Der „Turmbau zu Babel" als Leitmotiv: Ein Rückblick auf Leben und Werk Heinz Zemaneks

Hans Dieter Hellige

Der „Turmbau zu Babel" ist seit den 1950er-Jahren ein Leitmotiv in der Computer-Community, Informatik und Informationstechnik. Vorausgegangen war jedoch eine lange Vorgeschichte, die bis zu den frühneuzeitlichen Ideen und Plänen für eine Universalsprache auf logischer bzw. philosophischer Grundlage von Bacon, Descartes und Leibniz zurückreicht. Diese begründeten eine Tradition konstruierter Idealsprachen, die die zu Missverständnissen führende Vielgestaltigkeit und Vieldeutigkeit natürlicher Sprachen auf einen rationalen Wesenskern zurückzuführen hofften, um so die aus dem Turmbau erfolgte Sprachverwirrung wieder rückgängig zu machen. Durch eine große aufklärerische Anstrengung wollte man die Folgen des Turmbaus ungeschehen machen, indem Logik, Mathematik und rationale Philosophie durch Formalisierung eine künstliche „langue universelle" erschafften und damit auf einem höheren Niveau den Idealzustand der ursprünglichen Spracheinheit erneuerten [7, Kap. 14 und 15]. Die Programmatik der „Unity through Formalization" bestimmte den Gebrauch der Leitmetapher der babylonischen Sprachverwirrung in den Kalkülwissenschaften und insbesondere der Symbolischen Logik bis weit in das 20. Jahrhundert hinein.[1]

Die Anfänge des Turmbau-zu-Babel-Motivs im Computing und der Computer-Science standen, wie es Zuse in Deutschland und Weaver in den USA belegen, in der Tradition der rationalistischen Gesamtkonzepte der Symbolischen Logik von Leibniz bis zum Wiener Kreis. So hielt Konrad Zuse, an die Leibniz-Renaissance und den Logischen Positivismus anknüpfend, an dem Ziel einer „wirklich universellen, allgemein arithmetischen Formelsprache" fest, die gleichermaßen für mathematische, logische und symbolische Rechnungen jeder Art geeignet ist. Den von ihm ab 1939 konzipierten und 1945/46 ausgearbeiteten „Plankalkül" sah er als eine solche ideale „Bezugssprache zwischen verschiedenen Systemen" und als „universelle Sprache" für „beliebige schematisch-kombinative Aufgaben" ([42, S. 226, 228, 230], [41]). Wie Zuses logisches Sprachkonzept gingen auch die frühesten Bestrebungen der amerikanischen Computing-Community für eine formalisierte wissenschaftliche Universalsprache wesentlich auf Einflüsse des Wiener Kreises zurück. Hier war es Warren Weaver, der seit 1944/46 das Großprojekt einer universellen logikbasierten Wissenschaftssprache anstieß, die den voneinander isolierten „wissenschaftlichen Turmbauten" als gemeinsames Fundament und als Verständigungsebene dienen sollte. Er hoffte so, die spezialisierten Wissenschaften wieder theoretisch-methodisch einander anzunähern und das so wiedervereinigte Wissen der Welt in einer positivistischen Universalenzyklopädie zusammenzuführen. Denn die „confusion in human language" sei daran Schuld, „[that] the tower of science cannot grow into heaven" [19, zuerst publiziert in 12, S. 15–23].

[1] Die Formel „Unity through Formalization" stammt aus einem Memorandum des stark von Rudolf Carnap und Alfred Tarski beeinflussten theoretischen Biologen Joseph Henry Woodger, siehe [13, S. 164 f.].

DOI 10.1007/s00287-014-0841-1

Hans Dieter Hellige
Forschungszentrum artec, Universität Bremen, Bremen
E-Mail: hellige@uni-bremen.de

Die Metapher diente nach diesem vorherrschenden Verständnis nicht als ein Negativleitbild, das warnen und motivieren sollte, den unvermeidlichen Problemen und Grenzen komplexer Systementwicklungen mit vorsorgender Planung, Organisation und Bescheidenheit zu begegnen. Einzig Charles Babbage wurde sich bei seiner Beschäftigung mit „symbolic languages“ und „symbolic reasoning“ in der mechanischen Konstruktion sowie vor allem in der „analytical science“ und Mathematik bewusst, dass auch die exakten Wissenschaften nicht vor einem Wildwuchs bei den Notationssystemen gefeit sind. Denn auch hier würden ständig neue Zeichensymbole eingeführt, die „merely new ways of expressing well-known functions“ darstellten. Das veranlasste ihn zu dem warnenden Appell: „Unless some philosophical principles are generally admitted as the basis of all notation, there appears a great probability of introducing the confusion of Babel into the most accurate of all languages“ ([2, S. 140], [1, S. 409–424]). Doch dieses erste Beispiel für die Verwendung des Turmbau-Motivs als Mahnung an Wissenschaftler und Ingenieure stieß kaum auf Resonanz.

So dient der biblische Mythos erst seit der Mitte der 1950er-Jahre als ein warnendes Negativleitbild für das Scheitern überdimensionierter oder architektonisch inkonsistenter Systeme, für Kommunikationsprobleme aufgrund fehlender Standardisierung sowie generell für fundamentale Koordinations- und Organisationsmängel bei der Entwicklung von Informationssystemen. Die Spannweite von Rekursen auf den Turmbau zu Babel reicht vom bloßen Topos und von bildträchtiger Metaphorik bis zu systematischen Betrachtungen über architektonischen Wildwuchs, Standardisierungsdefizite und organisatorische Fehlplanungen. Zur Leitikone wurde dabei der „große Turmbau“ von Pieter Breughel d. Älteren von 1563, der wohl am überzeugendsten und ästhetisch gelungensten die verschiedenen Einzelmotive zu einer einzigen Bilderzählung integrierte. Gerade die symbolische Verdichtung des Zusammenhangs von Sprachverwirrung, Planungsdefiziten, mangelhafter Prozessorganisation und verletzlicher Systemarchitektur machte diese Bilderzählung so attraktiv für die Visualisierung von Planungs-, Strukturierungs- und Organisationsproblemen in der Technik und im Management. Breughels „Wiener Turmbau“ wurde so auch in

Abb. 1 Heinz Zemanek mit seinem Entwicklungsteam im Wiener IBM-Laboratorium (Foto aus dem Bildarchiv Heinz Zemaneks)

der Computer-Community, der Informatik und der Informationstechnik die Standardbildmetapher, wobei Heinz Zemanek eine Schlüsselrolle bei der Rezeption des Turmbau-Mythos in diesem Bereich einnahm.

