Softwareentwicklung im Team

„Anything goes“ vs. Methodik

Walter Kriha und Dr. Bernhard Scheffold
Walter.Kriha@ubs.com Bernhard.Scheffold@ubs.com

Ausgangsfragen

- Welchen Beitrag leisten Methodiken zum Erfolg eines Softwareprojektes?

- In welchem Umfeld kommen sie zum Tragen

- Welches Wechselspiel besteht zu den sozialen Strukturen der Arbeitsorganisation?

Motivation: Unsere Nöte

- Gibt es eine Methodik für erfolgreiche *internationale* Softwareprojekte?

- Könnte ein Open Source Entwicklungsstil erfolgreich woanders eingesetzt werden?

- Eine gewisse Ratlosigkeit nach gescheiterten prozessorientierten Projekten

- Der Kampf mit der Methodenpolizei (von Technical Organization Committees, Standard Coms. und anderen Prinzipienreitern)

Eine erste Kategorisierung

Anything goes	**Methodik**
Erfahrene Entwickler	Anfänger
Mapper	Packer
Deliverable	Process
Open Source	Embedded Control
SCRUM	RUP
XP	Waterfall
Kleine Firma/Proj.	Grosse Firma/Proj.
Hacker	Requirements/Specs steuern Prozess

In welche Spalte kommt „**Erfolg**“?

Zweifel an dieser Dichotomie!

- OpenSource hat sowohl Aspekte, die es in der einen bzw. anderen Sparte einordnen lassen (Peer Review, Tool Chain). Ähnliches gilt auch für ClosedSource.

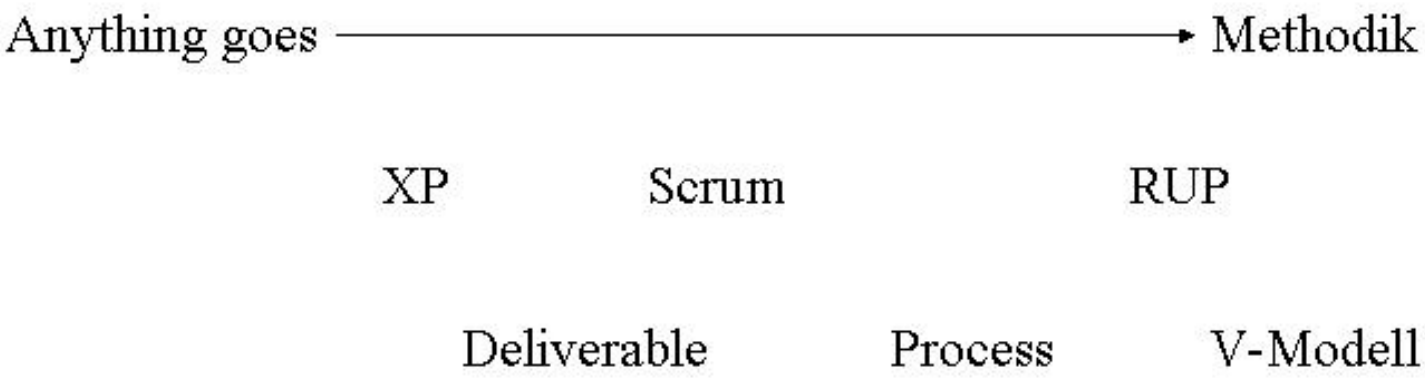
- „Hacking“ bedeutet nicht, dass kein Plan oder keine ordnende Idee vorhanden wäre

