Editorial

Liebe Leserinnen und liebe Leser,

Herzlich willkommen im neuen Jahr.

Mit großer Bestürzung habe ich vom unerwarteten Tod des Kollegen Riebisch gehört, mit dem ich in seinem Arbeitskreis „Traceability“ zusammengearbeitet und auch publiziert habe. Diese Ausgabe enthält einen Nachruf auf ihn.

Sie finden in dieser Ausgabe die Proceedings des Fachgruppentreffens Requirements Engineering und die des 13ten Symposium on Software Performance (SSP). Video-Aufnahmen der Vorträge des Fachgruppentreffens Requirements Engineering können Sie online hier ansehen:

https://av.tib.eu/series/1376/diskussionsaustausch+zum+thema+menschenzentriertes+requirements+engineering+der+fachgruppe+requirements+engineering+der+geschellschaft+fur+informatik

Ich wurde darauf aufmerksam gemacht, dass man meine „Informatik-Artikel in Frauensprache“ als sexistisch wahrnehmen kann. Als würde ich unterstellen, dass Frauen die Informatik in anderer Form nicht verstehen. Eigentlich hätte ich eher erwartet, dass Männer sich darüber beklagen, dass sie die „Frauensprache“ nicht verstehen oder dass ich ihnen das unterstellt habe. Ein solcher Scherz ist natürlich auch sexistisch. Ich entschuldige mich hiermit bei allen Geschlechtern. Natürlich gibt es auch Männer, die kochen und stricken können. Und die Leserinnen dieser Zeitschrift könnten sicher jeden der Begriffe selbst definieren. Spontan lag mir der Begriff „Parodie“ auf den Tasten, aber es ist keine Parodie, dass die Lebenswelten von Männern und Frauen, ihre Fähigkeiten und Vorlieben sich immer noch unterscheiden, zumindest im statistischen Mittel. Darum möchte ich gerne damit fortfahren, einige Softwaretechnik-Begriffe anhand von Beispielen „aus der Frauenwelt“ zu definieren. Ich sehe es als Experiment und hoffe, dass ich Sie damit anrege, bei der Weitergabe von Informatik-Wissen Ihre Beispiele auch mal abseits der ausgetretenen Pfade der Militär-Auto-Sport-Sprache zu pflücken. Mir macht das Experiment jedenfalls Freude, und ich grüble immer noch über den Unterschied zwischen dem Großvaterproblem und dem Großmutterproblem (siehe nächste Seite).

Andrea Herrmann

Acyclic Dependencies Principle ADP

Jedes System besteht laut Definition aus Komponenten, die voneinander abhängen. Das gilt für Software-Systeme genauso wie für Ihren Haushalt oder Ihre Familie oder jedes andere System. Wenn Sie diese Abhängigkeiten grafisch darstellen, dann entsteht ein sogenannter Abhängigkeitsgraph, in dem die Komponenten als Knoten dargestellt sind, die Abhängigkeiten als Kanten. Diese Kanten sind oft gerichtet, denn nur weil A von B abhängt, muss nicht auch A von B abhängen. Im Gegenteil gibt es oft eine natürliche Reihenfolge: Die Großmutter gebiert die Mutter, und die Mutter gebiert die Tochter und diese die Enkelin. In welcher Reihenfolge man die Komponenten eines Zimmers putzt, ist dagegen nicht unbedingt naturgegeben, aber wir suchen gerne eine sinnvolle Reihenfolge. Hier bestehen Abhängigkeiten wie „Bevor ich den Boden fege, staube ich die Regale ab, weil dabei Staub auf den Boden fällt“ und „Bevor ich die Regale abstaube, wische ich oben auf dem Schrank, weil dabei Staub auf die Regale fällt“. Also: In welcher Reihenfolge putzen Sie? Mit Hilfe eines azyklischen Abhängigkeitsgraphen (dependency graph) wie in Abbildung 1 fällt die Planung leicht: Zuerst putzen wir die Schrankoberseite, dann die Regale und dann den Fußboden.

