Specification-based test design

Zoltan Micskei, Istvan Majzik

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group

Main topics of the course

- Overview (1)
  - V&V techniques, Critical systems
- Static techniques (2)
  - Verifying specifications
  - Verifying source code
- **Dynamic techniques: Testing (7)**
  - Developer testing, **Test design techniques**
  - Testing process and levels, Test generation, Automation
- System-level verification (3)
  - Verifying architecture, Dependability analysis
  - Runtime verification

Test design techniques

Goal: Select test cases based on test objectives

Specification-based

- SUT: black box
- Only spec. is known
- Testing specified functionality

Structure-based

- SUT: white box
- Inner structure known
- Testing based on internal behavior

Learning outcomes

- Describe the goal of specification-based test design techniques (K2)
- Use test design techniques equivalence classes, boundary value analysis, decision tables and pair-wise testing to select test cases for simple programs (K3)

EXERCISE Triangle classification program

The program reads the lengths of the sides of a triangle (3 integers). The program writes out whether the triangle is equilateral, isosceles or scalene.

- » Glen Myers, The Art of Software Testing, 1979

Design test cases for this program!

EXERCISE Triangle classification program

- Issues with the specification?
- Solutions:
  - K. Beck (6 tests), R. Binder (65 tests), P. Jorgensen (185 tests)…
- Possible test cases:
  - Equilateral: 3,3,3
  - Isosceles: 5,5,2
    - Similarly for the other sides
  - Scalene: 5,6,7
  - Not a triangle: 1,2,5
    - Similarly for the other sides
  - Just not a triangle: 1,2,3
  - Invalid inputs
    - Zero value: 0,1,1
    - Negative value: -3,-5,-3
    - Not an integer: 2,2,'a'
    - Less inputs than needed: 3,4

Specification-based techniques

Equivalence classes

Boundary values

Decision tables

Combinatorial testing

Based on use cases

…

Equivalence class partitioning

- Input and output equivalence classes:
  - Data that are expected to cover the same faults (cover the same part of the program)
  - Goal: Each equivalence class is represented by one test input (selected test data) [induction]

- Highly context-dependent
  - Needs to know the domain and the SUT!
  - Depends on the skills and experience of the tester

Selecting equivalence classes

- Selection uses heuristics
  - Initial: valid and invalid partitions
  - Next: refine partitions
- Typical heuristics:
  - Interval (e.g. 1-1000)
    - < min, min-max, >max
  - Set (e.g. RED, GREEN, BLUE)
    - Valid elements, invalid element
  - Specific format (e.g. first character is @)
    - Condition true, condition false
  - Custom (e.g. February from the months)

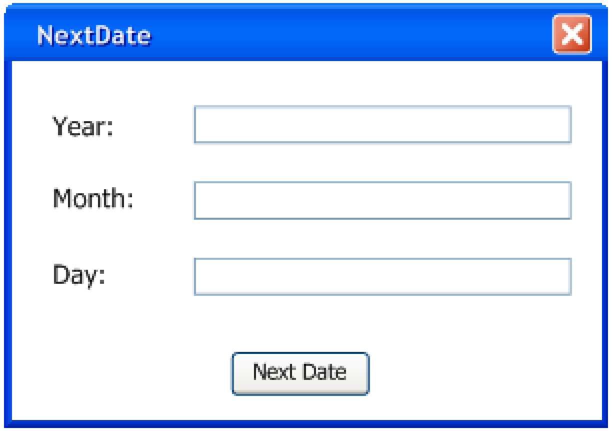
Deriving test cases from equiv. classes

- Combining equiv. classes of several inputs

- For valid (normal) equivalence classes:
  - test data should cover as much equivalence classes as possible

- For invalid equivalence classes:
  - first covering the each invalid equivalence class separately
  - then combining them systematically

EXERCISE NextDate program



- Calculates the next day based on the Gregorian calendar
- What are the equivalence classes for the inputs?
- What are the equivalence classes for the output?

