



Use Case Recycling by Extracting Business Rules

Rolf Götz

rolf.goetz@gmx.de

@rolfgoetz

- Warum Wiederverwendung gut für **Qualität** und **Produktivität** ist.
- Wie dies bei Use Cases sinnvoll angewendet werden kann.

Wie viele Fehler hat die durchschnittliche Applikation?

- grob vereinfachende Frage!
(hat? durchschnittlich? Applikation? Wann?)
- grob vereinfachte Antwort (<- C. Jones, 2008)
 - Potenzial zu 5 pro Function Point
 - Fehlerbehebungseffektivität 85% bis Roll-Out
 - → 0,75 pro Function Point
- Beispiele (Schätzungen)
 - Apple iPhone ca. 14.000
 - Toyota Prius Motor ca. 750
 - Enterprise Java Beans ca. 4.500

nicht alle gleich!

20% der Fehler stammen aus den Anforderungen



Defektpotenziale in
Fehler / Function Point

Requirements defects	1.00	} U.S.-Durchschnitt 2007
Design defects	1.25	
Coding defects	1.75	
Documentation defects	0.60	
Bad fixes	0.40	
Total	5.00	

Quelle: C. Jones, Applied
Software Measurement;
McGraw Hill, 2008

Anforderungsfehler sind teuer



- Anforderungs-Fehler sind schwer zu finden: **0,23 / FP** <- Tom Gilb
- So einen Fehler zu bemerken, zu finden, zu analysieren, zu beheben, ... den Fix in Betrieb zu nehmen kostet **9,3 Stunden Aufwand** <- Raytheon
- Eine Applikation
 - mittlerer Größe (1.000 FP)
 - hat 230 problematische Anforderungsfehler
 - deren Behebung 2.100 Stunden kostet
- ➔ **168.000 EUR** bei 80 EUR / h

Die *frühen* Phasen sind die für die Reduzierung von Fehlern wirtschaftlich sinnvollen Phasen (NICHT: Testen)

Ähnlich dem Prinzip des Größennachteils gilt in der SW-Entwicklung das *Prinzip des Nachteils der späten Phasen*.

Größennachteil / diseconomy of scale:

- Der für die Erstellung von Software nötige Aufwand nimmt mit der Komplexität der Software überproportional zu.
- Nimmt Komplexität zu, so nimmt der Aufwand noch mehr zu.

Phasen-Nachteil / diseconomy of phase:

- Der für die Behebung von Fehlern in der Software nötige Aufwand nimmt mit dem Abstand zwischen Ursache und Behebung *überproportional* zu.
- *Je später* ein Fehler behoben wird, *desto teurer* ist die Behebung (aufgrund notwendiger Nacharbeiten und weil manche Fehler später schwerer zu finden sind).

Wiederverwendung kann sehr geeignet sein, um Qualität und Produktivität zu steigern

	A	B	C	D	E	F	G	H	I	J	K	L	M
5													
6		Type 1 = Information technology and web applications						Small = < 1000 function points					
7		Type 2 = Systems, embedded, and commercial applications						Medium = 1000 to 10,000 function points					
8		Type 3 = Government and military applications						Large = > 10,000 function points					
9													
10		Methodology				Small	Medium	Large	Type 1	Type 2	Type 3	TOTAL	Average
11	Rang					< 1000		> 10,000					
12													
13	1	Reusability (zero-defect materials)				9,10	9,30	9,50	9,50	9,50	9,50	56,40	9,40
14	2	Personal Software Process (PSP)				9,13	9,25	9,20	9,40	9,26	9,28	55,52	9,25
15	3	Team Software Process (TSP)				8,00	9,50	9,33	9,45	9,27	9,30	54,85	9,14
16	4	Automated static analysis				9,10	9,00	9,00	9,20	9,35	9,00	54,65	9,11
17	5	Hybrid (CMM+TSP/PSP+others)				7,90	9,00	9,25	9,00	9,30	9,22	53,67	8,95
18	6	Inspections (code)				8,20	8,00	9,30	8,80	9,50	9,37	53,17	8,86
19	7	Object-oriented development (OO)				9,00	9,00	9,00	8,00	9,20	8,75	52,95	8,83
20	8	Automated security testing				7,00	8,00	9,30	8,80	9,20	9,15	51,45	8,58
21	9	Formal security analysis				5,00	8,00	9,50	9,30	9,25	9,50	50,55	8,43
22	10	Agile development				9,50	9,00	5,50	9,35	9,10	8,00	50,45	8,41
23	11	Inspections (requirements)				5,50	8,10	9,50	9,00	9,00	9,30	50,40	8,40
24	12	Time boxing				9,10	8,20	8,50	8,00	8,00	8,50	50,30	8,38
25	13	Formal risk management				5,00	8,00	9,00	9,00	9,00	9,00	49,00	8,27
133	121	Inadequate security controls				(2,00)	(4,00)	(9,00)	(9,90)	(10,00)	(10,00)	(44,90)	(7,48)
134	122	Inadequate progress tracking				(1,00)	(5,00)	(9,00)	(10,00)	(10,00)	(10,00)	(45,00)	(7,50)
135	123	Excessive schedule pressure				(1,00)	(5,00)	(10,00)	(9,50)	(9,75)	(9,75)	(45,00)	(7,50)
136	124	Inadequate cost tracking				(1,00)	(6,00)	(10,00)	(9,50)	(10,00)	(10,00)	(46,50)	(7,75)
137	125	Reusability (high defect volumes)				(2,00)	(6,00)	(9,00)	(10,00)	(10,00)	(10,00)	(47,00)	(7,83)

