DIPLOMARBEIT

Bernhard Mayr

ABBILDUNG VON GESCHÄFTSPROZESSEN IN EINER .NET ANWENDUNG MIT DER WINDOWS WORKFLOW FOUNDATION 4.0

2010

DIPLOMARBEIT

ABBILDUNG VON GESCHÄFTSPROZESSEN IN EINER .NET ANWENDUNG MIT DER WINDOWS WORKFLOW FOUNDATION 4.0

Autor: Bernhard Mayr, 21986

Studiengang: Informationstechnik

Seminargruppe: Kl08wStA

Erstprüfer: Prof. Dr. Ing. Volker Delport

Zweitprüfer: Dipl.-Ing. (FH) Roland Dorfer

Mondsee, Juni 2010

Bibliografische Angaben

Mayr, Bernhard

Abbildung von Geschäftsprozessen in einer .NET Anwendung mit der Windows Workflow Foundation 4.0

95 Seiten, 31 Abbildungen, 17 Tabellen, 50 Codebeispielen, 2 Anlagen mit Quelltexten und softwaretechnischen Grundlagen,

Hochschule Mittweida (FH), Fakultät Elektro- und Informationstechnik

Diplomarbeit, 2010

Referat

Die vorliegende Arbeit befasst sich mit der Thematik, wie Geschäftsprozesse in einer .NET Anwendung mithilfe der Windows Workflow Foundation 4.0 abgebildet werden können. Zunächst erfolgt im Zuge einer Generalisierung eine Definition von fundamentalen Begriffen aus dem Workflow Umfeld. Im Anschluss daran wird die Windows Workflow Foundation 4.0 im speziellen hervorgehoben und ermittelt, ob die Windows Workflow Foundation die an ein Workflow Management System gestellten Anforderungen erfüllen kann. Konkrete Beispiele, wie die eigentliche Umsetzung der Geschäftsprozesse erfolgen kann, werden erarbeitet und finden sich im Anhang der Diplomarbeit wieder.

Abstract

The dissertation in front of you is the product of a study of the practical applications of Windows Workflow Foundation 4.0. Presented first will be definitions and fundamental concepts to familiarize the reader with the subject and processes of workflow management. Next, the reader will be given a detailed look at the integration of Windows Workflow Foundation 4.0, in actual practice, in a workflow management process. The effectiveness of Windows Workflow Foundation 4.0 is presented with detailed data and results in the dissertation supplement.

Inhaltsverzeichnis

Abbildungsverzeichnis

Für alle Abbildungen gilt: Bei Abbildungen ohne Quellenangabe handelt es sich um eigene Darstellungen.

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

Abkürzung	Erläuterung
API	Application Programming Interface
ASP	Active Server Pages
BPEL	Business Process Execution Language
Bsp.	Beispiel
bzw.	beziehungsweise
CD	Corporate Design
CIL	Common Intermediate Language
CLR	Common Language Runtime
COM	Component Object Model
DMS	Dokumenten Management System
ff.	fortfolgend
ggf.	gegeben falls
GUID	Globally Unique Identifier
IDE	Integrated Development Environment
IIS	Internet Information Server
inkl.	inklusive
KVP	Key Value Pair
MS	Microsoft
SOA	Service Orientierte Architektur
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
u. a.	unter anderem
VS	Visual Studio
WCF	Windows Communication Foundation
WF	Windows Workflow Foundation
Wf	Workflow
WfMC	Workflow Management Coalition
WfMS	Workflow Management System
WPF	Windows Presentation Foundation

XAML	Extensible Application Markup Language
XML	Extensible Markup Language
z. B.	zum Beispiel

Vorwort

„Denken und danken sind verwandte Wörter;
wir danken dem Leben, in dem wir es bedenken."

Thomas Mann (1875-1955)
deutscher Schriftsteller, 1929 Nobelpreis für Literatur

Die vorliegende Arbeit wurde im Frühjahr 2010 im Zuge meiner Tätigkeit als Programmierer bei der Firma *AUER – Die Bausoftware GmbH* erstellt und als Diplomarbeit an der *Hochschule Mittweida - University of Applied Sciences* eingereicht.

Mein besonderer Dank gilt Herrn Studiendekan Prof. Dr. Ing. Volker Delport für die Unterstützung und Betreuung bei der Erstellung dieser Diplomarbeit und für die Übernahme des Prüfungsvorsitzes.

Die Themenwahl wurde durch meinen betrieblichen Betreuer und Zweitprüfer, Dipl.-Ing. (FH) Roland Dorfer maßgeblich initiiert. Auch ihm möchte ich an dieser Stelle für die permanente Unterstützung und für die Übernahme der Zweitkorrektur danken.

Ein großes Dankeschön geht weiters an die Korrekturleser, welche mir bei technischen und grammatikalischen Fragen zur Seite standen.

Danke!

Mondsee, im Frühjahr 2010

Bernhard Mayr

1 Übersicht

„Wer keine neuen Lösungen anwendet, muss neue Übel akzeptieren:
denn die Zeit ist der größte Neuerer.“

Francis Bacon (1561-1626)
englischer Staatsmann und Philosoph

Im einleitenden Kapitel werden die Motivation und die Aufgabenstellung dieser Diplomarbeit besprochen. Gleichzeitig erfolgt ein kurzer Überblick zu den einzelnen Kapiteln dieser Arbeit.

1.1 Motivation

Ein Großteil der heute verfügbaren Anwendungen basiert nicht auf einer Service-orientierten Architektur[1]. Bei einer SOA steht vor allem das Anbieten, Vermitteln und Nutzen von Diensten im Vordergrund. Bei herkömmlichen Anwendungen ist dies meist nicht der Fall. Somit können diese Anwendungen nur sehr schwer bis gar nicht in einen bestehenden Geschäftsprozess integriert werden. Um die Ergebnisse dieser Anwendungen trotzdem in einem Geschäftsprozess einfließen zu lassen, sind oftmals manuelle Tätigkeiten und Medienbrüche notwendig – dies stellt jedoch wiederum potenzielle Fehlerquellen dar.

Zudem ist das Abbilden von Geschäftsprozessen in einer Anwendung immer noch mit einem Hauch von Mystik umgeben. Und dies obwohl in heutigen Unternehmen Geschäftsprozesse und Workflows immer mehr an Bedeutung gewinnen. Lösungen wie

[1] kurz: SOA
[2] Der BizTalk Server stammt von der Firma Microsoft. Er dient der Integration, Verwaltung und

Microsoft BizTalk[2] Server sind zwar vom Namen her bekannt – es fehlt jedoch die konkrete Erfahrung damit.

Mit dem Erscheinen der aktuellen Version des .NET Frameworks 4.0 wurde auch die Windows Workflow Foundation überarbeitet.

Die Kombination dieser Faktoren war ausschlaggebend für unser Unternehmen, sich näher mit der Windows Workflow Foundation zu befassen. Der Zeitpunkt des Erscheinens der aktuellen Version des .NET Frameworks 4.0 ist dabei insofern günstig, da sich dies mit dem Start der Entwicklung einer neuen Programmgeneration deckt. Die Entscheidung, ob eine kommende Programmgeneration in der Lage ist Geschäftsprozesse abzubilden oder nicht, ist von so fundamentaler Bedeutung, dass sie bereits in einer sehr frühen Design- und Planungsphase berücksichtigt und geklärt werden muss.

Als Beispiel für eine alltägliche Umsetzung eines Geschäftsprozesses mittels eines Workflows in einem Softwareunternehmen kann folgende Situation angeführt werden: Zu den Geschäftszielen eines Softwareunternehmens gehören unter anderem die Erstellung, der Vertrieb und die Schulung von Software. Das Erstellen einer neuen Version kann als Geschäftsprozess angesehen werden. Als Unterstützung bei der Erstellung von Software kann eine automatische Vergabe von Versionsnummern im Zuge des Kompiliervorganges verwendet werden. Die automatisierte Generierung einer Versionsnummer und der Eintrag dieser in die erstellten binären Dateien kann als Workflow realisiert werden. Als konkretes Beispiel dafür sei das Produkt „Team Foundation Server 2010“ der Firma Microsoft angeführt. Dies setzt die Vergabe einer Versionsnummer laut Lamb [Lam10] nach dem oben angeführten Ablauf um. Bereits an diesem Beispiel erkennt man, dass ein Workflow dabei helfen kann alltägliche Tätigkeiten einer gewissen Automatisierung zu unterwerfen und damit das Fehlerpotenzial zu senken.

[2] Der BizTalk Server stammt von der Firma Microsoft. Er dient der Integration, Verwaltung und Automatisierung von Geschäftsprozessen, indem er Nachrichten abruft, konvertiert und die Kommunikation zwischen den verschiedenen Systemen verwaltet.

1.2 Zielsetzung

Diese Diplomarbeit soll die Frage beantworten, wie Workflows unter Zuhilfenahme der Windows Workflow Foundation 4.0 im Zuge einer auf dem Microsoft .NET Framework[3] 4.0 erstellten Software realisiert werden können. Neben theoretischen Grundlagen sollen auch praktische Beispiele mit Lösungsansätzen erstellt werden. Die Realisierung soll ferner auf aktuelle Thematiken und Konstrukte aus C# bzw. Visual Studio 2010 eingehen und diese bei Bedarf erörtern, um einen möglichst zusammenhängenden Lösungsansatz zu präsentieren.

Ziel dieser Diplomarbeit ist es nicht, Alternativen wie z. B. Microsoft BizTalk Server oder auf Java basierte Workflow Engines zu erörtern.

Verwandte Technologien, wie z. B. Windows Presentation Foundation[4] oder Windows Communication Foundation[5] werden im Zuge der Diplomarbeit erwähnt. Es wird jedoch bewusst nicht intensiver darauf eingegangen.

Sollten Gründe erkannt werden, die gegen die Verwendung der Windows Workflow Foundation 4.0 sprechen, so müssen diese dokumentiert werden und auch in Kapitel 5 nochmals angeführt werden.

In der Zusammenfassung dieser Diplomarbeit muss die Frage beantwortet werden können, ob die Windows Workflow Foundation 4.0 zur Abbildung von Geschäftsprozessen in einer .NET Anwendung geeignet ist.

---

[3] Ein Framework (englisch für Rahmenstruktur, Fachwerk) ist ein Programmiergerüst, das in der Softwaretechnik, vor allem im Umfeld der objektorientierten Softwareentwicklung, verwendet wird.

[4] kurz: WPF; Windows Presentation Foundation (Codenamen „Avalon“), ist ein Grafik Framework und Teil des Microsoft .NET Framework 3.0.

[5] kurz: WCF; Die Windows Communication Foundation (Codename „Indigo“), stellt in Microsoft Windows eine neue dienstorientierte Kommunikationsplattform für verteilte Anwendungen dar und ist ebenfalls Teil des Microsoft .NET Framework 3.0.

1.3 Kapitelübersicht

Die Diplomarbeit besteht aus fünf aufbauenden Kapiteln.

Kapitel 1 liefert eine allgemeine Einleitung zur Diplomarbeit.

Kapitel 2 (Generalisierung) beschäftigt sich mit grundlegenden Begriffsdefinitionen aus dem Workflow Umfeld. Es wird dabei absichtlich nicht auf die Terminologie der Windows Workflow Foundation 4.0 eingegangen. Dieses Kapitel soll einen Überblick über die Zusammenhänge liefern und stützt sich dabei hauptsächlich auf Literatur zum Thema Workflow Management.

In **Kapitel** 3 (Konkretisierung) werden zunächst Anforderungen an ein Workflow Management System gesammelt. Um beurteilen zu können, in welchem Umfang dieses Anforderungen durch die Windows Workflow Foundation 4.0 abgedeckt werden, wird der Aufbau und die interne Struktur der Windows Workflow Foundation 4.0 erklärt. Darin werden auch die Ziele, welche die Firma Microsoft mit der Windows Workflow Foundation 4.0 verfolgt, dargestellt. Im Anschluss daran wird erörtert, ob die Windows Workflow Foundation 4.0 alle Kriterien und Anforderungen, welche an ein Workflow Management System gestellt werden, erfüllt. Weiters werden bekannte Workflow Engines mit firmeninternen Standards bewertet. Zuletzt werden Vorteile, welche durch die Verwendung eines Workflow Management Systems entstehen, angeführt und wiederum mit der Windows Workflow Foundation 4.0 in Beziehung gebracht.

Kapitel 4 (Spezialisierung) beschäftigt sich mit der konkreten Problemlösung: Wie können Workflows mithilfe der Windows Workflow Foundation 4.0 in einer C# Anwendung abgebildet werden? Zunächst werden Activities, Workflows und der notwendige Host theoretisch erklärt und stets mit Auszügen von konkreten Beispielen untermauert. Darauf folgend werden Workflow Extensions, welche z. B. für das Persistieren und das Nachverfolgen von Workflows benötigt werden, angeführt. Die letzten beiden Punkte in diesem Kapitel beschäftigen sich mit dem Umgang von Fehlern in Workflows und mit der Integration des Workflow Designers in eine eigene Anwendung.

Schließlich werden in **Kapitel** 5 die Resultate der einzelnen Kapitel der Diplomarbeit noch einmal zusammengefasst. Hier werden auch die Leistungen des Diplomanden aus

seiner Sicht skizziert. Zusätzlich wird ein Ausblick auf mögliche Weiterentwicklungen gegeben. Schlussendlich wird die Frage beantwortet, ob die Windows Workflow Foundation 4.0 für die Abbildung von Geschäftsprozessen geeignet ist.

2 Begriffsbestimmungen und Definitionen

„Ein Begriff entsteht, indem eine produktive Kraft Reize gestaltet.“

Friedrich Nietzsche (1844-1900)
deutscher Philosoph, Dichter und klassischer Philologe

In der Begriffswelt rund um das Thema Workflow existieren viele Fachbegriffe und Missverständlichkeiten. Aus diesem Grund werden in diesem Kapitel vor allem Begriffe aus dem Workflow Umfeld erläutert und zueinander in Beziehung gebracht. In folgender Abbildung sind die wesentlichen Begriffe, beginnend beim Geschäftsprozess bis hin zur Einzelaufgabe dargestellt. Auf den folgenden Seiten werden diese nun genauer betrachtet.

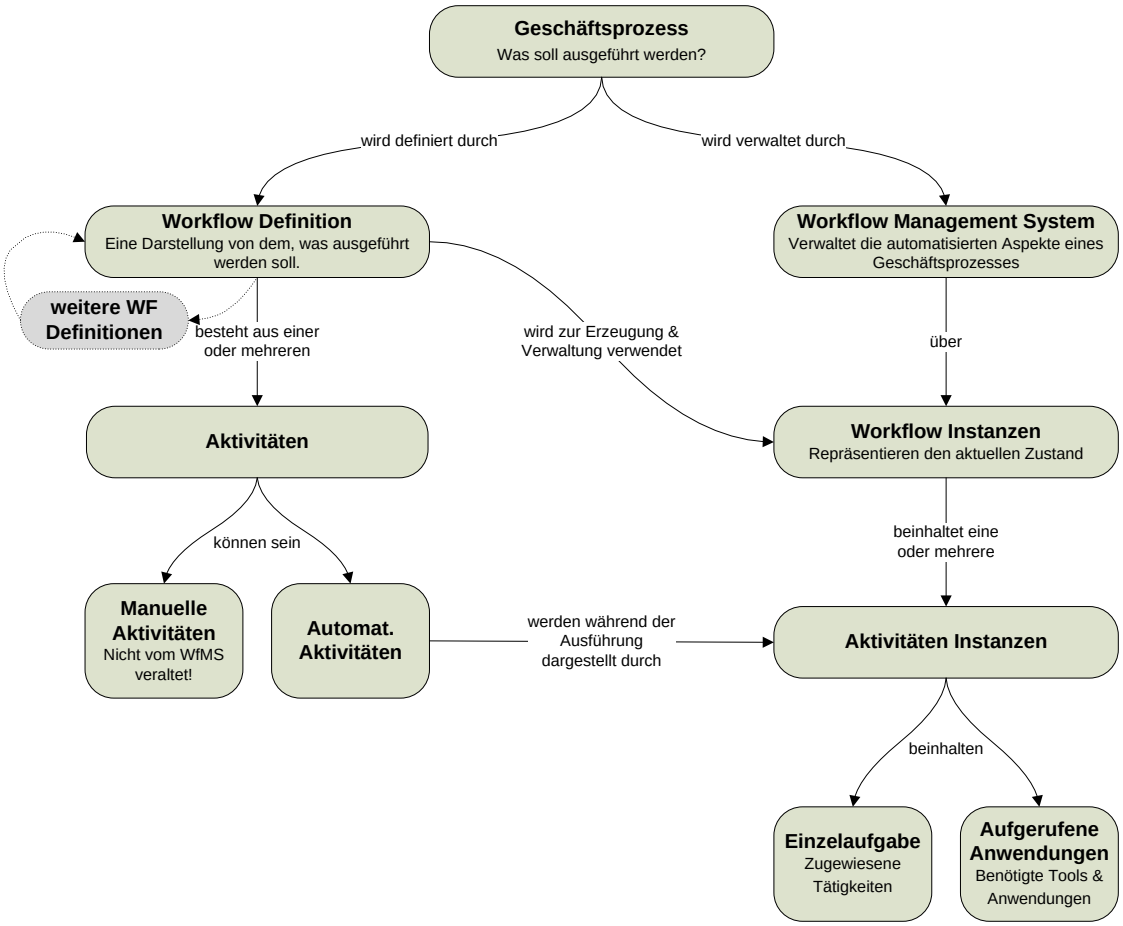


Abbildung 1 Abhängigkeiten und Zusammenspiel grundlegender Begriffe aus dem Workflow Umfeld. (Quelle: eigene Darstellung in Anlehnung an [Wor99 S. 7])

2.1 Geschäftsprozess

Für den Begriff „Geschäftsprozess“ konnte keine einheitliche Definition in der Literatur gefunden werden. Alle gefundenen Definitionen haben jedoch gemeinsam, dass sich ein Geschäftsprozess stets auf die Geschäftsziele eines Unternehmens bezieht. Stellvertretend wird folgende Definitionen angeführt.

Die Workflow Management Coalition[6] definiert einen Geschäftsprozess als eine Menge einer oder mehrerer verbundener Prozeduren oder Aktivitäten, welche gemeinsam ein Geschäftsziel oder eine Geschäftsstrategie verfolgen. Im Regelfall wird dies im Kontext

[6] Die Workflow Management Coalition (kurz: WfMC), eine Organisation mit Sitz in England, bemüht sich um Begriffsdefinitionen und Standardisierungen im Bereich der Prozess Automatisierung.

einer Organisationsstruktur betrachtet, welche die erforderlichen Rollen und Beziehungen definiert. [Wor99 S. 10]

Heuckeroth [Heu95 S. 2] hat folgende Unterscheidung für Geschäftsprozesse erarbeitet:

Tabelle 1 Einteilung von Geschäftsprozessen

Kriterium	**Beschreibung**
Routine Prozess	Routine Prozesse lassen sich weitgehend automatisieren, da sie sich kaum verändern und wiederkehrende Strukturen aufweisen. Des weiteren besitzen sie nur geringe Schnittstellen zu anderen Prozessen. Markant ist auch ihr hoher Grad an Arbeitsteilung.
Regelprozess	Bei Regelprozessen sind die Struktur und die Anzahl der Teilaufgaben noch kontrollierbar. Im Zuge der konkreten Ausführung müssen jedoch seitens der Mitarbeiter individuelle Anpassungen und Veränderungen am Prozess vorgenommen werden.
Einmalige Prozesse	Einmalige Prozesse lassen sich weder planen noch strukturieren und sind somit auch nicht automatisierbar. Bearbeitet wird dieser Prozess, welcher sich meist durch eine sehr geringe Arbeitsteilung definiert, von einzelnen Mitarbeitern oder z. B. von einem Projektteam.

Für die konkrete Umsetzung in unserer Applikation bedeutet dies, dass wir zunächst ermitteln müssen, welche Prozesse einer gewissen Routine unterliegen.

2.2 Workflow

Die treffendsten Übersetzungen des englischen Begriffes „Workflow“[7] in die deutsche Sprache lauten „*Arbeitsablauf*“ [Scr07 S. 3] oder „*Arbeitsfluss*“ [Sch08 S. 10].

Ein Workflow ist die vollständige oder teilweise Automatisierung eines Geschäftsprozesses. Dokumente, Informationen oder Aufgaben werden dabei gemäß einer Reihe von Verfahrensregeln zwischen verschiedenen Teilnehmern geführt und von diesen bearbeitet. [Wor99 S. 8]

[7] kurz: Wf

Eine schlichtere Definition verwendet Schattauer. Er definiert einen Workflow als eine endliche Folge von sequenziell oder parallel angeordneten Aktivitäten, welche durch ein Ereignis ausgelöst werden. [Sch08 S. 10]

Das Ziel eines Workflows sieht Scribner darin, dass die Zusammenarbeit aller am Geschäftsprozess involvierten Elemente, unter Zuhilfenahme von Software, zu einer erfolgreichen Abarbeitung eines Geschäftsprozesses führt. [Scr07 S. 3]

Diese Definitionen zeigen, dass es sich bei einem Workflow um eine automatisierte Abarbeitung eines Geschäftsprozesses handelt. Mehrere Workflows können einen Geschäftsprozess teilweise oder vollständig automatisiert abarbeiten. Des weiteren kann aus diesen Definitionen geschlossen werden, dass ein Workflow nicht mit einem Geschäftsprozess gleichzusetzen ist.

Im Unterschied zu Workflows beinhalten Geschäftsprozesse also stets einen wirtschaftlichen Aspekt und stellen einen erfolgsorientierten Vorgang in einem Unternehmen dar. Workflows können einen solchen Geschäftsprozess (oder Teile davon) automatisieren bzw. auf elektronische Art abbilden und dadurch unterstützen.

Wie aus Abbildung 2 hervorgeht, ist das Abbilden eines Geschäftsprozesses zwar technisch möglich; die Ausführung eines Geschäftsprozesses bleibt jedoch stets einem Workflow vorbehalten. Für diese Diplomarbeit ist vor allem die operative Ebene, also das Workflow Management, von Interesse.

Ein Geschäftsprozess behandelt demnach nur die Frage, **was** getan werden muss. Ein Workflow hingegen betrachtet sowohl **was** getan werden muss als auch **wie** dies umzusetzen ist.

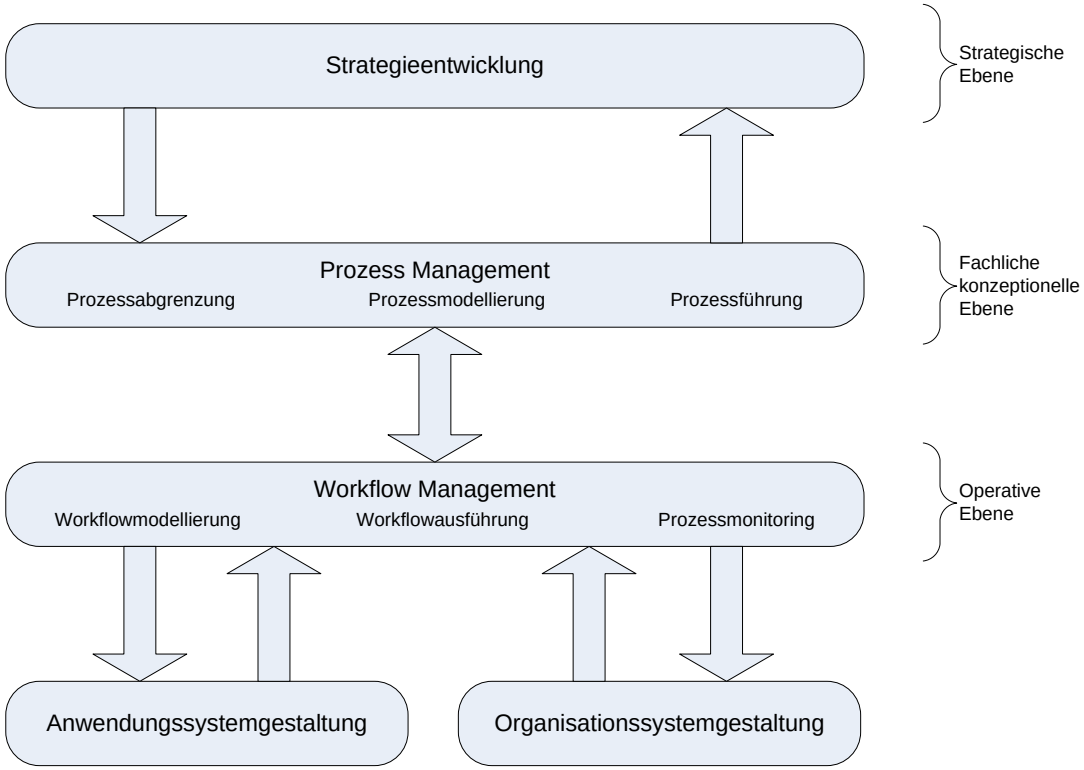


Abbildung 2 Beziehung zwischen Geschäftsprozess- und Workflowmanagement
(Quelle: eigene Darstellung nach [Sch08 S. 99])

Bei einer Workflow Definition[8] handelt es sich um die Darstellung eines Geschäftsprozesses in einer für ein Workflow Management System verständlichen Form. Eine Workflow Definition beinhaltet beliebige Aktivitäten und deren Verknüpfungen untereinander. Des weiteren sind in der Definition die Kriterien für den Start und das Ende eines Workflows enthalten. [Wor99 S. 11]

Die Workflow Definition kann auch als „*Schablone*“ [Ric04 S. 30] für eine konkrete Workflow Instanz betrachtet werden.

Modelliert werden können Workflow Definitionen z. B. in Extensible Application Markup Language[9] oder im BPEL[10] Format. BPEL basiert ebenfalls auf XML. [Sch08 S. 32 ff.]

[8] Synonyme: Prozessdefinition, Prozess Schema, Prozessmodel, Model Definition
[9] kurz XAML; Bei XAML handelt sich um eine auf XML basierte Beschreibungssprache. XAML wird detaillierter in den folgenden Kapiteln behandelt.
[10] Abkürzung für: Business Process Execution Language; Englisch für: Ausführungssprache für Geschäftsprozesse. Auf BPEL wird im Rahmen dieser Diplomarbeit nicht näher eingegangenen, da

2.3 Aktivitäten

Eine Aktivität beschreibt eine Arbeitseinheit, welche einen logischen Schritt innerhalb eines Workflows bildet. Es gibt zwei Arten von Aktivitäten:

- manuelle Aktivitäten, also ohne Computer Unterstützung und
- automatisierte Aktivitäten.

Automatisierte Aktivitäten benötigen manuelle und/oder maschinelle Ressourcen für die Ausführung. Eine Aktivität ist die kleinste Arbeitseinheit, welche durch ein Workflow Management System erfasst wird. [Wor99 S. 11]

Eine manuelle Aktivität kann nicht durch ein Workflow Management System verwaltet werden. Für ihre Erfüllung ist eine Interaktion mit einer Person notwendig. Aus Gründen der Vollständigkeit werden diese jedoch trotzdem oftmals in Modellen berücksichtigt. [Wor99 S. 15]

Das Erstellen eines Arbeitszeugnisses sei als Beispiel für eine manuelle Aktivität angeführt.

Eine automatisierbare Aktivität kann vollständig durch ein Workflow Management System erzeugt und abgearbeitet werden. [Wor99 S. 14]

Das Versenden einer Rechnung per E-Mail kann als Beispiel für eine automatisierbare Aktivität angeführt werden.

Vor allem in automatisierbaren Aktivitäten liegt großes Potenzial verborgen! Je mehr automatisierbare Aktivitäten vorhanden sind, umso effektiver kann eine Workflow Definition gestaltet werden. In Kapitel 4.3 (Seite 57 ff.) wird deshalb konkreter auf diese Thematik eingegangen.

Bei einer Einzelaufgabe handelt es sich um eine entsprechende Tätigkeit innerhalb einer Instanz einer Aktivität (welche wiederum innerhalb einer Workflow Instanz ausgeführt wird), die von einer Person manuell ausgeführt werden muss. Üblicherweise werden dem Benutzer die abzuarbeitenden Einzelaufgaben in Form einer Aufgabenliste

---

BEPL für unseren konkreten Anwendungsfall keine Beutung hat. Es wurde nur als mögliche Alternative zu XAML erwähnt.

präsentiert. Das Workflow Management System wird mittels Schnittstellen über den Fertigstellungsgrad informiert. [Wor99 S. 19]

Eine Entscheidung, welche durch eine Person getroffen werden muss, kann als Einzelaufgabe angesehen werden.