Denn Zemanek erkannte wohl als erster die Aktualität und den didaktischen Wert dieses Gemäldes: „Aus dem Wissen um Breughels Werk geht hervor, daß man nur Breughels Gedanken zu erraten und in die heutige Zeit und auf das Computerfeld übertragen muß, um Hinweise auf die Computer-Architektur zu erhalten. ... Hat Breughel den Zustand der heutigen Programmierung geahnt, als er dieses Symbolbild schuf? Das ist nicht nötig, denn die Denksituation für menschliche Großunternehmungen ist zu allen Zeiten die gleiche; die technische Erscheinung ist nur ausgewechselte Bekleidung“ [39]. Es gibt wohl keinen Vertreter der informatischen Zunft, der das Leitmotiv so häufig in seinen Schriften und Vorträgen verwendete und es zum Gegenstand systematischer Erörterungen, ja geradezu zum Leitgedanken seines Lebens gemacht hat. Er gewann dadurch eine besonders komplexe Sicht auf die didaktische Parabel, denn als Ingenieur erkannte er im Turmbau das Architekturversagen, als Logiker die Sprachverwirrung und als Wissenschafts- und Technikkritiker das Organisations- und Gesellschaftsversagen. Das Leitmotiv soll deshalb auch in das Zentrum des folgenden Rückblicks auf Zemaneks Werk gestellt werden.

Die Verknüpfung mit dem Computerbau verdankt es einem Aha-Erlebnis Zemaneks, das er immer wieder geschildert hat. Bei der Besichti-

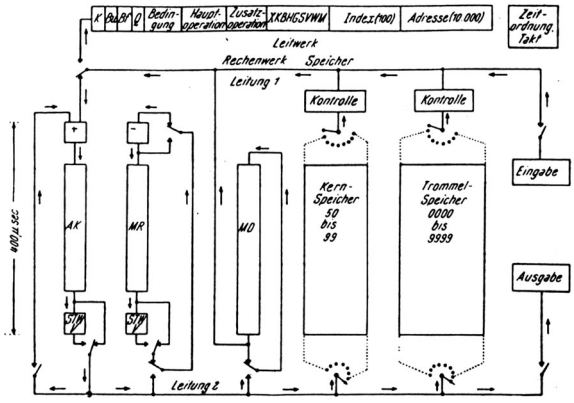


Abb. 2 Organisationsschema des Minima-Rechenautomaten „Mailüfterl" (Abbildung aus [23, S. 457])

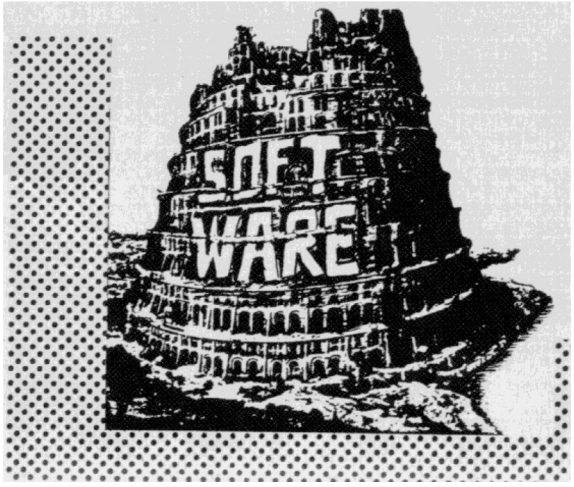


Abb. 3 Vignette im Computer-Magazin 10/1985

gung des Göttinger Rechenautomaten G2 von Heinz Billing im März 1956 erinnerte ihn eine auf einem angeschraubten Metallbalken nachträglich angebrachte Röhrenreihe an die Armeleutehäuser auf dem Turmbaugemälde. Spontan entschied er sich noch im selben Jahr, das Breughel-Bild als Warnzeichen vor architektonischen Fehlentscheidungen beim eigenen Rechnerbau im Wiener Labor aufzuhängen. Es sollte das Entwicklungsteam dazu anhalten, konsequent ein Abgleiten in architektonischen Wildwuchs zu vermeiden und klare Hardwarestrukturen mit möglichst wenigen elektronischen Komponenten anzustreben. Auch bei der Software sollten die Befehlssätze überschaubar und aus wenigen voneinander unabhängigen Befehlselementen kombiniert werden. „Der Gefahr, in dieser Vielzahl der Möglichkeiten die Übersicht zu verlieren und die Programmierung zu einer nur Spezialisten zugänglichen wissenschaftlichen Arbeit zu machen", wollte man durch geeignete Übersetzungsprogramme begegnen. Das „Mailüfterl" war somit als ein minimalistischer Gegenpol zum Großcomputer „Whirlwind" wie zum Turm zu Babel konzipiert ([39, S. 161], [24, S. 39 f.], [34, S. 104 ff.]).

Bei der Hardware gelang dies auch vorbildlich, wie es die besonders klaren, an van der Poels und Frommes Minima-Architektur angelehnten logischen Strukturen zeigen [24, S. 37–49]. Bei der Software dagegen kam erst während des Baus die Idee auf, die Maschine sowohl dezimal als auch binär zu betreiben, was allerdings durch die Adressierung des Magnettrommelspeichers nicht unterstützt wurde. Auch vergaß man die minimalistischen Vorsätze und entwickelte einen „einzigartigen Befehlsreichtum" mit einer „unglaublichen Anzahl von Befehlsmöglichkeiten", worauf man sehr stolz war. Erst später realisierte das Entwicklerteam, dass eine 10-mal so schnelle Maschine bei Beschränkung auf die klassischen 32 Befehle und folglich besserer Überschaubarkeit der Programmierung zum gleichem Resultat geführt hätte ([35, S. 149], [33, 40, S. 38 f.], http://conservancy.umn.edu/handle/107723).

Zemanek erkannte, dass die eigentlichen Probleme bei der Software und vor allem bei den Anwendungen lagen. Der Deutungsrahmen für die Turmbau-Erzählung erweiterte sich von der Diskrepanz zwischen elegantem Entwurf und architektonischen Unzulänglichkeiten zu einer allgemeinen Mahnung vor der Verfolgung zu ambitionierter Ziele, insbesondere bei der kybernetischen „Imitation menschlicher Kräfte" im Rechner. So schloss er bei der ersten Verwendung des Turmbau-Motivs in einer Publikation im Jahre 1958 die Aufzählung der Anwendungsziele für das Mailüfterl mit den Sätzen: „Die Nachbildung verschiedenster menschlicher Tätigkeiten durch eine Programmsteuerung bietet allgemein eine Fülle von Möglichkeiten. Bis zur Realisierung der vorliegenden Pläne wird viel Zeit vergehen. Das rechte Maß für die menschliche Unternehmung erfordert dauernde Auseinandersetzung; im Raum des Automaten erinnert eine Reproduktion von Pieter Breughels ‚Turmbau zu Babel' daran, daß sich zwischen jedes Projekt und seine Durchführung die menschliche Unzulänglichkeit schiebt" [23, S. 463]. Mit Zemaneks neuem Tätigkeitsschwerpunkt am Wiener IBM-Laboratorium in der Zeit

zwischen 1961 und 1976 trat dann bei den Turmbau-Interpretationen das Motiv der babylonischen Sprachverwirrung in den Vordergrund, wobei er nun auch auf den Tower-of-Babel-Metaphern-Diskurs der Programmiersprachen-Community anknüpfen konnte.

