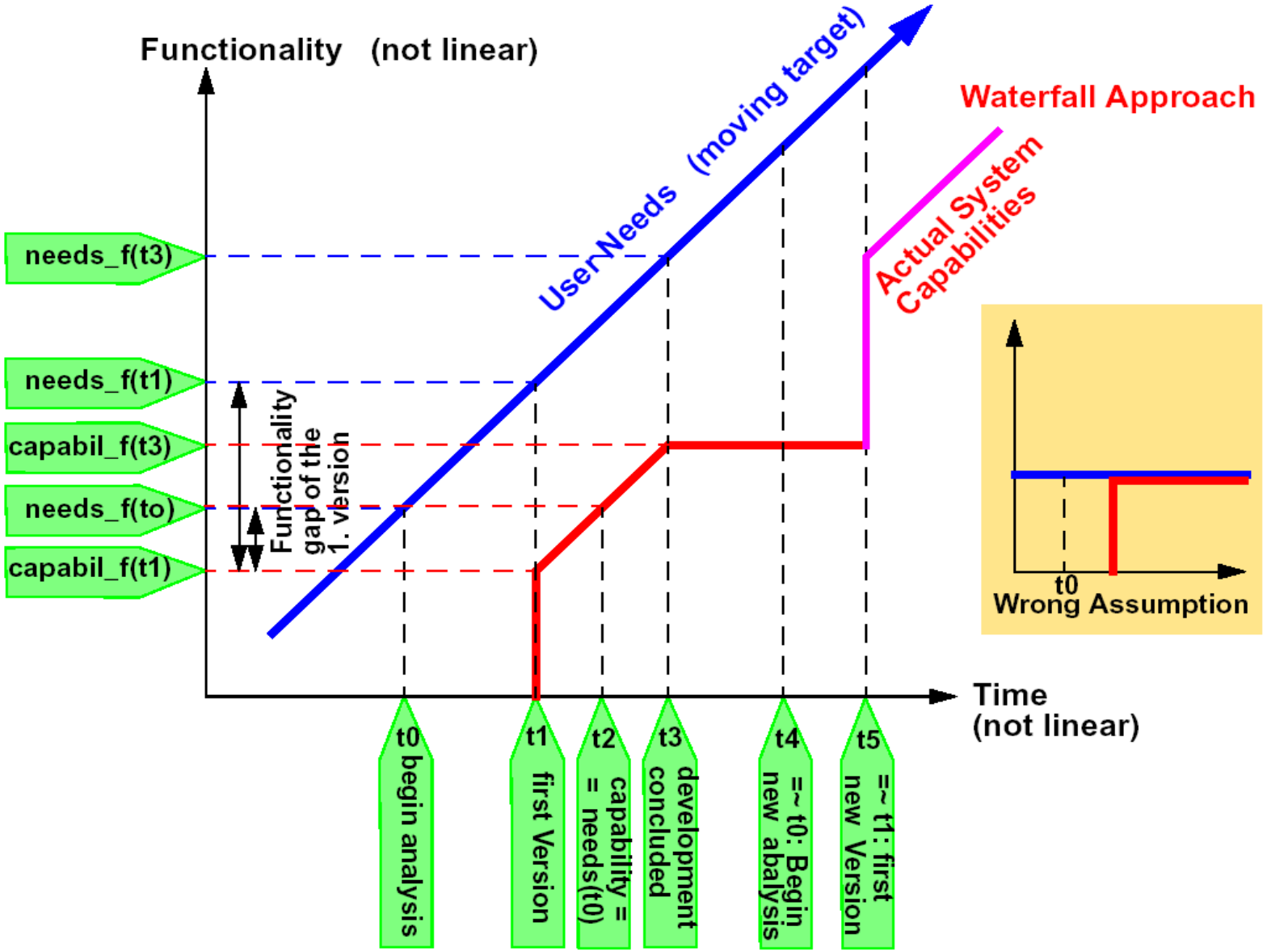
Lebendige Systemen ändern sich

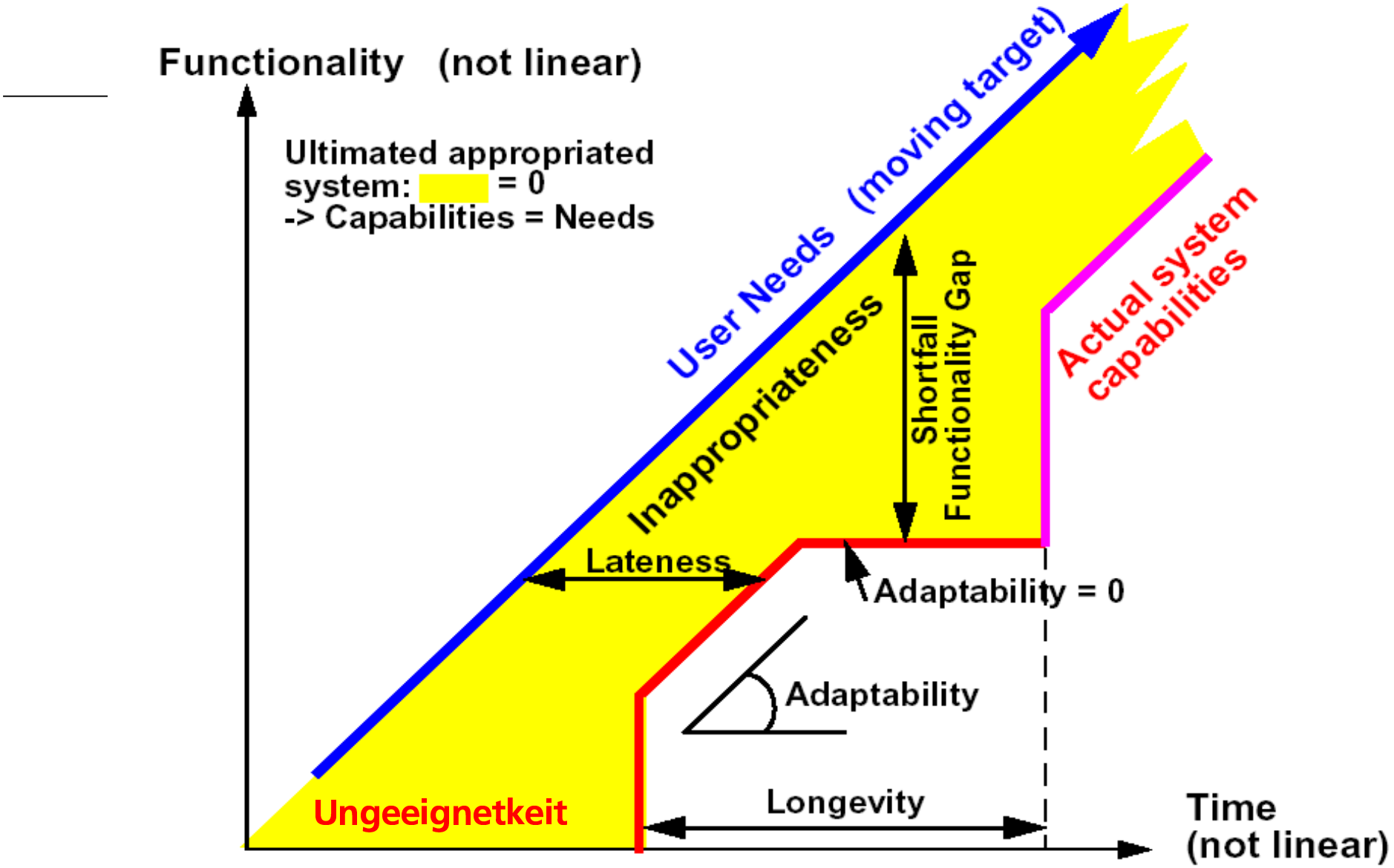


Requirements werden festgelegt

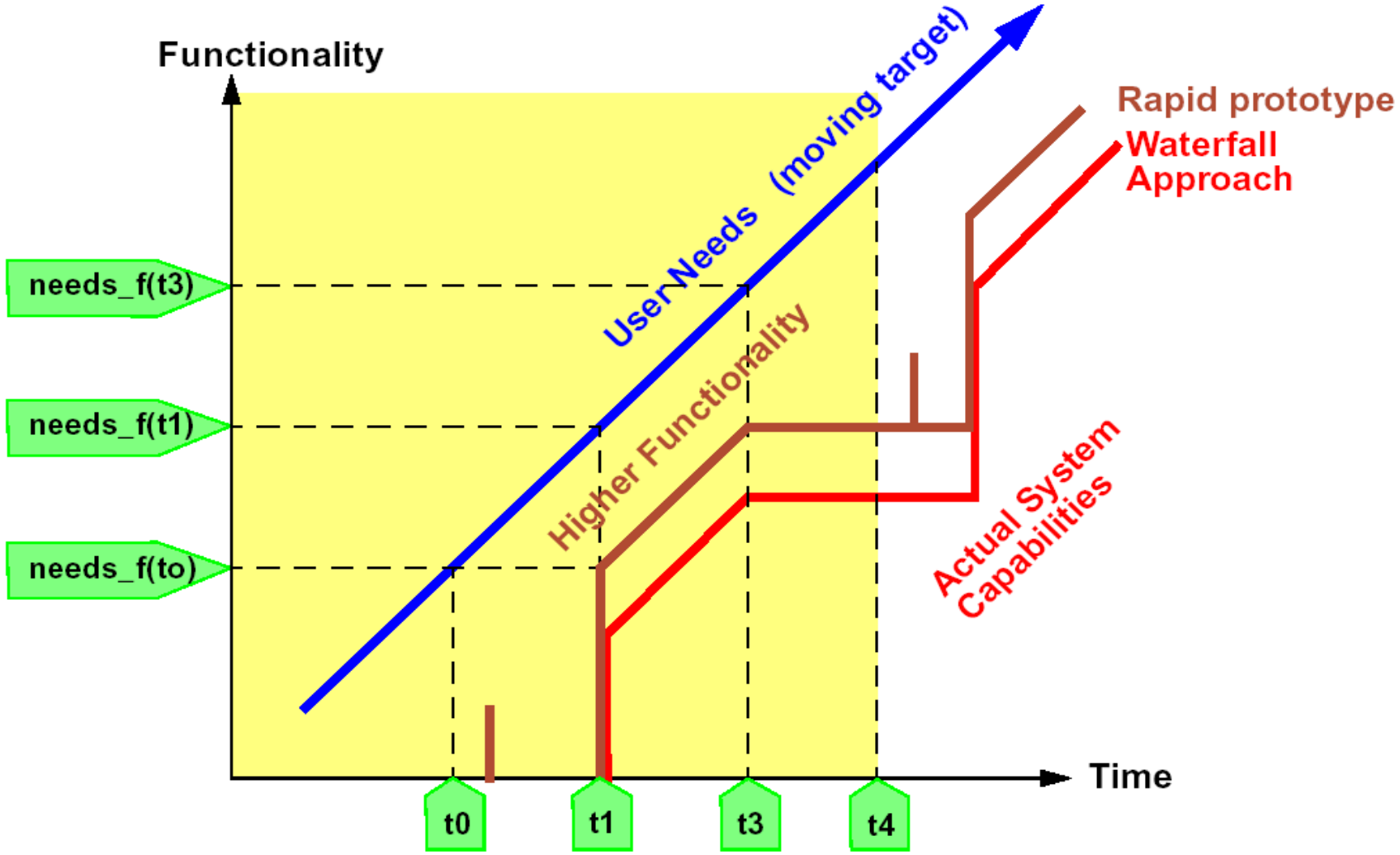
Ende der Entwicklung

Der Kunde kann mit dem System nicht weiter arbeiten

