Softwaretechnik

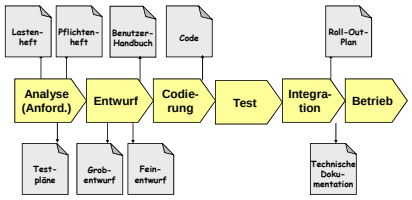
Zusammenfassung
zur Systematischen Entwicklung

1. Wieso Software Engineering?
2. Vom Einzelkämpfer zum Großprojekt

Systematische Softwareentwicklung

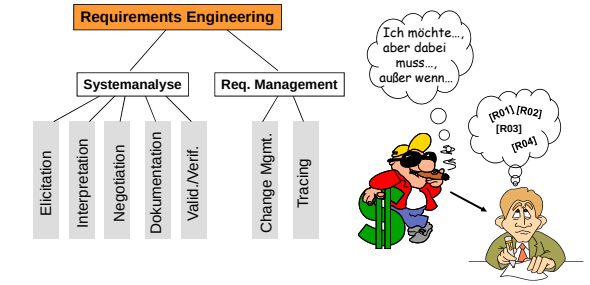
3. **Anforderungen und Test**
4. **Entwurf: Strukturen und NF Eigenschaften**
5. **Entwürfe notieren mit UML**
6. **Design Patterns**
7. Management: *Technik und Projektmanagement*

Software-Entwicklung: Der Ablauf
Systematik

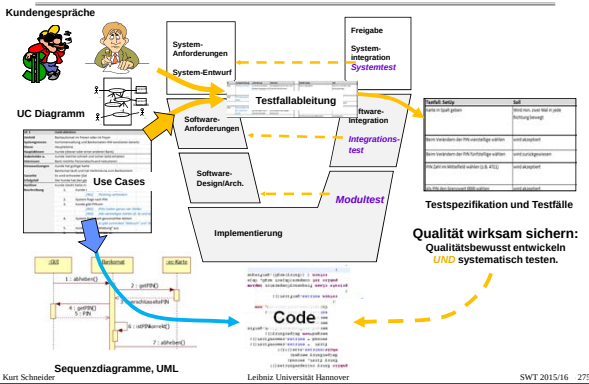


Was soll entwickelt werden?
Systematik: Requirements Engineering

Analyse (Anford.)

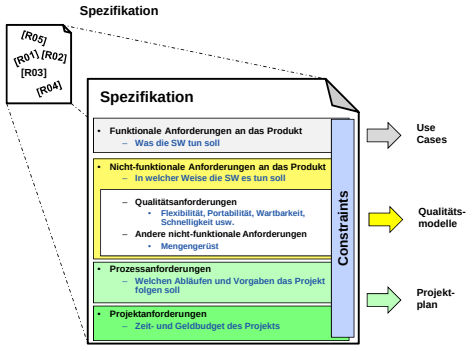


Software-Entwicklung und -Test
Systematik



Anforderungstypen in der Spezifikation
Systematik

Analyse (Anford.)



Gliederung eines Lastenhefts
Systematik

Analyse (Anford.)

1 Aufgabe und Grenzen
- 1.1 Was sind Aufgabe und Grenzen?
- 1.2 Stakeholders - wen betrifft es?
- 1.3 Was ist *in* den Grenzen, was nicht?

2 Begriffe und Glossar

3 Use Cases
- 3.1 Hauptakteure und ihre Hauptziele
- 3.2 Use Cases für Business-Prozesse
- 3.3 Use Cases für System-Prozesse

4 Eingesetzte Technologie - Anforderungen und Schnittstellen

5 Andere Anforderungen
- 5.1 an das Entwicklungsprojekt (Teilnehmer, Feedback, Abh.)
- 5.2-5.n Business rules, Performance, Security, Documentation, Usability, Maintenance and Portability, Unresolved or deferred

6 Risikovorsorge und organisatorisches Umfeld
- 6.1 Notlösung bei Ausfall; politische Anforderungen; erwartete System-Auswirkungen; Trainings-Bedarf; Annahmen und Voraussetzungen

Funktionale Anforderungen

Prozess- u. Projektanf.

Qualitäts-anforderungen

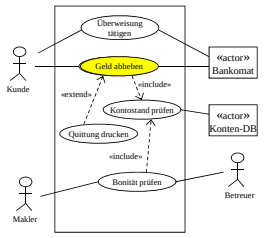
Projekt-management

In Anlehnung an: Cockburn, A.: Writing Effective Use Cases. Addison-Wesley, 2001

Überblick durch Use Case-Diagramme

Systematik

Analyse (Anford.)



Details im Use Case Template

UC Geld abheben (Teil 1)

Analyse (Anford.)

