Verwirrende Informatik III - Beschreibungen und Beschriebenes

Johannes Reich, johannes.reich@sap.com

6. Juli 2022

Zusammenfassung

Dieser Artikel ist der dritte einer Viererreihe, die eine Diskussion über die Grundlagen der Informatik anregen soll. In ihm wird das Verhältnis von Beschreibung und Beschriebenem thematisiert. Die zu ungenaue Unterscheidung zwischen Beschreibung und Beschriebenem wird als eine wesentliche Quelle von unnötiger Unklarheit in der Informatik angesehen.

In formalen Sprachen ist diese Unterscheidung die wesentliche Idee. In der natürlichen Sprache ist diese Unterscheidung ebenso wichtig, allerdings ist sie aspekthaft und kontextabhängig. Dies wird in der Informatik v.a. beim Modellkonzept relevant.

Es wird der Vorschlag gemacht, sich bei der Bedeutungserteilung von Beschreibung und Beschriebenem an der mathematischen Modelltheorie zu orientieren, in der eine (konsistente) Beschreibung ein Modell beschreibt. Das Beschriebene wäre damit das, was es zu verstehen gilt, während eine Beschreibung ein Verständnis voraussetzt, also eine "Interpretation" erfordert, um sie auf das "Beschriebene" zu beziehen. Damit würde unser intuitives Verständnis von "Modell" und "Beschreibung" tatsächlich mit der mathematischen Modelltheorie präzisiert.

Als Beispiele, in denen diese Unklarheiten zum Tragen kommen, werden die Model Driven Architecture (MDA) und die Ontologien diskutiert.

1 Einleitung

Dieser Artikel ist der dritte einer Viererreihe, die eine Diskussion über die Grundlagen der Informatik anregen soll. Im ersten hatte ich die Motivation geliefert und die meiner Ansicht nach die Informatik begründenden Konzepte von Systemen, Information, Berechenbarkeit und Daten vorgestellt. Im zweiten Teil hatte ich die Interaktion zwischen Systemen und die Koordination innerhalb von Systemen mit Hilfe des Kompositionskonzeptes thematisiert. In diesem, dritten Teil gehe ich auf das Verhältnis von Beschreibung und Beschriebenem ein. Im noch ausstehenden vierten Teil gehe ich schließlich der Frage nach, welchen Beitrag die Informatik zum Verständnis des Begriffs der Semantik oder Bedeutung leisten kann.

2 Beschreibung und Beschriebenes in formalen Sprachen

Eine wesentliche Eigenheit der Informatik ist die herausragende Bedeutung formaler Kalküle, etwa in der Form von Programmiersprachen. Dem Ansatz von Alfred Tarski [17] folgend sind formale Sprachen nach einem bestimmten Schema aufgebaut. Zunächst wird die Syntax definiert, die aus einer Menge zulässiger Zeichen zusammen mit einer Reihe von Regeln besteht, die beschreiben, welche Ausdrücke gebildet werden können. In einem zweiten Schritt wird dann die Semantik korrekt gebildeter Ausdrücke durch eine Interpretationsfunktion bestimmt, die diese Ausdrücke auf Entitäten abbildet, von denen wir annehmen können, dass sie existieren und über die wir in unserer gewöhnlichen Sprache sprechen können. Das bedeutet, dass wir bei formalen Sprachen vergleichsweise einfach zwischen Beschriebenem und Beschreibung unterscheiden können, weil diese Unterscheidung die wesentliche, ihrer Formalisierung zu Grunde liegende Idee ist.

Dies will ich kurz an dem Beispiel des Aussagenkalküls illustrieren. Unter der Annahme, dass wir schon wissen was eine Aussage ist und dass wir eine Menge von Elementaraussagen kennen, beschreibt der Kalkül, wie man aus einfachen Aussagen durch Und-, Oder-, oder Nicht-Verknüpfung weitere Aussagen erhalten kann. Um sicher zwischen den Ausdrücken des Kalküls und unserer normalen Sprache unterscheiden zu können, setze ich alle Ausdrücke des Kalküls in Anführungszeichen.