- SCRUM bzw. XP haben durchaus Methodik

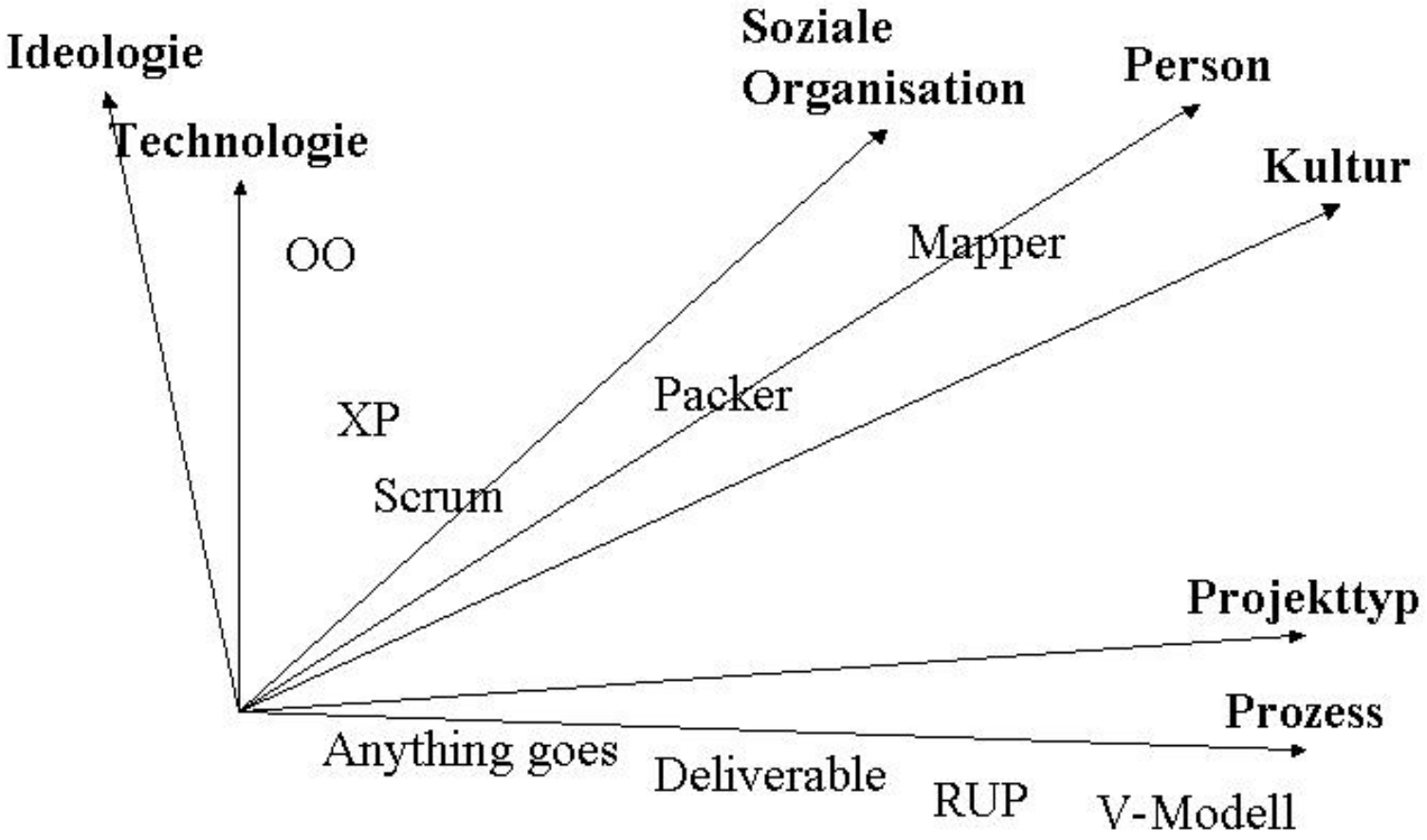
Die Wurzeln des „Anything Goes“

- Die Methodenbürokratie (Kommittees, Standards)
- Die Projektbürokratie (Resourcenbeschaffung, Organisation
- Rückbesinnung auf das eigentliche Problem: Programmieren von Software
- Emotionaler Konflikt: Entwicklersicht (intelligent, kreativ, kompetent, ...) mit Managementsicht (austauschbares Zahnrad)

