Bachelorarbeit

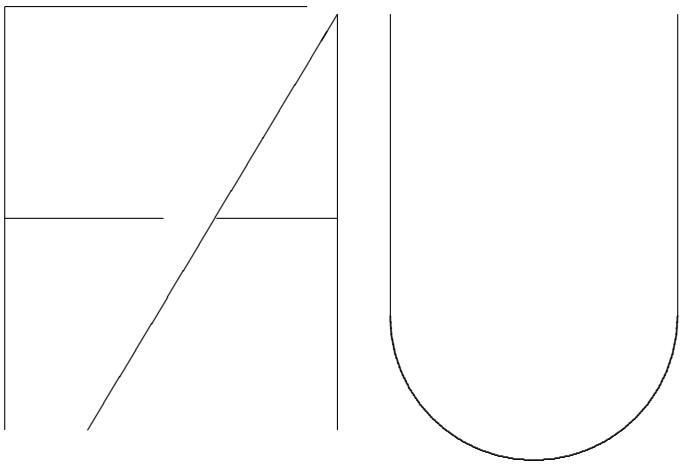
Ein Windows-Druckertreiber zum Einbringen von Dokumenten aus beliebigen Drittanwendungen als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten

Konstantin Tsysin

Lehrstuhl für Informatik 6
(Datenmanagement)

Department Informatik
Technische Fakultät

Friedrich Alexander-
Universität
Erlangen-Nürnberg



Ein Windows-Druckertreiber zum Einbringen von Dokumenten aus beliebigen Drittanwendungen als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten

Bachelorarbeit im Fach Informatik

vorgelegt von

Konstantin Tsysin

geb. 04.01.1988 in Ribnita

angefertigt am

Department Informatik
Lehrstuhl für Informatik 6 (Datenmanagement)
Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreuer: Univ.-Prof. Dr.-Ing. habil. Richard Lenz
Dipl.-Inf. Christoph P. Neumann

Beginn der Arbeit: 15.07.2011
Abgabe der Arbeit: 15.12.2011

Erklärung zur Selbständigkeit

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass diese Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Der Universität Erlangen-Nürnberg, vertreten durch den Lehrstuhl für Informatik 6 (Datenmanagement), wird für Zwecke der Forschung und Lehre ein einfaches, kostenloses, zeitlich und örtlich unbeschränktes Nutzungsrecht an den Arbeitsergebnissen der Bachelorarbeit einschließlich etwaiger Schutzrechte und Urheberrechte eingeräumt.

Erlangen, den 15.12.2011

_____

(Konstantin Tsysin)

Kurzfassung

Ein Windows-Druckertreiber zum Einbringen von Dokumenten aus beliebigen Drittanwendungen als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten

Zur Unterstützung institutionsübergreifender Abläufe im Gesundheitswesen im Rahmen des Projekts ProMed (Prozessunterstützung von adaptiv-evolutionären Informationssystemen in der Medizin) wird das Konzept aktiver Dokumente eingesetzt. Diese α-Docs stellen ein einheitliches, systemunabhängiges Format dar, die sowohl die Nutzdaten beinhalten als auch Synchronisations- und Koordinationsmechanismen bieten. Dabei stellt sich jedoch die Frage nach der Integration von Daten bestehender, oft inkompatibler Anwendungssysteme.

Dazu wird im Rahmen dieser Arbeit PrintPut vorgestellt, ein speziell konfigurierter Windows-Druckertreiber, der beliebige druckbare Dokumentformate in das standardisierte und weit verbreitete PDF konvertiert. Dies senkt den Synchronisationsaufwand zwischen den bestehenden Systemen und wirkt sich vereinfachend und kostensenkend aus, da Lizenzen für andere Programme, Schulungen oder die Programmierung von Konvertierungsprogramme eingespart werden können, so dass die Effizienz der Patientenversorgung für alle Beteiligten steigt.

Abstract

Windows print driver to bring in documents from any third party applications as a component of process support based on active documents

The ProMed-Project (Process support of adaptive-evolutionary information systems in healthcare) is using active documents to achieve better support of cross-organizational workflows in the health sector. These ?-Docs are a uniform, system independent Format, which contains user data as well as mechanisms of synchronization and coordination. This begs the question of integration of data from already existing and often incompatible application systems.

Therefore in the context of this thesis PrintPut, a specially configured Windows printing driver, will be presented. It is able to convert any printable document formats to the standardized and widespread PDF. This way the synchronization effort between the existing application systems is reduced, the processes are simplified and costs can be cut, since expenses for application licenses, trainings or the programming of conversion tools can be saved. Thus PrintPut is a possible solution to raise the efficiency of patient care for all parties concerned.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ACL	Access Control List
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CDA	Clinical Document Architecture
CIL	Common Intermediate Language
DLL	Dynamic Link Library
EMF	Enhanced Metafile
GDI	Graphic Device Interface
HL7	Health Level 7
HP	Hewlett-Packard
LPT	Line Print Terminal
MSDN	Microsoft Developer Network
MSI	Microsoft Software Installation
NSIS	Nullsoft Scriptable Install System
PCL	Printer Command Language
PDF	Portable Document Format
PE	Portable Executable
PInvoke	Platform Invoke
RedMon	Redirection Port Monitor

RPC	Remote Procedure Call
SP	Service Pack
UAC	User Account Control
USB	Universal Serial Bus
XML	Extended Markup Language
XPS	XML Paper Specification

1 Einleitung

Im digitalen Zeitalter wird ein beträchtlicher Teil der Arbeit am Computer ausgeführt. Immer mehr Aufgaben und Informationen werden mit Hilfe des Rechners bearbeitet. Die Treiber der Digitalisierung sind eine zunehmend steigende Verarbeitungsgeschwindigkeit, sinkende Speicherkosten und die enormen Speicherkapazitäten moderner Datenträger, die nur einen Bruchteil des Platzbedarfs dieser Dokumente in Papierform brauchen, die leichte Transportierbarkeit und Verbreitung elektronischer Dokumente und moderne Dokumentenmanagementsysteme. Außerdem ermöglicht die Digitalisierung eine effizientere und flexiblere Bearbeitung, Indizierung und Suche in großen Datenbeständen.

Aus diesen Gründen wird auch im Gesundheitswesen immer mehr digitalisiert, die Zeiten der langen Suche nach einer Patientenakte sind vorbei. Die Digitalisierung bringt aber auch Probleme mit sich. Denn es gibt kein einheitlich verbindliches Format, in dem die Behandlungsdaten abgelegt werden. Innerhalb einer Institution ist das Problem noch überschaubar, da man die eigenen Formate kennt. Aber bei institutionsübergreifenden Abläufen kommt dieses Problem schnell zum Vorschein, denn jede Praxis hat eigene Systeme und darauf laufende Applikationen, welche untereinander sehr oft nicht kompatibel sind. Ist bei einer Patientenbehandlung ein Eingreifen von mehreren Ärzten erforderlich, so kann es dazu kommen, dass die Behandlung verkompliziert wird. Schlimmstenfalls zu Ungunsten des Patienten, wenn zum Beispiel eine erneute Röntgenaufnahme gemacht werden muss, weil man mit der digitalen Form der Röntgenaufnahme der anderen Einrichtung nichts anzufangen weiß. Deshalb werden Daten sehr oft auch analog, also in gedruckter Form mitverschickt, was einen erheblichen Organisationsaufwand mit sich bringt und den Vorteil der Digitalisierung zu Nichte macht.

Das Projekt „Prozessunterstützung von adaptiv-evolutionären Informationssystemen in der Medizin“ (ProMed) zur Unterstützung institutionsübergreifender Abläufe im Versorgungsnetz durch Umsetzung einer verteilten Prozessunterstützung, die auf aktiven Dokumenten basiert, beschreibt einen möglichen Lösungsansatz, wie man diesen Problemen entgegenwirken kann.

1.1 Motivation

Die Patientenbehandlung in der Medizin wird von einem dokumentenorientiertem Paradigma geleitet. Alle Behandlungen und medizinischen Vorgänge werden dokumentiert und in der Patientenakte hinterlegt. Diesem grundlegenden Vorgehen folgt auch der Kernbaustein von ProMed, α-Flow. Es basiert auf dem dokumenten- und prozessorientierten Paradigmenansatz in heterogenen Informationssystemen [NL09].

α-Flow versucht, auf Basis elektronischer Patientenakten die institutionsübergreifende Patientenbehandlung zu ermöglichen. Die Koordination im verteilten Prozess erfolgt dabei durch die Dokumente selbst, die als aktive Dokumente eigenständig in der Lage sind, auf Aktionen zu reagieren. So ein aktives Dokument, das α-Doc, bildet die atomare Einheit des Informationsaustausches. Es steht für eine Fallakte und beschreibt eine Patientenbehandlung, zum Beispiel bei einem Nasenbruch. Dies ermöglicht, ähnlich wie bei einer elektronischen Gesundheitskarte, eine patientenbezogene Arzneimitteldokumentation, den Abruf des persönlichen Gesundheitsstatus und der bisherigen Diagnoseergebnisse zur Vermeidung unnötiger Doppeluntersuchungen [JN03].

Durch die selbstkoordinierenden und selbstbeschreibenden α-Docs wird ein einheitliches Format eingeführt, das es ermöglicht, auf beliebigen Systemen zu laufen. Die einzige Voraussetzung ist eine Java-Ausführungsumgebung, die für praktisch alle Systeme entwickelt und kostenlos zur Verfügung gestellt wird. So kann das α-Doc für den Transport der Daten und zur Realisierung des Patientenbehandlungsprozesses in beliebigen Einrichtungen benutzt werden. Das α-Doc trägt nicht nur die Nutzdaten der medizinischen Verwendung, sondern stellt auch Synchronisationsmechanismen zwischen den Einrichtungen und andere Funktionen bereit. Einige Features sind noch in Entwicklung, zum Beispiel eine mögliche Authentifizierung zum Schutz der Dokumente, eine Versionsverwaltung und History, die es ermöglichen, jede Änderung an der Fallakte zu protokollieren und gegebenenfalls zu begründen. Dies schafft Übersicht, verringert mögliche Fehler, bringt Ordnung und schafft ein stärkeres Verantwortungsbewusstsein bei den Ärzten. Mit der Einführung des α-Docs bleibt aber noch die Frage offen, wie ein Arzt sein digitales Dokument einer beliebigen Drittanwendung in ein α-Doc integriert, so dass jede andere Person dieses Dokument öffnen und damit arbeiten kann? Dazu gibt es verschiedenste Lösungswege. α-PrintPut stellt eine mögliche Lösung dar, dieses Problem zu lösen.

1.2 Ziele der Arbeit / Einordnung von α-PrintPut

Im a-Doc werden die medizinischen Dokumente abgelegt. Zurzeit kann das Originaldokument, zum Beispiel im Word- oder Portable Document Format (PDF)-Format, eingefügt werden. Dies ist aber wie beschrieben problematisch, denn die Dateien alleine, ohne einen eingebauten Viewer und/oder einer in das α-Doc eingebauten Möglichkeit, nur bestimmte, von α-Doc darstellbare Dateiformate zu erlauben, sind von den Programmen und dem System selbst abhängig, auf denen sie später geöffnet werden sollen. So sind viele Dateiformate gar nicht dokumentiert und lassen sich nicht ohne weiteres transportieren. Allein schon eine andere Programmversion oder ein fremdes Betriebssystem mit fehlenden Schriftarten kann dazu führen, dass das Dokument nicht mehr dargestellt werden kann. Andererseits will man sich auch nicht auf bestimmte Dateiformate beschränken. Denn wenn man nur bestimmte Dateiformate zulassen würde, so müsste man alle anderen auf dieses Dateiformat konvertieren können. Solche Konverter sind aber bei der Fülle an möglichen Dateiformaten nicht realistisch. Bereits die originalgetreue Darstellung ist nicht möglich, da auch für weit verbreitete Formate oft keine Spezifikation vorhanden ist.

Ziel dieser Arbeit ist es, eine mögliche Lösung zum Einbringen von Dokumenten aus beliebigen Drittanwendungen als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten zu entwickeln und damit ein Ziel von ProMed auf technischer Ebene, nämlich die Einbindung von Standards, zu fördern, indem Dokumente aus beliebigen Drittanwendungen in ein α-Doc integriert werden. Dies wird durch einen speziell konfigurierten Windows-Druckertreiber und der Konvertierung in das standardisierte und weit verbreitete PDF-Format erreicht.

Anschließend werden die Vor- und Nachteile dieser Lösung abgewogen und ein möglicher Ausblick zur Verbesserung und Reife erläutert.

2 Methodik / Vorgehen

In diesem Kapitel werden das Vorgehen und die Aktivitäten beschrieben, die zur Realisierung von α-PrintPut beigetragen haben.

Anfangs wird ausgiebig im Internet und Bibliotheken recherchiert, welche Möglichkeiten der Formatkonvertierung in Frage kommen. Schnell wird jedoch klar, dass dieser Weg eine Sackgasse darstellt und es kristallisiert sich heraus, dass die Windows-Druckfunktion eine Möglichkeit anbietet, das Dateiformat-Problem in den Griff zu bekommen.

Hierfür werden zunächst die Grundlagen zur Realisierung von α-PrintPut vorgestellt. Dabei wird zuerst der Aufbau von α-Flow und der α-Docs noch einmal gründlich analysiert und das daraus resultierende Dateiformat-Problem erläutert. Anschließend wird die Druckerarchitektur von Windows und der Unterschied zwischen Rastergrafiken und Vektorgrafiken detailliert betrachtet.

Im Anschluss werden die Systemumgebung und der Einsatzort von α-PrintPut erläutert und es werden sich daraus ergebene Anforderungen und Einschränkungen von α-PrintPut deutlich. Diese werden in die Lösungsumsetzung einbezogen.

Im nächsten Kapitel werden mögliche Wege diskutiert, den Druckvorgang für den Lösungszweck zu verwenden. Es werden verwandte Lösungsansätze unter die Lupe genommen und verglichen. Dabei werden die unterschiedlichen Schwerpunkte und Probleme dieser jeweiligen Lösungen klar und daraus Schlussfolgerungen für α-PrintPut gezogen.

Auf diesem Resultat aufbauend werden die für α-PrintPut in Frage kommenden Seitenbeschreibungssprachen vorgestellt und mit einander verglichen. Dabei wird eine Präferenz einer Sprache deutlich, die für PrintPut gewählt wird.

Dann werden noch kurz die Einschränkungen von .NET und die daraus resultierenden Probleme erläutert, die großen Einfluss auf die prototypische Umsetzung von α-PrintPut haben.

Im darauf folgenden Kapitel wird der Lösungsentwurf von α-PrintPut dargestellt. Zu Beginn wird der erkennbare Nutzen von α-PrintPut aufgeführt. Danach werden anhand eines üblichen Szenarios der Arbeit mit α-PrintPut bei einer Patientenbehandlung die Anforderungen an α-PrintPut ermittelt. Mit Bezug zu diesen Anforderungen wird dann der Lösungsansatz ausgearbeitet.

Auf Basis dieses Lösungsansatzes wird schließlich α-PrintPut als prototypische Umsetzung dieser Lösung entwickelt. Hierfür werden zunächst die relevanten Aspekte der Lösungsumsetzung erörtert und anschließend der prototypische Aufbau von α-PrintPut präsentiert.

Im darauf folgenden Kapitel werden noch kritische Anmerkungen zu α-PrintPut und dem α-Flow zusammengefasst.

Zum Schluss wird ein Resümee gezogen und abschließende Bemerkungen vorgebracht.

3 Grundlagen

Um zu verstehen wir α-PrintPut funktioniert, ist es unerlässlich die Funktionsweise von α-Flow, den Aufbau der α-Docs und das resultierende Dateiformat-Problem zu verstehen. Anschließend wird die Druckarchitektur in Windows und der Unterschied zwischen Rastergrafiken und Vektorgrafiken erklärt.

3.1 α-Flow

Der α-Flow Ansatz soll dazu beitragen, elektronischen institutionsübergreifenden Dokumentenaustausch zu vereinfachen. In unserem Fall sind dies primär inhaltsorientierte Abläufe, die unabhängig von den beteiligten Akteuren und ohne zentrale Verwaltungsmechanismen auskommen müssen. Sie entwickeln sich in nicht vorhersehbarem Ausmaß und werden zunehmend komplexer, da mit der Zeit immer mehr Patienten- und Behandlungsdaten zusammenkommen, die bei späteren Behandlungen relevant sein können, so dass diese Abläufe flexibel und anpassbar sein müssen, z.B. in Fällen einer Parallelbehandlung des Patienten durch mehrere Ärzte, dem Ausbruch mehrerer Krankheiten gleichzeitig etc. Dies erfordert einen komplexeren Ansatz, als ihn herkömmliche elektronische Datenaustauschprozesse wie z.B. Office-Programme und E-Mail bieten. Dazu kommt das Problem, dass bestehende Systeme und Infrastrukturen integriert werden müssen, was einen hohen Anpassungsaufwand auf beiden Seiten (bestehende und neue Systeme) erfordern würde. Um diese Probleme zu lösen, wird auf das Prinzip aktiver Dokumente zurückgegriffen. Die Inhalte werden von der Koordination der Abläufe getrennt und ein Werkzeug zur Integration bestehender und künftiger Inhalte in den α-Flow entwickelt.

3.2 α-Docs

Bevor das Prinzip aktiver Dokumente vorgestellt und im Kontext der Problemstellung dieser Arbeit erläutert wird, wird zunächst kurz auf die Charakteristika digitaler Dokumente im Allgemeinen eingegangen.

Digitale Dokumente unterscheiden sich von herkömmlichen, papierbasierten Dokumenten, nicht nur durch ihre Immaterialität und binär codierte Speicherung, sondern auch durch die Möglichkeiten unbegrenzter, nahezu kostenloser und momentaner Vervielfältigung und Weitergabe, einfache Änderbarkeit von Form und Inhalt, Unabhängigkeit vom genutzten Speichermedium, leichte maschinelle Erfassbarkeit und die Fähigkeit, mit Metadaten versehen zu werden [Sch96].

Neuere Konzepte, wie die strikte Trennung von Form und Inhalt (z.B. in Extended Markup Language (XML)), dynamische Dokumenterzeugung bzw. -änderung (z.B. durch Asynchronous JavaScript and XML (AJAX)) usw. führen zu vorher ungeahnten Einsatzmöglichkeiten. Dadurch ändert sich aber auch die Rolle eines Dokuments - vom passiven, in Papierform erstellten statischen Informationsträger hin zu einem dynamischen, sich erneuerndem und anpassendem Medium, das im Mittelpunkt der Arbeitsabläufe steht oder diese sogar selbst steuert. Im letzteren Fall handelt es sich um aktive Dokumente. Nach Lamarca et al. können aktive Dokumente zum Mittelpunkt dokumentenorientierter Workflows werden, wenn sie die Koordination übernehmen und von sich aus Prozessschritte initiieren und ausführen [LED+02].

Ein a-Doc besteht aus mehreren α-Cards, und zwar aus genau drei Coordination-Cards und beliebig vielen Content-Cards. Die Coordination-Cards beinhalten Prozessinformationen, wie die beteiligten Rollen im XML-Format. Die Content-Cards beschreiben die Prozessschritte der Patientenbehandlung. Eine Content-Card steht für einen Prozessschritt und enthält immer einen Descriptor und optional genau ein Payload. Der Descriptor besteht aus so genannten Adornments und stellt die Metadaten der α-Card dar. Das Payload ist das eigentliche medizinische Dokument mit den Nutzdaten eines beliebigen Dateiformats [NL09].

Die α-Docs sind aktive Dokumente, die eine Interaktion mit sich selbst erlauben, und genau dann aktiv werden, wenn sie mindestens eine α-Card enthalten, die aktive Eigenschaften besitzt. Dazu folgt eine α-Card einem Modell aktiver Eigenschaften, das festlegt, über welche aktiven Eigenschaften die α-Card verfügt, unter welchen Bedingungen eine aktive Eigenschaft ausgelöst wird, welche Verifikationsmechanismen genutzt werden, wie der aktive Code ausgeführt wird und welche Fehlerbehandlungs-, Abbruch- und Benachrichtigungsfunktionen genutzt werden. Somit ist ein Ablauf, der auf α-Docs aufbaut, nicht nur inhalts- sondern auch aktivitätsorientiert, so dass eine α-Card sowohl Eigenschaften eines Dokuments als auch einer Anwendung besitzt.

Weitere Beschreibungen von α-Flow und α-Docs finden sich in [NL10, NSWL11, NL12].

3.3 Dateiformat-Problem

Nachdem nun gezeigt wurde, wie α-Docs aufgebaut und verwendet werden, wird das Problem der Dateiformatinkompatibilität deutlich. Es gibt unzählbar viele Dateiformate im der Computerwelt [Bor95]. Jedes Format kann nicht nur unterschiedliche Arten von Inhalten einbinden, wie Bilder, Text etc., sondern hat auch besondere Eigenschaften, die es von anderen unterscheiden. So können bestimmte Dateiformate besonders schnell durchsucht werden oder eine möglichst hohe Kompressionsrate erreichen. Je nach Einsatzzweck wird das beste Format benutzt. Das Problem, das nun entsteht, ist, dass man zu jedem Dateiformat, wenn es nicht einen integrierten Viewer beinhaltet, auch ein Programm braucht, um dieses zu öffnen und an die enthaltenen Informationen zu gelangen. Ohne so einen Viewer hat man nur eine Folge von Binärdaten, mit der man nichts anfangen kann. Da man bei α-Docs beliebige Dateien als Payload abspeichern kann, ergibt sich das Problem, dass andere Personen diese Datei nicht mehr öffnen können, weil sie keinen passenden Viewer besitzen. Die spontane Lösung, die einem einfällt, ist die Nutzung eines Standardformats, welches den Ansprüchen des Informationsaustausches im Bereich der medizinischen Informatik genügt, und dieses für α-Doc zu verwenden. So gibt es zum Beispiel mit der Health Level 7 (HL7) einen auf XML basierenden Standard für den Austausch klinischer Dokumente namens Clinical Document Architecture (CDA) [1]. Dieses würde sich mit gewissen Anpassungen eignen. Jedoch gibt es zwei Probleme. Zum einen ist das Format nicht weit verbreitet, zum anderen ist der Umstieg auf CDA nicht ohne weiteres möglich. Denn in vielen Einrichtungen werden seit Jahren unterschiedlichste Programme und Dateiformate benutzt. Was macht man mit den bisherigen Dokumenten? Und wie integriert man diese üblichen Dateiformate in das CDA-Format? Es gibt keine Konverter dafür. Außerdem hat jedes Dateiformat seine Besonderheiten und Eigenschaften, die es von anderen unterscheidet. Man kann nicht ohne Weiteres beliebige Dateiformate in andere konvertieren. Viele Programme bieten zwar eine Export-Funktion oder Speicherung in einem anderen Format an, jedoch ist dies insgesamt gesehen eine unbefriedigende Situation. Um ein Format zu konvertieren muss man das Ausgangsformat und das Zielformat kennen und verstehen können. Dafür werden oft Spezifikationen zu den Formaten veröffentlicht. Leider ist dies aber eher selten der Fall. α-PrintPut geht einen anderen Weg und beschränkt sich nur auf die Darstellung des Inhalts. α-PrintPut nutzt die Druckerarchitektur des Systems aus, um das Dateiformat-Problem in den

1 siehe http://www.hl7.org/implement/standards/index.cfm

Griff zu bekommen. Die Folgen dieser Lösung werden in der Kritik am Ende der Arbeit erläutert.

