„Design“

Rund um den Computer heißt ja inzwischen fast alles Design – Hardware-Design, Software-Design, objektorientiertes Design, Datenbank-Design, User Interface Design, VLSI-Design, Gehäuse-Design – , und auf den ersten Blick sieht es so aus, als wäre alles, was diese so verschiedenen Tätigkeiten gemeinsam haben, der Begriff. Tatsächlich aber ist das kein Zufall, es gibt grundlegende Gemeinsamkeiten. Ich werde im folgenden auf diese Gemeinsamkeiten eingehen und versuchen zu erklären, welche grundlegenden Eigenschaften allen Design-Problemen gemeinsam ist.

Begriffsbestimmung

Prinzipiell steht das Wort „Design“ ja für zwei Dinge, den Prozess, also Design als Tätigkeit, und das Ergebnis, also Design als Eigenschaft eines Produktes.

Design als Tätigkeit ist nicht nur die Verbesserung bestehender oder zukünftiger Systeme oder Möglichkeiten, sondern auch die Veränderung der Zukunft. Sprechendes Beispiel dafür sind die zahlreichen Design-Probleme, die nur aufgrund früheren Designs überhaupt erst auftreten: Vandalismus, verstopfte Straßen und Umweltverschmutzung sind nicht nur gesellschaftliche, sondern in konkreten Projekten auch Design-Probleme, die durch die Arbeit früherer Designer, durch Design-Entscheidungen, erzeugt wurden, und mit denen sich Designer heute auseinandersetzen müssen. Im Computer finden wir zahlreiche solcher Beispiele, vom Von-Neumann-Flaschenhals über das Y2k-Problem bis zu Form und Anordnung von „Ok“ und „Cancel“-Buttons in Dialogen, wo frühere Designer Entscheidungen getroffen haben, die die Möglichkeiten heutiger Entwickler entscheidend beeinflussen.

Tatsächlich ist Design aber nicht nur das Lösen, sondern genausosehr auch das Auffinden von Problemen. Die Zahl und Art der Probleme, die in einer Designaufgabe gefunden werden, sind keine Funktion des Ansatzes, der Ideologie oder des Denkmodells des Designers und schon gar keine Funktion der „objektiven Problemlage“, sondern vor allem eine Funktion der Zeit. Je länger ich suchen kann, um so mehr Probleme werde ich finden, die es durch das Design zu lösen gilt. Die grundlegende Einschränkung dabei ist lediglich, was alles in Frage gestellt werden darf.

Ein sehr sprechendes Beispiel dafür ist die in der Literatur immer wieder zitierte Büro-Türschnalle. Ein (fiktiver) Design-Student wird mit dem Design einer neuen Türschnalle für ein Büro beauftragt. Er beginnt eine ausführliche Recherche über dieses Büro, um die Probleme mit der jetzigen Lösung zu finden. Er erkennt, dass das Problem nicht im Design der Türschnalle, sondern prinzipiell im Vorhandensein von Türschnallen liegt, also entwirft er selbstöffnende Türen. Schon nach kurzer Zeit packen ihn Zweifel, und die Lösung scheint in der Vermeidung von Türen im Büro zu liegen, und er entwickelt eine neue Organisationsform türloser Büros. Konsequent weitergedacht, wird der Student bei gesellschaftlichen Utopien der Arbeitswelt enden, und seine Lösung wird zwar umfassend, aber (leider?) nicht implementierbar sein, weil gesellschaftliche Rahmenbedingungen nicht zur Disposition stehen.

In einem vor kurzem am Institut durchgeführten Projekt, einer Designstudie für Selbstbedienungsgeräte im Bankbereich, stießen wir an eine ähnliche Grenze: hier waren die Überlegungen so weit gegangen, dass wir konsequenterweise ein Redesign der Datenbank-Strukturen fordern mussten, um die von uns entwickelten Benutzermodelle umsetzen zu können. Mit dieser Forderung stießen wir auf Unverständnis, schließlich waren wir die „Designer“, die sich mit dem Aussehen beschäftigen sollten. Das Datenbank-Design war nach der Wahrnehmung der Techniker aus „objektiven“ Anforderungen abgeleitet, und daher unantastbar. Dass auch für die Datenbank viele unterschiedliche, funktionierende, an verschiedene Randbedingungen angepasste Designs existieren, war für sie nicht nachvollziehbar.