EXERCISE NextDate equivalence classes

Input	Valid	Invalid
Month	V1: 30 day month V2: 31 day month V3: February	I1: >= 13 I2: <= 0 I3: not a number I4: empty
Day	V4: 1-30 V5: 1-31 V6: 1-28 V7: 1-29	I5: >= 32 I6: <= 0 I7: not a number I8: empty
Year	V8: 1582-9999 V9: not leap year V10: leap year V11: centurial year V12: centurial year (div. by 400)	I9: <=1581 I10: >= 9999 I11: not a number I12: empty
Special	V13: 1752.09.03-1752.09.13.	I13: 1582.10.5-1582.10.14.

Source: „How we test software at Microsoft", Microsoft Press, ISBN 0735624259, 2008.

EXERCISE NextDate test cases

A possible combination:

Test	Month	Day	Year	Other	Output
T1	V1 ∪ V2 ∪ V3	V6	V8		Érvényes
T2	V1	V4	V9 ∩ V8		Érvényes
T3	V2	V5	V10 ∩ V8		Érvényes
T4	V3	V6	V11 ∩ V8		Érvényes
T5	V3	V7	V12 ∩ V8		Érvényes
T6				V13	Érvényes
T7	I1				Hiba
T8	I2				Hiba
T9	I3				Hiba
T10	I4				Hiba
T11					Hiba
…					

Choosing valid values randomly

Have all valid classes at least once

One invalid, others valid

Specification-based techniques

Equivalence classes

Boundary values

Decision tables

Combinatorial testing

Based on use cases

…

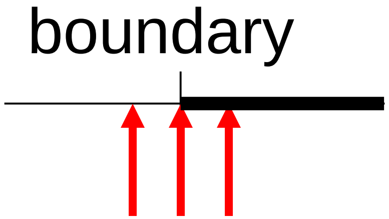
2. Boundary value analysis

- Examining the boundaries of data partitions
  - Focusing on the boundaries of equivalence classes
  - Both input and output partitions

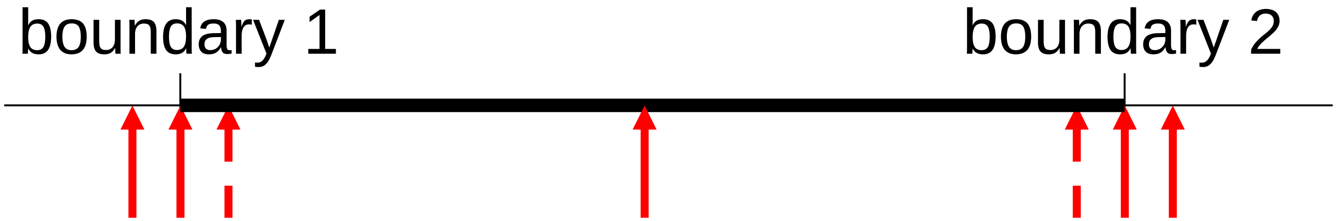
- Typical faults to be detected:
  - Faulty relational operators,
  - conditions in cycles,
  - size of data structures,
  - …

Typical test data for boundaries

- A boundary requires 3 tests:



- An interval requires 5-7 tests:



EXERCISE Boundaries for NextDate

- Month
  - Boundaries: 1, 12
  - Test data: 0, 1, (2), 3-10, (11), 12, 13
- Day
  - Boundaries: 1, 31
  - Test data: 0, 1, (2), 3-29, (30), 31, 32
  - Refinement: 28, 29, 30 can also be a boundary
- Year
  - Boundaries: 1582, 9999
  - Test data: 1581, 1582, (1583), 1584-9997, (9998), 9999, 10000

Specification-based techniques

Equivalence classes

Boundary values

Decision tables

Combinatorial testing

Based on use cases

…

Decision or cause/effect analysis

- Rules for connecting inputs and outputs
  - Business rules: price calculation, insurance, loan…
  - Technical: authentication system
- Connections for
  - Condition/cause: equiv. partitions of input parameters
  - Action/effect: equiv. partitions of output parameters
- Representations:
  - Cause-effect graphs
  - Decision tables

