Software Engineering

WS 2022/23, Sheet 10

Prof. Dr. Sven Apel
Yannick Lehmen, Maurice Vincon

Handout: 16.01.2022

Task 1

a) What are non–functional properties (NFPs) of software systems?

b) Explain the difference between quantitative and qualitative NFPs. Name three examples for each type.

c) Give an example of the practical relevance of NFPs and explain why this is relevant.

Solution

a) Any property of a product that is not related with functionality represents a NFP. They aren't concerned with *what* a program does, but instead with *how* a program does its work. Examples for NFPs are a program's performance or resource consumptions, but also usability or security.

b)
- *Quantitative*: All NFPs that are easy to measure and evaluate (performance, energy consumption, memory consumption, ...)
- *Qualitative*: All NFPs that are concerned with quality properties of a program (error freeness, security, robustness, ...). These properties are usually hard to measure and evaluate.

c) One domain where NFPs are relevant is embedded software for microcontrollers. The software that is executed on these controllers needs to be very memory and energy efficient, since there are not a lot of resources available on the microcontroller. When configuring a software system for that domain, the resource requirements are relevant for finding the optimal configuration.
As another example, NFPs can be used to complete partial configurations. Consider a user that is not an expert in the system and only selects a couple of features as a requirement. A configuration tool could select the remaining configuration options to optimize the resulting configuration towards one or more NFPs, e.g., to find a good tradeoff between performance and energy consumption.

Task 2

You are the NFP expert in a software company. The customers have reported a strange behavior when comparing the runtime and the energy consumption of certain variants of your software. You have been appointed to investigate this. The measurements, runtime, and energy models give you the following data (Consider the influence of features that do not appear in the model as 0):

Runtime: $263 \cdot A + 69 \cdot B + 179 \cdot B \cdot C - 13 \cdot D - 31.5 \cdot B \cdot C \cdot E$
Energy Consumption: $7 \cdot A + 5 \cdot B + 5 \cdot B \cdot C + 0.5 \cdot C - 0.25 \cdot F$

Do you find discrepancies when comparing the two models? Explain your reasoning.

Solution

To find discrepancies between the models, we assume that the energy consumption correlates with the runtime, meaning if a variant runs longer, it also requires more energy. Therefore, we can find discrepancies by finding factors in the model that violate this assumption.

	A	B	C	D	F	$B \cdot C$	$B \cdot C \cdot E$
Runtime (s)	263	69	0.0	-13	0.00	179	-31.5
Energy Consumption (kJ)	7	5	0.5	0	-0.25	5	0.0

Feature B: When comparing the influence of feature B on the runtime model and the energy model, we see that the small increase in runtime leads to an energy consumption that is as high as for the feature interaction {B, C}, which has a way higher runtime increase. More detailed: the feature interaction {B, C} adds 179s to the runtime while only increasing the energy consumption by 5kJ. Feature B has the same energy consumption but only adds 69s to the runtime.

Feature C: This feature has an influence on the energy consumption but not on the runtime. This could be because feature C causes some non–efficient (energy) tasks.

Feature D: This feature decreases the execution time but not the energy consumption. Meaning the program runs faster but uses the same amount of energy. This could mean that using feature D could make the program more time efficient while using the same energy resources.

Feature F: see feature C.

$\{B, C, E\}$: The interaction of the features B, C, and E seems to decrease the runtime while using the same amount of energy resources. The reasoning behind it is probably the same as for Feature D.

Task 3

Given the runtime measurements of a software with three features in Table 1, determine which feature interactions influence the runtime.

Table 1: Performance data for software system ABC

Configuration	Features			Runtime (s)
c_i	A	B	C	y_i
c_1	0	0	0	20
c_2	0	0	1	25
c_3	0	1	0	22
c_4	0	1	1	27
c_5	1	0	0	27
c_6	1	0	1	40
c_7	1	1	0	26
c_8	1	1	1	39

Solution

First step: Find the influences of the three features on the runtime:

- Base runtime: 20s (according to c_1)
- Influence of feature A: $y_5 - y_1 = 27s - 20s = 7s$
- Influence of feature B: $y_3 - y_1 = 22s - 20s = 2s$
- Influence of feature C: $y_2 - y_1 = 25s - 20s = 5s$

Second step: Compare the predicted values for the configurations that contain feature interactions with the measured values to determine an influence on the runtime caused by the interaction:

- c_4: Prediction: $20s + 2s + 5s = 27s$, which is exactly the measured time.
- c_6: Prediction: $20s + 7s + 5s = 32s$, which is smaller than the measured $40s$, which means that the interaction of features A and C slows the program down.
- c_7: Prediction: $20s + 7s + 2s = 29s$, which is larger than the measured $26s$, which means that the interaction of Feature A and B speeds up the program.
- c_8: Prediction: $20s+7s+2s+5s = 34s$, which is smaller than the measured $39s$, which means that the interaction of Features A, B, and C slows the program down. But when looking at the interactions {A, C} and {A, B} and factoring these in, then we have to add the $34s + 8s - 3s = 39s$, which is exactly the measured time.