3.4 Druckarchitektur

Im Windows-Betriebssystem arbeiten mehrere Benutzer, die sich einen Drucker teilen. Die Druckarchitektur betrifft also meistens mehrere Benutzer eines Systems. Diese Eigenschaft und daraus resultierende Folgen für α-PrintPut werden später im User Account Control (UAC)-Kapitel genauer erläutert.

Die Kernelemente der Druckarchitektur in Windows sind der Spooler-Dienst und die Menge der installierten Druckertreiber. Der Spooler-Dienst wird beim Systemstart, noch bevor sich ein Benutzer anmeldet, gestartet und läuft im Hintergrund als „spoolsv.exe“-Prozess unter dem System-Benutzerkonto in der Session 0. Dieser ist für die Druckaufgaben zuständig. Zu seinen Aufgaben gehören das Verwalten der Druckkomponenten in der Registry, das Entgegennehmen des Datenstroms, welchen die Applikationen bei einem Druckvorgang generieren, die Konvertierung des Datenstroms in ein für die Druckerhardware verständliches Format, die Entscheidung, ob ein Druckauftrag lokal oder entfernt (z.B. auf einem über das Netzwerk erreichbaren Rechner) verarbeitet wird und vieles mehr.

Führt der Benutzer einen Druckbefehl aus, so wird ein Druckauftrag erzeugt, indem zum Beispiel die Graphic Device Interface (GDI)-Funktionen aufgerufen werden. Die GDI dient als eine mögliche Programmierschnittstelle zu der Spooler Application Programming Interface (API) und wird meistens von einem Programm aufgerufen [Yua01]. Diese Schnittstelle sorgt dafür, dass die grafischen Informationen mit möglichst hoher Ausgabequalität angezeigt werden. Die GDI erzeugt entweder einen EMF -Datenstrom oder rendert zusammen mit dem Druckertreiber ein druckbares Bild und schickt dieses an den Spooler. EMF wird bei den meisten Windows-Programmen für den Druckvorgang benutzt, da es kleiner und vor allem portabler ist als das vom Drucker unterstützte RAW-Format. Manche Programme jedoch, wie der Adobe Reader, schicken beim Drucken direkt einen für den Drucker kompatiblen Datenstrom im RAW-Format ohne Umweg über die GDI [1].

1 vergleiche http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/sag_printconcepts_cpu_datatypes.mspx?mfr=true

Der Spooler besteht aus mehreren Komponenten, die alle mit den beschränkten Benutzerprivilegien laufen. Sie interpretieren den EMF-Datenstrom und erweitern ihn unter Umständen um zusätzliche Informationen, wie das Seitenlayout. Der Aufbau der Spoolerkomponenten wird in 3.1 dargestellt.

Die „winspool.drv“-Schnittstelle ermöglicht es auf die bereits beschriebene „spoolsv.exe“ über Interprozesskommunikation (RPC) zuzugreifen. Diese lässt dann den Vermittler „spoolss.dll“ entscheiden, welcher Druckanbieter (Print Provider), der das gewählte Druckergerät unterstützt, schließlich aufgerufen wird, um den Druckauftrag zu verarbeiten. Wenn die Druckerhardware lokal am System angeschlossen ist, so befinden sich der „Client“- und „Server“-Teil auf der gleichen Maschine [1].

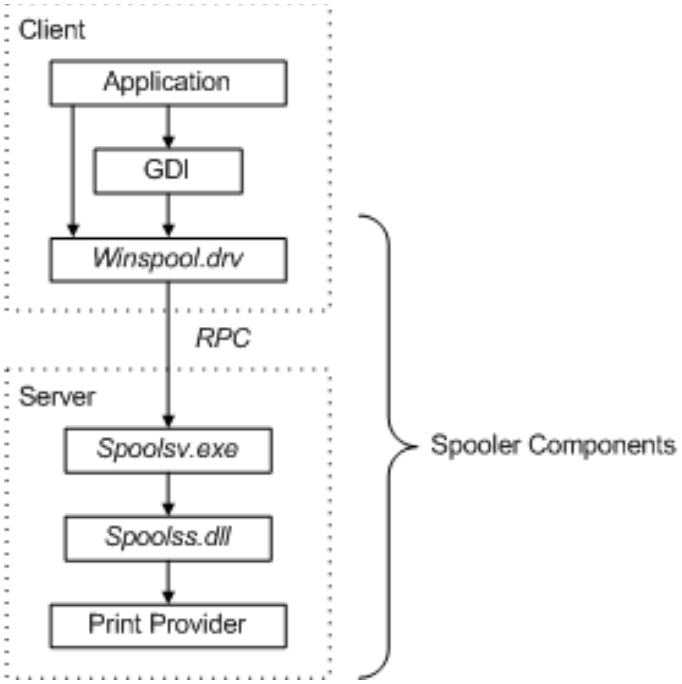


Bild 3.1: Aufbau der Spooler-Komponenten

Es gibt verschiedenste Druckanbieter. Windows besitzt einen lokalen Druckanbieter (localspl.dll), der alle Druckaufträge an lokal angeschlossene Drucker verarbeitet, einen Netzwerk-Druckanbieter (win32spl.dll), der alle Aufträge an ein fremdes System weiterleitet, in welchem die Druckaufträge an den im Fremdsystem wiederum lokalen Druckanbieter geleitet werden, und andere von Druckerherstellern erzeugte Netzwerk-Druckanbieter. Damit der Spooler (Spoolss.dll) die Druckanbieter aufrufen kann, müssen

1 vergleiche http://blogs.technet.com/b/askperf/archive/2007/06/19/basic-printing-architecture.aspx

alle Druckanbieter bestimmte Funktionen implementieren und zugänglich machen. Eine Liste dieser Funktionen kann man bei Microsoft Developer Network (MSDN) finden.

Die Abbildung 3.2 veranschalicht die möglichen Wege, die ein Druckauftrag über verschiedene Druckanbieter nehmen kann.

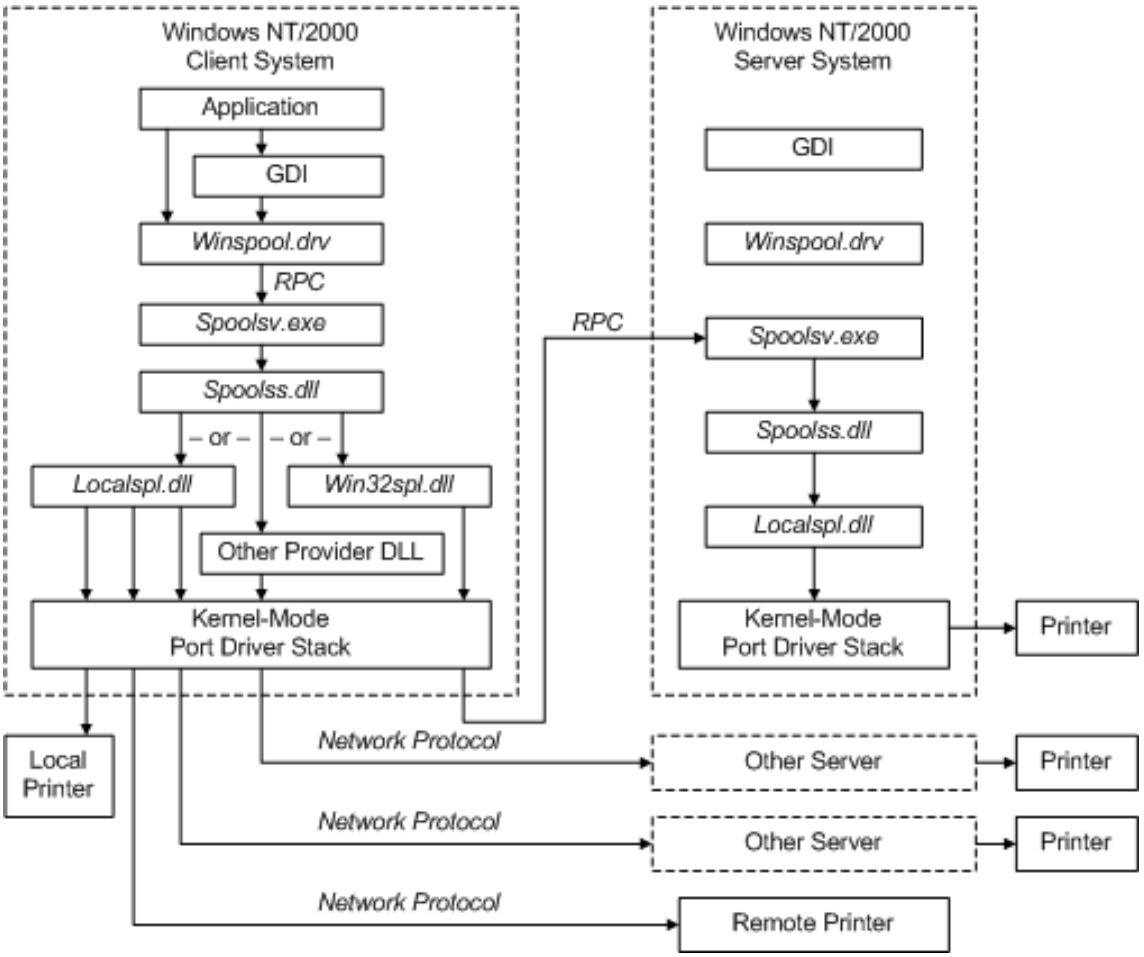


Bild 3.2: Mögliche Wege eines Druckauftrags über verschiedene Druckanbieter

Im Folgenden wird übersichtshalber und aufgrund der Relevanz für α-PrintPut nur auf den lokalen Druckanbieter eingegangen.

Ein in Windows installierter Drucker ist stets an genau einen bestimmten Druckertreiber, Drucker-Port und Druckprozessor gebunden und je nachdem variiert der Druckvorgang. Kommt der Druckauftrag beim Druckanbieter an, so wird dieser unabhängig vom Format des Datenstroms in einer oder mehreren Spool-Dateien (.spl), sowie Metadateien (.shd) auf der Festplatte im Spooler-Unterordner des Systemverzeichnisses gepuffert bzw. ausgelagert und später bei der Abarbeitung der im Puffer gehaltenen

Aufträge wieder eingelesen und überprüft, um welches Format es sich handelt. Wenn es sich um ein EMF-Format handelt, wird die Konvertierung mit Hilfe eines Druckprozessors angestoßen. In Windows gibt es nur einen vorinstallierten Druckprozessor, nämlich die winprint.dll, welcher die Spool-Daten einliest und transferiert, indem er sie mit Hilfe der GDI und unter Einbeziehung des Druckertreibers verarbeitet, so dass ein für den Drucker verständlicher Datenstrom, in den meisten Fällen im RAW-Format, entsteht. Ein Bespiel für den Datenstrom sind Printer Control Language-Befehle für viele Hewlett-Packard (HP)-Drucker oder Postscript-Befehle für Postscriptfähige-Drucker [Pre85, Sys85]. Dieser konvertierte Datenstrom wird dann wieder dem Spooler übergeben. Die Funktionsweise des lokalen Druckanbieters wird in 3.3 veranschaulicht.

Ist das Format für den Drucker kompatibel, so leitet der Druckanbieter den Datenstrom an den lokalen oder entfernten Drucker weiter. Der Zugriff des lokalen Druckanbieters auf einen lokalen oder entfernten Drucker läuft über den Druckmonitor (Print Monitor). Der Druckmonitor besteht aus dem Language Monitor, der dafür sorgt, dass das Ausgabeformat mit dem erwarteten Datenformat übereinstimmt und diesen nach Bedarf noch anpasst, sowie dem Port Monitor, der den konvertierten Druckdaten schließlich an den Drucker überträgt. Er bestimmt die Ausgabeschnittstelle, also den Anschluss des Druckers, zum Beispiel am COM- oder LPT-Port. Diese verbindet er mit dem unter Systemrechten laufenden Porttreiber, der den Zugriff auf die Ein- und Ausgabe des Hardwaregeräts ermöglicht. Zu diesen Porttreibern gehören zum Beispiel der serielle oder parallele Treiber. Sobald der Datenstrom den Port Monitor erreicht, ist der Druckauftrag aus Sicht des Betriebssystems beendet!

Zusammenfassend folgt ein üblicher Gesamtablauf eines Druckvorgangs (siehe 3.4):

1. Benutzer bestätigt den Druckvorgang in einer Anwendung und diese übergibt den Druckauftrag an die GDI-Engine.
2. Die GDI erzeugt einen EMF-Datenstrom unter Berücksichtigung des Druckertreibers und leitet diesen an den Spooler weiter.
3. Der Druckanbieter erkennt, dass es sich um einen EMF-Datenstrom handelt und schickt ihn zur Umwandlung an den Druckprozessor (print processor).
4. Dieser konvertiert das EMF-Format mit Hilfe der GDI und des Druckertreibers in geräteabhängigen Rohdatenstrom, der vom Drucker verstanden wird, und übergibt diesen wieder dem Spooler.
5. Der Druckanbieter sieht, dass es ein RAW-Datenformat ist und leitet diesen an den Druckmonitor weiter.

6. Schließlich erreicht der Datenstrom den Drucker.

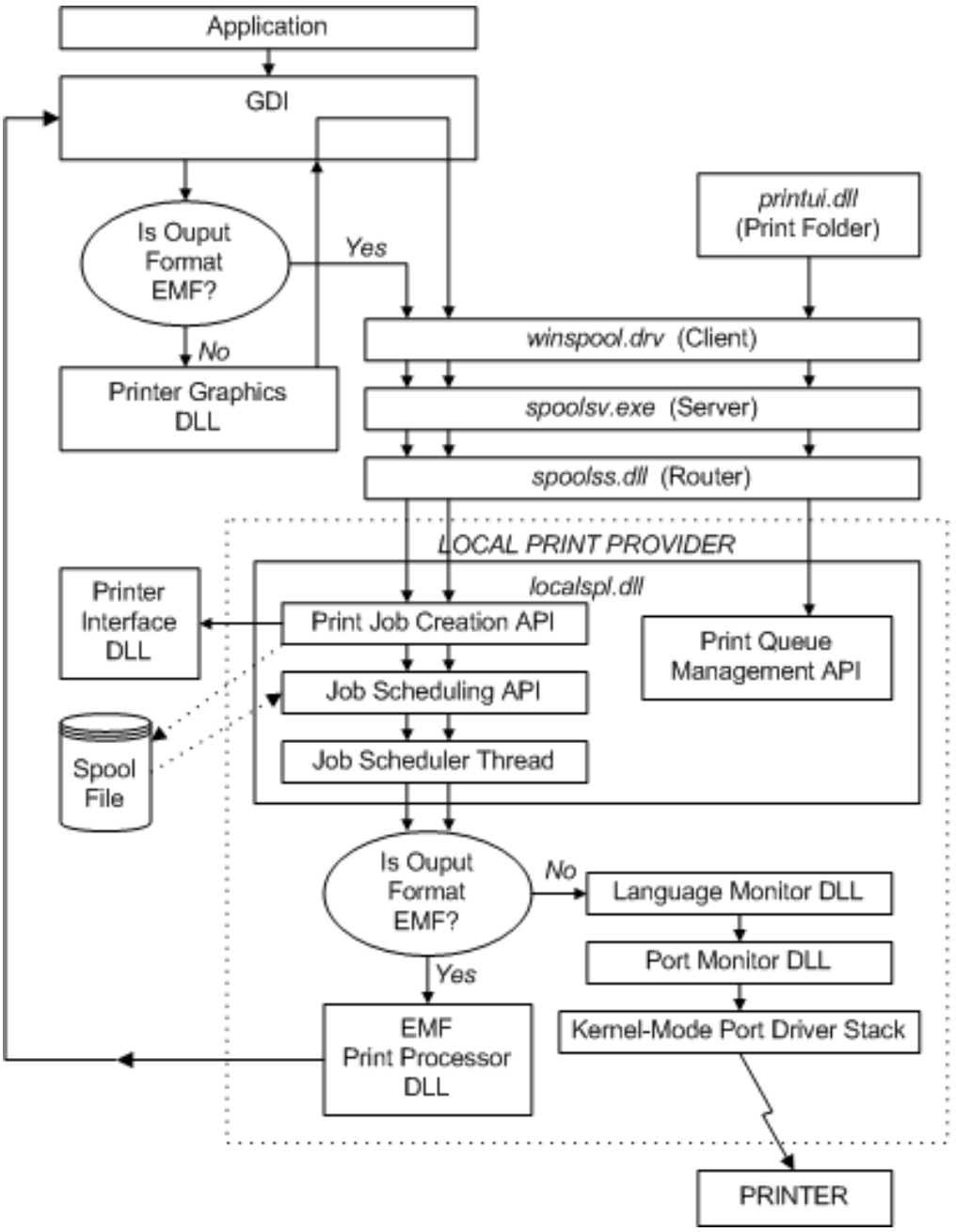


Bild 3.3: Funktionsweise des lokalen Druckanbieters

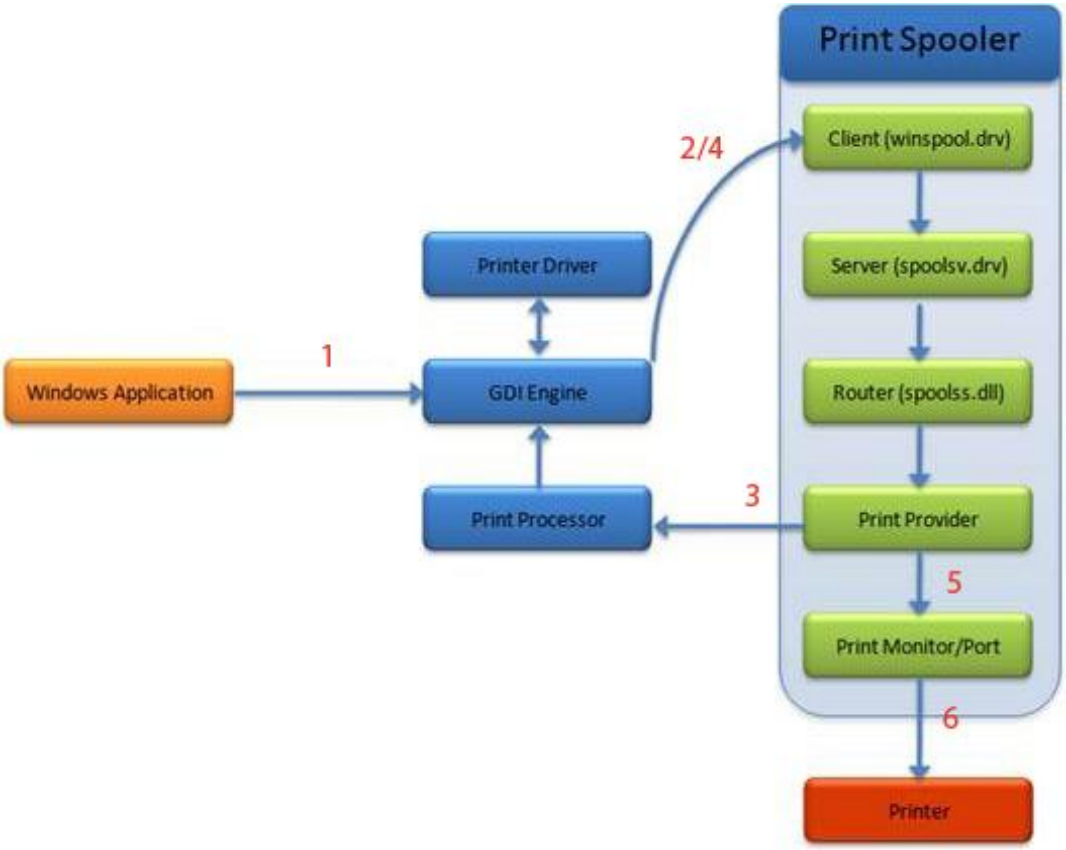


Bild 3.4: Gesamtablauf eines Druckvorgangs

3.5 Rastergrafiken vs. Vektorgrafiken

α-PrintPut versucht das Dateiformat-Problem zu lösen, indem die Darstellung des Inhalts eines beliebigen druckbaren Dokuments festgehalten wird. Dafür eignen sich zwei Typen von Grafiken: Rastergrafiken und Vektorgrafiken [HK01]. Rastergrafiken enthalten für jeden Bildpunkt die Farbinformation und den Wert für die Lichtundurchlässigkeit [FVDF+94]. Da ein Bild sehr viele Bildpunkte besitzt, sind Rastergrafiken immer relativ große Dateien. Vektorgrafiken beinhalten Objekte einfacher Form, wie Kreise oder Rechtecke, deren Parameter für die Darstellung des Bildes verantwortlich sind. Beispielsweise hat ein Kreis Informationen zu Farbe, Liniendicke und Radius. Ein Vorteil von Vektorgrafiken gegenüber den Rastergrafiken ist die Dateigröße. Vektorgrafiken sind üblicherweise kleiner als entsprechende Rastergrafiken. Der aber wohl wichtigste Punkt ist die Skalierbarkeit, denn Vektorgrafiken können beliebig ohne einen Qualitätsverlust skaliert werden [FJJ03]. Im Gegensatz zur Vektorgrafik verliert die Rastergrafik bei der Skalierung Informationen und erscheint dadurch verpixelt. Dies wird in 3.5 dargestellt. Die Nachteile von Vektorgrafiken sind, dass sie im Vergleich zu Rastergrafiken meist nicht deren Detailreichtum anbieten, da sie aus geometrischen Figuren zusammengesetzt sind, und dass der Prozessor für die Darstellung der Grafik mehr rechnen muss. Für Fotografien von Personen oder Landschaften sind Rastergrafiken die bessere Wahl, aber für den Zweck von α-PrintPut, der Darstellung von medizinischen Dokumenten, sind Vektorgrafiken vorzuziehen.