2.4 Instanz

Eine Instanz repräsentiert stets eine konkrete Ausprägung einer Definition (Workflow oder Aktivität). Eine Instanz wird durch ein Workflow Management System aufgrund einer Definition erstellt, kontrolliert, gesteuert und auch beendet. Wesentlich ist aber vor allem, dass eine Instanz immer über einen inneren Zustand und alle notwendigen Daten verfügt, welche für die Abarbeitung notwendig sind. [Wor99 S. 15]

Eine Instanz einer Aktivität wird im Kontext einer Workflow Instanz ausgeführt. Erzeugt und verwaltet werden diese durch ein Workflow Management System. Einer Workflow Instanz können mehrere Aktivitäten Instanzen zugeordnet sein. Eine Aktivität Instanz kann jedoch immer nur einer Workflow Instanz zugeordnet sein. [Wor99 S. 17]

2.5 Workflow Engine

Bei einer Workflow Engine handelt es sich um einen Software Service, welcher die Umgebung für die Ausführung einer Workflow Instanz bereitstellt. [Wor99 S. 57]

Die Ausführungsumgebung beinhaltet Funktionen

- zur Interpretation der Workflow Definition
- zur Erstellung einer Workflow Instanz
- und deren Ausführung
- Überwachung
- Beendigung
- und Persistierung

Die Workflow Engine stellt somit den Kern eines Workflow Management Systems dar.

2.6 Workflow Management System

Laut Workflow Management Coalition handelt es sich bei einem Workflow Management System[11] um ein System, welches Definition, Erstellung und Ausführung eines Workflows unter Zuhilfenahme einer Workflow Engine ermöglicht. Ein Workflow Management System muss in der Lage sein, die Workflow Definition zu interpretieren. Eine weitere Anforderung an ein Workflow Management System ist die Fähigkeit, mit allen Workflow Teilnehmern zu interagieren und wenn nötig Anwendungen und Werkzeuge aufzurufen. [Wor99 S. 9]

Nach Schattauer [Sch08 S. 24] dient ein Workflow Management System zur aktiven Steuerung arbeitsteiliger Prozesse und basiert oftmals auf einem Dokumenten Management System, da Dokumente häufig zentrale Bestandteile von Workflows sind. Auch Schattauer erwähnt, dass eine oder mehrere Workflow Engines zu den Hauptbestandteilen eines Workflow Management Systems gehören.

Die Ursprünge der Workflow Verarbeitung stammen auch laut Scribner [Scr07 S. 4] von der Dokumentenverarbeitung. Dokumente werden laufend von verschiedenen Stellen überarbeitet oder freigegeben. Sehr früh schon wollte man erreichen, dass bestimmte vordefinierte Aufgaben in Abhängigkeit vom Status eines Dokumentes eingeleitet werden.

Auch in unserem konkreten Anwendungsfall spielen Dokumente eine große Rolle. Obwohl ein Dokument in unserem Anwendungsfall nicht als solches vorliegt, sondern als binäre Datei, werden daraus Dokumente erstellt und oftmals für den Transport oder für die Archivierung verwendet.

[11] kurz WfMS

2.7 Zusammenfassung

In diesem Kapitel wurden die wesentlichen Begriffe aus dem Workflow Umfeld erklärt und zueinander in Beziehung gebracht.

Bereits jetzt kann die wesentliche Idee hinter dem Konzept „Workflow“ verstanden werden: Geschäftsprozesse sollen nicht wie bisher fest in einem Programm codiert werden, sondern in Form einer Workflow Definition vorliegen. Diese Definition wird unter Zuhilfenahme eines Workflow Management Systems interpretiert und ausgeführt.

Das magische Dreieck des Erfolges einer Unternehmung mit den drei Komponenten Zeit, Kosten und Qualität soll unter Zuhilfenahme eines Workflow Management Systems optimiert werden. Wie Abbildung 3 erkennen lässt, beeinflussen sich diese drei Steuergrößen gegenseitig.

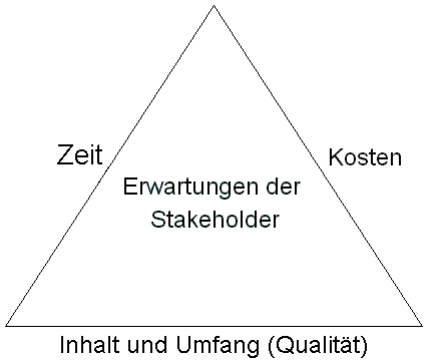


Abbildung 3 magisches Dreieck des Erfolges (Quelle: [Wik102])

Ein wesentliches Ziel ist es, mit Hilfe von Workflow Management die Durchlaufzeit der Geschäftsprozesse bei gleichbleibender oder steigender Qualität zu reduzieren.

3 Anforderungen an ein WfMS wie der Windows Workflow Foundation 4.0

„Es gibt keine größere Kraft als eine Idee, deren Zeit gekommen ist.“

Victor Hugo (1802-1885)
französischer Dichter

In diesem Kapitel werden verschiedene Anforderungen an ein Workflow Management System definiert und anschließend auf die Windows Workflow Foundation 4.0[12] projiziert. Des weiteren wird die Windows Workflow Foundation 4.0 mit weiteren bekannten Workflow Management Systemen in Bezug auf die Erfüllung firmeninterner Anforderungen verglichen. Um bewerten zu können, ob die Windows Workflow Foundation 4.0 diese Anforderungen erfüllen kann, wird im Detail auf die Architektur der WF 4.0 eingegangen. Abschließend werden anhand konkreter Einsatzgebiete für ein Workflow Management System die Vorteile der Verwendung der WF 4.0 angeführt.

3.1 Zu erfüllende Anforderungen

Im folgenden werden Anforderungen definiert, welche ein Workflow Management System erfüllen muss.

3.1.1 Firmeninterne Anforderungen

In Tabelle 2 und Tabelle 3 sind technische und nicht technische Vorgaben der Geschäftsführung und der Entwicklungsleitung an ein Workflow Management System angeführt. Diese Vorgaben müssen zwingend erfüllt werden. Für die spätere Verwendung werden diese mit T1-T10 und S1-S3 Nummeriert.

[12] Kurz: WF 4.0

Tabelle 2 firmeninterne technologische Voraussetzungen an ein WfMS

Nr	Anforderung
T1	Die Workflow Run Time muss auf dem aktuellen .NET Framework basieren. Auf JAVA basierende Lösungen scheiden aus, da dies einen Bruch in der Basistechnologie nach sich ziehen würde und zudem kein umfangreiches Domainwissen im Haus verfügbar ist.
T2	Die Oberfläche der Workflow Engine muss in eine WPF Oberfläche integriert werden können. Idealerweise basiert die Oberfläche der Workflow Engine selbst auf WPF.
T3	Als Datenbanksystem kommt ausschließlich Microsoft SQL infrage. SQL wird bereits eingesetzt. Ein zusätzliches Datenbanksystem ist somit indiskutabel.
T4	Die Installation der Workflow Run Time muss sowohl auf aktuellen Microsoft Desktopbetriebssystemen (Vista, Windows 7) als auch auf aktuellen Microsoft Serverbetriebssystemen (Windows Server 2003, Windows Server 2003 R2, Windows Server 2008) möglich sein. Auf Microsoft Windows 2000 und XP muss nicht Rücksicht genommen werden, da es sich dabei um veraltete Technologien handelt.
T5	Eine Erweiterung der Aktivitäten Bibliothek in Form von .NET Assemblies muss möglich sein.
T6	Neben einer Workflow Run Time muss auch ein grafischer Designer für die Workflow Definitionen zur Verfügung stehen. Vorzugsweise basiert dieser auf der WPF.
T7	Eine Integration in die Visual Studio 2010 IDE[13] ist wünschenswert aber nicht unbedingt notwendig.
T8	Aktuelle Literatur und technische Dokumentation zur Workflow Run Time muss vorhanden sein und vom Hersteller laufend aktualisiert werden.
T9	Schulungen für das System müssen angeboten werden. Werden zusätzlich Zertifizierungen angeboten, ist dies von Vorteil.
T10	Eine möglichst kurze Einarbeitungszeit für Entwickler ist wünschenswert.

Tabelle 3 firmeninterne nicht technologische Voraussetzungen an ein WfMS

Nr	Anforderung
S1	Es dürfen keine Lizenzkosten pro Kundeninstallation anfallen. Die Anschaffungskosten pro Entwickler sollten so gering wie möglich sein.
S2	Aufgrund internationaler Installationen muss die Oberfläche der Workflow Run Time anpassbar und internationalisierbar sein. Firmenlogos anderer Hersteller dürfen nicht aufscheinen.
S3	Wartung der Workflow Run Time muss durch den Hersteller erfolgen.

[13] Abkürzung für: Integrated Development Environment. Englisch für: Integrierte Entwicklungsumgebung

3.1.2 Systemanforderungen

Im besten Fall sollte ein Workflow Management System nach Richter [Ric04 S. 145 ff.] folgende Systemanforderungen erfüllen:

- **Erweiterbarkeit**: Dadurch wird garantiert, dass unterschiedliche Abläufe unterstützt werden. Des weiteren ist es die Voraussetzung dafür, dass auch spätere Anforderungen bearbeitet werden können.
- **Anpassbarkeit**: Die dynamische Anpassbarkeit stellt sicher, dass rasch auf aktuelle Bedürfnisse und Änderungen reagiert werden kann. Im Idealfall können Abläufe dynamisch angepasst werden.
- **Wiederverwendbarkeit**: Wiederverwendbarkeit fordert, dass ein Workflow bereits vorhandene Komponenten (z. B. Aktivitäten) verwenden kann, bzw. dass ein Workflow selbst in einem übergeordneten Workflow verwendet wird.
- **Offenheit**: Bereits existierende Hard- und Softwaresysteme, welche für die Ausführung eines Geschäftsprozesses notwendig sind, können in einem offenen System verwendet werden.
- **Skalierbarkeit**: Skalierbarkeit fordert, dass auch eine hohe Anzahl an Workflow Instanzen effizient ausgeführt werden.

3.1.3 Forderungen nach Trennung in Run Time und Build Time

Nach Hollingsworth [Hol95 S. 6 ff.] müssen alle Workflow Management Systeme auf höchster Ebene folgende drei Funktionsbereiche aufweisen:

Tabelle 4 Funktionsbereiche für Workflow Management Systeme nach Hollingsworth

Ebene	Beschreibung
Entwicklungsebene	In der Entwicklungsebene wird ein Geschäftsprozess analysiert und in eine Form umgewandelt, welche von Maschinen interpretiert werden kann. Das Ergebnis dieser Umwandlung ist die Workflow Definition, welche vom Workflow Management System interpretiert und ausgeführt werden kann.
Laufzeitkontrollebene	In der Laufzeitkontrollebene werden aus der Workflow Definition konkrete Workflow Instanzen, und aus Aktivitäten konkrete Aktivität Instanzen gebildet, zur Ausführung gebracht und während der Ausführung kontrolliert bzw. überwacht. Wesentlich in dieser Ebene

	ist die Workflow Engine. Diese ist für die Bildung und Terminierung der Instanzen verantwortlich. Auch der Aufruf von Anwendungen fällt in Ihren Aufgabenbereich.
Laufzeitinteraktionsebene	Die Laufzeitinteraktionsebene ist zuständig für die Kommunikation mit dem Anwender. Vor allem manuelle Aktivitäten kommunizieren in dieser Ebene mit dem Anwender.

Aus Abbildung 4 wird die Aufteilung nach Build Time und Run Time sowie die Zuordnung der drei Hauptebenen ersichtlich.

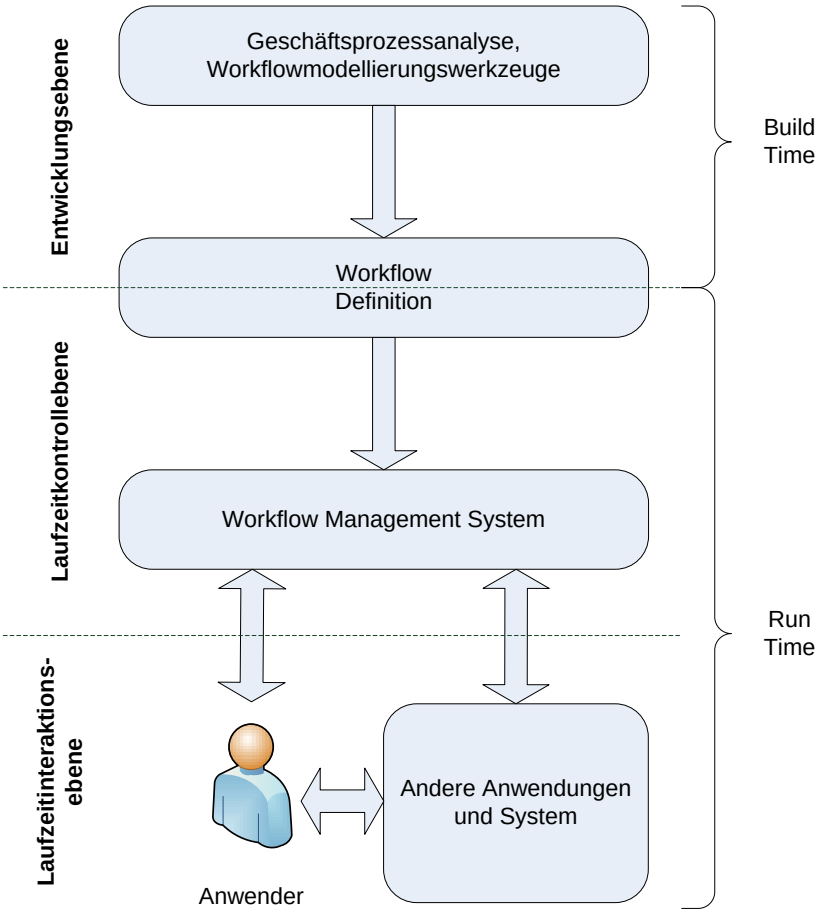


Abbildung 4 Funktionsbereiche eines Workflow Management Systems (Quelle: eigene Darstellung in Anlehnung an (Schattauer, 2008 S. 25), (Richter-von Hagen, et al., 2004 S. 141) und (Hollingsworth, 1995 S. 7))

Im Zuge der Build Time wird ein Geschäftsprozess analysiert und daraus wird ein Workflow modelliert. Ergebnis der Build Time ist eine maschinell verwendbare

Workflow Definition. Diese Workflow Definition wird mittels der Run Time zur Ausführung gebracht. Die Run Time bedient sich ggf. anderer Anwendungen und Informationssysteme. Auch die Kommunikation mit dem Anwender erfolgt nur innerhalb der Run Time. [Sch08 S. 25]

Im Zuge des Entwurfes und der Modellierung eines Geschäftsprozesses finden auch Simulationswerkzeuge und Analysewerkzeuge oftmals ihren Einsatz in der Build Time. [Ric04 S. 141 ff.]

3.1.4 Abbildung von Geschäftsprozessen in Form einer Workflow Definition

Geschäftsprozesse (oder Teile davon) müssen in Form einer Workflow Definition hinterlegt werden können. Eine Workflow Definition ist das Ergebnis der Build Time und liegt in Form einer (deklarativen) Beschreibung vor. Dies wurde bereits unter Punkt 2.2 (Seite 9 ff.) erörtert.

Eine Workflow Definition dient als Schnittstelle zwischen der Build Time und der Run Time eines Workflow Management Systems. [Sch08 S. 26]

3.1.5 Instanzen müssen über innere Zustände verfügen

Bereits unter Punkt 2.4 (Seite 13 ff.) wurde erwähnt, dass eine Instanz über innere Zustände verfügen muss. Der innere Zustand einer Workflow Instanz spiegelt den Fortschritt bei der Abarbeitung der enthaltenen Aktivitäten wieder und repräsentiert gleichzeitig den aktuellen Systemzustand. Folgende mögliche Zustände führt die Workflow Management Coalition an: [Wor99 S. 47]

Tabelle 5 mögliche Zustände für Workflow Instanzen laut WfMC

Zustand	**Beschreibung**
Initialisiert	Die Workflow Instanz wurde erzeugt. Die Startbedingung ist jedoch noch nicht erfüllt.
Laufend	Die Workflow Instanz befindet sich in Ausführung und eine oder mehrere Aktivitäten können, in Abhängigkeit von Ihren Startbedingungen, gestartet sein.
Aktiv	Von einer oder mehreren Aktivitäten existieren konkrete Aktivität Instanzen, welche auch bereits gestartet sind.

Unterbrochen	Die Workflow Instanz befindet sich in einer Art Ruhezustand. In diesem Zustand werden keine weiteren Aktivitäten gestartet.
Beendet	Die Workflow Instanz hat die Ende Bedingung erreicht. Es werden lediglich noch diverse abschließende System Aktivitäten (z. B. Protokollierungen) ausgeführt.
Abgebrochen	Die Ausführung der Workflow Instanz wurde aufgrund eines Fehlers oder eines Benutzereingriffes abgebrochen.
Archiviert[14]	Die Workflow Instanz wurde in einen unbegrenzt anhaltenden Archivierungszustand versetzt und verweilt in diesem, bis die Instanz wieder aufgenommen wird. Vor allem bei lang laufenden Workflows wird von dieser Technik Gebrauch gemacht.

3.2 Windows Workflow Foundation 4.0

Die Windows Workflow Foundation 4.0 ist Bestandteil des Microsoft .NET Frameworks 4.0 und ergänzt dieses um Programmbibliotheken, eine Workflow Engine und Werkzeuge zur Erstellung Workflow basierter Anwendungen. Microsoft adressiert damit Software Entwickler, welche Geschäftsprozesse in eigenen Anwendungen integrieren möchten. Die Windows Workflow Foundation repräsentiert dabei kein fertiges Produkt oder gar ein voll umfängliches Workflow Management System. Es werden jedoch in den Programmbibliotheken die grundlegenden Bestandteile eines Workflow Management Systems, wie z. B. eine Workflow Engine, Basis Aktivitäten und Werkzeuge zur Erstellung einer Workflow Definition bereitgestellt. [Sch08 S. 49]

Die Windows Workflow Foundation 4.0 benötigt stets eine Host[15] Anwendung. Aufgabe der Host Anwendung ist es u. a. den gewünschten Workflow zu starten, zu persistieren und auch abzubrechen. Als Host Anwendung kann jede beliebige .NET Anwendung verwendet werden. Damit kann eine Host Anwendung mit einem „*virtuellen Betriebssystem*“ [Dob09 S. 84] aus Sicht des Workflows verglichen werden.

[14] Synonym: persistiert
[15] Englisch für: Gastgeber. Anmerkung: Dies scheint in diesem Fall die passendste Übersetzung zu sein.

Die folgende Abbildung 5 zeigt die Bestandteile der Windows Workflow Foundation 4.0.

Abbildung 5 Bestandteile der Windows Workflow Foundation 4.0 (Quelle: [Jac10])

Tabelle 6 Bestandteile der Windows Workflow Foundation 4.

Bestandteil	**Beschreibung**
Workflow	Besteht aus Activities. Er besitzt eine Startbedingung und eine Endebedingung.
Activity Library	Eine Sammlung an Standard Aktivitäten[16], welche um eigene Activities erweitert werden kann.
WF Runtime	Die eigentliche Run Time, welche für die Ausführung des Workflows zuständig ist.
Extensions	Mittels Extensions kann der Funktionsumfang der Run Time erweitert werden. Persistence[17] und Tracking[18] sind bereits zwei Standard Extensions.
Host	Beinahe jede .NET 4.0 Anwendung kann als Host Anwendung fungieren.

[16] Die englische Bezeichnung lautet „Basic Activity Library“ (kurz: „BAL“)
[17] Englisch für: Persistenz
[18] Englisch für: Folgen - auch im Sinne von Fehlersuche

VS[19] Designer	Ein Workflow Designer integriert sich in die Visual Studio Entwicklungsumgebung.
VS Debugger	In der Visual Studio Umgebung können Workflows schrittweise abgearbeitet werden.
Rehosted Designer	Auf einfache Art und Weise kann die eigene Applikation um einen auf WPF basierenden Workflow Designer erweitert werden.

Die erste Version der Windows Workflow Foundation wurde mit .NET Framework 3.0[20] ausgeliefert.

Wie aus der folgenden Abbildung 6 zu erkennen ist, besteht das aktuelle .NET Framework 4.0 grundlegend aus dem .NET Framework 2.0 und den erweiterten Foundations[21].

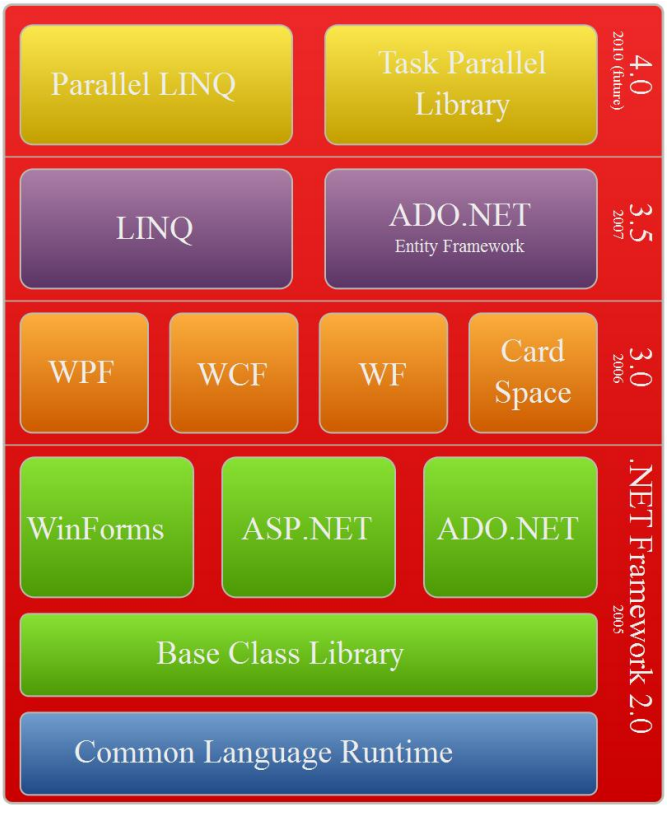


Abbildung 6 Der .NET Framework Stack (Quelle: [Wik10])

[19] Abkürzung für: für Visual Studio; Visual Studio ist eine von der Firma Microsoft integrierte Entwicklungsumgebung.
[20] Die Auslieferung erfolgte im November 2006. Der Codename lautete: Win FX
[21] Englisch für: Fundament

Eine dieser Foundations ist die Windows Workflow Foundation 4.0.

Microsoft hat angedeutet, dass die Windows Workflow Foundation in Zukunft einen Eckpfeiler der Service Oriented Architecture (kurz „SOA“) bilden wird. [Sur10 S. 23]

Die hochgesteckten Erwartungen, welche Microsoft bereits in die Windows Workflow Foundation 3.x steckte, wurden jedoch leider nicht erfüllt. Laut Mackey [Mac10 S. 133] sind dafür u. a. folgende Faktoren verantwortlich zu machen:

- Der Workflow Designer konnte mittels dem angebotenen API[22] nur sehr umständlich in vorhandene Anwendungen integriert werden.
- Die Ausführungsgeschwindigkeit war sehr schlecht.
- Das Erstellen eigener Workflow Activities war komplizierter als es hätte sein müssen.
- Der Austausch von Informationen und Daten zwischen Activities war sehr mühsam.
- Unterstützung für Nachrichtenaustausch und die Integration der Windows Communication Foundation war nur beschränkt vorhanden.
- Das Hosting Model war für zahlreiche Entwickler sehr verwirrend.
- Der Workflow Designer wurde als umständlich empfunden.

Mit Einführung des .NET Framework 3.5 wurden keine nennenswerten Erweiterungen an der Windows Workflow Foundation ausgeliefert.

Auch Chappel kommt zu dem Schluss, dass diese ersten beiden Versionen der Windows Workflow Foundation zwar funktional waren, sie sich jedoch nicht als Mainstream Technologie behaupten konnten. [Cha09 S. 3]

Um die Akzeptanz sowohl bei Entwicklern als auch bei Anwendern zu steigern, wurde mit dem Erscheinen des .NET Framework 4[23] die Workflow Foundation grundsätzlich überarbeitet. [Mil09].

[22] Abkürzung für: Application Programming Interface; Englisch für: Programmierschnittstelle
[23] Das .NET Framework 4.0 ist am 12. April 2010 erschienen.

Im speziellen führt Milner [Mil09] folgende Änderungen an:

- Der Workflow Designer wurde komplett überarbeitet. Hauptaugenmerk dabei wurde auf Gebrauchstauglichkeit und Leistung gelegt. Der Designer ist nun in der Lage, auch sehr umfangreiche Workflow Definitionen zu bearbeiten. Da der Designer selbst auf WPF basiert, kann er leicht in eigene WPF Oberflächen integriert werden.
- Mit der Workflow Foundation 4.0 wurde die Verwendung von Variablen und Argumenten in Workflows überarbeitet und vereinheitlicht.
- Eine Flowchart[24] Activity ist neu hinzugekommen. Diese Activity ermöglicht Verzweigungen (auch zu früheren Activities) und Konstrukte wie „Switch / Case“.
- Das Programmiermodel wurde komplett überarbeitet und ist dadurch einfacher und robuster geworden. Die Activity ist somit in den Mittelpunkt des Programmiermodels gerückt. Die Activity kann sowohl für die Darstellung von eigentlichen Activities als auch von Workflows verwendet werden. Ferner ist das Programmiermodel nun vollständig deklarativ. Innerhalb einer WorkflowDefinition kann somit kein ausführbarer Programmcode mehr verwendet werden. Damit einhergegangen ist die Steigerung der Performance bei der Ausführung von Workflows um bis zu 100 %. [Dob09 S. 90]
- Die Integration der Windows Communication Foundation wurde wesentlich verbessert. Neue Activities zum Senden, Empfangen und Bestätigen von Nachrichten, welche alle auf der Windows Communication Foundation beruhen, wurden ergänzt.

Als weitere Änderung führt Collins den Wegfall des State Machine[25] Workflows an. [Col10 S. 22]

Wobei es laut Mackey durchaus möglich ist, dass die State Machine Workflows zu einem späteren Zeitpunkt wieder durch Microsoft ergänzt werden. [Mac10 S. 139]

[24] Englisch für: Flussdiagramm, Ablaufplan
[25] Englisch für: Zustandsmaschine, endlicher Automat

In Abbildung 7 ist die Architektur der Windows Workflow Foundation 4.0 dargestellt. Wie daraus zu erkennen ist, bildet die WorkflowApplication[26] den zentralen Bestandteil in dieser Architektur und repräsentiert eine konkrete Workflow Instanz. Für jede Instanz eines Workflows wird eine eigene Instanz eines Objektes vom Type WorkflowApplication erzeugt. Alle Instanzen arbeiten unabhängig voneinander die enthaltenen Activities ab. [Col10 S. 460]

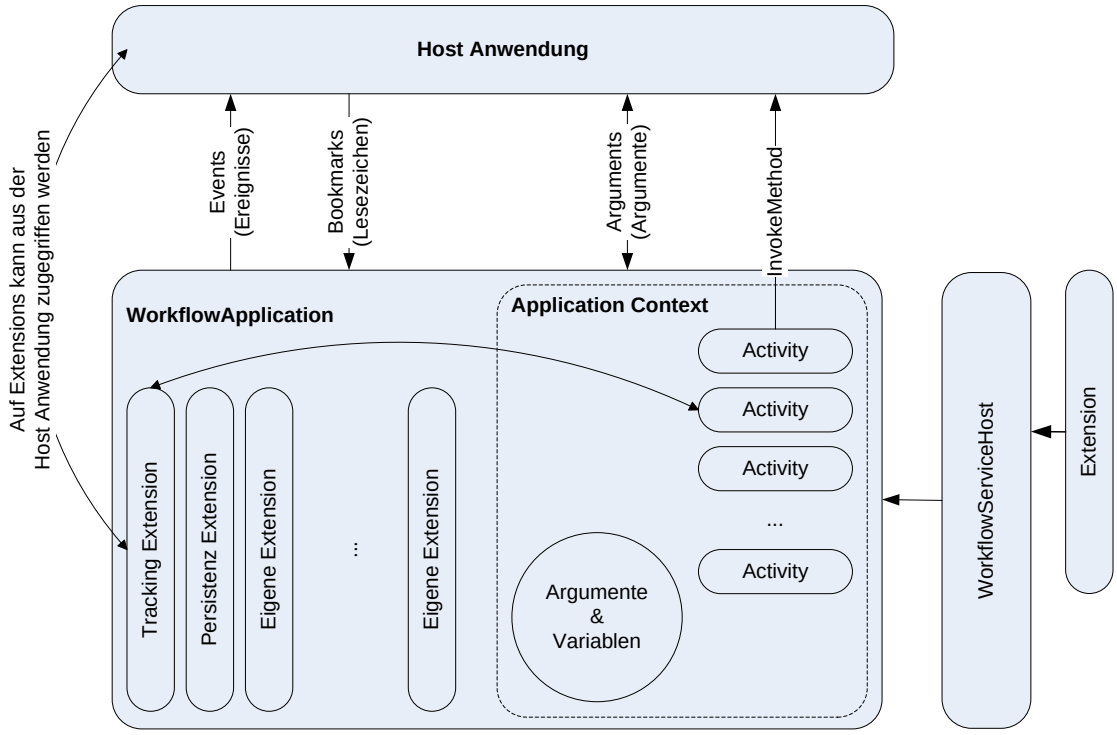


Abbildung 7 Architektur der Windows Workflow Foundation 4.0 (Quelle: eigene Darstellung in Anlehnung an [Col10 S. 460])

3.2.1 Activities, Workflow Instanzen und Variablen/Argumente

Eine WorkflowApplication führt eine Collection[27] an Activities aus. Die Definition entsprechender Activities nimmt einen Großteil der Entwicklungszeit eines Workflows in Anspruch. Die Windows Workflow Foundation 4.0 bietet bereits ein Set an Standard Activities in Form der Basic Activity Library an. Zusätzlich dazu können auf einfache

[26] Klasse aus dem System.Activities Namespace welche eine Reihe von Methoden und Eigenschaften zur Verwaltung von Aktivitäten in einer Workflowinstanz bietet.
[27] Englisch für: Sammlung

Art und Weise eigene Activities[28] erstellt und verwendet werden. Die Windows Workflow Foundation 4.0 stellt zudem einen Designer bereit, der es erlaubt, die Activities auf einfache Art und Weise grafisch zu verknüpfen. Ausgeführt werden die Activities in dem Kontext einer Anwendung[29], welche den Zustand dieser Workflow Instanz beinhaltet. Ein besonderes Merkmal der Windows Workflow Foundation 4.0 stellt die explizite Definition von Variablen und Argumenten dar. Argumente unterscheiden sich dahingehend von Variablen, dass Argumente an einen Workflow übergeben werden und auch retour geliefert werden können. Variablen hingegen besitzen einen beschränkten Gültigkeitsbereich. Wird ein Workflow in einen beständigen Speicher persistiert (z. B. in einer SQL[30] Datenbank), so gilt dies auch für den Inhalt der Argumente und Variablen. Activities sind stets zustandslos. Der Zugriff auf Argumente und Variablen erfolgt über den Kontext der Anwendung. Die Ausführung von Activities kann von Regeln abhängig gemacht werden. Diese Regeln manifestieren sich in der Workflow Definition z. B. in Form von Verzweigungen. Somit ist es nicht zwingend erforderlich, dass zwei unterschiedliche Instanzen eines Workflows denselben Weg durch die Activities vollziehen. Ein Datenaustausch zwischen verschiedenen Instanzen eines Workflows ist nicht vorgesehen, da jede Instanz für sich unabhängig von anderen Instanzen ausgeführt wird. [Col10 S. 461 ff.]

