Hintergründe und Zusammenhänge

Wenn eine Idee, wie der WAM-Ansatz, in die Jahre gekommen und durch viele Hände gegangen ist, wird langsam unklar, wer welche Anteile in welchem Zusammenhang dazu beigetragen hat. Viele Ideen anderer Autoren wurden aufgegriffen und wir wollen festhalten, wer welche Einflüsse auf die Ausprägung des Ansatzes hatte. Diesen Fragen soll hier kurz nachgegangen werden. Dabei werden charakteristische Merkmale von WAM hervorgehoben.

- **WAM ist aus einer Idee bei der Entwicklung einer Programmierumgebung entstanden**

Unstrittig ist der Ursprung von WAM: In der GMD (Gesellschaft für Mathematik und Datenverarbeitung; heute: Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme IAIS) haben seit Anfang der achtziger Jahre bis 1991 Reinhard Budde, Karl-Heinz Sylla und Heinz Züllighoven in einem Institut für Softwaretechnologie gearbeitet. Die Idee zu WAM kam bei der Entwicklung einer Programmierumgebung auf [Budde et al. 85]. Dort wurde die Frage diskutiert, was Programmierer eigentlich tun und vor allem, womit sie es tun. Die Unix Shell mit den in einer Pipe verbundenen Tools war interessant. Als klar wurde, dass Tools mit oder auf irgendetwas arbeiten müssen, war die Idee von Werkzeugen und Materialien geboren.

- **WAM ist ein Ansatz der anwendungs- und objektorientierten Softwareentwicklung**

Am Anfang von WAM stand die Einsicht, dass eine rein technische Betrachtung von Softwareentwicklung selten zu gebrauchstauglichen Softwaresystemen führt, die Anwender begeistern. Deshalb wurden weitergehende Ansätze wichtig: anwendungsorientierte Softwareentwicklung mit Prototyping [Budde et al. 84, Streich et al. 83] und die skandinavisch geprägte Interpretation der Objektorientierung [Ehn 88, Nygaard 90],[Budde et al. 87]. Dabei wird die Objektorientierung nicht als Programmiertechnik gesehen, sondern als eine Möglichkeit der Simulation, bei der fachliche Begriffe, Gegenstände und Abläufe ohne größere Brüche in Softwarekonstrukten abgebildet werden..

- **Anwendungsorientierte Softwareentwicklung ist auch ein Kommunikations- und Lernprozess**

Ein wesentliches Merkmal von WAM ist sicherlich die Anwendungsorientierung. Für Auftraggeber und vor allem Anwender soll etwas Nützliches geschaffen werden. Um dies möglichst rasch zeigen zu können, wurde von Anfang an Prototyping eingesetzt. Grundlegend für das Verständnis von Prototyping war die entsprechende Konferenz mit dem Hauptartikel von Christiane Floyd [Floyd 84]. Ihr menschenzentrierter STEPS-Ansatz, in dem Entwickler und Benutzer gleichberechtigt bei der Softwareentwicklung zusammenarbeiten, hat die evolutionär zyklische Vorgehensweise von WAM ebenfalls entscheidend geprägt [Floyd et al. 89]. Floyds Verständnis von Softwareentwicklung als Kommunikations- und Lernprozess ist als grundlegender Autor-Kritiker-Zyklus in WAM aufgenommen worden.

- **Werkzeug, Material und Automat sind die prägenden Entwurfsmetaphern**

Um 1987 erschienen die ersten Arbeiten zur Werkzeugidee. Karl-Heinz Sylla hatte sich intensiv mit dem Smalltalk-System beschäftigt und war auf das MVC-Paradigma gestoßen (in [Gamma et al. 96] als Model-View-Controller-Muster beschrieben). [Budde et al. 87] handelt von Werkzeugen und „Arbeitsobjekten“; dort wird die grundlegende Architektur eines interaktiven Werkzeugs als eine Interpretation des Paradigmas beschrieben. Die Einbettung der Idee von interaktiven Werkzeugen in einen erkenntnistheoretischen Werkzeugbegriff beschrieben Reinhard Budde und Heinz Züllighoven [Budde & Züllighoven 90], [Budde & Züllighoven 92]). Eine objektorientierte Architektur von Werkzeugen und Materialien verbunden durch Aspekte wurde schon damals konzipiert, aber erst später publiziert [Bäumer et al. 95b].

