7 Epilog

7.1 Tool-Beschreibung

Was noch zu tun bleibt, ist eine kurze Beschreibung dessen abzugeben, was die kleine Applikation kann und im Gegensatz zur üblichen Handhabung auch das, was sie nicht kann. Denn manchmal ist das, was die Nutzung einer Software verhindert, weniger ihre Fehler oder ihre Unhandlichkeit, als die enttäuschten Hoffnungen ihrer Anwender.

Also was ist es überhaupt? Es ist ein Tool zur Bewertung von Dateisystemen anhand der **4fF**-Methode. Ausgelegt ist es auf ca. 50 Dateien, die zwar aus mehreren Dateisystemen stammen dürfen, doch müssen diese unbedingt unabhängig voneinander sein. Warum? Weil Felder nur Dateien zugeordnet werden können und Eigenschaften nur Feldern, nicht aber Dateisystemen. Damit ist nicht nur für Dateien erforderlich, dass ihre Namen eindeutig sind, sondern auch für Dateifelder und Eigenschaften. Dies aber bedeutet unabhängige Dateisysteme. Stellen Sie sich zum Beispiel eine Auftrags- und Projektverwaltung vor, die nicht zusammengehören würden. Beide haben sicher Kunden-Dateien mit überwiegend gleichen Eigenschaften. Da beide Anwendungen aber nicht zusammenhängen, müssen auch deren Eigenschaften deutlich unterschieden sein, weil sonst die Tentakelzahl als Verwendung dieser Eigenschaft im Dateisystem falsch berechnet wird. Und auch bei Vorgänger und Nachfolgern aus den zwei unterschiedlichen Kundendateien ist viel zuviel Aufmerksamkeit geboten, um die gemeinsame Unterbringung der beiden Dateisysteme in denselben Systemdateien des **4fF**-Tools anzuraten. In solch einem Fall sollten die Anwender lieber zwei verschiedene Kopien der **4fF**-Dateien verwenden, um die Trennung der Dateisysteme zu gewährleisten.

Aber innerhalb dieser Einschränkung lassen sich kleine Dateisysteme damit recht übersichtlich abbilden. Aus den vier grundlegenden Angaben Eigenschaft, Wertezahl, Vorgänger und Nachfolger werden die feld- und aufgabenzogenen Feld- und Dateitypen gewonnen mit der Möglichkeit, sie graphisch darzustellen und zu drucken, wobei der Druck sogar teilweise individuell geändert werden kann.

Abbildung 396 Fall einer kleinen Projektverwaltung nach Analyse (sichtbare Unterscheidung zwischen Stamm- und Bewegungsdateien)

Die Eigenschaften geben darüberhinaus noch einen Einblick in die Struktur des Dateisystems, doch was hier fehlt ist, eine dateiübergreifende Struktur-Graphik. Auch ist die Eingabe der vier grundlegenden Angaben nicht maschinell unterstützt, Werte können nicht als Formeln angegeben werden, und eine Verwertung über die Informierung der Anwender hinaus ist gar nicht vorgesehen. Es ist eben nur eine erste, einfache Art eines solchen Tools, was auch die einzige Ausrede dafür ist, dass Systemdateien und zu verwaltende Daten gemeinsam bearbeitet werden.

So, damit habe ich alles Wesentliche gesagt. Ich schätze, dass ich damit keinen Marketing-Profi hinterm Ofen hervorlocken kann, doch das ist auch nicht mein Job.

7.2 Working Concepts

Nun, da Sie dieses kleine Tool mit seiner kleinen Aufgabe und den vielen Fehlern im Laufe der Entwicklung gesehen haben, werden Sie vielleicht im ersten Augenblick widersprechen, wenn Sie meine Behauptung erneut hören, dass ausführliche Detailkonzepte kontraproduktiv sind. Hätte ich dies und jenes besser berücksichtigt, dann wäre mir dieser und jener Fehler garantiert nicht unterlaufen, werden Sie einwenden. Genau das tun die meisten Leute in unserer Branche nämlich, fast bedingungslos in die vollständige Vorab-Planung zu vertrauen, sonst könnten sie nicht an ihr System der Konzipierer, Programmierer und Tester glauben.