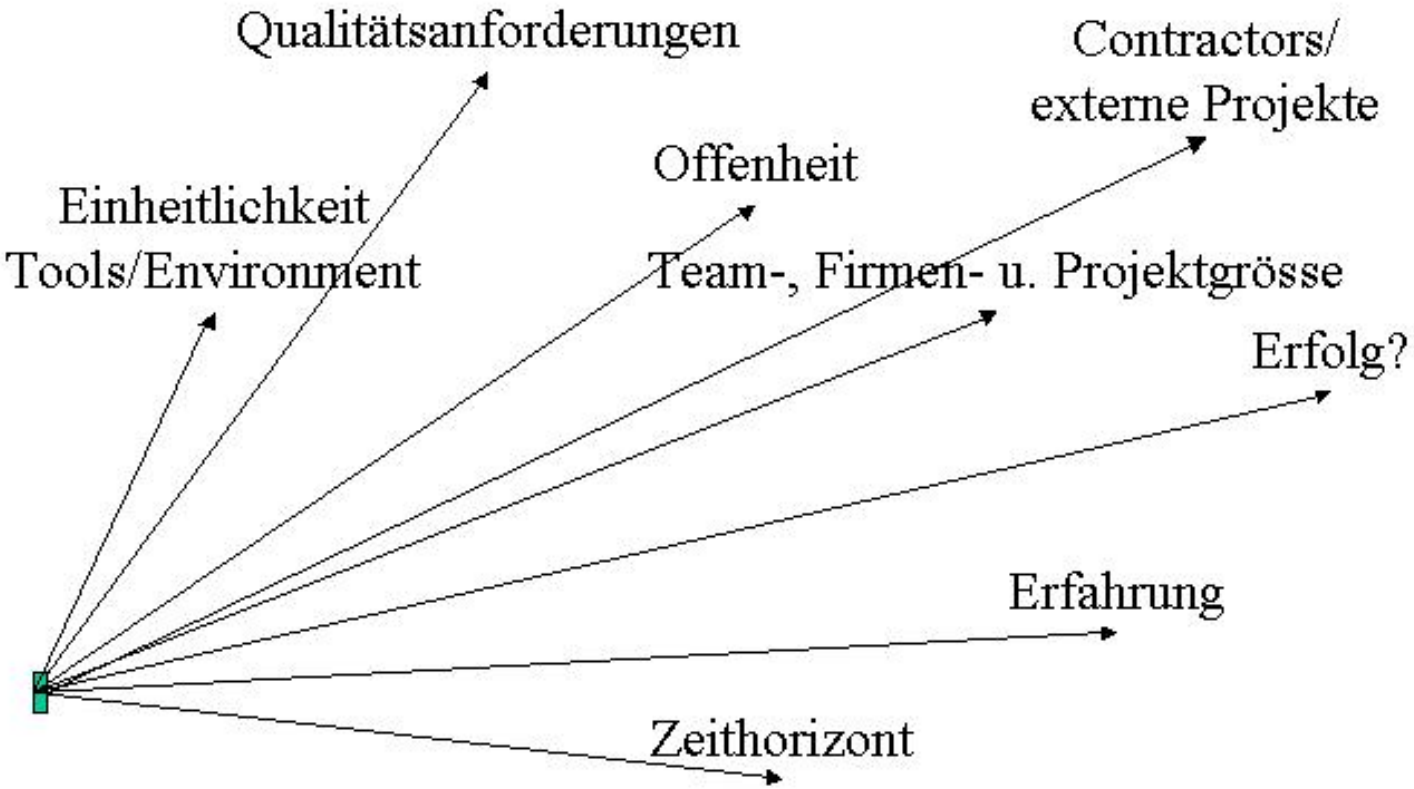
Zunehmende formale Zwänge



Dimensionen der Vorgehensweisen



Weitere Dimensionen der SE



Erstes Ergebnis

„Anything Goes“ gibt es praktisch nicht als Vorgehensweise. Fast jede Vorgehensweise hat methodische Anteile – sei es in der technischen, sozialen, Prozess- oder persönlichen Dimension.

Ebensowenig lässt sich die Frage nach dem Erfolg einfach durch Kategorisierung beantworten. Deshalb sollen jetzt persönliche Erfahrungen auf den jeweiligen Anteil bzw. die Rolle von Methodiken untersucht werden.

Beobachtungskategorien

Technologie Ideologie	Soziales Kultur	Prozess Projekt

Erfahrungen

- Unix Kernel Entwicklung (Sinix) [1986-1991]
- Embedded Control – Military Project [1992-1993]
- OO-Framework (Polytool) [1994-1996]
- Kreditapplikation [1997-1998]
- Verteiltes OO-Bankensystem [1997-1998]
- Aktuelle Finanzapplikation [1999-]