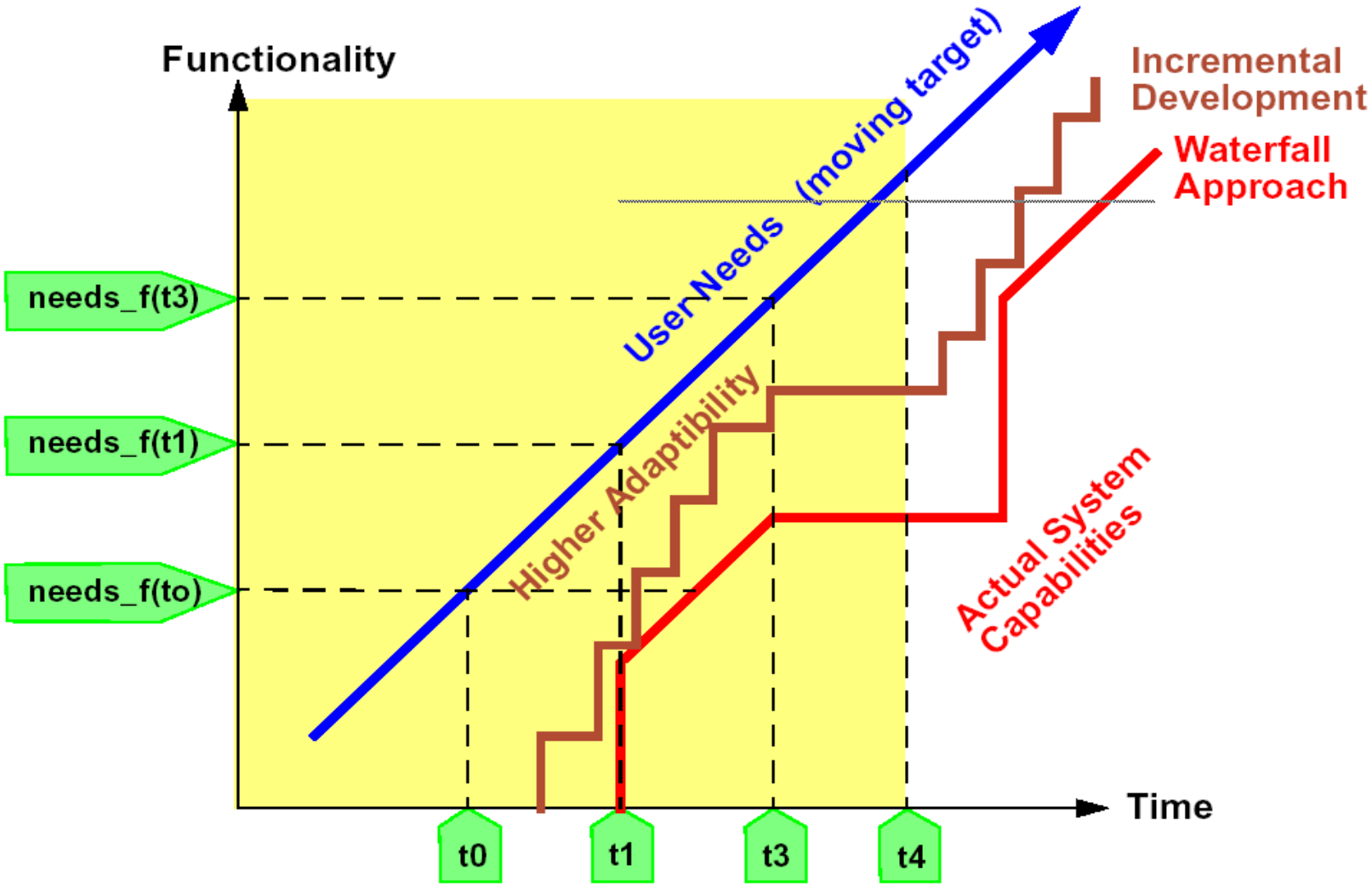

Functionality (not linear)
User Needs (moving target)
Waterfall Approach
Actual System Capabilities
needs_f(t3)
needs_f(t1)
capabil_f(t3)
needs_f(to)
capabil_f(t1)
Functionality gap of the 1. version
t0
Wrong Assumption
Time (not linear)
t0 begin analysis
t1 first Version
t2 capability = = needs(t0)
t3 development concluded
t4 =~ t0: Begin new abalysis
t5 =~ t1: first new Version


Functionality (not linear)
Ultimated appropriated system: = 0
-> Capabilities = Needs
User Needs (moving target)
Inappropriateness
Shortfall Functionality Gap
Actual system capabilities
Lateness
Adaptability = 0
Adaptability
Ungeeignetkeit
Longevity
Time (not linear)