Ausgangspunkt ist die arbeitswissenschaftliche und philosophische Einsicht, dass Menschen bei ihrer täglichen Aufgaben Materialien mit Werkzeugen bearbeiten. Dies gilt nicht nur im Handwerk, sondern ebenso für Bürotätigkeiten oder Softwareentwicklung. Also haben wir die handwerklichen Begriffe „Materialien“ und „Werkzeuge“ als Metaphern für Arbeit am Computer interpretiert. Alle für die Erledigung von Arbeitsaufgaben mit dem Computer notwendigen Elemente in einer interaktiven Anwendungssoftware werden als Werkzeuge und Materialien modelliert und so für den Anwender und den Entwickler deutlich greifbarer. Später kamen als weitere Metapher hinzu: der Automat, die Arbeitsumgebung, der Behälter.

Das Kürzel *WAM* wurde an der Universität Hamburg geprägt. Guido Gryczan und Heinz Züllighoven begannen ab 1991 in den Lehrveranstaltungen des neu eingerichteten Arbeitsbereichs Softwaretechnik die Konzepte zu unterrichten. Aus der Langfassung des Grundmusters Werkzeug-Aspekt-Material wurde auch durch Guido Gryczans Dissertation [Gryczan96] dann Werkzeug-Automat-Material; woraus die Hamburger Studierenden dann rasch WAM machten.

- **WAM orientiert sich an den Aufgaben der verschiedenen Arbeitsplätze**

Der WAM-Ansatz ist aus softwaretechnischer Sicht eine anwendungsorientierte Interpretation der Objektorientierung; diese zeigen zahlreiche Veröffentlichungen. So war der Artikel [Budde et al. 89] Anstoß der langjährigen Zusammenarbeit mit der RWG (Rechenzentrale der Württembergischen Genossenschaften, heute Fiducia IT AG) in Stuttgart, in deren Rahmen das GEBOS-System (Genossenschaftliches Bürokommunikations- und Organisationssystem) als eines der ersten großen objektorientierten Anwendungssysteme im Bankenbereich entwickelt wurde [Bürkle et al. 92]. Innovativ war die Ausrichtung an unterschiedlichen Arbeitsplatztypen: Expertenarbeitsplatz, Funktionsarbeitsplatz, Gruppenarbeitsplatz etc. Um die fachlichen Zusammenhänge und Abläufe zu verstehen, wurden Bänker systematisch mit Szenariotechniken in den Entwicklungsprozess einbezogen. Diese fachlich und technisch sehr erfolgreiche Anwendung hat der WAM-Idee zum wirtschaftlich professionellen Durchbruch verholfen.

- **Das gemeinsame Verständnis der Arbeitsabläufe und der Fachbegriffe ist zentral für die WAM-Vorgehensweise.**

In den ersten Anwendungsprojekten wurde deutlich, wie wichtig es für Entwickler ist, die Aufgaben und Abläufe der Anwender und vor allem ihre Fachsprache zu verstehen. Für gemeinsame Workshops wurden Techniken und Hilfsmittel gesucht, die eine gleichberechtigte Zusammenarbeit der verschiedenen Beteiligten unterstützen sollten. Die gängigen graphischen Darstellungsmittel wie Netze und UML-Diagramme erwiesen sich als ungeeignet, da sie im Sinne von Stein Braten [Braten 73] das Modellmonopol der Informatiker verstärkten. Zum gemeinsamen Verständnis des Einsatzkontextes wurden in der Fachsprache der Anwender geschriebene Glossare für uns zum wichtigen Instrument. Jeder Begriff wurde anhand seiner Umgangsformen, also der Tätigkeiten die der Anwender an- und mit ihm durchführt, beschrieben. Der Vorschlag von Peter Checkland [Checkland 75], sog. Rich Pictures als allgemeinverständliche Diagrammtechnik zu verwenden, inspirierte Heinz Züllighoven und Ingrid Wetzel zu den Kooperationsbildern [Krabbel et al. 96], aus denen sich dann die werkzeuggestützte exemplarische Geschäftsprozessmodellierung (eGPM) [Breitling & Hofer 2012] entwickelte. Die eGPM hat sich bei der gemeinsamen Modellierung, Analyse und Gestaltung von Arbeitsprozessen und ihrer IT-Unterstützung als wichtigste originäre Technik des WAM-Ansatzes erwiesen.