3.2.2 Extensions

Obwohl die Activities in einer bestimmten Reihenfolge ausgeführt werden, bieten Extensions[31] allen Activities spezielle Dienste an. Auf Extensions können sowohl Activities als auch die Host Anwendung zugreifen. Im allgemeinen werden Extensions von der Host Anwendung erzeugt, konfiguriert und anschließend der WorkflowApplication hinzugefügt. Die Activities können dann wie erwartet auf diese bereitgestellten Extensions zugreifen. [Col10 S. 461 ff.]

---

[28] Sogenannte Custom Activities
[29] Sogenannter Application Context
[30] Abkürzung für: „Structured Query Language“; einer Datenbankabfrage- und -manipulationssprache für relationale Datenbanksysteme.
[31] Englisch für: Erweiterung

Eine spezielle Extension stellt das Objekt InstanceStore[32] dar. Diese Extension ist zuständig für das Persistieren des aktuellen Zustandes eines Workflows und natürlich auch für das Wiederherstellen des Zustandes bei Bedarf. Diese Fähigkeit, kombiniert mit den zustandslosen Activities, verleiht der Windows Workflow Foundation eine sehr hohe Skalierbarkeit. Eine WorkflowApplication wird erzeugt, um die konkrete Instanz eines Workflows auszuführen. Wird diese Instanz persistiert, so kann auch die Instanz der WorkflowApplication aus dem Speicher entfernt werden. Nur aktive Instanzen belegen demnach Systemressourcen. [Col10 S. 461 ff.]

Eine weitere vordefinierte Erweiterung wird verwendet, um den Ablauf in einem Workflow nachzuvollziehen. Diese Tracking Extension wird ebenfalls der WorkflowApplication hinzugefügt und empfängt dadurch entsprechende Tracking Ereignisse. In Abhängigkeit von der konkreten Ausprägung der Tracking Extensions werden mit diesen Ereignissen unterschiedliche Aktionen ausgeführt. [Col10 S. 461 ff.]

Natürlich können auch eigene Extensions erstellt werden. Dies ist immer dann zu empfehlen, wenn ein Zugriff auf/von mehreren Activities erfolgen soll. [Col10 S. 461 ff.]

3.2.3 Host Anwendung

Die konkrete Instanz eines Objektes vom Typ WorkflowApplication muss in einer Host Anwendung ausgeführt werden. Die eigene Anwendung kann eine Workflow Run Time bereitstellen. Die Host Anwendung muss ein Objekt vom Type WorkflowApplication instanziieren, die gewünschten Extensions erzeugen und bekannt machen und abschließend den Workflow starten.

[32] Englisch für: Instanz Speicher

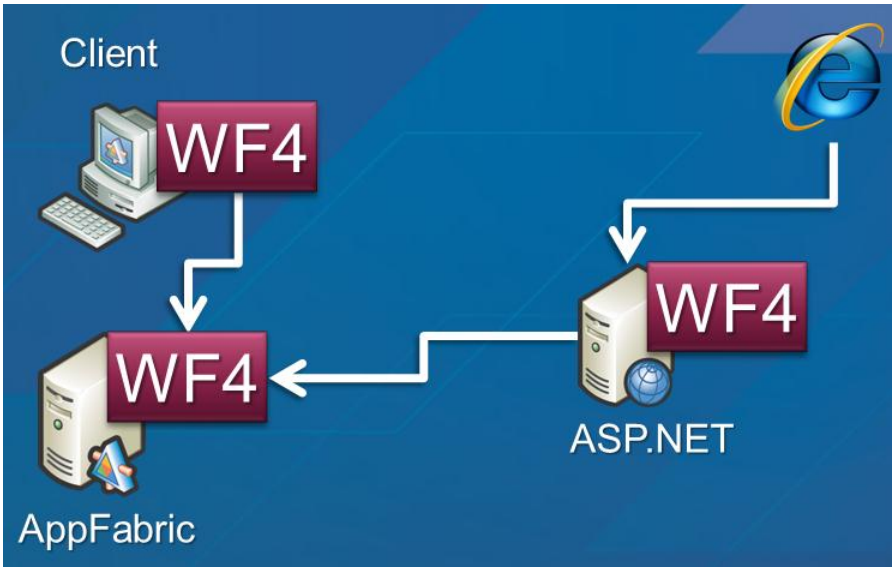


Abbildung 8 Jede auf der CLR basierte Anwendung kann als Host Anwendung für einen Workflow dienen. (Quelle: [Jac10])

In Abhängigkeit von der Anwendung kann es notwendig sein, dass der Workflow erst beim Auftreten eines Ereignisses gestartet wird – z. B. nach dem Eintreffen einer Nachricht oder einer Anfrage. In solchen Situationen empfiehlt es sich, ein Objekt vom Type WorkflowServiceHost[33] zu verwenden. Dieser kann WCF Nachrichten empfangen und eine Instanz eines WorkflowApplication Objektes erstellen, um auf diese Nachricht zu reagieren. Vorbereitend muss der WorkflowServiceHost mit den gewünschten Extensions konfiguriert werden. [Col10 S. 461 ff]

Auf folgende Arten kann die Host Anwendung mit einem Workflow kommunizieren:

- Argumente können einem Workflow übergeben werden bzw. als Antwort erhalten werden.
- Die Host Anwendung kann auf Ereignisse reagieren, welche durch die Workflow Instanz ausgelöst werden.
- Der Workflow kann mittels einer InvokeMethod Activity eine Methode in der Host Anwendung aufrufen.

[33] Stellt den Host für workflowbasierte Dienste bereit. Ist im Namespace „*System.ServiceModel*“ zu finden.

- Die Host Anwendung kann eine Workflow Instanz fortsetzen, indem auf ein Bookmark[34] reagiert wird.
- Generische Custom Activities können Objekte retournieren.
- Extensions können im Host erzeugt und in den Activities verwendet werden (und umgekehrt).

[34] Englisch für: Lesezeichen

3.3 Untersuchung bekannter WfMS hinsichtlich der Erfüllung interner Anforderungen

Die Windows Workflow Foundation 4.0 ist nicht die einzige auf dem Markt verfügbare Workflow Lösung für Microsoft Windows Betriebssysteme.

In Tabelle 7 sind jene weiteren Produkte dargestellt, zu denen im Vorfeld Informationen eingeholt wurden. Mit den Produkten Windows Workflow Foundation 3 & 3.5 (P1) und Microsoft BizTalk Server (P2) wurden zudem praktische Erfahrungen gesammelt.

Tabelle 7 Bekannte Workflow Management Systeme

Nr	**Produkt**
P1	Windows Workflow Foundation 3 & 3.5
P2	Microsoft BizTalk Server
P3	Staffware Process Suite
P4	IBM WebSphere MQ Workflow
P5	FLOWer
P6	COSA
P7	Sun ONE iPlanet Integration Server
P8	SAP Business Workflow
P9	FileNet P8 BPM Suite
P10	WebSphere Process Server
P11	Oracle BPEL Process Manager

Unter Punkt 3.1.1 (Seite 16 ff.) sind die firmeninternen Forderungen an ein Workflow Management System angeführt. Dabei handelt es sich um sowohl um technische als auch um wirtschaftliche Anforderungen.

Die folgende Tabelle 8 beinhaltet das Ergebnis einer Voranalyse bekannter WfMS. In einer Matrix sind die firmeninternen Anforderungen den Workflow Management Systemen gegenüber gestellt.

Tabelle 8 Matrix firmeninterner Anforderungen an ein WfMS und bekannter WfMSs

	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9	P 10	P 11	WF 4.0
T1	☒	☑	☑	☒	☑	☒	☒	☒	☒	☒	☒	☑ [Mac10 S. 133 ff.]
T2	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑ [Mil09]
T3	☑	☑	☑	☑	☑	☑	☑	☑	☒	☑	☒	☑ [Col10 S. 177 ff.]
T4	☑	☒	☑	☑	☑	☑	☒	☑	☑	☑	☒	☑ [Mic1001]
T5	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑ [Col10 S. 79]
T6	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑ [Mil09]
T7	☑	-	-	-	-	-	-	-	-	-	-	☑ [Col10 S. 4]
T8	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑ [Mic1009]
T9	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
T10	☑	☒	☑	☑	☑	☑	☒	☒	☑	☑	☒	☑
S1	☑	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☑
S2	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
S3	☒	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

☑…Bedingung erfüllt; ☒…Bedingung nicht erfüllt; -…nicht zutreffend

Aufgrund der Voraussetzungen, welche in Tabelle 2 und Tabelle 3 (Seite 16 ff.) angeführt sind, scheiden die Produkte P1 bis P11 aus! Einzig und allein die Windows Workflow Foundation 4.0 erfüllt sämtliche technische und allgemeine Grundvoraussetzungen, welche firmenintern an ein Workflow System gestellt werden. Aufgrund dieser Erkenntnis wurde einer näheren Beschäftigung mit den angeführten Workflow Management Systemen (P1 bis P11) aus Tabelle 7 nicht nachgegangen.

3.4 Vertiefte Untersuchung der WF 4.0 hinsichtlich der Erfüllung gestellter Anforderungen an ein WfMS

Es folgt eine Analyse, in welchem Umfang die Windows Workflow Foundation 4.0 die Anforderungen an ein Workflow Management System aus Punkt 3.1 (Seite 16 ff.) erfüllt. Die Erfüllung der firmeninternen Anforderungen laut Punkt 3.1.1 wurde im vorherigen Punkt 3.3 bereits diskutiert.

3.4.1 Erfüllung der Systemanforderungen durch die WF 4.0

In Tabelle 9 ist festgehalten, in welchem Umfang die Systemanforderungen aus Punkt 3.1.2 (Seite 18), welche Richter an ein WfMS stellt, durch die WF 4.0 abgedeckt werden.

Tabelle 9 Systemanforderungen an ein WfMS in Bezug auf die WF 4.0

Systemanforderung	**Beschreibung**
Erweiterbarkeit	Die Windows Workflow Foundation 4.0 basiert auf dem .NET Framework und kann mittels Custom Activities[35] erweitert werden. [Col10 S. 80] Nachdem Microsoft die Windows Workflow Foundation als Eckpfeiler für eine neue Plattform betrachtet, kann davon ausgegangen werden, dass auch in zukünftigen Versionen Kompatibilität garantiert ist. [Sur10 S. 23]
Anpassbarkeit	Workflows können von außen mittels Argumenten gesteuert werden. Argumente können jeden beliebigen Typs sein. Variablen können innerhalb eines Workflows für die Zwischenspeicherung von Werten und auch für Verzweigungen verwendet werden. [Mac10 S. 140] Workflows können auf zwei unterschiedliche Arten definiert werden: entweder in Form einer XAML-Datei oder in Form von Programmcode. In beiden Fällen kann eine dynamische Anpassbarkeit erreicht werden. XAML-Code kann auch von Code Generatoren automatisiert erzeugt werden. [Mil09]

[35] Englisch für: kundenspezifisch, individuell gefertigt.

Wiederverwendbarkeit	Ein Workflow ist eine Aktivität, die mehrere Kindaktivitäten enthalten kann. [Dob09 S. 86] Eigene Aktivitäten mit beliebiger Komplexität können in Form von Custom Activities erstellt und in Workflows verwendet werden. [Cha09 S. 14]
Offenheit	Aktivitäten beinhalten Objekte, welche auf der CLR basieren. Aus CLR Objekten kann mittels COM auch auf ältere Applikationen zugegriffen werden. [Dor05 S. 3 ff.] Hard- und Software, welche durch CLR bzw. COM unterstützt wird, kann somit für die Ausführung der Geschäftsprozesse verwendet werden.
Skalierbarkeit	Die Möglichkeit, Workflows in einen persistenten Zustand versetzen zu können, trägt zur Skalierbarkeit bei. Nur aktive Workflows belegen damit Ressourcen. Dobric bezeichnet dies auch als „*Hibernate*[36]*-Zustand*“ [Dob09 S. 84]. Im Falle einer ASP.NET[37] Anwendung kann mittels AppFabric[38] eine Lastverteilung erreicht werden. [Cha09 S. 23] Jede gestartete Workflow Instanz wird in einem eigenen Thread ausgeführt. Dies trägt ebenfalls zur Skalierbarkeit bei. [Dob09 S. 86]

Die Anforderungen, welche Richter an ein Workflow Management System stellt, können somit als erfüllt betrachtet werden.

[36] Englisch für: Winterschlaf (abhalten)

[37] Bei ASP.NET (Active Server Pages .NET) handelt es sich um eine serverseitige Technologie von Microsoft zum Erstellen von Webanwendungen auf Basis des .NET Framework.

[38] Dabei handelt es sich um eine Betriebsumgebung für .NET Anwendungen. Diese stellt Dienste wie Caching (Zwischenspeicherung), Zugriffskontrolle, Anwendungskopplung, Management und Überwachung zur Verfügung.

3.4.2 Trennung in Run Time und Build Time bei der WF 4.0

In Tabelle 10 ist dargestellt, in welchem Umfang die Windows Workflow Foundation 4.0 der Anforderung nach Trennung in Build Time und Run Time gerecht wird.

Tabelle 10 Umsetzung von Build Time und Run Time in der Workflow Foundation 4.0

Ebene	**Umsetzung in der Workflow Foundation 4.0**
Entwicklungsebene	Für die Windows Workflow Foundation können Workflow Definitionen auf zwei verschiedene Arten vorliegen: • Als XAML-Code • In Form von „*Coded Workflows*“ [Col10 S. 23] Beide Varianten liefern als Ergebnis die Definition eines Workflows, welche vom Workflow Management System zur Ausführung gebracht werden kann. [Col10 S. 5 ff.]
Laufzeitkontrollebene	Auch bei der Windows Workflow Foundation werden aus der Workflow Definition im Zuge der Ausführung konkrete Instanzen von Workflows und Activities gebildet. [Mil09] Anwendungen können nur aus Activities heraus aufgerufen werden. [Col10 S. 21]
Laufzeitinteraktionsebene	Für die Kommunikation mit Diensten sind eigene Activities in der Basic Activity Library vorhanden. Diese sind für den Versand und Empfang von Nachrichten bzw. deren Bestätigungen ausgelegt. Die Kommunikation mit Diensten erfolgt in der Regel mittels WCF. [Col10 S. 95] Eine Kommunikation bzw. Interaktion mit dem Anwender (z. B. über die Hostanwendung) wird ebenfalls unterstützt. [Col10 S. 123 ff.]

Nachdem keiner dieser drei Punkte durch die Windows Workflow Foundation 4.0 verletzt wird, muss die Anforderung nach Trennung in Build Time und Run Time als erfüllt betrachtet werden.

3.4.3 Abbilden von Geschäftsprozesse in Form einer Workflow Definition

Jacobs stellt in Abbildung 9 drei mögliche Fälle für die Ausführung von Befehlen dar.

Abbildung 9 Von native Code zu Wf4 (Quelle: [Jac10])

Tabelle 11 Befehlsebenen

Ebene	Beschreibung
Instruction, Native Code	Native Code basiert auf der Ausführung von Befehlen bzw. Instruktionen. Beispiele dafür sind z. B. (veraltete) MS-DOS Anwendungen oder auch aktuelle C++ Applikationen, welche auf das .NET Framework verzichten.
Verwalteter CLR[39] Code	Verwalteter Code basiert auf Objekten. De facto ist bei der Verwendung der CLR alles ein Objekt. Objekte beinhalten natürlich letztendlich auch Instruktionen. Aus zahlreichen Gründen, z. B. Wiederverwendbarkeit, sind diese Instruktionen aber in Objekten gekapselt.
Wf4	Die Windows Workflow Foundation 4.0 hingegen basiert auf Activities. Die kleinste Ausführungseinheit innerhalb eines Workflows ist die Activity. Konkret handelt es sich dabei um ein Object aus dem Namespace `System.Activities`. Activities bestehen aus Objekten und diese wiederum aus Instruktionen. Es wurde somit eine weitere Schicht der Abstraktion eingeführt.

[39] Die Common Language Runtime (CLR) ist die Laufzeitumgebung von .NET und stellt somit den Interpreter für den standardisierten Zwischencode, der Common Intermediate Language (CIL), dar.

Auch Dobric definiert die Aktivitäten als kleinste „*semantische Elemente*“ in einer Workflowanwendung. [Dob09 S. 86]

Durch diese erneute Kapselung von Objekten in Activities erreicht man einen höheren Grad an Abstraktion. Erst dieser hohe Abstraktionsgrad führt dazu, dass ein Workflow vollkommen deklarativ beschrieben werden kann und dazu ein extrem hohes Maß an Agilität aufweist. [Jac10]

Im Gegensatz zur imperativen Programmierung, bei der das **Wie** im Vordergrund steht, fragt man in der deklarativen Programmierung nach dem **Was**, das berechnet werden soll. [Bei09]

Oder anders ausgedrückt:

- Die Aufgabe von Software ist es, Geschäftsprobleme zu lösen.
- Die Aufgabe von Code ist es, die EDV Lösungen dafür anzubieten.

Ein wesentliches Ziel der Windows Workflow Foundation 4.0 ist eine Agilität über vier Dimensionen: [Jac10]

Diese sind:

- Thread
- Prozess
- Computer
- Zeit

Dank des hohen Abstraktionsgrades kann Arbeit dann ausgeführt werden, wenn entsprechende Daten bereit sind, und zwar unter Verwendung gerade zur Verfügung stehender Ressourcen.

Die Windows Workflow Foundation 4.0 ermöglicht es zudem, dass Workflow Definitionen sowohl in Form einer XAML Datei als auch in Form einer Programmiersprache wie z. B. C# oder VB.NET erstellt werden können. Als Vorteil erwähnt Schattauer die Tatsache, dass XAML Dateien nur interpretiert und nicht kompiliert werden müssen. [Sch08 S. 59] XAML Dokumente könnten somit auch zur Laufzeit von anderen Systemen erstellt werden.

Der deklarative Ansatz ist demnach vollständig in der Windows Workflow Foundation 4.0 umgesetzt. Das Kriterium aus Punkt 3.1.4 ist somit erfüllt. Weiters wurden an dieser Stelle die Folgen der deklarativen Agilität aufgezeigt.

3.4.4 Instanzen verfügen über innere Zustände

Unter Punkt 3.1.5 (Seite 20 ff) wurde definiert, über welche inneren Zustände eine Workflow Instanz verfügen sollte.

Die folgenden Zustände werden als empfohlen angeführt:

- Initialisiert
- Laufend
- Aktiv
- Unterbrochen
- Beendet
- Abgebrochen
- Archiviert

Der `Activity` Type verfügt über folgende Ereignisse: [Mic1008]

Tabelle 12 Ereignisse des Activity Types

Ereignis	**Beschreibung**
Canceling	Tritt auf, wenn die Ausführung der Activity abgebrochen wird.
Closed	Tritt ein, wenn die Ausführung einer Activity abgeschlossen wurde.
Compensating	Tritt beim Ausführen einer Kompensierungsmethode für eine Activity auf.
Executing	Tritt beim Ausführen einer Activity auf.
Faulting	Tritt ein, wenn während der Ausführung eine Ausnahme auftritt.
StatusChanged	Tritt bei jeder Statusänderung auf.

In Abbildung 10 ist der Zustandsgraph einer Activity dargestellt. Klar zu erkennen sind dabei die möglichen Übergänge der Stati.

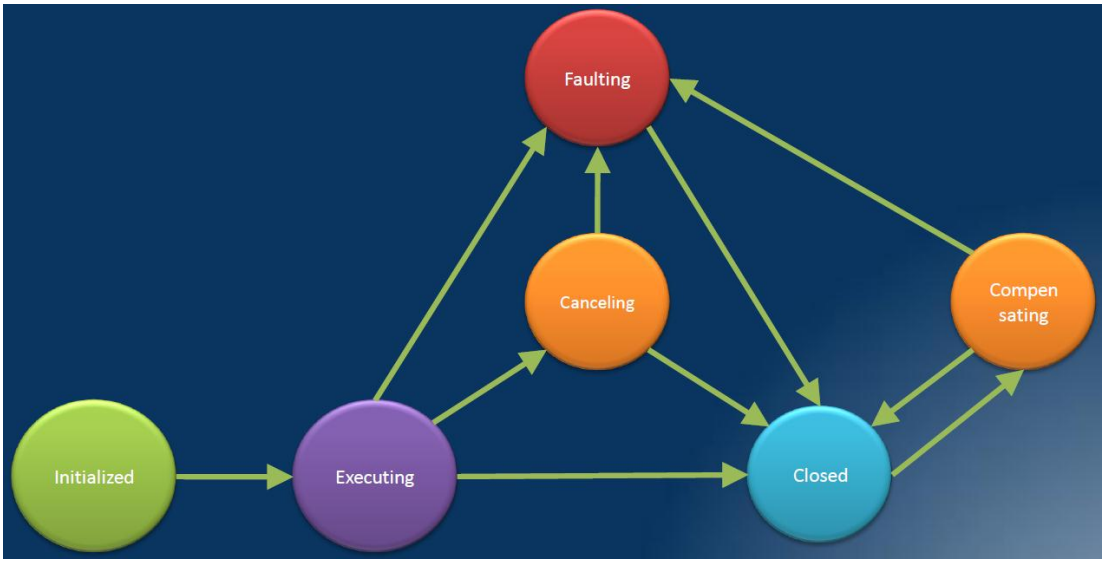


Abbildung 10 Zustandsgraph einer Activity (Quelle: [McL09])

Die Klasse WorkflowApplication hingegen verfügt über folgende Ereignisse: [Mic10]

Tabelle 13 Ereignisse des WorkflowApplication Objektes

Ereignis	**Beschreibung**
Aborted	Wird aufgerufen, wenn der Workflow abgebrochen wird.
Completed	Wird aufgerufen, wenn der Workflow abgeschlossen wird. Als weitere Unterscheidung wird ein Completed noch unterteilt in: • Canceled: Die Workflow Instanz wurde abgebrochen. • Closed: Die Workflow Instanz wurde normal beendet. • Executing: Die Workflow Instanz ist noch in Ausführung. • Faulted: Die Workflow Instanz wurde mit einem Fehler beendet.
Idle	Wird aufgerufen wenn die aktuelle Workflowinstanz betriebsbereit ist aber auf einen Befehl wartet.
OnUnhandledException	Wird aufgerufen, wenn im aktuellen Workflow eine unbehandelte Ausnahme auftritt.
PersistableIdle	Wird aufgerufen, wenn die aktuelle Workflowinstanz in einem Datenspeicher gespeichert wurde, aber weiterhin betriebsbereit ist und auf einen Befehl wartet.
Unloaded	Wird aufgerufen, wenn der aktuelle Workflow entladen wird.

Sowohl die einzelne Activity als auch die Run Time Engine verfügen über zahlreiche Ereignisse und können damit über Zustandsänderungen informieren. Bei den Zuständen, welche die Workflow Management Coalition anführt, handelt es sich um Empfehlungen. Diese decken sich nicht zu 100 % mit der konkreten Umsetzung in der Windows Workflow Foundation 4.0. Dies wird aber nicht unbedingt als Einschränkung oder Verletzung des Kriteriums gewertet, da ja die Windows Workflow Foundation 4.0 zum Teil sogar umfangreichere Informationen liefert als von der WfMC gefordert.

3.5 Einsatzgebiete

Im folgenden soll aufgezeigt werden, für welche Einsatzgebiete sich die Verwendung eines Workflow Management Systems, wie der Windows Workflow Foundation 4.0, eignet. Es ist unabdingbar, dass das gewünschte Verhalten der angestrebten Anwendung im Vorfeld ausreichend definiert wird.

Schattauer zählte folgende Anwendungsszenarien für ein Workflow Management System auf. [Sch08 S. 62]

3.5.1 Langlaufende Prozesse

Grundsätzlich kann jeder Geschäftsprozess auch direkt als Programmcode (ohne die Verwendung einer Workflow Run Time) realisiert werden. Wird jedoch auf eine Workflow Lösung verzichtet, so ist es nicht so leicht möglich, langlaufende Prozesse zu persistieren. Beim Persistieren eines Workflows wird sein innerer Zustand, der Inhalt der übergebenen Argumente, der Inhalt der internen Variablen und die aktuelle Position in einen dauerhaften Speicher, meist eine SQL Datenbank, hinterlegt. Zu einem späteren Zeitpunkt kann der Workflow ab dieser Position wieder fortgesetzt werden. Sobald ein Workflow persistiert wurde, werden durch diese Instanz keine Ressourcen mehr belegt. [Cha09 S. 5]

Bemerkenswert ist dabei auch, dass der Workflow nicht unbedingt auf jener Maschine fortgesetzt werden muss, von welcher er persistiert wurde. Im Falle von ASP.NET Lösungen trägt dies zur Skalierbarkeit der Anwendung und zu einer gewissen Lastverteilung bei. [Cha09 S. 23]

Wird ein Geschäftsprozess nun nicht durch die Verwendung einer Workflow Run Time realisiert, sondern der Ablauf direkt in Form von Programmcode abgebildet, so muss der Entwickler selbst dafür Sorge tragen, dass der Zustand langlaufender Prozesse[40] selbst im Falle eines Neustarts des Hosts nicht verloren geht. Das Framework der Windows Workflow Foundation 4.0 entlastet in diesem Fall den Entwickler.

3.5.2 Parallele Prozesse und Multithreading

Jede konkrete Workflow Instanz wird nebenläufig in einem eigenen Thread ausgeführt. An eine Kommunikation zwischen diesen Instanzen ist nicht gedacht. Diese Instanzen können sich gegenseitig auch nicht beeinflussen, da sie isoliert voneinander ausgeführt werden. Sollte in einer Instanz ein Laufzeitfehler auftreten, so bleiben die restlichen Instanzen davon unbeeinflusst. [Sch08 S. 63]

Multithreading kann zwar auch auf herkömmliche Art und Weise in Form von Code realisiert werden, aber auch hier nimmt die Workflow Run Time der WF 4.0 dem Entwickler Arbeit ab.

3.5.3 Workflow Änderungen zur Laufzeit

Trotz einer optimalen Modellierungsphase für die Geschäftsprozesse werden Anpassungen unvermeidbar sein. Geschäftsprozesse unterliegen stets einem gewissen Wandel. Ein weiterer Vorteil der Windows Workflow Foundation ist es, dass Workflow Definitionen selbst zur Laufzeit adaptiert werden können. [Sch08 S. 63]

Im Falle einer Realisierung des Geschäftsprozesses in Form von Code müsste für eine Anpassung ein neues Release (mit einer neuen Versionsnummer) ausgeliefert und installiert werden. Der Aufwand dafür ist meist unvergleichbar höher als die Anpassung einer (deklarativen) Workflow Definition.