Bild 3.5: Skalierbarkeit der Vektorgrafik und Rastergrafik im Vergleich

4 Systemumgebung

α-PrintPut wird für eine institutionsübergreifende Patientenbehandlung eingesetzt und sollte deshalb möglichst viele Betriebssysteme von Microsoft unterstützen und dabei zukunftsorientiert sein, damit es auch in zukünftigen Windows-Versionen läuft oder zumindest möglichst geringe Anpassungen dafür nötig sein werden. Deshalb wurde α-PrintPut auf Betriebssysteme ab Windows 2000 ausgerichtet, da die damals eingeführte Spooler-API bis heute noch in den Systemen verankert ist und ältere Windows-Versionen praktisch kaum noch eingesetzt werden und einen deutlichen Mehraufwand in der Entwicklung bereitet hätten. Man muss immer die Vor- und Nachteile abwägen, denn durch die Integration von älteren Betriebssystemen können einige erst in späteren Versionen von Windows eingeführte Systemschnittstellen nicht benutzt werden. Andererseits aber kann man nicht erwarten, dass in dem Einsatzort, den verschiedensten Arztpraxen und Krankenhaussystemen mit sehr vielen Computern, immer die neusten Windows-Lizenzen und aktuelle Betriebssystemversionen installiert sind. Denn viele Systeme können, bzw. dürfen sogar nicht aktualisiert werden, wenn die dafür entwickelten unersetzlichen Programme nicht mehr auf den neuen Betriebssystemen laufen, wie es bei vielen Altsystemen der Fall ist. Die Migration dieser Altsysteme ist oft sehr aufwendig, kostspielig und komplex. Darum ist eine möglichst große Abdeckung gewünscht, welche α-PrintPut ermöglicht, indem es sowohl 32-Bit-Systeme als auch 64-Bit-Systeme ab Windows 2000 bis zum heutigen Stand (Windows 7) unterstützt.

Diese Unterstützung wird durch zwei Vorgehensweisen erreicht: Einerseits durch die Wahl des .NET-Frameworks als Plattform für die Entwicklung von α-PrintPut. Denn mit dem .NET-Compiler wird plattformunabhängiger Common Intermediate Language (CIL)-Code generiert. Dieser wird erst zur Laufzeit von dem Just-In-Time-Compiler (Jitter) der .NET-Laufzeitumgebung (der .NET Execution Engine) ausgeführt. [Tro10] Die Abbildung 4.1 stellt diesen Ablauf dar.

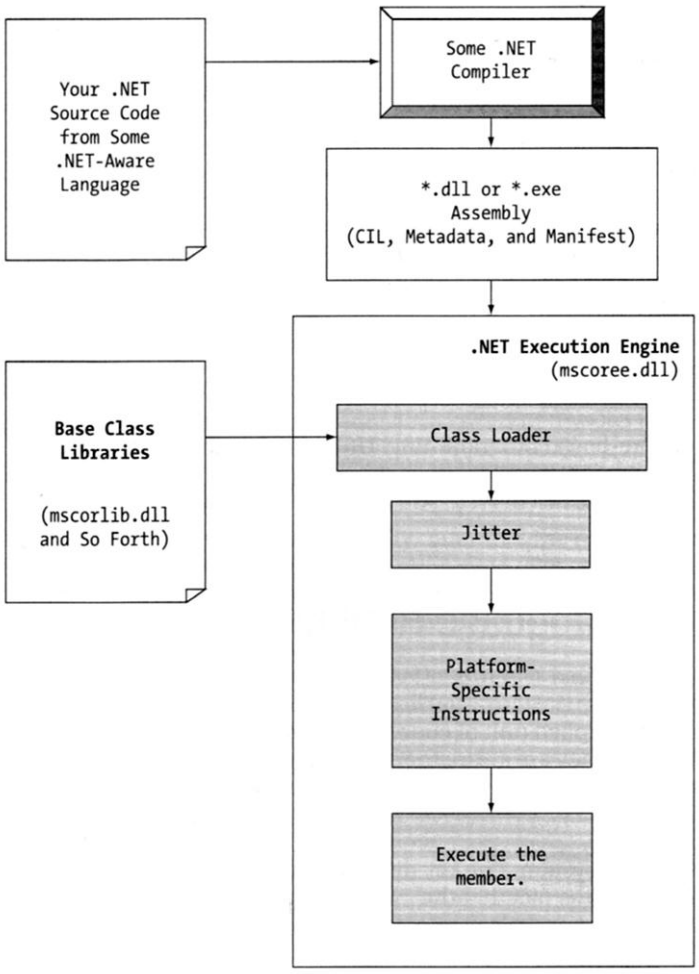


Bild 4.1: Ablauf der Code-Generierung und Ausführung im .NET-Framework

Dies ermöglicht α-PrintPut auf allen Plattformen auszuführen, auf denen eine .NET-Laufzeitumgebung installiert ist. Das .NET-Framework mit der Laufzeitumgebung wird von Microsoft für Betriebssysteme ab Windows 98 zur Verfügung gestellt. Diese Plattformunabhängigkeit wird jedoch bei α-PrintPut teilweise aufgebrochen. Der Grund hierfür ist, dass α-PrintPut an bestimmten Stellen sehr systemnah mit dem Betriebssystem arbeitet. Insbesondere bei der Installation und Konfiguration des benötigten Druckertreibers. Darauf wird in einem späteren Kapitel noch detailliert eingegangen.

Der andere Grund für die besondere Systemkompatibilität und Interoperabilität ist einerseits der Abwärtskompatibilität von Microsoft zu verdanken, die es ermöglicht, Programme, die für ältere Betriebssysteme geschrieben worden sind, meist ohne größere Probleme oder Anpassungen auch in neuen Betriebssystemen laufen zu lassen. Ich habe mich bewusst gegen die Möglichkeit entschieden, α-PrintPut systemabhängig zu

entwickeln, also immer zu prüfen, welche Version des Systems und welches Service Pack (SP) auf diesem installiert ist und je nach Fall anders zu reagieren. Denn der Code müsste stets für neue Windows-Versionen oder Updates (Service Packs) aktuell gehalten werden und alle möglichen Systemversionen prüfen können.

Dies hätte viel Wartungsaufwand und für α-PrintPut nur wenig Nutzen gebracht. Außerdem könnte ein fehlendes Feature im System, welches die ordnungsgemäße Funktionsweise von α-PrintPut verhindert, im Nachhinein durch ein eingespieltes Update wiederhergestellt werden. Deshalb wurde bei der Entwicklung von α-PrintPut darauf geachtet, eine möglichst weite Abdeckung der Betriebssysteme zu erreichen, indem auf die Implementierung neu eingeführter Systemschnittstellen verzichtet wurde, wobei dennoch stets eine Zukunftsorientierung verfolgt wurde. So wurden auch die Besonderheiten des ab Windows Vista eingeführten UAC berücksichtigt. Die Einwirkungen der UAC auf α-PrintPut werden später noch genauer betrachtet.

Zurzeit wird mindestens das .NET-Framework 2.0 für α-PrintPut benötigt. Dieses Framework gibt es für Windows 2000, 2003 und XP. Die neueren .NET-Frameworks für Vista und Windows 7 können ebenfalls eingesetzt werden, da diese abwärtskompatibel sind. Eine detaillierte Auflistung der .NET-Frameworks und der unterstützten Betriebssysteme findet man auf der MSDN-Seite.

5 Verwandte Lösungsansätze

Nachdem nun das Problem der Formatinkompatibilität und der Funktionsweise der Druckarchitektur von Windows erklärt wurde, wird nun gezeigt wie man den Druckvorgang so anpassen könnte um aus beliebigen druckbaren Dateiformaten ein einheitliches Format bekommt, das unabhängig vom Anwendungsprogramm, dem Betriebssystem und der Hardwareplattform ist und für die Payloads der α-Cards in den α-Docs benutzt werden kann.

Dieser Lösungsweg ist nicht neu und wurde bereits von vielen ähnlichen Programmen durchgeführt, jedoch gibt es für diesen Lösungsweg viele verschiedene Realisierungsmöglichkeiten. Im Folgenden werde ich einige der Alternativen aufzählen und ihre jeweiligen Stärken und Schwächen aufzeigen, die schließlich auch Einfluss auf die Realisierung von α-PrintPut hatten.

5.1 EMF Printer und Spool-Datei

Zum einen gibt es die so genannten virtuellen EMF-Drucker. Wie im Kapitel zur Druckarchitektur erklärt wandelt die GDI mit Hilfe des Druckertreibers den an den Drucker gesendeten Datenstrom in das EMF-Format um. Anschließend wird diese EMF-Datei als Spool-Datei auf die Festplatte geschrieben, bevor sie weiterverarbeitet wird. Die EMF-Drucker nutzen verschiedene Methoden, um an diese EMF-Datei ranzukommen und sie zu extrahieren. Ein gutes Bespiel dafür ist das Open Source Projekt „EMF Printer““, der Virtual Printer oder PrintMulti.

Diese nutzen einen virtuellen Druckertreiber, um an die EMF-Datei direkt zu gelangen. Virtual Printer und PrintMulti nutzen dafür einen eigenen Druckprozessor, der von der Spool-Datei eine Kopie anlegt und diese weiterverarbeitet. Virtual Printer und EMF-Printer funktionieren aber zurzeit leider nicht unter Windows Vista, Windows 7 und auf 64-Bit Plattformen. Einer der Gründe dafür ist, dass mit Einführung von Windows Vista die Druckertreiber nicht mehr im Kernel-Mode laufen dürfen. Der Vorteil der EMF-Printer ist, dass sie unabhängig von dem Drucker-Treiber funktionieren, sie laufen mit beliebigen Druckern, für sie ist nur wichtig, dass der Druckauftrag über die

GDI im EMF-Format kommt. Eine mögliche alternative Idee wäre den Spool-Ordner zu überwachen und jede ankommende Datei kopieren zu lassen. Das Problem dabei ist, dass dies bei allen gedruckten Dateien passieren würde und die Überwachung, da sie immer in zeitlichen Abschnitten stattfindet, relativ prozessorbelastend wäre und das System ausbremsen würde. Außerdem werden die Spool-Dateien nach dem Drucken wieder gelöscht, außer man setzt die Funktion, Druckeraufträge nach dem Drucken zu behalten, in den Druckereinstellungen.

Ein weiteres Problem ist, dass EMF-Spool-Dateien nicht mit normalen EMF-Dateien kompatibel sind, da sie noch zusätzliche Informationen bereithalten und eine etwas andere Struktur haben. Obwohl das EMF-Format bereits von Microsoft dokumentiert wird, gibt es immer noch viele nicht dokumentierte Teile, die von einer Internet-Community analysiert und veröffentlicht worden sind, so zum Beispiel auch die Möglichkeit, aus einer EMF-Spool-Datei die EMF-Datei zu extrahieren. Bei dem EMF-Format handelt es sich um eine Vektorgrafik. Die Unterschiede zu einer Rastergrafik wurden bereits im Kapitel „Rastergrafiken vs. Vektorgrafiken“ klargemacht. Es gibt auch einige Tools, die EMF-Dateien darstellen können, jedoch ist das Format noch nicht wirklich verbreitet und seit Windows Vista nicht mehr besonders gefördert. Es wird immer mehr von dem XML Paper Specification (XPS)-Format abgelöst. Doch das Hauptproblem von diesem Verfahren ist die Tatsache, dass Programme nicht immer über die GDI und das EMF-Format drucken! Einige Programme können auch den Drucker an der GDI vorbei ansprechen und ihm die Daten im kompatiblen RAW-Format, zum Beispiel PostScript oder Printer Command Language (PCL), schicken. In diesem Fall versagen die EMF-Printer komplett. Deshalb bietet sich diese Lösung nicht für α-PrintPut an.

5.2 RAW Printer und Redirection Port

Ein anderer Lösungsansatz geht den Weg, den verarbeiteten RAW-Datenstrom, der an den Druckerport geschickt werden soll, abzufangen bzw. umzuleiten in eine Datei. Viele Programme unterstützen diese Funktion von sich aus. Dies sieht man daran, dass im Druckdialog die Möglichkeit besteht, in eine Datei zu schreiben. Jedoch gibt es nach dieser Aktion keine Möglichkeit die Datei automatisch weiter zu verarbeiten, um sie in das α-Doc zu integrieren. Auch gibt es keine Möglichkeit in den Druckprozess einzugreifen und die einzigen Einstellungsmöglichkeiten sind diese, die der Drucker anbietet.

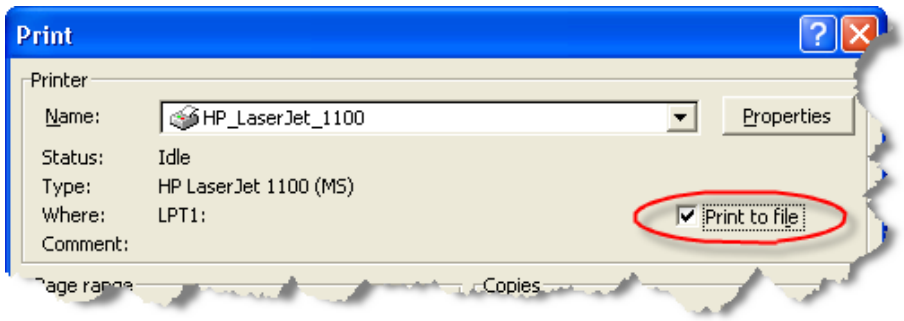


Bild 5.1: Möglichkeit der Umleitung in eine Datei

Außerdem bieten viele Programme diese Funktion gar nicht an und darum kann man sich auf diese magere Lösung nicht verlassen. In diesem Fall kann man einen neuen Drucker hinzufügen und in der Konfiguration als Port „FILE:“ wählen. Dies hat zur Folge, dass beim Drucken an diesen Drucker ein Fenster erscheint und nach einem vollständigen Pfad inklusive Dateinamen fragt, in welchen der Datenstrom geschrieben werden soll. Die anderen Probleme bleiben aber weiterhin.

Bild 5.2: Umleitungs-Dialog

So ist dieser Lösungsansatz sehr unhandlich und bietet praktisch keine Anpassungsmöglichkeit. Eine automatisierte Weiterverarbeitung ist nicht möglich, da der „FILE:“-Port und der daran hängende Port Monitor von Windows zur Verfügung gestellt sind, und keine Modifizierung anbietet.

Es gibt jedoch alternative „Port Monitors“, die neben der Umleitung in eine Datei mehr Flexibilität anbieten. So zum Beispiel das Open-Source Projekt Redirection Port Monitor (RedMon). RedMon fängt den Datenstrom ab und kann diesen als Datei speichern lassen, an einen anderen Drucker im System schicken, oder ein beliebiges Programm starten und den Datenstrom in den Standardeingabedatenstrom des erzeugten Prozesses schreiben, so dass dieser sich dann um die Weiterverarbeitung selbst kümmern muss. RedMon kann also vielseitig benutzt werden, je nachdem was man erreichen will. RedMon unterstützt

offiziell noch nicht Windows Vista und Windows 7 und hat keine 64-Bit Unterstützung, dies ist aber in Arbeit und soll Ende 2011 umgesetzt sein. Eine inoffizielle 64-Bit-Version wurde von Gilles Vollant veröffentlicht, enthält aber noch einige Fehler. So funktioniert die Einrichtung von RedMon nicht über die grafische Oberfläche (siehe Abbildung 5.3). Jedoch lässt sich RedMon einfach direkt über Eingriffe in die Windows-Registry installieren, konfigurieren und entfernen.

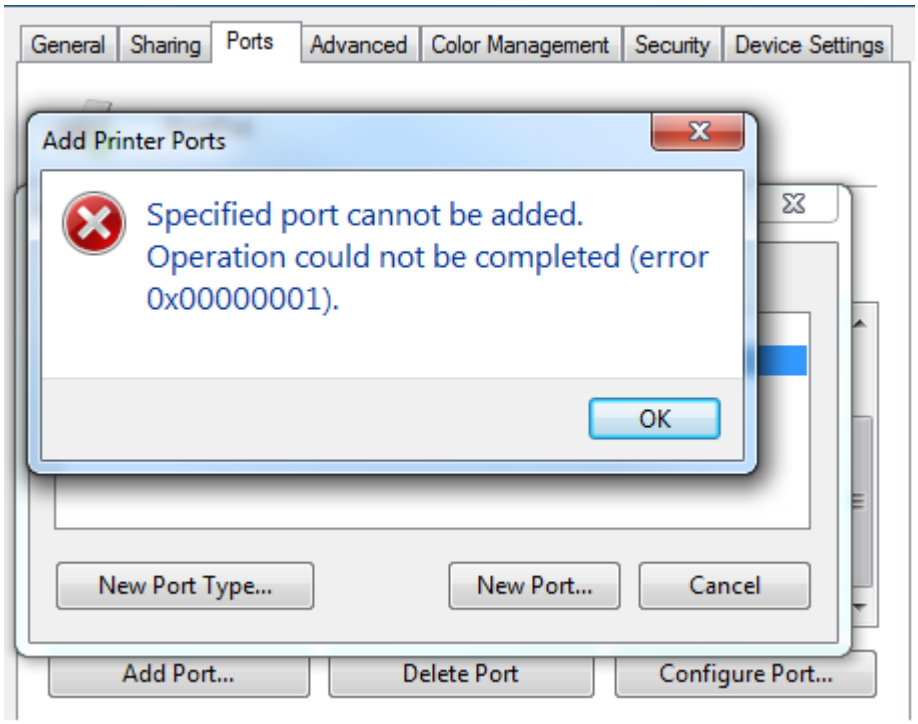


Bild 5.3: Fehler bei Konfiguration von RedMon

Der Nachteil dieser Verfahren, die den verarbeiteten RAW-Datenstrom abfangen, ist, dass der ankommende Datenstrom davon abhängt, an welche Art von Drucker man druckt bzw. welches Format der Drucker erwartet. Zusammen mit einem Postscript-Drucker könnte man so aus beliebigen Anwendungen mit dem RedMon die dazugehörigen Postscript-Dateien bekommen und diese an ein beliebiges Programm zur Weiterverarbeitung schicken.

5.3 Zusammenfassung

Zusammenfassend kann man sagen, dass die in Windows integrierten Möglichkeiten zwar mit jedem Programm funktioniert, die Dateiumleitung aber keinerlei Anpassung und

Eingreifen in den Druckvorgang ermöglicht und damit den Anforderungen von α-PrintPut nicht genügt.

Die EMF-Printer bieten eine gute Lösung an, da sie in den Druckprozess eingreifen und beliebige benutzerdefinierte Aktionen zulassen. Das EMF-Format ist jedoch nicht besonders verbreitet. Zusammen mit Konvertern wie ImageMagick könnte man dieses Problem zwar in den Griff kriegen, indem man die EMF-Dateien in handlichere weit verbreitetere Formate konvertieren lässt. Das Hauptproblem ist jedoch, dass die EMF-Printer nicht von allen Programmen die Druckaufträge abfangen können und ziehen damit im Vergleich zur RedMon-Lösung den Kürzeren. Denn diese vereint die ganzen positiven Eigenschaften und beschreibt genau die Funktionalität, die α-PrintPut benötigt und umsetzt.

Nun stellt sich die Frage, welches Format man für α-Doc verwenden sollte. Im Druckerbereich haben sich zwei Sprachen durchgesetzt, PCL von HP und Postscript von Adobe. Beide sind sehr mächtig. Außerdem stellt sich die Frage, ob man diese Dateien für das α-Doc benutzen sollte, oder es Vorteile hätte, diese in ein anderes Format zu konvertieren. So tut das Programm PDFCreator RedMon zusammen mit einem Postscript-Drucker benutzen, um Postscript-Dateien generieren zu lassen und diese dann mit Ghostscript in das PDF-Format zu konvertieren. PDF, PCL und Postscript können alle Vektorgrafiken beschreiben, jedoch gibt es einige grundliegende Unterschiede.

6 Printer Command Language, Postscript und Portable Document Format im Vergleich

Sowohl HP's PCL, als auch Adobe's Postscript sind Seitenbeschreibungssprachen. Sie beschreiben den exakten Aufbau einer Seite und ermöglichen so dem Drucker den Druck des Dokuments. Beide haben ihre Vor- und Nachteile. PCL-Treiber übernehmen die meiste Arbeit des Rendern, also der Generierung der Grafik aus dem PCL-Datenstrom, und schickt den entstandenen Datenstrom an den Drucker. Die Postscript-Treiber hingegen senden die Seitenbeschreibung an den Drucker und dieser sorgt für das Rendern. Dadurch, dass das Rendern bei PCL am Computer geschieht und dieser schneller als der Drucker arbeitet, kann deutlich schneller gedruckt werden. Für den Fall von α-PrintPut spielt dies aber keine Rolle, da der Postscript-Datenstrom nur virtuell an einen Drucker gesendet wird und ebenfalls direkt im Computer verarbeitet wird. PCL ist eine einfachere Sprache als α-PrintPut und bietet weniger Möglichkeiten als Postscript an und kann dadurch im schlimmsten Fall qualitativ schlechtere Ergebnisse liefern. Außerdem gibt es bei PCL das Problem, dass die Drucker den Druckauftrag leicht unterschiedlich durchführen, denn PCL ist Druckerabhängig, während Postscript Geräteunabhängig funktioniert. Darum ist Postscript dem PCL vorzuziehen.