Eine besondere Beachtung verdienen die umgangssprachlichen Ausdrücke, die wir verwenden, um die Regeln der Syntax und Semantik zu formulieren. In der Aussagenlogik spricht man auch von Aussageformen. Dort treten Variablen als Platzhalter für Kalkülausdrücke in natürlichsprachlichen Ausdrücken auf, die ansonsten syntaktisch den Kalkülausdrücken entsprechen. Aussageformen notiere ich wie Kalkülausdrücke in Anführungszeichen, wenn sie Symbole des Kalküls enthalten und die Variablen aus dem Bereich der Umgangssprache, die für Kalkülausdrücke stehen, symbolisiere ich mit einem vorangestellten \$-Zeichen, um sie von den Variablen, die Teil des Kalküls sind, sicher unterscheiden zu können.

Die erlaubten Zeichen des Aussagenkalküls werden festgelegt durch das Wertealphabet $\{''w'',''f''\}$, der Menge der Operatorenzeichen $\{''\vee'',''\wedge'',''\neg''\}$, sowie der Menge der Variablen für Aussagen $V = \{''p'',''q'', etc.\}$.

Die Regeln für den Aufbau von Ausdrücken sind:

1. $''w''$ und $''f''$ sind Ausdrücke;
2. jede Variable ist eine Ausdruck;
3. sind $\$a$ und $\$b$ Ausdrücke, dann sind auch $''\neg\$a''$, $''\$a \vee \$b''$ und $''\$a \wedge \$b''$ Ausdrücke.

Die Interpretation eines Ausdrucks $\$a$, $\mathcal{I}_b(\$a)$ liefert uns seine Bedeutung und besteht aus

1. einer Belegung aller Variablen mit Wahrheitwerten: $b : V \to \{wahr, falsch\}$

2. einer rekursiven Vorschrift, die die Bedeutung der Ausdrücke bestimmt:

   (a) $\mathcal{I}_b(''w'') = wahr$; $\mathcal{I}_b(''f'') = falsch$;

   (b) $\mathcal{I}_b(''p'') = b(''p'')$;

   (c) $\mathcal{I}_b(''\neg\$a'') = wahr[falsch]$, wenn $\mathcal{I}_b(\$a) = falsch[wahr]$;

   (d) $\mathcal{I}_b(''\$a \vee \$b'') = wahr$, wenn $\mathcal{I}_b(\$a) = wahr$ oder $\mathcal{I}_b(\$b) = wahr$;

   (e) $\mathcal{I}_b(''\$a \wedge \$b'') = wahr$, wenn $\mathcal{I}_b(\$a) = wahr$ und $\mathcal{I}_b(\$b) = wahr$.