3.5.4 Überwachung und Steuerung des Workflows

Der Ablauf innerhalb eines Workflows kann mittels Transaktionen und Kompensationen gesteuert werden. [Sch08 S. 63]

---

[40] Zeiträume im Bereich Tage, Wochen, Monate, Jahren sind keine Seltenheit.

Von Transaktionen wird gefordert, dass sie erfolgreich abgeschlossen werden oder gar nicht. [Gen07 S. 149]

Langlaufende Workflows können oftmals mehrere Tage für die Ausführung benötigen. Für den Fall, dass ein solcher Workflow nicht erfolgreich abgeschlossen werden kann, benötigt man unter Umständen spezielle Lösungsansätze, um bereits durchgeführte Änderungen wieder rückgängig machen zu können. Die Windows Workflow Foundation bietet für solche Fälle Unterstützung in Form von kompensierbaren Activities an. Diese Activities verfügen über Methoden für Compensation[41], Confirmation[42] und Cancellation[43]. [Col10 S. 319 ff.]

3.5.5 Koordination von verteilten Anwendungen

Die Basic Activity Library verfügt über Activities zum Senden[44] und Empfangen[45] von Nachrichten bzw. zum Bestätigen dieser. Dazu bedient sie sich der Windows Communication Foundation. Damit kann ein Workflow mit entfernten WebServices Daten austauschen.

Ein spezieller Host, der WorkflowServiceHost, ist dafür ausgelegt, WCF Nachrichten abzuhorchen, und bei Eintreffen einer entsprechenden Nachricht einen Workflow zu instanziieren. [Col10 S. 300]

AppFabric[46] ermöglicht es auf einfache Art und Weise, einen Workflow in einem Prozess zu hosten, welcher durch den Internet Information Server (kurz IIS) gestartet wird. AppFabric stellt dabei nur eine einfach zu wartende Umgebung zu Verfügung. [Cha09 S. 21]

---

[41] Englisch für: Kompensation
[42] Englisch für: Quittierung
[43] Englisch für: Stornierung
[44] Es handelt sich um die Activities Send und SendAndReceiveReply
[45] Es handelt sich um die Activities Receive und ReceiveAndSendReply
[46] Microsoft bietet mit Windows Server AppFabric eine Infrastruktur für das Hosten und Verwalten von Workflows, welche innerhalb von Anwendungen im Internet Information Server laufen.

3.5.6 Asynchrone Abläufe

Ein Workflow selbst kann sowohl synchron[47] als auch asynchron[48] zu seiner Host Anwendung ausgeführt werden. [Dob09]

Die Basic Activity Library der Windows Workflow Foundation 4.0 verfügt über Activities[49], welche die parallele (also nebenläufig bzw. asynchron) Ausführung mehrerer unabhängiger Activities ermöglicht.

Bei der Parallel Activity ist zu beachten, dass die darauf folgende Activity erst ausgeführt wird, wenn alle Activities innerhalb der Parallel Activity beendet sein. Schlägt eine der parallel auszuführenden Activities fehl, gilt die ganze Parallel Activity als fehlgeschlagen. [Dob09 S. 89]

Das Erzeugen einer solchen Parallelität erfolgt in der Windows Workflow Foundation 4.0 mit wesentlich weniger Aufwand als im herkömmlichen Programmcode.

3.5.7 Komplexe Abläufe und häufige Änderungen

Der Workflow Designer der Windows Workflow Foundation 4.0 ist in der Lage, selbst sehr komplexe Zusammenhänge in übersichtlicher Art und Weise darzustellen. Somit ist es auch für Nicht-Entwickler möglich, diese besser verstehen zu können oder sogar zu adaptieren. Sind die komplexen Geschäftsprozesse hingegen native in Code abgebildet, besteht für den Laien keine Möglichkeit, diese Zusammenhänge zu überblicken. Die grafische Aufbereitung trägt auch zu einer besseren Dokumentation bei. [Sch08 S. 65]

3.5.8 Schlussfolgerung

Vor dem Einsatz der Windows Workflow Foundation muss man sich also über die Komplexität der zu erstellenden Anwendung im Klaren sein. In unserem konkreten Fall ist eine genügend hohe Komplexität vorhanden, die den Einsatz einer Workflow Run Time gerechtfertigt erscheinen lässt. Verteilte Anwendungen hingegen spielen nur eine untergeordnete Rolle.

---

[47] z. B. durch Verwendung von WorkflowInvoker
[48] z. B. durch Verwendung von WorkflowApplikation
[49] Es handelt sich um die Activities Parallel und ParallelForEach<T>

3.6 Zusammenfassung

In diesem Kapitel wurde das Konzept der Windows Workflow Foundation 4.0 erklärt. Auf alle wesentlichen Funktionsbereiche wurde dabei ausreichend eingegangen.

Des weiteren wurde geprüft, ob die Windows Workflow Foundation 4.0 imstande ist, die gestellten Anforderungen an ein Workflow Management System zu erfüllen.

Nachdem die Windows Workflow Foundation 4.0 die gestellten Anforderungen an ein Workflow Management System erfüllt (dies sind: Systemanforderungen; Trennung nach Run Time und Build Time; Abbildung von Geschäftsprozessen in einer Workflow Definition; Inneren Zustände von Instanzen) kann die Windows Workflow Foundation 4.0 als ein Workflow Management System laut Kapitel 2 betrachtet werden. Zudem werden die firmeninternen Anforderungen an ein Workflow Management System einzig und allein durch die Windows Workflow Foundation 4.0 abgedeckt.

Zuletzt wurden sinnvolle Anwendungsfälle für die Windows Workflow Foundation 4.0 aufgezählt. Damit wurden die Vorteile, welche durch die Verwendung der WF 4.0 entstehen, hervorgehoben.

Die Windows Workflow Foundation 4.0 erfüllt somit alle gestellten Anforderungen und kann für die Abbildung von Geschäftsprozessen in einer .NET Anwendung verwendet werden!

4 Abbilden v. Geschäftsprozessen in einer NET Anwendung mit der WF 4.0

„Alles Große vermögen wir nur aus einem gehörigen Abstand zu ihm zu erkennen. Wer an einem Berg mit einer Lupe steht, bemerkt nur Sandkörner und Insekten.“

Paul Ambroise Valéry (1871-1945)
französischer Schriftsteller

In diesem Kapitel steht die praktische Umsetzung im Vordergrund. Als Entwicklungsumgebung wurde Microsoft Visual Studio 2010 Premium verwendet.

4.1 Relevante Namespaces

In Abbildung 11 sind in einer übersichtlichen Darstellung die wesentlichen Namespaces aus dem Umfeld der Windows Workflow Foundation 4.0 dargestellt.

Abbildung 11 Namespaces aus dem Workflow Milieu (Quelle: Ausschnitt aus einer Grafik von [Mic1003])

Der Assistent von Visual Studio 2010 zum Anlegen eines neuen Projektes bietet unter der Rubrik Workflow die Möglichkeit, eine Workflow Console Application anzulegen.

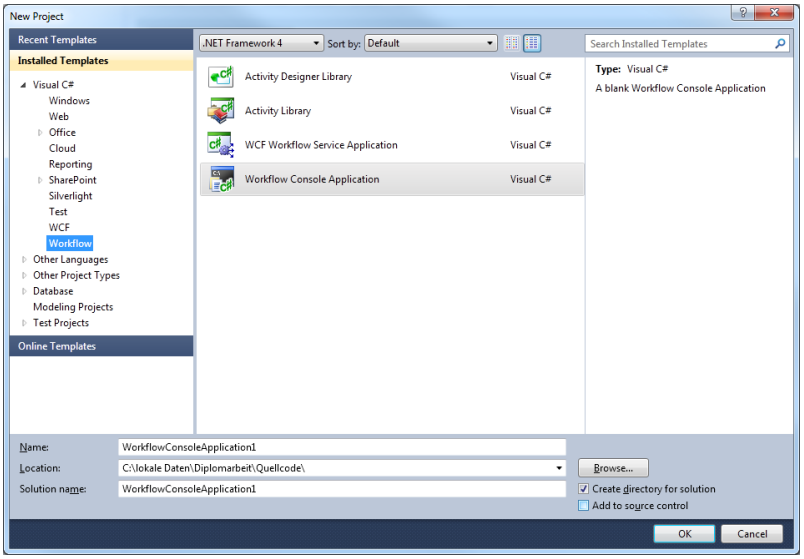


Abbildung 12 Dialog zum Anlegen eines neuen Projektes in Visual Studio 2010 RC

Wählt man diese Option, so wird ein neues Projekt erstellt. Dieses neue Projekt beinhaltet automatisch eine XAML-Datei für die Definition eines Workflows und verweist auf folgende Namespaces aus dem Workflow Umfeld:

```
using System.Activities;
using System.Activities.Statements;
```

Quellcode 1 Standard Namespaces einer Workflow Console Application

Dies ist das Grundgerüst einer Anwendung, welches Geschäftsprozesse unterstützen kann.

Die Option Activity Library entspricht im Wesentlichen einer normalen ClassLibrary mit zusätzlichen Referenzen zu notwendigen Workflowassemblies. Activity Designer Library ist zu verwenden, um eine Activity mit Designer Funktionalität zu erweitern. Der Projekttype WCF Workflow Service Applikation erstellt einen Service, der eine WCF Nachricht empfangen kann und dadurch einen Workflow startet. [Dob09 S. 85]

4.2 Basic Activity Library

Workflows bestehen aus Activities. Microsoft liefert einen Grundstock an Activities in Form der Basic Activity Library (BAL) mit aus. Diese kann und sollte aber um eigene Activities erweitert werden. Abbildung 13 zeigt die Klassenhierarchie der Workflow Activities.

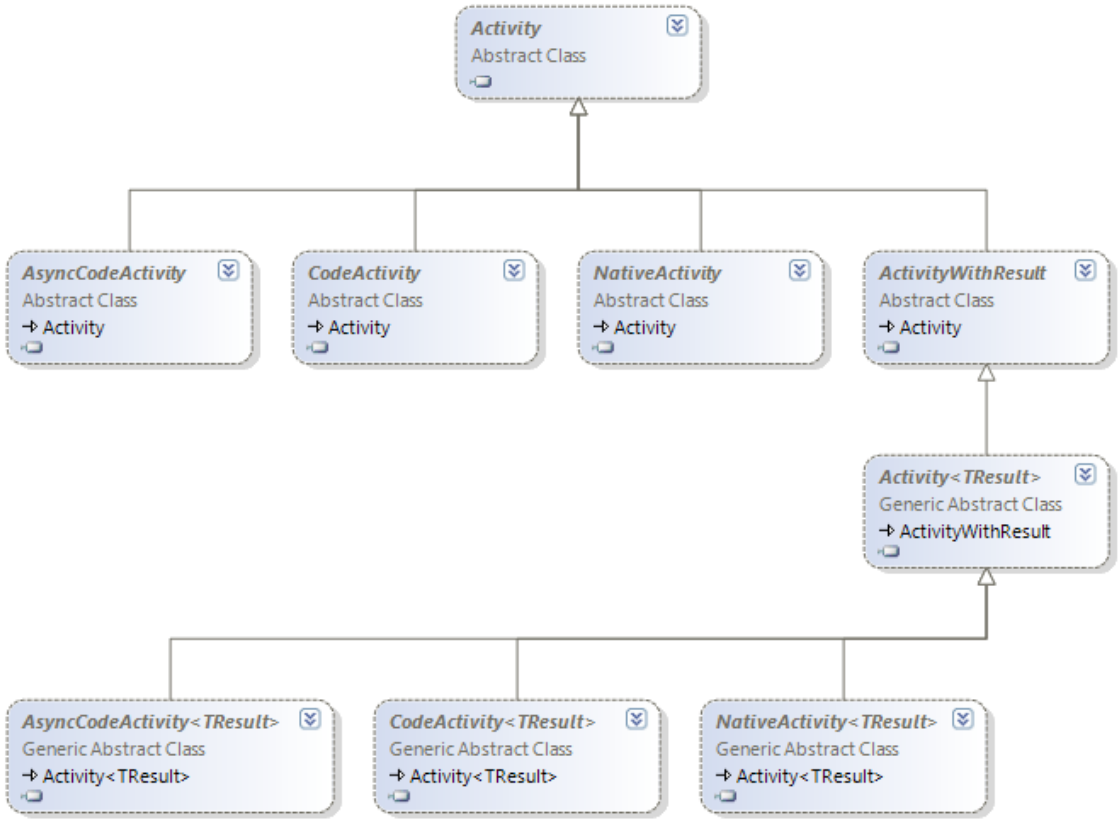


Abbildung 13 Klassenhierarchie von Workflow Activities

Wie daraus zu erkennen ist, stammen letztlich alle Activities von der abstrakten Klasse Activity ab. Activities können z. B. durch eine Kombination bereits bestehender Activities gebildet werden. Das Erstellen von Activities kann mittels Programmcode oder vollständig in XAML erfolgen.

Die folgenden Abbildungen führen die Elemente der Basic Activity Library an. Sämtliche Abbildungen wurden Microsoft Visual Studio entnommen.

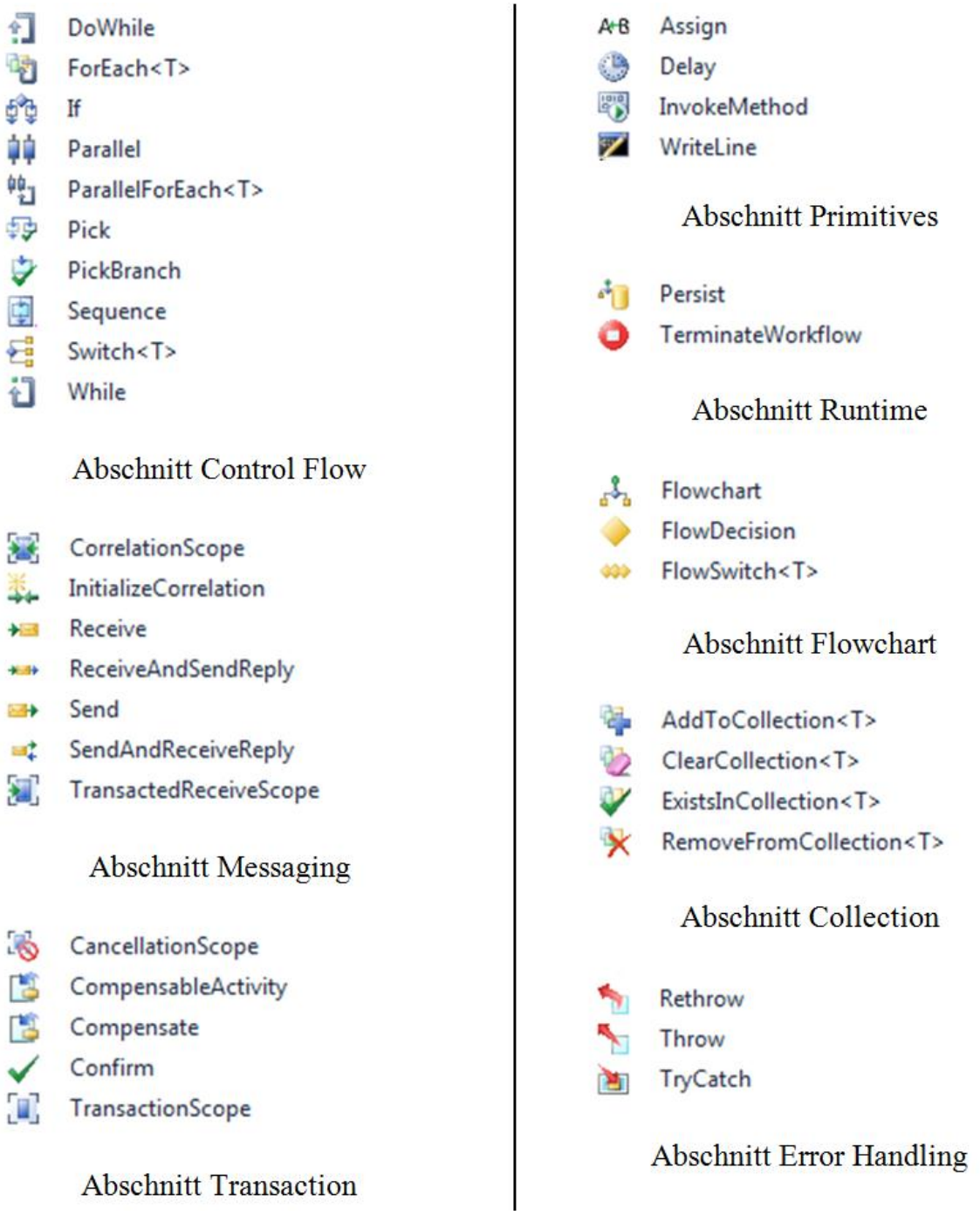


Abbildung 14 Elemente der Basic Activity Library

Im Rahmen dieser Diplomarbeit können nicht alle Activities im Detail erklären, da dies den Umfang der Arbeit nur unnötig erhöhen würde. Stellvertretend werden jedoch einige essenzielle Activities und die Besonderheiten bei deren Verwendung im Detail erörtern.

Dazu gehören:

- Assign
- WriteLine
- If
- Sequence
- FlowChart
- InvokeMethod

Passende Beispiele für die Umsetzung sind jeweils im Anhang A zu finden.

4.2.1 Assign Activity

Die Aufgabe der Assign Activity ist das Zuweisen eines Wertes an eine Variable oder ein Argument. Die Aufgabe der Activity kann mit der Aufgabe des `=` Operators verglichen werden.

Bei der Assign Klasse handelt es sich um eine generische Klasse. Somit kann sie grundsätzlich jeden Datentyp unterstützen. Im Bsp. Quellcode 2 wird eine Integer Variable verwendet. Deshalb findet sich ein Objekt vom Type `Assign<int>` wieder. Die `To` und `Value` Eigenschaften verwenden ebenfalls Template Klassen und müssen vom selben Typ (in diesem Fall `<int>`) definiert werden. Bei der `To` Eigenschaft handelt es sich um eine `OutArgument` Klasse, welche die verwendete Variablen Klasse im Konstruktor benötigt. Die `Value` Eigenschaft verwendet eine generische `InArgument` Klasse. Für diesen Konstruktor wird eine Lambda Expression verwendet.

```
// inkrementieren der Variable counter um 1
new Assign<int>
{
	// Zuweisen des "friendly name"
	DisplayName = "Wertezuweisung",

	// To spezifiziert das Argument,
	// an das ein Wert zugewiesen werden soll.
	// Mittels OutArgument wird auf die Variable 'counter' zugegriffen
	To = new OutArgument<int>(counter),

	// Value repräsentiert den Wert, welcher zugewiesen werden soll.
	// Get liefert den Wert der Variable im angegeben Kontexte
	// 'env' repräsentiert den Kontext der Activity.
	Value = new InArgument<int>(env => counter.Get(env) + 1)
}
```

Quellcode 2 Assign Activity in Code

An dieser Stelle sei auf Punkt Lambda Expression in Anhang B hingewiesen.

Die Assign Activity kann natürlich auch im Workflow Designer verwendet werden.

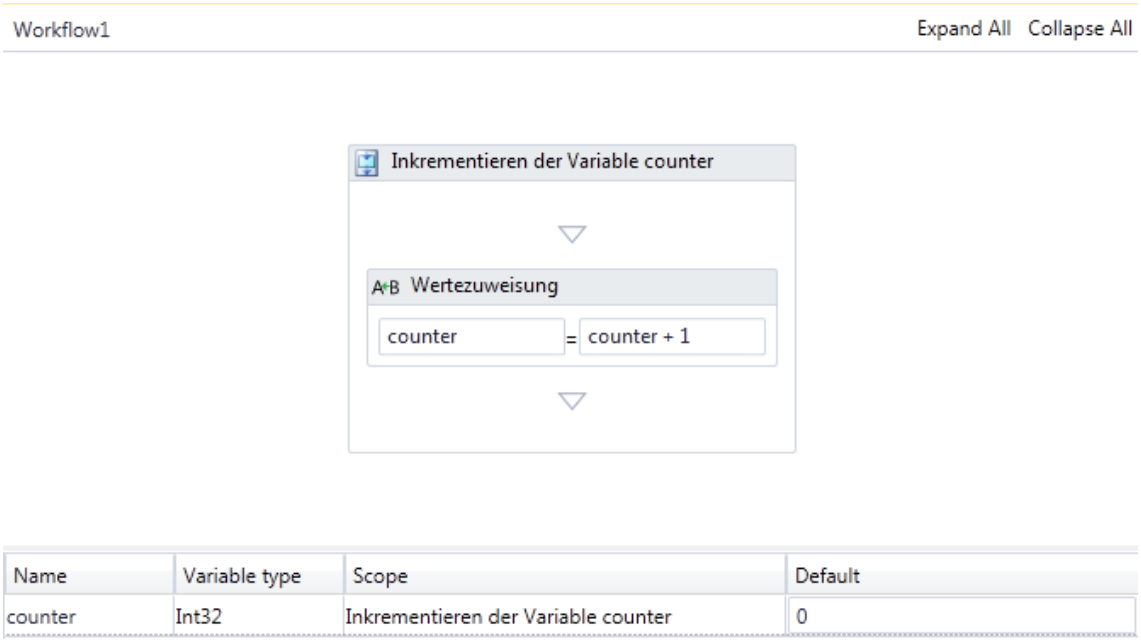


Abbildung 15 Assign Activity im Workflow Designer

In Abbildung 15 ist zu erkennen, wie der Wert einer Variablen inkrementiert wird. Dabei ist zu beachten, dass im Workflow Designer (genauer gesagt im Expression Editor) ausschließlich VB.NET Syntax zu verwenden ist. [Mac10 S. 143]

Laut einer Vermutung von Damir [Dob09 S. 88] könnte dies aber zu einem späteren Zeitpunkt durch Microsoft geändert werden.

Microsoft argumentiert die Verwendung der VB.NET Syntax damit, dass diese für die Zielgruppe des Workflow Designers einfacher zu verstehen sei als C# Syntax.

Im Anhang A befinden sich zwei Beispiele, welche die Verwendung der Assign Activity demonstrieren:

- Assign Activity in Code
- Assign Activity in XAML

4.2.2 WriteLine Activity

Die WriteLine Activity dient dazu, einen angegebenen Text auf einem Objekt vom Type TextWriter zu schreiben. Üblicherweise wird der Text an eine Console ausgegeben. Vor allem im Stadium des Entwurfs eines Workflows hat sich diese Activity als sehr praktisch herausgestellt.

Verwundern mag auf den ersten Blick vielleicht der folgende Ausdruck:

```
Text = new InArgument<string>
(
      env => "Wert= " + counter.Get(env).ToString()
)
```

Quellcode 3 Zuweisen des Textes einer WriteLine Activity mittels Lambda Ausdruck

Der Ausdruck demonstriert das Zuweisen des Textes der WriteLine Activity. Neben einem String soll dabei auch der Wert einer Variablen (`counter`) ausgegeben werden. Zu beachten ist, dass auch hier der Zugriff auf die Variable wieder über den Kontext der Activity (repräsentiert durch `env`) erfolgen muss. Die Verwendung eines Lambda Ausdruckes empfiehlt sich auch hier.

Die beiden Beispiele im Anhang A

- Assign Activity in Code
- Assign Activity in XAML

demonstrieren ebenfalls die Verwendung der WriteLine Activity.

4.2.3 If Activity

Die If Activity entspricht dem klassischen If Konstrukt. In Abhängigkeit von einer Bedingung wird einer von zwei möglichen Zweigen ausgeführt. Ähnlich erfolgt auch die konkrete Umsetzung. Eine If Activity besitzt drei Properties[50]: `Condition`, `Then` und `Else`. `Condition` ist vom Typ `InArgument<bool>`. Bei `Then` und `Else` handelt es sich wieder um Activities.

[50] Englisch für: Eigenschaften

Die folgende Code Zeile zeigt die Verwendung einer If Condition:

```
Condition = ExpressionServices.Convert<bool>(env=>alter.Get(env)>=18)
```

Quellcode 4 Zuweisen einer If-Condition mittels ExpressionServices

Bei `env => alter.Get(env)` handelt es sich lediglich um einen Lambda Ausdruck, der den Wert einer Variablen (alter) liefert.

Anschließend wird verglichen, ob der Inhalt der Variablen `alter` größer oder gleich 18 (Jahren) ist. Das Ergebnis dieses Vergleiches ist ein boolescher[51] Wert. Es handelt sich also um eine Lambda Projektion[52], da ein Datentyp Integer übergeben wurde und ein Datentyp Bool retour kommt.

Die statische Methode `Convert<T>` der Klasse ExpressionServices wird verwendet um ein Objekt vom Type `InArgument<T>` zu erzeugen, welche das `Condition` Property erwartet.

Die Properties `Then` und `Else` sind vom Type Activity. Ihnen kann unmittelbar eine neue Activity zugewiesen werden. Das folgende Beispiel veranschaulicht dies:

```
// wird ausgeführt, wenn Condition = True ist
Then = new WriteLine()
{
      DisplayName = "Volljährig",
      Text = "Zugang erteilt!"
}
```

Quellcode 5 Zuweisen einer Activity an den Then Zweig

Dem Then Zweig wird eine neue anonyme WriteLine Activity zugewiesen. Der Else Zweig ist nicht extra angeführt, da er sich analog verhält.

Wie aus Abbildung 11 (Seite 45) zu erkennen ist, wird für die If Activity ein Verweis auf den Namespace `System.Activities.Statements` benötigt.

In der folgenden Abbildung 16 ist die Umsetzung einer If Activity im Workflow Designer dargestellt. Deutlich zu unterscheiden sind dabei die Condition sowie der Then und der Else Zweig.

---

[51] True oder False
[52] Eine Erklärung der Lambda Projektion findet sich in Anhang B.

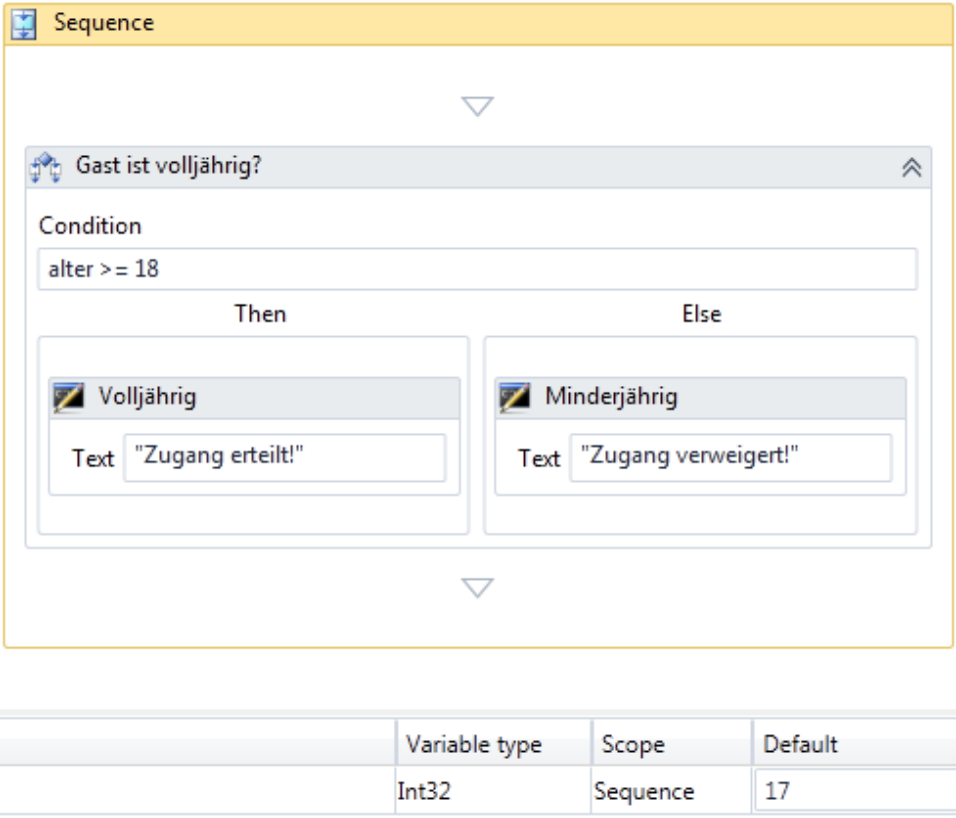


Abbildung 16 If Activity im Workflow Designer

Am unteren Ende von Abbildung 16 ist die Definition der Variablen `alter` zu erkennen. Zu beachten ist bei Variablen, dass diese einen Scope[53] besitzen. In diesem Fall ist die Variable im Bereich der Sequence gültig. Die Variable ist vom Type Int32 und wurde mit dem Wert 17 initialisiert.