Sinix

Projekttyp **Prozess**	**Technologie** **Ideologie**	**Soziales** **Kultur**	**Erfolg**
In-house System-Entwicklung Kein definierter Entwicklungsprozess Planungshorizont 3 Monate	„Unix“ als Systemmetapher Nur Code/Runtime zählen Source-, Version-Ctrl Automatic Build, Frühe Vernetzung, sehr gute Infrastruktur, gepflegt durch eigene Leute	Source-Code-orientiert, Persönliche Qualifikation, Keine Rollentrennung Unpolitisch Open-door, 30% Kontraktoren (die besten)	Starkes Wachstum, Hohe Qualität Wirtschaftlicher Erfolg Kaum gescheiterte Entwicklungen

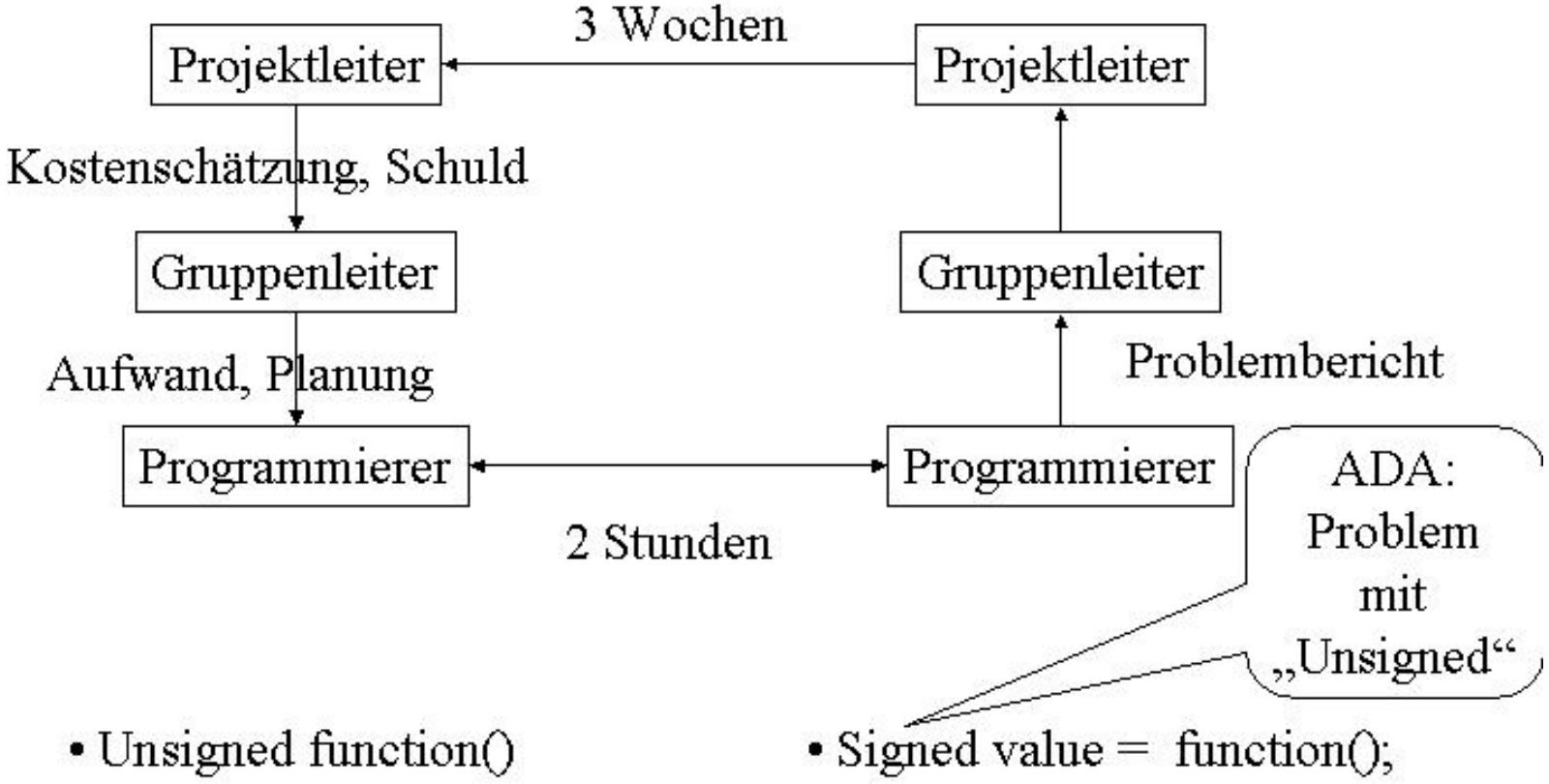
Unix als Lebenswelt

- Die Systemmetapher macht explizite Requirements, Designs und Methodik unnötig
- Die Unix Umgebung sorgt für die gleiche Sprache zwischen den Entwicklern
- Techniken und Problemlösungen werden „vererbt“