Cause-effect analysis

- Cause-effect graph (Boole graph)
  - Source: equivalence partitions of input parameters
  - Sink: equivalence partitions of output parameters
  - Intermediate: OR, AND, NOT
- Using for test design
  - Covering paths in the graph
  - Truth tables (see Digital design)
  - Originated from HW testing

Decision tables

- Represent each input/output partition with Booleans (conditions/actions)
- Rules will be the test cases
- (Can be represent transposed)

	Rule 1	Rule 2	Rule N
Conditions			
Condition 1	T	T	
Condition 2	F	T	
…			
Actions			
Action 1	X		
Action 2		X	
….			

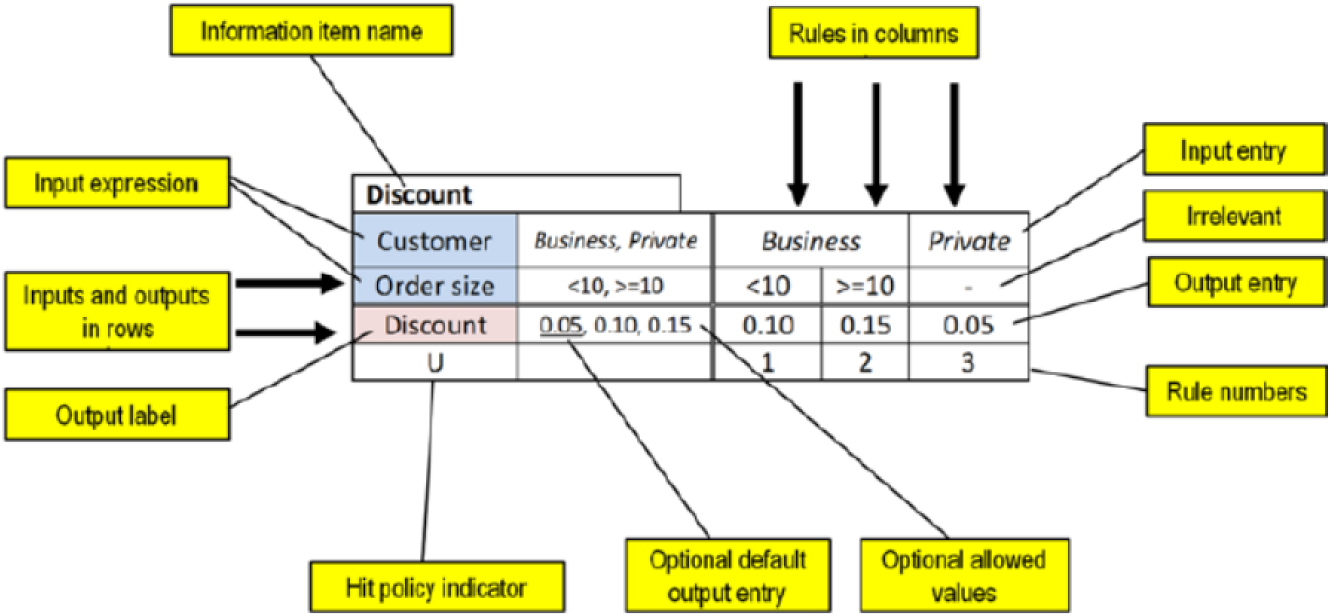
EXERCISE Decision table

The final price of the order is calculated based on discounts. If the user has a membership card (silver 2%, gold 3%), this global discount is always applied. There are also price dependent discounts. If before applying global discounts the total amount to pay is greater than 100 EUR then the discount is 1%, if it is greater than 200 EUR then the discount is 2%.

Create a decision table!