Jedoch sind das Postscript- und PCL-Format als Seitenbeschreibungssprache für Drucker gedacht und für die Grafikdarstellung in der Windowswelt überhaupt nicht verbreitet. Es gibt zwar Tools, wie zum Beispiel GSview, eine grafische Schnittstelle für den Postscript-Interpreter Ghostscript, mit denen man die Dateien öffnen kann, jedoch ist dieser auf kaum einem Computer standardmäßig vorinstalliert. Ein wesentlich verbreiteteres Dateiformat ist das PDF-Format. Das PDF-Format ist wie Postscript ebenfalls gerätunabhängig und stellt die Seiten als Vektorgrafiken dar. Mit Hilfe von Ghostscript lässt sich die Postscript-Datei in eine PDF-Datei sehr einfach umwandeln. Dabei bleiben alle Informationen über Vektor- und Pixel-Elemente erhalten! Der Vorteil von PDF-Dokumenten ist, dass diese sehr weit verbreitet sind und praktisch einen Stan-

dard im Netz zur Veröffentlichung von Dokumentenbeschreibungen darstellen. Sie bieten die Möglichkeit an, in den Dokumenten zu navigieren. Die Postscript-Dateien müssen vom Anfang bis zum Ende des Dokumentes verarbeitet werden, während PDF-Seiten beliebig dargestellt werden können. Ein weiterer Vorteil von PDF-Dateien ist, dass sie platzsparender sind, da sie mit genormten Kompressionsalgortithmen komprimiert sind. Schließlich lassen sich PDF-Dateien auch noch verschlüsseln. Jedoch gibt es auch Nachteile von PDF-Dateien. Die Konvertierung in das PDF-Format ist sehr komplex und es kann viel schiefgehen. Ein Fehler bei der Konvertierung einer Seite sorgt dafür, dass die ganze Seite leer bleibt. Außerdem gibt es aufgrund der Komprimierung und der darauf folgender Intransparenz der PDF-Dateien nur begrenzte Möglichkeiten, das Dokument zu bearbeiten. Will man eine Änderung machen, so muss man das Originaldokument wieder öffnen und den Konvertierungsprozess erneut starten. Aber auch bei Postscript-Dokumenten ist es nicht wirklich besser, man kann nur Grundelemente, wie einzelne Symbole oder Wörter, wenn man sich auskennt und mit enormem Aufwand, selber bearbeiten ohne diese neu generieren lassen zu müssen. PDF eignet sich für das Arbeiten am Desktoprechner eher. Darum wird es auch bei α-PrintPut als Ausgabeformat verwendet.

7 Einschränkungen von .NET

α-PrintPut wird in der systemeigenen Sprache C# der .NET Laufzeitumgebung entwickelt. Das .NET Framework bietet eine umfassende Sammlung von Werkzeugen und APIs, jedoch ist es noch relativ jung und es gibt Funktionen, die noch nicht integriert worden sind. Im Kapitel Systemumgebung wurde beschrieben, wie die Laufzeitumgebung von .NET funktioniert. Diese ermöglicht die Ausführung aller .NET-Applikationen und bildet den Kern bei der Ausführung von .NET–Code. Sie tut den „managed Code“ (Intermediate Language-Code) in den nativen Maschinencode übersetzen, der anschließend vom Prozessor ausgeführt werden kann und bietet noch weitere Funktionalitäten an, wie die automatische Speicherverwaltung und Verifizierung der Typsicherheit an.

Früher hatte das Betriebssystem kein .NET vorinstalliert. Windows Vista war das erste Betriebssystem mit integriertem .NET Framework. Doch auch mit Einführung ist es immer noch so, dass die meisten Systemkomponenten mit der Windows 32-API oder COM entwickelt worden sind und Schnittstellen zu diesen bereitstellen. Solche Komponenten werden direkt in Maschinencode übersetzt und ausgeführt und laufen nicht unter der .NET-Laufzeitumgebung. Code, der nicht von der Laufzeitumgebung verwaltet wird, bezeichnet man „unmanaged Code“. So ist die Druckarchitektur in Windows in unmanaged Code geschrieben, woraus sich einige Probleme ergeben. Denn diese besteht aus ausführbaren Dateien im Portable Executable (PE)Format. Ein Beispiel dazu sind die Dynamic Link Library (DLL)-Dateien, die Funktionen exportieren, um von anderen Programmen ausgeführt werden zu können. So werden in Windows auch die meisten Systemaufrufe getätigt.

Hier liegt schon das erste Problem, denn beim Aufruf vom fremden unmanaged Code aus dem eigenen .NET-Code verlässt man die Laufzeitumgebung und gibt damit zum Beispiel die Typsicherheit auf. Diese Aufrufe werden durch den Platform Invoke (PInvoke) Dienst ermöglicht. Es sorgt für den reibungslosen Ablauf der nicht verwalteten Funktionen der fremden DLL und kümmert sich um die Konvertierung der Argumente und des Rückgabetyps. Dieser Vorgang wird Marshalling genannt und sorgt dafür, dass der Datentyp außerhalb der Laufzeitumgebung richtig interpretiert werden kann. In Abbildung 7.1 wird dieser Ablauf modelliert.

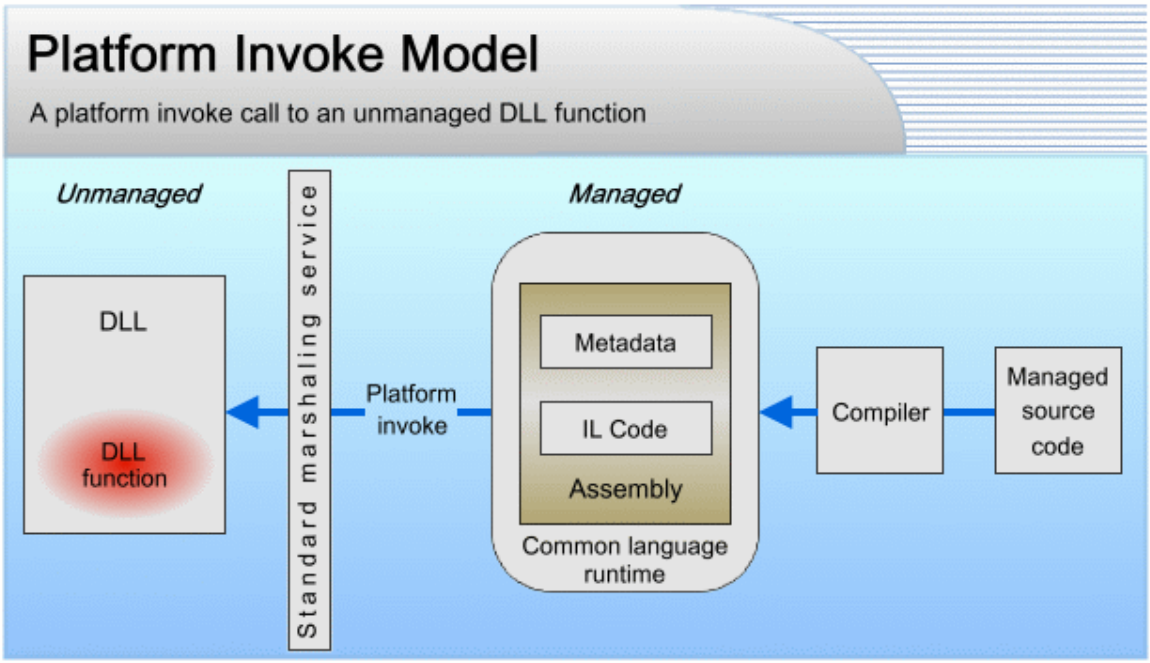


Bild 7.1: Ablauf eines PInvoke Aufrufs

Wie der übergebene Parameter verwendet werden soll, ist dem Entwickler überlassen. Dieser muss das im Code mit dem Attribut MarshalAs festlegen. Leider ist es nicht einfach herauszufinden, wie eine Systembibliothek über .NET-Code richtig aufgerufen wird. Im Internet findet man eine Community, die erfolgreich benutzte Aufrufe von Systemfunktionen dokumentiert. Auf der Seite http://pinvoke.net findet man viele gute Beispiele und Informationen zu diesem Thema. Bei α-PrintPut wird vor allem die WinAPI aufgerufen, da von dieser einige Funktionen im .NET-Framework nicht umgesetzt worden sind. Dies liegt daran, dass α-PrintPut sehr systemnah mit dem arbeiten muss. Des Weiteren muss man bei Nutzung von PInvoke in Visual Studio in den Einstellungen des Projekts unter dem Menü Build „allow unsafe code“ wählen, sonst kann das Projekt nicht gebaut werden.

Die Konvention, dass Systembibliotheken Funktionen exportieren, damit sie von anderen Programmen ausgeführt werden sollen, gilt aber auch andersherum. Will man für Windows eigene Usermode-Treiber, Bibliotheken, oder im Falle von α-PrintPut einen Port Monitor programmieren, so verlangt das Betriebssystem, dass die Datei im PE-Format vorliegt und bestimmte Funktionsrümpfe exportiert bzw. anbietet, die aufgerufen werden. Dies wird zum Beispiel auch bei einem Port Monitor verlangt. Abb. 7.2 zeigt die Funktionsrümpfe eines Port Monitors.

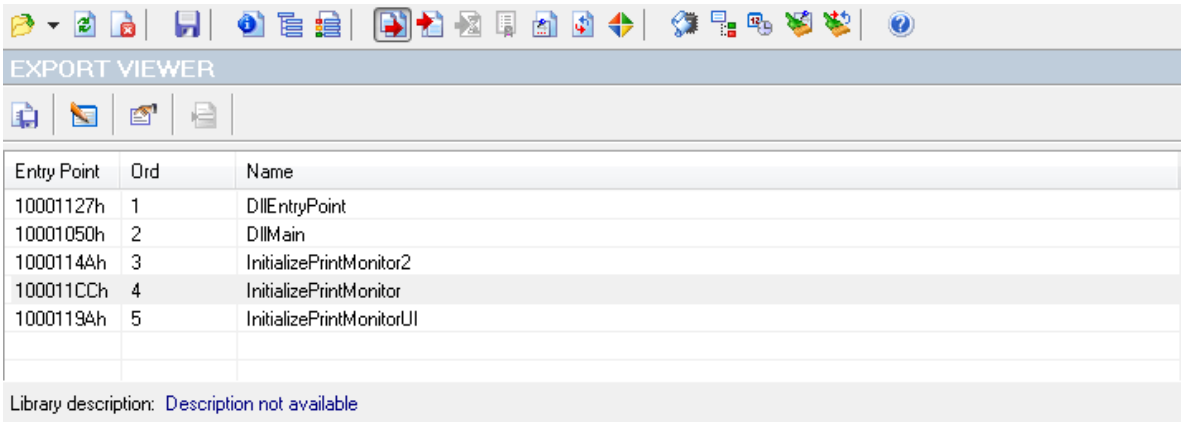


Bild 7.2: Exportierte Funktionsrümpfe

So muss ein Port Monitor einige Funktionen anbieten, wie die InitializePrintMonitor(), die immer nach dem Laden der DLL als allererstes aufgerufen wird. Dies ist jedoch bei den .NET-Assemblies, die ganz anders strukturiert und aufgebaut sind, nicht möglich. Darum gibt es keine Möglichkeit, in .NET einen Druckertreiber oder Port Monitor zu programmieren. Allgemein gibt es keine Möglichkeit Kernel-Mode Treiber in C# zu programmieren, da diese über die .NET Laufzeitumgebung ausgeführt derzeit im User-Mode laufen und dies sich in naher Zukunft nicht ändern wird.

8 Lösungsentwurf

In diesem Kapitel wird der Lösungsentwurf von α-PrintPut dargestellt. Zuerst wird der erkennbare Nutzen von α-PrintPut aufgezeigt. Dann werden anhand eines üblichen Szenarios der Patientenbehandlung die Anforderungen an α-PrintPut deutlich. Diese werden klar ausformuliert und mit Bezug der verwandten Lösungsansätze wird dann ein Lösungsansatz ausgearbeitet.

8.1 Erkennbarer Nutzen von α-PrintPut

α-PrintPut schafft eine formatunabhängige Anbindung beliebiger druckbarer Dateiformate mit dem α-Doc. α-PrintPut macht es möglich, die Daten in dem einheitlichen standardisierten PDF-Format einzubinden. Dies ermöglicht den reibungslosen Austausch von Informationen in übergreifenden Institutionsprozessen und minimiert den Aufwand der Synchronisation der Programme und Systeme der unterschiedlichen Institutionen. Dadurch kann einiges an Kosten für Programmlizenzen eingespart werden, denn der Nutzer (Arzt/Mitarbeiter) muss kein neues Dateiformat erlernen oder zusätzliche Software anschaffen. Er kann seine bewährten Programme weiterhin benutzen und damit kostspielige und zeitintensive Umschulungen umgehen. Sogar Programme, die keine Export-Funktion oder Konvertierungsfunktion für die eigenen Dateiformate unterstützen, können bei vorhandener Druckfunktionalität den Dateiinhalt in beliebige α-Docs integrieren.

Durch die Möglichkeit des unproblematischen Transfers medizinischer Daten profitiert auch der Patient. Denn mögliche Probleme und Verkomplizierungen des Prozesses, wenn die gelieferten Dateien nicht geöffnet werden können und nachbestellt oder gar neu erstellt werden müssen, können umgangen werden. Der Patient wird schneller behandelt und vor doppelten Untersuchungen verschont. Dies wirkt sich auch auf seine Gesundheit aus, denn in der Medizin zählt jede Sekunde und viele unvermeidbare Untersuchungen, wie das Röntgen, sind gesundheitsschädigend und sollten nicht aufgrund Formatinkompatibilität wiederholt werden müssen.

8.2 Anwendungsszenario

Um die Anforderungen an α-PrintPut zu erkennen, wird kurz ein übliches Patientenbehandlungsszenario dargestellt: Dr. Medicus ist Allgemeinarzt und Hausarzt vieler Patienten. Er stellt Diagnosen und leitet die Patienten bei Bedarf weiter an Spezialisten. Dabei fällt immer viel Bürokratie an. Da seine Kollegen seit einiger Zeit bereits auf elektronische Fallakten schwören, beschließt er aus diesem Grund auch in seiner Praxis die α-Docs einzuführen um von den vielen Vorteilen zu nutzen.

In der Praxis gibt es viele medizinische Apparate, die an den Computer angeschlossen sind. Unter anderem ein Ultraschall- und Röntgengerät. Auch viele andere Tätigkeiten, wie die Blutanalyse im hauseigenen Labor werden mit Hilfe von Computern umgesetzt. Die dabei entstehenden elektronischen Dokumente haben verschiedenste Formate. Als Betriebssystem werden in der Praxis unterschiedliche Versionen des Windows-Betriebssystems eingesetzt.

Dr. Medicus befürchtet, dass seine Kollegen diese Dokumente nicht lesen könnten und war deshalb gegenüber elektronischem Dateitransfer immer sehr skeptisch. Dies liegt auch daran, dass viele seiner Kollegen sich mit Computern nicht besonders gut auskennen. Aber mit α-PrintPut und den α-Docs hat er eine gute Lösung für das Problem gefunden.

8.2.1 Installation und Deinstallation von α-PrintPut

Die Installation von α-PrintPut gestaltet sich sehr einfach. Nur die Wahl des Programmverzeichnisses und eine Bestätigung, es ist keinerlei komplizierter Eingriff nötig und der Installationsdialog ist bei allen Systemen identisch. Am Schluss wird er noch gefragt, ob α-PrintPut als Standarddrucker installiert werden soll. Da Dr. Medicus noch andere Drucker benutzt, verneint er. Sofort merkt Dr. Medicus, dass in allen Programmen im Druckdialog der neue virtuelle Drucker erscheint. Sogar bei anderen Benutzern am gleichen Rechner kann dieser benutzt werden und muss nicht extra für jeden Benutzer installiert werden. Das freut Dr. Medicus, denn die anderen Angestellten in der Praxis benutzen oft USB-Sticks am Computer zum Transport von Daten und haben aus Sicherheitsgründen keine Rechte, Manipulationen am System durchzuführen. Die Deinstallation gestaltet sich genauso einfach und entfernt restlos alle Dateien und Einstellungen. Davon ist Dr. Medicus ist begeistert, denn er ist kein wirklicher Experte am Computer und fürchtet komplexe Einstellungen und ist froh, wenn ihm die Arbeit abgenommen wird.

8.2.2 Benutzung von α-PrintPut zur Integration von beliebigen Dokumenten

Kurz erklärt Dr. Medicus seinen Angestellten, wie sie α-PrintPut benutzen sollen. Durch die Installation des virtuellen Druckers ist der PrintPut-Drucker für alle Programme über den Druckdialog wählbar. Die Mitarbeiter arbeiten wie üblich mit ihren Programmen und gehen, wenn ein Dokument angefertigt ist und in die elektronische Fallakte eingefügt werden soll, in den Druckdialog, wählen PrintPut als Drucker und starten den Druckauftrag. Schon nach kurzer Zeit erscheint ein Dialog, der den Nutzer nach Informationen über die Datei erfragen tut, die in die Fallakte eingefügt werden soll. Manche Felder wie den Namen der Fallakte oder den Benutzernamen hat der Dialog bereits vorgewählt. Der Nutzer kann dies noch schnell nach seinen Bedürfnissen anpassen und ergänzt noch fehlende Informationen, wie die Wahl der Fallakte, in welche die konvertierte PDF-Datei als α-Card-Payload eingefügt werden soll. Er hat auch für jedes Feld die Möglichkeit, den gewählten Eintrag als Standard zu speichern, so dass dieser bei der nächsten Nutzung bereits vorgewählt ist und der Nutzer somit nur noch bestätigen muss. Nach der Bestätigung dauert es kurz und der Benutzer wird über den Status des Vorgangs, also ob das Eintragen erfolgreich war, mit einer Meldung informiert. Damit ist auch schon die Bearbeitung der Fallakte abgeschlossen und sie kann an den nächsten zuständigen Arzt übertragen werden.

8.2.3 Setzen der Einstellungen von α-PrintPut

Dr. Medicus will das Setzen der Standardeinstellungen machen können, ohne einen Druckauftrag anzustoßen. Dafür ruft er die „PrintPut Settings“-Verknüpfung im Startmenü auf. Diese Einstellungen werden benutzerkontenspezifisch gespeichert, so dass jeder seine persönlich oft benutzten Einstellungen behält.

An diesem Szenario sieht man, wie α-PrintPut in einer Praxis benutzt wird, um beliebige Dokumente auf einfache Weise in die elektronische Fallakte übernehmen kann. Selbstverständlich war dies nur ein Beispielszenario, das sich erweitern lässt. Aber es werden bereits jede Menge Anforderungen an α-PrintPut deutlich erkennbar.

8.3 Anforderungen an α-PrintPut

Bevor α-PrintPut umgesetzt werden kann, müssen die Anforderungen, also Bedingungen oder Fähigkeiten, die es erfüllen muss, spezifiziert werden. Anforderungen können

unterschiedlichste Ursachen haben. Neben den Nutzern des Systems spielt auch die Systemumgebung eine große Rolle, welche das Softwaresystem erfüllen muss.

Anforderungen sind sehr wichtig, da fehlende oder nicht hinreichend beschriebene Anforderungen zu gravierenden Fehlern in der Umsetzung führen können. So wird das Risiko, dass α-PrintPut dem Kunden nicht nützt oder gefällt, minimiert. Da es in diesem Fall um eine innovative Idee geht und keine Möglichkeit besteht, den Endkunden zu befragen, wird von dem beschriebenen Szenario ausgegangen, um herauszufinden welche Anforderungen der Nutzer an α-PrintPut stellen würde.

Die Anforderungen an α-PrintPut unterteilen sich in funktionale und nicht-funktionale Anforderungen. Die funktionalen Anforderungen beschreiben die Arbeitsweise und das Verhalten, also was α-PrintPut tun bzw. können muss. Zu den nichtfunktionalen Anforderungen gehören die qualitativen und systembezogenen Anforderungen an das Produkt.

Funktionale Anforderungen an α-PrintPut

Bereitstellung eines Installers - α-PrintPut muss einen Installer bereitstellen, der sich um die Installation und Konfiguration des PrintPut-Druckers, Monitors und aller relevanten Aspekte kümmert. Der Installer muss dem Benutzer die Möglichkeit bieten, PrintPut als Standarddrucker zu installieren. Außerdem muss dieser eine Deinstallationsroutine besitzen, die alle Spuren von α-PrintPut auf dem System beseitigt. Die Deinstallation soll über den üblichen Windows-Software-Dialog aufgerufen werden können.

Integration in das System - α-PrintPut wird als spezieller Druckertreiber in das System integriert. Er legt auch verschiedene Startmenüeinträge für alle Benutzer an, so dass jeder benutzerspezifische Einstellungen an α-PrintPut vornehmen kann. α-PrintPut muss von allen Benutzern des Systems aufgerufen werden können. Dafür muss der PrintPut-Drucker für alle Benutzer sichtbar und in allen Programmen im Druckdialog wählbar sein.

Mehrbenutzer- und Parallelbetrieb - α-PrintPut muss von mehreren Benutzern parallel verwendet werden können. Sowohl über mehrere lokal angemeldete Benutzer und dem Benutzerwechsel ohne Abmeldung, als auch über Fernzugriff, sollen Benutzer mit α-PrintPut arbeiten können. Außerdem muss jeder Nutzer mehrere Druckaufträge parallel starten können.

Anpassbarkeit des Konvertierungs- und Integrationsvorgangs - α-PrintPut muss dem Benutzer in einem Dialog die Möglichkeit anbieten, das α-Doc zu wählen

und Einstellungen vorzunehmen. Der Benutzer soll den Konvertierungs- und Integrationsvorgang anpassen können, um seinen Bedürfnissen gerecht zu werden. Außerdem muss die Möglichkeit gegeben sein, die gesetzten Einstellungen merken zu lassen, damit sie beim nächsten Aufruf wieder vorgewählt zur Verfügung stehen.

Änderung der gespeicherten Einstellungen - Die gespeicherten Einstellungen müssen benutzerspezifisch hinterlegt werden, so dass jeder Benutzer persönlich gewählte Vorauswahlen festsetzen kann. Die Änderung dieser gespeicherten Einstellungen soll auch ohne Aufruf des Druckvorgangs jederzeit vorgenommen werden können.

Umsetzung der Funktionalität - α-PrintPut muss das druckende Dokument in das PDF-Format konvertieren und in das α-Doc integrieren können. Die dafür benötigten Informationen werden vom Benutzer bereitgestellt. α-PrintPut muss den Nutzer über das Resultat des Vorgangs informierten und den Vorgang protokollieren.

Nichtfunktionale Anforderungen an α-PrintPut

Qualitative Anforderungen an α-PrintPut - α-PrintPut muss intuitiv einfach bedienbar sein. Unnötige komplexe Funktionalitäten sollten vor dem Benutzer versteckt werden. Die Bedienung soll sich an dem üblichen Windows Look and Feel und dem α-Editor orientieren.

α-PrintPut muss schnell auf Aktionen reagieren. Lange Wartezeiten und hängende Programme sind zu vermeiden. Die Konvertierung kann bei größeren Dokumenten länger dauern. Dennoch sollte das Programm für den Benutzer ansprechbar sein.

α-PrintPut muss zuverlässig arbeiten. α-PrintPut darf nicht unkontrolliert abstürzen! Der Fehler muss zumindest angezeigt oder protokolliert werden.