- **Grundlegend für WAM ist die Strukturähnlichkeit zwischen fachlichen und softwaretechnisch architektonischen Modellen.**

Fachliche Begriffe und Gegenstände in Software abzubilden ist eine Idee, die sich schon lange in anwendungsorientierten Ansätzen und in den frühen Arbeiten von Allan Kay findet [Kay 77]. WAM greift diese Sichtweise mit dem Prinzip der Strukturähnlichkeit auf. Die Fachlichkeit soll sich aber nicht nur für die Anwender im User Interface oder im Benutzungsmodell wiederfinden lassen, sondern die Konstruktion der Softwareelemente und der gesamten Architektur bestimmen [Bäumer et al. 97]. So sollen fachlich orientierte Klassen die zentralen Begriffen und Prozessen des Anwendungsbereichs modellieren. Die Architektur eines Anwendungssystems soll die Begriffsgebäude und die Dienstleistungsbündel einer Domäne erkennbar abbilden. Dies fördert die Kommunikation zwischen den Beteiligten und die kontinuierliche Weiterentwicklung von Software, da sich Änderungen im Anwendungskontext oder in der Technologie wechselseitig gut zuordnen lassen.

- **Software-Qualität in WAM bezieht sich auf Zusammenhang zwischen fachlicher und technischer Architektur.**

Werkzeuge, Materialien und Automaten lassen sich als Entwurfsmetaphern in der Software-Architektur relativ einfach als Elemente der GUI oder insgesamt im Benutzungsmodell abbilden. Arbeitsprozesse können dazu passend über Checklisten und Laufzettel modelliert und unterstützt werden [Gryczan 1996]. Doch das Konzept der Strukturähnlichkeit geht weiter. Die Abbildung von Gegenständen und Prozessen auf die Elemente der tieferen logischen und technischen Schichten sollen ebenfalls in einer WAM-Architektur erkennbar sein. Dazu dienen sog. fachliche Dienste oder Services [Schuler & Otto 2000]. Sie stellen den Werkzeugen und Automaten an ihrer Schnittstelle sog. Dienstleistungsbündel zur Verfügung, über die Sammlungen von Materialien und fachliche Prozesse auf diesen Materialien zugänglich sind. So lässt sich elegant die Welt der fachlichen Gegenstände und Prozesse mit der technischen Dimension von Workflow Engines und Persistenzmedien verbinden.

Eine Softwarearchitektur nach WAM hat üblicherweise die Dimensionen Handhabung, Fachlichkeit und Technologie. Dabei stellt sich die Frage, wie gut die jeweilige konkrete Architektur eine Softwareanwendung für ihre verschiedenen Zwecke geeignet ist. Dies wird nicht nur durch das Maß ihrer Gebrauchstauglichkeit ausgedrückt. Hinzu kommen innere Qualitätsmerkmale wie Verständlichkeit, Änderbarkeit oder Testbarkeit [Lilienthal 2008]. Entsprechend der generellen Orientierung an hoher Softwarequalität gehören Konzepte, Techniken und Vorgehensweisen der konstruktiven Qualitätssicherung zur Vorgehensweise nach WAM. Ob konkrete Architekturen den Regeln der WAM-Modellarchitektur entsprechend, lässt sich mit Architektur-Tools wie dem Software Tomograph überprüfen.

- **WAM betrachtet den Gegenstand Software und den inkrementell evolutionären Entwicklungsprozess.**

WAM betrachtet die beiden zentralen Aspekte der Softwareentwicklung – den Gegenstand Software und den Prozess, in dem Software entsteht, eingesetzt und verändert wird. Anforderungen an den Gegenstand wie Strukturähnlichkeit, Anwendungsorientierung und die fachlich motivierten Entwurfsmetaphern haben unmittelbare Auswirkungen auf den Entwicklungsprozess. Dazu kommt, dass Softwareentwicklung, aus Sicht von WAM, Lernen und Kommunizieren in enger Rückkopplung zwischen den Beteiligten bedeutet. Damit sind die Grundzüge einer agilen Vorgehensweise formuliert. Teilergebnisse wie Inkremente, Releases oder Iterationen werden in WAM als Mittel zum Zweck „Gebrauchstauglichkeit“ gesehen [Lippert et al. 2003]. Der Zusammenhang zur WAM-Modellarchitektur wird über sog. Bebauungspläne hergestellt, in dem ein System als „Vision“ skizziert und nach Kernsystem, Ausbaustufen und Spezialsysteme aufgeteilt wird. So lassen sich schrittweise fachlich sinnvoll einsetzbare Systemversionen in agilen Prozessen erstellen.