Standardized notation (decision tables)

- OMG’s Decision Model And Notation (DMN)
- Represent decision’ requirements, rules…



Discount

Customer	Business, Private	Business		Private
Order size	<10, >=10	<10	>=10	-
Discount	0.05, 0.10, 0.15	0.10	0.15	0.05
U		1	2	3

Source: OMG

Specification-based techniques

Equivalence classes

Boundary values

Decision tables

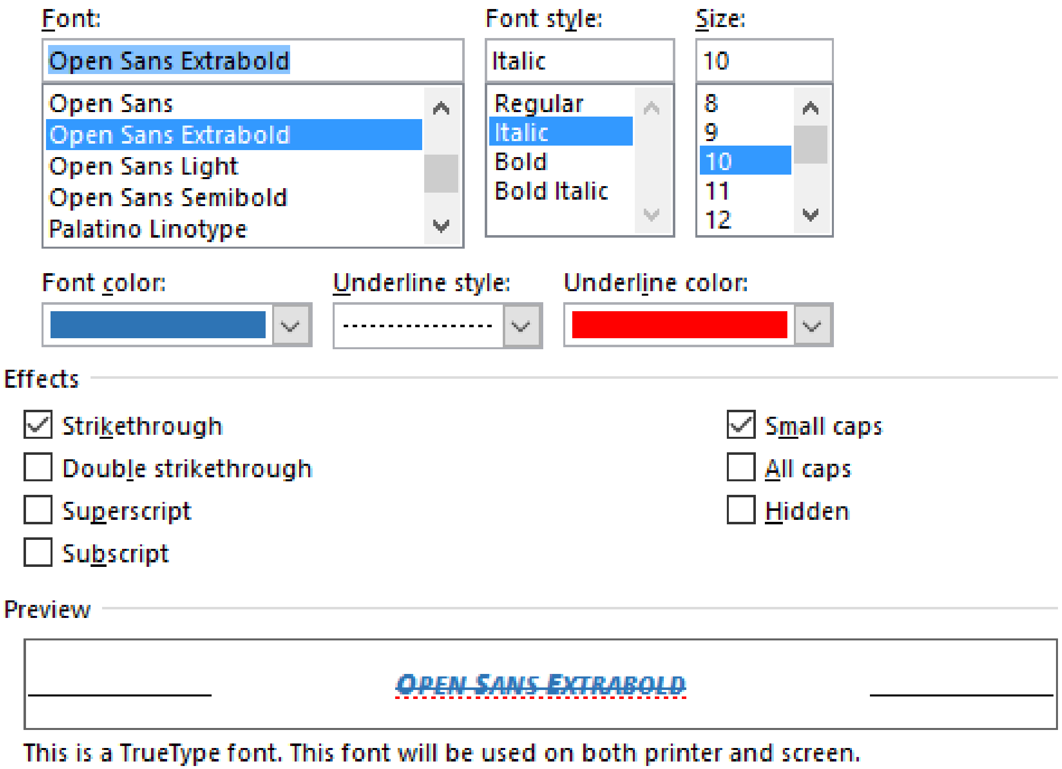
Combinatorial testing

Based on use cases

…

When there are many input parameters

- Failures are caused by (specific) combinations
- Testing all combinations: too much test cases
- Rare combinations may also cause failures



Font: Open Sans Extrabold

Open Sans
Open Sans Extrabold
Open Sans Light
Open Sans Semibold
Palatino Linotype

