

Software-Entwicklung: Formale Probleme und kreative Prozesse – Versuch einer Typologie

Abstract zu einem Referat auf dem Workshop „Werkzeug – Denkzeug ... Transmedialität kreativer Prozesse“ am 11./12.02.2011

Irmhild Rogalla, Institut für praktische Interdisziplinarität (Institut PI), Berlin

Software-Entwicklung gilt gemeinhin als rationale Leistung schlechthin. Hier werden Abstraktionen mittels Algorithmen zu maschinellen Funktionen - „Programmen“ -, die ihrerseits wieder als Werkzeug oder Mittel zum Zweck dienen. Andererseits erfordern Entwicklungsprozesse unbestritten immer auch kreative Lösungen. Ja mehr noch: Die Entwicklung der Informationstechnik und 'des' Internets steht seit langem für Innovationen und Veränderungen schlechthin.

Stellen also Rationalität und Kreativität in der Software-Entwicklung keinen Widerspruch dar? Unter Rationalisierung soll hier mit Max Weber die „Entzauberung der Welt“, also die prinzipielle Beherrschung aller Dinge durch Berechnen, verstanden werden. Kreativ ist zunächst einfach derjenige Prozess, in dem Menschen etwas Neues hervorbringen. Tätigkeiten oder ihre Ergebnisse können dann kreativ sein, Kreativität bezeichnet die entsprechende Fähigkeit. Allerdings verbinden sich mit Kreativität viele unterschiedliche Assoziationen, die vom Basteln im Kindergarten bis zur epochemachenden Innovation, von der unkonventionellen Problemlösung bis zum unkontrollierten Chaos reichen. Statt von kreativem soll daher im Folgenden von indeterminiert-schöpferischem Handeln die Rede sein. Noch deutlicher erscheint dann der Widerspruch zwischen Rationalität, Berechnung und damit Determinismus auf der einen und Kreativität, Schöpfertum und damit Indeterminismus auf der anderen Seite.

Wo treten in der Software-Entwicklung formale Probleme und Berechnungen auf? Welche indeterminiert-schöpferischen Leistungen werden vollbracht, was wird hier von wem wie gestaltet?

Ein Einblick in ausgewählte Phasen der Software-Entwicklung hilft hier weiter. Den Kern der Software-Entwicklung bildet die Implementierung. Sie hat sich in den letzten Jahrzehnten deutlich gewandelt: Das mechanische Codieren wird zunehmend mit entsprechende Techniken und Werkzeugen automatisiert. Programmieren bedeutet daher im wesentlichen Algorithmisieren. Dies entspricht der Anwendung definierter, schematischer (Berechnungs-)Verfahren zur Lösung bestimmter Klassen von Aufgaben. Nun wusste aber schon Kant, dass es für das Anwenden von Regeln selbst keine Regeln geben kann. Die erste formale Herausforderungen in der Software-Entwicklung heißt also Algorithmisierung. Ihre Bewältigung erfordert indeterminiert-schöpferisches Handeln durch auswählen, kombinieren, anwenden, konkretisieren und möglicherweise auch erfinden von (Berechnungs-)Verfahren.

Voraussetzung der Algorithmisierung ist die Formalisierung, also die formale oder schematische Darstellung der zu lösenden Aufgaben mit Hilfe eines Kalküls aus entsprechenden Symbolen, Formations- und Umformungsregeln (die Algorithmen sein können, aber nicht müssen) sowie Axiomen. Einfachstes Beispiel eines mathematischen Kalküls ist die Arithmetik. Formalisierung heißt in einem ganz einfachen Fall also eine sprachliche Beschreibung (Textaufgabe: Martin hat drei Äpfel, Monika fünf Birnen, Max eine Orange, wie viel Obst haben die Kinder zusammen?) in eine dem Kalkül entsprechende schematische Darstellung ($3+5+1 = x$) umzuwandeln. Die zweite formale Herausforderung in der Software-Entwicklung heißt Formalisierung. Ihr Gegenstand ist der Schritt von der Bedeutung zur Abstraktion, zur formalen oder symbolischen Darstellung. Ihre Bewältigung erfordert indeterminiert-schöpferisches Handeln in der Darstellung und damit Interpretation von Sachverhalten in schematischer oder eben formaler Weise. Denn hierbei gibt es häufig eine Vielzahl unterschiedlicher Möglichkeiten, sei es durch die Verwendung unterschiedlicher Kalküle oder auch innerhalb eines Kalküls. („Verteilen Sie das Obst gleichmäßig“ kann unter anderem als $(3+5+1)/3$, als $3/3+5/3+1/3$ oder als

$(3/16+5/16+1/16)/3$ formalisiert werden, arithmetisch ist das Ergebnis immer 3. Im realen Leben besteht allerdings zwischen drei unbestimmten Früchten, ihrer gleichmäßigen Aufteilung und Obstsalat ein deutlicher Unterschied.)

