profi.com®
we make IT work

CASE STUDY

# TESTAUTOMATISIERUNG MIT „CODED UI“

## Die Aufgabe

Unser Auftraggeber ist einer der vier Global Player in den Bereichen Wirtschaftsprüfung, Steuerberatung und Unternehmens- bzw. Managementberatung. Mit rund 10.000 Mitarbeitern ist das Unternehmen an mehr als 20 Standorten in Deutschland präsent.

Das Beratungsunternehmen verfügt über exzellente Branchen- und Marktkenntnisse, die nach Wirtschaftssegmenten und wesentlichen Märkten spezialisiert eingesetzt werden. Dabei fließen die weltweiten Projekterfahrungen der Spezialisten zusammen.

Bei der Entwicklung des unternehmenseigenen „Workforce-Managementsystems“ (WFM) wünschte der Kunde eine Testautomatisierung, die durch die profi.com AG mit einem Fünf-Personen-Team umgesetzt wurde. Innerhalb kurzer Zeit wurde ein Automatisierungsgrad von mehr als 20 Prozent erreicht.

Die Implementierung eines effizienten Testprozesses umfasste die Einbindung verschiedener Subsysteme und deren Schnittstellen sowie die Synchronisation zwischen der Entwicklungsumgebug Microsoft Visual Studio und dem Microsoft Testmanager.

## Unternehmensspezifische Software für den optimalen Personaleinsatz

10.000 Mitarbeiter

Optimierung ist heutzutage in allen Bereichen ein häufig genanntes Schlagwort. Für ein großes Beratungsunternehmen ist es daher notwendig, die eigenen Ressourcen optimal zu planen und einzusetzen. Dies ist für einen kleinen Pool an Mitarbeitern (bis zu 10 Leute) sicherlich ohne große Mühe mit einfachen Mitteln wie Microsoft Excel möglich. Für hinreichend große Unternehmen (ab 100 Mitarbeiter) ist dies nicht mehr so einfach.

In unserem speziellen Beispiel sprechen wir von 10.000 Mitarbeitern, die optimal eingesetzt werden sollen. Es ist logisch, dass hierfür eine neue, an das Unternehmen angepasste Softwarelösung entwickelt werden muss. Die Anwendung sollte die klassischen Aspekte eines modernen „Workforce-Managementsystems“ (WFM) beinhalten und auch in Sachen Usability und Anwenderfreundlichkeit punkten.