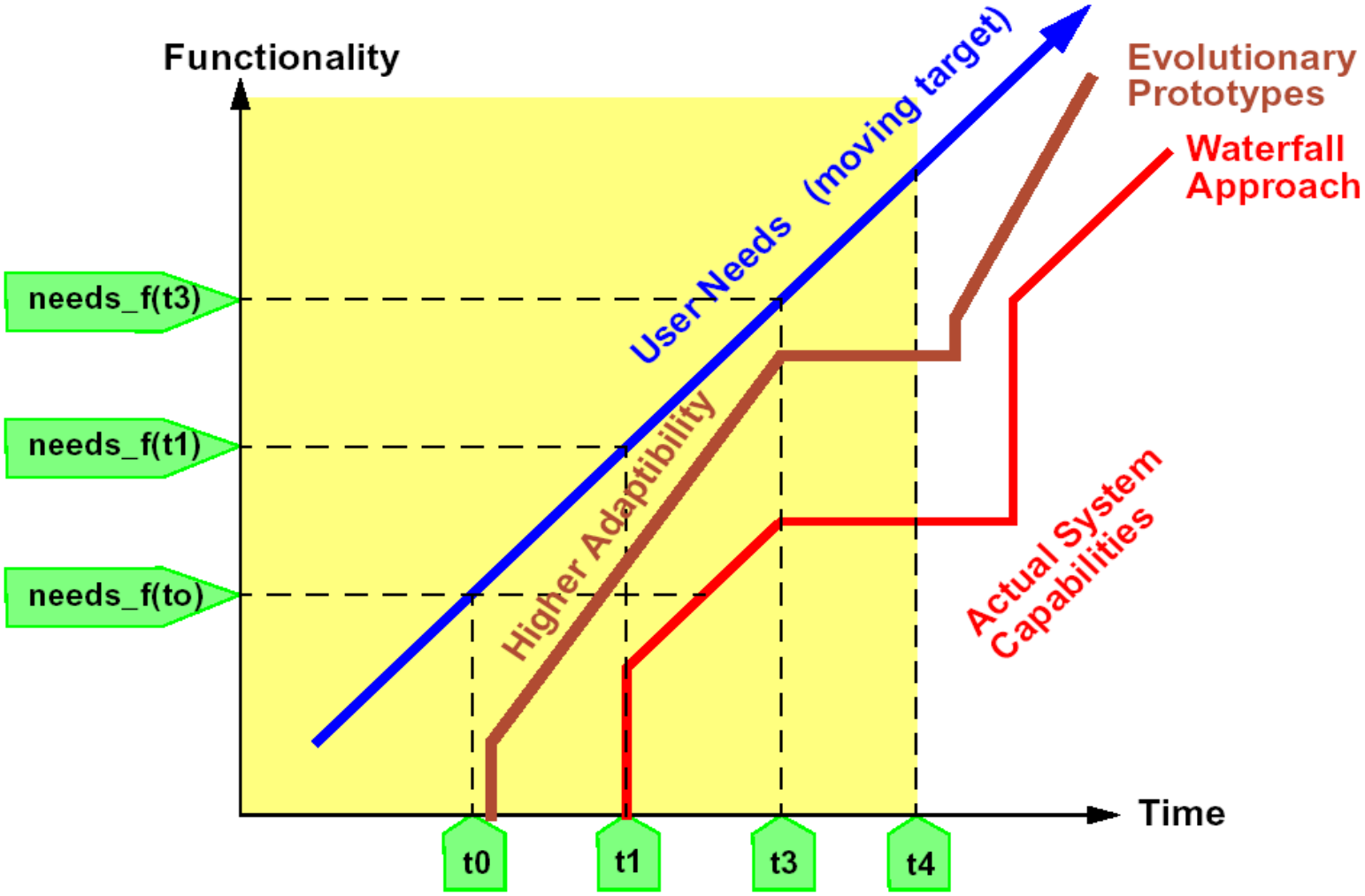
Rapide prototype



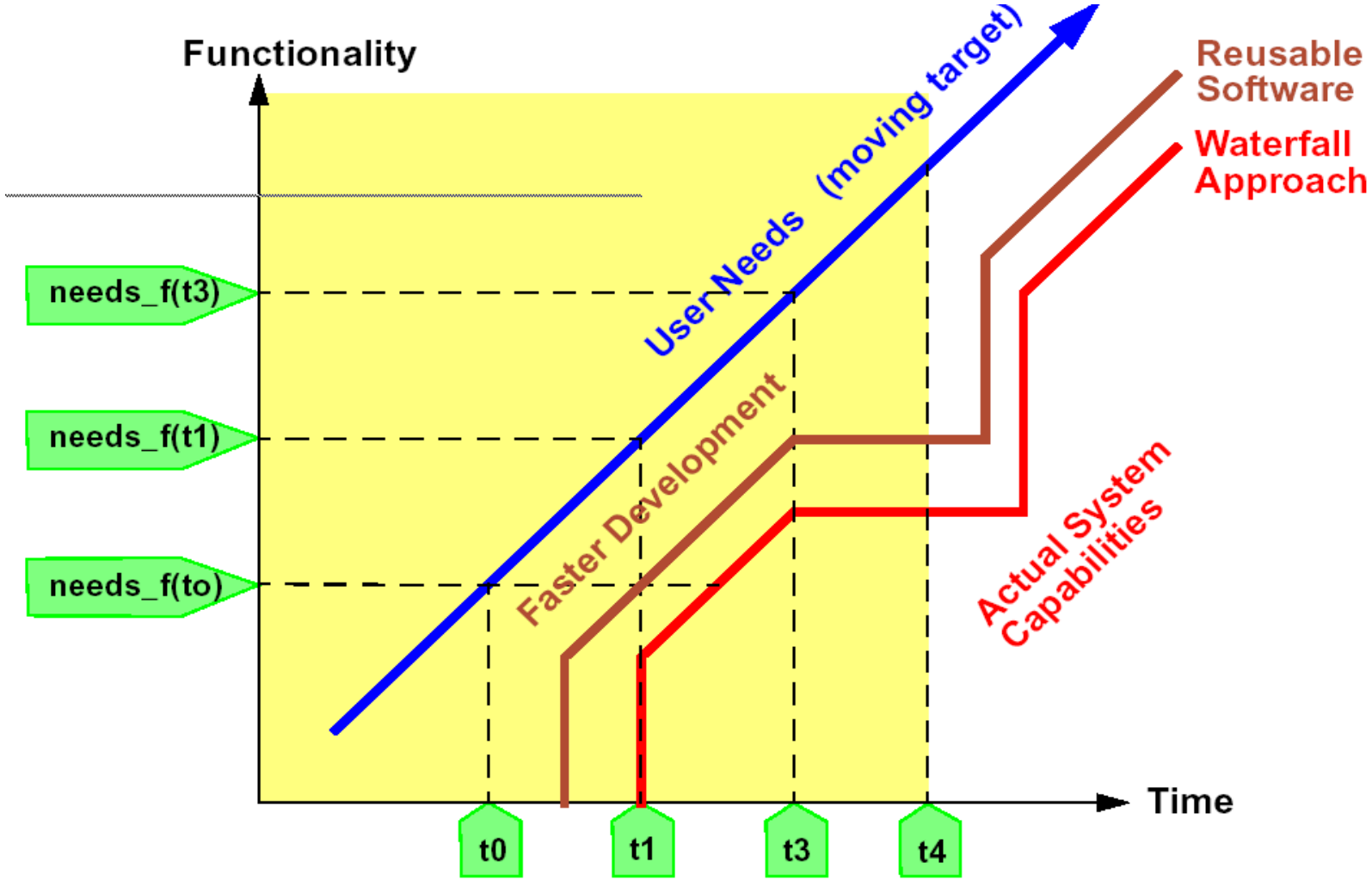
Incremental Development



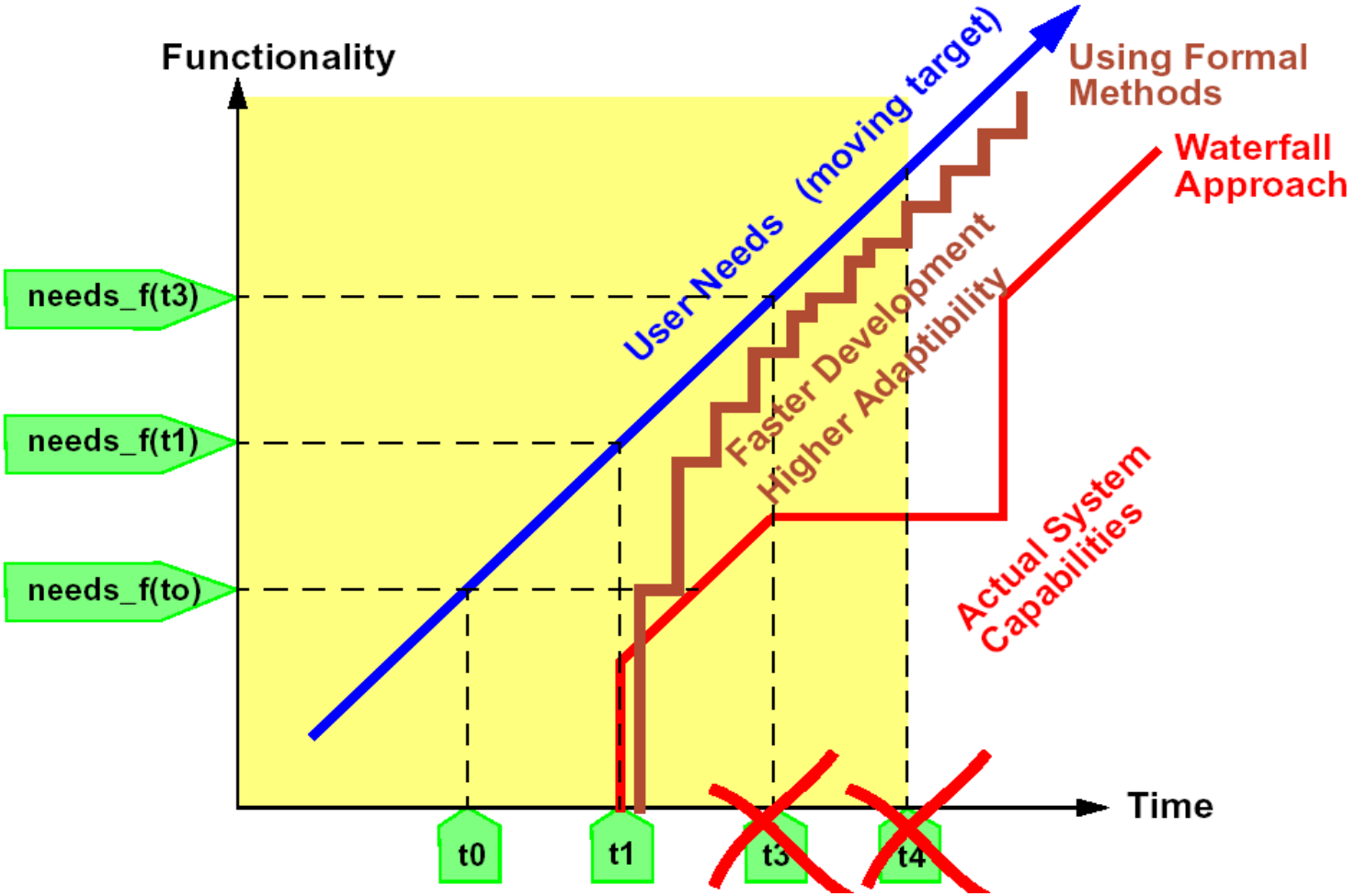
Eveolutionary Prototypes



Reuse


Functionality
needs_f(t3)
needs_f(t1)
needs_f(to)
UserNeeds (moving target)
Faster Development
Reusable Software
Waterfall Approach
Actual System Capabilities
Time
t0
t1
t3
t4

Mit Formale Methoden & Code Generierung



Und am Ende... doch Fehlerhaft!