BARBARA PAECH

WAS LEISTET DIESE SOFTWARE FÜR MICH?

Über das Verständnis von Software

In der digitalen Gesellschaft ist Software allgegenwärtig. Nicht nur Entwickler*innen sprechen über Software, auch Nutzer*innen unterhalten sich über Softwareprodukte, z.B. über die Apps auf ihren Mobiltelefonen, über die Tücken von Textverarbeitung, Navigation oder Software im Büro. Sie beschreiben die Features der Software, d.h. was die Software für sie leistet. Ein Feature ist dabei jede für Nutzer*innen sichtbare Eigenschaft einer Software. Das Verständnis von Features ist geprägt durch Marketingbeschreibungen, die Nutzung und andere Nutzer*innen. Es ist bekannt, dass dieses Verständnis zusammenhängt mit der von den Nutzer*innen wahrgenommenen Bedienbarkeit und Nützlichkeit und damit auch der erfolgreichen Nutzung von Software im Beruf.[1] Wie das Verständnis aussieht ist aber bisher kaum erforscht.

Bei einem Marsilius-Treffen hatte ich die Fellows gebeten, 10 Features eines Texteditors aufzuschreiben. Wie zu erwarten gab es wenig Übereinstimmung dabei. Die Rückmeldungen waren insbesondere von unterschiedlicher Granularität. Viele listeten einzelne Formatierungsfunktionen auf, wie Schriftgröße. Andere nannten Formatierung als ein Feature und Einzelne betrachteten ganz andere Bereiche, wie z.B. die Kommunikation mit anderen Windows-Funktionen.

In der Informatik beschäftigen sich das Requirements Engineering und die Mensch-Maschine-Interaktion mit diesem Verständnis. Im Requirements Engineering[2] soll durch Kommunikation zwischen Hersteller und Entwickler*innen einerseits und Auftraggeber und Nutzer*innen andererseits ein Verständnis über die Anforderungen (zu entwickelnder Features) einer Software hergestellt werden. Aufgrund von Zeitmangel (Nutzer*innen werden nicht freigestellt), Missverständnissen und Konflikten und

insbesondere, weil es schwer ist, sich zukünftige Features vorzustellen, muss über die ganze Entwicklungszeit kontinuierlich und iterativ eine Abstimmung gesucht werden. Dies wird insbesondere bei der agilen Entwicklung unterstützt.[3] Die Requirements Engineering Praxis und Forschung stellt Methoden zur Beschreibung von Anforderungen bereit, beschäftigt sich aber nicht mit dem gewonnenen Verständnis.[4] Das Gebiet der Mensch-Maschine-Interaktion[5] stellt die Schnittstelle zwischen Mensch und Maschine in den Mittelpunkt. Praxis und Forschung beschäftigen sich damit, wie man die Software leicht bedienbar machen kann und dabei insbesondere die Produkterfahrung verbessert. Es geht um das Verständnis der Schnittstelle (wie kann ich ein Feature zur Ausführung bringen), aber nicht um das Verständnis der Softwareleistungen insgesamt. Die Leistungen prägen die Außensicht auf Software. Die Innensicht ist geprägt durch die Komponenten und Abläufe in der Software, die die Features bereitstellen. Diese Innensicht beschreiben die Entwickler*innen im Detail durch den Code und zu Kommunikationszwecken abstrakter durch die Unified Modeling Language (UML).[6] Es fehlt eine Sprache für die Außensicht auf Software, mit der Nutzer*innen ihr Verständnis gut beschreiben können.