Quelle: C. Jones, SCORING AND EVALUATING SOFTWARE METHODS, PRACTICES, AND RESULTS, Nov 2008

Ziel: Reduziere die Anzahl der Anforderungsfehler



BENCHMARK [U.S.-Schnitt 2007]: 0,23 / FP

GOAL [wir, Ende 2010]: 0,15 / FP

STRETCH [2010]: 0,10 / FP



Ambition: Kosten für Überarbeitung reduzieren

Strategie: Fördere Zero-Defect-Wiederverwendung von Use Cases

Was macht einen Use Case einzigartig?

- Use Cases sind strukturell oft ähnlich
 - innerhalb einer Applikation
 - über mehrere Applikationen hinweg

Stoßrichtung 1:
Abstraktion /
Standardisierung
der Abläufe

- Veränderlich sind die Geschäftsregeln
 - definieren „das Geschäft“
 - ändern sich von Zeit zu Zeit

Stoßrichtung 2:
Verbesserung der
Regelformulierung

Qualitätsziele für Use Cases

- **Lesbar.**
 - Umfassend und verständlich für die Leserschaft
- **Klar.**
 - Klar und vollständig genug, um Designs und Tests abzuleiten
- **No Design.**
 - Keine unnötigen Design-Einschränkungen
- **Strukturierend.**
 - Decken die Grenzen zwischen den verschiedene Geschäftsmodulen auf
- **Wieder verwendbar.**
 - Innerhalb der Applikation und über Applikationen hinweg
- **Änderbar.**
 - Über die Laufzeit der Applikation hinweg, und auch darüber hinaus



Ein Use Case aus der Praxis

Schritt	Name	Beschreibung
#1	Optionen anzeigen	Das Mobilgerät zeigt die Optionen an.
#2	Bestellung wählen	Der registrierte Gast wählt "Bestellung" auf dem Mobilgerät.
#3	Menü anzeigen	Das Mobilgerät zeigt das Menü des Gastes an, mit allen Gerichten, Beilagen und Getränken. <i>Kommentar: nicht die Sachen anzeigen, von denen wir wissen, dass der Gast sie nicht mag.</i>
#4	Bestellung a	... eweilige Checkbox
#5	Bestellung b	... m Mobilgerät
#6	Bestätigung	... ails.
#7	Verfügbarkeit prüfen	Das Mobilgerät klärt mit der Küche, ob die bestellten Sachen vorhanden sind. <i>Kommentar: Das System darf keine Bestellung erlauben, die nicht verfügbar ist.</i>
#7.1	Fehlermeldung zeigen	Falls ein oder mehrere Sachen aus der Bestellung nicht verfügbar sind, zeigt das Mobilgerät eine Meldung mit den Details an. Zurück zu Schritt #3.
#7.2a	Bestand neu berechnen	Falls alle Sachen verfügbar sind, subtrahiert das Mobilgerät je 1 von der Anzahl der vorhandenen Sachen.
#7.2b	Bestellung anschicken	Das Mobilgerät schickt die Bestellung an das Küchen-Display.

WARUM?
 Weil meine Altanwendung das so macht.
 Weil ich mir das so vorstellen kann.
 ...



Fokus

Geschäftsregel:

- Einschränkung
- Richtlinie
- Voraussetzung
- Berechnung
- Bedingung

Extrahierte Geschäftsregeln

Nr.	Name	Beschreibung	Typ
#1	Status Gast Registrierung	Gäste müssen registriert sein, um das Gerät nutzen zu können.	Einschränkung
#2	Angebot	Gerichte und Drinks dürfen nicht <unliebsam> sein.	Einschränkung
#3	Verfügbarkeit	Das System soll keine <Bestellung annehmen> für Sachen, die nicht verfügbar sind. <i>Notiz: nicht anbieten ??</i>	Richtlinie
#4	Bestand reduzieren	Jede Bestellung reduziert den Bestand um die Anzahl der bestellten Sachen.	Berechnung
#5	Mögliche Bestellung	Bestellt werden können Gerichte, Beilagen und Getränke.	Einschränkung



Der leichtere Use Case

Schritt	Name	Beschreibung
#1	Bestellung wählen	Der Gast wählt die Bestell-Funktion.
#2	Menü anzeigen	Das Mobilgerät zeigt das Menü des Gastes an.
#3	Bestellung auswählen	Der Gast wählt aus den Gerichten, den Beilagen und den Getränken.
#4	Küche informieren	Das Mobilgerät informiert die Küche über die Bestellung.