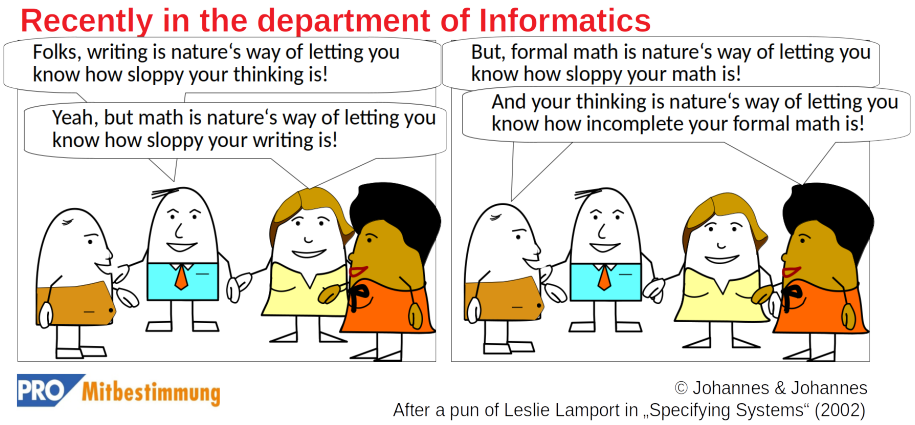

Abbildung 1: Der semantische Zirkel der Formalisierung

3 Beschreibung und Beschriebenes in natürlichen Sprachen

Offensichtlich ist die Unterscheidung zwischen Beschreibung und Beschriebenem auch in unserer natürlichen Sprache, wozu auch die gewöhnliche mathematische Ausdrucksweise gehört, sinnvoll. Allerdings ist sie hier wesentlich schwieriger zu treffen, was man spätestens dann merkt, wenn man in der Informatik mit dem Wort "Modell" konfrontiert wird. Bezeichnet dieses Wort nun eine Beschreibung oder etwas Beschriebenes?

Wie wird dieses Wort in der Mathematik benutzt? In der Modelltheorie als Teilgebiet der mathematischen Logik, ist ein Modell eine mathematische Struktur im Sinne einer Trägermenge mit darauf definierten Relationen, auf die eine Menge von Aussagen zutrifft. Aus Sicht der Modelltheorie ist damit ein Modell genau das, was jemand "verstehen" muss, um die Wahrheit von Aussagen zu überprüfen. Es gilt der Vollständigkeitssatz von Gödel, dass eine Theorie im Sinne einer Menge formaler Sätze genau dann ein Modell hat – es also etwas zu verstehen gibt –, wenn sich aus dieser Menge formaler Sätze keine Widersprüche formal ableiten lassen. Ein Beispiel sind etwa die natürlichen Zahlen inkl.

der Nachfolgeoperation, die ein Modell der (formal verstandenen) Peanoaxiome sind.

Dies ist auch eine der Verwendungen des Wortes "Modell" in den Ingenieurwissenschaften. Mithilfe eines Modells versteht die Ingenieurin den sie interessierenden Aspekt der Welt. In diesem Sinne ist ein Modell gerade *keine* formale Beschreibung, sonderen wiederum eine realweltlichen Entität, die absichtlich gewisse Aspekte der "modellierten" Entität aufweist. Nicht die Symbole der Differenzialgleichung sind das Modell, sondern die Gleichheitsrelation zwischen den Zustandswerten und deren zeitlichen oder räumlichen Änderungen, die wir mit der Differentialgleichung beschreiben. Nicht das Diagramm ist das Modell, sondern das was es bezeichnet. Usw.

Das Problem ist nun, dass Beschreibungen in der natürlichen Sprache weder konsistent, noch vollständig – ja noch nicht einmal "sprachlich" im engeren Sinne sein müssen und trotzdem ihren Zweck erfüllen können. Betrachten wir etwa eine Beschreibung im Sinne einer Spezifikation: Die meisten Ingenieure würden der Behauptung zustimmen, dass eine Spezifikation eines Systems das System beschreibt. So schreibt etwa Leslie Lamport in [12] [S. 1] "*A specification is a written description of what a system is supposed to do*".

Nun kann ich einer Ingenieurin aber auch das System selbst in die Hand drücken und sagen: bau mal nach (– bitte). D.h. man kann auch ein System selbst als eine "Spezifikation" für entsprechende Kopien ansehen, wie man auch einen Algorithmus als Element einer Äquivalenzklasse betrachten kann, das diese und damit alle weiteren Elemente "beschreibt" im Sinne von "festlegt".

Der Unterschied zwischen einer Beschreibung und dem Beschriebenen scheint mir daher im natürlichsprachlichen Ausdruck eher im Verwendungskontext der sprachlichen Ausdrücke zu liegen, als primär in den verwendeten Ausdrücken. Hinzu kommt, dass wir natürlichsprachlich in der Lage sind, mehrere Aspekte gleichzeitig auszudrücken und wir uns dann auf unsere Zuhörer verlassen, die Dinge schon richtig zu sortieren - zur Not können sie ja nachfragen.