Man kennt diese Eigenschaft von Design-Problemen ja auch aus dem Software-Design: Jedes Problem hat im Prinzip unendlich viele Lösungen, von denen einige nicht einmal das Schreiben von Software beinhalten. (Es ist eine sehr seltene Tugend von Software-Entwicklern, wenn sie auch diese Lösungen in Betracht ziehen können).

Eigenschaften von Design-Problemen

Design ist also eine Aufgabe, deren Charakter weniger dem des Puzzlelegens entspricht, wo ein definiertes Ziel erreicht werden soll, und das Erreichen dieses Ziels auch klar festgestellt werden kann – das Puzzle ist fertig. „Fertig“ ist in Bezug auf Design eine subjektive Feststellung (auch das ist eine Eigenschaft, die wir von der Software-Entwicklung kennen).

Eine passendere Analogie für Design ist das Malen eines Bildes (aus dem andere ja dann vielleicht ein Puzzle machen). Und genauso, wie jeder Maler ein anderes Bild malt, wird jede Designerin ein vorgegebenes Design-Problem anders lösen. Und „fertig“ ist, wie jeder Maler bestätigen wird, kein objektiv eintretender Zustand, sondern eine Frage des Moments. (Im Design ist „fertig“ meist eine Frage äußerer Randbedingungen, also das Ende von Zeit oder Geld).

Ein leicht nachvollziehbarerer, weil selbsterfahrener Vergleich für diese Eigenschaft von Design-Problemen ist das Schreiben einer Deutsch-Schularbeit (Österreich: Bezeichnung für Klausur in der Schule), also die Erstellung eines themenbezogenen Textes unter engen zeitlichen Randbedingungen. Wie wir uns alle erin-

nern, ist die Fertigstellung hier immer nur durch das Ende der zur Verfügung stehenden Zeit bedingt, und noch Stunden später tauchen „bessere“ Formulierungen, neue Gedankengänge, wichtige Ergänzungen auf. (Auch dieser Text wird nicht „fertig“ sein, wenn ich ihn an Uwe Pirr schicke, weder in meiner Wahrnehmung noch in der der LeserInnen. Viele Fragen werden offen bleiben, wie jeder Text viele Fragen offen lässt.)

In diesem Sinne gibt es auch keine „richtige“ Lösung, lediglich bessere und schlechtere Lösungen im Sinne jeweils nur einer Sichtweise. Genau diese Eigenschaft von Design-Problemen ist es auch, die TechnikerInnen und DesignerInnen vereint, in ein gemeinsames Boot setzt. In diesem Boot wollen TechnikerInnen nicht sitzen, sie fühlen sich unwohl. Schließlich ergibt sich nach Ansicht der Technikwissenschaften die Problemlösung aus der methodischen Erfassung der Requirements und Constraints, denn Design ist lediglich eine Umformulierung des Problems. Genau diese Vorstellung ist jedoch falsch, nicht nur deshalb, weil Design-Probleme auf praktisch unendlich vielen Wegen gelöst werden können (Anm: das bedeutet natürlich, dass bei der methodischen Erfassung und Lösung von Problemen diese Vielzahl durch die in Methodenform gegossenen Grundannahmen und Wertvorstellungen gekontert wird: nur solche Lösungen, die im methodischen Spektrum entstehen können, kommen dann noch in Frage – ein höchst zweifelhafter Ansatz!), sondern auch, weil Requirements und Constraints gar nicht vollständig erfasst werden können. In der Praxis zeigt sich das so, dass unwesentlich erscheinende (weil nicht formal oder methodisch erfassbare) Randbedingungen einfach ignoriert werden, das Problem wird auf seine formalisierbaren Komponenten reduziert (wiederum entsprechend der Grundannahmen und Wertvorstellungen der Technikwissenschaften).

