UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Software Engineering im Wintersemester 2021/2022
Prof. Dr. Martin Leucker, Malte Schmitz, Stefan Benox, Julian Schulz, Benedikt Stepanek, Friederike Weilbeer, Tom Wetterich

Übungszettel 11 (Lösungsvorschlag)

22.01.2022

Abgabe bis Donnerstag, 27. Januar um 23:59 Uhr online im Moodle.

Aufgabe 11.1: Sequenzdiagramme komplexer Prozesse

4 Punkte, mittel

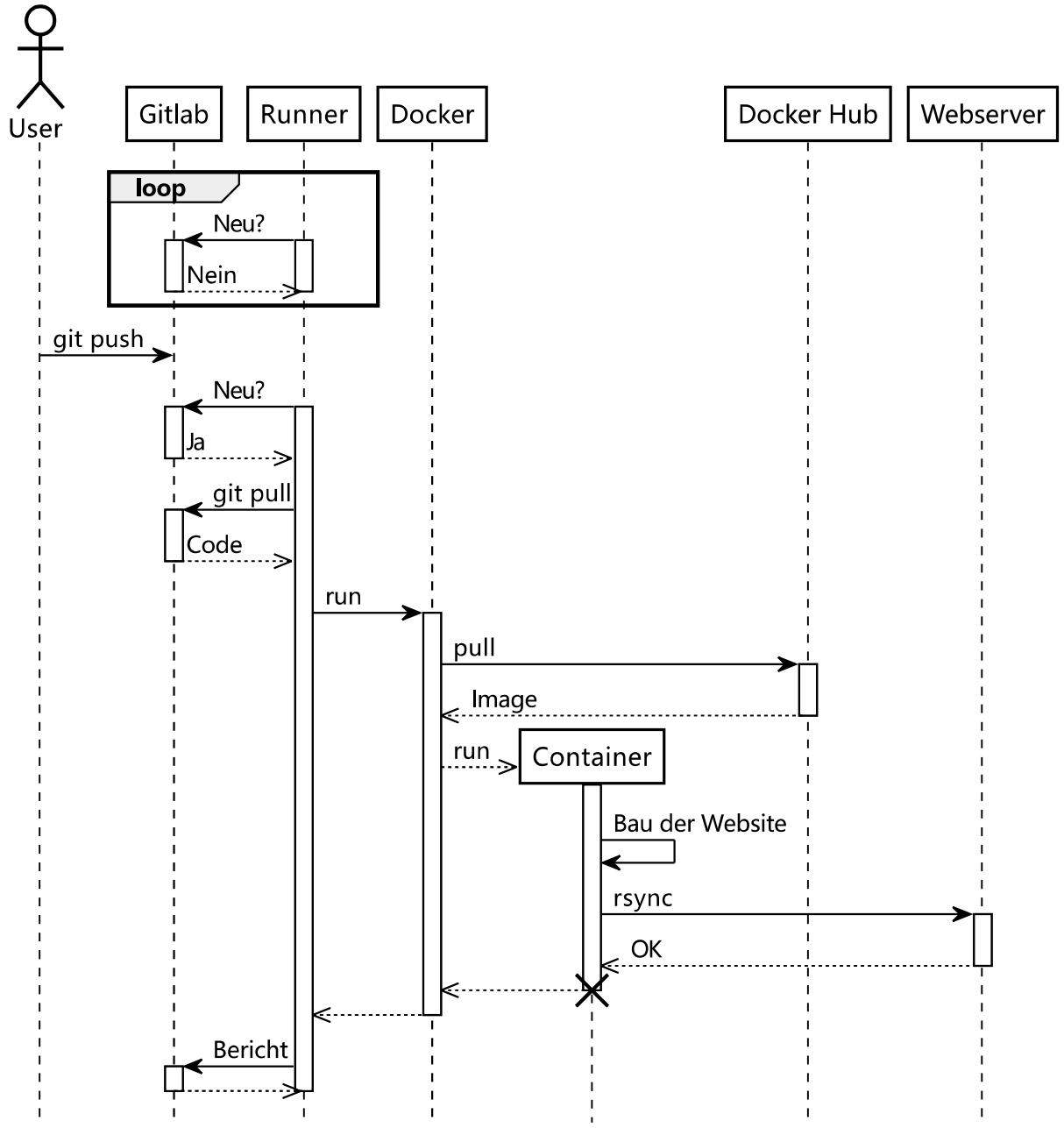
Ein *Static Site Generator (SSG)* erzeugt eine Website aus Konfigurationsdateien, wie zum Beispiel Markdown-Dateien. Recherchieren Sie, wie ein SSG von einem *Shared Gitlab Runner* automatisch in einem *Docker Container* ausgeführt und die gebaute Website anschließend mit rsync auf einen Webserver deployed werden kann.