Mein Vorschlag ist daher, mit Modell das zu bezeichnen, was es zu verstehen gilt, während eine Beschreibung ein Verständnis voraussetzt, also eine "Interpretation" erfordert, um sie auf das "Beschriebene" zu beziehen.

4 Verwirrung durch ungenaue Unterscheidung zwischen Beschreibung und Beschriebenem

Ein ungenau verstandenes Verhältnis zwischen Beschreibung und Beschriebenem kann für erhebliche Verwirrung sorgen. Zwei prominente Beispiele sind meiner Ansicht nach die "Model driven Architecture" und die "Ontologien".

4.1 Model Driven Architecture

Ein wichtiges Beispiel, wo diese Unterscheidung nicht ausreichend ausgeprägt wurde, ist meinem Verständnis nach die sogenannte "Model-driven-Architecture

(MDA)" der Object Management Group [14, 15]. Das kann man leicht daran erkennen, dass in den Texten dieser Gruppierung sich an vielen Stellen die Wörter "modellieren" und "beschreiben" austauschen lassen, ohne dass sich die erahnte Bedeutung verändert.

In der MDA sind nach eigenen Angaben die Begriffe System, Modell und Metamodell zentral. Systeme werden sehr unspezifisch definiert als *"a collection of parts and relationships among these parts that may be organized to accomplish some purpose."*. Von Supersystemen wird gesagt, dass sie auch als Umgebung ihrer Subsysteme betrachtet werden können. Ein Modell im Sinne der MDA ist *"information selectively representing some aspect of a system based on a specific set of concerns.* und ein Metamodell ist dann *"a model that defines a modeling language and is also expressed using a modeling language."* [15].

Das Systemkonzept der MDA korrespondiert eher mit dem Kompositionskonzept als dem Systemkonzept dieses Artikels. Wenn man der Vorstellung folgt, dass ein System mit seiner Umgebung interagiert, die Umgebung im Zweifelsfall also wiederum als System modelliert, dann wäre es unsinnig im Sinne dieses Artikels davon zu reden, dass ein Supersystem eine Umgebung seiner Subsysteme ist – weil es gerade zwischen den Subsystemen und dem Supersystem keine Interaktionsbeziehung, sondern eine "Ist-Teil-von"-Beziehung besteht. Genau genommen ist die MDA-Systemdefinition so unspezifisch, dass schlicht unklar bleibt, ob die vertretene Auffassung, Supersysteme seien als Umgebung ihrer Subsysteme aufzufassen, auch aus Sicht der MDA zutreffend ist oder nicht.

Ebenso käme niemand auf die Idee, gestützt auf den Informationsbegriff dieser Artikelserie zu sagen, "Ein Modell ist Information, die ... <irgendeine einschränkende Bedingung>". Und die versuchte Definition des Metamodells zeigt sehr schön den verwirrenden Bezug in der MDA zwischen natürlich-sprachlichen Beschreibungen und dem Beschriebenen.

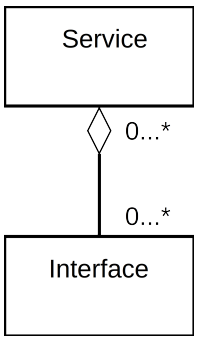

Abbildung 2: Ein typisches UML 2.0 Klassendiagramm, das mittels Aggregation ausdrücken soll, dass ein Service kein, ein oder mehrere Interfaces hat und ein Interface zu keinem, einem oder mehreren Services gehören kann (Ausschnitt aus Abb. 2 in [2]).