Diese Fehlannahme von TechnikerInnen über die Natur von Design-Problemen, also der Glaube an die mögliche Vollständigkeit der Problemdefinition, lässt sich beispielsweise in der Geschichte des „methodischen Konstruierens“ nachzeichnen.

In den 70er und 80er Jahren wurden in der BRD wissenschaftliche Konstruktions- und Entwicklungsmethoden (vor allem in der Maschinentechnik) propagiert, die zusammenfassend als „methodisches Konstruieren“ bezeichnet wurden. Interessanterweise finden diese Methoden in der Praxis kaum Verwendung, es werden „aus dem Gesamtprogramm meist nur einzelne geeignet erscheinende Arbeitsschritte und Methoden herausgelöst“. In einer Studie des Forschungszentrums Arbeit und Technik werden als Gründe dafür u.a. angeführt, dass „der Leitgedanke des vollständigen Konstruktionsalgorithmus als Rationalisierungsziel ein Irrweg sei, da große Teile der Problemdefinition erst während des Konstruktionsprozesses vorgenommen würden“. Auch würden im Bestreben „einer Rationalisierung oder gar Algorithmisierung bzw. Automatisierung des Konstruktionsprozesses dessen soziale und psychologische Bedingungen nicht oder zu wenig berücksichtigt“. (Alle Zitate aus: „Leitbilder und historisch-gesellschaftlicher Kontext der frühen wissenschaftlichen Konstruktionsmethodik“, Hans Dieter Hellige, Universität Bremen, artec-Paper Nr. 8, Januar 1991)

In dieser Studie werden also zwei wesentliche Erfahrungen festgehalten: einerseits sind Requirements nicht vollständig erfassbar, schon gar nicht, bevor man mit der Konstruktion/dem Design beginnt; andererseits werden beim Versuch der vollständigen Erfassung von Requirements wesentliche Aspekte vernachlässigt. Es sind genau diese vernachlässigten Aspekte, die sich gegen Projektende „zu Wort melden“, und weil sie unter den Tisch gekehrt wurden, wird der Auftraggeber über das Ergebnis entsetzt sein, wird die implementierte Lösung weder benutzerfreundlich noch praxistauglich sein.

Der Schritt weg vom Wasserfall-Modell der Softwareentwicklung, hin zum evolutionären Entwicklungsmodell des Rapid Prototyping reflektiert diese Erkenntnis, und kann als Versuch gesehen werden, genau diese methodischen Probleme zu lösen. Tatsächlich fühlen sich viele TechnikerInnen mit der unbestimmten Natur des zyklischen Entwicklungsmodells sehr unwohl und versuchen, diese Unsicherheit methodisch zu kompensieren – ein Versuch, der die oben aufgezeigten Probleme oft durch die Hintertür wieder hereinläßt.

Schlussfolgerungen

Die Eigenschaften von Design-Problemen – die Unmöglichkeit der vollständigen Erfassung aller Requirements, das Fehlen eines objektiv feststellbaren Endpunktes, und die prinzipielle Unendlichkeit des Lösungsraumes – sind offenbar allen Bereichen der Informatik gemein, in denen wir von Design sprechen. Design ist also ein offener, unbestimmter Prozess, der sich aufgrund dieser Eigenschaften der Formalisierung und methodisch-algorithmischen Bearbeitung auf für TechnikerInnen unangenehme Weise entzieht. Einer starren Methodik zuviel Platz im Designprozess einzuräumen bedeutet zwangsläufig, wesentliche Aspekte des Problems zu vernachlässigen, so dass auf diesem Wege nur Lösungen geschaffen werden können, die unter sehr beschränkten, wirklichkeitsfremden Bedingungen funktionieren.

Die konstruktive Auseinandersetzung mit Problemen erfordert einen offenen, unbestimmten Prozess, der dem Design-Problem in jeder Phase angepasst werden kann.

Peter Purgathofer*
purg@igw.tuwien.ac.at

* Herr Dr. Purgathofer ist Leiter der Arbeitsgruppe „User Interface Design“ im Institut für Gestaltungs- und Wirkungsforschung an der TU Wien.