Der früheste Beleg für das Babel-Motiv ist hier eine Präsentation der Programmiererin im Department of Defense Betty Jo Ellis beim „6th Annual Symposium on Computers and Data Processing" in Estes Park, Colorado, im Juli 1959. Als Überschrift für ihre Betrachtung über den zunehmenden Wildwuchs im „jargon of computers and programming" wählte sie den Vers Genesis 11,9: „Therefore is the Name of it called Babel; because the Lord did there confound the language of all the earth".[2] Doch dieses Paper fand offenbar keinerlei Resonanz, sodass es erst kürzlich wiederentdeckt wurde. Auch das berühmte „Babel 1960"-Titelbild im Januar-Heft 1961 der „Communications of the ACM" ist noch kein Beleg für eine allgemeine Verbreitung des Leitmotivs, es erlangte seine große Beachtung erst nach einer Reihe von Jahren. Die Verknüpfung des Inhaltes des Heftes, der Paper des „ACM Compiler Symposium" vom November 1960, zum Tower-of-Babel-Motiv hatte sich nicht einmal aus den Konferenzbeiträgen ergeben, sondern ging allein auf den Illustrator zurück. Es häuften sich zwar schon seit den 1950er-Jahren die Klagen über das durch die Proliferation der Maschinen ausgelöste Auseinanderfallen der Programmiersprachen, ohne dass dabei jedoch auf die biblische Sprachverwirrung Bezug genommen wurde. Das gilt selbst für den bekannten „Historical Survey" über Programmiersysteme und -sprachen von Saul Rosen von 1964, der die entstandene Vielfalt an Sprachkonzepten beschrieb und sogar schon am Beispiel der ALGOL-Entwicklung die Dialektik von Standardisierungsbemühungen erkannte: „The initial result of this first attempt at the standardization of Algebraic languages was the proliferation of such languages in great variety" [15, S. 7]. Die Verbindung der Proliferationsdebatte mit dem Sprachenturm-Motiv kam erst in der zweiten Hälfte der 1960er-Jahre zustande, nicht zuletzt durch das wegweisende Werk und die Aufsätze von Jean Sammet [16, 18].

[2] Die Präsentation ist von Nathan Ensmenger in den Betty Jo Ellis Papers im Charles Babbage Institute entdeckt worden, siehe seine Blog-Eintragung vom 12.9.2013 (http://thecomputerboys.com/?cat=15).

Auch Zemanek wurde durch seine Beschäftigung mit der ALGOL-Implementierung und der Formalen Definition der Programming Language I (PL/I) ständig mit der eskalierenden Vermehrung von Programmiersprachen und vergeblichen Versuchen ihrer Eindämmung konfrontiert. Er hatte sich in seiner „Kybernetischen Periode" seit dem Ende der 1950er-Jahre intensiv mit Automatensystemen, abstrakten Strukturen und formalen Sprachen beschäftigt und hoffte nun, noch ganz unter dem Einfluss des wittgensteinschen „Tractatus" und der „naiven Periode des Computerpositivismus", dass durch die „clarifying power of logic and transparent models" auch das Problem der automatischen Übersetzung und der babylonischen Sprachverwirrung bei Programmiersprachen gelöst werden könne [25, 36, bearb. Fassung in 14, S. 237–302, bes. S. 276]. Das Schicksal der Proliferation hatte auch die PL/I erfahren, obwohl sie 1963/64 von IBM in Verbindung mit der Nutzervereinigung SHARE als eine „universal language" für alle Arten von Programmierern und Anwendungsgebieten entwickelt worden war. Sie sollte eine Synthese aus Fortran, ALGOL und Cobol darstellen, die die Errungenschaften der Sprachen der ersten Generation vereinigt, diese dadurch ablöst und so analog zur Computerfamilie System/360 und zum Betriebssystem OS/360 die Systemvielfalt bei Programmiersprachen ein für alle Mal überwindet. Doch das Resultat war eine typische Komitee-Entwicklung, ein Kompromiss, der keine Nutzergruppe richtig zufrieden stellte und sich daher schnell in eine Reihe von Dialekten aufspaltete. Die PL/I wurde so wie auch ALGOL 68 ein typisches Beispiel für die von Frederick Brooks „Second-System-Effect" genannte Fehlerkonstellation, bei der die Kumulierung aller Anforderungen mit dem Ziel von „greater richness and power of expression led to greater complexity both in language and in language use" ([6, Kap. 5, bes. S. 64 ff.], [21, S. 1211]).

Diesem wechselseitigen Hochschaukeln von Proliferation und Einheitsbestrebungen hoffte Heinz Zemanek nun mithilfe einer einheitlichen Metasprache zu begegnen. Im Jahre 1964 überzeugte er das IBM-Management, dass die Integration der unterschiedlichen Programmiersprachen-Kulturen in der NPL bzw. PL/I nur durch Formalisierung zu lösen war, durch eine axiomatische Programmiersprache für die Definition von Programmiersprachen: „a formal definition was the only way to survive"

Abb. 4 Logo des Wiener IBM-Laboratoriums VDL-Dokumente vor dem Breughel-Bild bei der Wiener IFIP Working Conference on Formal Language Description Languages, Vienna, September 1964 (Foto aus dem Bildarchiv Heinz Zemaneks)

[33, S. 43 ff.]. Von 1964 bis 1968 wurde unter seiner Leitung, anknüpfend an John McCarthy und Peter Landin, eine formale Definition der PL/I erarbeitet, die auf der Grundlage der Prädikatenlogik die abstrakte Syntax einer Sprache und eine „Abstract Interpreting Machine“ entwickelte. Diese transformiert konkrete Programme mithilfe eines Syntaxzerlegers und Übersetzers in ein abstraktes Programm, das die Sprachkonstrukte durch Zustände und Zustandswechsel beschreibt, transparent und vergleichbar macht und so die sprachlichen Benennungen und Definitionen eindeutig macht [4, 34, S. 81–86]. Die dabei entwickelten Methoden und formalen Spezifikationen wurden dann 1969/70 zu einer „Formal Definition Technology“ verallgemeinert und zur Systemdefinitionsmethode Vienna Development Method (VDM) systematisiert.