Nehmen wir als Beispiel das "Modell" aus [2]: eine Klasse "Service" hat eine

Relation "has a" im Sinne der Aggregation mit einer weiteren Klasse "Interface" (s. Abb. 2). Was ist eine Klasse? Was ist ein Service? Was ist ein Interface? Nichts an diesem "Modell" ist klar. Nehmen wir an, eine Klasse wäre eine Äquivalenzklasse von Systemen in unserem Sinne, also eine Menge von Systemen, die gemäß einem Kriterium zueinander äquivalent sind, mit ihrer Kombination aus Systemfunktion und dem von ihr gekapselten inneren Zustand. Dann wäre auch das Interface, das ja auch als "Klasse" modelliert wurde, ebenfalls eine solche Äquivalenzklasse von Systemen. Dann wäre die Beziehung zwischen den Äquivalenzklassen "Service" und "Interface" eine hierarchische Teil-Ganze-Beziehung, die im besten Fall durch äquivalente Interaktionen einzelner Elemente der "Interface"-Klasse mit weiteren, unerwähnten Subsystemen zustande käme, so dass beide zusammen ein Supersystem bilden, das wiederum Element der Äquivalenzklasse "Service" wären.

Das wäre ein gänzlich anderes Verständnis von System und Interface wie ich es im zweiten Artikel dieser Serie erläutert hatte. Dort hatte ich die Auffassung vertreten, dass ein Interface eines Systems alle Angaben des Systems umfasst, die ein (noch zu bestimmender) Kompositionsoperator benötigt. Das umfasst auf jeden Fall je nach Interface-Klasse unterschiedliche Teile eines Systems. Das wird in diesem "Modell" vollständig ignoriert. Gut, könnte man sagen, dann ist die Identifikation von "Klasse" und einer Äquivalenzklasse mathematischer Strukturen a la "System" im Sinne dieses Artikels eben falsch. Die Frage, die sich dann aber sofort stellt ist: Welche Identifikation in Bezug auf eine mathematische Struktur ist dann die richtige? Und, ist sie immer dieselbe? Dazu werden in [2] keine klaren Angaben gemacht. In einem gewissen Sinn ist demnach die sogenannte MDA ganz und gar nicht "model driven".

4.2 Ontologien

In den 1980er Jahren führte man im Bereich der Forschung zu künstlicher Intelligenz sogenannte Ontologien ein als Modellierungswerkzeug, wie auch als Komponente Wissensbasierter Systeme ein. Nach Thomas A. Gruber [8], ist eine Ontologie eine "*explicit specification of ... the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them*", also eine explizite Spezifikation einer Konzeptualisierung im Sinne von [7].

C. Feilmayr und W. Wöß [6] behaupten entlang dieser Tradition, "*An ontological representation allows modeling meaning in systems that are to be implemented using a programming language and a database schema*" und dass ein Ontologie-basiertes Applikationsdesign zum einen "*a clear and machine-interpretable basis for meaning*" erlaube und zum anderen dazu führe, dass "*concepts, their meaning, and their relationships can be shared*".

Aktuell positionieren sowohl die IEC [10] als auch die DIN/DKE Normungsroadmap Industrie 4.0 [4] Ontologien als eine "semantische Technologie", die es erlaubt, die Interoperabilität zwischen IT-Applikationen zu vereinfachen.

Zunächst fällt rein empirisch auf, dass so verstandene Ontologien, trotz ihrer nun schon über 30-jährigen Geschichte bei weitem nicht so erfolgreich zur

besseren Interoperabilität von IT-Applikationen beitragen, wie gemäß ihres Anspruchs eigentlich zu erwarten wäre. Francesco Lelli [13] gibt einen Überblick über die Heterogenität der Ansätze, mithilfe von Ontologien die Welt des "Internet of Things" zu fassen. Der populärste fehlgeschlagene Versuch, mittels einer Ontologie semantische Interoperabilität zu vereinfachen, ist wohl der des "Semantic Web" [18]. Woran liegt das?

Die Frage ist: Stimmt denn überhaupt die Annahme, dass Ontologien die Bedeutung und Beziehungen zwischen Konzepten formal repräsentieren? Nehmen wir einmal die Beziehung "Ein Ding hat 1..n Eigenschaften". Wenn man das weiß, kann man etwa die Eigenschaften eines Dings aufzählen. Oder man könnte alle Dinge erfahren, die eine bestimmte Eigenschaft haben.