Systembezogene Anforderungen an α-PrintPut - α-PrintPut muss Betriebssysteme ab Windows 2000 bis Windows 7 unterstützen und sowohl auf 32-Bit- als auch auf 64-Bit-Systemen funktionieren.

Wie α-PrintPut diesen Anforderungen genügt und aufgebaut wird folgt im nächsten Kapitel.

8.4 Konzeptionelle Lösung

In diesem Kapitel werden die einzelnen Komponenten vorgestellt und ihre Funktionsweise beschrieben. Anschließend wird das Zusammenspiel der Komponenten anschaulich dargestellt.

8.4.1 α-PrintPut-Komponenten

Die Umsetzung von α-PrintPut besteht grob aus mehreren Teilen. Dem Installationspaket, der PrintPut-Komponente, die nach dem Betätigen des Druckvorgangs aufgerufen wird, dem PrintPut-Dialog zur Kommunikation mit dem Benutzer, dem Konverter für die Konvertierung der Postscript-Datei in PDF und dem α-Injector mit welchem die PDF-Datei in das α-Doc integriert wird.

8.4.1.1 Installationspaket

Das Installationspaket installiert und konfiguriert den geforderten Drucker inklusive Treiber und Monitor vor. Sowohl die Installation als auch die Deinstallation der Druckerkomponenten muss in einer bestimmten Reihenfolge passieren, da die Druckerinstallation und Konfiguration auf einander aufbaut. Zuerst werden alle benötigten Daten in das Installationsverzeichnis extrahiert. Dann wird als erstes der Print Monitor installiert. Nun wird ein neuer Port des installierten Print Monitor-Typs, identifiziert durch seinen einmaligen Namen, hinzugefügt. Dieser muss einmalig sein, da im System keine zwei Druckerports gleichen Namens erlaubt sind. Anschließend wird ein postscriptfähiger Druckertreiber installiert. Der Druckertreiber muss für PostScript-Drucker kompatibel sein, damit der generierte Datenstrom für im Postscript-Format verläuft. Schließlich wird der PrintPut-Drucker hinzugefügt, der mit dem installiertem Port verbunden ist, den postscriptfähigen Druckertreiber und den Standard-Windows-Druckprozessor verwendet und als Datentyp das RAW-Format erwartet. Dann wird der Benutzer noch gefragt, ob er den neuen Drucker als Standarddrucker eingerichtet werden soll und abschließend werden die Komponenten noch für die Nutzung mit PrintPut konfiguriert. Bis auf die Pfadangabe und der Wahl der Konfiguration des Standardmonitors sollte die Installation vollautomatisch ohne Benutzereingabe ablaufen.

Die Deinstallation der Druckerkomponenten läuft im Grunde genau in umgekehrter Reihenfolge ab, da in Benutzung verwendete Komponenten nicht entfernt werden dürfen, muss zuerst der Drucker entfernt werden, der die anderen Komponenten benutzt. Jetzt kann der Druckertreiber vom System entfernt und der Port gelöscht werden. Gibt es keinen Port des Print Monitors in Benutzung, so kann auch dieser entfernt werden. Schließlich werden die benutzten Verzeichnisstrukturen und benutzte, nicht mehr benötigte Registry-Einträge von α-PrintPut entfernt. Die Deinstallation verläuft komplett automatisch ohne Interaktion mit dem Benutzer.

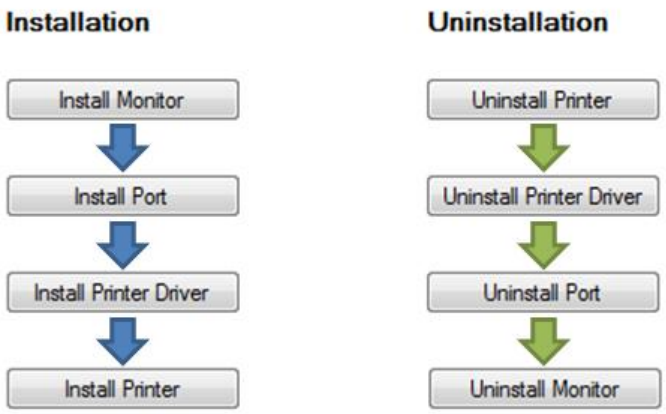


Bild 8.1: Installationsreihenfolge der Druckerkomponenten

8.4.1.2 PrintPut

Durch die Installation und Konfiguration der Druckerkomponenten hat man nun erreicht, dass der PrintPut-Drucker in allen Programmen zur Verfügung gestellt wird. Wenn nun ein Druckauftrag eines beliebigen Programms ausgelöst wird, so wird mit Hilfe des Port Monitors der Datenstrom an PrintPut geleitet. Dieser Vorgang wird in den Abbildungen 8.2 und 8.3 dargestellt. Dies geschieht so, dass der PrintPut-Prozess aufgerufen wird und in seinen Standardeingabekanal der Datenstrom direkt geschrieben wird. Dafür benötigt der Port Monitor jedoch die Information, wo sich der PrintPut-Prozess befindet. Diese Information wird bei der Installation an einem bestimmten Ort abgelegt. Da der Port Monitor die Verbindungsstelle zwischen den User-Mode-Komponenten der Spooler-Architektur und den Kernel-Mode-Gerätetreiber darstellen wird PrintPut vorerst im Kernel Mode in der Session 0 unter dem System-Benutzer gestartet. Aus diesem heraus kann er bei neueren Betriebssystemen mit der Einführung der Benutzerkontensteuerung aufgrund der Session 0 – Isolation nicht direkt mit dem Benutzer kommunizieren. Was sich unter der Benutzerkontensteuerung verbirgt und welchen Einfluss sie auf die Entwicklung und Funktionsweise von α-PrintPut hatte, wird im gleichnamigen Unterkapitel von „Relevante Aspekte der Lösungsumsetzung“ erörtert.

Auch wegen der Tatsache, dass an einem System mehrere Benutzer gleichzeitig aktiv sein können, benötigt PrintPut eine Möglichkeit, herauszufinden welcher Benutzer den Auftrag aufgegeben hat. Deshalb muss der Port Monitor PrintPut diese Information mitliefern. Kennt PrintPut den Benutzer, so kann er diesem den PrintPut-Dialog einblenden, in welchem der Benutzer noch einzelne Informationen und Einstellungen ergänzen kann, bevor die Konvertierung und Integration in das α-Doc stattfinden.

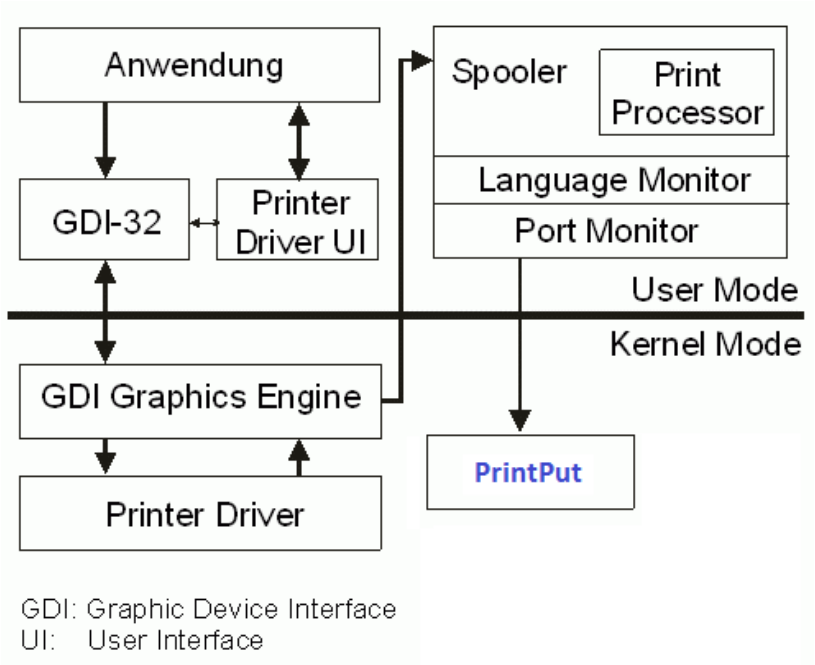


Bild 8.2: Umleitung der Druckdaten zur PrintPut-Komponente

8.4.1.3 PrintPut-Dialog

Der PrintPut-Dialog dient zur Kommunikation mit dem Benutzer. Er stellt die Benutzerschnittstelle zu PrintPut dar. In diesem Dialog ergänzt der Benutzer noch fehlende Informationen, wie die Wahl des α-Docs, in welches die erzeugte End-Datei kommt. Einzelne Informationen werden im Dialog bereits vorgewählt. Diese Informationen holt er sich zum einem aus benutzerspezifischen Daten. Zum anderen werden ihm die Daten beim Aufruf durch PrintPut mitgegeben. Der Benutzer ergänzt die notwendigen Informationen und bestätigt den Vorgang. Im Dialog kann der Nutzer auch für die meisten Felder den gewählten Wert als Standard festlegen. Dabei werden die Daten in einen benutzerspezifischen Ort gespeichert und beim nächsten Aufruf vom Print-Dialog vorbelegt.

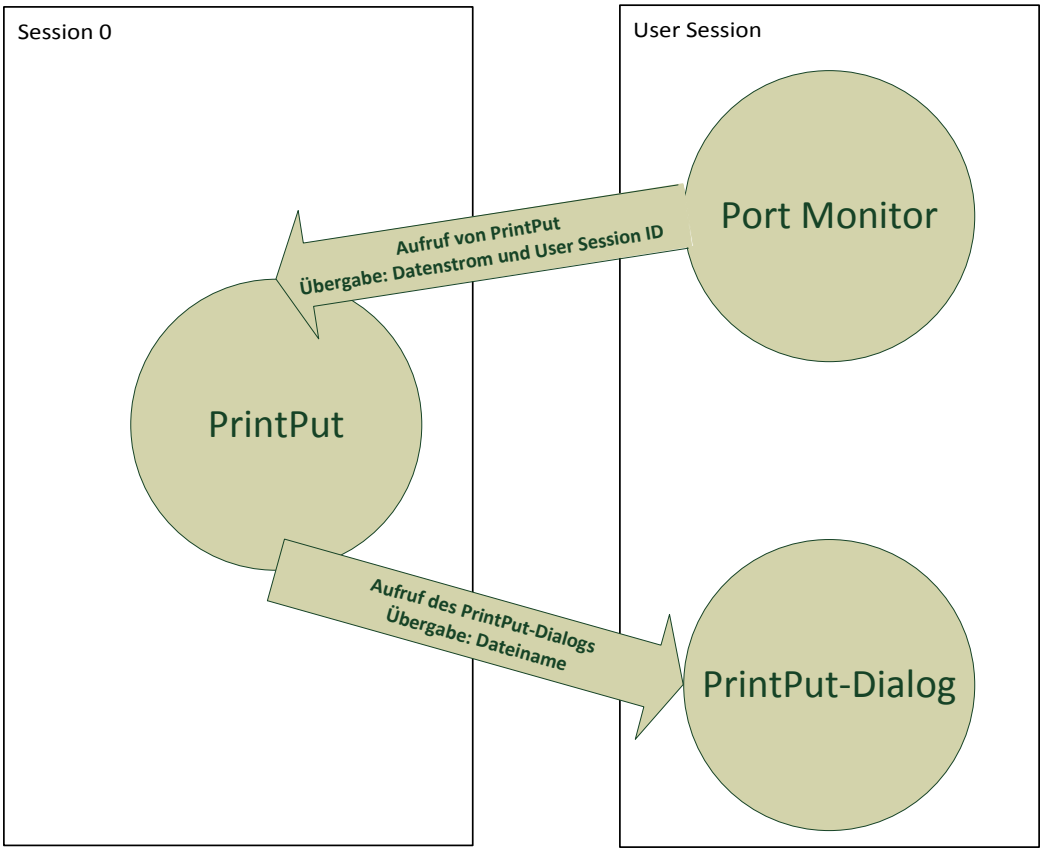


Bild 8.3: Ablauf zum Aufruf des PrintPut-Dialogs beim Benutzer

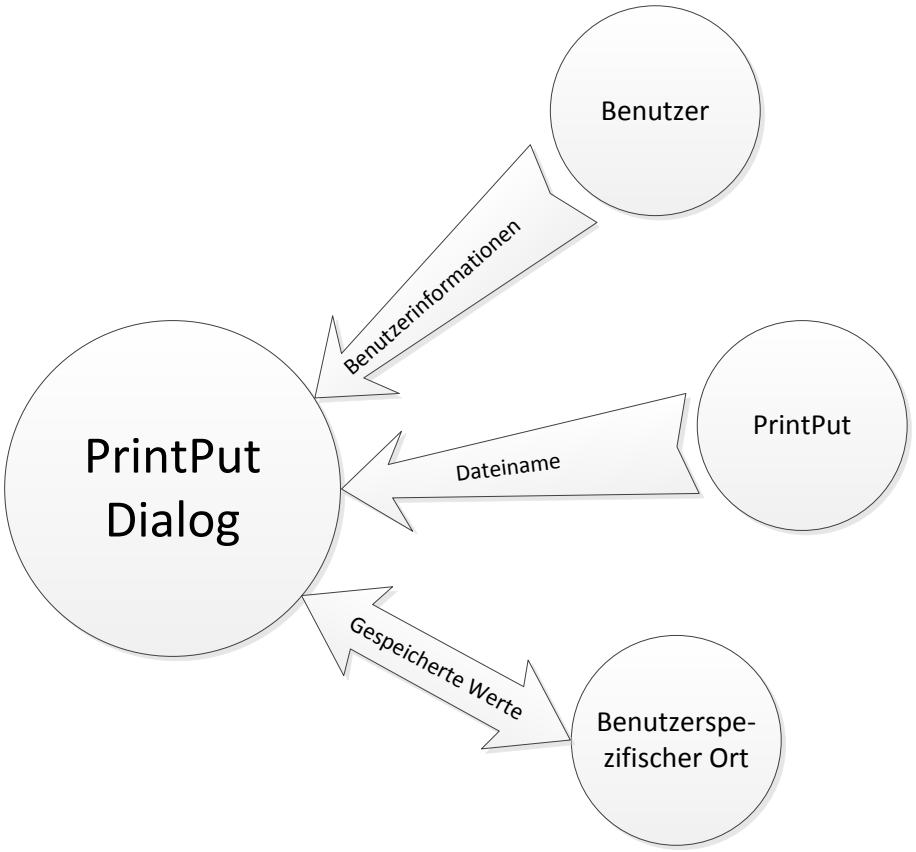


Bild 8.4: Datenflüsse beim PrintPut-Dialog

8.4.1.4 Konverter

Nach Abschluss der Wahl der Einstellungen, wird der Konverter aufgerufen, um die Postscript-Daten in das geforderte PDF-Format zu konvertieren. Hierfür werden dem Konverter die Postscript-Daten und die vom Benutzer gewählten Einstellungen aus dem PrintPut-Dialog übergeben. Der Konverter generiert dann daraus die PDF-Daten. Schließlich werden diese PDF-Daten dem α-Injector übergeben.

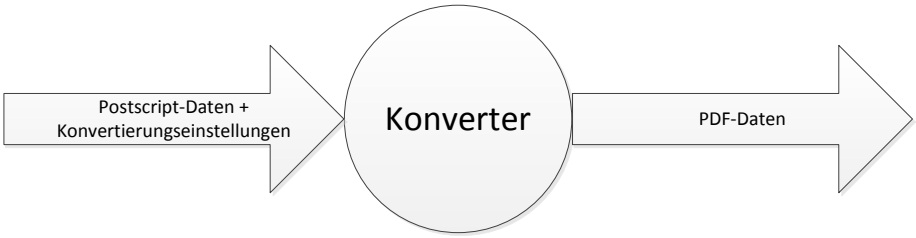


Bild 8.5: Ablauf der Konvertierung

8.4.1.5 α-Injector

Der α-Injector ist an sich kein Baustein von α-PrintPut. Es ist das Bindeglied zwischen α-PrintPut und dem α-Doc. Dieses sorgt für die Integration der übergebenen PDF-Datei in das α-Doc. Dabei wird eine neue α-Card erstellt und die ankommenden PDF-Daten als Payload eingetragen. Die noch benötigten Informationen, wie Alpha-Card Title, Actor ID usw. zum Erzeugen der α-Card werden vom Benutzer beigesteuert. Wenn die PDF-Daten beim α-Injector angekommen sind, ist die Arbeit von α-PrintPut abgeschlossen.

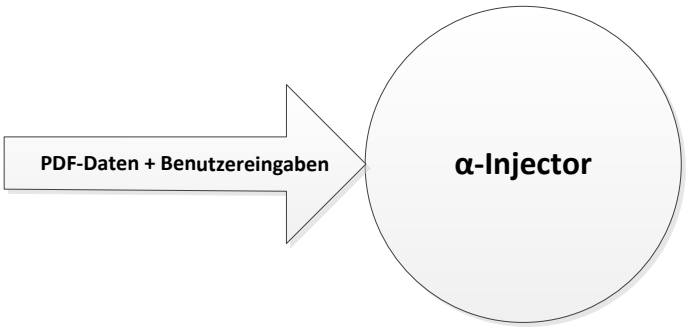


Bild 8.6: Aufruf des α-Injector

8.4.2 Zusammenspiel der Komponenten

Die PrintPut-Komponenten greifen und wirken ineinander ein. Der Installer stellt die Grundlage für α-PrintPut bereit. Er stellt die Komponenten bereit und bereitet das System für die Nutzung von α-PrintPut vor. Die Installer-Komponente ist von den anderen Komponenten entkoppelt und führt die Arbeit für sich alleine aus. Die anderen Komponenten aber greifen stark ineinander. Dies ist wichtig zu verstehen, da so klar wird, wie der Datenstrom generiert, konvertiert und schließlich in das α-Doc integriert wird. Für das bessere Verständnis wird das Zusammenspiel der Komponenten in einer Grafik veranschaulicht.

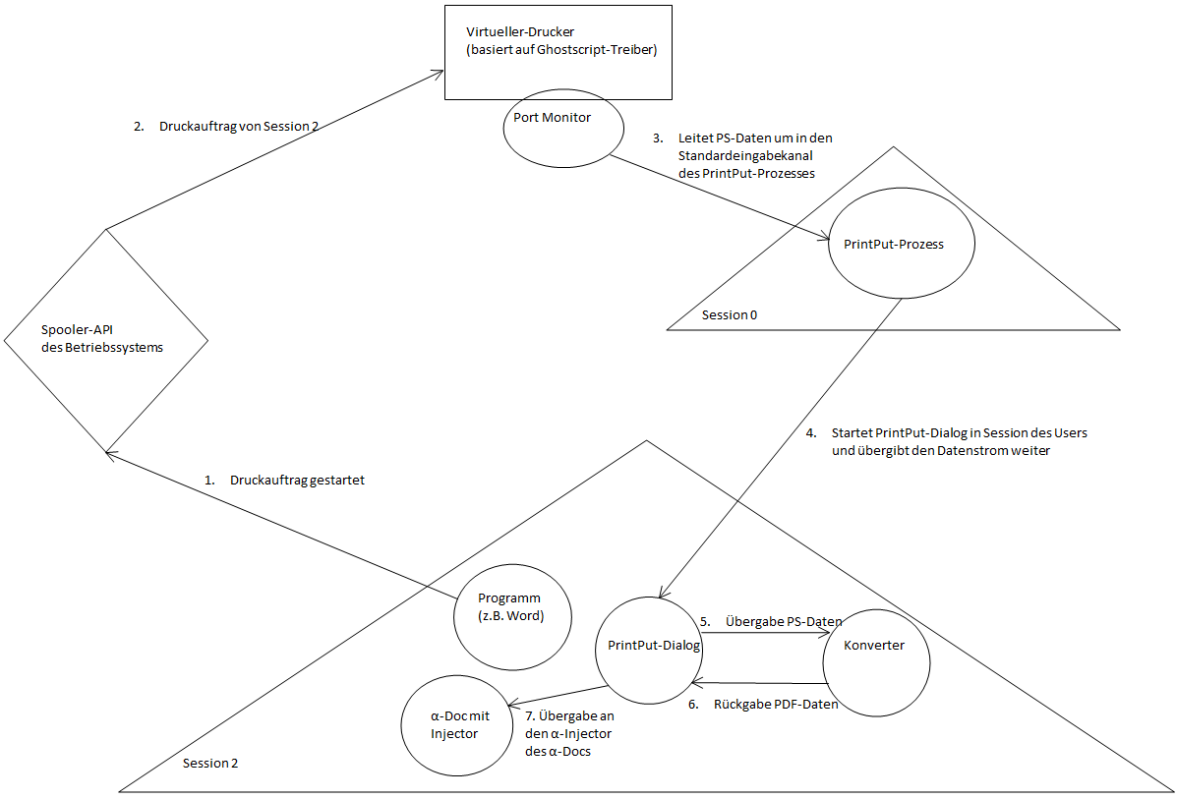


Bild 8.7: Zusammenspiel der Komponenten in zeitlicher Reihenfolge

Wenn ein Benutzer in seinem Programm im Druckdialog den virtuellen PrintPut-Drucker wählt und den Druckauftrag startet, wird dieser Auftrag über die GDI an dem Spooler des Betriebssystems geleitet. Nach der Konvertierung des Datenstroms in das Postscript-Format gelangt der Datenstrom schließlich beim Port Monitor. Dieser startet nun den PrintPut-Prozess und übergibt ihm über den Standardeingabekanal die Postscript-Daten. Der PrintPut-Prozess, der in der Session 0 läuft kann nicht ohne

weiteres mit dem Benutzer kommunizieren und startet deswegen den PrintPut-Dialog-Prozess in der Benutzersession und übergibt diesem den Postscript-Datenstrom weiter. Der PrintPut-Dialog kommuniziert mit dem Benutzer, nimmt Konfigurationseinstellungen und benötigte Informationen entgegen und lässt dann den Postscript-Datenstrom in einen PDF-Datenstrom mit Hilfe des Konverters konvertieren. Anschließend übergibt er die generierten PDF-Daten an den α-Injector des vom Benutzer vorhin gewählten α-Docs.

Nachdem nun der Lösungsansatz ausgearbeitet wurde, wird nun die prototypische Lösung von α-PrintPut basierend auf diesem Ansatz vorgestellt.

9 Prototypische Umsetzung von α-PrintPut

Nachdem nun der allgemeine Lösungsentwurf ausgearbeitet wurde, wird aus dieser Basis eine prototypische Umsetzung geschaffen. Dafür werden zunächst die relevanten Aspekte der Lösungsumsetzung beschrieben und anschließend der prototypische Aufbau von α-PrintPut präsentiert.