Stellen Sie den Ablauf in einem Sequenzdiagram dar. Verwenden Sie dabei folgende Akteure:

- Der **User**, der Änderungen an der Konfiguration der Website ins Gitlab pushed.
- Das **Gitlab**, in dem sich das Repostory mit den Konfigurationsdaten der Website befindet.
- Der **Gitlab Runner**, der regelmäßig das Gitlab polled, um Änderungen im Gitlab zu bemerken und sich in diesem Fall den aktuellen Stand aus dem Gitlab pulled, um damit den Build-Prozess im Docker zu starten und am Ende den Build-Bericht wieder an das Gitlab schickt.
- Das **Docker** auf dem Gitlab Runner, das vom Docker Hub eine Docker-Image herunterlädt, aus diesem Image einen neuen Container erzeugt und in diesem den Build-Prozess laufen lässt.

- Der **Container**, in dem die Website gebaut und anschließend die gebauten Dateien per rsync auf den Webserver synchronisiert wird.
- Der **Docker Hub**, der das Docker-Image bereitstellt.
- Der **Webserver**, der die gebaute Website hostet.

▼ Lösungsvorschlag



▶ Quelltext des Diagramms

Aufgabe 11.2: Analyse einer algebraischen Spezifikation

5 Punkte, schwer

Betrachten Sie folgende algebraische Spezifikation. Dabei seien `Nat` und `Bool` die aus der Vorlesung bekannten Spezifikationen.

```
spec Poly = Nat then
sorts
    poly = empty | make(poly,nat)
ops
    terms: (poly) nat,
    addPoly: (poly, poly) poly,
    evaluate: (poly, nat) nat
vars
    p, r: poly,
    a, b, n: nat
axioms
    terms(empty) = zero
    terms(make(p, zero)) = terms(p)
    terms(make(p, succ(a))) = succ(terms(p))

    addPoly(empty, p) = addPoly(p, empty) = p
    addPoly(make(p, a), make(r, b)) = make(addPoly(p, r), add(a, b))

    evaluate(empty, n) = zero
    evaluate(make(p, a), n) = add(mult(evaluate(p, n), n), a)
end
```

1. Beschreiben Sie die oben spezifizierte Datenstruktur in eigenen Worten. Worum handelt es sich bei der Datenstruktur? Zeigen Sie an einem Beispiel auf, wie bestimmte Objekte der Datenstruktur erzeugt werden können. (1 Punkt)

   ▼ **Lösungsvorschlag**

   Es wird eine Datenstruktur für Polynome spezifiziert. Das leere Polynom `empty` ist $e(x) = 0$ und `make` bildet aus einem Polynom p und einer natürlichen Zahl n ein Polynom höherer Ordnung $q(x) = p(x) \cdot x + n$.

   So kann zum Beispiel das Polynom $f(x) = 3x^3 + 2x + 5$ durch `make(make(make(make(empty, 3), 0), 2), 5)` und das Polynom $g(x) = 3x$ durch `make(make(empty, 3), 0)` erzeugt werden.

   Die Operation `terms` gibt die Anzahl der Terme eines Polynoms zurück. Für obiges Beispiel gilt `terms(f) = 3` und `terms(g) = 1`.

   Die Operation `addPoly` addiert zwei Polynome, so gilt zum Beispiel `addPoly(f, g) = make(make(make(make(empty, 3), 0), 5), 5)`, denn es gilt $f(x) + g(x) = 3x^3 + 5x + 5$.

Die Operation `evaluate` wertet ein Polynom an einer gegebenen Stelle aus. So gilt zum Beispiel `evaluate(g, 1) = succ(succ(succ(zero)))`, denn $g(1) = 3$.

2. Erläutern Sie das Axiom `evaluate(make(p, a), n) = add(mult(evaluate(p, n), n), a)`. (1 Punkt)

▼ **Lösungsvorschlag**

`make(p, a)` beschreibt das Polynom $f(x) = p(x) \cdot x + a$, entsprechend gilt $f(n) = p(n) \cdot n + a$.

3. Entwickeln Sie ein Modell $\mathcal{M}$, welches das Erzeugungsprinzip erfüllt. Geben Sie auch die Trägermenge an. Nehmen Sie dabei an, dass $\mathcal{N}$ ein geeignetes Modell für `Nat` ist. (1 Punkt)

   Verwenden Sie für `poly` die Trägermenge $\mathbb{N}^*$, also die Menge aller endlichen Sequenzen über der Menge der natürlichen Zahlen.