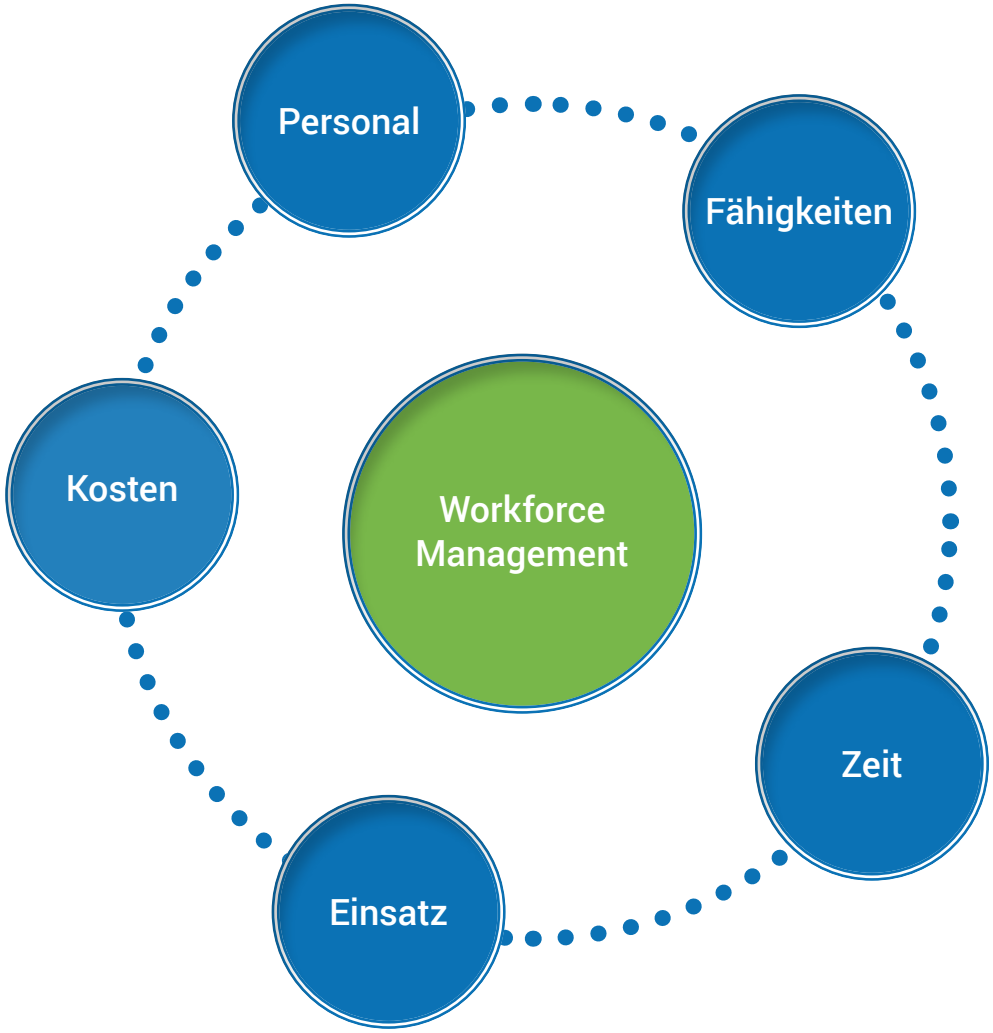



Abbildung 1: Anforderungen an ein WFM-System

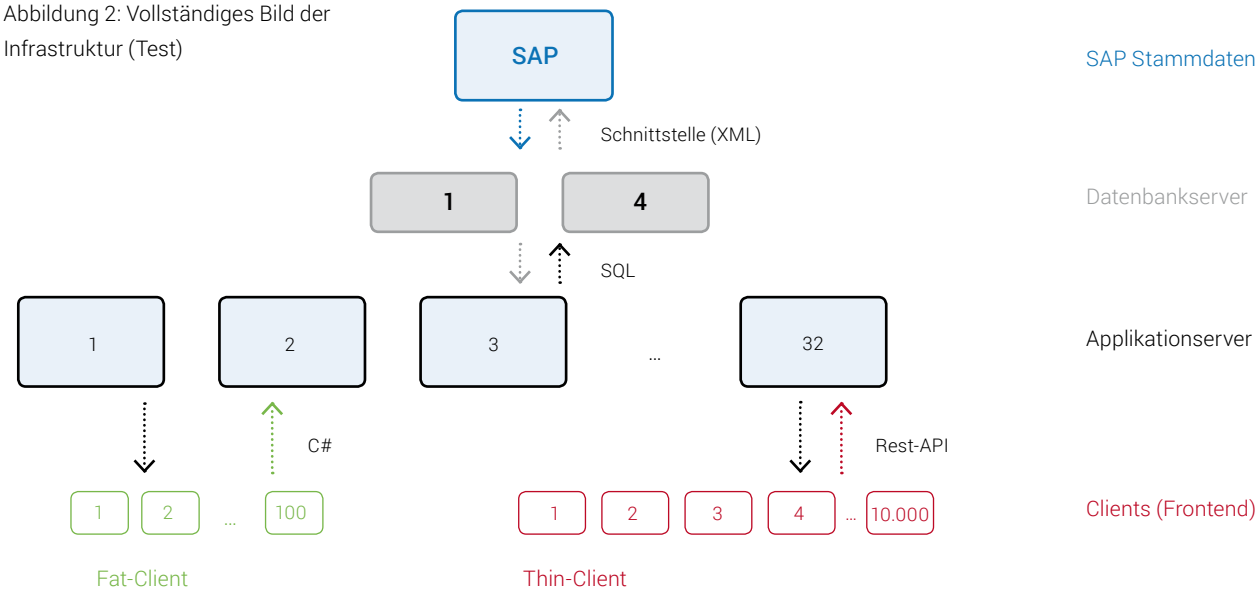
## Komplexes Zusammenspiel zwischen Anwendungen und Infrastruktur