Natürlich geht es nicht ohne Konzepte. Sie müssen beispielsweise klar und deutlich wissen, was der Output Ihrer Arbeit sein soll, was das zu konzipierende Gerüst denn mindestens bieten muss. Problemlösungen erfordern also exakte Maßgaben für alles, was als Ergebnis dienen soll. Doch nicht nur diese Ergebnis-Schnittstellen müssen einwandfrei bestimmt sein, auch die Art und der Umfang der sonstigen zu beachtenden Schnittstellen muss gegeben sein. Wesentlich ist dabei jedoch, dass nur das „Was“ bedeutsam ist, nicht aber das „Wie“. In welchem Ausmaße diese Schnittstellen tatsächlich in Anspruch genommen werden müssen, wie das Alles tatsächlich umgesetzt wird, ist auch durch Ihre Lösung bereits von sich aus gegeben. Doch eben nicht durch Ihre geplante, sondern durch Ihre am Ende tatsächlich funktionierende Lösung.

Da Sie das nicht im luftleeren Raum tun, sondern sich mit Ihrer Belegschaft und Ihrer Entwicklungsumgebung der maschinellen Ausstattung der Kunden und deren Arbeitsorganisation unterordnen müssen, sind ungeheuer viele Dinge automatisch festgelegt. Klar, genau das versuchen Sie zu beschreiben in Ihrem Konzept, nicht wahr? Die Bedingungen, die Wechselwirkungen der Kundensituation, innerhalb derer Sie die Aufgabe zu erledigen haben, deren gewünschtes Resultat so glasklar und detailliert aufgelistet worden ist.

Wie gehen Sie also vor, wenn Sie diese detaillierten Konzepte machen? Sie nehmen Pflichtenhefte oder sonstige Anforderungsprofile, beschreiben die Voraussetzungen, Inputdaten, Ausstattung, Beschränkungen und sonstige Einflussgrößen, die Ihnen gesagt werden, die Sie sich erfragen oder die Ihnen sonstwie auffallen und daraus gewinnen Sie ein Bild dessen, was Ihre Software tun soll. Das ist auch völlig korrekt so und bisher noch ohne jede Verbesserungsmöglichkeit.

Dann jedoch setzen Sie sich an Ihren Computer und konzipieren. Sie schreiben es nieder in Dokumente für Ihre Kunden, Sie setzen sich an graphische Tools, die Ihnen die Hierarchien und Beziehungen der Objekte demonstrieren sollen und erarbeiten womöglich die ganzen Attribute und Interaktionen oft genug bis auf Feld- und Methoden-Ebene herunter. Im Falle eines mächtigen Case-Tools können Sie daraus sofort die Programme erstellen lassen und erkennen, ob Ihr Konzept denn auch funktioniert, doch merkwürdigerweise sind auf dem gängigen Markt der Software-Hersteller Produkte aus solchen Case-Tools nicht die großen Beherrscher. Sie sind auch in den Stellenangeboten praktisch nicht zu finden, also denke ich, dass dieser Ausnahmefall bei den generellen Betrachtungen über Detailkonzepte vernachlässigt werden kann. Für den Normalfall gehe ich also davon aus, dass Sie Ihr kunstvoll ausgearbeitetes Detailkonzept dann an Ihre Programmierer weiterreichen, die daraus die endgültigen Programme erstellen, die ihrerseits dann in die Testabteilungen durchgereicht werden.

Doch habe ich durch mein Tagebuch nicht demonstriert, wie selbst kleine Änderungen manchmal funktionierende Abläufe zerstören konnten? Sie meinen, durch ein genaues Konzept hätte dies vorhergesehen werden können?

Sie meinen also, ein Konzept kann all die vielfältigen Voraussetzungen und Bedingungen eines Systems beschreiben, das bereits einer solch kleine Applikation wie der vorliegenden zugrunde liegt?

Einspruch.

Bis vor kurzem galt das Dreikörper-Problem in der Physik als unlösbar in dem Sinne, dass es keine Formel, keine a priori-Lösung gibt, die sein Verhalten für alle Anfangsbedingungen und beliebige Zeitabschnitte beschreibt. Nur der jeweils nächste „Schritt“ ist formelmäßig beherrschbar. Nun hat ein findiger mathematischer Kopf die gesuchte Gleichung aufgestellt, doch diese Gleichung ist wirklich die hohe Kunst der Mathematik - und das Problem? Drei Körper, idealisiert zu irgendeinem punktähnlichen Identitätsobjekt, wechselwirken in einer einzigen Art und Weise miteinander, das ist alles.

Drei Identitäten mit einer einzigen Art der Wechselwirkung - und das erforderte die ganze Kunst der Mathematik des vergangenen 20. Jahrhunderts, wenn diese drei Identitäten in einem interagierenden System dargestellt wer-

den sollen. Vier Identitäten sind schon gar kein Thema mehr.