Font style: Italic

Regular
Italic
Bold
Bold Italic

Size: 10

8
9
10
11
12

Font color: 🟦

Underline style: ------------------

Underline color: 🟥

Effects

☑ Strikethrough
□ Double strikethrough
□ Superscript
□ Subscript

☑ Small caps
□ All caps
□ Hidden

Preview

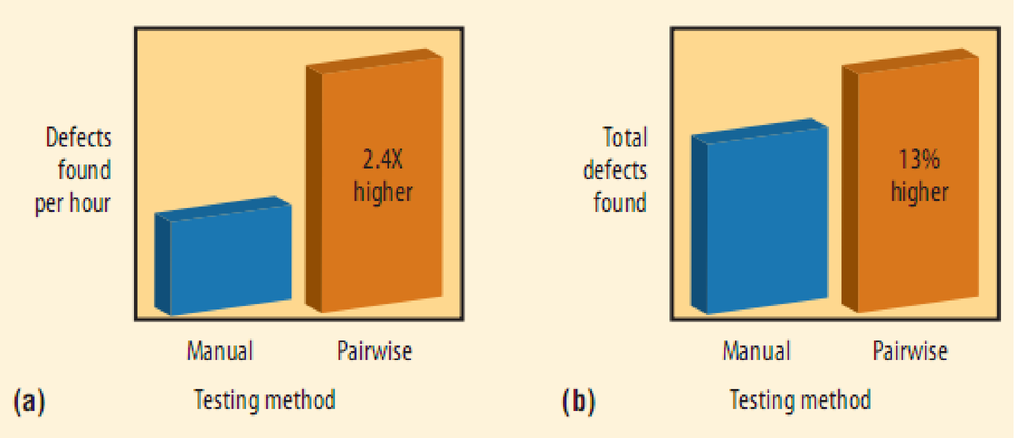
OPEN SANS EXTRABOLD

This is a TrueType font. This font will be used on both printer and screen.

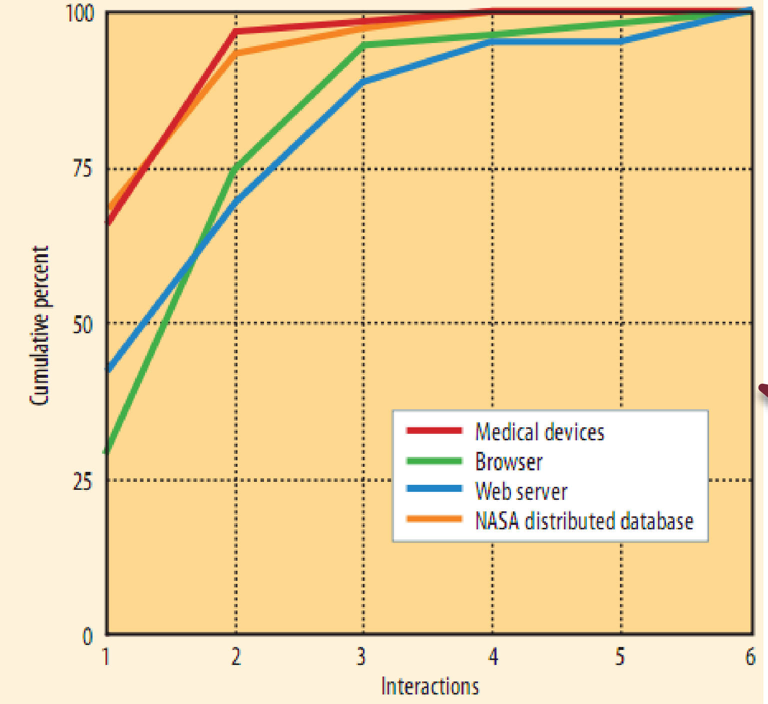
Combinatorial testing techniques

- Ad hoc („best guess”)
  - Intuition, requirements, typical faults…
- Each choice
  - Every choice in at least one test
  - Can miss important combination
- N-wise testing
  - For each arbitrary n parameters, testing all possible combinations of their potential values
  - Special case (n = 2): pairwise testing

Efficiency of n-wise testing



Comparing ad hoc and pairwise testing (10 projects)



Many faults are triggered by specific combinations of at least 2 parameters (or even 3-6)