Der Workflow Designer speichert die Workflow Definition in Form einer XAML-Datei. Im Folgendem ist ein Auszug aus dieser XAML-Datei dargestellt.

```
<If Condition="[alter >= 18]" DisplayName="Gast ist volljährig?" >
      <If.Then>
            <WriteLine DisplayName="Volljährig" Text="Zugang erteilt!" />
      </If.Then>
      <If.Else>
            <WriteLine DisplayName="Minderjährig" Text="Kein Zugang!" />
      </If.Else>
</If>
```

Quellcode 6 XAML-Code einer If Activity

[53] Englisch für: Gültigkeitsbereich

Selbst in der XAML-Datei sind die Bereiche `If`, `Condition`, `Then`, `Else` und `WriteLine` deutlich auszumachen. Dieser Ausschnitt verdeutlicht auch nochmals das deklarative Prinzip einer Workflow Definition wie in 2.2 (Seite 9) gefordert.

Die beiden Beispiele im Anhang A:

- If Activity in Code
- If Activity in XAML

demonstrieren ebenfalls die Verwendung der If Activity.

4.2.4 Sequence Activity

Eine der prominentesten Activities aus der Gruppe der Control Flow ist die Sequence. Sie ist kein Statement, sondern ein Container für weitere Aktivitäten. Dies ist z. B. illustriert in Abbildung 16 (Seite 53).

In Abbildung 17 ist ein Design Pattern[54] einer Sequence Activity dargestellt.

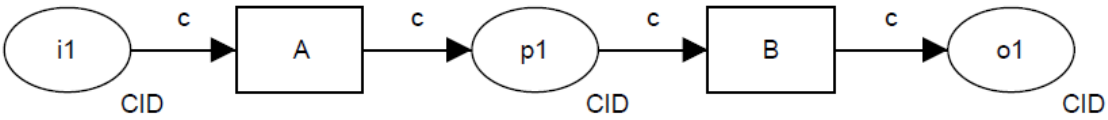


Abbildung 17 Design Pattern einer Sequence (Quelle: [Rus06 S. 9])

Eine Activity (B) in einem Workflow Prozess wird aktiviert, nachdem die vorhergehende Activity (A) beendet ist. [Rus06 S. 8]

Neben Activities kann eine Sequence auch Variablen beinhalten. Liegt der Scope einer Variablen auf einer Sequence, so kann diese Variable auch in allen darunter liegenden Activities verwendet werden. [Dob09 S. 88]

[54] Englisch für: Entwurfsmuster. Design Patterns sind Vorlagen für wiederkehrende Entwurfsprobleme in der Softwarearchitektur.

An dieser Stelle sei auf Anhang B hingewiesen. Dort ist zu sehen, wie mittels der Verwendung anonymer Typen auf einfache Art und Weise eine Sequence erstellt werden kann.

Für die Steuerung des Workflows können in einer Sequence Activity z. B. die If oder Switch Activity verwendet werden.

Die Beispiele im Anhang A:

- Assign Activity in Code
- Assign Activity in XAML
- If Activity in Code
- If Activity in XAML

demonstrieren ebenfalls die Verwendung der Sequence Activity.

4.2.5 FlowChart Activity

Bei der FlowChart[55] Activity handelt es sich ebenfalls um einen Container für weitere Activities. [Mil09]

Activities in einer Flowchart Activity sind jedoch mittels „*Decision Trees*“ [Col10 S. 33] untereinander verknüpft und werden nicht wie bei der Sequence Activity in einer sequenziellen Reihenfolge ausgeführt, sondern können in jeder beliebigen Ablaufreihenfolge ausgeführt werden.

Verknüpft werden die Activities innerhalb einer Flowchart Activity mittels Pfeilen. Diese Pfeile bestimmen den Ablauf des Workflows. Die Pfeile können auch auf davor liegende Activities zeigen. Bei der Sequence Activity werden diese Verbindungspfeile automatisch generiert und können nicht beeinflusst werden. [Col10 S. 34]

Damir weist darauf hin, dass es aber auch in einer Flowchart Activity nur einen „*Single Path of Execution*“[56] [Dob09 S. 90] geben kann. Design Patterns wie Split und Join sind damit nicht möglich.

---

[55] Englisch für: Flussdiagramm, Ablaufdiagramm

Eine Flowchart Activity verfügt über einen dezidierten Startpunkt. Dieser ist in Abbildung 18 dargestellt. [Col10 S. 34]

Abbildung 18 Startpunkt einer Flowchart Activity

Für die Steuerung des Workflows stehen Activities wie FlowDecision und FlowSwitch<T> zur Verfügung. [Col10 S. 35 ff.]

Das Beispiel im Anhang A:

- Flowchart Activity in XAML

demonstriert die Verwendung von FlowChart, FlowDecision, FlowSwitch und Parallel.

4.2.6 InvokeMethod Activity

Diese Activity kann verwendet werden, um eine Methode einer Klasse aufzurufen. In welchem Namespace sich die aufgerufene Klasse befindet ist dabei irrelevant. Weiters ist es nicht notwendig, dass diese Klasse von einer Workflow spezifischen Basisklasse abgeleitet wird. [Col10 S. 86 ff.]

Die Activity InvokeMethod wird im wesentlichen durch folgende drei Eigenschaften definiert:

- TargetType
- TargetObject
- MethodName

TargetType definiert das Objekt, welches die aufzurufende Methode enthält. Bei der Verwendung einer statischen Klasse kann diese direkt mittels TargetType verwendet werden. Sollte eine Methode einer nicht statischen Klasse verwendet werden, so kann

---

[56] Englisch für: alleiniger Ausführungspfad

z. B. eine Variable vom Typ dieser Klasse definiert werden und in späterer folge an TargetObject zugewiesen werden. Entweder TargetObject oder TargetTyp muss angegeben werden.

MethodName kennzeichnet den Namen jener Methode, welche bei der Ausführung der Activity aufgerufen wird. Die aufgerufene Methode muss als public deklariert sein.

Argumente werden der aufgerufenen Methode mittels der Auflistung Parameters übergeben. Wesentlich dabei ist, dass Reihenfolge und Type der Methodensignatur entsprechen müssen.

Result beinhaltet den Rückgabewert der aufgerufenen Methode.

Das Beispiel im Anhang A:

- InvokeMethod

demonstriert die Verwendung von dieser Activity.

4.3 Erweitern der BAL mittels Custom Activities

Unter Punkt 4.2 (Seite 47) wurden essenzielle Vertreter der Basic Activity Library vorgestellt. Nur mit Mitgliedern der BAL kann aber nur schwerlich ein funktionaler Workflow definiert werden. Die große Flexibilität der Windows Workflow Foundation besteht darin, dass eigene Activities (sogenannte Custom Activities) erstellt und verwendet werden können. Für die Beantwortung der Aufgabenstellung dieser Diplomarbeit ist dies ein wesentlicher Punkt.

Damir bezeichnet die Activity als das semantisch kleinste „*Artefakt*“ [Dob09 S. 86] in einem Workflow.

Die Windows Workflow Foundation erlaubt es nicht, dass ausführbarer Code in einer Workflow Definition vorhanden ist. Jeglicher Code muss innerhalb einer Activity ausgeführt werden. Bis zur Version 3.5 der Windows Workflow Foundation konnte eine CodeActivity innerhalb einer Workflow Definition verwendet werden. Mit der Version 4.0 der Windows Workflow Foundation wurde diese jedoch zu einer abstrakten Klasse

geändert. Dies geht auch aus Abbildung 13 (Seite 47) hervor. Somit kann die CodeActivity nicht mehr direkt in einer Workflow Definition verwendet werden. Sie stellt jedoch die Basis für zahlreiche Mitglieder der Basic Activity Library dar. [Col10 S. 80]

Tabelle 14 Basisklasse für Custom Activities

Basisklasse	**Beschreibung**
Activity	Wird als Basisklasse verwendet, wenn Activities aus anderen Activities erzeugt werden. Bei der Verwendung von XAML wird üblicherweise davon abgeleitet.
CodeActivity	Eine einfache Basisklasse, welche die schnelle Realisierung einer Custom Activity ermöglicht.
AsyncCodeActivity	Im Unterschied zur CodeActivity werden abgeleitete Activities asynchron ausgeführt.
NativeActivity	Die NativeActivity bietet erweiterten Zugriff auf die Run Time. So können abgeleitete Activities z. B. Bookmarks definieren.

4.3.1 Übergabe von Argumenten an Activities

Bevor konkret auf das Erstellen einer Custom Activity eingegangen wird, muss noch geklärt werden, auf welche Art und Weise Argumente an eine Activity übergeben werden können bzw. von dieser empfangen werden können.

Activities können mit Argumenten versehen werden. Damit wird gesteuert, welche Daten in eine Activity bzw. aus einer Activity wandern. Argumente von Activities können durchaus mit Argumenten von Methoden der imperativen Programmierung verglichen werden. Argumente haben stets eine Richtung[57] und einen Typ. Bei der Definition eines Workflows kann die oberste Activity (meist eine Sequence oder Flowchart) mit Argumenten versehen werden. Auf diese Argumente kann bei der Verwendung des Workflows von außen zugegriffen werden. [Mil09]

Argumente unterscheiden sich dahingehend von Variablen, dass Argumente keine Scope Einschränkung besitzen. Argumente gelten stets für den ganzen Workflow. [Col10 S. 50]

[57] Zur Verfügung stehen: In, Out und InOut

Abbildung 19 zeigt die Klassenhierarchie von Argument Klassen. Deutlich zu erkennen ist, dass letztlich alle drei Klassen von der Basisklasse Argument abgeleitet sind.

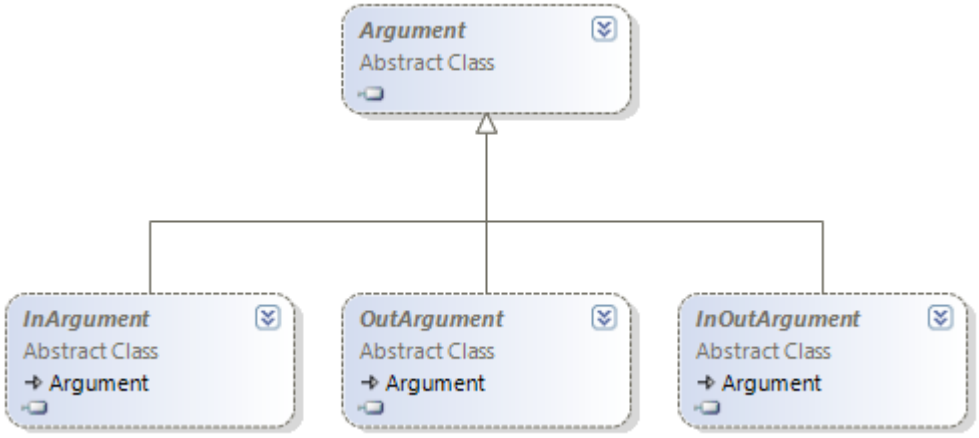


Abbildung 19 Klassenhierarchie von Argumenten

Innerhalb der Klasse der Custom Activity sind nun öffentliche Eigenschaften zu deklarieren.

```
public InArgument<string> Caption { get; set; }
public OutArgument<DialogResult> Result { get; set; }
```

Quellcode 7 Deklaration zwei Argumente

Die kürzeste Schreibweise ist in Quellcode 7 zu sehen. Es wird dabei eine öffentliche generische Eigenschaft vom Type `InArgument<string>` mit dem Namen Caption deklariert. Diese Eigenschaft verfügt über eine Methode zum Zuweisen ihres Wertes und zum Lesen ihres Wertes. Result hingegen ist vom Type `OutArgument<DialogResult>`. Result wird somit an den Verwender der Activity zurück geliefert.

Argumente von Custom Activities können zusätzlich mit Attributen versehen werden. Attribute werden dabei in eckigen Klammern über das Argument gestellt.

Mit einem Attribut lässt sich das Laufzeitverhalten praktisch aller .NET Elemente beeinflussen. Attribute werden erst zur Laufzeit ausgewertet. [Küh091 S. 367]

Mögliche Attribute für Argumente sind z. B. RequiredArgument und DefaultValue.

RequiredArgument kennzeichnet ein Attribut als verpflichtendes Attribut. Im Designer wird dies im Fehlerfall durch einen roten Punkt auf der Activity gekennzeichnet.

```
[RequiredArgument]
public InArgument<string> Caption { get; set; }
```

Quellcode 8 Deklaration eines verpflichtenden Argumentes

Das Attribut DefaultValue hingegen ist praktisch für optionale Attribute. Es kann damit ein Standardwert definiert werden.

```
[DefaultValue(MessageBoxButtons.OK)]
public InArgument<MessageBoxButtons> Buttons { get; set; }
```

Quellcode 9 Deklaration eines optionalen Argumentes inkl. Standardwert

Innerhalb der Activity erfolgt der Zugriff auf die Werte von Argumenten wieder über den Kontext der Activity. Der Kontext der Activity wird in der Execute Methode als Parameter mit übergeben.

```
protected override void Execute(CodeActivityContext context)
```

Quellcode 10 Überschreiben der Execute Methode

Die Execute Methode muss von der Custom Activity überschrieben werden. In ihr ist die eigentliche Funktionalität zu hinterlegen. [Col10 S. 82]

Nun steht in der Execute Methode ein Objekt der Klasse CodeActivityContext zur Verfügung. Dies muss beim Zugriff auf die Argumente verwendet werden wie die folgenden beiden Codefragmente veranschaulichen.

```
String caption = context.GetValue<string>(Caption);
```

Quellcode 11 lesender Zugriff auf den Wert eines Arguments

```
context.SetValue<DialogResult>(Result, dialogResult);
```

Quellcode 12 schreibender Zugriff auf den Wertes eines Arguments

4.3.2 Custom Activities um ein Design erweitern

Einer der größten Argumente für die Verwendung der Windows Workflow Foundation 4.0 ist die Möglichkeit, die Ablauflogik eines Workflows deklarativ erfassen zu können. Endanwendern kann das manuelle Erstellen und Warten von XAML-Dateien nicht zugemutet werden. Um die Akzeptanz bei diesen Personen zu steigern, kommt dem grafischen Design von Workflows eine große Bedeutung zu. Custom Activities können mit einer eigenen, auf der Windows Presentation Foundation basierenden Oberfläche

versehen werden. Damit stehen automatisch Techniken der WPF[58] wie die Verwendung von Styles, Trigger und Data Binding zur Verfügung. Die Windows Presentation Foundation 4.0 bietet einige User Controls[59] an, welche das Design von Custom Activities unterstützen und erleichtern. [Mil09]

Tabelle 15 Häufig verwendete User Controls für das Design von Custom Activities.

User Controls	Verwendungszweck
ActivityDesigner	Repräsentiert das Wurzel Element eines jeden Designs.
WorkflowItemPresenter	Wird verwendet, um eine einzelne Custom Activity anzuzeigen.
WorklfowItemsPresenter	Stellt einen Container dar, in welchem mehrere Custom Activities dargestellt werden können.
ExpressionTextBox	Wird verwendet für die Eingabe von Ausdrücken (z. B. bei Argumenten).

Visual Studio 2010 beinhaltet eine Vorlage für ein Projekt vom Type Activity Designer Library und eine Vorlage für eine Datei vom Typ Activity Designer. Diese können und sollten verwendet werden, um die entsprechenden „*Artefakte*“ [Mil09] zu erstellen. Nachdem der Designer damit initialisiert wurde, kann mit der grafischen WPF Oberfläche für die Custom Activity begonnen werden. Für das Design der Oberfläche wird ein Zugriff auf die Eigenschaft der Custom Activity benötigt. Der Zugriff darauf erfolgt mittels einem Objekt vom Type ModelItem.

ModelItem ist eine Abstraktion der zugrunde liegenden Custom Activity und beinhaltet sämtliche Eigenschaften dieser Activity. [Mic1004]

Bleibt noch die Frage zu klären, auf welche Art und Weise nun das konkrete Custom Activity mit ModelItem verknüpft wird. Das Beispiel in Quellcode 13 zeigt die Deklaration einer öffentlichen Klasse für eine Custom Activity mit dem Namen ShowMessageBox. Tabelle 14 (Seite 58) führt die Basisklassen für eine CodeActivity an.

[58] Auf die angeführten Techniken kann im Detail im Rahmen dieser Diplomarbeit leider nicht eingegangen werden.
[59] Englisch für: Benutzersteuerelement

```
[Designer(typeof(MessageBoxActivityDesigner))]
public sealed class ShowMessageBox : CodeActivity
```

Quellcode 13 Attribut für Designer einer Activity

Die Deklaration ist mit einem Attribut vom Namen Designer versehen. Mittels `typeof` wird der Klasse Designer jene Klasse übergeben, welche für das Design zuständig ist. `MessageBoxActivityDesigner` ist in diesem Beispiel der Name der Designerklasse.

Wie in Tabelle 15 (Seite 61) angeführt, bildet ActivityDesigner stets das Wurzelelement eines Designs. Im Attribut x:Class findet sich der Name der Klasse (in diesem Beispiel `MessageBoxActivityDesigner`) wieder. In Beispiel Quellcode 13 ist zu sehen ist, wie dieser Name für das Designer Attribut der Custom Activity Klasse verwendet wird.

```
<sad:ActivityDesigner
x:Class="MessageBoxActivity.MessageBoxActivityDesigner"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:s="clr-namespace:System;assembly=mscorlib"

xmlns:sad="clr-namespace:System.Activities.Presentation;
assembly=System.Activities.Presentation"

xmlns:sadv="clr-namespace:System.Activities.Presentation.View;
assembly=System.Activities.Presentation"

xmlns:sadc="clr-namespace:System.Activities.Presentation.Converters;
assembly=System.Activities.Presentation">
```

Quellcode 14 Deklaration eines Activity Designer in XAML

Die restlichen mit xmlns beginnenden Zeilen dienen zu Deklaration der benötigten Namespaces.

Eine ExpressionTextBox ermöglicht das Editieren von Ausdrücken, welche für Argumente benötigt werden. [Mil09]

Der Unterschied zu einer herkömmlichen Textbox besteht darin, dass dieses Control nicht nur einfachen Text editieren lässt, sondern auch Intellisense[60] und eine Syntaxprüfung auf Basis VB.NET anbietet. Quellcode 15 zeigt den wesentlichen Teil der Deklaration einer ExpressionTextBox in XAML.

[60] Bei IntelliSense handelt es sich um ein Hilfsmittel zur automatischen Vervollständigung bei der Bearbeitung von Quellcode.

```
<sadv:ExpressionTextBox …
Name="ctrlCaption"
AutomationProperties.AutomationId="Caption"

Expression="{Binding Path=ModelItem.Caption, Mode=TwoWay,

Converter={StaticResource ArgumentToExpressionConverter},
ConverterParameter=In}"

ExpressionType="{x:Type TypeName=s:String}"

OwnerActivity="{Binding Path=ModelItem, Mode=OneWay}"/>
```

Quellcode 15 Deklaration einer ExpressionTextBox

Mit dem Attribut OwnerActivity wird die Verbindung zu ModelItem hergestellt. Von Wichtigkeit sind noch die Attribute Expression, Converter und ExpressionType. Es wird damit das Binding zwischen der ExpressionTextBox und einem ArgumentToExpressionConverter hergestellt.

Ein ArgumentToExpressionConverter ist das Bindeglied zwischen Argumenten (öffentliche Eigenschaft einer Custom Activity Klasse) und Expressions (Eingabe im Designer). Ein ArgumentToExpressionConverter ist zuständig für das Konvertieren zwischen diesen beiden Ebenen. Für ArgumentToExpressionConverter wird der Namespace System.Activities.Presentation.Converters benötigt. Mit dem folgenden XAML-Code kann der ArgumentToExpressionConverter dem Resource Dictionary hinzugefügt werden. [Mic1005]

```
<sad:ActivityDesigner.Resources>
      <ResourceDictionary>
            <sadc:ArgumentToExpressionConverter
            x:Key="ArgumentToExpressionConverter" />
      </ResourceDictionary>
</sad:ActivityDesigner.Resources>
```

Quellcode 16 Deklaration eines ArgumentToExpressionConverter in XAML

Abbildung 20 zeigt die fertige Oberfläche einer Custom Activity. Der rote Punkt in der Grafik signalisiert, dass bei diesem Argument ein Fehler vorhanden sein muss. Das Argument Caption ist ein Objekt vom Typ String. Der ArgumentToExpression Converter erkennt in der Syntax der Eingabe einen Fehler und veranlasst die ExpressionTextBox diesen Fehler darzustellen. Der Fehler im konkreten Beispiel ist das fehlende Anführungszeichen am Ende.

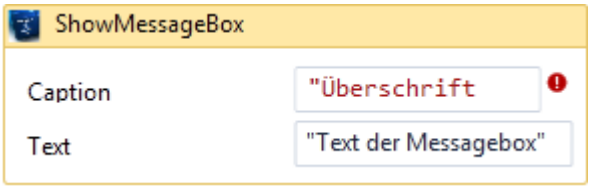


Abbildung 20 fertig GUI einer Custom Activity mit zwei Eingabefeldern

Objekte der Klasse ActivityDesigner verfügen über eine Eigenschaft Icon. Damit kann das Icon der Custom Activity beeinflusst werden. [MSM10] Ein individuelles Icon ist in Abbildung 20 ebenfalls zu erkennen.

4.3.3 Erstellen von Unit Tests für eine Custom Activity

Eine Activity stellt die kleinste „*semantische Einheit*“ [Dob09 S. 86] in einem Workflow dar.

Einheiten können vollständig isoliert vom restlichen System in Form von Unit Tests auf deren korrekte Funktionalität hin überprüft werden. [McC04 S. 511 ff.] Um sicherzustellen, dass das Programm auch nach Korrekturarbeiten und Wartungsarbeiten noch die gewünschten Ergebnisse liefert, sind Unit Tests zu empfehlen.

Visual Studio unterstützt den Entwickler auch bei solchen Unit Tests. Im folgenden Bsp. Quellcode 17 ist ein Unit Test für eine Custom Activity dargestellt.

```
[TestMethod]
public void ShowMessageBoxMustReturnOKVariante2()
{
	// erstellen eines Dictionarys mit den Übergabeparametern
	IDictionary<string, object> arguments =
	new Dictionary<string, object>
	{
		{ "Caption", "Hallo" },
		{ "Text", "Windows Workflow Foundation 4.0" },
		{ "Buttons", MessageBoxButtons.OK },
		{ "Icon", MessageBoxIcon.Information }
	};

	// synchrones Ausführen der Activity und empfangen der Antwort
	var output = WorkflowInvoker.Invoke(new
	MessageBoxActivity.ShowMessageBox(), arguments);

	// der zurückgelieferter Wert muss OK sein - alles andere ist falsch
	Assert.AreEqual(System.Windows.Forms.DialogResult.OK,
	output["Result"]);
}
```

Quellcode 17 Unit Test für eine Custom Activity

Zuerst wird ein Objekt vom Type Dictionary erstellt und mit den gewünschten Werten befüllt, welche an die Activity übergeben werden sollen. Anschließend wird die Custom Activity mittels der statischen Invoke Methode der Klasse WorkflowInvoker zur Ausführung gebracht. Das zuvor erzeugte Objekt mit den Argumenten wird dabei als Aufrufparameter mit übergeben. Das Ergebnis des Workflows wird in einer Variable namens Output gespeichert. Der Type dieser Variable wird vom Compiler selbstständig mittels Typinferenz (siehe Anhang B) ermittelt.

Der eigentliche Unit Test erfolgt nun mittels der statischen Klasse Assert[61]. Die Methode AreEqual prüft, ob das tatsächliche Ergebnis (Istwert) mit dem gewünschten Ergebnis (Sollwert) übereinstimmt. Das gewünschte Ergebnis muss dabei aufgrund der Aufrufparameter zu ermitteln sein. In Abbildung 21 ist das Ergebnis von zwei Unit Tests dargestellt. In diesem Fall liegt ein positives und ein negatives Testergebnis vor.

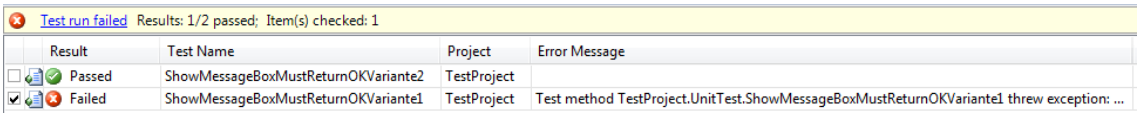


Abbildung 21 Ergebnis von zwei Unit Test

Unit Tests sollten bereits in einem sehr frühen Stadium begonnen werden und laufend gewartet werden. Dies kann wesentlich zur Qualität der Software beitragen. Bei der sogenannten Test-First-Development[62] Technik werden zuerst die Testfälle geschrieben, bevor irgendwelcher Quelltext programmiert wird. [McC04 S. 241]

Als konkretes praktisches Beispiel zum Themenkomplex Custom Activity wurde eine Custom Activity erstellt, welche eine MessageBox darstellen kann. An Argumenten soll der Custom Activity übergeben werden:

- Caption und Text für den Inhalt der Messagebox
- die anzuzeigenden Schaltflächen und
- das gewünschte Icon

[61] Englisch für: behaupten
[62] Synonym: Test-driven development

Jene Schaltfläche welche bestätigt wurde, soll als Argument retour geliefert werden. Diese Aufgabe soll in zwei Varianten gelöst werden: in XAML und in Form von C# Code. Für beide Varianten sollen zusätzlich Unit Tests umgesetzt werden. Für die imperative Variante ist ein Design inkl. einem Icon vorzusehen.

Die konkreten Umsetzungen sind im Anhang A zu finden.

- Deklarative Lösung: siehe Custom Activity in XAML
- Imperative Lösung: siehe Custom Activity in Code

4.4 Workflow

Collins definiert einen Workflow als eine Collection in sich verschachtelter Klassen und deren Eigenschaften. [Col10 S. 24]

Folgende Quellcodezeilen wurden in den vorangegangenen Beispielen bereits öfters erfolgreich eingesetzt. Es wird damit eine statische Methode deklariert, welche ein Objekt vom Type Activity an den Verwender der Methode zurückliefert.

```
static Activity CreateWorkflow()
{
   //…
   return new Sequence()
   {
   //…
   }
}
```

Quellcode 18 Codefragment für das Erzeugen eines Workflows

Im konkreten Fall wird eine Sequence zurückgeliefert. Wie aus Abbildung 13 (Seite 47) hervorgeht, ist die Klasse Sequence von der Klasse Activity abgeleitet. Somit erkennt der Compiler dies als typensicher an. [Col10 S. 25]

Das Ergebnis der Methode CreateWorkflow wurde dann unmittelbar zu Ausführung gebracht. Somit ist die Behauptung von Collins korrekt, dass ein Workflow eine Collection verschachtelter Activities sei.

Schlussfolgern kann man daraus, dass ein Workflow im einfachsten Fall aus einer einzelnen Activity bestehen kann.

4.4.1 Austausch von Argumenten mit einem Workflow

In der Regel müssen an einen Workflow Argumente übergeben werden bzw. liefert der Workflow Argumente retour. Genauer gesagt müssen dem Host, welcher den Workflow zur Ausführung bringt, die Argumente übergeben werden. Nach der Ausführung muss dieser Host die Ergebnisse des Workflows bereitstellen. Der Austausch von Argumenten an Workflows erfolgt über Klassen, welche das Interface IDictionary implementiert haben. Bei IDictionary handelt es sich um eine Kombination aus Schlüssel und Wert[63].

IDictionary selbst ist von ICollection abgeleitet. [Küh091 S. 338 ff.]

4.4.2 Verwendung generischer Activity Klassen

Aus Abbildung 13 (Seite 47) ist zu erkennen, dass neben den bis jetzt verwendeten Klassen für die Activities auch generische Ausprägungen davon existieren. Diese können mittels einer Return Anweisung jeden beliebigen Datentyp retour liefern.

```
public sealed class MyInteger: CodeActivity<int>
{
        //…
        protected override int Execute(CodeActivityContext context)
        {
                int x = 0;
        //…
                return x;
        }
}

//…

int result = WorkflowInvoker.Invoke(new MyActivities.MyInteger());
```

Quellcode 19 Codefragment für die Anwendung einer generischen CodeActivity

Zuerst ist im obigen Beispiel die Definition einer Klasse vom Type CodeActivity<T> zu erkennen. In der überschriebenen Execute Methode wird mittels einer Return Anweisung ein Objekt vom Typ Integer an den Verwender zurückgeliefert. Im zweiten Teil sieht man die Verwendung von WorkflowInvoker und wie damit das Ergebnis unmittelbar einer Variablen zugewiesen werden kann.

---

[63] Auf Englisch: Key Value Pair, kurz KVP

Eine konkrete Umsetzung ist in Anhang A zu finden.

- siehe WorkflowInvoker

4.4.3 Deklarative Workflows

Wie bereits erörtert, können Workflows vollständig deklarativ erstellt und in Form einer XAML-Datei gespeichert werden. Das folgende Codefragment zeigt den Beginn einer Workflow Definition im XAML Format.

```
<Activity mc:Ignorable="sap"
x:Class="WorkflowConsoleApplication.Workflow1" … >
<x:Members>
      <x:Property Name="Aufrufparameter" Type="InArgument(x:String)" />
      <x:Property Name="Returnparamater" Type="OutArgument(x:String)" />
      <x:Property Name="AufrufUndReturn" Type="InOutArgument(x:String)" />
</x:Members>
</Activity>
```

Quellcode 20 Codefragment einer Workflow Definition in XAML

Deutlich zu erkennen ist, dass auch hier lediglich eine Activity deklariert wird. Dies entspricht der Behauptung von Collins unter Punkt 4.4 (Seite 66), dass ein Workflow eine Collection von Activities sei.