   Für Sequenzen nutzen wir folgende Notation: Eine Sequenz

   $$w = \langle w_0, w_1, \ldots, w_{n-1} \rangle \in \mathbb{N}^*$$

   der Länge $|w| = n$ besteht aus n Zahlen. Die leere Sequenz ist entsprechend $\langle\rangle \in \mathbb{N}^*$. Sei

   $$v = \langle v_0, v_1, \ldots, v_{m-1} \rangle \in \mathbb{N}^*$$

   eine weitere Sequenz der Länge $|v| = m$. Dann ist die Konkatenation

   $$v \mathbin{\&} w = \langle v_0, v_1, \ldots, v_{m-1}, w_0, w_1, \ldots, w_{n-1} \rangle$$

   eine weitere Sequenz $v \mathbin{\&} w \in \mathbb{N}^*$ der Länge $|v \mathbin{\&} w| = n + m$.

▼ **Lösungsvorschlag**

Ein Modell für eine algebraische Spezifikation besteht aus einer Trägermenge für jede Sorte, eine Übersetzung der Konstanten und Konstruktoren auf Elemente der Trägermenge und eine Implementierung der Operationen durch mathematische Funktionen. Wir verwenden die Trägermenge $\mathbb{N}^*$ für die Sorte `poly`.

Das Modell $\mathcal{M}$ besteht aus der Konstanten $\mathbf{empty}^{\mathcal{M}} \in \mathbb{N}^*$, dem Konstruktor $\mathbf{make}^{\mathcal{M}}: \mathbb{N}^* \times \mathbb{N} \to \mathbb{N}^*$ und folgenden Operationen:

$$\begin{aligned}\mathbf{terms}^{\mathcal{M}}&: \mathbb{N}^* \to \mathbb{N}\\ \mathbf{addPoly}^{\mathcal{M}}&: \mathbb{N}^* \times \mathbb{N}^* \to \mathbb{N}^*\\ \mathbf{evaluate}^{\mathcal{M}}&: \mathbb{N}^* \times \mathbb{N} \to \mathbb{N}\end{aligned}$$

Die Konstante $\mathbf{empty}^{\mathcal{M}} \in \mathbb{N}^*$ definieren wir als die leere Sequenz:

$$\mathbf{empty}^{\mathcal{M}} = \langle\rangle$$

Die Konstruktorfunktion $\mathbf{make}^{\mathcal{M}}: \mathbb{N}^* \times \mathbb{N} \to \mathbb{N}^*$ sei gegeben durch

$$\mathbf{make}^{\mathcal{M}}(p, x) = p \mathbin{\&} \langle x \rangle$$

Die Funktionen der Operatoren seien schließlich gegeben durch

$$\begin{aligned}\mathbf{terms}^{\mathcal{M}}(\langle\rangle) &= \mathbf{zero}^{\mathcal{N}}\\ \mathbf{terms}^{\mathcal{M}}(p \mathbin{\&} \langle x \rangle) &= \begin{cases}\mathbf{add}^{\mathcal{N}}(\mathbf{succ}^{\mathcal{N}}(\mathbf{zero}^{\mathcal{N}}), \mathbf{terms}^{\mathcal{M}}(p)) & \text{fa}\\ \mathbf{terms}^{\mathcal{M}}(p) & \text{so}\end{cases}\\ \mathbf{addPoly}^{\mathcal{M}}(\langle\rangle, p) &= p\\ \mathbf{addPoly}^{\mathcal{M}}(p, \langle\rangle) &= p\\ \mathbf{addPoly}^{\mathcal{M}}(p \mathbin{\&} \langle x \rangle, q \mathbin{\&} \langle y \rangle) &= \mathbf{addPoly}^{\mathcal{M}}(p, q) \mathbin{\&} \langle \mathbf{add}^{\mathcal{N}}(x, y) \rangle\\ \mathbf{evaluate}^{\mathcal{M}}(\langle\rangle, n) &= \mathbf{zero}^{\mathcal{N}}\\ \mathbf{evaluate}^{\mathcal{M}}(p \mathbin{\&} \langle x \rangle, n) &= \mathbf{add}^{\mathcal{N}}(x, \mathbf{mult}^{\mathcal{N}}(n, (\mathbf{evaluate}^{\mathcal{M}}(p, n))))\end{aligned}$$