- **WAM berücksichtigt unvollständige Systemspezifikationen und den permanenten Wandel im Einsatzkontext von Software.**

Die Informatik ist traditionell bei der Modellierung von Software und ihres Einsatzkontextes von der Annahme eine wohldefinierten Systems ausgegangen: Software als Gegenstand lässt sich umfassend formal spezifizieren; der Einsatzkontext eines Softwaresystems lässt sich vor der Entwicklung vollständig durch eine Anforderungsspezifikation festlegen. Beide Annahmen haben sich für Anwendungssysteme in Organisationen als fragwürdig erwiesen. Lehman hat festgestellt, dass Anwendungssoftware schon bei der Entwicklung beginnt, die sie einsetzende Organisation zu verändern [Lehman 80]. Wegner stellt die These auf, dass interaktive Software mit dem klassischen Modell der Turingmaschine als formale Grundlage nicht ausreichend zu beschreiben ist [Wegner 97]. Dazu kommt, dass die große Zahl der Systeme einer Anwendungslandschaft und die sich ständig verändernden Prozesse einer Organisation eine vollständige und korrekte Modellierung als Grundlage eines Entwicklungsvorhabens im laufenden Betrieb faktisch unmöglich macht [Züllighoven & Lilienthal 2014].

Der WAM-Ansatz berücksichtigt mit seinem iterativ zyklischen Vorgehen, bei dem ein Softwaresystem schrittweise in enger Abstimmung mit den Beteiligten entwickelt und eingesetzt wird, diese Grenzen klassischer geschlossener Ansätze. WAM macht die Planung und Spezifikation selbst zum Gegenstand kontinuierlicher Anpassung an Unschärfen der Modellierung und Veränderungen im Einsatzkontext.

- **Literatur**

[Bäumer et al. 95b] D. Bäumer, R. Budde, K.-H. Sylla, G. Gryczan, H. Züllighoven: Objektorientierte Konstruktion von Software-Werkzeugen und -Materialien. Informatik-Spektrum, Band 18, Heft 4, August 1995, Berlin, Heidelberg: Springer-Verlag, S. 203-210.

[Bäumer et al. 97] Bäumer, D., Gryczan, G., Knoll, R., Lilienthal, C., Riehle D., Züllighoven, H. Framework Development for Large Systems. Communications of the ACM, October 1997. Vol. 40, No. 10

[Braten 73] Braten, Stein. Model monopoly and communication: Systems theoretical notes on democratization. Acta Sociologica, 1973, 16. Jg., Nr. 2, S. 98-107.

[Breitling & Hofer 2012] Holger Breitling, Stefan Hofer: Beispielhaft gut modelliert: Exemplarische Geschäftsprozess-Modellierung in der Praxis. OBJEKTspektrum,06/2012, S. 8-13.

[Budde et al. 84] R. Budde, K. Kuhlenkamp, L. Mathiassen, H. Züllighoven (eds.): Approaches to Prototyping. Berlin, Heidelberg: Springer-Verlag, 1984.

[Budde et al. 85] R. Budde, K.-H. Sylla, K. Kuhlenkamp, H. Züllighoven: Software-Entwicklungsmethodik für eine Programmierumgebung in Prolog. In: Molzberger, P., Zemanek, G.V.. (Hrsg.): Software-Entwicklung: Kreativer Prozeß oder formales Problem. German Chapter of the ACM, Berichte 22, S. 103-116, 1985.

[Budde et al. 87] R. Budde, K.-H. Sylla, K. Kuhlenkamp, H. Züllighoven: Bib - ein Bibliographie-System. Und: Programmentwicklung mit Smalltalk. Und: Ein Erfahrungsbericht über die Arbeit mit Smalltalk. In "SMALLTALK verstehen und anwenden", H.-J. Hoffmann (Hrsg.), München, Wien: Carl Hanser Verlag, 1987.