Weiters sind in dem XAML Fragment drei Argumente dargestellt. Das Attribut Type kennzeichnet dabei die Richtung und den Datentyp des Argumentes.

Um einen deklarativen Workflow, welcher in Form einer XAML-Datei vorliegt, ausführen zu können, muss dieser in den Speicher geladen und in eine Activity transformiert werden. Dazu in der Lage ist die statische Methode Load der statischen Klasse ActivityXamlServices aus dem Namespace System.Activities.XamlIntegration. Die Methode Load verfügt über mehrere Überladungen. So können Streams, Dateinamen oder auch diverse Reader als Quellen verwendet werden. Folgendes Beispiel demonstriert die Verwendung der Methode Load.

```
// Laden des Workflows aus einem Stream und
// transformieren in eine Activity
Activity wf = ActivityXamlServices.Load(stream);
```

Quellcode 21 Verwendung von ActivityXamlServices.Load

Wie in dem Beispiel zu erkennen ist, liefert die Methode Load ein Objekt vom Type Activity retour. Dieses kann unmittelbar zur Ausführung gebracht werden.

Das Beispiel im Anhang A:

- LoadWfFromXAML

demonstriert die Verwendung der MethodeLoad der Klasse ActivityXamlServices.

Schlussfolgerung:

Denkbar wären somit auch Lösungen, welche die Workflow Definition in einer SQL Datenbank vorhalten. Dies könnte für die praktische Umsetzung in unserem Unternehmen von Bedeutung sein.

4.5 Hosts für die Ausführung eines Workflows

Ein wesentlicher Bestandteil der Integration einer Workflow Lösung in eine Software, ist der Host. Dessen Aufgabe ist ja, die übergebene Workflow Definition zur Ausführung zu bringen.

Damir vergleicht dies mit einem „*virtuellen Betriebssystem*“ [Dob09 S. 84] für den Workflow.

Microsoft bietet mehrere Möglichkeiten dafür an:

- WorkflowInvoker
- WorkflowApplication
- WorkflowServiceHost
- AppFabric

Diese unterschiedlichen Hosts unterscheiden sich z. B. grundlegend in Funktionalität, Komplexität und Einsatzgebiet.

4.5.1 WorkflowInvoker

Die einfachste Art und Weise um einen Workflow auszuführen, bietet die statische Klasse WorkflowInvoker mit ihrer statischen Methode Invoke. Invoke führt den Workflow synchron im Thread des Aufrufers auf. WorkflowInvoker verfügt zwar auch über Methoden um einen Workflow asynchron auszuführen, bietet aber generell keine Möglichkeit, die Workflow Instanzen zu steuern. Aus diesen Gründen wird

WorkflowInvoker häufig für Testumgebungen und kleinste Applikationen eingesetzt. [Mic1006]

Da WorkflowInvoker den Workflow im Thread der Anwendung synchron ausführt, ist die Anwendung so lange blockiert, bis der Workflow beendet ist. [Col10 S. 143]

Die Klasse WorkflowInvoker wurde in den vorangegangen Beispielen bereits laufend verwendet. Hinlängliche Beschreibungen zur Verwendung sind unter den Punkten 4.3.3 und 4.4.2 zu finden.

Das Beispiel WorkflowInstances im Anhang liefert einen Vergleich für WorkflowInvoker und WorkflowApplication bei der Ausführung eines Workflows.

4.5.2 WorkflowApplication

Im Gegensatz zu WorkflowInvoker bietet WorkflowApplication eine sehr detaillierte Kontrolle über die auszuführende Instanz. WorkflowApplication führt den Workflow in einem eigenen Thread asynchron aus. Änderungen am Zustand des Workflows werden mittels Ereignissen den aufrufenden Thread bekannt gegeben. Es sei in diesem Zusammenhang nochmals auf Abbildung 7 (Seite 26) verwiesen.

Die Klasse WorkflowApplication informiert mittels folgender sechs Ereignisse die Anwendung über Änderungen des internen Zustandes der Instanz.

Diese Ereignisse lauten:

- Aborted
- Completed
- Idle
- OnUnhandledException
- PersistableIdle
- Unloaded

Unter Punkt 3.4.4 (Seite 38 ff.) sind diese Ereignisse bereits beschrieben.

Eine konkrete Anwendung der Klasse WorkflowApplication ist im Beispiel WorkflowInstances im Anhang A zu finden. Dieses Beispiel veranschaulicht weiters, wie auf die oben angeführten Ereignisse reagiert werden kann.

4.5.3 WorkflowServiceHost

Objekte der Klasse WorkflowServiceHost ermöglichen es, einen Workflow als WCF Dienst für Clientanwendungen zu konfigurieren und verfügbar zu machen.

WorkflowServiceHost ist in der Lage WCF Nachrichten zu empfangen. Nach dem Eintreffen der entsprechenden Nachricht wird ein Workflow durch Verwendung der Klasse WorkflowService instanziiert.[64]

Im Allgemeinen spricht man dann von einem WorkflowService, wenn die Ablaufsteuerung mittels Windows Workflow Foundation realisiert ist und für die Kommunikation die Windows Communication Foundation verwendet wird. [Cha09 S. 18]

4.5.4 Windows Server AppFabric

Ein Workflow Service basiert auf der WCF, ist über SOAP[65] ansprechbar und implementiert einen Workflow. Der Application Server Dublin[66] bietet dafür den passenden Application Host. [Cha09 S. 20], [Dob101 S. 100]

Zum Zeitpunkt der Erstellung dieser Diplomarbeit war Windows Server AppFabric lediglich als ß2 Version verfügbar.

Microsoft möchte mit Windows Server AppFabric eine Infrastruktur für das Hosten und Verwalten von Workflows anbieten, welche innerhalb von Anwendungen im Internet Information Server laufen. [Cha09 S. 21]

Abbildung 22 liefert einen Überblick über Windows Server AppFabric. Deutlich zu erkennen sind die Bereiche Persistence, Hosting, Monitoring und Caching.

---

[64] Das Anbieten eines Dienstes spielt für den konkreten Anwendungsfall in unserem Unternehmen keine Rolle. Aus diesem Grund wird der WorkflowServiceHost nicht näher behandelt! Er wurde lediglich der Vollständigkeit halber angeführt.

[65] SOAP (Simple Object Access Protocol) ist ein Netzwerkprotokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Remote Procedure Calls durchgeführt werden können.

[66] Da die konkrete Verwendung von Windows Server AppFabric derzeit für unsere Anwendung nicht in Betracht kommt, wird in dieser Diplomarbeit nicht näher darauf eingegangen.

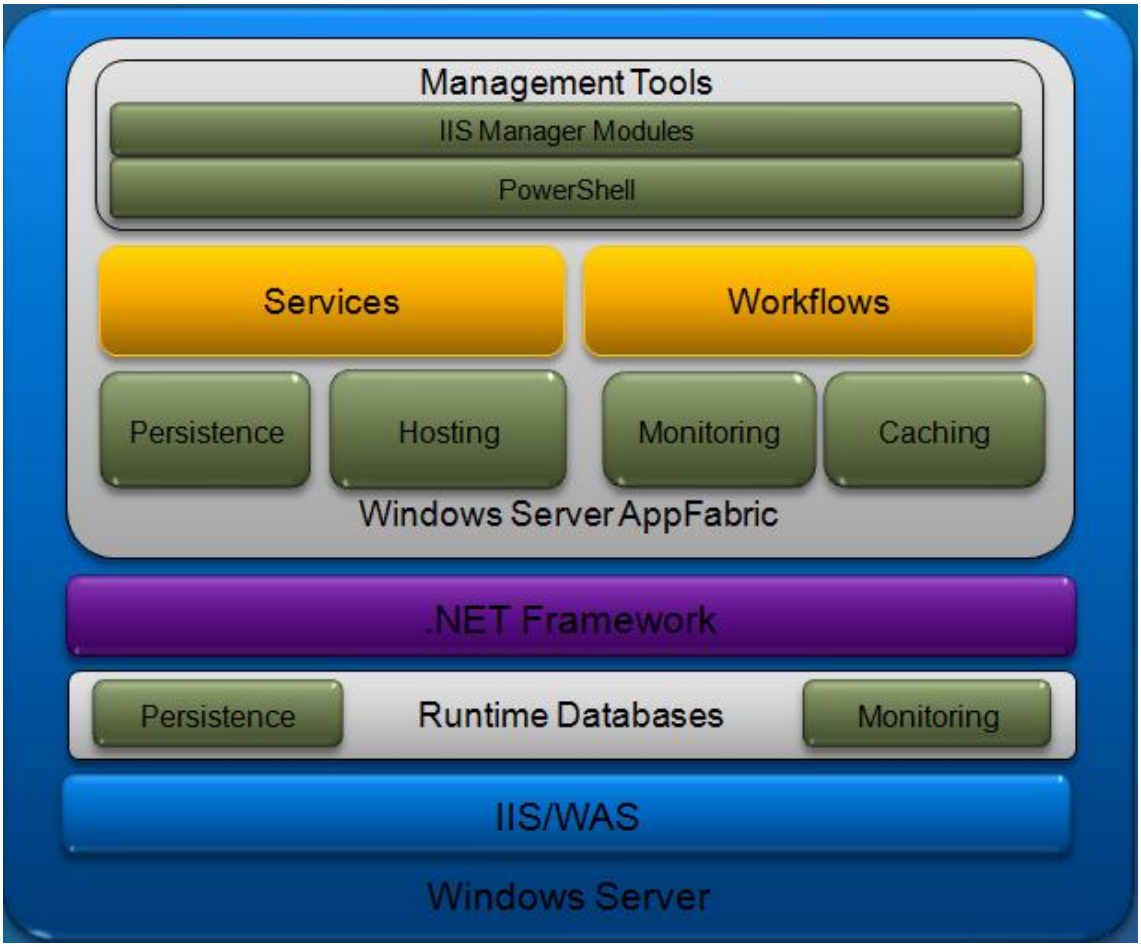


Abbildung 22 Windows Server AppFabric (Quelle: [Jac10])

4.6 Workflow Extensions

Extensions erweitern eine Workflow Lösung um ein konfigurierbares Verhalten. Die vorhandenen Activities definieren die einzelnen Schritte im Workflow. Das Umfeld, in welchem diese ausgeführt werden, kann mittels Extensions beeinflusst werden. [Col10 S. 211]

Auch Milner betont, dass die leichtgewichtige Workflow Run Time mittels Extensions um Funktionalitäten erweitert werden kann. [Mil09]

Extensions zeichnen sich durch folgende zwei Eigenschaften aus:

- Extensions sind konfigurierbar.
- Der Zugriff auf Extensions kann von der Host Anwendung als auch von den Activities erfolgen.

Neben den vordefinierten Extensions für Persistenz und das Aufzeichen von Tracking Informationen können auch eigene Extensions entwickelt und verwendet werden.

Ein häufiger Anwendungsfall für eine einfache Extension ist z. B. die Übergabe eines Connection-Strings für einen Datenbankzugriff von der Host Anwendung an die Activities.

Die Quellcode Beispiele Quellcode 22 und Quellcode 23 veranschaulichen dies. Bei dem Objekt SimpleExtionsion handelt es sich um eine sehr einfache Klasse. Im Konstruktor der Klasse kann ein Connection-String übergeben werden. Dieser kann in späterer Folge mittels einer öffentlichen Eigenschaft abgefragt werden.

Zunächst wird eine neue Instanz der Klasse WorkflowApplication erzeugt. Die Klasse WorkflowApplication verfügt über eine Eigenschaft mit dem Namen Extensions. Mittels einer Add Methode kann dieser Auflistung eine neue Extension hinzugefügt werden. Die wird im folgenden Quellcode realisiert.

```
// WorkflowApplication zum Ausführen des Workflows verwenden
WorkflowApplication instance = new WorkflowApplication(CreateWorkflow());

// Erzeugen eines Objektes vom Type SimpleExtension
// und Übergabe des Connection Strings zur Datenbank
SimpleExtension _dbExtension = new SimpleExtension(_connectionString);

// Fügt der Auflistung der Erweiterungen ein
// Objekt vom Typ SimpleExtension hinzu
instance.Extensions.Add(_dbExtension);
```

Quellcode 22 Hinzufügen einer Extension

Der Zugriff auf die Extension innerhalb einer Activity erfolgt über den Kontext der Activity. Das Beispiel Quellcode 23 veranschaulicht dies.

```
protected override void Execute(CodeActivityContext context)
{
       // Abrufen des Connection Strings und auslösen eines Fehlers,
       // sollte dieser nicht vorhanden sein
       SimpleExtension ext = context.GetExtension<SimpleExtension>();
       if (ext == null)
       {
              throw new InvalidProgramException
              ("Ungültiger Connection-String!");
       }
       //…
       DataClassesDataContext dc = new
       DataClassesDataContext(ext.ConnectionString);
       //…
}
```

Quellcode 23 Zugriff auf Eigenschaften einer Extension in einer Activity

Die Execute Methode stellt den Ausführungskontext zur Verfügung. Bei GetExtension handelt es sich um eine generische Methode. Dies liefert eine Extension vom angegeben Typ retour.

Das Beispiel im Anhang A:

- SimpleExtension

demonstriert die Verwendung einer Extension.

4.7 Persistenz

Unter Persistenz[67] versteht man die Eigenschaft eines Objekts, durch die seine Lebensdauer permanent und damit unabhängig von der Ausführung des erzeugenden Programmes wird. [DAT09]

Persistenz wird demnach also vor allem bei lang laufenden Workflows zum tragen kommen. Lang laufende Prozesse können sich dabei durchaus über mehrere Monate erstrecken.

Es gibt zwei Möglichkeiten, einen Workflow zu persistieren: [Dob10 S. 52]

- Der Host verwendet den Befehl Persist des Objektes WorkflowApplication.
- Durch die Verwendung der Persist Activity in einem Workflow.

Nach dem Eintreffen eines Persist Befehls bzw. nach dem Ausführen einer Persist Activity serialisiert die Workflow Run Time alles notwendige in einem Persistence Store. Dabei handelt es sich um ein Medium, in welchem die Informationen zum Zustand des Workflows dauerhaft abgelegt werden können. Im einfachsten Fall wird dafür eine SQL Datenbank verwendet.

Wie sich ein Workflow beim Persistieren verhalten soll, kann durch die Eigenschaft PersistableIdle der Workflow Instanz gesteuert werden. PersistableIdle liefert ein Objekt vom Type PersistableIdleAction retour. Dabei handelt es sich um eine Enumeration mit folgenden Werten: [Mic1011]

---

[67] Synonym: Beständigkeit

Tabelle 16 PersistableIdleAction Enumeration

Membername	Beschreibung
None	Gibt an, dass keine Aktion ausgeführt wird.
Unload	Gibt an, dass die WorkflowApplication den Workflow persistieren und entladen soll.
Persist	Gibt an, dass die WorkflowApplication den Workflow persistieren soll.

Im Beispiel Quellcode 24 wird die Verwendung von PersistableIdle demonstriert.

```
application.PersistableIdle = (e) =>
{
      return PersistableIdleAction.Unload;
};
```

Quellcode 24 Verwendung von PersistableIdle

Der Unterschied zwischen Unload und Persist ist jener, dass bei Unload die Workflow Instanz zusätzlich entladen wird und dadurch die belegten Ressourcen frei gegeben werden.

Um einen entladen und persistierten Workflow zu reaktivieren, ist die Methode Load der WorklfowApplication zu verwenden. Load dient dazu, die angegebene Workflowinstanz aus einem Instanzspeicher in den Arbeitsspeicher zu laden. Load erfordert die Angabe der gewünschten Workflow Instanz in Form einer Instanz ID. Dabei handelt es sich um eine 128 Bit GUID[68]. Beispiel Quellcode 25 zeigt die Verwendung von Load.

```
// laden eine Workflows aufgrund der ID
application.Load(id);

// an der Stelle des Lesezeichens wir der Workflow fortgesetzt
application.ResumeBookmark("Lesezeichen", input);
```

Quellcode 25 Reaktivierung eines persistierten Workflows

Weiters ist im obigen Beispiel ResumeBookmark[69] zu erkennen. ResumeBookmark kennzeichnet jene Stelle im Workflow, an der mit der Ausführung fortgesetzt werden soll. Ein solches Lesezeichen muss im Workflow definiert werden. In Tabelle 14 (Seite 58) ist bereits beschrieben, dass die Verwendung von Bookmarks nur möglich ist, wenn

[68] Globally Unique Identifier
[69] Bookmark: Englisch für Lesezeichen

die Custom Activity von NativeActivity abgeleitet ist. Quellcode 26 zeigt die Verwendung von CreateBookmark. Ein Bookmark benötigt lediglich ein Namen und eine Callback Methode, an der Fortgesetzt wird.

```
protected override void Execute(NativeActivityContext context)
{
      string name = this.BookmarkName.Get(context);
      // Erstellt einen Punkt, an dem eine NativeActivity-Aktivität
      // passiv warten kann, bis die Wiederaufnahme erfolgt.
      context.CreateBookmark(name, new BookmarkCallback(OnReadComplete));
}
```

Quellcode 26 Verwendung von CreateBookmak

Wie bereits erwähnt, wird der Zustand der Workflow Instanz in einem nicht flüchtigen Speicher serialisiert. Das Beispiel Quellcode 27 zeigt die notwendigen Schritte, um einen SQL Datenbank als InstanceStore zu konfigurieren.

```
// Initialisiert eine neue Instanz der SqlWorkflowInstanceStore-Klasse.
instanceStore = new SqlWorkflowInstanceStore(connectionString);

// Erstellt einen Instanzhandle.
InstanceHandle handle = instanceStore.CreateInstanceHandle();

// Führt einen Dauerhaftigkeitsbefehl synchron aus
InstanceView view = instanceStore.Execute(handle, new
CreateWorkflowOwnerCommand(), TimeSpan.FromSeconds(30));

handle.Free();
instanceStore.DefaultInstanceOwner = view.InstanceOwner;
```

Quellcode 27 Verwendung von SqlWorkflowInstanceStore

Bei InstanceStore handelt es sich um eine abstrakte Klasse, von der alle Persistence Provider abgeleitet werden. In diesem Beispiel wird die Klasse SqlWorkflowInstanceStore Klasse verwendet. Diese Verwendet einen SQL Datenbank. Um eine Verbindung zu einer SQL Datenbank herstellen zu können, ist ein Connection-String notwendig. Dieser wird SqlWorkflowInstanceStore im Konstruktor übergeben. Der Execute Methode erfordert die Übergabe eines Handle, eines Command und einem TimeOut Wert und liefert dafür ein Objekt vom Type InstanceView. InstanceView entspricht in etwas einem Connection Handle. [Col10 S. 197]

Zu beachten dabei ist, dass eine Workflow Instanz nicht zwangsweise auf jenem Host fortgesetzt werden muss, auf dem er persistiert wurde. Dies ist ja eine wesentliche Forderung aus Punkt 3.5.1 (Seite 40).

Das Beispiel Persistenz im Anhang A demonstriert die Verwendung einer SqlWorkflowInstanceStore zum persistieren eines einfachen Workflows.

Für die Konfiguration der notwendigen Tabellen stellt Microsoft zwei SQL Skripte bereit. Mittels der beiden Dateien `SqlWorkflowInstanceStoreSchema.SQL` und `SqlWorkflowInstanceStoreLogic.SQL` kann eine SQL Datenbank um die benötigten Tabellen und Verweise erweitert werden. Die Beiden Dateien sind im Anhang zu finden.

Abbildung 23 veranschaulicht den relativ einfachen Worklfow aus dem Beispiel Persistenz. Dieses Beispiel besteht aus zwei Projekten: Phase 1 und Phase 2.

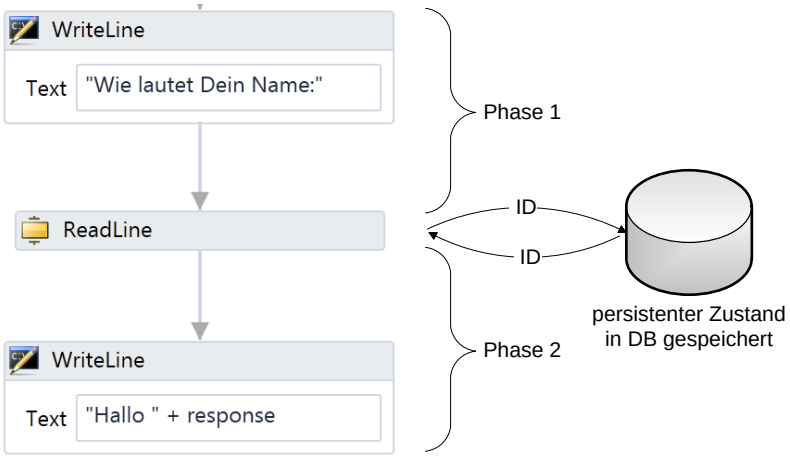


Abbildung 23 Erklärung zu Beispiel Persistenz im Anhang A

Das Projekt Phase 1 fordert mittels einer WriteLine Activity den Benutzer zur Eingabe seines Namens auf. Darauf folgt die Custom Activity ReadLine. ReadLine erstellt ein Bookmark mit dem Namen „Lesezeichen“ und setzt den Wert der Eigenschaft CanInduceIdle auf True. Dies veranlasst die WorkflowApplication Instanz dazu, den Workflow zu persistieren und zu entladen. Anschließend wird die Applikation Phase 1 beendet. Die Applikation Phase 2 zeigt zunächst alle in der Datenbank gefunden Workflow Instanzen an. Nach Auswahl einer Instanz wird diese in geladen und fordert erneut zur Namenseingabe auf. Im Anschluss wird der Workflow beim Bookmark fortgesetzt.

4.8 Tracking

Bei Tracking[70] handelt es sich um eine Workflow Extension, welche eine Infrastruktur für die Überwachung eines Workflows bereit stellt. Dabei werden Datensätze über Schlüsselereignisse gesammelt, gefiltert und für die Weiterverarbeitung angeboten. Um einen Workflow überwachen zu können, müssen keine Änderungen oder Anpassungen am Workflow selbst vorgenommen werden – jeder Workflow kann demnach überwacht werden. [Mic1010]

In Abbildung 24 ist der Aufbau einer Überwachungsinfrastruktur dargestellt. Es kommt dabei zur Anwendung eines „*Veröffentlichen-und-Abonnieren-Paradigma*“ [Mic1010].

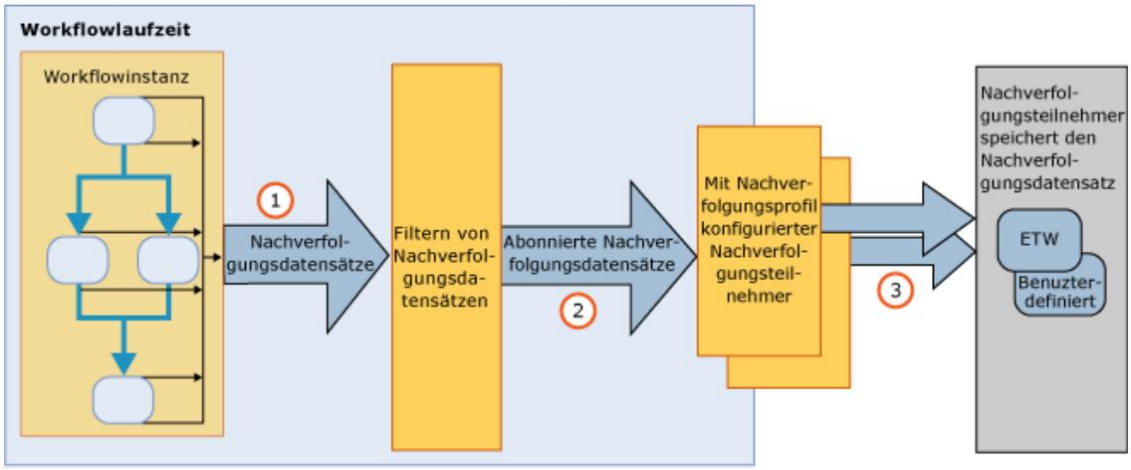


Abbildung 24 allgemein gehaltene Ansicht der Überwachungsinfrastruktur (Quelle: [Mic1010])

Eine Workflow Instanz veröffentlicht im Zuge der Ausführung eines Workflows Nachverfolgungsdatensätze[71]. Abonnenten[72] dieser Datensätze werden in Form von Extensions registiert und haben Zugriff auf die gesammelten Tracking Records. Die Überwachungsinfrastruktur ermöglicht die Anwendung von Filtern. Ein Überwachungsteilnehmer ist somit in der Lage, eine Teilmenge aller vorhandenen Überwachungseinträge zu abbonieren. Auf welche Art und Weise der Überwachungsteilnehmer die Überwachungseinträge verarbeitet, bleibt ihm überlassen.

[70] Englisch für: Nachverfolgung
[71] Synonym: Tracking Records
[72] Synonyme: Überwachungsteilnehmer, Tracking Participants

Als Einsatzgebiete für die Überwachung bieten sich z. B. folgende Szenarien an:

- Diagnose von Workflows
- Analyse von geschäftsrelevanten Daten in „Echtzeit“

4.8.1 Event Tracing for Windows (ETW)

Event Tracing for Windows (ETW) verwendet die Ereignisanzeige des Betriebssystems zum Verwalten der Protokolleinträge. [Mil09]

Genauer gesagt, werden die Anwendungs- und Dienstprotokolle dafür verwendet. Diese bilden eine neue Kategorie von Ereignisprotokollen. Darin werden keine Ereignisse mit systemweiten Auswirkungen gespeichert, sondern Ereignisse einzelner Anwendungen oder Komponenten. Der Bereich der Analytischen Protokolle ist gedacht für die Aufzeichnung von Fehlern in Anwendungen, welche nicht durch einen Benutzereingriff gelöst werden können. Standardmäßig sind diese Protokolle ausgeblendet. Abbildung 25 zeigt einen Ausschnitt aus der Windows Hilfe. Darin sind die notwendigen Schritte für die Aktivierung angeführt.

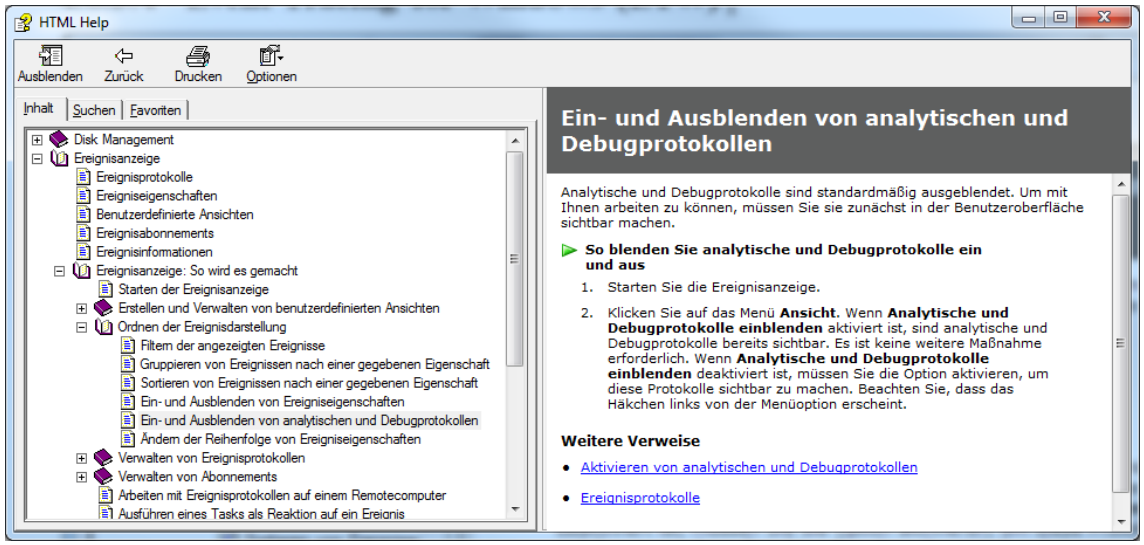


Abbildung 25 Ein- und Ausblenden von analytischen und Debugprotokollen

In Abbildung 26 ist ein Bildschirmausschnitt der Windows Ereignisanzeige dargestellt. Der dargestellte Bereich zeigt, dass die Ausführung einer WriteLine Activity begonnen hat.