Unter Verwendung des üblichen Modells $\mathcal{N}$ können obige Definitionen vereinfacht werden:

$$\begin{aligned}\mathbf{empty}^{\mathcal{M}} &= \langle\rangle\\ \mathbf{make}^{\mathcal{M}}(p, x) &= p \mathbin{\&} \langle x \rangle\\ \mathbf{terms}^{\mathcal{M}}(\langle\rangle) &= 0\\ \mathbf{terms}^{\mathcal{M}}(p \mathbin{\&} \langle x \rangle) &= \begin{cases}1 + \mathbf{terms}^{\mathcal{M}}(p) & \text{falls } x \neq 0\\ \mathbf{terms}^{\mathcal{M}}(p) & \text{sonst}\end{cases}\\ \mathbf{addPoly}^{\mathcal{M}}(\langle\rangle, p) &= p\\ \mathbf{addPoly}^{\mathcal{M}}(p, \langle\rangle) &= p\\ \mathbf{addPoly}^{\mathcal{M}}(p \mathbin{\&} \langle x \rangle, q \mathbin{\&} \langle y \rangle) &= \mathbf{addPoly}^{\mathcal{M}}(p, q) \mathbin{\&} \langle x + y \rangle\\ \mathbf{evaluate}^{\mathcal{M}}(\langle\rangle, n) &= 0\\ \mathbf{evaluate}^{\mathcal{M}}(p \mathbin{\&} \langle x \rangle, n) &= x + n \cdot \mathbf{evaluate}^{\mathcal{M}}(p, n)\end{aligned}$$

4. Weisen Sie für Ihr Modell $\mathcal{M}$ das Erzeugungsprinzip nach. (1 Punkt)

▼ Lösungsvorschlag

Alle Polynome $\langle x_1, \ldots, x_n \rangle$ können erzeugt werden durch

$$\mathbf{make}^{\mathcal{M}}(\ldots \mathbf{make}^{\mathcal{M}}(\mathbf{empty}^{\mathcal{M}}, x_1), \ldots x_n),$$

wobei $x_1, \ldots, x_n$ aus $\mathcal{N}$ sind.

5. Zeigen Sie, dass das Axiom

```
addPoly(make(p, a), make(r, b)) = make(addPoly(p, r), add(a, b))
```

von Ihrem Modell $\mathcal{M}$ erfüllt wird. Nehmen Sie dabei an, dass $\mathcal{N}$ das übliche Modell für Nat ist. (1 Punkt)

▼ Lösungsvorschlag

$$\begin{aligned}
&\mathbf{addPoly}^{\mathcal{M}}(\mathbf{make}^{\mathcal{M}}(p, a), \mathbf{make}^{\mathcal{M}}(r, b)) \\
&= \mathbf{addPoly}^{\mathcal{M}}(p \,\&\langle a \rangle, r \,\&\langle b \rangle) \\
&= \mathbf{addPoly}^{\mathcal{M}}(p, r) \,\&\langle \mathbf{add}^{\mathcal{N}}(a, b) \rangle \\
&= \mathbf{make}^{\mathcal{M}}(\mathbf{addPoly}^{\mathcal{M}}(p, r), \mathbf{add}^{\mathcal{N}}(a, b))
\end{aligned}$$

Aufgabe 11.3: Vier-Ohren-Modell

3 Punkte, leicht

1. Beschreiben Sie in eigenen Worten die Theorie des Vier-Ohren-Modells. (2 Punkte)

▼ Lösungsvorschlag

Das Vier-Ohren-Modell besagt, dass eine Nachricht auf vier verschiedene Weisen verstanden werden kann. Nach dem Modell besitzt jede Nachricht zwischen einem Sender und einem Empfänger eine Botschaft auf der

Sachebene, aber auch eine *Apellseite*, *Beziehungsseite*, *Selbstkundgabe-Seite*, bei der der Sender Informationen über den Wunsch gegenüber dem Empfänger, der empfundenen Beziehung zwischen Sender und Empfänger und dem eigenen Empfinden Preis gibt.

2. Warum kann die Kenntnis des Modells im Softwareentwicklungsprozess von Vorteil sein? (1 Punkt)

▼ **Lösungsvorschlag**

Sich über diese vier Seiten bewusst zu sein, kann die Kommunikation im Team sowohl für Sender als auch Empfänger angenehmer, verständnisvoller und effektiver machen. Eine gute Kommunikation ist ein Kern-Element guter Teamarbeit, wie sie in allen Bereichen des Software-Entwicklungsprozesses nötig ist.