This means that there are interactions between the features {A, C} and {A, B}. The complete performance influence model extracted from these measurements is

$$20 + 7 \cdot A + 2 \cdot B + 5 \cdot C - 3 \cdot A \cdot B + 8 \cdot A \cdot C$$

Task 4

Create the CART of the data in Table 2. Do so until you reach a depth of 3. This means that you have three layers under the root of the tree. Use the formulas from the lecture and provide the sample mean and presence condition of every branch.

Table 2: Performance data for software system 12345

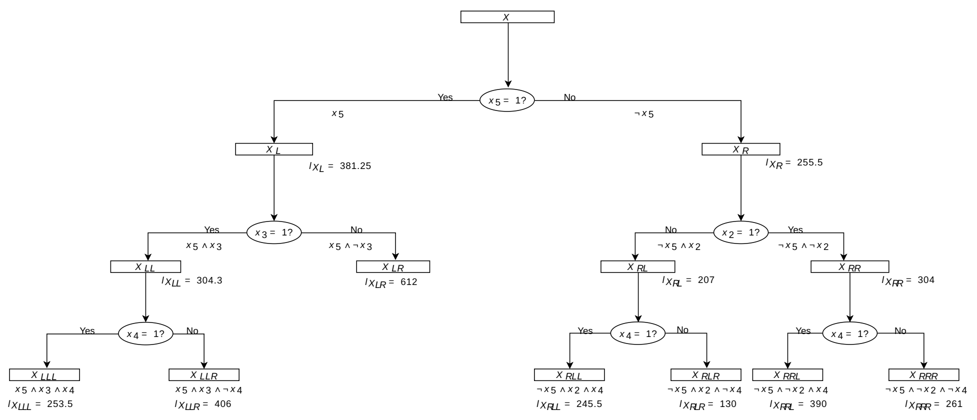
Configuration	Features					Runtime (s)
$\boldsymbol{c_i}$	x_1	x_2	x_3	x_4	x_5	y_i
$\boldsymbol{c_1}$	1	0	1	1	1	207
$\boldsymbol{c_2}$	1	0	1	0	0	342
$\boldsymbol{c_3}$	1	1	0	1	0	222
$\boldsymbol{c_4}$	1	1	1	1	1	300
$\boldsymbol{c_5}$	1	0	0	0	0	180
$\boldsymbol{c_6}$	1	1	0	0	1	612
$\boldsymbol{c_7}$	1	1	1	1	0	269
$\boldsymbol{c_8}$	1	0	1	0	1	406
$\boldsymbol{c_9}$	1	1	1	0	0	130
$\boldsymbol{c_{10}}$	1	0	0	1	0	390

Solution

Example calculation for the first split:

- x_1 can never be the split since there is no configuration where it is turned off.
- x_2:
  - $l_{x_{2L}} = (\frac{1}{5} \times (222 + 300 + 612 + 269 + 130)) = 306.6$
  - $l_{x_{2R}} = (\frac{1}{5} \times (207 + 342 + 180 + 406 + 390)) = 305$
  - sq-err-loss(l) $= (222 - 306.6)^2 + (300 - 306.6)^2 + (612 - 306.6)^2 + (269 - 306.6)^2 + (130 - 306.6)^2 = 133,071.2$
  - sq-err-loss(r) $= (207 - 305)^2 + (342 - 305)^2 + (180 - 305)^2 + (406 - 305)^2 + (390 - 305)^2 = 44,024$
  - sum $= 133,071.2 + 44,024 = 177,095.2$
- x_3: sum $= 163,481.3$
- x_4: sum $= 169,149.2$
- x_5: sum $= 139,150.2$ ✓

Figure 1: The complete CART


X
x5 = 1?
Yes
No
x5
¬x5
X L
lXL = 381.25
X R
lXR = 255.5
x3 = 1?
Yes
No
x5 ∧ x3
x5 ∧ ¬x3
X LL
lXLL = 304.3
X LR
lXLR = 612
x2 = 1?
No
Yes
¬x5 ∧ x2
¬x5 ∧ ¬x2
X RL
lXRL = 207
X RR
lXRR = 304
x4 = 1?
Yes
No
X LLL
x5 ∧ x3 ∧ x4
lXLLL = 253.5
X LLR
x5 ∧ x3 ∧ ¬x4
lXLLR = 406
x4 = 1?
Yes
No
X RLL
¬x5 ∧ x2 ∧ x4
lXRLL = 245.5
X RLR
¬x5 ∧ x2 ∧ ¬x4
lXRLR = 130
x4 = 1?
Yes
No
X RRL
¬x5 ∧ ¬x2 ∧ x4
lXRRL = 390
X RRR
¬x5 ∧ ¬x2 ∧ ¬x4
lXRRR = 261