VDL und VDM wurden zwar eine dauerhafte Bereicherung des Instrumentariums der Informatik, doch ihr eigentliches Ziel erreichten sie nicht. Der Weg eines „Universal Language Document“, wie die „Vienna Definition Language“ ursprünglich heißen sollte, löste nicht das Proliferationsproblem. Formalisierung und Modellierung von Programmiersprachen schufen zwar höhere Klarheit, deckten Konstruktionsfehler auf, doch die Rechnermodelle der Sprachen näherten sich dem Komplexitätsgrad des Originals oder übertrafen es sogar noch. Obwohl die VDL unter vergleichbaren Ansätzen noch die praktikabelsten Sprachdefinitionsmechanismen aufwies, wurde sie mit schwer lesbaren 1500 Seiten Umfang, wie Jean Sammet resümiert, „too difficult and impractical for compiler writers to use in their development work“ [17, 21, S. 1215]. Da der Aufwand einer Bereinigung der PL/I und noch mehr für die Schulung der Programmierer und letztlich auch der User bei ständig fortschreitender Technik gewaltig gewesen wäre, verzichtete die IBM, wie Zemanek resigniert feststellte, der Verwirrung Zügel anzulegen und die Chance einer Vereinheitlichung der Programmiersprache, der Systemsprache und der Metasprache zu ergreifen: „Die Formale Definition wäre der rechte Weg gewesen, aber dieser Weg war nicht gangbar. Die geballt logischen Geräte und Systeme der Informationstechnik müssen auf intuitiv-pragmatische Weise eingesetzt werden, mit unvollständiger Kenntnis und ohne genaue Anpassung an die Anwendungssituation“ [34, S. 86].

So wurden die langjährigen Arbeiten an der „Giant Formal Architecture“ am Ende selber zu einem unabgeschlossenen „Tower of Babel“, vor dem die Turmbau-Mahnung auf dem Briefbogen einer von ihm 1964 einberufenen IFIP-Konferenz und das Breughel-Bild im Meetingroom doch bewahren sollten [37, S. 251–270; zit. nach der rev. Fassung in 14, Kap. 1.4, bes. S. 110, 94 ff., 110 ff.]. Die von Zemanek im Prinzip für möglich gehaltene Einigung auf eine einzige Programmiersprache scheiterte an der

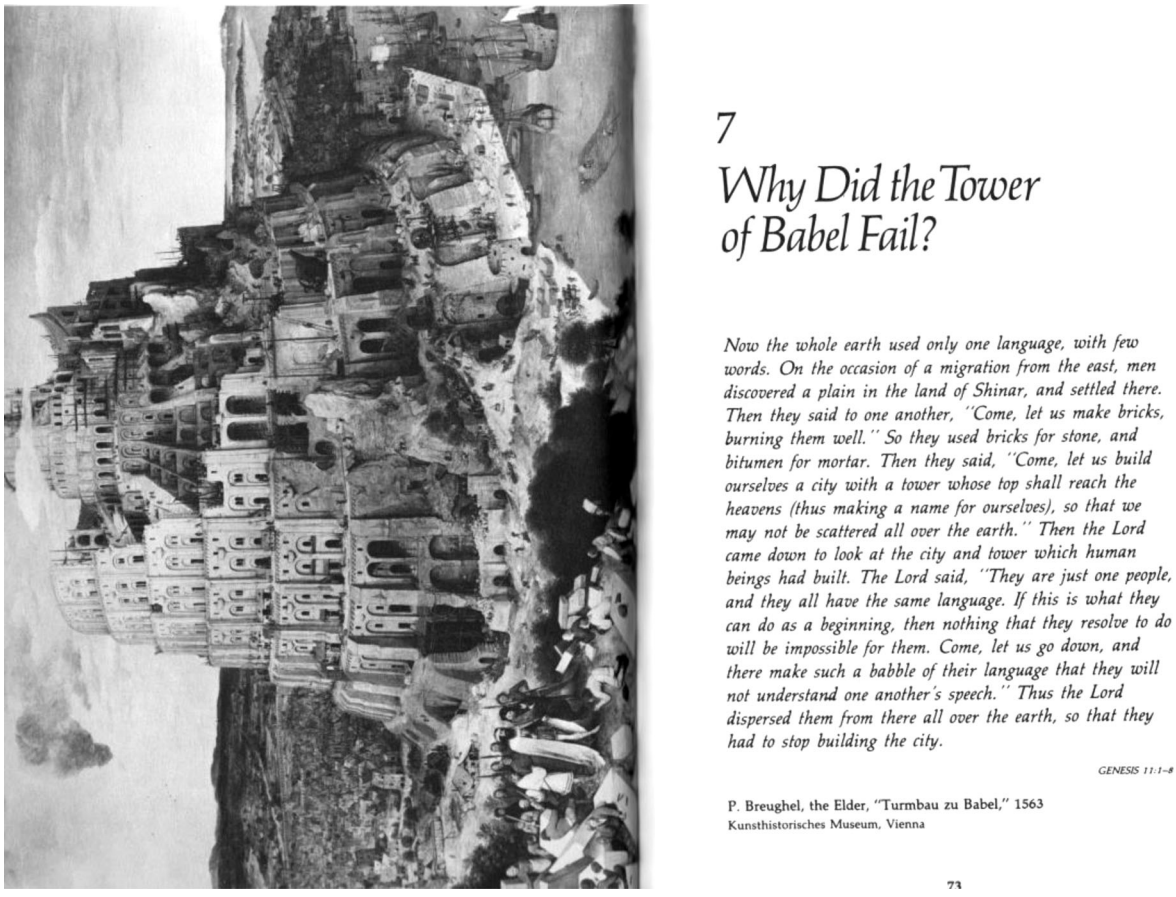
Vielfalt der Benutzerkategorien und -bedürfnisse. So wuchs die Zahl der Programmiersprachen immer weiter, Firmen und selbst Fachleute der zur Überwindung des Sprachenchaos gegründeten IFIP WG 2.1 beteiligten sich an der Schaffung neuer Sprachen „und vergrößerten die Sprachverwirrung ... gegen diese Verwirrung gab es und gibt es kein Mittel. ... Es gibt keinen John von Neumann für all diese Diskrepanzen. Und man kommt auf den Verdacht, daß es für sie einen John von Neumann, einen Ordnung schaffenden Geist, überhaupt nicht geben kann“ [40, S. 14–16].