9.1 Relevante Aspekte der Lösungsumsetzung

Zwei besondere Aspekte haben auf die Entwicklung von α-PrintPut eingewirkt, die nun vorgestellt werden. Zum einem die Benutzerkontensteuerung und zum anderen die Installationsroutine.

9.1.1 Windows Benutzerkontensteuerung (User Account Control)

Mit Windows Vista und Windows Server 2008 wurde die Benutzerkontensteuerung (UAC) eingeführt, die auch in Windows 7 Bestandteil des Systems und standardmäßig aktiviert ist [MHB10, Rus07, Sel08]. Dieses Feature schränkt die Rechte von Benutzerkonten ein und bietet darüber hinaus noch weitere Schutzaspekte, welche unautorisierte Änderungen am System oder die andere Benutzer im System beeinflussen, verhindern sollen.

Vor Windows Vista hatte jeder Standard-Benutzer volle Administrativrechte, er konnte beliebige Programme installieren, starten und die Systemkomponenten und Systemeinstellungen somit beliebig manipulieren. Dies hatte aber sicherheitskritische Folgen, denn Schadprogramme nutzten diese Privilegien aus um das System zu infiltrieren und kompromittieren. Mit Einzug von UAC hat ein Standard-Benutzer nur Rechte die meisten Programme auszuführen und benutzerspezifische Änderungen am System vorzunehmen.

Änderungen, die das gesamte System betreffen, sowie Installationen und andere administrativen Aufgaben am System sind ihm jedoch untersagt. Bei der Anmeldung in

das System werden die Anmeldeinformationen überprüft und es wird geschaut, welche Zugriffs-Tokens zur Verfügung stehen. Das Besondere dabei ist, dass sogar ein Administrator-Benutzer zuerst mit den Rechten eines Standard-Benutzers, also mit dem Standard-Benutzer-Token, arbeitet. So laufen auch die Programme, die von einem Benutzerkonto der Benutzergruppe „Administrator“ gestartet werden, erst mal mit eingeschränkten Rechten. Neu erzeugte Prozesse erben das Token des Vaterprozesses und können im laufenden Betrieb nicht ihre Privilegien ändern. Ein Prozess hat immer genau einen Token. Auf Wunsch kann der Administrator-Benutzer aber auch dem Programm mit Hilfe der UAC explizit die administrativen „Privilegien“ gewähren, also das Programm mit dem Administrator-Token starten, um zum Beispiel Änderungen am System vorzunehmen. Einige Programme benötigen zum Starten erweiterte (administrative) Rechte und werden in Windows durch das „Schild“-Symbol dargestellt. Beim Aufruf kommt dann automatisch die Benutzerkontensteuerung zum Vorschein und verlangt eine Bestätigung vom Administrator bzw. Benutzernamen und Passwort eines Administratorkontos bei einem Standardbenutzer. Dieses Zwei-Token-System ermöglicht das sichere Arbeiten und erlaubt die bequeme kurzfristige Erhöhung der Privilegien bei Bedarf (ähnlich Linux und dem sudo-Befehl). Es gibt zwar Möglichkeiten, die Benutzerkontensteuerung zu deaktivieren, dies greift aber stark in die Systemsicherheit ein, beeinflusst damit auch andere Programme und gefährdet schließlich den Schutz des Betriebssystems und ist deshalb keine Option für α-PrintPut.

α-PrintPut benötigt nur für die Installation administrative Privilegien. Nach der Installation kann jeder Standard-Benutzer α-PrintPut aufrufen. Dies wird erreicht, indem in mehreren Bereichen auf die Besonderheiten von UAC eingegangen wird, denn UAC beeinflusst die Entwicklung und Planung eines Programms enorm. UAC hat vor allem in den folgenden Bereichen ausschlaggebend in die Entwicklung von α-PrintPut eingewirkt

Signierung

Um das Programm zu signieren benötigt man ein von autorisierten Zertifizierungsstellen ausgestelltes Zertifikat, welches man gegen Bezahlung und Vorlage des Ausweises beantragen muss. Für den Entwicklungs- und Testbetrieb kann man auch Testzertifikate selbst erstellen und verwenden. Der Unterschied zwischen signierten und unsignierten Programmen macht sich bereits bei der Ausführung im Benutzerkontensteuerungsdialog bemerkbar. Signierte Anwendungen haben ein vertrauenerweckenderes Aussehen. Unsigniert:

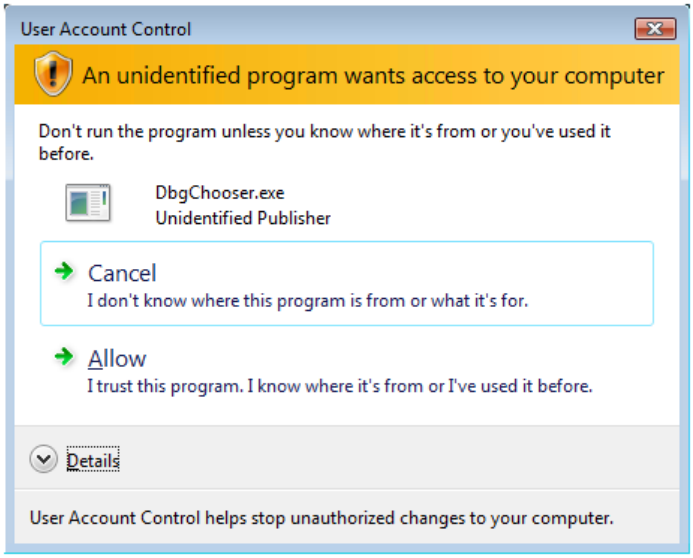


Bild 9.1: Aufruf einer unsignierten Anwendung

Signiert:

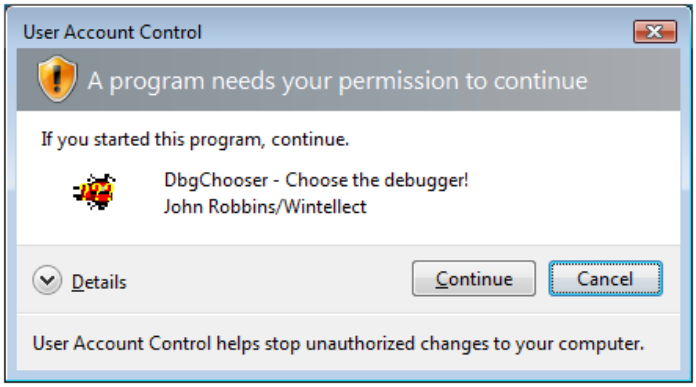


Bild 9.2: Aufruf einer signierten Anwendung

Außerdem gibt es Gerüchte, dass Microsoft in Zukunft nur signierte Anwendungen, die Administratorrechte benötigen, zulässt. Da jedoch für das Projekt keine finanziellen Mittel zur Verfügung stehen und die Signierung die Funktionalität zum heutigen Zeitpunkt nicht beeinflusst, wird bei α-PrintPut auf eine Zertifizierung verzichtet! Will man das Projekt in Zukunft aber signieren so findet man im Internet ausführliche Anleitungen dazu.

Virtualisierung bei fehlendem Manifest

Ein Manifest ist eine XML-Datei, die mit einem Programm ausgeliefert, oder direkt in das Programm als Ressource integriert werden kann. Programme ohne ein Manifest laufen wegen UAC im Kompatibilitätsmodus (virtualisiert).

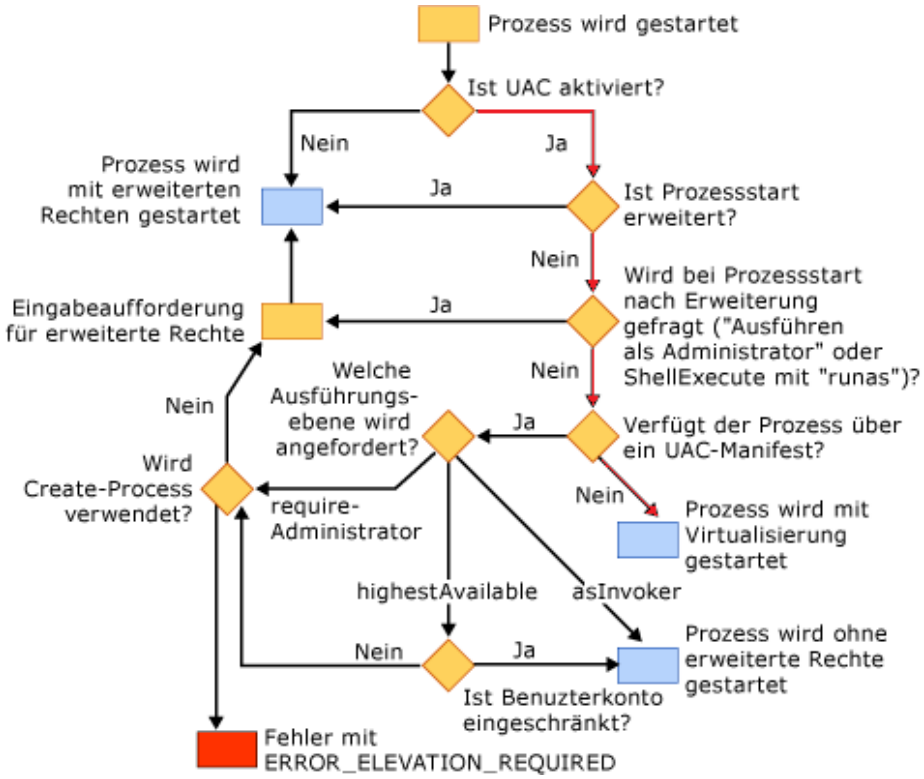


Bild 9.3: Ablaufdiagramm eines Prozessstarts

Das hat Vor- und Nachteile. Der Vorteil ist, dass das Programm keine evaluierten Rechte benötigt und nicht wegen mangelnder Rechte abstürzt. Dies funktioniert so, dass Schreibzugriffe in Programm- oder Systemverzeichnisse und in bestimmte Bereiche der Registry automatisch umgeleitet werden in von der UAC bereitgestellten lokale Datei- und Registry-Pfade, dem so genannten Virtual Store. Die Folgen sind aber gravierend: Für das Programm sieht alles in Ordnung aus und es denkt, dass es globale Daten manipuliert, jedoch werden in Wirklichkeit nur lokale Änderungen an von der UAC bereitgestellten Ordnern und Registry gemacht. Ein weiteres Problem ist, dass 64-Bit Anwendungen nicht automatisch auf den Virtual Store zugreifen können. Außerdem wird diese temporäre Lösung der Umleitung in zukünftigen Windows-Versionen wahrscheinlich abgeschafft.

Darum ist es unerlässlich, dass α-PrintPut ein Manifest bekommt. Visual Studio 2008 und neuer tun standardmäßig ein Manifest für das Programm erzeugen, welches den Token des Vaterprogramms übernimmt.

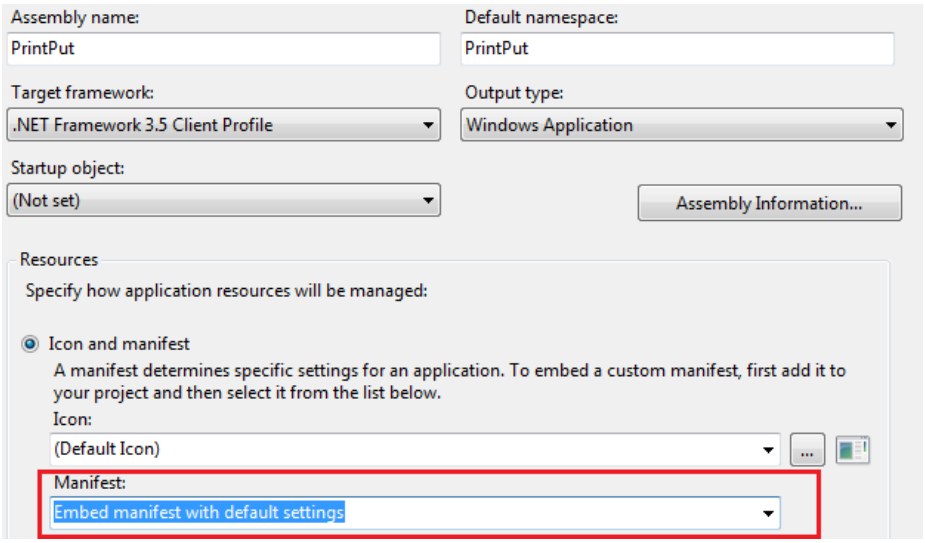


Bild 9.4: Einbindung des Manifests in Visual Studio

Dies wird im Manifest mit „asInvoker“ beschrieben und bei α-PrintPut auch benutzt.

```
<?xml version="1.0" encoding="utf-8"?>
<asmv1:assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1" xmlns:asmv1=
  <assemblyIdentity version="1.0.0.0" name="MyApplication.app"/>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
    <security>
      <requestedPrivileges xmlns="urn:schemas-microsoft-com:asm.v3">
        <!-- UAC Manifest Options
            If you want to change the Windows User Account Control level replace the
            requestedExecutionLevel node with one of the following.

        <requestedExecutionLevel  level="asInvoker" uiAccess="false" />
        <requestedExecutionLevel  level="requireAdministrator" uiAccess="false" />
        <requestedExecutionLevel  level="highestAvailable" uiAccess="false" />

            If you want to utilize File and Registry Virtualization for backward
            compatibility then delete the requestedExecutionLevel node.
        -->
        <requestedExecutionLevel level="asInvoker" uiAccess="false" />
      </requestedPrivileges>
    </security>
  </trustInfo>
</asmv1:assembly>
```

Bild 9.5: Setzen der richtigen Einstellung im Manifest

Da α-PrintPut nicht nur von Administratoren benutzt wird, sondern hauptsächlich von Benutzern ohne erweiterte Rechte (administrative Privilegien) ausgeführt wird, wür-

de im letzteren Fall der Versuch, geschützte Systemdateien oder Pfade zu verändern, den Fehler „Zugriff verweigert“ auslösen. Deshalb wurde bei der Entwicklung beachtet, benutzerspezifische Daten, wie gewählte Einstellungen in α-PrintPut in benutzerspezifischen Pfaden abzulegen. In der Registry werden die α-PrintPut-Einstelleungen unter „HKEY_CURRENT_USER\SOFTWARE\PrintPut“ und im Dateisystem unter einem Unterordner des benutzerspezifischen „Application Data“-Ordner, den man über die Umgebungsvariable %AppData% abrufen kann, abgelegt. Globale Dateien, welche die Benutzer teilen werden in einem vom Betriebssystem extra dafür speziell eingerichteten Verzeichnis gespeichert. Dafür standen zwei Möglichkeiten zur Wahl. Den „Public“- und den „CommonProgramFiles“-Ordner, die je nach System woanders liegen und man diese über die Umgebungsvariablen %Public% und %CommonProgramFiles% abrufen kann. Für die globalen Daten von α-PrintPut wird ein Unterordner „PrintPut“ im „CommonProgramFiles“-Ordner angelegt und verwendet, da der „Public“-Ordner auch oft für Netzwerkfreigaben benutzt wird und keine Person aus dem Netzwerk oder gar aus dem Internet Zugriff auf Daten von α-PrintPut haben sollte.

Session 0 Isolation

In Windows laufen unterschiedliche Windows Sessions (oft auch Sitzungen genannt) und erlauben es mehrere Benutzer an einem Betriebssystem zu arbeiten. In Windows XP liefen die Systemdienste, Systemprogramme, Systemtreiber und Benutzerprogramme nach der ersten Anmeldung zusammen in der Session 0.

Ab Windows Vista laufen aus Sicherheitsgründen nur noch die Systemdienste, Systemprogramme und Systemtreiber in dieser Session und jeder anmeldende Benutzer bekommt eine neue eigene Session mit einer höheren Nummer (Session 1, 2, usw.). Dies gilt auch für Anmeldungen über z.B. Terminal Services (Remote Desktop).

Durch diese Trennung können die Dienste, die auf der Session 0 laufen nicht mehr mit dem Benutzer interaktiv (z.B. über eine grafische Benutzerschnittstelle) kommunizieren, sprich der Service kann einem angemeldeten Benutzer keine Nachricht (z.B. in Form eines Dialogs) direkt schicken. Stattdessen öffnet sich dabei das „Interactive services dialog detection“-Fenster, mit dessen Hilfe man sehr unbequem das vom Service an den Benutzer geschickte Dialog anzeigen kann. Dabei wird der laufende Desktop des Benutzers umgemapt auf den Session-0-Desktop. Dort wird dann mit dem Dialog gearbeitet und kehrt anschließend wieder zurück zum eigenen Desktop. Auch da gibt es Gerüchte, dass Microsoft diesen „Interactive service dialog detection“ -Dienst in zukünftigen Versionen nicht mehr anbieten wird.

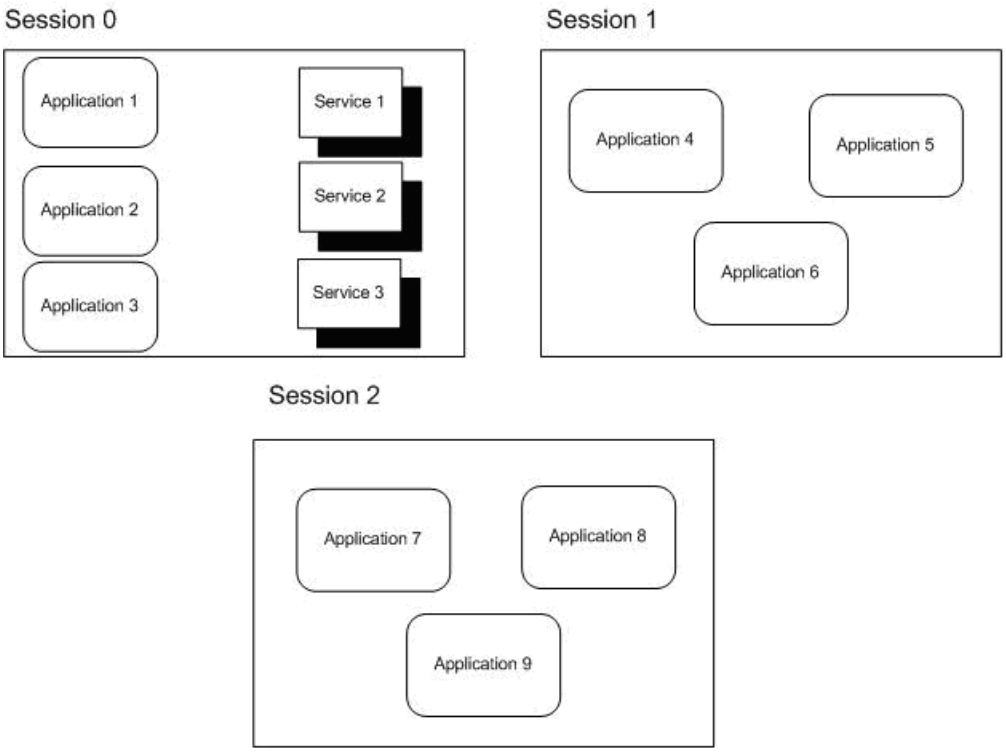


Bild 9.6: Sessions vor Windows Vista

Ebenso wird α-PrintPut durch den Port-Monitor beim Druckvorgang in der Session 0 gestartet und kann erst mal nicht direkt mit dem Benutzer interagieren. Da die Client-Programme von sich aus nicht auf gemeinsame Ressourcen der Programme und Dienste der Session 0 zugreifen können, wird es erschwert die an PrintPut geschickte PostScript-Datei dem Client bereitzustellen. Um Daten zu übergeben muss man auf andere Mechanismen zurückgreifen, wie zum Beispiel Übergabe über den Standardeingabekanal oder mit Hilfe von Named Pipes. Bei letzterem muss man dann aber auch explizit Zugriffsrechte (Access Control List (ACL)) setzen, damit der Client auf das Pipe zugreifen darf! Bei PrintPut wird die Übergabe über den Standardeingabekanal durchgeführt und auf die Named Pipes verzichtet, weil diese erst im .NET Framework 3.5 integriert sind und die Nutzung dieser dazu geführt hätte, dass α-PrintPut nicht auf Windows 2000-Systemen lauffähig wäre. Alternativ hätte man die Named Pipes auch direkt über PInvoke ansprechen können, dies ist aber recht kompliziert und sehr fehleranfällig.