[Budde et al. 89] R. Budde, K.-H. Sylla, K. Kuhlenkamp, H. Züllighoven: Der Entwurf objektorientierter Systeme. Handbuch der modernen Datenverarbeitung, Wiesbaden: Forkel-Verlag, Heft 145, S. 13-24, 1989.

[Budde & Züllighoven 90] R. Budde, H. Züllighoven: Software-Werkzeuge in einer Programmierwerkstatt, Berichte der GMD, Nr. 182, München, Wien: Oldenbourg, 1990.

[Budde & Züllighoven 92] R. Budde, H. Züllighoven: Software Tools in a Programming Workshop. In: Ch. Floyd, H. Züllighoven, R. Budde, R. Keil-Slawik (eds.): Software Development and Reality Construction. Berlin, Heidelberg: Springer-Verlag, 1992.

[Bürkle et al. 92] U. Bürkle, G. Gryczan, H. Züllighoven: Erfahrungen mit der objektorientierten Vorgehensweise in einem Bankenprojekt Informatik-Spektrum, Band 15, Heft 5, Berlin, Heidelberg: Springer-Verlag, 1992, S. 273-281.

[Checkland 75] Checkland P. 1975. The Development of Systems Thinking by Systems Practice – a methodology from an action research program. Progress in Cybernetics and Systems Research 2: 278–283.

[Ehn 88] Ehn, Pelle. Work-oriented design of computer artifacts. Dissertation. Umeå University, 1988.

[Floyd 84] Floyd, Christiane: A Systematic Look at Prototyping. In: Approaches to Prototyping. Budde, Reinhard, Kuhlenkamp, Karin, Mathiassen, Lars, Züllighoven, Heinz (Hrsg.), Springer Berlin Heidelberg,1984. S. 1-18

[Floyd et al. 89] Floyd, Christiane, Fanny-Michaela Reisin, and Gerhard Schmidt. "STEPS to software development with users." ESEC'89. Springer Berlin Heidelberg, 1989. 48-64. http://link.springer.com/chapter/10.1007/3-540-51635-2_32#page-1

[Gryczan96] G. Gryczan: Prozeßmuster zur Unterstützung kooperativer Tätigkeit. Wiesbaden: Deutscher Universitätsverlag, 1996 (DUV: Informatik).

[Krabbel et al. 96] A.Krabbel, I. Wetzel, S. Ratuski. Objektorientierte Analysetechniken für übergreifende Aufgaben. In: Jürgen Ebert (Hrsg.): GI-Fachtagung Softwaretechnik '96, September 1996, S. 65 – 72

[Kay 77] Kay, Alan. "Microelectronics and the personal computer." Scientific American. 1977.

[Lehman 80] Lehman, Meir M. "Programs, life cycles, and laws of software evolution." Proceedings of the IEEE 68.9 (1980): 1060-1076.

[Lilienthal 2008] Lilienthal, C., Komplexität von Softwarearchitekturen - Stile und Strategien. Dissertation, elektronische Veröffentlichung bei der Staats- und Universitätsbibliothek Hamburg, 16.07.2008, www.sub.uni-hamburg.de/opus/volltexte/2008/3725/

[Lippert et al. 2003] Lippert, M., Roock, S., Wolf, H., Züllighoven. "Developing complex projects using XP with extensions." Computer 36.6 (2003): 67-73.

[Nygaard 90] Nygaard, K. (1990, January). Program development as a social activity. In PDC (pp. 4-13).

[Schuler & Otto 2000] Schuler, Norbert; Otto, Michael. Laufzeitarchitekturneutrale Softwarekonstruktion mit fachlichen Services. Diplomarbeit, Fachbereich Informatik, Softwaretechnik, 2000.

[Streich et al. 83] H. Streich, K.-H. Sylla, H. Züllighoven: Anmerkungen zum Prototyping. In: I. Kupka (Hrsg.): Proceedings der 13. GI-Jahrestagung. Informatik-Fachberichte Nr. 73, Berlin, Heidelberg: Springer Verlag, 1983.

[Wegner 97] Wegner, Peter. "Why interaction is more powerful than algorithms." Communications of the ACM 40.5 (1997): 80-91.

[Züllighoven & Lilienthal 2014] Heinz Züllighoven, Carola Lilienthal. Der Turmbau zu Babel: Grenzen der Informatik. OBJEKTspektrum, 01/2014, S. 24-28.