Militärische Applikation

Projekttyp **Prozess**	**Technologie** **Ideologie**	**Soziales** **Kultur**	**Erfolg**
In-house System-Entwicklung, embedded Control Externe SW-Partner beteiligt, Vorgehensmodell vor-geschrieben, Realität: informelle Requirements, konstruktives Hacking, teilweise im Stil von (Xtreme) Scrum Nur bei Hardware "Test" Prozesse	„Hardware-Qualität" als Systemmetapher, Realität: Portierung von PD Software, „Test" extrem wichtig Source-, Version-Ctrl Frühzeitige Vernetzung, Gute Infrastruktur, gepflegt durchs eigene Projekt	Source-Code-orientiert, Persönliche Qualifikation, Keine Rollentrennung Unpolitisch, keine Kontraktoren, enge persönliche Beziehung	Erfolgreich abgeschlossen durch grossen persönlichen Einsatz des Teams. Erfolg durch Umgehung der Methodik

V-Modell: Theorie u. Praxis



OO-Framework (Polytool)

Projekttyp **Prozess**	**Ideologie** **Technologie**	**Soziales** **Kultur**	**Erfolg**
In-house Reengineering Projekt Anfangs Coad/Yourdon (Mismatch) Später eigene Toolentwicklung und Methodik aus Technologie/ Ideologie heraus	OO, Design Patterns als Kommunikationsmittel Source-, Version-Ctrl Automatic Build, späte Vernetzung, Projektinfrastruktur gepflegt durchs Team, Frameworking, C++/CORBA Know How	Source-Code-orientiert, Persönliche Qualifikation, konstantes Lernen Keine Rollentrennung Unpolitisch, keine Kontraktoren, enge persönliche Beziehung, Technik treibt Spezifikation	Erfolgreich abgeschlossen durch grossen persönlichen Einsatz des Teams. Framework-Technologie begann sich auszuzahlen

Akito Morita (Sony): Walkman

Der Walkman war ein Projekt der Sony Entwicklung – eine Vision ohne Beteiligung von Marketing/Produktplanung etc. Requirements kamen aus der Entwicklung selbst.

Kreditapplikation

Projekttyp **Prozess**	**Technologie** **Ideologie**	**Soziales** **Kultur**	**Erfolg**
Applikation für Eigenbedarf Analyse und Endusertest im Haus, Coding bei Softwarehaus Pendenzen-System, Gantt-Charts Weekly Build	Fat-Client OO, C++ Komponenten Source-Ctrl Version-Ctrl Infrastruktur von eigener Gruppe	Binaries Rollenabgrenzung (Architekt, BO, DO, PM) Politik Resourcen-Denken	Funktional aber instabil Scheitern durch Politik

Kreditapplikation Umfeld

Resourcendenken

Entwickler/ Skill	OOA	OOD	C++	Java	Basic
Hugo	2	3	1	1	1
Petra	1	0	3	0	5

Qualitätsbewusstsein

```
ClassA::acceptConcreteBO (BO* p)
{
    CBO* q = dynamic_cast<CBO*>(p);
    q->foo(); // q ist ungeprüft!
    // ...
}
```

statt (besser!)

```
ClassA::acceptConcreteBO (ConcreteBO* p)
{
    p->foo();
    // ...
}
```

Aktuelles Finanzprojekt

Projekttyp **Prozess**	**Technologie** **Ideologie**	**Soziales** **Kultur**	**Erfolg**
Applikation für Eigenbedarf	Web-Applikation Batch-Processing	Eine Sandbox für alle Entwickler	Projekt noch nicht abgeschlossen
Analyse/Entwicklung/ Test im Haus	Perl, PL/SQL, JavaScript, Java	Keine strikte Rollenabgrenzung	
Spezifikation	Source-, Release-Ctrl	Technologiefokus	
Keine Methodologie	Infrastruktur von eigener Gruppe	Unpolitisch	
Meilensteine/Reviews/ Anwender-Tests		Viele Kontraktoren	
		Resourcen-Denken	