Daraus resultierte eine mehrfache Ernüchterung: Heinz Zemanek musste, bestärkt durch die nun folgende Beschäftigung mit dem Spätwerk Wittgensteins, erkennen, dass die noch immer erhoffte Einheitlichkeit der Programmierung „ein unerreichbares Ziel“ bleibt, denn auch die Formalisierung von künstlichen Sprachen ist weder lückenlos möglich noch sind die verbleibenden Bedeutungsunklarheiten formal entscheidbar: „Formalization of formalization is impossible. We need informal language to make a formal language understood“ [37, S. 95]. Und hierbei schlichen sich sofort wieder unterschiedliche Verständnisse ein, zumal in dem Großvorhaben analog zum Turmbau sehr viele Programmierer, Mathematiker und Ingenieure mit unterschiedlichen Mentalitäten und differierenden „Sprachspielen“ zusammenkamen, sodass es wie schon bei ALGOL auch bei der VDL zu „different styles of rigorous semantic specification for defining the meanings of syntactically specified constructs“ kam [20].

So verwundert es nicht, dass besonders ausführliche Deutungen der Turmbau-Erzählung und des Breughel-Bildes im Kontext der nachträglichen Aufarbeitung des erfolgreichen Scheiterns seiner „Formalen Periode“ stehen. Über sie formulierte er die Erkenntnis, dass sich die wildwüchsige Sprachvermehrung auch über die Metaebene nicht zurückdrängen ließ, da die Sprachverwirrung aus der Vielfalt der Akteure, der Heterogenität ihrer Mentalitäten und der Komplexität einer sich stets wandelnden Realität resultierte, die sich alle zusammen der sauberen Lösung der Formalisierung widersetzten: „Der Traum von der Perfektion bleibt ein Traum, er wird nie Wirklichkeit werden“ ([37, S. 95], [28, 38], [33, S. 45 ff.]). Man komme somit nicht umhin, die „well-defined grounds“ der formalen Definitionen zu verlassen und sich den „undefined grounds“ zu stellen, denn: „Die Welt richtet sich nicht nach dem Programmierungssystem, sondern das Programmierungssystem muß der Welt dienen. Die Perfektionierung der Programmiersprachen war der falsche Weg, die Informatik muß sich um die Imperfektionierung des Rechenmaschinengebrauchs kümmern. Aus dieser Quelle her wird sie zur Ingenieurwissenschaft, zur Kunst des Kompromisses, die den wahren Ingenieur ausmacht“ [26].

Anfang der 1970er-Jahre kam Heinz Zemanek deshalb zu dem Schluss, dass der Konflikt zwischen der „wild technology“ und der „ordered technology“, die dem Turmbau-zu-Babel-Syndrom zugrunde liegt, nur auf der Ebene der Architektur, der Organisation und des Managements zu lösen ist [29, S. 481]. Damit verschob sich der Schwerpunkt seiner Interpretation der Turmbau-Erzählung hin zur Architektur- und Designtheorie. Breughels Bild wurde nun zum Sinnbild des Architektur-, Organisations- und Managementversagens, zur Negativfolie, von der her sich die Kriterien und Strukturen einer „guten Architektur“ bestimmen ließen. Aus diesen Überlegungen heraus entstand ab 1972 Zemaneks Konzept einer Theorie der „Abstrakten Architektur“, einer übergreifenden Struktur- und Designlehre für Computer- und Softwaresysteme, Gebäudearchitekturen sowie Verkehrs- und Versorgungssysteme. Er knüpfte dabei unmittelbar an die Architekturtraktate von Frederick Brooks und Gerrit Blauuw an, insbesondere an deren „key idea“ der „consistency“ als Inbegriff einer „good architecture“ (siehe dazu [33, S. 55 ff.], sowie ausführlich [9, Kap. 4]). In diesem Zusammenhang ermunterte Zemanek Blauuw zu der bekannten ersten Ausformulierung der Qualitätskriterien für Rechnerarchitekturen [5, wiedergedr. in 8, S. 14–35]. War die „generalized architecture“ bei Zemanek anfangs nur als Erweiterung der „formalization“ gedacht, so wurde sie ab 1976 zu seinem dritten Haupttätigkeitsfeld, das er als IBM-Fellow mit nun allerdings deutlich verringertem Mitarbeiterstab über neun Jahre lang bearbeitete.

Den letzten Anstoß für das Thema hat, so lässt sich vermuten, Brooks 1975 erschienenes Meisterwerk „The Mythical Man-Month“ gegeben. Denn auch dieser theoretische Essay über Systemdesign und Projektmanagement verarbeitete eine Turmbau-zu-Babel-Erfahrung im IBM-Konzern, das Desaster des OS/360. Die Vereinheitlichungs-



7

Why Did the Tower of Babel Fail?

Now the whole earth used only one language, with few words. On the occasion of a migration from the east, men discovered a plain in the land of Shinar, and settled there. Then they said to one another, "Come, let us make bricks, burning them well." So they used bricks for stone, and bitumen for mortar. Then they said, "Come, let us build ourselves a city with a tower whose top shall reach the heavens (thus making a name for ourselves), so that we may not be scattered all over the earth." Then the Lord came down to look at the city and tower which human beings had built. The Lord said, "They are just one people, and they all have the same language. If this is what they can do as a beginning, then nothing that they resolve to do will be impossible for them. Come, let us go down, and there make such a babble of their language that they will not understand one another's speech." Thus the Lord dispersed them from there all over the earth, so that they had to stop building the city.

GENESIS 11:1–8

P. Breughel, the Elder, "Turmbau zu Babel," 1563
Kunsthistorisches Museum, Vienna

73

Abb. 5 Das Tower-of-Babel-Leitmotiv am Beginn der Reevaluation des OS/360-Desasters in Frederick P. Brooks „Mythical Man Month" [6, S. 72 f.]

bestrebungen zur Überwindung der Proliferation bei Operating Systems mündeten beim OS/360 wie bei der PL/I und der VDL in einer Komplexitätsfalle. Nach dem Vorbild der erfolgreichen Überwindung der Systemvielfalt der IBM-Computerlinien durch das System/360 sollte auch das Betriebssystemchaos durch ein „single operating system concept" [22] für das gesamte Anwendungsspektrum beseitigt werden, wodurch eines der größten und komplexesten Programme, die je entwickelt wurden, entstand. Doch aufgrund anfangs zu geringer Mittelausstattung und anschließender Organisationsfehler entwickelte sich das ambitiöse Vorhaben zu einem einzigen Debakel.