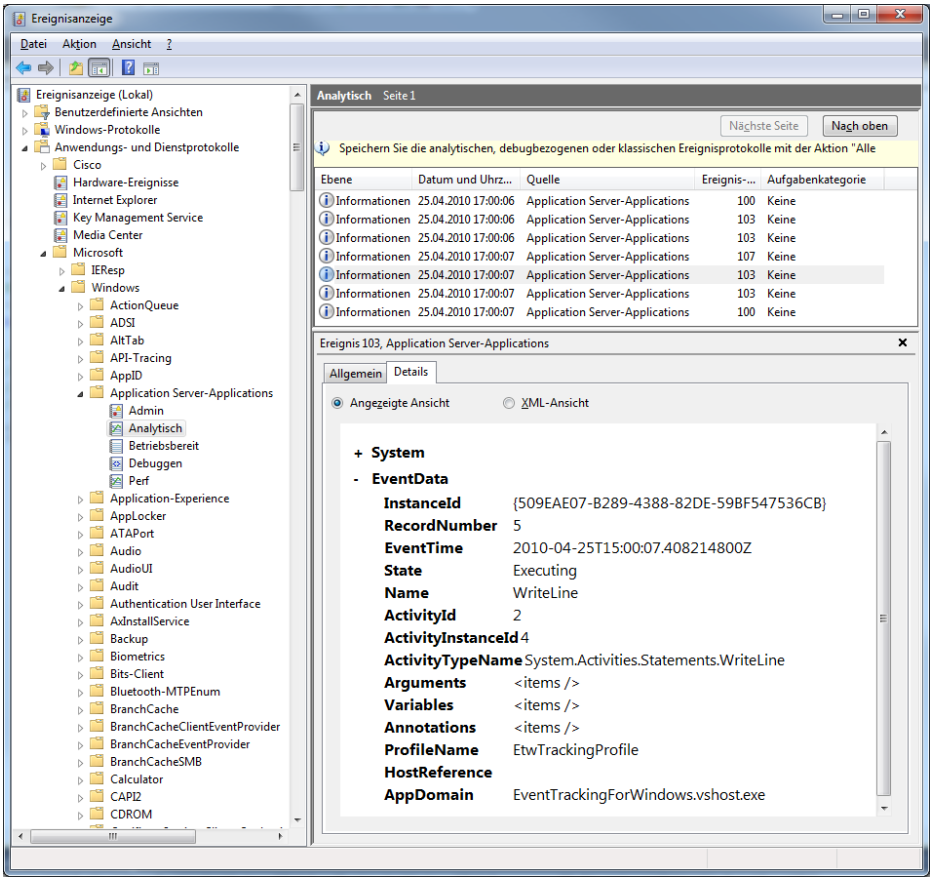


Abbildung 26 Windows Ereignisanzeige

Der Zugriff auf das Ereignisprotokoll muss nicht über die Ereignisanzeige von Windows erfolgen. Die Einträge können auch programmtechnisch ausgewertet werden. Weiters stehen Tools wie z. B. `wevtutil`[73] zur Verfügung.

Diese Art der Überwachung eignet sich demnach optimal für die Analyse von Fehlern und sollte nicht andere Zwecke verwendet werden.

Im folgenden sind die notwendigen programmtechnischen Schritte zum Aufzeichnen von Überwachungsinformationen im Windows Ereignisprotokoll angeführt.

In Abbildung 27 ist die Klassenhierarchie eines EtwTrackingParticipants[74] dargestellt. Wie man daraus erkennen kann, stammt diese Klasse von TrackingParticipant ab.

[73] Wevtutil dient zum Abrufen von Informationen zu Ereignisprotokollen, Ausführen von Abfragen und Exportieren, Archivieren und Löschen von Protokollen.
[74] Englisch für: Teilnehmer

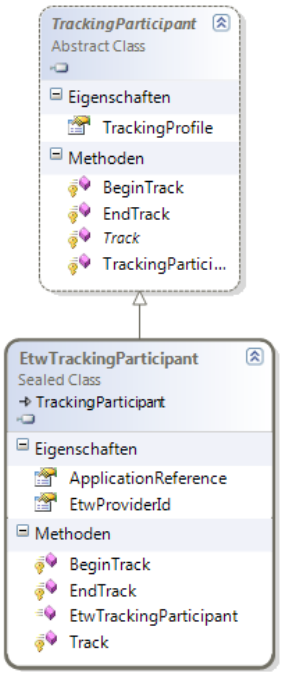


Abbildung 27 Klassenhierarchie eines EtwTrackingParticipants

Bei einem EtwTrackingParticipant handelt es sich um einen Verbraucher von Workflownachverfolgungsdaten, der sich um die Weitergabe der Daten an das Windows Ereignisprotokoll kümmert. Im Beispiel Quellcode 28 ist die Deklaration einer Variable vom Type EtwTrackingParticipant ersichtlich.

```
private EtwTrackingParticipant _etwTracking;
```

Quellcode 28 Deklaration eines EtwTrackingParticipants

Quellcode 29 zeigt, die Konfiguration eines TrackingProfiles. Ein TrackingProfile dient zur Definition, welche Ereignisse überwacht werden sollen. Dazu enthält ein TrackingProfile eine Eigenschaft Names Queries. [Col10 S. 234]

In Tabelle 17 sind die möglichen Arten von Überwachungsdatensätze dargestellt.

Tabelle 17 Einteilung von Überwachungsdatensätzen

Query	**Beschreibung**
WorkflowInstanceQuery	Beinhaltet Daten über die Workflow Instanz.
BookmarkResumptionQuery	Beinhaltet Daten im Zusammenhang mit Lesezeichen und deren Fortsetzung.
ActivityStateQuery	Beinhaltet Daten über eine spezifische Activity.
CustomTrackingQuery	Beinhaltet benutzerdefinierte Überwachungsdaten.

Im TrackingProfile kann nun mittels der Queries Eigenschaft definiert werden, welche Ereignisse überwacht werden sollen. Dies ist im Folgenden Beispiel Quellcode 29 dargestellt.

```
// Ein Consumer von Workflownachverfolgungsdaten,
// der ein Ereignisablaufverfolgungs-Ereignis
// (Event Tracking for Windows, ETW) an eine ETW-Sitzung ausgibt,
// die die Daten aus dem Nachverfolgungsdatensatz enthält.
_etwTracking = new EtwTrackingParticipant()
{
	// Erstellt in einem TrackingParticipant ein
	// Abonnement für Workflownachverfolgungsdatensätze
	TrackingProfile = new TrackingProfile()
	{
		Name = "EtwTrackingProfile",

		// Ruft die Auflistung von TrackingQuery-Objekten ab,
		// die die Datensätze definieren, die dieses
		// Nachverfolgungsprofil abonniert.
		Queries =
		{
			// Aufzeichnen von Starten und Beenden
			// von Workflow Instanzen
			new WorkflowInstanceQuery()
			{
				States =
				{
					WorkflowInstanceStates.Started,
					WorkflowInstanceStates.Completed
				},
			},

			// Alle Informationen zu Lesezeichen aufzeichnen
			new BookmarkResumptionQuery()
			{
				Name = "*"
			},

			// alle Ereignisse von WriteLine
			// Activities aufzeichnen
			new ActivityStateQuery()
			{
				ActivityName = "WriteLine",
				States = { "*" },
			},

			// Aufzeichnen eigener Tracking-Informationen
			// in allen Activities
			new CustomTrackingQuery()
			{
				Name = "*",
				ActivityName = "*"
			}
		}
	}
};
```

Quellcode 29 Erstellen eines EtwTrackingParticipant und eines TrackingProfile

Zuletzt muss der konfigurierte TrackingParticipant als Exentsion zur Workflow Instanz hinzugefügt werden. Dies ist im Beispiel Quellcode 30 zu erkennen.

```
// Hinzufügen zu den Extensions
instance.Extensions.Add(_etwTracking);
```

Quellcode 30 Hinzufügen eines EtwTrackingParticipants

Das Beispiel

- EventTracking for Windows (ETW)

im Anhang A demonstriert die Umsetzung.

4.8.2 Tracking Informationen in einer SQL Datenbank speichern

Für die Aufzeichnung von Überwachungsinformationen über einen längeren Zeitraum eignet sich natürlich auch eine SQL Datenbank. Um dies zu realisieren, ist es notwendig, einen eigenen TrackingParticipant zu erstellen. In Abbildung 28 ist eine entsprechende Klasse abgebildet. Ähnlich wie beim EtwTrackingParticipants aus Abbildung 27, ist auch der SqlTrackingParticipant von der Klasse TrackingParticipant abgeleitet.

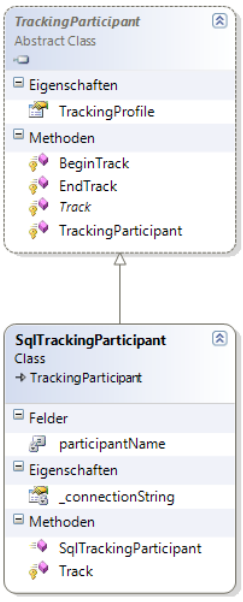


Abbildung 28 Klassenhierarchie eines SQL Tracking Participant

Wesentlich für die Implementierung ist dabei ein überschreiben der Methode Track. Im Beispiel Quellcode 31 ist der Rumpf einer entsprechenden Methode dargestellt. Der

Methode Track wird ein Objekt vom Typ TrackingRecord übergeben. Alle Nachverfolgungsdatensätze eines Workflows sind von dieser abstrakten Klasse abgeleitet. Mögliche Ausprägungen sind: [Mic1012]

- ActivityScheduledRecord
- ActivityStateRecord
- BookmarkResumptionRecord
- CancelRequestedRecord
- CustomTrackingRecord
- FaultPropagationRecord
- WorkflowInstanceRecord

Im folgenden Beispiel wird nun geprüft, um welche Art es sich bei der konkreten TrackingRecord Instanz handelt, indem mittels dem Operator AS eine Konvertierung durchgeführt wird.

```
protected override void Track(TrackingRecord record, TimeSpan timeout)
{
       WorkflowInstanceRecord instanceTrackingRecord = record as
       WorkflowInstanceRecord;
       if (instanceTrackingRecord != null)
       {
       // Einfügen eines neuen Datensatzes in die Tabelle TrackInstance
       }

       BookmarkResumptionRecord bookTrackingRecord = record as
       BookmarkResumptionRecord;
       if (bookTrackingRecord != null)
       {
       // Einfügen eines Datensatzes in die Tabelle TrackBookmark
       }

       ActivityStateRecord activityStateRecord = record as
       ActivityStateRecord;
       if (activityStateRecord != null)
       {
       // Einfügen eines Datensatzes in die Tabelle TrackActivity
       }

       CustomTrackingRecord customTrackingRecord = record as
       CustomTrackingRecord;
       if (customTrackingRecord != null)
       {
       // Einfügen eines Datensatzes in die Tabelle TrackUser
       }
}
```

Quellcode 31 Realisierung eines SqlTrackingParticipant

Nach der Identifizierung von TrackingRecord können die entsprechenden Informationen in einer SQL Datenbank serialisiert werden. Der Zugriff auf eine SQL Datenbank kann z. B. mittels LinqToSql erfolgen. In Abbildung 29 sind die erstellen Klassen für einen SqlTrackingParticipant abgebildet.

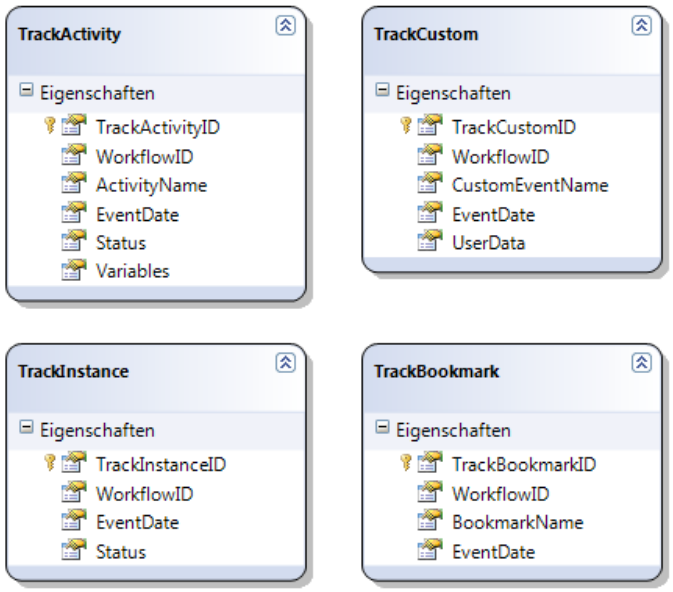


Abbildung 29 LinqToSql Klassen für den SqlTrackingParticipant

Das Beispiel

- SQLTrackingParticipant

Im Anhang A zeigt eine konkrete Umsetzung davon.

4.9 Umgang mit Fehlern

Die Tatsache, dass Workflows deklarativ aufgebaut sind, verhindert nicht, dass sie nicht auch Fehler verursachen können.

Die Basic Activity Library bietet eine TryCatch Activity an. Im Try Bereich sind jene Activities zu platzieren, welche potenziell einen Fehler erzeugen können. Im Catch Bereich sind ein oder mehrere Catch Objekte zu platzieren. Für jeden erwarteten Fehler muss ein eigens Catch Objekt vorhanden sein. Der Finaly Bereich ist optional. Er wird

nach den Try Activities bzw. nach sämtlichen Catch Activities aufgerufen. [Col10 S. 70]

In Abbildung 30 ist eine TryCatch Activity dargestellt. Zu erkennen ist, dass im Catch Bereich nur eine Art von Exceptions behandelt wird: eine OutOfStockException.

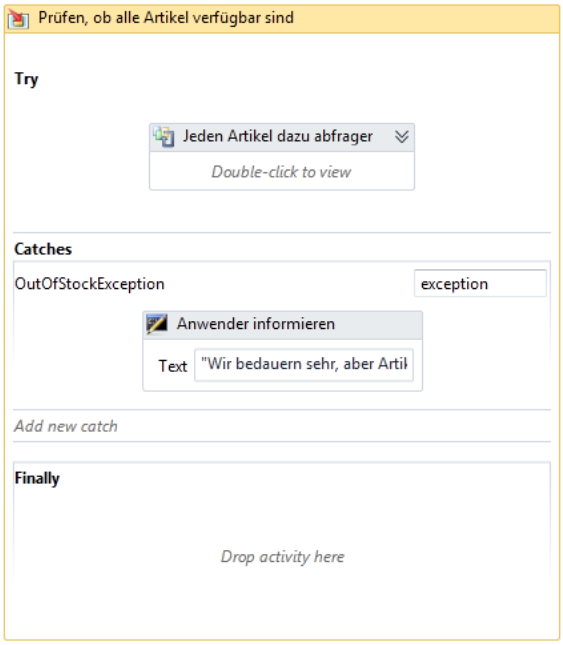


Abbildung 30 TryCatch Activity

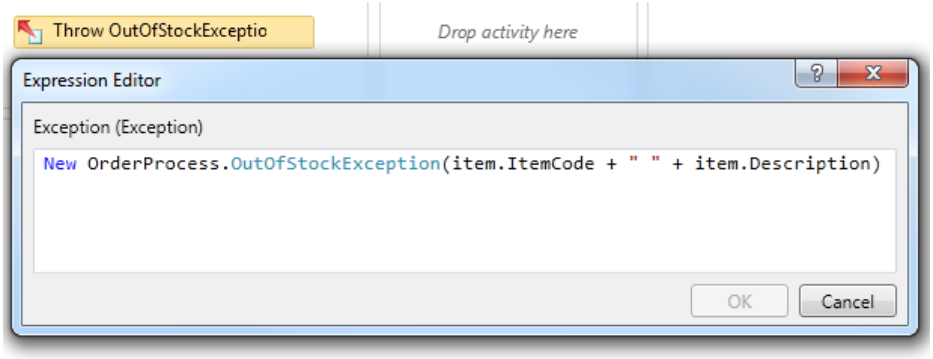
Bei dieser OutOfStockException handelt es sich um eine selbst definierte Exception. Dies kann in Form einer eigenen Klasse, welche von Exception abgeleitet wird, realisiert werden. Beispiel Quellcode 32 zeigt ein Codefragment einer entsprechenden Klasse inkl. einem Konstruktor, welcher einen String erwartet.

```
public class OutOfStockException : Exception
{
	public OutOfStockException(string message): base(message)
	{
		// ...
	}
}
```

Quellcode 32 Fragment für das Definieren einer Klasse für eigene Exceptions

Ausgelöst kann eine solche Exception mittels einer Throw Activity werden. [Col10 S. 73]

In Beispiel Quellcode 33 ist der Inhalt der Exception Eigenschaft der Throw Activity dargestellt. OrderProcess kennzeichnet in diesem Fall den Namespace, in welchem die OutOfStockException zu finden ist.



Quellcode 33 Aufruf einer eigenen Exception mittels einer Throw Activity

Das Beispiel ExceptionHandling im Anhang demonstriert, wie eine Fehlerbehandlung in einen deklarativen Workflow integriert werden kann.

Analog zum Try Catch Konstrukt in C#, wandern auch Workflow Exceptions in der Hierarchie nach oben, bis sie behandelt werden. Wichtig ist nun die Positionierung der Fehlerbehandlung. [Col10 S. 77]

In Quellcode 34 ist fast der gesamte Workflow in einem Try Bereich behandelt. Dies ist natürlich ein sehr einfacher Weg, um mit Sicherheit alle Fehler behandeln zu können, aber man verliert damit schnell die Möglichkeit gezielt einen Fehler abzufangen.

Zur Erklärung: Wenn eine Exception auftritt, und diese wird in der eigentlichen Activity nicht behandelt, so wird diese Activity abgebrochen und damit auch alle untergeordneten Activities nicht mehr ausgeführt. Wenn im Beispiel Quellcode 34 ein Artikel als fehlend erkannt wird, so wird im Then Zweig einer If Activity eine Exception ausgelöst. Der Try Block ist aber erst bei der ForEach Schleife angebracht. Da diese Exception nicht früher behandelt wird, steigt sie in der Aufrufliste nach oben. Somit wird auch die ForEach Activity vorzeitig beendet. Das bedeutet, dass beim ersten Auftreten eines Fehlers keine weiteren Prüfungen durch den Workflow erfolgen. Dabei kann es sich nun um ein beabsichtigtes Verhalten oder um einen falsch entworfenen Geschäftsprozess handeln.

```
Sequence
{
   PrüfeBestand (Try)
   {
      Prüfe jeden Artikel(ForEach)
      {
         Wenn der Artikel nicht verfügbar ist(If)
         {
            Exception auslösen (Then)
         }
      }
   }
   (Catch)
   {
   }

   Restlichen Activities
}
```

Quellcode 34 Positionierung der Try Catch Activities, Pseudocode Variante 1

Sobald ein Fehler auftritt, wird die Ausführung unterbrochen und im Catch Block fortgesetzt. Der Workflow wird dann vor dieser Position fortgesetzt.

Im Beispiel Quellcode 35 wurde der Try Catch Block unmittelbar um die If Activity gelegt. Damit wird eine Exception noch for der ForEach Activity behandelt. Somit werden auch die restlichen Artikel geprüft.

```
Sequence
{
   Prüfe jeden Artikel(ForEach)
   {
      (Try)
      {
         Wenn der Artikel nicht verfügbar ist(If)
         {
            Exception auslösen (Then)
         }
      }
      (Catch)
      {
      }
   }
   Restlichen Activities
}
```

Quellcode 35 Positionierung der Try Catch Activities, Pseudocode Variante 2

Im letzten Beispiel zu dieser Thematik, zu sehen in Quellcode 36, wurde der Try Catch Block außerhalb der Sequence platziert. Somit kann erreicht werden, dass bei einer Exception der ganze Workflow nicht mehr ausgeführt wird.

```
(Try)
{
   Sequence
   {
      Prüfe jeden Artikel(ForEach)
      {
         Wenn der Artikel nicht verfügbar ist(If)
         {
            Exception auslösen (Then)
         }
      }
      Restlichen Activities
   }
}
(Catch)
{
}
```

Quellcode 36 Positionierung der Try Catch Activities, Pseudocode Variante 3

4.10 WorkflowDesigner

Das Bereitstellen eines einfach zu implementierenden Designers für die Darstellung und für das Bearbeiten von Workflows war eine der Hauptanforderungen an das Entwicklungsteam der Windows Workflow Foundation 4.0. Die Designer der früheren Versionen waren zwar funktionell, aber sehr komplex in der Handhabung. Der aktuelle Designer kann mit wenig Aufwand in einer WPF Anwendung dargestellt werden. [Mil09]

In Abbildung 31 ist eine minimalistische WPF Anwendung zu sehen, die als zentrales Element einen Designer beinhaltet. Im Linken Bereich werden Activities zur Verwendung angeboten und im rechten Bereich die jeweiligen Eigenschaften dazu.

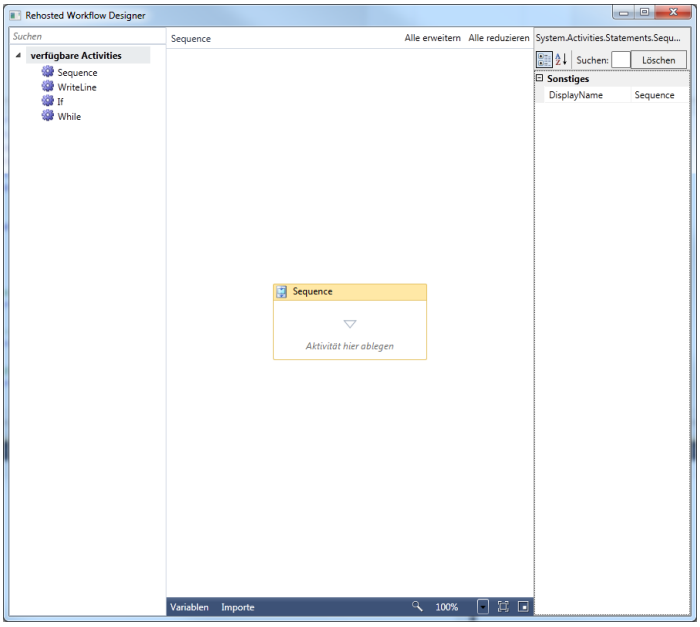


Abbildung 31 WPF Anwendung mit einem Workflow Designer

In Beispiel Quellcode 37 ist der wesentliche Teil des dazu notwendigen C# Programmcodes dargetsellt.

```
// MetaData registrieren
new DesignerMetadata().Register();

// Erstellen eines Workflow Designers
WorkflowDesigner wd = new WorkflowDesigner();

// Designer mit einer neuen Sequence Activiy füllen
wd.Load(new Sequence());

// Darstellen des Designers in der Mitte der Anwendung
BereichDesigner.Child = wd.View;

// Darstellen der Eigenschften im Rechten Bereich
BereichProperty.Child = wd.PropertyInspectorView;
```

Quellcode 37 Verwenden des Workflow Designers

Zunächst ist es notwendig, die für den Designer benötigen Metadaten zu registrieren. Am Anschluss daran wird ein neues Objekt vom Typ WorkflowDesigner instanziiert und mit einer leeren Sequence Activity gefüllt. Die letzten beiden Zeilen weisen die Views den entsprechenden Bereichen in einem WPF-Grid zu.

Im Beispiel Quellcode 38 Ist das Fragment einer XAML Datei dargestellt. Am Beginn des Codefragments ist zu erkennen, dass der Namespace System.Activities.Presentation.Toolbox hinzugefügt wird. Dies ist notwendig, um in Folge die erlaubten Activities darzustellen.

```
xmlns:sapt="clr-
namespace:System.Activities.Presentation.Toolbox;assembly=System.Activitie
s.Presentation"

…

<sapt:ToolboxControl>
      <sapt:ToolboxCategory CategoryName="verfügbare Activities">
            <!-- Sequence Activity -->
            <sapt:ToolboxItemWrapper AssemblyName=
                  "{StaticResource AssemblyName}" >

                  <sapt:ToolboxItemWrapper.ToolName>
                        System.Activities.Statements.Sequence
                  </sapt:ToolboxItemWrapper.ToolName>

            </sapt:ToolboxItemWrapper>

            <!-- WriteLine Activity -->

            <!-- If Activity -->

            <!-- While Activity -->

      </sapt:ToolboxCategory>
</sapt:ToolboxControl>
```

Quellcode 38 Fragment ein XAML Datei für die Darstellung eines Workflow Designers

Im Anschluss daran ist zu erkennen, dass eine Sequence Activity hinzugefügt wird. Auf dieser Art und Weise können eigene Bereich mit den verfügbaren bzw. erlaubten Activites aufgebaut werden.

Die Beispiele:

- WorkflowDesigner
- Windows Workflow Foundation 4.0 Explorer

im Anhang A demonstrieren beide die Verwendung des Workflow Designers.

4.11 Zusammenfassung

In diesem Kapitel wurde anhand von konkreten Beispielen erklärt, wie einem die Windows Workflow Foundation 4.0 beim Abbilden von Geschäftsprozessen unterstützen kann.

Die dazu notwendigen softwaretechnischen Voraussetzungen (wie z. B. Anonyme Typen, Lambda Expresssions, Typinferenz, vereinfachte Objektinstanziierung, Generics und Ereignisse) sind in Anhang B erörtert.

Zunächst wurde die Anwendung von Activities der BAL und Custom Activies erklärt.

Darauf folgend werden Workflows genauer besprochen. Wesentlich dabei ist der Austausch von Argumenten zwischen der Host Anwendung und dem Workflow.

Auf die Unterschiede zwischen WorkflowInvoker und WorkflowApplikation wurde danach eingegangen. Aufgrund der geringen Bedeutung für unser konkretes Einsatzgebiet, wurde auf WorkflowServiceHost und AppFabric nicht näher eingegangen.

Zuletzt wurde demonstriert, wie die Workflow Run Time mittels Extensions erweitert werden kann und wie ein ErrorHandling realisiert werden kann.

Alle vorgestellten Beispiele sind im Anhang A nochmals kurz beschrieben und voll funktionsfähig.

5 Ergebnisse und Ausblick

„Die Zukunft erkennt man nicht, man schafft sie.“

Stanisław Brzozowski (1878-1911)
polnischer Philosoph

5.1 Ergebnisse der vorliegenden Arbeit

Unter Punkt 1.2 sind die Zielsetzungen dieser Diplomarbeit festgehalten. Die zentrale Aufgabe ist die Beantwortung der Frage, ob die Windows Workflow Foundation für die Abbildung von Geschäftsprozessen in einer .NET Anwendung verwendet werden kann. Nachdem die Windows Workflow Foundation 4.0 sämtliche theoretischen und praktischen Anforderungen erfüllt, kann diese Frage mit einem klaren „*Ja*“ beantwortet werden. Diese Antwort stellt bereits das erste Ergebnis dar.

Auch die geforderten Beispiele wurden erstellt und finden sich im Anhang A wieder. Diese sind voll funktionsfähig.

Im Rahmen dieser Diplomarbeit war es leider nicht möglich, die Windows Workflow Foundation voll umfänglich zu beleuchten. In dieser Arbeit sind jene Punkte hervorgehoben, welche für das Erstellen von weiteren Prototypen unabdingbar sind.

5.2 Worin besteht mein eigener Anteil?

Für das Erstellen von Kapitel 2 war eine intensive Beschäftigung mit einschlägiger Literatur zum Thema Workflow Management notwendig. Begriffe aus diesem Umfeld waren mir bis zu diesem Zeitpunkt nur sehr oberflächlich bzw. gar nicht vertraut. Das Kapitel beinhaltet fundamentale Definitionen.

In Kapitel 3 wurden die Erkenntnisse von Kapitel 2 nun auf die Windows Workflow Foundation 4.0 umgelegt. Dazu war es notwendig, verfügbare Literatur zum Thema Windows Workflow Foundation 4.0 zu beschaffen und zu studieren. Erschwerend kam dabei dazu, dass lange Zeit keine Literatur verfügbar war. Bei der Windows Workflow

Foundation 4.0 handelt es sich um eine sehr aktuelle Technologie. Als erste Informationsquelle diente deshalb das Internet und im speziellen die von Microsoft betriebenen Foren.

Kapitel 4 beschäftigt sich mit der praktischen Umsetzung des in den vorhergegangen beiden Kapiteln aufgebauten Wissens. Dafür war es für mich zunächst notwendig, grundlegende Konzepte und Sprachelemente von C# auszuarbeiten. Im Anschluss daran wurden jene Themen erarbeitet und mit konkreten Beispielen hinterlegt, welche aus meiner Sicht als wesentlich anzusehen sind. Manche Beispiele und Textpassagen mussten dafür mehrfach überarbeitet werden, da im Zuge der unterschiedlichen Versionen (ß1, ß2 und RC) von Visual Studio 2010 und dem damit verbundenen .NET Framework 4.0 relevante Namespaces und Klassen von Microsoft überarbeitet wurden. Die von mir erarbeiteten Beispiele sind bewusst kompakt gehalten und behandeln stets einen konkreten Problemfall.

5.3 Sollte die Thematik fortgesetzt werden?

Im Zuge dieser Diplomarbeit konnten keine Faktoren ermittelt werden, welche gegen die Verwendung der Windows Workflow Foundation 4.0 für die Abbildung von Geschäftsprozessen sprechen. Es wird somit empfohlen, sich weiterhin mit der Thematik Windows Workflow Foundation 4.0 zu befassen!