Finanzprojekt – Technologie

- Entwickler werden hinzugenommen, um Resourcenknappheit zu lösen
- Ein neuer Entwickler bringt „seine“ Technologie mit (Perl, Java, PL/SQL)

Verteiltes OO-Bankensystem

Projekttyp **Prozess**	**Technologie** **Ideologie**	**Soziales** **Kultur**	**Erfolg**
In-house Reengineering Projekt. Getragen von externem Personal Requirements, Specifications, Deliverables, Deadlines, Reviews. Prozess dient der politischen Absicherung	Tool-zentrisch. Teure Spezialisten pro Tool bringen Erfolg Keine Systemmetapher Source-, Version-Ctrl Projektinfrastruktur gepflegt durchs Team Framework C++/CORBA Eiffel für OOA und OOD	Binary-orientiert, Rollentrennung Workshops verpönt. Schlechte Kommunikation Politisch, 90% Kontraktoren, Neulinge in OO und Corba Kulturkampf englisch/deutsch extern/intern	Misserfolg

Otelo or you‘ll never walk alone!

- Alle Deliverables da und alle Deadlines eingehalten
- Alle Reviews durchgeführt und erfolgreich abgeschlossen
- Software Technology Process von der IBM benutzt und kontrolliert von der IBM

Resultat:

Nichts greifbares nach 2 Jahren. Hohe Konventionalstrafen. Prozess und Methodologie werden bei der IBM in Frage gestellt.

Erfolgsfaktoren

Harte Faktoren

Weiche Faktoren

Projekterfolg in Relation zu Prozess/Projekt

- In unseren Projekten haben wir keinen Prozess bzw. Methodologie gefunden, der sicher zum Erfolg führt.
- Tatsächlich schienen etliche der Projekte ganz ohne definierten Prozess auszukommen.
- Pikanterweise scheiterten gerade die Projekte, die am meisten auf Prozessmethodik (Requirements, Specifications, Deliverables, Reviews etc.) setzten.
- Soziale/kulturelle Factoren beim „Team Building“ scheinen wichtiger zu sein.
- Sprachbarierren sind in Architektur- und Design-Diskussionen besonders kritisch.
- Ein strikter Fokus auf die Technologie – keine Politik, keine Formalismen – scheint wichtig zu sein.

Projekterfolg in Relation zu Technologie/Ideologie

- In unseren Projekten haben wir keine Technologie gefunden, die sicher zum Erfolg führt.
- Auch die Kenntnisse der Teammitglieder bezüglich neuer Technologien sind nicht wirklich bedeutsam.
- Wichtiger scheint ein starkter Technologiefokus zu sein: die Bereitschaft Technologie als eigenes Gebiet anzuerkennen und ernst zu nehmen
- Dies ist eng verknüpft mit persönlichen Faktoren wie dem Stolz auf die Beherrschung von Technologie bis hin zur Bereitschaft selbst Requirements zu erstellen – aus der Kenntnis der Technologie heraus (Beispiel: Sony's Walkman)
- Wichtig für den Technologiefokus ist auch ein weitgehend politikfreies Umfeld

Projekterfolg in Relation zu Sozialer Organisation/Kultur

- Generell sehen wir hier den grössten Einfluss auf den Projekterfolg.
- Funktionierende Teams sichern Erfolg. Person statt “Resource”. Das Team als kleinste Einheit von Resourcen.
- Kulturkämpfe (und Sprachdefizite) können gerade während Analyse und Design verheerene Wirkung haben
- Kontraktoren: Sehr problematisch, ebenso wie In-House Entwicklung bei nicht-Software Firmen

Rational Unified Process – „Kauf dir einen Erfolgsprozess“

- “Der Prozess sichert den Erfolg”
- Soziale Faktoren sind nicht Teil von RUP (und werden damit implizit auch nicht als wichtig angesehen).
- Der Technologiefokus ist gering, Prozess dominiert. Die starke Use-Case-Orientierung ist nicht ungefährlich für Systemdesigns.

Die Gewichtung von Prozess/Methodologie, Technologie und sozialer Organisation widerspricht den Erfahrungen aus unseren Projekten.