Die konkrete Umsetzung dieses WFM-Systems wurde unter folgenden Rahmenbedingungen realisiert:

- Entwicklungsumgebung von Microsoft „Team Foundation Server“ (TFS) und „Visual Studio“
- Genutzte Programmiersprache: C#, DevExpress, SQL u.a.
- Schnittstelle zu SAP für Personal-/ Stammdaten

Aufgrund der Schnittstellen zu Fremdsystemen wird schnell klar, dass sowohl in der Implementierung als auch im Testprozess selbst die Applikation nicht als „Stand-Alone“-Anwendung betrachtet werden darf. Ein vollständiges Bild der Infrastruktur verschafft die folgende Grafik, welche alle entscheidenden Bestandteile der IT-Architektur darstellt.

Abbildung 2: Vollständiges Bild der Infrastruktur (Test)




## Unterschiedliche Anwender benötigen verschiedene Daten und Workflows

Die Infrastruktur bestand aus einer Schnittstelle zum SAP-System für die Personal- und Stammdaten. Diese wurde über ein gängiges XML-Austauschformat realisiert und parallel zur Entwicklung der Applikation getestet. Als Datenbank wurde der weit verbreitete Microsoft SQL-Server aufgesetzt, die Frontend- und Backendservices wurden in C# entwickelt. Dabei galt es, in der Realisierung zwei unterschiedliche Frontend-Clients zu beachten:

- Fat-Client (C#, .NET Windows Forms)
- Thin-Client (C#, DevExpress)

Der **Fat-Client ist dabei als eigenständige Windows-Anwendung** zu installieren und zu nutzen und für einen Anwenderkreis von 50-100 Personen zur Planung, Disposition und Genehmigung von Dispositionen gedacht.

Der **Thin-Client** als „Light“-Variante der Anwendung **ist im Browser zu bedienen** und soll von den ca. 10.000 Mitarbeitern zur Selbstdisposition bzw. Ansicht der Selbstdispositionen genutzt werden.

Anhand der gestellten Anforderungen erschließt sich, dass der Testanteil für diese Applikation nicht unerheblich sein wird. Die **Gesamtanzahl belief sich auf zirka 1.000 durchzuführende Testfälle**, die nach einer ausführlichen Studie der Anforderungen, Gesprächen mit den Entwicklern und Einbringen der eigenen Expertise dokumentiert wurden.

Für einen vollständigen manuellen Regressionstest waren so in etwa zwei Wochen notwendig. Hier setzte das Team der profi.com AG um Dr. Carsten Neise an, um den Testprozess sehr effizient in Richtung Testautomatisierung voranzutreiben.

## Balance zwischen Granularität der Testfälle und Aufwand der Durchführung

Der Testprozess wurde komplett im **Microsoft Testmanager** realisiert. Dieses Tool bietet sich an, da die Verknüpfung zur Entwicklungsumgebung (Microsoft Visual Studio) sehr eng ist und man somit ein vollständiges Testmanagementsystem zur Verfügung hat.

Die Testfälle sollten gleich nach den etablierten Grundprinzipien der Testautomatisierung erstellt werden.

- Hohe Modularität
- Hohe Wiederverwendbarkeit
- Hohe Wartbarkeit
- Sinnvolle Verwendung von Parametern

Die schwierige Aufgabe, die es zu lösen galt, bestand darin, die Balance zwischen dem Grad der Granularität und dem Aufwand für die Erstellung der Testfälle zu finden. Sind die Testfälle zu granular gestaltet ist der Aufwand für die manuelle Durchführung zu hoch, jedoch ist dabei die Testautomatisierung sehr einfach. Ist die Granularität zu niedrig gestaltet, wird die manuelle Durchführung einfacher, jedoch ergibt sich ein erheblicher Nachteil bei der Testautomatisierung, da häufig genutzte Funktionen immer wiederkehrend implementiert werden.