Bei der Analyse von dessen Ursachen in dem Buch bediente sich Brooks ebenfalls des Vergleiches mit dem „Babel-Project", dem ersten und spektakulärsten „engineering fiasco". Die Anregung hierfür erhielt er von Zemanek, zu dem er seit den 1960er-Jahren in lockerer Verbindung stand und der ihm auch eine Reproduktion des Wiener Breughels für das Buch beschaffte.[3] Brooks legte das Gemälde seinem retrospektiven Managementaudit des Turmbaus zugrunde, wobei er wie Zemanek zu dem Ergebnis kam, dass das Großvorhaben wegen fehlender Kommunikation und Koordination bereits zum Scheitern verurteilt war, bevor es überhaupt an technologische Grenzen stieß. Er deckte dann in dem „Large Programming Project" des OS/360 vergleichbare Kommunikationslücken und Organisationsmängel auf, die auch dort zu Gruppeneifersucht und Streitigkeiten führten: „Schedule disaster, functional misfits, and system bugs all arise because the left hand doesn't know what the right hand is doing" [6, S. 74]. So gelangte die Verknüpfung der Analyse von Architekturdefiziten und

[3] Persönliche Mitteilung von Heinz Zemanek, Februar 2010.

Projektmanagementfehlern mit der Ausdeutung der Turmbau-Erzählung in die viel gelesene „Bible of Software-Engineering“ und wurde dadurch in der Folgezeit zu einem festen Bestandteil von Design- und Managementreflexionen in der „architectural community“.

Doch die IBM-Fehlschläge erwiesen sich nicht nur als ideales Beobachtungsfeld für das Turmbau-zu-Babel-Syndrom, sondern sie wurden auch Ausgangspunkt für die Suche nach neuen Ansätzen zur architektonischen Bewältigung von Systemkomplexität. Daher ist es auch kein Zufall, dass im Zusammenhang mit grundsätzlichen Problemen bei besonders ambitionierten Hardware- und Softwareprojekten der IBM in den 1960er-Jahren die Begriffe „computer *architecture*“ und „software *architecture*“ und in den 1970er-Jahren generalisierte Architektur- und Designtheorien hervorgingen ([9, S. 438–484], [10, S. 15–19], abrufbar unter: http://www.uni-bremen.de/de/artec/mitglieder/prof-dr-hans-dieter-hellige.html, [11]). Mit ihnen sollte auch in Multiakteur-Konstellationen und hyperkomplexen Problemstrukturen die konzeptionelle Integrität gewahrt werden. So entwickelten Brooks und Blauuw ihr normatives Architekturkonzept und Heinz Zemanek schloss sich daran mit seinem Programm einer „Abstrakten Architektur“ an, d. h. einer „verallgemeinerten Entwurfstheorie“, einer gebiets- und disziplinübergreifenden Qualitätsnormenlehre und Gestaltungsmethode, die schließlich zu einem integralen Ansatz der soziotechnischen System- und Organisationsgestaltung ausgebaut werden sollte.

Zemanek stellte seine „Architektonischen Leitideen für Systeme“ in Form einer Reihe von „seminal papers“ vor, die jeweils die Hauptbestandteile umrissen: den verallgemeinerten Architekturbegriff, eine erweiterte Systematik der Qualitätskriterien und ein Phasenmodell der Entwurfsmethodik ([27, S. 1–6], [30, S. 1–42], [31, 32, S. 99–125], [34, 6. Vorlesung]). Bei dem Idealmodell des Designprozesses unterscheidet er die *natürliche Phase*, den naiven ersten Entwurf, und die sich anschließende *kombinatorische oder Baumeisterphase*, in der, wie er es am Beispiel des Turmbaus darstellt, der „Weg von der Universalität zur Spezialität ... in die Verwirrung, in die Differenzierung der Mentalitäten“ führt. Dem soll die „*Phase der Architektur*“ entgegentreten, in der der „Architekt“ als „Fachmann für Gestaltung“ die Einzelheiten aus dem Gesamtkonzept heraus optimiert und den Entwurfsprozess durch sein richtungsweisendes „architectural model“ steuert. Der Architekt muss dabei mit jedem beteiligten Akteur in seiner Sprache reden, er ist, wie Boris Beizer es bündig formuliert hat, „a linguist, a polyglot in a tower of babel“ [3, S. 10 f.]. Die letzte Phase in Zemaneks Entwurfsmethodik sollte dann den Nutzer in das Zentrum stellen und die durch Abstraktion und Formalisierung produzierte Alltagsferne und Unmenschlichkeit wieder zurücknehmen. Allerdings konnte er, wie es den meisten phasenorientierten Designtheorien ergeht, die saubere Trennung der einzelnen Entwurfsschritte am Ende nicht durchhalten.

Überhaupt gelang Zemanek die Zusammenfassung zu einer „geschlossenen Theorie“ nicht mehr, weil er, wie er es 1991 formulierte, „noch nicht weiß, wie man das Thema geschlossen darstellt. Noch habe ich aber die Hoffnung, und vielleicht gelingt es mir, dieses dritte Großkapitel meines Lebens in passender Weise zusammenzufassen. Bisher hat es sich als zu sperrig, als wuchernd, unter meinen Händen und in meinen Überlegungen aus der Fasson wachsend erwiesen, voller Querbeziehungen, die nach weiteren Überlegungen rufen“ [34, S. 125]. Es hat den Anschein, als ob sich die Komplexität, die über den strukturierenden und organisierenden Zugriff der Architektur beherrscht werden sollte, auf der Ebene der Architekturtheorie reproduzierte. Es fehlt an einer Architektur der Architekturtheorie, doch die ist offenbar genauso wenig möglich wie eine Formalisierung der Formalisierung. So wie der Informatiker mit einem Nebeneinander von formaler Ordnung und informeller Unordnung leben muss, so bleibt auch der Entwurfsprozess am Ende für Zemanek weiterhin „ein widersprüchliches Konzept, ein Ringen um Kompromisse aller Art“ [34, S. 133, 163].

Architektur- und Designtheorien werden damit keinesfalls sinnlos, aber die Schlüsselproblematik verschob sich für ihn nun zur Beziehung *zwischen* der Informationstechnik und der Gesellschaft. Demgemäß weitete sich der Interpretationsrahmen für den Babel-Mythos ins Grundsätzliche. Er sah nun verstärkt die Gefahr, dass das reine Zeichengebäude der Informationstechnik mit ihren unglaublichen Möglichkeiten „ins Wilde tobt“, wenn sie nicht den menschlichen Unzulänglichkeiten Rechnung trägt und durch menschliche Ziele begrenzt wird:

„Die heutige technische Welt ist zu einem solchen Turmbau geworden, und die Computerwelt im besonderen fügt einen abstrakten Turm ebenso gewaltiger Dimensionen hinzu“ [34, S. 104]. Folglich widmete sich Heinz Zemanek seit den 1980er-/1990er-Jahren verstärkt den gesellschaftlichen und gesellschaftshistorischen Fragen des Computers und der IT. Man kann ihn deshalb mit gutem Recht als einen Vordenker der Gesellschaftsinformatik bezeichnen, dem auch der GI-Fachbereich Informatik & Gesellschaft so manche Erkenntnis zu verdanken hat.