In meinem Marsilius-Jahr habe ich die Zeit genutzt, mich diesem Verständnis auf zwei Wegen zu nähern. Zum einen habe ich zusammen mit Michel Wensing eine empirische Studie durchgeführt, in der wir erforscht haben, wie Ärzt*innen Informationstechnologiesysteme in der Primärversorgung erlernen, verstehen und nutzen. Neben einer Beschreibung dieser Verhaltensweisen untersuchten wir auch deren Zusammenhang mit der allgemeinen Technikaffinität. Diese Studie und ihre Ergebnisse sind näher beschrieben in Michel Wensings Bericht zum *Erlernen von Informationstechnologie-Systemen im Gesundheitswese*n (S. 194). Wie auch schon in früheren Kooperationen mit Kolleg*innen aus dem Bereich Medizininformatik, war die interdisziplinäre Zusammenarbeit sehr fruchtbar und erstaunlich einfach. Ein wesentlicher Grund dafür ist, dass die Durchführung empirischer Studien in unseren Disziplinen ähnlich ist (wenn auch in der Informatik weniger rigide als in der Medizin).

Zum anderen habe ich die Gelegenheit genutzt, die Sichtweisen anderer Disziplinen kennenzulernen. In allen Disziplinen werden Systeme betrachtet und deren Leistungen beschrieben. Die Informatik hat sich sehr schnell entwickelt. Vielleicht kann sie von anderen Disziplinen lernen, welche Konzepte wesentlich sind im Kontext von Leistungen? Ich habe deshalb mit vielen Fellows ein ca. einstündiges Gespräch geführt und dabei in Bezug auf Leistung Träger, Begrifflichkeiten und Vorgehensweisen

erfragt. Die folgende Zusammenfassung gibt mein grobes Verständnis wieder, das aus Sicht der jeweiligen Disziplinen deutlich verfeinert und evtl. auch relativiert werden müsste. Ich vergleiche die Disziplinen mit dem Blick der Informatik auf ein Computersystem (also das System, das die Software ausführt).

In der UML werden drei typische Systembeschreibungen für unterschiedliche Innensichten auf ein System eingesetzt:

- Komponentendiagramm: Das System als Netzwerk von Komponenten/Akteuren und deren Beziehungen.
- Aktivitätsdiagramm: Das System mit Ein- und Ausgaben und Prozessen, die die Eingaben in die Ausgaben überführen.
- Zustandsdiagramm: Das System als Regelkreis mit Zuständen und Transitionen zwischen ihnen, die durch Ereignisse ausgelöst werden.

Betont man eine Sicht, so kann man auch von *Systemtyp* sprechen.

Zuordnung Beispielsystem zu Systemtypen	Soziale Subsysteme	System mit Akteuren, Komponenten und Beziehungen (Netzwerk)	(Durch-Staat) Geregeltes System	System mit Ein- und Ausgaben und Prozessen	System mit Stoffkreislauf
Literatursystem	X				
Kultursystem	X	X			
Familiensystem		X			
Gesellschaft		X			
Klinik		X			
Gehirn		X			
Politisch/ Sozial/ Rechts-System		X	X		
Wirtschaftssystem		X	X	X	
Gesundheitssystem		X	X	X	
Umweltsystem		X	X	X	X
Computersystem		X	X	X	

Tabelle 1 gibt einen Überblick über die genannten Systeme (als Träger von Leistungen) und ihre Gruppierung als Systemtypen.

Die Systemtypen der UML werden überwiegend auch in den anderen Disziplinen benutzt. Darüber hinaus wurden noch der Stoffkreislauf in der Chemie genannt sowie der Begriff der sozialen Subsysteme (vgl. Luhmann[7]), der sowohl das Literatursystem (also die Erzeugung und den Gebrauch von Literatur) als auch das Kultursystem (Entstehung von und Leben in der Kultur) charakterisiert. In vielen Disziplinen werden wie in der Informatik mehrere Systemtypen verwendet.