Wieviele Identitäten, sprich Objekte, weist dagegen eine normale Software-Applikation auf? Wieviele Arten von Wechselwirkung, sprich öffentliche Methoden, weist ein Objekt auf? Ein Detailkonzept zu erstellen, heißt hier die Formel zu suchen, die die Lösung a priori zur Verfügung stellt, in Mathematik und Physik mühevoll bis zu drei Objekten mit je einer gleichen Methode erfolgreich bestanden, wobei hier das Augenmerk auf die Einschränkung „für alle Anfangsbedingungen und beliebige Zeitabschnitte“ gilt. Doch genau das soll doch auch Ihr Konzept leisten, nicht wahr? Die Lösung der Aufgabe für alle oder wenigstens alle erlaubten Anfangsbedingungen zu bieten und daraus den gewünschten Output zu erzielen. Wievielen Einflussfaktoren ist alleine meine kleine **4fF**-Applikation unterworfen? Es sind ja nicht nur die Dateien mit ihren recht wenigen Attributen, die hier als charakteristisch erkannt und verwendet werden, es sind die vielfältigen Tätigkeiten der Anwender, die über all die Buttons und Fenster zugelassen sind und eine unerhörte Bandbreite von Abläufen erzeugen können. Das hätte ich niemals aufs Papier niederschreiben können! Denn was das Dreikörper-Problem als Vorzug aufweist gegen jede Software-Lösung, ist nach dem notwendigen äußeren Anstoß die völlige Isolation von der Umgebung, es ist ein abstraktes Konstrukt, das hochidealisierte Identitäten mit der Reduktion auf eine einzige Art der Wechselwirkung verbindet, eine formelmäßig berechenbare Wechselwirkung wohlgemerkt. Formelmäßige Berechenbarkeit bedeutet aber totale Wiederholbarkeit des Verhaltens dieser betrachteten Identität, um in der Sprache der Information zu bleiben. Das heißt, dass es genügt, die Zyklen des wiederholbaren Verhaltens zu finden, um damit alle Varianten des Verhaltens in Raum und Zeit zu erfassen. Diese Zyklen des wiederholbaren Verhaltens sind dann die a priori-Lösung des Systems. Hochidealisierte Identität bedeutet darüberhinaus keinerlei sonstige Attribute, die irgendwelche sonstigen Wechselwirkungen ausüben können und völlige Isolation heißt, dass es keine Schnittstellen zu beachten gibt, von denen äußere Einwirkungen stammen könnten.

Das sind also die Voraussetzungen für Lösbarkeit mithilfe einer Formel, die Voraussetzungen dafür, dass dynamische Methoden über zeitlose Abbildungen umfassend beschrieben werden können. Hier liegt nämlich das Augenmerk auf „zeitloser Abbildung“, denn nichts anderes ist eine Formel. Und auch ein Konzept ist undynamisch, nicht wahr? Ein Dokument, eine Graphik, eine Source sind in sich unveränderlich, sie können gespeichert werden, weil sie sich auf stabile Muster reduzieren lassen. Sie sind immer nur Möglichkeiten von Dynamik, ähnlich wie in der Quantenphysik die verschiedenen Wellengleichungen mögliche Lösungen darstellen, die durch irgendwelche Instanzierungsprozesse dann „ausgewählt“ werden, um es unprofessionell auszudrücken.

Es ist genau der Unterschied zwischen Code-Klasse und Objekt-Klasse, der hier das Problem deutlich macht, ein Unterschied, den Sie im Laufe Ihrer Arbeit in der Programmierung objektorientierter Entwicklungsumgebungen sehr genau kennenlernten.

Ein Detailkonzept zur vollständigen Darstellung eines Software-Problems ohne die Dynamik der Aufgabe ist deshalb wohl gar nicht möglich, wenn es erst einmal über das Stadium eines „niedrigdimensionalen“ formalen Algorithmus hinausgeht. Und wenn Sie dann noch daran denken, dass häufig noch während der Programmierung Änderungswünsche eintreffen und berücksichtigt werden müssen, dann sehen Sie, dass Detailkonzepte spätestens diese neu auftauchenden Fälle unmöglich vollständig abdecken können.

Das heißt aber überhaupt nicht, dass es generell keine Lösungen gibt, sonst gäbe es keine Software-Industrie und wir beide wären arbeitslos. Die Lösungen sind nur nicht a priori gegeben, das beweisen solche Namen wie „Wasserfall-Modell“ oder „iteratives Modell“ für die Programmkonzeption. Was bedeutet dies? Dass Korrekturen, die sich erst im Laufe der Programmierung und des Tests ergeben, sich rückwirkend auf das Konzept auswirken können und dieses dann wiederum die Programmierung beeinflusst. Das ist unsere Art der Arbeit, nicht wahr?