Überblickt man die Verwendung des Turmbau-Motivs bei Heinz Zemanek, so lässt sich parallel zum Lebenslauf ein signifikanter Wandel der Interpretationsschwerpunkte feststellen. Er nutzt die biblische Geschichte und die breughelsche Bilderzählung jeweils wie ein Gleichnis, das situationsbezogen gedeutet wird. Entsprechend verschieden sind die Lehren, die er daraus zieht. So dient der Turmbau-Mythos:

- als ständige Warnung für Systemgestalter vor mangelhafter konzeptioneller Integrität,
- als Aufforderung zu Bemühungen für einvernehmliche Notationen, Schnittstellen, Werkzeuge und Vorgangsweisen,
- als Appell für gesteigerte architektonische Anstrengungen und konsequentes Projektmanagement,
- als Forderung nach einer Organisationslehre und gesellschaftsinformatischer Einbettung der Systementwicklung und
- als Mahnung zum Erkennen und zur Berücksichtigung menschlicher Grenzen und Unvollkommenheiten.

Die Turmbau-Erzählung fungiert hier sowohl als bildstarke Metapher, als richtungsweisendes Leitbild und denkanstoßende Parabel. Der narrative Charakter vermittelt dabei komplexe Erfahrungen über Problemkonstellationen, Gefahrensituationen und situative Zusammenhänge, die nicht als explizites, systematisierbares Wissen verfügbar sind. Derartige narrative Formen der Erfahrungsverdichtung enthalten somit keine Gewissheiten, sie fordern vielmehr zur eigenen Deutung auf, zum Situationserkennen und Problemvergleich. Sie haben insofern eine prophylaktische Rolle, die die Beteiligten zum frühzeitigen Erkennen möglicher Irrwege und typischer Fehlerkonstellationen animiert. Die starke Bildlichkeit erleichtert ihre Präsenz im Problemspeicher von Systementwicklern und Technikbewertern. So haben Negativleitbilder wie das von Zemanek formulierte „Turmbau-zu-Babel-Syndrom“, der von Brooks plastisch ausgemalte „Second-System-Effekt“ und die „Alles-auf-einen-Streich-Lösung“, für die auch die Redensart „put all the eggs in one basket“ verwendet wird, ein hohes Erinnerungspotenzial.

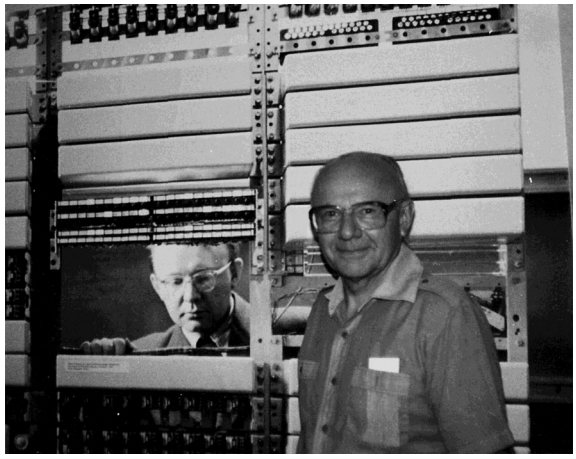


Abb. 6 Zemanek im Jahr 2000 vor einem Foto vom Bau des Relais-Versuchsrechners URR1 von 1952 (Foto aus dem Bildarchiv Heinz Zemaneks)

Heinz Zemanek hat damit aus eigener Erfahrung heraus visuelle und narrative Formen der Designreflexion und der Wissenschaftskritik entwickelt, die in der neueren Konstruktionsmethodik und Designtheorie unter dem Begriff „Storytelling“ laufen. Bei ihm sollte man vielleicht besser von „Historytelling“ sprechen, denn er nutzte permanent die Computer-, Informatik- und Technikgeschichte als Erfahrungsmaterial für die System- und Technikbewertung. Wer ihn auf Tagungen, bei Vorträgen oder am Telefon ansprach, bekam umgehend tiefe Einblicke in historische Zusammenhänge der Wissenschafts- und Geistesgeschichte der Informatik und Computerwelt. Diese jahrzehntelang inspirierende Oral-History-Quelle ist nun für immer versiegt, es bleiben uns nur noch das reichhaltige Zemanek-Schriften-Archiv[4] und umfangreiche Nachlasspapiere, die, hoffentlich bald gut erschlossen und über das Internet zugänglich, den

[4] Der Katalog der Heinz-Zemanek-Schriften-Sammlung der Hauptbibliothek der TU Wien ist mit 511 Einträgen zugänglich über die Einwahl: http://kitt.ub.tuwien.ac.at/ALEPH.

Informatikern auch weiterhin als Erfahrungsschatz dienen können.