System vs. Leistung	Leistung = Zustand = Kennzahl (bewertbar)	Leistung = Funktion = Handlungsermöglichung
Literatursystem	Alphabetisierung	
Familiensystem	Schutz	Aufstieg
Klinik	Qualität	**Abrechenbare** Leistung, z.B. Operation
Gehirn		Kognition
Politisch/Sozial/ Rechts-System	Gerechtigkeit	-
Wirtschaftssystem	Wirtschaftsleistung	
Gesundheitssystem	Qualität	
Umweltsystem	**Wirkung von Stoffen**	
Computersystem	Sicherheit, Effizienz	Ausführbare Funktionen
Produkt /Gegenstand	*Feature (Marketing)*	*Feature (Marketing)*
Person	*Körperliche Leistung*	*Aufgabe,* ***Soziale Kompetenz, Rechtsfähigkeit***

Tabelle 2 gibt einen Überblick über die für die Systeme genannten Leistungen (Beispiele)

Wie in der Informatik betrachten viele Disziplinen zwei Arten von Leistungen:

- Leistung als Zustand bzw. Eigenschaft, die bewertbar ist und sich durch Kennzahlen beschreiben lässt.
- Leistung als eine Funktion, die etwas ermöglicht (insbesondere Handlungen).

Die Wirkung von Stoffen passt nicht ganz in das Schema, lässt sich aber eher einer Eigenschaft als einer Funktion zuordnen. In vielen Bereichen wird über Produkte bzw. Gegenstände sowie Personen gesprochen. Deren typische Leistungen sind in den letzten beiden Zeilen beschrieben. Herausgehoben sind Begriffe, die in der Informatik nicht standardmäßig angewendet werden. Die Abrechnung einzelner Software-Features etablierte sich erst in den letzten Jahren im Bereich des service-orientierten bzw. Cloud Computing. Fähigkeiten eines Computersystems, über die Bereitstellung von

Funktionen hinaus, werden vorrangig im Bereich intelligenter Systeme thematisiert, z.B. bei Sprachverarbeitung oder automatischen Entscheidungen. Interessant ist auch, dass z.B. bei einem politischen System die Leistung des Systems betrachtet wird, gleichzeitig aber auch Forderungen an die Akteure im System gestellt werden. Die Leistungen der Akteure müssen den Forderungen entsprechen, damit das System seine Leistung erbringen kann. In der Informatik wird der Begriff des design-by-contract ähnlich verwendet: nur wenn der Aufruf einer Komponente die Vorbedingung der Komponente erfüllt, kann die Komponente die (mit der Nachbedingung) versprochene Leistung bringen.

Es war sehr interessant zu sehen, wie unterschiedlich die Begriffe System und Leistungen in den verschiedenen Disziplinen verwendet werden, bzw. welche Aspekte davon jeweils fokussiert werden. Im Vergleich mit der Informatik ergaben sich keine völlig neuen Aspekte. Das kann natürlich auch daran liegen, dass ich zu sehr aus der Perspektive der Informatik gefragt habe. Es wurde aber auch deutlich, dass andere Disziplinen Aspekte betrachten, die erst mit der Entwicklung der Technologien für die Informatik relevant werden (wie z.B. abrechenbare Leistungen oder Kompetenzen). Es wäre interessant, diesen Weg weiter auszuloten, um ggf. Aspekte zu identifizieren, die in der Zukunft für die Informatik relevant werden können.

In vielen Gesprächen ging es auch darum, wie man das Verständnis von etwas untersucht. Dabei wurde unterschieden (wie bei dem Begriff der Perzeption allgemein):

- Verständnis als Ergebnis der Wahrnehmung. Andere Begriffe in diesem Zusammenhang waren Repräsentation und die ethische Perspektive
- Verständnis als Prozess der Wahrnehmung oder auch Rezeption.