Und genau hier ist Omnis Studio einfach unschlagbar. Weil Sie die Detail-Realisierung, ohne die keine Software trotz aller Schwierigkeiten auskommt, nicht in Dokumente oder Graphiken fassen müssen, sondern gleich in

dynamisch agierender Umgebung niederschreiben können. Sie können das dynamische Abbild, das Ihr Gehirn von dem Problem machte, nicht nur in seinen statischen Komponenten festhalten, sondern gleich in seinen so anfälligen Beziehungen untereinander ausprobieren und das alles so überschaubar, dass selbst größere Projekte im Griff bleiben.

Sie können „working concepts“ erzeugen, eine Art von Prototyping, die sowohl von Ihren Kunden als auch von den Programmierern, die die Sache weiter bearbeiten sollen, in Aktion erlebt werden können. Bei Ersterem dekken Sie mit diesem Vorgehen frühzeitig Missverständnisse auf, bei Letzterem erreichen Sie einen höheren Grad an Verständnis, was die Aufgabe letztendlich ist. Denn noch eine weitere Schwierigkeit folgt dem Detailkonzept in der Praxis oft auf dem Fuß. Software ist eine Abbildung des gewünschten Systems und seines gewünschten Verhaltens im Computer und zu seiner Erstellung müssen in jedem aktiv beteiligten menschlichen Gehirn möglichst gleiche Abbildungen erzeugt werden, wenn dieses Verfahren wirklich funktionieren soll. Dieses Kommunikationsproblem alleine zeigt, dass Software-Erstellung möglichst wenig vertikal zerlegt werden sollte.

Es ist nun einmal nicht möglich, fertige Konzepte ohne das Produkt selbst zu erstellen. Gerade die Programmierer empfinden es häufig genug als schmerzhaft, auf Konzeptmängel zu stoßen und versuchen zu müssen, Klärung von Sachverhalten zu erreichen, die längst geklärt schienen, ohne jedoch die Kommunikationsmöglichkeiten genießen zu können, die die hauptberuflichen Konzipierer benutzen dürfen. Die Tücke liegt im Detail, das bewahrheitet sich viel zu oft, doch das richtige Verfahren, diese Tücke zu beherrschen, liegt nicht in absoluter Planung, sondern in der zielgerichteten Durchführung mit hoher Kommunikationsfrequenz und der Möglichkeit ständiger Überprüfung des bisher Erarbeiteten.

Es ist augenscheinlich viel effektiver, Grobplanung zu betreiben und sie nach der aktuellen Situation ins Detail überzuführen, als alle Eventualitäten auszuloten und vorherzubestimmen. Warum augenscheinlich? Abgesehen von der Existenz des Dreikörper-Problems ist es die Methode der Natur, ihre Aufgaben mit der Unendlichkeit eines umgebenden Universums abzustimmen. Schnittstellen werden mit einem Überangebot an Kontaktstellenen ausgestattet, um sowohl die Schnelligkeit der Kontaktierung als auch die Genauigkeit der Konstaktstelle zu gewährleisten. Überschüssige Kontaktelemente, die nicht in Wechselwirkung mit einem Gegenüber treten können, sterben genau deshalb sehr schnell ab. Nicht punktgenaue Lösungen sind das Rezept der Natur, sondern „flächige“, um es bildhaft auszudrücken. Lösungen, die viele Möglichkeiten abdecken können, anstatt auf eine einzige fixiert zu sein.

Eine komplexere Variante demonstriert die Natur in mindestens zwei sehr mächtigen „Software“-Systemen, dem Immunsystem und dem Gehirn. In beiden Fällen bietet der genetische Bauplan eine Grobplanung an mit Regelwerk und genügend Arbeitsmaterial und in beiden Fällen wird das Nützliche geformt oder vom Überflüssigen aussortiert durch die aktuelle Umgebung mit all ihren Bedingungen und Risiken und Querschlägern, durch „Lernen“. Lernen ist die Berücksichtigung der Umwelt durch Speicherung ihres Verhaltens zur Ermittlung unterscheidbarer Systeme und ihrer wiederholbaren Vorgängen, zur Bestimmung informativer Prozesse der eigenen Umgebung.

Wenn die Natur in Millionen von Jahren keine bessere Lösung für alle Eventualitäten fand, dann gibt es vielleicht auch keine?