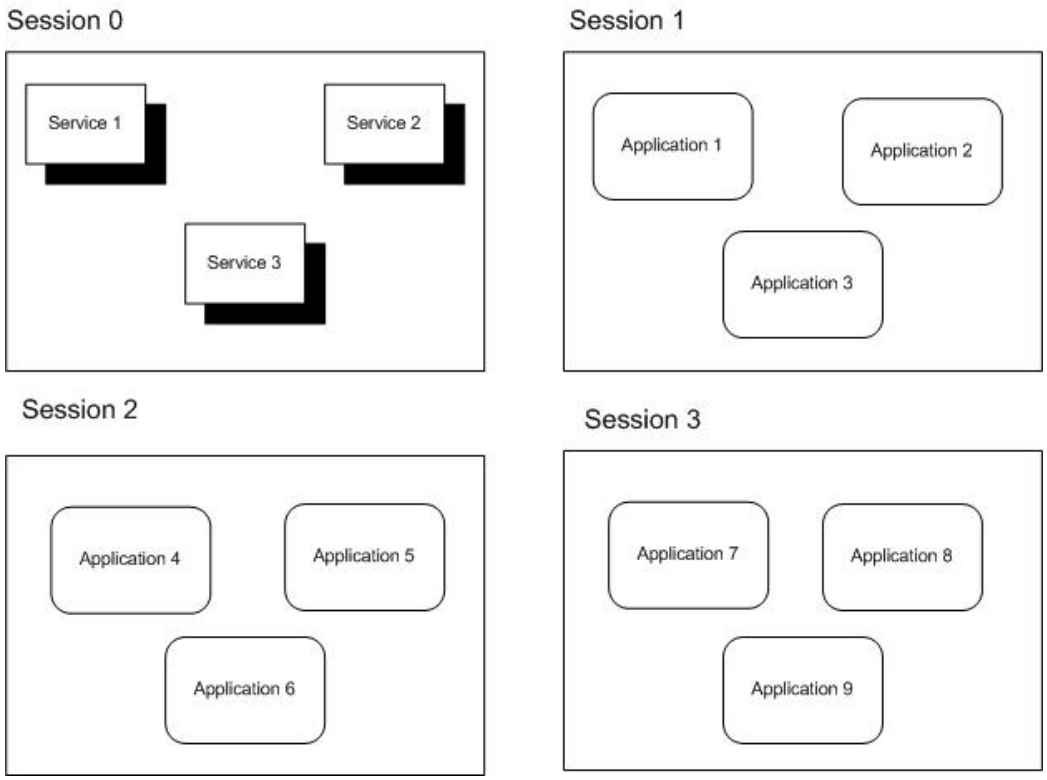


Bild 9.7: Sessions nach Einführung von Windows Vista

Ein weiteres Problem besteht darin, bei mehreren am System angemeldeten Benutzer herauszufinden, wer den Druckauftrag ausgelöst hatte um diesem den PrintPut-Dialog anzuzeigen. Für dieses Problem gibt es mehrere Lösungsmöglichkeiten. Mit Hilfe der Systemfunktion WTSGetActiveConsoleSessionId() kann man die Session des derzeit am Rechner aktiven Benutzers herausfinden. Dies reicht aber bei mehreren gleichzeitig

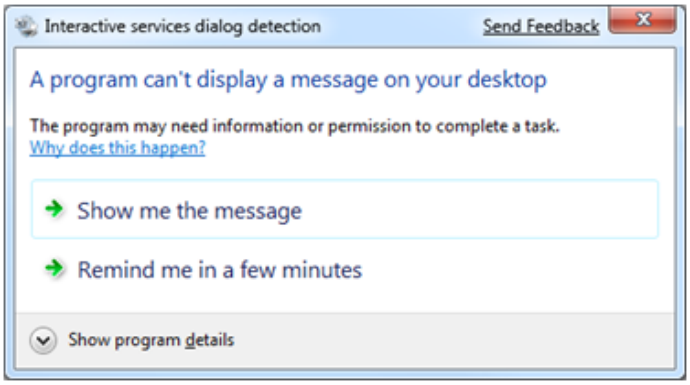


Bild 9.8: Ïnteractive service dialog detection“ -Dialog

angemeldeten Benutzern nicht aus! Die Lösung, die bei α-PrintPut angewandt wird, ist dass der Port-Monitor, der, wie alle Spooler-Komponenten, mit Benutzerprivilegien läuft die Session des Benutzers gibt und diese dem PrintPut-Prozess beim Aufruf mitteilt. Kennt der PrintPut-Prozess nun die Session des Benutzers so kann er nun mit Hilfe der Systemfunktion WTSSendMessage() einen einfachen Dialog beim Nutzer anzeigen lassen, oder aber über CreateProcessAsUser() einen neuen Prozess in der Benutzersession starten. Da WTSSendMessage() zu wenig Freiraum bietet wird bei PrintPut direkt ein neuer Prozess, der PrintPut-Dialog, gestartet.

9.1.2 Installation

Die Installation von α-PrintPut spielt eine nicht zu unterschätzende Rolle, denn die korrekte Installation und Einrichtung der beteiligten Komponenten (Druckertreiber, Port-Monitor, Konverter) und natürlich PrintPut an sich stellt die Grundlage für die ordnungsgemäße Funktionalität sicher.

Als Installer bieten sich viele verschiedene Produkte und Lösungen an. Außerdem wäre das Programmieren eines eigenen Installers möglich. Da jedoch kostenlose etablierte und erprobte Installer zur Wahl zur Verfügung standen, welche die gewünschten Funktionalitäten und Anforderungen erfüllten, ist die Entwicklung eines eigenen Installers nicht erforderlich.

Die Ansprüche an die Installer waren nicht unerheblich. Abgesehen von den einfachen Aufgaben, die aus dem Kopieren der Programmdateien in ein vom Benutzer gewähltes Verzeichnis und setzen der benötigten Registry-Einträge, bestehen, sollte der Installer flexibel genug sein und die Möglichkeiten anbieten, Skripts und oder selbst geschriebenen Code auszuführen, der benötigt wird um den Druckertreiber, den Monitor und den Port auf dem System zu installieren und einzurichten. Außerdem sollte bedacht werden, dass in einem großen Unternehmen bzw. einer großen Organisation, wie einem Krankenhaussystem, festgelegte Verwaltungs- und Organisationsstrukturen etabliert sind mit strengen Vorschriften und Sicherheitsbestimmungen, welche nur bestimmten Rollen bestimmte Rechte in den jeweiligen Abteilungen gewährt, die zu diesem Zeitpunkt nicht bedacht werden können. Dennoch sollte die Installation möglichst einfach für möglichst viele Nutzer der unterschiedlichsten Betriebssysteme gestrickt sein. Man sollte also mit einer Installation möglichst vielen Systemen bzw. Benutzerprofilen α-PrintPut zugänglich machen. Gleichzeitig sollte ein und dieselbe Installation möglichst viele verschiedene Betriebssysteme und Architekturen (32 und 64 Bit) abdecken, jedoch auch die Möglichkeit

bieten feingranular die Nutzung bestimmten Rollen/Benutzern zu gewähren bzw. zu verwehren.

Zur Wahl stehen hierbei zwei unterschiedliche Bereitstellungsstrategien. Zum einen das herkömmliche Setup, das an die Benutzer verteilt wird und an jedem Rechner von einem Administrator installiert werden muss und damit α-PrintPut allen Benutzern dieses Systems zugänglich macht. Die granulare Rechtevergabe kann bzw. muss dann vom Administrator gesetzt werden. Zum anderen gibt es die Möglichkeit einer Veröffentlichung der Anwendung an einem zentralen Speicherort, von welchem die Installation dann durchgeführt werden kann. Hier kann man den granularen Zugriff über Zugriffsmöglichkeiten an einer zentralen Stelle setzen. Dafür empfiehlt Microsoft, Installationen basierend auf der Windows Installer-Technologie oder aber ClickOnce (ab .NET Framework 2.0), zu verwenden. ClickOnce hat etliche Vorteile gegenüber dem Windows Installer, scheidet jedoch aufgrund der fehlenden Möglichkeit, Treiber zu installieren, aus.

Da α-PrintPut aber für den Einsatz in möglichst vielen Einrichtungen eingesetzt zu werden entwickelt wird, wurde die Installation von α-PrintPut als herkömmliches Setup-Projekt festgelegt.

Nach einiger Recherche standen schließlich einige gute Produkte zur Wahl, welche den oben beschriebenen Ansprüchen genügten: Das in Visual Studio 2010 Ultimate integrierte Setup-Projekt, welches ein auf dem „Windows Installer“ basierte Installationsdatei im Microsoft Software Installation (MSI)-Format generiert und die kostenlosen Open-Source-Projekte „Inno Setup“ und Nullsoft Scriptable Install System (NSIS).

Der Vorteil von dem Visual Studio 2010 integriertem Setup-Projekt ist die Tatsache, dass er eine direkte Möglichkeit anbietet, selbstgeschriebenen .Net-Code in den Installations- und Deinstallationsvorgang einzuschleusen. Dies geschieht über so genannte Die Open Source Projekte bieten dafür nur Skripte oder Konstrukte in anderen Sprachen an. Ein weiterer Vorteil ist, dass das Setup-Projekt in der gleichen Solution wie die anderen PrintPut-Projekte liegt und die Erstellung des Setups damit sehr vereinfacht wird. Es braucht nur die Einstellung gesetzt zu werden die aus den anderen Projekten erstellten Assemblys sollen in das Setup-Projekt integriert werden. Schon werden diese bei der Installation in den Installationsordner entpackt.

Aus diesem Grund wird für α-PrintPut der integrierte Windows Installer verwendet.

9.2 Software-Entwurf

In diesem Abschnitt wird das bisher erworbene Wissen und Information in den Software-Entwurf von α-PrintPut einbezogen. Zuerst werden die einzelnen Komponenten entworfen und anschließend in einem Installationspaket zusammengesetzt. Anschließend wird das Ergebnis noch zusammengefasst.

9.2.1 Port Monitor

Der Port Monitor stellt das Anfangsglied in der Kette der PrintPut-Komponenten dar. Er tut ankommende Druckaufträge an den PrintPut-Prozess weiterleiten. Im Kapitel Einschränkungen von .NET wurde geklärt, dass in .NET keine Möglichkeit existiert einen Port-Monitor umzusetzen. Es gibt zwar Möglichkeiten den Port-Monitor in einer anderen Sprache zu schreiben, die Windows PE-Bibliotheken generiert, jedoch wird bei der Umsetzung von α-PrintPut darauf verzichtet. Stattdessen wird das in den verwandten Lösungsansätzen beschriebene RedMon für diesen Vorgang eingesetzt. Diese erfüllt alle geforderten Funktionen vom Protokollieren des Druckvorgangs bis hin zum Aufruf des PrintPut-Prozesses. Die Übergabe der Session-ID geschieht dabei so, dass die Umgebungsvariablen beim Aufruf des PrintPut-Prozesses erweitert werden um zusätzliche Schlüsselwerte. RedMon gibt so dem startenden Prozess die Möglichkeit aus den Umgebungsvariablen Informationen zum Benutzernamen, dem Dokumenttitel, der Benutzersession und einiger anderer Werte herauszulesen. Der Port Monitor ist die einzige Komponente die auf einem 64-Bit System ebenfalls als 64-Bit-Version vorhanden sein muss. Alle anderen Komponenten von α-PrintPut können auch als 32-Bit Komponenten fungieren. Der Nachteil von RedMon ist, dass die 64-Bit Version keine Möglichkeit anbietet RedMon über einen grafischen Dialog zu konfigurieren. Dies ist für α-PrintPut aber kein Problem, da der PrintPut-Installer alle nötigen Komponenten ohne Benutzereingriff einrichten tut. RedMon wird dabei direkt über die Registry konfiguriert. So wird die Einstellung gesetzt, dass RedMon den ankommenden Datenstrom an den PrintPut-Prozess weiterleiten tut. RedMon erlaubt es von sich aus einen beliebigen Prozess sowohl in der Session 0, als auch in der Session, aus der der Druckauftrag kam zu starten. Letzteres wird durch die Einstellung „Run as User“ eingerichtet und würde die Notwendigkeit des PrintPut-Prozesses beseitigen. Jedoch hat diese Funktion noch einige Schwächen. Zum einen werden die Umgebungsvariablen des Benutzers nicht gesetzt. Dies hat zur Folge dass der Benutzer zwar den PrintPut-Dialog bekommt, aber die benutzerspezifischen Einstellungen würden für den System-Benutzer abgelegt werden. Auch die Protokollierung

mit Log-Dateien würde in einem falschen Verzeichnis ablaufen und wenn der Benutzer zum Beispiel auf den Desktop wechselt um ein α-Doc zu wählen würde er stattdessen auf dem Desktop des Systembenutzers landen und seine Dateien nicht vorfinden können! Zum anderen würde ein Aufruf des Druckvorgangs aus dem Netzwerk zum Absturz des Spooler-Services führen, welcher dann manuell wiedergestartet werden müsste. Beide Probleme sind für den reibungslosen Ablauf von α-PrintPut extrem wichtig, deshalb wird diese Funktion nicht benutzt, sondern von der PrintPut-Komponente selber umgesetzt.

9.2.2 PrintPut

Die PrintPut-Komponente wird von dem Port-Monitor aufgerufen. Sie stellt praktisch den Vermittler zu dem PrintPut-Dialog dar. Es sorgt für den Session-Wechsel in die Benutzersession und kümmert sich darum, dass der übergebene Postscript-Datenstrom an den PrintPut-Dialog weitergeleitet wird.

Nach dem Aufruf liest PrintPut aus dem Standardeingabekanal den übergebenen Datenstrom ein. Anschließend wird aus der Registry der dort hinterlegte Installationspfad von PrintPut ausgelesen. Je nach Betriebssystem (32 oder 64Bit) liegt dieser Pfad woanders. Anschließend wird aus den von RedMon manipulierten Umgebungsvariablen die Session-ID und der Dokumentenname ermittelt. Anschließend wird mit Hilfe der Session-ID durch Aufruf der Systemfunktion WTSQueryUserToken() der Benutzertoken ermittelt, der benötigt wird um einen Prozess in der Session des Benutzers zu starten. Dieser Token steht dabei als Ersatz für die Eingabe des Benutzernamen und Passworts. Da nicht jeder an diesen Token gelangen soll, ist der Aufruf von WTSQueryUserToken() nur dem Systembenutzer genehmigt. Nicht einmal Administratoren können diese Funktion benutzen. Anschließend wird dieser Token mit Hilfe der DuplicateTokenEx()-Systemfunktion zu einem Primärtoken konvertiert. Mit dem Installationspfad aus der Registry, dem PrintPut-Dialog-Dateinamen, der in den Konfigurationsdateien von PrintPut hinterlegt wird und dem Primärtoken wird nun der PrintPut-Dialog-Prozess in der Benutzersession gestartet.

Zuvor werden aber noch die Umgebungsvariablen des Benutzers gesetzt und der Standardeingabekanal umgeleitet, so dass PrintPut direkt in den gestarteten PrintPut-Dialog-Kindsprozess den Postscript-Datenstrom schreiben kann. Anschließend wird der Standardeingabekanal geschlossen und alle offenen Handles geschlossen.

Der Aufbau der PrintPut-Komponente wird in der folgenden Abbildung dargestellt. Die Komponente besteht aus sechs Klassen. Dabei wird die Settings-Klasse für die Konfigurationsdaten verwendet.

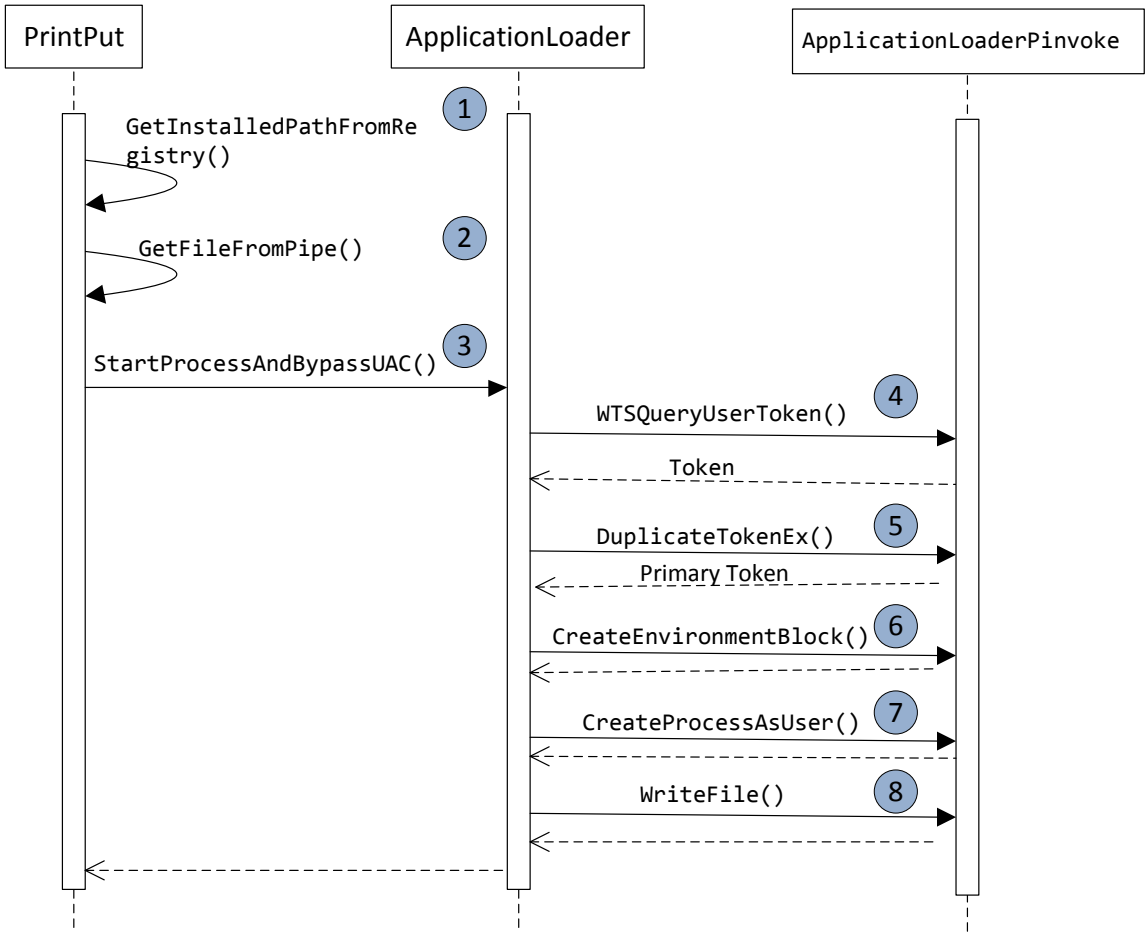


Bild 9.9: Sequenzdiagramm der PrintPut-Komponente

Die Program-Klasse mit der Main-Methode wird als erstes aufgerufen. Diese liest den Standardeingabekanal ein und holt den Installationspfad aus der Registry.

Die EnvironmentCheck-Klasse wird benutzt um herauszufinden, welche Windows-Version gerade läuft. Dies geschieht, indem man den Wert des öffentlichen Attributs Is64BitOperatingSystem abfragen tut.

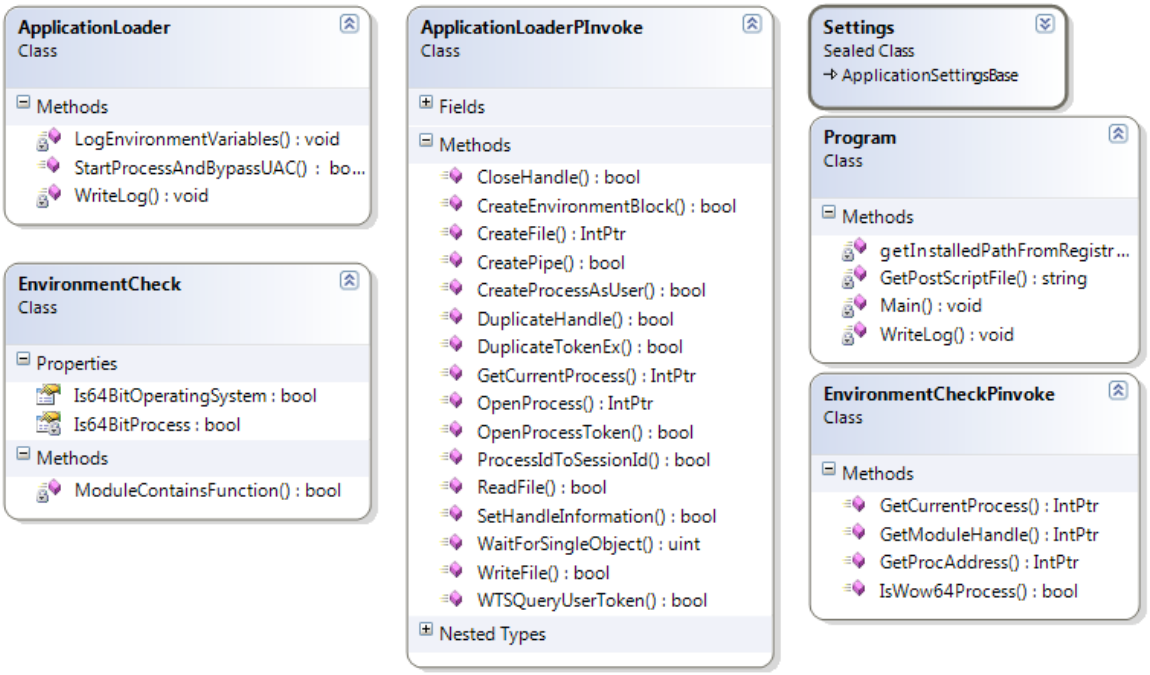


Bild 9.10: Klassendiagramm der PrintPut-Komponente

Die ApplicationLoaderPInvoke-Klasse ist eine Hilfsklasse der ApplicationLoader-Klasse. Sie kapselt alle benötigten PInvoke-Aufrufe der Betriebssystemfunktionen. Die ApplicationLoader-Klasse ist für den Aufruf des PrintPut-Dialogs verantwortlich. Sie bietet die öffentliche StartProcessAndBypassUAC()-Methode an. Diese tut mit Hilfe der User-Session-ID den Benutzertoken holen, den PrintPut-Dialog-Prozess starten und die Postscript-Daten übergeben. Damit wird die Session-0-Isolation der UAC umgangen.

9.2.3 PrintPut-Dialog

Der PrintPut-Dialog dient als grafische Benutzerschnittstelle. Beim Aufruf bekommt der Print-Dialog vom Vaterprozess (PrintPut) die Postscript-Daten über den Standardeingabekanal mitgeschickt. Der Nutzer kann in diesem Konvertierungseinstellungen vornehmen und benötigte Daten für die Integration in ein α-Doc ergänzen.

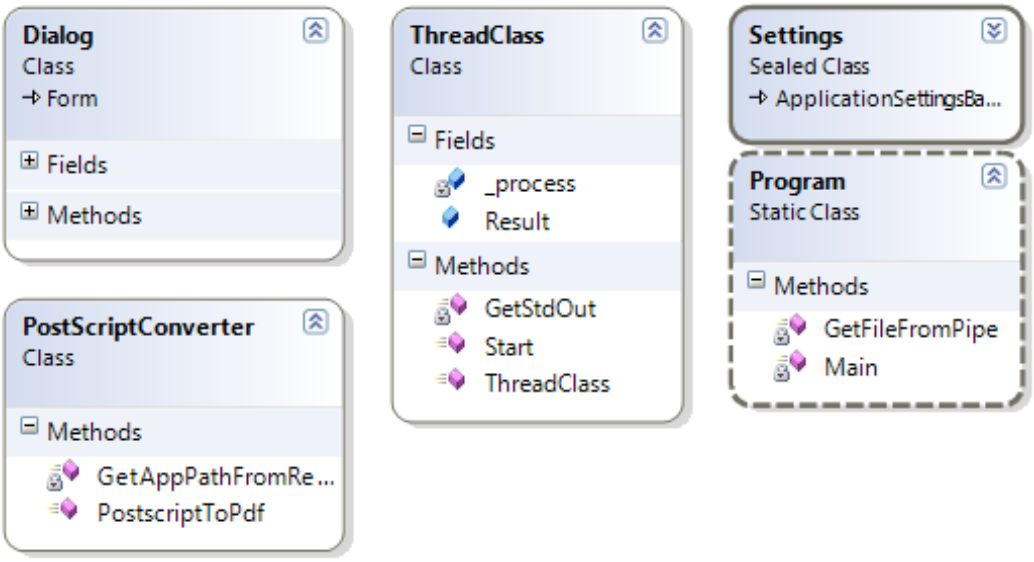


Bild 9.11: Klassendiagramm des PrintPut-Dialogs

Der PrintPut-Dialog besteht aus fünf Klassen. Zum einen wieder eine Settings-Klasse für die Konfigurationsdaten und selbstverständlich die Program-Klasse mit der Main-Methode. In dieser wird der, wenn übergeben, Postscript-Datenstrom eingelesen. Dann wird die eigentliche Form für den Nutzer sichtbar. Diese wird durch die Dialog-Klasse repräsentiert. Nun gibt es eine Fallunterscheidung. Wurde dem PrintPut-Dialog kein Datenstrom übergeben, so kann der Nutzer nur die Default-Einstellungen ändern. Der PrintPut-Dialog fungiert dabei als Settings-Dialog. Dieser wird ebenfalls aufgerufen, wenn der Nutzer die „PrintPut Settings“ im Startmenü aufruft.