Ersteres kann durch Beobachtungen untersucht werden. Dabei sind Handlungen (Repertoire) und Material (Gegenstände, Aussagen, Texte) wichtig, in denen sich das Verständnis äußert. Dieses Ergebnis ist durch Vorerfahrung und Kompetenzen geprägt. Zweiteres kann untersucht werden durch die Änderung des (z.B. zeitlichen, örtlichen) Kontexts der Wahrnehmung. Erste Schritte davon haben wir im Projekt mit Michel Wensing umgesetzt, indem wir die Studienteilnehmer*innen gefragt haben, welche Leistungen sie wie gut kennen, aber auch wie sie das Verständnis jeweils erlangt haben. Weiterhin haben wir die Vorerfahrungen anhand der Technikaffinität zu einem gewissen Grad erfasst.

Die oben beschriebenen Erkenntnisse meiner Gespräche sind eingegangen in einen Forschungsantrag, in dem ich mit einem Kollegen untersuchen möchte, wie Nutzer*innen in verschiedenen Kontexten und Situationen über Software sprechen, welche Ontologie dieser Sprache unterliegt und wie man diese Sprache verbessern kann, damit Nutzer*innen und Entwickler*innen effektiver miteinander kommunizieren.

Dies ist natürlich nicht der einzige Weg, die Kommunikation zu verbessern. Sehr wichtig ist auch, die informatische Bildung allgemein an den Schulen zu verbessern. Diese geht über die digitale Bildung hinaus.[8] In unseren Gesprächen wurde immer wieder deutlich, dass viele Fellows an der Schule etwas Informatikunterricht hatten. Dieser wirkte aber eher abschreckend, da der Wissensstand der Schüler*innen und insbesondere die Erfahrung im Umgang mit dem Computer sehr unterschiedlich waren und die Lehrer*innen das in der kurzen Zeit nicht ausgleichen konnten. Viele Fellows bedauern insbesondere, dass sie nicht programmieren gelernt haben, weil sie das nun bei ihrer Forschung brauchen, z.B. für die Auswertung von Daten mit R und Phython. Informatik muss ein Pflichtfach an Schulen werden!

In vielen Berichten ist schon ein Loblied auf das Marsilius-Kolleg gesungen worden. Ich reihe mich in den Kanon ein. Es ist sehr bereichernd, sich ohne Projektdruck mit anderen Disziplinen auseinanderzusetzen. Wir haben viele Gemeinsamkeiten und Unterschiede entdeckt. Für mich persönlich sehr bereichernd waren auch die Besuche bei den Fellows zu den Gesprächen. Dadurch habe ich die Universität nochmal ganz neu kennengelernt. Bernd Schneidmüller und Thomas Rausch in der Leitung, unterstützt durch Tobias Just, haben es uns leicht gemacht, uns kennenzulernen, uns wissenschaftlich auszutauschen und durch die verschiedenen Formate, wie z.B. „Marsilius kontrovers“ uns in die Diskussion in der Universität und nach außen einzubringen. Ich danke ihnen dafür und der Universität, dass sie diesen Freiraum ermöglicht.

1 Vgl. **E. R. Bravo**, **M. Santan** und **J. Rodon**: *Information Systems and Performance: the Role of Technology, the Task and the Individual*, in: *Behaviour & Information Technology* 34 (2015), S. 247-260.

2 Vgl. **K. Pohl** und **Ch. Rupp**: *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering*, San Rafael, CA 2015.

3 Vgl. https://agilemanifesto.org/, aufgerufen am 14.5.2019.

4 Vgl. **S. Lauesen**: *Software Requirements: Styles and Techniques*, Boston, MA 2001.

[5] Vgl. **M. Hassenzahl**: *Interaktive Produkte wahrnehmen, erleben, bewerten und gestalten*, in **M. Eibl** et al.: *Knowledge Media Design – Grundlagen und Perspektiven einer neuen Gestaltungsdisziplin*, München 2005, S. 151-171.

[6] https://www.uml.org/, aufgerufen am 14.5.2019.

[7] https://de.wikipedia.org/wiki/Systemtheorie_(Luhmann), aufgerufen am 14.5.2019.

[8] https://www.informatics-europe.org/images/documents/informatics-education-acm-ie.pdf.