Literatur

1. Babbage C (1830) Notation, zit. nach: Campbell-Kelley M (ed) (1989) Works of Babbage, Bd. 1. Pickering & Chatto, London, New York University Press, New York, pp 409–424
2. Babbage C (1864) Passages from the Life of a Philosopher. London, zit. nach: Campbell-Kelley M (ed) (1989) Works of Babbage, Bd. 11. Pickering & Chatto, London, New York University Press, New York, New York, pp 140
3. Beizer B (1971) The Architecture and Engineering of Digital Computer Complexes, 2 Bde. Bd 1. New York, London
4. Bekic H (1984) On the formal definition of programming languages. In: Programming Languages and Their Definition. Lecture Notes in Computer Science, Bd. 177. Plenum Press, pp 86–106
5. Blaauw GA (1972) Computer Architecture. Elektron Rechenanl 14(4):154–159
6. Brooks F (1975/95) The Mythical Man-Month. Essays in Software Engineering. Anniversary Edition, Reading, MA, Menlo Park, CA, London
7. Eco U (1997) Die Suche nach der vollkommenen Sprache. Hanser, München
8. Hasselmeier H, Spruth WG (1974) Rechnerstrukturen. Oldenbourg, München, Wien
9. Hellige HD (2004) Die Genese von Wissenschaftskonzepten der Computerarchitektur: Vom „system of organs" zum Schichtenmodell des Designraums. In: Hellige HD (Hrsg) Geschichten der Informatik. Visionen, Paradigmen und Leitmotive. Springer, Berlin, Heidelberg, New York
10. Hellige HD (2006) Software Engineering Approaches Before the Notion. Paper presented at the conference „Pioneering Software in the 1960s in Germany, The Netherlands, and Belgium" Centrum Wiskunde & Informatica (CWI) in Amsterdam 2.–4. November 2006. http://www.uni-bremen.de/de/artec/mitglieder/prof-dr-hans-dieter-hellige.html
11. Hellige HD (2008) Wissenschaft vs. Design: Konstruktionslehren für den Maschinenbau, den Computer und die Software im historischen Diskursvergleich. In: Warnke M, Weber-Wulff D (Hrsg) Kontrolle durch Transparenz – Transparenz durch Kontrolle. Pro Business, Berlin, S 120–124
12. Locke WN, Booth AD (Hrsg) Machine translation of languages: Fourteen Essays. MIT Press, Cambridge, MA, New York
13. Neurath O et al. (1937) Towards an Encylopedia of Unified Science. In: McGuiness BF (ed) (1987) Unified Science Vienna Circle Collection, Vol. 19. Springer
14. Moskaliuk SS (ed) (2001) Heinz Zemanek. Series Classics of World Science, Vol 8. TIMPANI, Kiew
15. Rosen S (1964) Programming Systems and Languages: A Historical Survey. In: AFIPS Vol 26,1, Proceedings of the April 21–23, Spring Joint Computer Conference, 7
16. Sammet JE (1969) Programming Languages: History and Fundamentals. Prentice-Hall, Englewood Cliffs, NJ
17. Sammet JE (1981) History of IBM's technical contributions to high level programming languages. IBM J Res Develop 25(5):532
18. Sammet JE (1991) Some Approaches to, and Illustrations of, Programming Language History. Ann Hist Comput 13(1):38 ff
19. Weaver W (1955) Translation. MIT Press, Carlsbad, NM, July 15, 1949
20. Wegner P (1972) The Vienna Defintion Language. Comput Surv 4:1–8
21. Wegner P (1976) Programming Languages? The First 25 Years. IEEE T Comput 25:12
22. Weizer N (1981) A History of Operating Systems. Datamation 27(1):122
23. Zemanek H (1958) „Mailüfterl", ein dezimaler Volltransistor-Rechenautomat. Elektrotech Maschinenbau 75(15/16):457
24. Zemanek H (1959) Der Volltransistor-Rechenautomat des Instituts für Niederfrequenztechnik der Technischen Hochschule in Wien. Math Method Oper Res 3:1
25. Zemanek H (1966) Semiotics and Programming Languages. Commun ACM 9(3): 139–143
26. Zemanek H (1971) Was ist Informatik?. Elektron Rechenanl 13(4):160
27. Zemanek H (1973) Generalized architecture. In: Advances in Cybernetics and Systems Research, Proceedings of the European Meeting on Cybernetics and Systems Research, Wien 1972, Transcripta, London 1973, pp 1–6
28. Zemanek H (1973) Philosophie der Informationsverarbeitung. ntz 26(8):388
29. Zemanek H (1975) Formalization – History, Present, and Future. In: Hackl CE (ed) Programming Methodology. 4th Informatik Symposium, Wildbad, September 1974. Lecture Notes in Computer Science, Bd 23. Springer, Berlin, Heidelberg, New York
30. Zemanek H (1980) Abstract Architecture. General Concepts for System Design. In: Bjørner D (ed) Abstract Software Specifications, Proceedings of the Copenhagen Winterschool 1979. Lecture Notes in Computer Science, Bd 86. Springer, Berlin, Heidelberg, New York, pp 1–42
31. Zemanek H (1983) Wird der Computer die Technik vermenschlichen? Elektrotech Maschinenbau 100(11):448–458
32. Zemanek H (1986) Gedanken zum Systementwurf. Ein von Gebäude und Computer generalisierter Architekturbegriff, der auch für Fahrzeuge und Verkehrssysteme nützlich sein könnte. In: Maier-Leibnitz H (Hrsg) Zeugen des Wissens. Hase & Köhler, Mainz, S 99–125
33. Zemanek H (1987) An Interview with Heinz Zemanek (Oral History 127), conducted by William Aspray on 14 and 16 February 1987. Charles Babbage Institute. http://conservancy.umn.edu/handle/107723)
34. Zemanek H (1992) Das geistige Umfeld der Informationstechnik. Springer, Berlin, Heidelberg, New York
35. Zemanek H (1993) Das „Mailüfterl". Ein österreichischer Aufbruch ins Computerzeitalter. In: Rafeiner O (Hrsg) Patente, Marken, Muster, Märkte. Der gewerbliche Rechtsschutz international. Manz'sche Verlags- und Universitätsbuchhandlung, Wien
36. Zemanek H (1993/2001) Philosophische Wurzeln der Informatik im Wiener Kreis. In: Schefe P, Hastedt H, Dittrich Y, Keil G (Hrsg) Informatik und Philosophie Dagstuhl, 21.–25. September 1992. BI-Wissenschaftsverlag, Mannheim 1993, S 85–117
37. Zemanek H (1994/2001) Early Foundations of Formal Modelling and Language Specification: Vienna Definition Language (VDL) and Vienna Development Method (VDM). In: Brunnstein K, Raubold E (eds) (1994) Applications and Impacts. Information Processing '94, Vol 2
38. Zemanek H (2003) Fünf Jahrzehnte Computerentwicklung. In: Wandlungen der Telekommunikation und des Computers. Alcatel SEL Stiftung, SR 50, Stuttgart 21. http://www.verbundkolleg-berlin.de/AndereVeranstaltungen/stuttgart2.pdf
39. Zemanek H (2004) Konrad Zuse und die Systemarchitektur, das Mailüfterl und der Turmbau zu Babel. In: Hellige HD (Hrsg) Geschichten der Informatik. Visionen, Paradigmen und Leitmotive. Springer, Berlin, Heidelberg, New York, S 141–170
40. Zemanek H (2008) Mailüfterl, Algorithmen, Formale Definition und Semiotik. In: Chroust G, Lenz M (Hrsg) Forum Informatik. Erlebte Meilensteine der Informatik. 4 Generationen berichten. SEA-Publications: SEA-SR-17, Institute for Systems Engineering and Automation 13
41. Zuse K (1948/49) Über den allgemeinen Plankalkül als Mittel zur Formulierung schematisch-kombinativer Aufgaben. Arch Math 1(6):441–449
42. Zuse K (1968) Gesichtspunkte zur sprachlichen Formulierung in Vielfachzugriffsystemen unter Berücksichtigung des „Plankalküls". In: Händler W (Hrsg) Teilnehmer-Rechensysteme, NTG-Tagung vom 21.–23. September 1966 in Bad Hersfeld, Oldenbourg, München, Wien