Aufgabe 9

Nathan Ritter
5566519
Lorenzo Tecchia
5581906
2023.06.18

Contents

Chapter 1

Task 1

(a) In computer science, information hiding is the principle of segregation of the design decisions in a computer program that are most likely to change, thus protecting other parts of the program from extensive modification if the design decision is changed. Whereas the "need to know" principle states that: depending on the level of clearance that stakeholders have on the system, they access a certain amount of knowledge about the system itself. So for some level of knowledge about the system the information hiding principle get implemented for some users/stakeholders. [3]

(b) The information hiding principle is useful when the design of a piece of software gets broken down into smaller parts, modules in this case. And different teams work on different modules of the same software. In this case the different teams do not need to know how other modules internally work, they just need to know how they can fit their module features into the whole system and/or with the other modules. A clear example of the information hiding principle would be how developer nowadays work with APIs and how they don't need to know or in most cases, they are not allowed to know how the systems behind those APIs work. It's like working with a black box. [3]

(c) Design by contract prescribes that software designers should define formal, precise and verifiable interface specifications for software components, which extend the ordinary definition of abstract data types with preconditions, post-conditions and invariants. These specifications are referred to as "contracts", in accordance with a conceptual metaphor with the conditions and obligations of business contracts.

(d) Invariants, postconditions, preconditions, are all constrains for the DbC principle. They are actuated by the OCL to improve the use of the Unified Modelling Language. So they act as an enhancement for UML in order to deliver a better understanding of the system that a UML diagram is depicting, usually a Class Diagram, because postconditions and preconditions are constrains for methods of classes and invariants are constraints for classes. [2][1]

Chapter 2

Task 2

(a) The score for the task must be greater than 1. The size for the solution for a student must be equal to the size of the task for an exam. If a student has passed an exam means that exists a solution for that exam with more than 0 points.

(b) Solution:
1.

```
context exam inv c4: exam.task -> size() = 1
```

2.

```
context exam inv c5: not exist exam.post_exam.
    post_exam
```

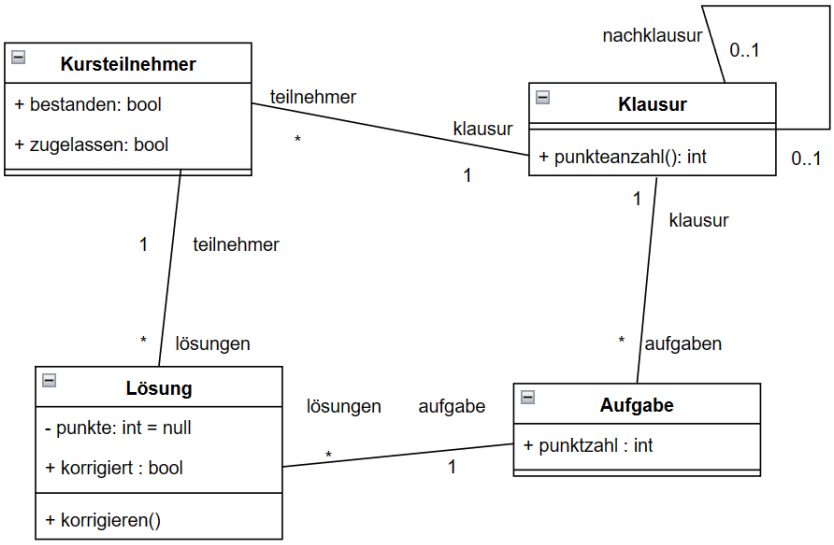
3.

```
c6:
    student->forall(s| s.passed implies s.solution->
        exists forall(s.tasks.solutions))
```

Chapter 3

Task 3

(a) korrigieren() should look at the solution for each question and give it points depending on how correct/incorrect it is.



(b) constraints for korrigieren with only one run per instance

```
context Loesung::korrigieren() pre: aufgabe != null
context Loesung::korrigieren() pre: teilnehmer != null
context Loesung::korrigieren() pre: not korrigiert
context Loesung::korrigieren() post: korrigiert = true
context Loesung::korrigieren() post: punkte <= aufgabe.
    punktzahl
```

```
6 context Loesung::korrigieren() post: punkte >= 0
```

(c) constraints for korrigieren with multiple runs per instance

```
1 context Loesung::korrigieren() pre: aufgabe != null
2 context Loesung::korrigieren() pre: teilnehmer != null
3 context Loesung::korrigieren() post: korrigiert = true
4 context Loesung::korrigieren() post: punkte <= aufgabe.
    punktzahl
5 context Loesung::korrigieren() post: punkte >= 0
```

(d) constraints for punkteanzahl()

```
1 context Klausur::punkteanzahl post: result = aufgaben->
    collect(punktzahl)->sum()
```

Bibliography

[1] Jordi Cabot and Martin Gogolla. "Object Constraint Language (OCL): A Definitive Guide". In: *Proceedings of the 12th International Conference on Formal Methods for the Design of Computer, Communication, and Software Systems: Formal Methods for Model-Driven Engineering*. SFM'12. Bertinoro, Italy: Springer-Verlag, 2012, pp. 58–90. ISBN: 9783642309816. DOI: 10.1007/978-3-642-30982-3_3. URL: https://doi.org/10.1007/978-3-642-30982-3_3.

[2] *Design by contract - Wikipedia — en.wikipedia.org*. [Accessed 16-Jun-2023]. URL: https://en.wikipedia.org/wiki/Design_by_contract.

[3] *Information hiding - Wikipedia — en.wikipedia.org*. [Accessed 16-Jun-2023]. URL: https://en.wikipedia.org/wiki/Information_hiding#References.