Schritt 1: Geschäftsregeln extrahieren

Process: Extracting Business Rules

Tag: process for extracting Business Rules from Pragmatic Use Cases; Version: 2009-03-10; Owner: Rolf.Goetz@planetproject.com; Status: revised draft;
Source: <http://planetproject.wikidot.com/improve-reusability-of-use-cases-by-extracting-business-rule>

Entry Conditions

- E1: Use cases have been written (and validated by subject matter experts (SMEs)) *â€™Sometimes it's better to have SMEs to validate business rules when they see them already separated from the rest. Sometimes not.*
- E2: You are prepared to document the extracted business rules properly. *â€™Be it as a separate document or in a separate case template (simple) or in a list of business rules for all use cases (advanced, and far better for more than 5 use cases).*

Steps and Notes

- S1: Assume you have a use case with sections of some sort for a description, trigger conditions and normal / other flows. *â€™other written material that says something about the business could be just as good.*
- S2: In every section find statements that constrain the use case. Look for the words "must be", "must have", "make sure that", and also "always", "every", "never", "no", "all", "may", "can". *careful here, make sure the statement is true, i. e. validate it.*
- S3: In every section find statements that enable actions. Look for the words "when", "if", "as soon as", "in case", and time triggers.
- S4: In every section find statements that describe computations. Look for the words "and", "divisors", formulas in general. *â€™What else can be named a computation?*
- S5: In every section find statements that describe inferences. Look for the words "if â€¦ then".
- S6: In every section find statements that describe definitions. Look for the words "define". *A classical rule to be implemented as a parameter.*
- S7: Especially in the flow sections, look at every decision point and find the rules that govern the decision making in this particular spot.
- S8: Search the Normal, Alternative and Exceptional flows for concrete values like key-words, number ranges, states. Their abstractions are the business rules. *Note: this also helps in identifying the essential steps in a use case.*
- S9: For every statement found, see if it can be re-written into a whole new sentence AND re-write the remaining part of the description by using a reference to that new sentence. If you can do this, you have probably found a business rule in that statement. Give it a name and or a number for reference purposes.
- S10: Make sure your Business Rules can be clearly distinguished from the rest of the specification. *â€™By putting them in an extra chapter of your specification.* Using a word processor consider using references, so that you only need to write each rule once, and display it many times.
- S11: Consider writing a requirement that requires some robustness concerning the change of these rules. *â€™you will have to predict in which circumstances and to what extent you expect the rules to change*

Exit Conditions

Once you have completed the process, if you are sure there are no more Business Rules hiding. You can do this by using the above definition of Business Rules as a checklist for sifting through all use cases.

In some practice you can use steps S2 through S8 to form questions for interview purposes.

Welcome page

FAQ

Please use the Templates

Site members

Honor Roll

Recent changes

List all pages

Site Manager

Page tags

agile arguments change defects downloads
estimation evolution goals lean offshore
principles priorities process processes project
quality requirements risk rules tasks template
tenders thinking usecases writing

Add a new page

Most wanted articles

1. [Improve the Re-Usability of Use Cases by Extracting Business Rules](#)
2. [Use Case Content Patterns](#)
3. [Rules for Checking Use Cases](#)
4. [List All Pages](#)
5. [Specifying Goals](#)

[More...](#)

What others say

"Your articles provide great help finding the real problems!" - Harjo K.

"There is so much wisdom buried in projects. Rolf does a great job in digging it out and write down the stuff with no fluff." - Sven B.

[More...](#)

Stay updated

 Custom RSS Feeds

5 Powerful Principles to Challenge Arguments

Type: Principles

Sources: Scott Berkun: [How to Detect Bullshit](#); and my all-time-favourite checklist: [12 Tough Questions](#) by Tom Gilb.

[fold](#)

Table of Contents

- [Gist](#)
- [Summary of Principles](#)
- [Principles and Notes](#)
- [Related Pages](#)

Gist

to provide a general checklist for the suspicious. The 5 principles show behavioural patters people use to make you do things in their favour, but not necessarily in your fa advisors, colleagues, consultants, counsels, managers, salespersons, vendors.