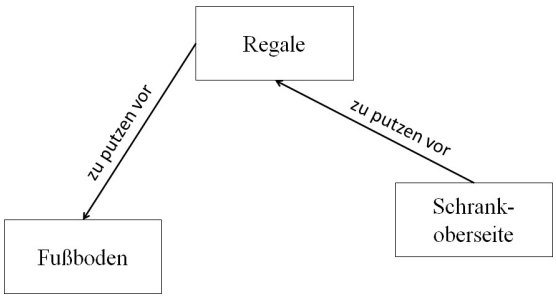

Abbildung 1: azyklischer Abhängigkeitsgraph

Knifflig wird es, wenn wir zusätzlich noch diese Abhängigkeit hinzufügen: „Bevor ich oben auf dem Schrank wische, fege ich den Boden, damit die Leiter einen guten Stand findet.“ Wie Sie in Abbildung 2 sehen, ist nicht nur eine dritte Abhängigkeit dazu gekommen, sondern es ist ein Kreis entstanden, ein Zyklus. Nun ist es nicht mehr möglich, die drei Tätigkeiten in eine lineare Reihenfolge zu bringen, ohne eine der Beziehungen zu verletzen

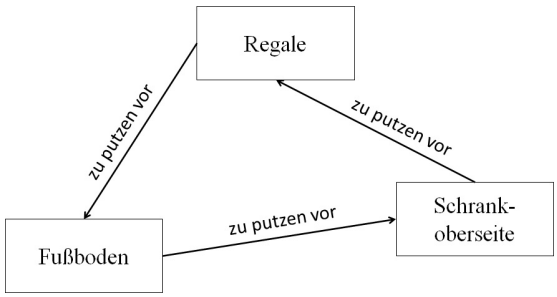

Abbildung 2: zyklischer Abhängigkeitsgraph

Dieses Problem sollten wir genauso vermeiden wie die Zeitreisende das Großmutterproblem: Wenn die Enkelin mit der Zeitmaschine in die Vergangenheit reist und dort die Mutter ihrer eigenen Mutter wird, dann enthält der Stammbaum ebenfalls einen Zyklus der „ist-Mutter-von“-Beziehungen.

Darum lautet das Acyclic Dependency Principle ADP: „Der Abhängigkeitsgraph der Komponenten eines Software-Systems darf keine Zyklen enthalten“. Andernfalls lassen sich die Komponenten weder in einer Reihenfolge noch hierarchisch anordnen. Zyklen können zu Deadlocks oder Endlosschleifen führen. Das ADP ist eines von zahlreichen Software-Design-Prinzipien.

Falls der Abhängigkeitsgraph doch einen Zyklus enthält, kann man diesen eventuell auflösen. In unserem obigen Beispiel kann man zuerst grob fegen, dann die Schrankoberfläche putzen, anschließend die Regale abstauben und zuletzt den Fußboden gründlich fegen. Das Fegen des Fußbodens wird also in zwei Komponenten zerlegt wie in Abbildung 3 dargestellt. Dieser refaktorisierte Abhängigkeitsgraph ist nicht mehr zyklisch.

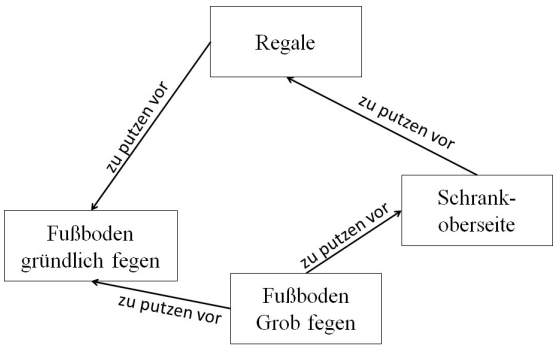

Abbildung 3: refaktorisierter Abhängigkeitsgraph ohne Zyklus

In einem Software-System könnte so eine „zu-putzen-vor“-Beziehung beispielsweise lauten „zu definieren vor“, z. B. bei includes.

Andrea Herrmann