Wurde der PrintPut-Dialog jedoch von dem PrintPut-Prozess aufgerufen und per Parameter durch ein „-“ mitgeteilt, dass Postscript-Daten kommen, so liest der PrintPut-Dialog diese ein und bietet dem Nutzer in der Form die Möglichkeit der Konvertierung und Integration des Datenstroms in ein α-Doc. Die Art des Dialogs wird durch einen Flag manipuliert. Dies wird in der folgenden Abbildung veranschaulicht.

Die PostScriptConverter-Klasse kümmert sich um den Aufruf des Konverters. Die Methode PostscriptToPdf bekommt einen Postscript-Daten als String übergeben und gibt PDF-Daten wiederum als String zurück.

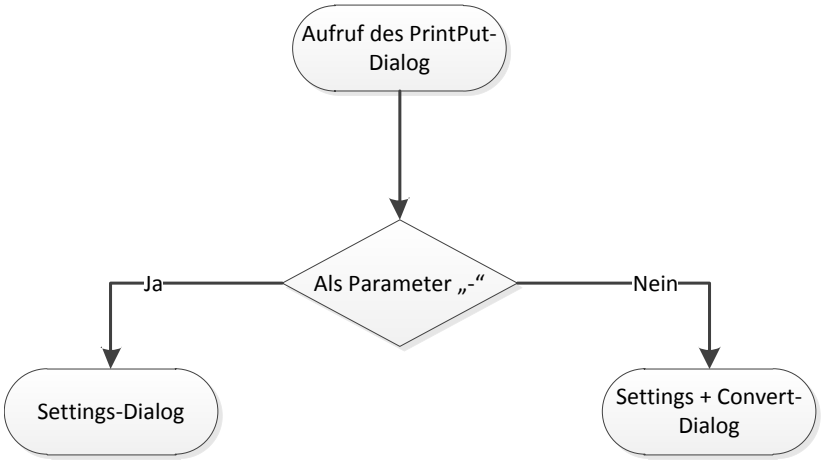


Bild 9.12: Ablaufsteuerung durch Parameter bei Aufruf des PrintPut-Dialogs

Die ThreadClass-Klasse wird in diesem Fall benötigt, weil der Konverter direkt die Postscript-Daten in den Standardeingabekanal geschrieben bekommt und die erzeugenden PDF-Daten wieder direkt in den Standardausgabekanal zurückschreibt. Da dabei nur ein begrenzter Puffer geschieht, müssen die beiden Aktionen parallel stattfinden. Bei Auslastung eines Puffers kann einer der Threads nicht weiterarbeiten und dadurch kommt der andere wieder zum Einsatz. Genauso verhält es sich andersherum. Somit ist keine explizite Synchronisation notwendig.

9.2.4 Konverter

Der Konverter der Postscript-Daten zu PDF-Daten übernimmt bei PrintPut das Open-Source-Projekt Ghostscript. Dieses leistet zurzeit die beste Arbeit, die man für umsonst bekommen kann. PrintPut steuert den Konverter über Parameter an. Der Benutzer kann die Parameteraufrufe über verschiedene Einstellungen im PrintPut-Dialog nach seinen Wünschen bzw. Bedürfnissen anpassen. Der Vorteil von PrintPut gegenüber einigen anderen Lösungen ist, dass PrintPut Ghostscript direkt in die Installation integriert. So muss Ghostscript nicht extra eigenständig installiert werden. Der generierte PDF-Datenstrom wird schließlich dem α-Injector der im Print-Dialog gewählten α-Card zur Integration gestellt.

9.2.5 α-Injector

Der α-Injector ist wie in der konzeptionellen Lösung beschrieben das Bindeglied zwischen PrintPut und dem α-Doc. Durch Aufruf mit bestimmten Parametern wird dem α-Injector die PDF-Datei übergeben. Bisher wurde mit Absicht darauf verzichtet die Postscript-Daten oder PDF-Daten auf der Festplatte dazwischen zu lagern. Dies hat den Vorteil, dass man sich nicht um mögliche Probleme mit dem Dateisystem kümmern müsste. Leider ist dies beim α-Injector derzeit nicht möglich. Darum wird bei der Übergabe der PDF-Datei an den α-Injector die PDF-Datei in eine temporäre Datei ausgelagert und nach der Integration gelöscht. Die PDF-Datei wird dann als Payload in eine neue α-Card eingefügt. Die dazu benötigten Informationen, wie Actor-ID, Alpha-Card Title etc. werden dabei aus den gewählten Einstellungen im PrintPut-Dialog übernommen. Diese werden über Parameter dem α-Injector mitgeteilt. Der α-Injector ist die einzige Komponente, die nicht in PrintPut integriert ist. Diese ist Teil der α-Docs und stellt auf diese Weise eine Verbindung nach außen bereit.

9.2.6 PrintPut-Setup

Alle Komponenten von PrintPut werden in der Setup-Applikation gebündelt. Diese besteht aus den einzelnen Komponenten, einem Banner und dem benutzerdefiniertem Code der während der Installation und Deinstallation ausgeführt wird. Dies wird durch eine Überschreibung der Basis-Installer-Klasse erreicht. Bei der Installation wird zuerst der Installationsvorgang der Basisklasse ausgeführt und anschließend die für PrintPut benötigten Druckerkomponenten installiert. Bei der Deinstallation verhält es sich andersherum. Dort werden zuerst die Druckerkomponenten entfernt und anschließend wird der Deinstallationsvorgang der Basisklasse aufgerufen. Wie bei der konzeptionellen Lösung beschrieben ist hierbei die Reihenfolge wichtig. Denn ohne die entpackten Daten können die Druckerkomponenten nicht installiert werden. Außerdem ist die Reihenfolge der Installation der Komponenten wichtig, da sie auf einander aufbaut.

Im Klassendiagramm sieht man, dass die PrintPut-Installation sehr kompex ist. Dies liegt daran, dass die Installation von Druckern, Ports und Monitoren sehr systemnah ist und keine fertigen .NET-Framework-Klassen dafür bereitstehen.

In der Settings-Klasse werden die Konfigurationsdaten für PrintPut hinterlegt. Dort stehen die Dateinamen, die für die Installation verwendet werden.

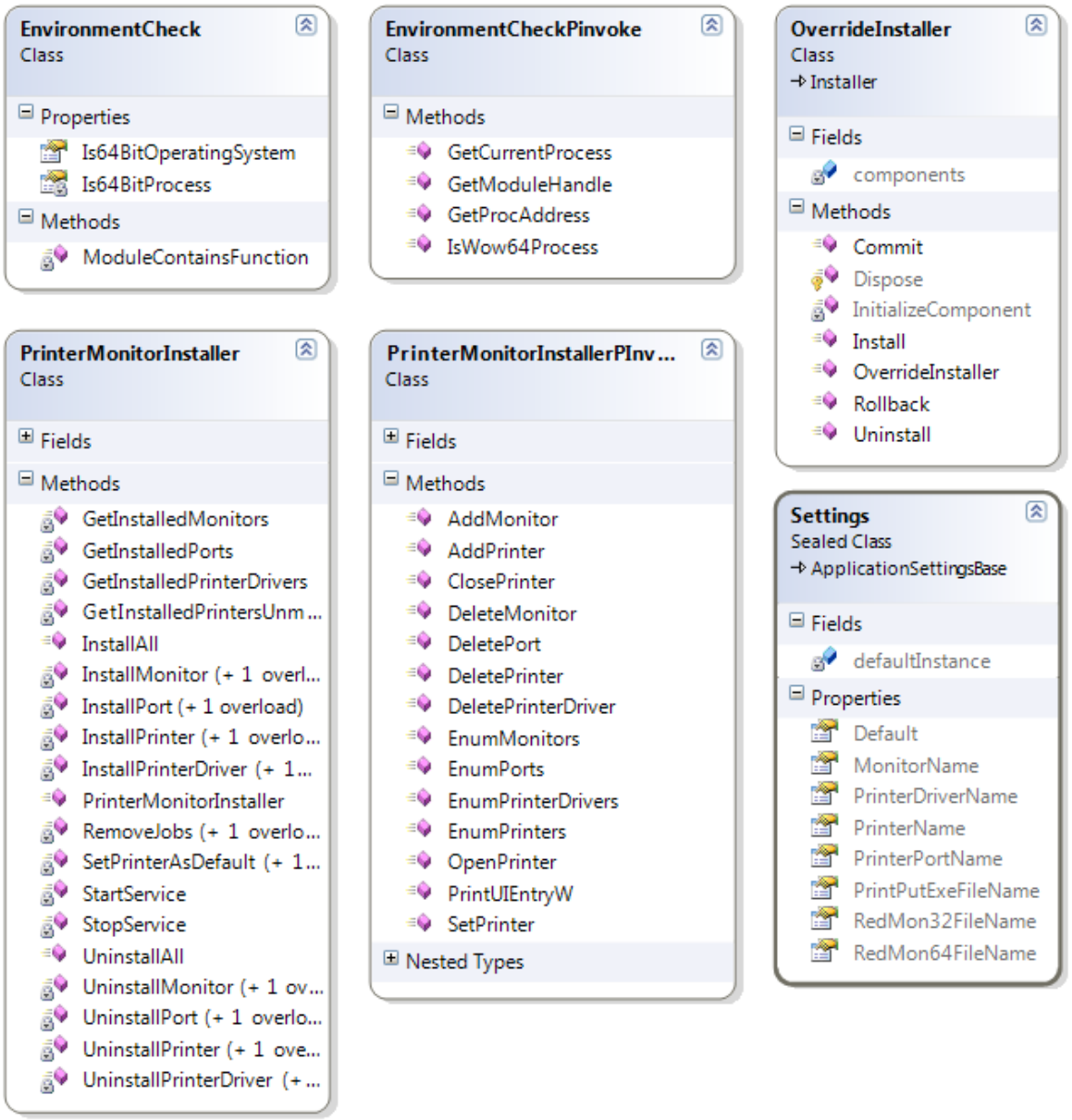


Bild 9.13: Klassendiagramm der Setup-Komponente

Die EnvironmentCheck-Klasse, die bereits bei der PrintPut-Komponente aufgeführt wurde dient der Erkennung von 64-Bit Systemen. Dies ist wichtig, denn dadurch wird die richtige RedMon-Version installiert. Ansonsten ist die Installation für 32-Bit-Systeme ausgelegt und funktioniert auch auf 64-Bit Systemen.

Beim Starten des Installationsvorgangs wird eine Instanz des Override-Installers erzeugt. Anschließend wird dessen Install()-Methode aufgerufen. In dieser wird zunächst

die Installation an die Basisklasse weitergeleitet. Dies führt dazu, dass alle Dateien kopiert und Registry-Einträge geschrieben werden. Anschließend wird jedoch noch eine Instanz des PrinterMonitorInstallers erzeugt und dessen InstallAll()-Methode aufgerufen.

Der PrinterMonitorInstaller ist zuständig für die Installation, Konfiguration und Deinstallation der PrintPut-Druckerkomponenten. Mit Hilfe der PInvoke-Hilfsklasse werden bei der Installation nacheinander der Port Monitor, der Drucker-Port, der Druckertreiber und schließlich der PrintPut-Drucker installiert. Als Druckertreiber wird der in Ghostscript mitgelieferte Treiber benutzt. Es wäre zwar möglich einen beliebigen mit Windows mitgelieferten Postscript-Druckertreiber zu installieren, aber für die bessere Kompatibilität zu Ghostscript und um unabhängig von den Windows-Druckertreibern zu sein wird der Ghostscript-Druckertreiber bevorzugt. Die Deinstallation verläuft entsprechend entgegengesetzt. Zuerst werden die Druckerkomponenten entfernt und anschließend die Dateien und Registry-Einträge gelöscht.

9.2.7 Zusammenfassung

In diesem Kapitel wurde eine beispielhafte prototypische Umsetzung des Lösungsentwurfs aus Kapitel 8 vorgestellt. Diese Umsetzung soll nur als Beispiel dienen und aufzeigen, wie beliebige Dokumente von Fremdanwendungen in einem einheitlichen Format im α-Doc abgelegt werden können und auf diese Weise die prozessübergreifende Patientenbehandlung vereinfachen zu können. Diese Lösung ist keinesfalls perfekt. Im folgendem Kapitel wird dazu kritisch Stellung genommen.

10 Kritik

In diesem Kapitel werden die Schwächen der PrintPut-Lösung aufgezeigt. Dabei werden verschiedene Problembereiche deutlich.

10.1 Verlust von Informationen

Durch die Konvertierung in das PDF-Format verliert man immer Informationen und besondere Eigenschaften des Ausgangsformats. Dies kann sehr problematisch werden. So können hochauflösende medizinischen Aufnahmen durch die Umwandlung wertlos erscheinen. Denn bei PDF wird eine nicht reversible Bilder-Komprimierung durchgeführt. Durch die Wahl des PDF-Formats zur Darstellung beliebiger Dokumente wurde ein Kompromiss geschlossen, denn das PDF-Format hat nicht nur Vorteile. Für bestimmte Szenarien eignet es sich nicht. So kann ein medizinisches Formular nach der Konvertierung in das PDF-Format nicht mehr vom nächsten Arzt elektronisch ausgefüllt bzw. bearbeitet werden. Allgemein ist bei α-Docs keine Modifizierung des Dokumentes nachträglich möglich.

Derzeit wird auch nur ein Payload pro α-Card erlaubt. So ist es auch nicht möglich das Originaldokument zusammen mit der generierten PDF hinzuzufügen. Dies könnte man jedoch im Rahmen der Future Work umsetzen. Ebenso die Generierung einer PDF aus mehreren verschiedenen Dokumenten. Ähnliche Lösungen zu PrintPut haben so eine Funktion bereits.

10.2 Abhängigkeit von Drittanbietern

Eine weitere Schwäche ist, dass PrintPut nur in Windows-Systemen lauffähig ist. In anderen Abteilungen oder Einrichtungen können auch Linux- oder Macintosh-Systeme zum Einsatz kommen. Dort greift der Lösungsansatz mit PrintPut nicht.

Außerdem baut PrintPut auf Drittprogrammen auf. So wird ein PDF-Reader zur Darstellung der PDF-Dateien benötigt, Ghostscript zur Konvertierung und RedMon als

Port Monitor zur Weiterleitung des Druckauftrags. Dadurch macht sich der Lösungsansatz abhängig von Fremdanbietern. Dies muss nicht unbedingt ein Nachteil sein, vor allem da Ghostscript und RedMon Open Source Lösungen darstellen. Eine Möglichkeit der Darstellung von PDF-Dateien in den α-Docs direkt über den α-Editor könnte die Abhängigkeit von den PDF-Readern lösen.

10.3 Umsetzung von PrintPut

Bei der Umsetzung von PrintPut wurden einige technische Feinheiten beschlossen, die kritisch begutachtet werden sollten. So werden bis auf die Übergabe an den α-Injector die Postscript und PDF-Datenströme direkt zwischen den PrintPut-Komponenten ausgetauscht. Da die Daten nie ausgelagert wurden, befanden sie sich die ganze Zeit im Arbeitsspeicher. Das Problem dabei ist, dass Postscript-Datenströme eine beachtliche Größe annehmen können und dies dazu führen kann, dass größere Dokumente bei Knappheit von Arbeitsspeicher nicht konvertiert werden können.

Ein weiteres Problem stellt das Sicherheitsrisiko von PrintPut dar. Da der PrintPut-Prozess in der Session 0 mit System-Privilegien aufgerufen wird, könnte eine Schadsoftware dies versuchen auszunutzen um selber an diese Rechte zu gelangen. Es würde bereits ausreichen den PrintPut-Prozess mit einer bösartigen Datei zu ersetzen und schon würde beim nächsten Druckvorgang das Schadprogramm an Systemrechte gelangen und beliebige Manipulationen am System durchführen können.

11 Abschließende Bemerkungen

Im Laufe dieser Arbeit wurde eine Lösungskonzept basierend auf einem virtuellen Windows-Drucker konzepiert und implementiert, das beliebige Druckbare Dokumentformate in das PDF konvertiert und eine direkte Integration in elektronische Fallakten ermöglicht. Auf diese Art und Weise soll ein organisationsübergreifander Dokumentenaustausch im Gesundheitswesen auf Basis von elektronischen Fallakten kostengünstig und mit geringem Aufwand eingeführt werden, da keine zusätzlichen Anwendungen, Schulungen oder Anpassungsprozesse benötigt werden.

PrintPut baut auf der Druckarchitektur des Betriebssystems Micrsoft Windows als des meistverbreiteten Endanwendersystems auf, dabei werden Versionen ab Windows 2000 unterstützt, das .NET-Framework ab Version 2.0 ist Mindestvoraussetzung für die Nutzung. Bei der Implementierung von PrintPut wurde insbesondere auf die Spezifika der Benutzerkontensteuerung, die mit Windows Vista eingeführt worden ist, eingegangen, die sich als erschwerend erwiesen haben.

Selbstverständlich ist auch PintPut nicht frei von Kritikpunkten. So gibt es bei der Konvertierung in das PDF-Format die Gefahr des Informationsverlustes,z.B. bei der Umwandlung hochauflösender medizinischer Grafiken, die komprmiert werden. Auch ist das PDF trotz seiner weiten Verbreitung und Standardisierung ein Dateiformat, das sich als problematisch erweisen kann, da die nachträgliche Bearbeitung von Dokumenten, die in das PDF konvertiert worden sind, für Endnutzer kaum noch möglich. Auch die Systemabhängigkeit von Windows ist ein Kritikpunkt, der trotz der dominanten Stellung des Windows-Betriebssystems auf dem Endanwendermarkt durchaus berechtigt ist.

Abschließend läßt sich festhalten, dass PrintPut ein möglicher Lösungsansatz ist, der zwar praktikabel, jedoch nicht makellos ist. Es bietet im Rahmen der Problemstellung eine Möglichkeit, die Integration bestehender Datenbestände in α-Flow zu erleichtern und ist damit ein Beitrag zur unterstützung institutionsübergreifender Abläufe im Rahmen des ProMed-Projektes.

Literaturverzeichnis

[Bor95] G. Born. *The file formats handbook*. International Thomson Computer Press, 1995.

[FJJ03] J. Ferraiolo, F. Jun, and D. Jackson. Scalable vector graphics (svg) 1.1 specification. *World Wide Web Consortium (W3C)*, 2003.

[FVDF+94] J.D. Foley, A. Van Dam, S.K. Feiner, J.F. Hughes, and R.L. Phillips. *Introduction to computer graphics*, volume 55. Addison-Wesley, 1994.

[HK01] F.S. Hill and S.M. Kelley. *Computer graphics: using openGL*. Prentice hall Upper Saddle River, NJ, 2001.

[JN03] K. Jähn and E. Nagel. *e-Health*. Springer Verlag, 2003.

[LED+02] A. LaMarca, W.K. Edwards, P. Dourish, J. Lamping, I. Smith, and J. Thornton. Taking the work out of workflow: mechanisms for document-centered collaboration. In *ECSCW'99*, pages 1–20. Springer, 2002.

[MHB10] Sara Motiee, Kirstie Hawkey, and Konstantin Beznosov. Investigating user account control practices. In *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, CHI EA '10, pages 4129–4134, New York, NY, USA, 2010. ACM.

[NL09] Christoph P. Neumann and Richard Lenz. alpha-Flow: A Document-based Approach to Inter-Institutional Process Support in Healthcare. In *Proc of the 3rd Int'l Workshop on Process-oriented Information Systems in Healthcare (ProHealth '09) in conjunction with the 7th Int'l Conf on Business Process Management (BPM'09)*, Ulm, Germany, September 2009.

[NL10] Christoph P. Neumann and Richard Lenz. The alpha-Flow Use-Case of Breast Cancer Treatment – Modeling Inter-Institutional Healthcare Workflows by Active Documents. In *Proc of the 8th Int'l Workshop on Agent-based*

Computing for Enterprise Collaboration (ACEC) at the 19th Int'l Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2010), Larissa, Greece, June 2010.

[NL12] Christoph P. Neumann and Richard Lenz. The alpha-Flow Approach to Inter-Institutional Process Support in Healthcare. *International Journal of Knowledge-Based Organizations (IJKBO)*, 2(3), 2012. Accepted for publication.

[NSWL11] Christoph P. Neumann, Peter K. Schwab, Andreas M. Wahl, and Richard Lenz. alpha-Adaptive: Evolutionary Workflow Metadata in Distributed Document-Oriented Process Management. In *Proc of the 4th Int'l Workshop on Process-oriented Information Systems in Healthcare (ProHealth'11) in conjunction with the 9th Int'l Conf on Business Process Management (BPM'11)*, Clermont-Ferrand, France, August 2011.

[Pre85] A. Press. *PostScript language reference manual*. Addison-Wesley Longman Publishing Co., Inc., 1985.

[Rus07] M. Russinovich. Inside windows vista user account control. *Technet*, 2007.

[Sch96] L. Schamber. What is a document? rethinking the concept in uneasy times. *Journal of the American Society for Information Science*, 47(9):669–671, 1996.

[Sel08] Manuel Selling. Sicherheitsprinzipien in neuen Windows-Systemen. *cms-journal 30*, 2008.

[Sys85] Adobe Systems. *PostScript language tutorial and cookbook*, volume 1. Addison-Wesley, 1985.

[Tro10] Andrew Troelsen. *Pro C# 2010 and the .NET 4 Platform, Fifth Edition*. Apress, Berkely, CA, USA, 5th edition, 2010.

[Yua01] F. Yuan. *Windows graphics programming: Win32 GDI and DirectDraw*. Prentice Hall, 2001.