Summary of Principles

1. People are uncertain and tend to ignorantly stretch the facts.
2. People with weak arguments often have not made their homework on the topic.
3. People tend towards urgency when you are asked to make a decision with some hidden consequence.
4. People without a clear understanding of their point of view tend to inflate the language used.
5. People tend to have stronger arguments if they know someone is present who is hard to deceive.

Principles and Notes

P1) People are uncertain and tend to ignorantly stretch the facts.

- How do you know?
- What are your sources? How can I check them?
- Who told you that?
- Can you quantify the improvement?

Note: Carefully watch the answerer. If he needs a while, maybe uncomfortably shifting position, there's a good chance he's either making something up or needs time to f argument.

P2) People with weak arguments often have not made their homework on the topic.

- What is the counter argument?
- Who else shares this point of view?
- What are the risks of this, and what will you do about it?
- Can you quantify the improvement?
- How does your idea affect my goals and my budgets?
- What would make you change your mind?
- Have we got a complete solution?

Note: As from any facts, one can draw a set of reasonable interpretations, not just one. Someone with intimate knowledge won't have great difficulties taking a different n



Schritt 2: Use Case Content Patterns

- Muster von Abläufen, Schritten und anderen Merkmalen wiederkehrender Modellierungsprobleme
 - by Martin Langlands
 - sehr universell, dennoch gut verständlich
 - Kontext wird vor allem durch Geschäftsregeln definiert
- Englischer Artikel dazu:
<http://planetproject.wikidot.com/use-case-content-patterns>
- -> Hier passt das SELECTOR - Pattern



SELECTOR

Schritt	Beschreibung
#1	<p>Der Akteur definiert die Suche mithilfe einer Menge von Kriterien. Siehe Geschäftsregeln. Kriterien sind z. B. Status Gast Registrierung (Regel #1), Verfügbarkeit (Regel #3), Mögliche Bestellung (Regel #5)</p>
#2	<p>Das System findet Objekte, die den Kriterien genügen. Dadurch ergibt sich die Menge von Gerichten, Beilagen und Getränken, die dem Gast angezeigt wird</p>
#3	<p>Das System zeigt die gefundenen Objekte an, mit definierten Merkmalen für jedes. Merkmale: Name, Zutaten, Preis, schon-mal-bestellt</p>
#4	<p>Der Akteur wählt ein oder mehrere der angezeigten Objekte aus und bestätigt seine Auswahl. Ich will dies und das. Bitte bring' es. Das ist der Kern des Use Case.</p>
#5	<p>Das System gibt die Steuerung an den aufrufenden UC zurück, mit der Menge der ausgewählten Objekte. Hier ist also der Aufrufer dafür verantwortlich, die Küche zu informieren.</p>

Schritt 3: Geschäftsregeln formulieren

- angelehnt an RuleSpeak™ von Donald G. Ross / Dr. Jürgen Pitschke
- Regelwerk / Satzstrukturen speziell für Geschäftsregeln
- verbessert Klarheit, Eindeutigkeit, Lesbarkeit
- Deutsche und Englische Version erhältlich

RuleSpeak™-DISCLAIMER: Diese Unterlagen können intern frei für nicht-kommerzielle Zwecke benutzt werden. Die kommerzielle Nutzung oder Weiterverbreitung jeglichen Teils dieser Unterlagen ist ohne Zustimmung von Business Rule Solutions, LLC (BRS) und BCS-Dr. Jürgen Pitschke nicht gestattet. Für Lizenzen und Weiterverwendung sprechen Sie uns bitte an. Kopieren Sie diese Notiz in jede Reproduktion.

RuleSpeak™-Beispiel

- Verfügbarkeit [ohne RuleSpeak™]:
 - Das System soll keine Bestellung annehmen für Sachen, die nicht verfügbar sind.
- Verfügbarkeit [mit RuleSpeak™]:
 - Das System darf Bestellungen nur für verfügbare Sachen annehmen.
- Verbesserungen
 - Qualifikation (ohne Nebensatz, „verfügbar“) nun kürzer, damit Regel insgesamt prägnanter.
 - Klare Verbindlichkeit („darf nur“ vs. „soll“)

Zusammenfassung

- Geschäftsregeln aus UC extrahieren
- + Use Case Content Patterns anwenden
- + Geschäftsregeln klar formulieren
- = stabile, wiederverwendbare, klare Abläufe
 - + veränderliche, verständliche, änderbare Geschäftsregeln
- ➔ weniger Anforderungsfehler
- ➔ geringere Überarbeitungskosten

Kommentare und Anfragen bitte jetzt direkt oder an:

rolf.goetz@gmx.de

follow me on Twitter: @rolfgoetz
ClearConceptualThinking.net
PlanetProject.wikidot.com