UC 1	**Geld abheben**
Umfeld	Bankautomat im Freien oder im Foyer
Systemgrenzen	Kontoverwaltung und Bankomaten-HW existieren bereits
Ebene	Hauptebene
Hauptakteure	Kunde (dieser oder einer anderen Bank)
Stakeholder u. Interessen	Kunde möchte schnell und sicher Geld erhalten Bank möchte Personalaufwand reduzieren
Voraussetzungen	Kunde hat gültige Karte Bankomat läuft und hat Verbindung zum Banksystem
Garantie	Es wird entweder (Geld ausgezahlt und abgebucht) oder (weder ausgezahlt noch abgebucht
Erfolgsfall	Der Kunde hat den gewünschten Geldbetrag erhalten; dieser Betrag wurde von seinem Kont
Auslöser	Kunde steckt Karte in den Leser
Beschreibung	1. Kunde steckt Karte in den Leser 2. System fragt nach PIN 3. Kunde gibt PIN ein 4. System fragt nach gewünschter Aktion 5. Kunde wählt "Abhebung" aus 6. System bietet Beträge an 7. Kunde gibt Wunschbetrag an 8. System prüft Verfügbarkeit und zahlt Betrag aus 9. System schließt die Sitzung, gibt Karte aus.

Fortsetzung folgt …

Use Case

UC Geld abheben (Teil 2)

Analyse (Anford.)

Beschreibung	1.	Kunde steckt Karte in den Leser
	2.	System fragt nach PIN
	3.	Kunde gibt PIN ein
	4.	System fragt nach gewünschter Aktion
	5.	Kunde wählt "Abhebung" aus
	6.	System bietet Beträge an
	7.	Kunde gibt Wunschbetrag an
	8.	System prüft Verfügbarkeit und zahlt Betrag aus
	9.	System schließt die Sitzung, gibt Karte aus.
Erweiterungen	3a	WENN keine gültige PIN eingegeben, DANN zurück zu 2 (nach drei Fehlversuche
	8a	WENN Wunschbetrag nicht verfügbar ist DANN
	8a.1	bietet das System den höchsten verfügbaren Betrag an
	8a.2	Kunde bestätigt, dass er ihn akzeptiert
		8a.2a: Falls nicht akzeptiert, beende Sitzung (9).
	8a.3	System zahlt Betrag aus, weiter bei 9
Technologie	Bei Geräten mit Irissensor ist alternativ zur PIN-Eingabe auch ein Iris-Scan möglich	

Anforderungen in UC einbetten

oder unter den Use Case schreiben

UC 1	**Geld abheben**	
Umfeld	Bankautomat im Freien oder im Foyer	
Systemgrenzen	Kontoverwaltung und Bankomaten-HW existieren bereits	
Ebene	Hauptebene	
Hauptakteure	Kunde (dieser oder einer anderen Bank)	
Stakeholder u. Interessen	Kunde möchte schnell und sicher Geld erhalten Bank möchte Personalaufwand reduzieren	
Voraussetzungen	Kunde hat gültige Karte Bankomat läuft und hat Verbindung zum Banksystem	
Garantie	Es wird entweder (Geld ausgezahlt und abgebucht) oder (weder ausgezahlt noch abgebucht).	
Erfolgsfall	Der Kunde hat den gewünschten Geldbetrag erhalten; dieser Betrag wurde von seinem Konto abgebucht	
Auslöser	Kunde steckt Karte in den Leser	
Beschreibung	1. Kunde steckt Karte in den Leser	
	[R01] Phishing verhindern	
	2. System fragt nach PIN	
	3. Kunde gibt PIN ein	
	[R02] PINs haben genau vier Stellen	
	[R03] Alle vierstelligen Zahlen (0..9) sind als PIN erlaubt	
	4. System fragt nach gewünschter Aktion	
	[R04] Es gibt zumindest "Abbruch" und "Abhebung"	
	5. Kunde wählt "Abhebung" aus	
	6. System bietet Beträge an	
	[R05] Es wird angeboten: 50, 100, 150, 200, 300 EUR und "anderer Betrag"	
	7. Kunde gibt Wunschbetrag an	
	8. System prüft Verfügbarkeit und zahlt Betrag aus	
	[R06] Als verfügbar gilt: Kontoguthaben plus Überziehungskredit	
	9. System schließt die Sitzung, gibt Karte aus.	
Erweiterungen	3a WENN keine gültige PIN eingegeben, DANN zurück zu 2 (nach drei Fehlversuchen: Sperrung)	
	[R07] Nach drei PIN-Fehleingaben erfolgt Sperrung	
	[R08] Fehlversuche auf Karte speichern und auch in Folgesession mitzählen	
	8a WENN Wunschbetrag nicht verfügbar ist DANN	
	8a.1 bietet das System den höchsten verfügbaren Betrag an	
	[R09] Als verfügbar gilt: Kontoguthaben plus Überziehungskredit	
	[R10] Wenn vorhandene Scheine nicht reichen, kürze ebenfalls	
	8a.2 Kunde bestätigt, dass er ihn akzeptiert	
	8a.2a: Falls nicht akzeptiert, beende Sitzung (9).	
	8a.3 System zahlt Betrag aus, weiter bei 9	
Technologie	Bei Geräten mit Irissensor ist alternativ zur PIN-Eingabe auch ein Iris-Scan möglich	