Aber: was macht eine IT-Applikation aus dieser Beziehung? Sie erweitert ihren Datentyp für Entitäten um einen Wert für "Ding" und einen Wert für "Eigenschaft" und setzt diese in eine vom Entwickler vorgedachte, parametrisierte $1 : n$-Beziehung. Diese Repräsentation unterscheidet sich strukturell in nichts von einer behaupteten Beziehung "Ein Klhxggs hat 1:n Qxxytpgs". Auch hier ließen sich alle Qxxytpgs aufführen oder alle Klhxggs mit einer bestimmten Qxxytpgs auflisten.

Tatsächlich ist unsere alltagssprachliche Interpretation der Terme "Ding" und "Eigenschaft" und deren Interpretation durch eine IT-Applikation voneinander grundverschieden. Das Beispiel des Konzepts der Eigenschaft zeigt diesen Unterschied sehr schön. Im Bereich der Ontologien ist man der Auffassung, Eigenschaften ließen sich durch einen Konsenses, etwa in Form eines Industriestandards, wohldefinieren und dann bräuchte man dafür nur noch einen eindeutigen Namen als Bezeichner und alle wüssten, was gemeint ist [5].

Ich möchte zwei Beispiele anführen, die zeigen sollen, dass der Eigenschaftsbegriff deutlich komplexer ist und eine natürlichsprachliche Definition nicht unbedingt dazu führt, dass alle Leser ein ausreichend gleiches Verständnis erlangen werden.

Zunächst die temporalen Eigenschaften von Systemen: In der theoretischen Informatik werden einfache temporale Eigenschaften von Systemen mit Bezug auf konkrete Systemmodelle als Menge von Sequenzen konsekutiver Systemzustände (execution traces) definiert werden, etwa als Safety- oder Livenesseigenschaften [1]. Von dieser Idee ausgehend haben Michael R. Clarkson und Fred Schneider [3] festgestellt, dass man für viele wichtige temporale Eigenschaften diesen Ansatz auf Mengen von Mengen solcher Sequenzen erweitern muss. Sie sprechen daher von "Hyperproperties". Typische Beispiele solcher Hypereigenschaften sind etwa eine garantierte mittlere Antwortzeit oder eine mittlere Verfügbarkeit.

Weiterhin sind viele wichtige Eigenschaften ihrer Natur nach relational in dem Sinne, dass sie sich in ihrer Bestimmung immer auch auf weitere Systeme beziehen. Ein wichtiges Beispiel sind Sicherheitseigenschaften. In der modernen Kryptographie wird Sicherheit mittlerweile in Bezug auf die Fähigkeiten eines Gegners definiert [11], etwa dass dieser in der Lage ist, probabilistische Polynomialzeit-Algorithmen zu verwenden oder dass ihm gewisse Informationen zur Verfügung stehen. Eine Einsicht, die natürlich auch in der gewöhnlichen

Kriminologie gilt. Diese kontextuale Eigenschaft gewisser Eigenschaften macht ihre Existenz u.U. zeitabhängig: Was heute noch als "sicher" galt, muss es morgen nicht mehr sein – ohne dass sich das eigentliche System unter Betrachtung geändert hat.

Wo findet sich eine solche Unterscheidung in unserem "ontologischen Modell"? Diese Beispiele zeigen nicht, dass man Eigenschaften nicht durch Werte ausdrücken kann. Sie zeigen aber, dass die Annahme, eine solche Repräsentation der Ausprägung einer Eigenschaft in Verbindung mit einer textuellen, natürlichsprachlichen, möglichst einfachen und knappen Beschreibung würde in einem ontologischen Sinne das Problem der Mehrdeutigkeit lösen, falsch ist.