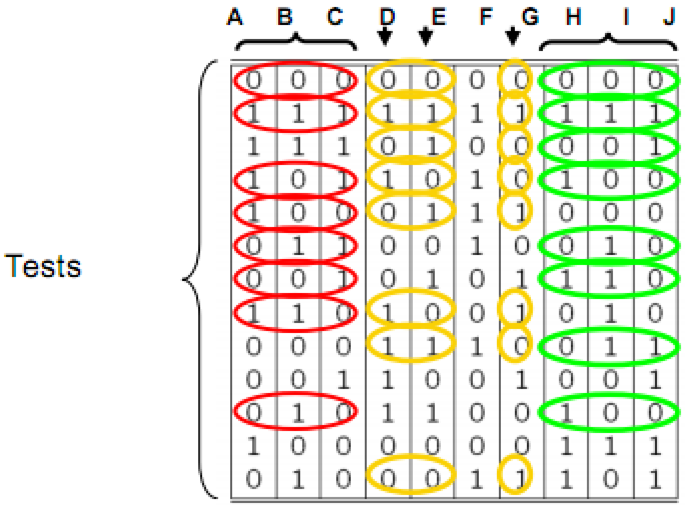
Source: R. Kuhn et al. „Combinatorial Software Testing", IEEE Computer, 42:8, 2009

EXERCISE Pair-wise testing

- Given input parameters and potential values:
  - OS: Windows, Linux
  - CPU: Intel, AMD
  - Protocol: IPv4, IPv6
- How many combinations are possible?
- How many test cases are needed for pairwise testing?
- A potential test suite:
  - T1: Windows, Intel, IPv4
  - T2: Windows, AMD, IPv6
  - T3: Linux, Intel, IPv6
  - T4: Linux, AMD, IPv4

N-wise testing: theory and practice

- Theory: constructing a coverage array



A	B	C	D	E	F	G	H	I	J
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	0	0	0	0	1
1	0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	1	0	0	0
0	1	1	0	0	1	0	0	1	0
0	0	1	0	1	0	1	1	1	0
1	1	0	1	0	0	1	0	1	0
0	0	0	1	1	1	0	0	1	1
0	0	1	1	0	0	1	0	0	1
0	1	0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	1	1	1
0	1	0	0	0	1	1	1	0	1

Figure 3. 3-way covering array

Source: D. R. Kuhn, R. N. Kacker, Y. Lei
Practical Combinatorial Testing
NIST Special Publication 800-142

- Tools (see http://www.pairwise.org)
  - PICT: Pairwise Independent Combinatorial Testing (MS)
  - ACTS - Advanced Combinatorial Testing Suite (NIST)