Mindestens ein Test pro Anforderung

Systematik

Test

Auslöser	Kunde steckt Karte in den Leser	
Beschreibung	1. Kunde steckt Karte in den Leser	
	[R01] Phishing verhindern	
	2. System fragt nach PIN	
	3. Kunde gibt PIN ein	
	[R02] PINs haben genau vier Stellen	
	[R03] Alle vierstelligen Zahlen (0..9) sind als PIN erlaubt	
	4. System fragt nach gewünschter Aktion	
	[R04] Es gibt zumindest "Abbruch" und "Abhebung"	

Requirements präzisieren

Testidee spezifizieren

ID	Kurzbeschreibung	Anforderung	Rationale	Testfall: SetUp	Soll
[R01]	*Phishing verhindern*	*ec-Karte soll mehrfach teilweise eingezogen und ausgegeben werden*	Vorsatzgeräte können dann nicht mehr den Streifen lesen	Karte in Spalt geben	Wird min. zwei Mal in jede Richtung bewegt
[R02]	*PINs haben genau vier Stellen*	*Akzeptiere nur PINs mit exakt vier Stellen*	Internationaler Standard; auch nicht mehr Stellen erlauben	Beim Verändern der PIN vierstellige wählen	wird akzeptiert
				Beim Verändern der PIN fünfstellige wählen	wird zurückgewiesen
[R03]	*Alle vierstelligen Zahlen (0..9) sind als PIN erlaubt*	*Alle vierstelligen Zahlen aus den Ziffern 0 bis 9 sind als PIN erlaubt*	Einige Banken erlauben das, also muss überall diese Eingabe erlaubt sein	PIN Zahl im Mittelfeld wählen (z.B. 4711)	wird akzeptiert
				Als PIN den Grenzwert 0000 wählen	wird akzeptiert

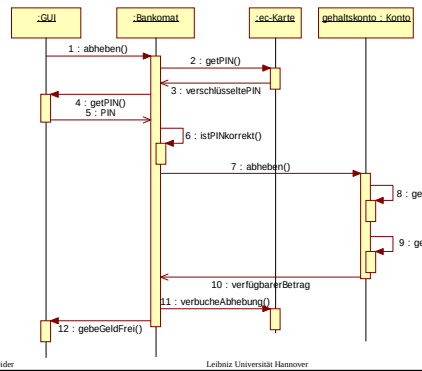
Eigentliche Testfälle müssen vollständig konkret sein:

```
T1: assert(waehlePIN(4711), true)   T3: assert(waehlePIN(50431), false)
T2: assert(waehlePIN(0000), true)
```


Use Cases umsetzen in Sequenzdiagramme

Systematik

Entwurf



Programmstruktur anlegen

Systematik: Architektur, Entwurf, Patterns

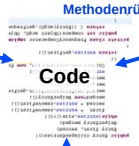
Entwurf | Codie-rung

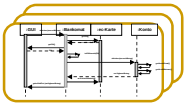
Viele Diagramme für Dynamik

Kommunikationsdiag.

Teile von Methodenrümpfen mehrerer Klassen

Methodenrümpfe





Sequenzdiag. und andere

Klassen, Attribute, Methoden-Schnittstellen (Parameter, Typen)

Bankomat

-istPINkorrekt()
+kontostandAnzeigen()
+abheben()

Ein Klassendiagramm: Basis und Statische Aspekte

Guter Programmcode

Systematik

Codie-rung

Sie wählen im Quellcode:

- Klassennamen
- Methodennamen
- Leerzeichen
- Kommentare
- Einrückungen
- Variablentypen
- Schleifen
- Rückgabewerte
- …

Ziele

- Verständlichkeit
- Lesbarkeit

```
// Collapse list of entries into a code tree. Return its
private Entry collapseEntries(){
    Entry first, second;
    HelperEntry merged;

    if(entries.isEmpty()){
        return null;    // empty list, no tree can be buil
    }

    while(entries.size()>1){
        // Sort list by frequency, lowest frequency first
        Collections.sort(entries, new FrequencyComparator()

        first  = entries.removeFirst();
        second = entries.removeFirst();

        // merge elements and put result back into list
        merged=new HelperEntry();
        merged.setFrequency(first.getFrequency()+ second.g

        // in Huffmann codes, it does not matter who gets
        merged.setOne(first);
        merged.setZero(second);

        // put result back (will be resorted if loop conti
        entries.addFirst(merged);
    };
```


Software-Entwicklung und -Test

Nochmal die ganze Systematik