Eine Ontologie würde nach meinem Verständnis ihrem Anspruch daher nur gerecht, wenn die Konzepte selbst so klar wären, dass sie sich tatsächlich durch ein Symbol in einer formalen Sprache repräsentieren ließen. Aber genau das ist in der Regel nicht der Fall, schon gar nicht in einer Welt mit Lug und Trug. Insofern trifft die von C. Feilmayr und W. Wöß in [6] erwähnte wichtige Limitierung, dass man sich für eine Ontologie auf ein gemeinsames Verständnis der grundlegenden Begriffe geeinigt haben muss, nur die Spitze des Eisbergs. Dazu passt, dass auch keine Einigkeit über die grundlegenden Konzepte der Interoperabilitätsdomäne wie System, Interaktion, Interface, Semantik, Service, Applikationsschicht, Verwaltungsschale, Teilmodell, Fähigkeit, etc. besteht. Ja, tatsächlich noch nicht einmal eine hinreichende Einigkeit über das Konzept der Ontologie selbst, etwa in Abgrenzung zur Taxonomie [6]. So formuliert Martin Hepp [9] "*In fact, it is a kind of paradox that the seed term of a novel field of research, which aims at reducing ambiguity about the intended meaning of symbols, is understood and used so inconsistently*".

Wie sollen Ontologien unter diesen Umständen die semantische Interoperabilität zwischen IT-Applikationen wesentlich erleichtern? Das ist kein Argument gegen die Formalisierung von Beziehungen zwischen Konzepten, sondern gegen das Verständnis solcher Formalisierungen als eine "machine-interpretable basis for meaning". Stattdessen basieren sie selbst, wie alle Formalisierungen, auf von ihnen unabhängigen natürlichsprachlich schon vorhandenen Bedeutungen. Entsprechend sind ihre Anwendungsfälle zu verorten.

Tatsächlich lässt sich ohne Bezug zu einem Interaktionskontext kaum ermessen, in welcher Tiefe und Breite die Interaktionspartner ein gemeinsames Verständnis einer "Konzeptualisierung" benötigen. Nehmen wir das Schachspielen als Beispiel. Die Regeln sind schnell erklärt und die notwendige Sprache für das Spiel ergibt sich aus der Mitteilungsnotwendigkeit der Spielzüge. Das war's. Was hätten wir alles definieren können, wenn wir ohne den konkreten Bezug zur Interaktion des Schachspielens angefangen hätten, eine "Wissensrepräsentation" über Schachspielen aufzubauen, was alles hinter dem Konzept des Spielzuges steckt, etc. – vielleicht noch in einem erforderlichen Konsens mit anderen? Interessanterweise ist der Term "Spielzug" in der erforderlichen Sprache eines Schachdialogs überflüssig.

5 Diskussion

Eines meiner Ziele in diesem Artikel war aufzuzeigen, dass wir in unserer natürlichen (Ingenieurs-) Sprache auf Grund ihrer Kontextsensibilität wirklich aufpassen müssen, um Beschreibungen und Beschriebenes voneinander sinnvoll abzugrenzen.

Hier kommt uns die Mathematik in zweierlei Weise zur Hilfe. Zum einen hat sie uns mit der Formalisierung ein Konzept aufgezeigt, was die Trennung zwischen Beschreibung und Beschriebenem zum Programm erhoben hat. Zum anderen lässt sie uns über die Dinge die wir meinen in unserer natürlichen Sprache mit einer ansonsten nicht möglichen Präzision sprechen (s. Abb. 1).

Mein Vorschlag ist, sich bei der Bedeutungserteilung von Beschreibung und Beschriebenem an der mathematischen Modelltheorie zu orientieren und mit Modell das zu bezeichnen, was es zu verstehen gilt, während eine Beschreibung ein Verständnis voraussetzt, also eine "Interpretation" erfordert, um sie auf das "Beschriebene" zu beziehen. Das hätte den Charme, dass wir unser intuitives Verständnis von "Modell" und "Beschreibung" tatsächlich mit der mathematischen Modelltheorie präzisieren können.