Eng verbunden mit der Formalisierung ist die Modellierung. Bei der Modellierung geht es um die *angemessene* Darstellung als solche. Sie steht im Vordergrund und nicht eine formale Darstellung, obwohl häufig formale Methoden verwendet werden. Bei der Modellierung müssen auf jeden Fall die generellen Merkmale von Modellen überhaupt berücksichtigt werden: Modelle sind *Repräsentationen*, sie stellen nur *ausgewählte Aspekte* dessen dar, was sie repräsentieren. Zudem werden diese Aspekte *im Hinblick auf bestimmte Zielgruppen, Zwecke und Absichten* ausgewählt.

Die dritte formale Herausforderung in der Software-Entwicklung heißt Modellierung. Die erzeugten Modelle können unterschiedlich in Darstellung und Formalisierungsgrad sein. So sind Zeichnungen oder sprachliche Beschreibungen („Martin hat drei Äpfel, ...“, vgl. o.) häufig noch recht anschaulich, Business-Prozess-Modelle oder Entity-Relationship-Diagramme schon weniger. Modellieren erfordert ganz offensichtlich indeterminiert-schöpferisches Handeln: Nicht ohne Grund hat sich für diese Tätigkeit ein Begriff durchgesetzt, der aus der Kunst stammt. Indeterminiert-schöpferisches Handeln umfasst insbesondere die Auswahl der darzustellenden Aspekte und das Darstellen selbst, möglicherweise auch die Auswahl der Modellierungsmethode und das Verfahren, die Schritte der Modellierung selbst. („Verteilen Sie das Obst gleichmäßig“ ist also offensichtlich eine zumindest unvollständige Modellierung, da wichtige Aspekte fehlen.)

Die letzte hier zu berücksichtigende Phase der Software-Entwicklung ist das Erheben der Anforderungen (auch: „Requirements Engineering“). Sie erfolgt häufig gleichzeitig mit dem Modellieren, aber aus einer anderen Perspektive. Denn Ziel dieser Phase ist das Wissen darum, welche Aspekte der Realität und der zu bewältigenden Aufgabe für die zu entwickelnde Software relevant sind. Hierbei handelt es sich nicht nur um eine *formale* Herausforderung. Sicherlich müssen alle erhobenen Anforderungen letztlich modellierbar, formalisierbar und implementierbar sein. Andererseits bedeutet Requirements Engineering in nicht unerheblichem Maße, Realität, auch die Realitäten anderer Personen, zu analysieren (hier ist auch die Frage zu stellen, ob die Kinder Obstsalat möchten oder lieber Obst gegen Bonbons tauschen ...) und (damit) zu interpretieren. Dies wiederum ist unbestritten eine schöpferische Tätigkeit, mindestens mit dem Übersetzen, eher sogar mit Malerei oder Fotografie vergleichbar. Darüber hinaus kann Requirements Engineering – je nach Aufgabenstellung und Umfeld – ein hohes Maß „sozialer Kompetenz“ erfordern, was häufig nur ein anderer Ausdruck für die (kreative) Gestaltung von Kommunikations- und Interaktionsprozessen ist.

Offensichtlich verbinden sich also in der Software-Entwicklung Rationalität und Kreativität, Berechnung und Schöpfertum, Determinismus und Indeterminismus. Das Ergebnis, eine implementierte Software oder ein informationstechnisches System ist ohne Frage hoch formal. Im Prozess der dort hinführt müssen die Entwickler aber ebenso unbestreitbar auf unterschiedliche Weise Neues schaffen, sei es im Sinne relativer oder radikaler Kreativität. Dies bestätigen sowohl wissenschaftliche Untersuchungen und einschlägige Erfahrungsberichte: Von „Programmieren ist doch eine Kunst“ über „experimentelle Codeoptimierung“ bis zur „making creativity happen“ reichen hier die Stichworte. Kurz: Formalisieren ist eine kreative Tätigkeit! Die Wahrnehmung und Darstellung dieser Kreativität erfordert allerdings ein angemessenes Handlungsmodell, welches nicht – wie viele zweckrationale oder gar technokratische Ansätze – das Sein durch ein Sollen verdeckt. Ein solches Handlungsmodell muss zunächst Situationen als „Horizonte von Möglichkeiten“ und einen weiten, Denken einschließenden, Handlungsbegriff zugrunde legen. Dann lässt sich (auch) exploratorisches, interaktives wie indeterminiert-schöpferisches Handeln und sein „kreisen“ oder „Schleifen ziehen“ um Aufgaben und deren Lösungen erfassen. Ja mehr noch: Handeln und Lernen, Kreativität und Rationalität, Erfahrung und Wissen werden in einem Modell miteinander so verbunden, dass sie sich gegenseitig ergänzen.