Aus technischer Sicht ist anzumerken, dass bis jetzt leider nur sehr wenig Literatur verfügbar ist, welche sich explizit dem Thema Windows Workflow Foundation 4.0 widmet. In absehbarer Zeit sollen folgende Neuerscheinungen publiziert werden:

- *Pro WF: Windows Workflow in .NET 4.0*
  Bruce Bukovics
  ISBN: 978-1-4302-2721-2
  850 Seiten
  Apress, geplant für Juni 2010

- *Windows Workflow Foundation 4.0 Unleashed*
  Matt Winkler
  ISBN: 978-0672330681
  700 Seiten
  Sams, geplant für Oktober 2010

Das Studium dieser beiden Werke ist unbedingt anzuraten.

Bevor mit der Entwicklung entsprechender Prototypen begonnen werden kann, sollte durch das Produktmanagement ein Grundstock an sinnvollen Custom Activities und Standard Workflows ermittelt werden.

Natürlich darf auch auf die Anwender der Software nicht vergessen werden. Ausgewählte Key User mit entsprechender Erfahrung könnten evtl. bereits in einem früheren Stadium auf diese neue Technologie aufmerksam gemacht und damit sensibilisiert werden. Es ist fraglich, ob der breiten Masse von Anwendern das Bearbeiten von Workflows zugemutet werden kann/soll. Dies stellt zusätzlich eine Herausforderung an unterstützende Unterlagen, wie z. B. Handbücher und Schulungsmaterial, dar.

Anhang A: praktische Beispiele

Auf der CD sind im Verzeichnis QuellCode Beispiele zu finden. Es handelt sich dabei stets um voll funktionsfähige Beispiele für einen bestimmten Themenkreis. Der Quell Code ist mit ausreichend Kommentaren versehen. Mache Beispiele benötigen eine SQL Datenbank. Die Skripte zum Anlegen der benötigten Tabellen sind jeweils im Verzeichnis SQL zu finden.

Kapitel	Code Beispiel	Beschreibung
4.2.1	Assign Activity in Code	Die Verwendung der Assign Activity im Zuge eines minimalen Workflows, welcher in Code realisiert wurde, wird demonstriert.
4.2.1	Assign Activity in XAML	Die Verwendung der Assign Activity im Zuge eines minimalen Workflows, welcher in XAML realisiert wurde, wird demonstriert.
4.2.3	If Activity in Code	Die Verwendung der If Activity im Zuge eines minimalen Workflows, welcher in Code realisiert wurde, wird demonstriert.
4.2.3	If Activity in XAML	Die Verwendung der If Activity im Zuge eines minimalen Workflows, welcher in XAML realisiert wurde, wird demonstriert.
4.2.5	Flowchart Activity in XAML	Demonstriert die Verwendung von Flowchart, FlowDecision, FlowSwitch und Parallel in einem Workflow.
4.2.6	InvokeMethod	Die Verwendung der Activity InvokeMethod wird in zahlreichen Beispielen demonstriert. Die Methoden einer Klasse werden dabei auf unterschiedlichste Art und Weise verwendet – auch asynchron.
4.3	Custom Activity in Code	Demonstriert das Erstellen einer Custom Activity in C# Code. Zusätzlich wird auch eine Oberfläche dazu erstellt.
4.3	Custom Activity in XAML	Demonstriert das Erstellen einer Custom Activity in XAML.
4.4.3	LoadWfFromXAML	Dieses Beispiel zeigt die Verwendung der Methode ActivityXamlServices.Load, um einen Workflow, welche in Form eines

		MemoryStream vorliegt, zur Ausführung zu bringen.
4.5.1	WorkflowInvoker	Startet einen generischen Workflow mittels WorkflowInvoker und demonstriert den Zugriff auf den Return Parameter.
4.5.1, 4.5.2	WorkflowInstances	Liefert einen Vergleich von WorkflowInvoker und WorkflowApplication. Derselbe Workflow wird in diesem Beispiel auf unterschiedliche Arten zur Ausführung gebracht. Es ist zu erkennen, dass WorkflowApplication wesentlich mehr Möglichkeit zur Steuerung des Workflow bietet als WorkflowInvoker.
4.6	SimpleExtension	Die Übergabe eines Connection Strings von der Host Anwendung an Custom Activities wird mittels einer einfachen Extension realisiert. Für dieses Beispiel wird eine SQL Datenbank benötigt. Ein Script zum Erzeugen der Tabelle ist im Ordner SQL zu finden. Der Connection String wird aus der Konfigurationsdatei der Anwendung ausgelesen. Vor der Verwendung ist darauf zu achten, dass dieser dort korrekt eingetragen ist. Auf Fehlerprüfungen wurde bewusst verzichtet, um den Programmcode möglichst kompakt zu halten. Ein weiteres interessantes Detail ist der Zugriff auf die Daten mittels LINQ-To-SQL Klasse. Diese übernimmt das Mapping der Felder und stellt automatisch Methoden zum Lesen und Schreiben der Datensätze zur Verfügung.
4.7	Persistenz	Diese Solution besteht aus drei Projekten und demonstriert das Persistieren eines Workflows. Zu beachten ist, dass eine SQL Datenbank benötigt wird. Die notwendigen Skripte zum Anlegen der Tabellen sind im Verzeichnis SQL zu finden. Danach ist in den Projekten Phase 1 und Phase 2 der Connection String entsprechend anzupassen! Phase 1 erstellt einen neuen Workflow und

		persistiert diesen in der SQL Datenbank. Projekt Phase 2 lädt einen persistierten Workflow aus der Datenbank und setzt diesen fort. Im Projekt ActivityLibrary wird der gemeinsam genutzte Workflow definiert.
4.8.1	EventTracking for Windows (ETW)	Dieses Beispiel demonstriert die Aufzeichnung von Überwachungsinformationen im Windows Ereignisprotokoll unter Verwendung eines EtwTrackingParticipant Objektes.
4.8.2	SqlTrackingParticipant	Es wird ein eigener TrackingParticipant erstellt, welche die gesammelten Informationen in einer SQL Datenbank speichert. Als Workflow dient ein Spiel – Zahlenraten: es muss der Wert einer Zufallszahl erraten werden. Somit wurde ein etwas länger andauernder Workflow geschaffen. Im Ordern SQL befinden sich die notwendigen SQL Skripte zum Anlegen der Tabellen.
4.9	ExceptionHandling	Im Zuge eines fiktiven Geschäftsprozesses für einen Bestellvorgang wird durch fehlende Artikel eine Exception ausgelöst und ausgewertet.
4.10	WorkflowDesigner	Diese Anwendung zeigt auf einfachste Art und Weise, wie ein Designer für Workflows in eine eigene WPF Anwendung integriert werden kann.
4.10	Windows Workflow Foundation 4.0 Explorer	Dabei handelt es sich um eine umfangreichere Applikation, welche den HostedDesigner im Mittelpunkt hat. Workflows können schrittweise abgearbeitet werden. Der momentane Zustand des Workflows und der Activities wird über zahlreiche Tracking- und Loginformationen dargestellt. Eigene Custom Activities sind im Designer zur Verfügung gestellt. Die erzeugten Workflows können gespeichert und erneut geladen werden.

Anhang B: Softwaretechnische Grundlagen & Voraussetzungen

Ein Verständnis der in den folgenden Unterpunkten enthaltenen Sprachelemente ist Voraussetzung für die in Anhang A angeführten Beispiele.

Anonyme Typen

Bei der Umsetzung von Workflows in Code werden häufig anonyme Instanzen von Klassen verwendet. Klassen wie Sequence, WriteLine und If im folgenden Beispiel Quellcode 39 werden zwar instanziiert, jedoch niemals benannt.

```
new Sequence ()
{
   Activities =
   {
      New WriteLine (){…},
      new If (){…},
      new Assign (){…}
      new Sequence ()
      {
         new If (){…},
         …
      }
   }
}
```

Quellcode 39 Pseudocode für anonyme Klasseninstanzen

Dieser Ansatz ist einer Technik mit dem Namen „*Functional Construction*“ [Col10 S. 26] sehr ähnlich.

Functional Construction wird häufig beim Erstellen von XML Bäumen angewandt. [Mic1002]

Wichtig dabei ist die Tatsache, dass Code nicht an eine Variabel zugewiesen werden kann. Nur ein Zeiger auf Code kann einer Variablen zugewiesen werden. [Pia07 S. 24]

Da anonyme Methoden häufig ihre Verwendung bei der Definition von Workflows finden, erschien mir dieser kurze Exkurs angebracht. Das Verständnis dieser Thematik ist für die folgenden Kapitel Voraussetzung.

Lambda Expression

Der folgende Ausdruck stellt eine Lambda Expression[75] dar:

```
env => counter.Get(env) + 1
```

Quellcode 40 Lambda Expression

Bei einem Lambda Ausdruck handelt es sich um eine anonyme Methode, die Ausdrücke und Anweisungen enthalten kann und für die Erstellung von Delegates verwendet werden kann.

Lambda Ausdrücke verwenden den => Operator. Links davon werden die Eingabeparameter angegeben, rechts davon der Ausdruck oder ein Anweisungsblock. Der Anweisungsblock eines Lambda Ausdruckes benötigt generell eine geschweifte Klammer wie jeder andere Anweisungsblock auch und kann beliebig viele Argumente enthalten. Häufig (wie auch in diesem Fall) anzutreffen sind Lambda Ausdrücke, deren einzige Anweisung ein `return` ist. In einem solchen Fall dürfen die `return` Anweisung und die geschweiften Klammern {} weggelassen werden. Der Lambda Ausdruck vom Beispiel Quellcode 40 besitzt nur einen Parameter. Sind mehrere Parameter vorhanden, so müssen diese durch Komma getrennt werden und der gesamte Parameter Ausdruck muss mit runden Klammern umschlossen sein. Bei einem Lambda Ausdruck mit einer leeren Parameterliste müssen jedoch die runden Klammern angegeben werden. Liegt nur ein Parameter vor, so können die runden Klammern entfallen. Im Folgenden sind Beispiele für gültige Lambda Ausdrucke angeführt:

```
(double a, double b) => {return a + b}
(a, b) => a + b
a => a + a
() => a
```

Quellcode 41 Beispiele für gültige Lambda Ausdrücke

Ein Lambda Ausdruck, der lediglich eine `return` Anweisung enthält, wird als „*Ausdrucksrumpf*“ bezeichnet. [Alb08 S. 129 ff.], [Pia07 S. 31]

---

[75] Englisch für: Ausdruck

Der Datentyp der Rückgabe eines Lambda Ausdruckes kann sich vom Datentyp des Parameters unterscheiden. Liegt ein solcher Lambda Ausdruck vor, wird von einer Projektion gesprochen. Die folgende Anweisung zeigt eine solche Projektion. [Alb08 S. 129 ff.], [Pia07 S. 34]

```
(zeichenkette) => zeichenkette.Length
```

Quellcode 42 Beispiel für eine Projektion in Form eines Lambda Ausdruckes

Dabei wird eine Zeichenfolge übergeben und deren Länge geprüft. Der Rückgabewert ist vom Typ Integer.

Ein Prädikat hingegen liefert einen booleschen Wert als Ergebnis einer Operation. Das folgende Beispiel verdeutlicht dies:

```
(betrag) => betrag > 100
```

Quellcode 43 Beispiel für ein Prädikat in Form eines Lambda Ausdruckes

Es gibt keine offizielle Bezeichnung für den Lambda Operator =>. Häufig wird als Bezeichnung „*such that*“ im Falle eines Prädikates und „*becomes*“ im Falle einer Projektion angewandt. Andere Quellen verwenden generell nur „*goes to*“. [Pia07 S. 32]

Auf Lambda Ausdrücke wurde deshalb an dieser Stelle und in dieser Tiefe eingegangen, da sie in den folgenden Beispielen häufig verwendet werden. Vor allem der Zugriff auf Argumente und Variablen wird oftmals mittels Lambda Ausdrücken realisiert, da der Quellcode dadurch deutlich kürzer wird. Auch in Kombination mit Events und Delegates finden Lambda Expressions ihre Verwendung.

Typinferenz

Bei der Deklaration einer Variable muss deren Datentype mit angegeben werden. Ab C# 3 gibt es dafür eine kleine Erleichterung – genannt Typinferenz. Diese gestattet es, Variablen mit dem Schlüsselwort `var` zu deklarieren, ohne dabei den Datentyp angeben zu müssen. [Küh091 S. 490 ff.]

```
var x = 5;          // implizit vom Typ Integer
var s = "Hallo";    // implizit vom Typ String
```

Quellcode 44 Typinferenz

Das Schlüsselwort `var` bewirkt, dass der Compiler automatisch den am besten passenden Datentyp, aufgrund des Ausdruckes rechts vom = Zeichen für die Variable, wählt. Ab dann wird die Variable vom Compiler behandelt, als wäre sie explizit deklariert worden. [Alb08 S. 48] Damit ist auch die Typensicherheit garantiert.

Einschränkungen der Typinferenz laut Kühnel [Küh091 S. 490 ff.]:

- Typinferenz kann nur auf lokale Variablen angewandt werden.
- Die Initialisierung muss bei der Deklaration erfolgen.
- Null (im Sinne von Nothing) darf nicht im Zuge der Initialisierung zugewiesen werden.
- Rückgabeparameter von Methoden dürfen nicht mit var deklariert werden.
- Methodenparameter dürfen nicht mit var deklariert werden.

Typinferenz ist ein wichtiges Sprachelement in C# und anderen objektorientierten Programmiersprachen. Durch einen sinnvollen Einsatz von Typinferenz kann Code wesentlich kürzer und trotzdem typensicher gestaltet werden.

Vereinfachte Instanziierung von Objekten

Die herkömmliche Methode, um ein Objekt zu instanziieren, welches die Schnittstelle IDictionary implementiert hat, schaut wie folgt aus.

```
IDictionary<string, object> arguments = new Dictionary<string, object>();
arguments.Add("Caption", "Hallo");
arguments.Add("Text", "Windows Workflow Foundation 4.0");
arguments.Add("Buttons", MessageBoxButtons.OK);
arguments.Add("Icon", MessageBoxIcon.Information);
```

Quellcode 45 Variante 1 um ein Objekt zu erzeugen, welches IDictionary implementiert und hinzufügen von Kombinationen aus Schüssel und Wert.

Es wird zuerst ein entsprechendes Objekt instanziiert und anschließend werden unter Verwendung der Add Methode mehrere Key-Werte Kombinationen hinzugefügt.

Ab C# 3.0 ist eine verkürzte Schreibweise erlaubt. Die Startwerte können im Zuge der Erstellung des Objektes in geschweiften Klammern mit angegeben werden. Einzelne Eigenschaften sind mittels Komma zu trennen. Bei dieser Notation kann zusätzlich auch noch auf die leeren runden Klammern verzichtet werden. [Küh091 S. 151 ff.]

```
IDictionary<string, object> arguments =  new Dictionary<string, object>
{
      { "Caption", "Hallo" },
      { "Text", "Windows Workflow Foundation 4.0" },
      { "Buttons", MessageBoxButtons.OK },
      { "Icon", MessageBoxIcon.Information }
};
```

Quellcode 46 Variante 2 um ein Objekt zu erzeugen, welches IDictionary implementiert und hinzufügen von Kombinationen aus Schüssel und Wert.

Beide Varianten liefern dasselbe Ergebnis. Bei den Beispielen im Anhang sind beide Schreibweisen in Verwendung. Bei wenigen Argumenten wird Variante 2 wahrscheinlich effektiver sein.

Generics - generische Datentypen

Generics unterstützen dabei, Programmcode so zu erstellen, dass dieser mit verschiedenen Datentypen verwendet werden kann. Bei Generics wird dies in Form eines Templates erreicht. Dieses Template repräsentiert einen Platzhalter und wird in der Form <T> dargestellt. Generics sind trotzdem typensicher. [Alb08 S. 108 ff.]

Bei einer generischen Methode werden bereits in der Signatur der Methode generische Parameter deklariert. Das folgende Beispiel Quellcode 47 demonstriert eine generische Methode und zeigt auch deren Verwendung.

```
/// <summary>
/// Vertauschen der Werte von zwei Parametern
/// </summary>
/// <typeparam name="T">Typ von a und b</typeparam>
/// <param name="a">Parameter a erhält den Wert von Parameter b</param>
/// <param name="b">Parameter b erhält den Wert von Parameter a</param>
static void Vertausche<T>(ref T a, ref T b)
{
      T temp = a;
      a = b;
      b = temp;
}

// …

// Verwendung der Funktion Vertausche mit Type int
int zahl1 = 5;
int Zahl2 = 6;
Vertausche<int>(ref zahl1, ref Zahl2);
Vertausche(ref zahl1, ref Zahl2);
// Verwendung der Funktion Vertausche mit Type string
string string1 = "Hallo";
string string2 = "Workflow";
Vertausche<string>(ref string1, ref string2);
Vertausche(ref string1, ref string2);
```

Quellcode 47 generische Methode und deren Verwendung

In diesem Beispiel wurde die Funktion Vertausche einmal mit dazu verwendet, zwei Integer Zahlen zu vertauschen, im zweiten Fall werden zwei Strings vertauscht. Der Code kann demnach für mehrere Datentypen verwendet werden. Wie man aus diesem Beispiel auch erkennt, kann die Methode auch ohne <T> aufgerufen werden. Dem Compiler ist der Datentyp der Variablen ohnehin bekannt.

Bei den vorhandenen Klassen des .NET Framework werden Generics zum Teil sehr intensiv eingesetzt.

Ereignisse vom Type Action<T>

Die Klasse WorkflowApplication bietet u. a. ein Ereignis Unloaded an.

```
public Action<WorkflowApplicationEventArgs> Unloaded { get; set; }
```

Quellcode 48 Unloaded Event der Klasse WorkflowApplication

Bei Action<T> handelt es sich um eine generische Methode, welcher ein Parameter übergeben wird. Sie liefert keinen Wert zurück. [Mic1007]

Es gibt in C# zwei unterschiedliche Möglichkeiten, sich darauf zu verbinden:

- mittels Delegate
- mittels Lambda Expression

Im Beispiel Quellcode 49 wird ein Delegate verwendet.

Bei einem Delegate handelt es sich um einen „*Funktionszeiger*“ [Küh091 S. 289].

```
instance.Unloaded = delegate(WorkflowApplicationEventArgs e)
{
//…
};
```

Quellcode 49 Binden an ein Event mittels Delegate

Oder anders ausgedrückt handelt es sich bei einem Delegate um ein Objekt, das den Zeiger auf eine Objektmethode beschreibt. Im genannten Beispiel wird dies nun mit einer anonymen Methode kombiniert.

Dasselbe Ergebnis kann auch mittels eines Lambda Ausdruckes erreicht werden. Dies wird in Beispiel Quellcode 50 veranschaulicht.

```
instance.Unloaded = (WorkflowApplicationEventArgs e) =>
{
//…
};
```

Quellcode 50 Binden an ein Event mittels eines Lambda Ausdruckes

Da C# hier zwei völlig unterschiedliche Methoden für eine Problemlösung anbietet, erschien es mir wichtig darauf hinzuweisen, dass beide Varianten dasselbe Ergebnis bewirken.

Literaturverzeichnis

[Alb08]. **Albabari, Joseph und Albabari, Ben. 2008.** *C# 3.0 Pocket Reference.* Sebastopol : O'Reilly, 2008. 978-0-596-51922-3.

[Bei09]. **Beijer de, Maurice. 2009.** Online Workflow 4 presentation. *The Problem Solver.* [Online] 1. Dezember 2009. [Zitat vom: 16. April 2010.] Dokument: WF4_Online.pptx. http://msmvps.com/blogs/theproblemsolver/archive/2009/12/01/online-workflow-4-presentation.aspx.

[Cha09]. **Chappell, David. 2009.** *The Workflow Way - Understandung Windows Workflow Foundation.* Redmond : Microsoft Corporation, 2009.

[Col10]. **Collins, Mark J. 2010.** *Beginning WF: Windows Workflow in .NET 4.0.* New York : Apress, Inc., 2010. 978-1-4302-2485-3.

[DAT09]. **DATACOM Buchverlag GmbH. 2009.** Persistenz. *IT Wissen.* [Online] DATACOM Buchverlag GmbH, 26. November 2009. [Zitat vom: 30. April 2010.] http://www.itwissen.info/definition/lexikon/Persistenz-persistence.html.

[Dob09]. **Dobric, Damir. 2009.** Neues Flussbett. *dotnetpro.* 2009, 12.

[Dob10]. **—. 2010.** Im Schwebezustand. *dotnetpro.* 2010, 1.

[Dob101]. **—. 2010.** Weltweite Workflows. *dotnetpro.* 2010, 2.

[Dor05]. **Dorfer, Roland. 2005.** *Diplomarbeit: Migration von COM-Komponenten in .NET Assemblies.* Salzburg : Fachhochschule Salzburg, 2005.

[Gen07]. **Gennick, Jonathan. 2007.** *SQL kurz & gut.* 2. Auflage. Köln : O'Reilly, 2007. 978-3-89721-522-1.

[Heu95]. **Heuckeroth, Okka. 1995.** *Workflow-Managment: Möglichkeiten und Granzen.* Hamburg : Diplomarbeiten Agentur, 1995. 978-3-8386-0293-6.

[Hol95]. **Hollingsworth, David. 1995.** WfMC - Workflow Management Coalition. *The Workflow Reference Model.* [Online] 19. Jänner 1995. [Zitat vom: 3. April 2010.] http://www.wfmc.org/Download-document/TC003v11-WfMC-Workload-Reference-Model.html.

[Jac10]. **Jacobs, Ron. 2010.** *AppFabric and Window Workflow 4.* [Präsentation] Redmond : Microsoft, 2010. Bestandteil von VS2010TrainingKit.

[Küh091]. **Kühnel, Andreas. 2009.** *Visual C# 2008.* Bonn : Galileo Press, 2009. 978-3-8362-1172-7.

[Lam10]. **Lamb, Jim. 2010.** How to Create a Custom Workflow Activity for TFS Build 2010 RC. *msdn.* [Online] Microsoft, 12. Februar 2010. [Zitat vom: 30. März 2010.] http://blogs.msdn.com/jimlamb/archive/2009/11/18/how-to-create-a-custom-workflow-activity-for-tfs-build-2010.aspx.

[Mac10]. **Mackey, Alex. 2010.** *Introducing .NET 4.0: with Visual Studio 2010.* New York : Apress, Inc., 2010. 978-1-4302-2455-6.

[McC04]. **McConnell, Steve. 2004.** *Code Complete.* Redmond : Microsoft Press, 2004. 978-3-86063-593-3.

[McL09]. **McLoughlin, John. 2009.** A lap around WF 4.0. *The .NET Developer Network.* [Online] 10. Februar 2009. [Zitat vom: 16. April 2010.] http://www.dotnetdevnet.com/downloads/Meetings/2009/February/ALapAroundWF.pdf.

[Mic10]. **Microsoft Corporation. 2010.** WorkflowApplication-Klasse. *MSDN.* [Online] Microsoft Corporation, 2010. [Zitat vom: 9. April 2010.] http://msdn.microsoft.com/de-de/library/system.activities.workflowapplication(VS.100).aspx.

[Mic1001]. **—. 2010.** Installing the .NET Framework. *MSDN.* [Online] Microsoft Corporation, 2010. [Zitat vom: 09. April 2010.] http://msdn.microsoft.com/de-de/library/5a4x27ek(en-us,VS.100).aspx.

[Mic1002]. **—. 2010.** Functional Construction (LINQ to XML). *MSDN*. [Online] Microsoft Corporation, 2010. [Zitat vom: 11. April 2010.] http://msdn.microsoft.com/en-us/library/bb387019.aspx.

[Mic1003]. **—. 2010.** .NET Framework 4 and Extensions. *MSDN*. [Online] 2010. [Zitat vom: 11. April 2010.] http://download.microsoft.com/download/E/6/A/E6A8A715-7695-493C-8CFA-8E0C23A4BE1D/098-115952-NETFX4-Poster.pdf.

[Mic1004]. **—. 2010.** ModelItem Class. *MSDN*. [Online] 2010. [Zitat vom: 13. April 2010.] http://msdn.microsoft.com/de-de/library/bb189392(en-us).aspx.

[Mic1005]. **—. 2010.** ArgumentToExpressionConverter Class. *MSDN*. [Online] 2010. [Zitat vom: 13. April 2010.] http://msdn.microsoft.com/en-us/library/system.activities.presentation.converters.argumenttoexpressionconverter(VS.100).aspx.

[Mic1006]. **—. 2010.** Visual Studio 2010 and .NET Framework 4 Training Kit. *Microsoft Download Center*. [Online] 12. April 2010. [Zitat vom: 14. April 2010.] \Labs\IntroToWF\Lab.docx. http://www.microsoft.com/downloads/details.aspx?FamilyID=752cb725-969b-4732-a383-ed5740f02e93&displayLang=en.

[Mic1007]. **—. 2010.** Action<T> Delegate. *MSDN*. [Online] 2010. [Zitat vom: 14. April 2010.] http://msdn.microsoft.com/en-us/library/018hxwa8.aspx.

[Mic1008]. **—. 2010.** ActivityExecutionStatus Enumeration. *MSDN*. [Online] Microsoft Corporation, 2010. [Zitat vom: 16. April 2010.] http://msdn.microsoft.com/en-us/library/system.workflow.componentmodel.activityexecutionstatus.aspx.

[Mic1009]. **—. 2010.** Windows Workflow Foundation. *MSDN*. [Online] Microsoft Corporation, 3. März 2010. [Zitat vom: 16. April 2010.] http://msdn.microsoft.com/de-de/library/dd489441.aspx.

[Mic1010]. **—. 2010.** Nachverfolgung und Ablaufverfolgung für Workflows. *MSDN.* [Online] Microsoft Corporation, 2010. [Zitat vom: 25. April 2010.] http://msdn.microsoft.com/de-de/library/ee513992.aspx.

[Mic1011]. **—. 2010.** PersistableIdleAction Enumeration. *MSDN.* [Online] Microsoft Corporation, 2010. [Zitat vom: 30. April 2010.] http://msdn.microsoft.com/en-us/library/system.activities.persistableidleaction.aspx.

[Mic1012]. **—. 2010.** TrackingRecord Class. *MSDN.* [Online] Mircosoft Corporation, 2010. [Zitat vom: 1. Mai 2010.] http://msdn.microsoft.com/en-us/library/system.activities.tracking.trackingrecord.aspx.

[Mil09]. **Milner, Matt. 2009.** *A Developer's Introduction to Windows Workflow Foundation (WF4).* Redmond : Microsoft, 2009.

[MSM10]. **MSMVPS. 2010.** Changing the Icon on a custom activity designer . *The Problem Solver.* [Online] 25. Jänner 2010. [Zitat vom: 14. April 2010.] http://msmvps.com/blogs/theproblemsolver/archive/2010/01/25/changing-the-icon-on-a-custom-activity-designer.aspx.

[Pia07]. **Pialorsi, Paolo und Russo, Marco. 2007.** *Introducing Microsoft LINQ.* Redmond : Microsoft Press, 2007. 978-0-7356-2391-0.

[Ric04]. **Richter-von Hagen, Cornelia und Stucky, Wolffried. 2004.** *Business-Process- und Workflow-Management: Prozessverbesserung durch Prozess-Management.* Wiesbaden : Vieweg+Teubner, 2004. 978-3519004912.

[Rus06]. **Russel, Nick, et al. 2006.** Workflow Control-Flow Patterns: A Revised View. *Workflow Patterns.* [Online] 2006. [Zitat vom: 12. April 2010.] http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf.

[Sch08]. **Schattauer, Ronny. 2008.** *Human-Workflows.* Saarbrücken : VDM Verlag Dr. Müller Aktiengesellschaft & Co. KG, 2008. 978-3-639-07628-8.

[Scr07]. **Scribner, Kenn. 2007.** *Microsoft Windows Workflow Foundation - Schritt für Schritt.* Unterschleißheim : Microsoft Press Deutschland, 2007. 978-3866455030.

[Sur10]. **Surhone, Lambert M., Timpledon, Miriam T. und Marseken, Susan F. 2010.** *Process-Driven Application.* Mauritius : Betascript Publishing, 2010. 978-613-0-48669-3.

[Wik10]. **Wikipedia. 2010.** .NET. *Wikipedia.* [Online] 31. März 2010. [Zitat vom: 04. April 2010.] http://de.wikipedia.org/wiki/.NET.

[Wik102]. **—. 2010.** Projektmanagement. *Wikipedia.* [Online] 2. April 2010. [Zitat vom: 3. April 2010.] http://de.wikipedia.org/wiki/Projektmanagement.

[Wor99]. **Workflow Management Coalition. 1999.** WfMC - Workflow Management Coalition. *Terminology & Glossary.* [Online] 11. Mai 1999. [Zitat vom: 30. März 2010.] http://www.wfmc.org/Download-document/TC-1011-Term-Glossary-V3.html.

Stichwortverzeichnis

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Ing. Bernhard Mayr, 21986

Mondsee, Juni 2010