Dann kann ich sagen: "ich modelliere ein System mit einem Tupel aus Alphabeten, Zustandsfunktionen, Zeitfunktion und Systemfunktion" – und werde verstanden als: *"Aha, er meint, dass ich mit dieser mathematischen Struktur verstehe, was ein System ist."*. Und ich kann sagen, "Ich beschreibe das Verhalten meines Systems mit einem I/O-TS" – und werde verstanden als: *"Aha, er meint das I/O-TS und das Verhalten eines System sind zweierlei und ich muss das Verhalten kennen, um zu verstehen, dass das I/O-Transitionssystem das Verhalten beschreibt. Hat er sich hier wirklich präzise ausgedrückt? Was ist denn das Verhalten des Systems, wenn nicht das I/O-TS? Ah stimmt, hier hat er ja definiert, dass ein I/O-TS das Verhalten eines Systems repräsentiert, genau dann wenn seine Transitionsrelation mit dem System irgendwie kompatibel ist. Aber, hätte er nicht auch "modellieren" sagen können. Dann hätte er doch gemeint, dass das I/O-TS ein Modell des Systemverhaltens ist. Das hätte ich auch verstanden . . . "*.

Mit dieser Unterscheidung wird auch die Beziehung zwischen Softwareengineering und Systemengineering deutlich (z.B. [16]): Das Softwareengineering beschäftigt sich vordringlich mit der Frage, wie man informatische Systeme am geschicktesten beschreibt, während das Systemengineering sich vordringlich mit dem Design von Systemen, in erster Ordnung unabhängig von ihrer Beschreibung, beschäftigt. Und ihre enge Beziehung wird durch die zwischen Beschreibung und Beschriebenem erzwungen.

Literatur

[1] C. Baier and J.-P. Katoen. *Principles of Model Checking.* The MIT Press, 2008.

[2] *Details of the Asset Administration Shell. Part 2 – Interoperability at Runtime – Exchanging Information via Application Programming Interfaces (Version 1.0RC02).* Bundesministerium für Wirtschaft und Energie (BMWI), 2021.

[3] M. R. Clarkson and F. B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.

[4] *Deutsche Normungsroadmap Industrie 4.0, Version 4.* DIN e.V. und DKE, 2020.

[5] U. Epple, M. Mertens, F. Palm, and M. Azarmipour. Using Properties as Semantic Base for Interoperability. *IEEE transactions on industrial informatics*, page 9 Seiten, 2017. Online-First.

[6] C. Feilmayr and W. Wöß. An analysis of ontologies and their success factors for application to business. *Data & Knowledge Engineering*, 101:1–23, 2016.

[7] M. R. Genesereth and N. J. Nilsson. *Logical foundations of artificial intelligence.* Morgan Kaufmann, 1987.

[8] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, 43(5):907–928, 1995.

[9] M. Hepp. Ontologies: State of the art, business potential, and grand challenges. *Ontology Management*, pages 3–22, 2008.

[10] *Semantic interoperability: challenges in the digital transformation age.* IEC, 2019.

[11] J. Katz and Y. Lindell. *Introduction to Modern Cryprography.* Chapman & Hall/CRC, 2 edition, 2015.

[12] L. Lamport. *Specifying systems.* Addison-Wesley Boston, 2002.

[13] F. Lelli. Interoperability of the time of industry 4.0 and the internet of things. *Future Internet*, 11(2):36, 2019.

[14] The mda foundation model, ormsc draft. Technical report, Object Management Group, 2010.

[15] Model driven architecture (mda)–mda guide rev. 2.0. Technical report, Object Management Group, 2014.

[16] A. Pyster, R. Adcock, M. Ardis, R. Cloutier, D. Henry, L. Laird, M. Pennotti, K. Sullivan, J. Wade, et al. Exploring the relationship between systems engineering and software engineering. *Procedia Computer Science*, 44:708–717, 2015.

[17] A. Tarski. Der Wahrheitsbegriff in den formalisierten Sprachen. *Studia Philosophica. Commentarii Societatis Philosophicae Polonorum*, 1:261 – 405, 1935. Auch in: Tarski, Givant, McKenzie (1986), S. 53 – 197.

[18] O. L. Tim Berners-Lee, James Hendler. The Semantic Web: a new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 284(5):34–43, 2001.