Mit Hilfe der Verwendung der modularen Testfälle war es uns möglich, die Coded-UI Testfälle in zwei Varianten durchzuführen.

- Testblock
- „Use Case“

Ein „Use Case“ ist eine Verknüpfung einer oder mehrerer Testblöcke. Dabei wurde zumeist der Testblock im sogenannten „Happy Path“ durchfahren. Das Ziel eines solchen Szenarios war das Überprüfen eines vollständigen „Use Case‘s“ aus der Sicht der späteren Anwender.

Im Gegensatz dazu wurden die Testblöcke auch einzeln absolviert. Dabei wurde auf ganz spezielle Eingaben für diesen separaten Testblock geachtet, um somit das Verhalten bei typischen Eingaben der Anwender zu prüfen.

Für eine Durchführung der Coded UI-Testfälle müssen die Vor- und Nachbedingungen gegeben sein, ebenso wie die Objekterkennung garantiert sein muss. Für das Herstellen der Vor- und Nachbedingungen wurden SQL-Skripte verwendet. Die Objekterkennung konnte leider nicht zu 100% realisiert werden, da Standard DevExpress Controls abgeändert und die vererbten Komponenten leider nicht mehr für den „Objekt Spy“ zugänglich waren. Für die restlichen Controls aus Standardklassen war die Objekterkennung kein Problem.

- Vor-/Nachbedingung per SQL-Skripte
- Objekterkennung nicht immer gegeben (Gantt-Controls, Grid-Controls), da keine DEV-Express Standards

Auch wenn zirka 70 % der wichtigen Kernfunktionen grafisch nicht automatisierbar waren, konnte eine **Automatisierung aller Testfälle von 20 % erreicht** werden. Schon bei diesem Anteil machten sich die geringen Wartungsaufwände durch die Modularisierung bemerkbar.

## Testen ohne vollständige Infrastruktur mit TFS-API

Zum Zeitpunkt des Beginns der Testautomatisierung bestand noch keine vollständig fertige Testumgebung. Es fehlten die „TestController“. Damit war eine Ausführung der Coded UI-Testfälle durch den Microsoft Testmanager nicht möglich. Diese mussten aus Visual Studio ausgeführt werden.

Das Team um Dr. Carsten Neise machte sich die sogenannte „TFS-API“ zu Nutze, mit welcher die **Durchführung der Coded UI-Testfälle aus dem Visual Studio** erfolgte und eine **Synchronisation (Testfallstatus, Test Report) mit dem Microsoft Testmanager automatisiert am Ende der Testfälle** erfolgte.

Abbildung 3: Nutzung der TFS-API für die Durchführung der Coded-UI Testfälle


**TFS-API**

- Parameter
- Result File
- Setzen der Run-Status (im MTM)

## Das Fazit

Durch die modular aufgebauten und automatisierten Testfälle ergeben sich für den Kunden zahlreiche Verbesserungen, die sich sowohl funktional als auch ressourcenseitig als Vorteil erweisen:

- Ausführung der Testfälle ist manuell als auch automatisiert möglich
- Testautomatisierung ist ohne Infrastruktur (Modul TestLab im TFS) möglich, durch Nutzung der TFS-API
- Testfalldurchführung konnte um den Faktor 10 verkürzt werden (in diesem Bereich, wo die Objekterkennung funktioniert)
- Testfalldurchführung war zu jedem beliebigem Zeitpunkt möglich

Nach den ersten Schritten in Richtung einer breit angelegten Testautomatisierung stellten sich bei unserem Kunden schnell spürbare Erfolge ein. Und dank kompetenter und individueller Lösungen arbeitet die profi.com AG bereits an den nächsten Schritten und unterstützt den Kunden bei:

- Der Erstellung von Unittests
- Der Erstellung von Last- und Performancetest
- Dem Aufbau der Testumgebung (Microsoft TestLab)
- Der Ausführung der Coded UI-Tests in der neuen Testinfrastruktur
- Der Integration der vom DevExpress-Standard abweichenden graphischen Controls, um diese ebenso zu automatisieren