Specification-based techniques

Equivalence classes

Boundary values

Decision tables

Combinatorial testing

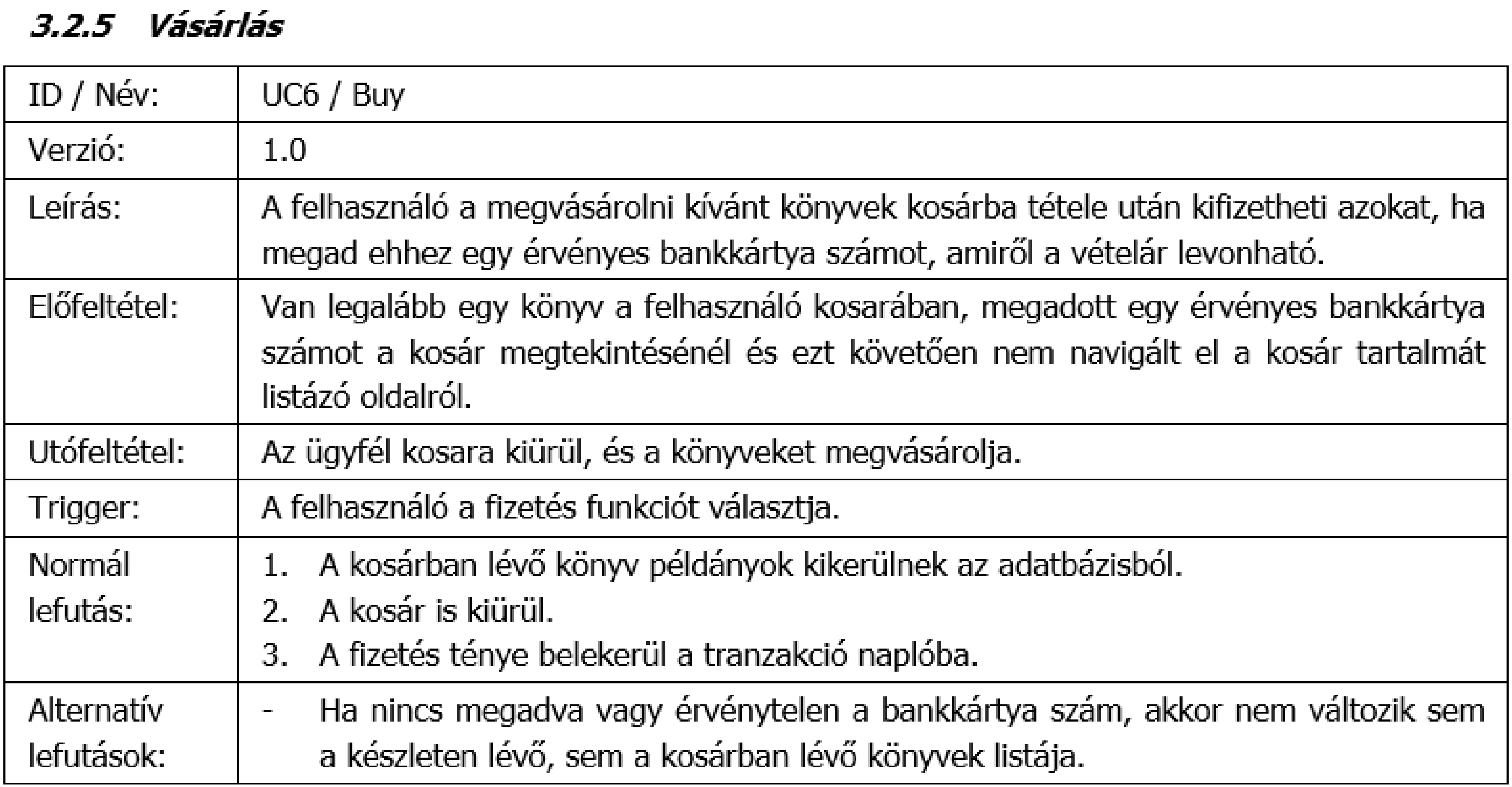
Based on use cases

…

Deriving tests from use cases

- Typical test cases:
  - 1 test for main path („happy path”, „mainstream”)
    - Oracle: checking post-conditions
  - Separate tests for each alternate path
  - Tests for violating pre-conditions
- Mainly higher levels (system, acceptance…)

EXERCISE Deriving tests from a use case



3.2.5 Vásárlás

ID / Név:	UC6 / Buy
Verzió:	1.0
Leírás:	A felhasználó a megvásárolni kívánt könyvek kosárba tétele után kifizetheti azokat, ha megad ehhez egy érvényes bankkártya számot, amiről a vételár levonható.
Előfeltétel:	Van legalább egy könyv a felhasználó kosarában, megadott egy érvényes bankkártya számot a kosár megtekintésénél és ezt követően nem navigált el a kosár tartalmát listázó oldalról.
Utófeltétel:	Az ügyfél kosara kiürül, és a könyveket megvásárolja.
Trigger:	A felhasználó a fizetés funkciót választja.
Normál lefutás:	1. A kosárban lévő könyv példányok kikerülnek az adatbázisból. 2. A kosár is kiürül. 3. A fizetés ténye belekerül a tranzakció naplóba.
Alternatív lefutások:	- Ha nincs megadva vagy érvénytelen a bankkártya szám, akkor nem változik sem a készleten lévő, sem a kosárban lévő könyvek listája.

SUMMARY

Test design techniques

- Specification and structure based techniques
  - Many orthogonal techniques
  - Every techniques need practice!
- Only basic techniques are used commonly ☹
  - Exception: safety-critical systems
    (e.g. DO178-B requires MC/DC coverage analysis)
- Combination of techniques is useful:
  - Example (Microsoft report):

    specification based: 83% code coverage

    + exploratory: 86% code coverage

    + structural: 91% code coverage