HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

FLUGHAFENZÜRICH

Studienarbeit, Abteilung Informatik

Smart Meeting Planner

Hochschule für Technik Rapperswil

Herbstsemester 2014
18. Dezember 2014

Autoren:	Marc Juchli & Lorenz Wolf & Thomas Charrière
Betreuer:	Prof. Dr. Luc Bläser
Projektpartner:	Flughafen Zürich AG
Arbeitsperiode:	15.09.2014 - 19.12.2014
Arbeitsumfang:	240 Stunden, 8 ECTS pro Student

Inhaltsverzeichnis

Kapitel 1

Abstract

Die Organisation von Meetings in grösserem Geschäftsumfeld, insbesondere beim Flughafen Zürich, gestaltet sich als zunehmend komplizierter und wird durch die bestehenden Mail- und Kalendersysteme noch zu wenig gut unterstützt. Vielfach müssen dafür verschiedene Dienste wie z.B. der Microsoft Exchange Kalender, Doodle Pools und klassische Mails/Anrufe kombiniert eingesetzt werden, um den Prozess der Terminfindung bis zur Termineinladung zu meistern. Dies liegt daran, dass die einzelnen Dienste nur spezifische Aspekte für sich besonders gut unterstützen (z.B. gemeinsamer Kalender, Termin-Umfrage).

Ziel von Smart Meeting Planner (kurz SMP) ist es daher, den immer wiederkehrenden Aufwand für das Finden eines geeigneten Termins mit mehreren Teilnehmern zu minimieren. SMP soll dabei intelligente Vorschläge erzeugen und die Möglichkeit bieten, die Teilnehmer aktiv ins Auswahlverfahren einbinden zu können. Für die Terminfindung soll SMP dabei die Kalenderinhalte aller Teilnehmer miteinbeziehen.

Im Rahmen dieser Arbeit wurde zunächst ein konzeptioneller Workflow für das Finden eines Termins definiert. Dieser ist auf die Systemlandschaft von Flughafen Zürich AG (FZAG) zugeschnitten. Es ist jedoch auch denkbar, diesen Workflow in einem anderen technischen Umfeld zu implementieren. In einem weiteren Schritt wurde ein entsprechender Prototyp entwickelt. Dabei wurde besonders auf die nahtlose Integration in Outlook 2013 und einer sicheren Anbindung zum Exchange Server Wert gelegt. Der Prototyp wurde fortlaufend in einer Testumgebung validiert welche dem IT-Umfeld von FZAG nachgeahmt ist.

Das Resultat der Arbeit ist ein Prototyp, welcher eine Möglichkeit zur Terminfindung aufzeigt. Diese Lösung muss jedoch noch mittels internen Tests bezüglich Usability und Integration bei FZAG, zu einem fertigen Produkt ausgearbeitet werden.

Kapitel 2

Eigenständigkeitserklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt sind oder mit dem Betreuer schriftlich vereinbart wurden,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben,
- dass wir keine durch Copyright geschützten Materialien (z. B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Rapperswil, den 18. Dezember 2014

Marc Juchli

Lorenz Wolf

Thomas Charrière

Kapitel 3

Ausgangslage

3.1 Problembeschreibung[1]

Die Organisation von Meetings in grösserem Geschäftsumfeld, insbesondere beim Flughafen Zürich, gestaltet sich als zunehmend komplizierter und wird durch die bestehenden Mail- und Kalendersysteme noch zu wenig gut unterstützt. Vielfach müssen dafür verschiedene Dienste wie z.B. der Microsoft Exchange Kalender, Doodle Pools und klassische Mails/Anrufe kombiniert eingesetzt werden, um den Prozess der Terminfindung bis zur Termineinladung zu meistern. Dies liegt daran, dass die einzelnen Dienste nur spezifische Aspekte für sich besonders gut unterstützen (z.B. gemeinsamer Kalender, Termin-Umfrage).

Im konkreten Fall von Microsoft Exchange Server mit Outlook Clients ist zwar gemeinsame Termineinladung und Einsicht in die Ressourcen (Kalender von Personen und Räumen) gut unterstützt. Jedoch fehlt ein effizienter Mechanismus, um freie Terminvorschläge zwischen beteiligten Personen mit entsprechenden freien Räumlichkeiten zu eruieren. Dies lässt sich durch Einsatz einer externen Termin-Umfrage (wie z.B. Doodle) bewältigen, wobei dies jedoch in vielen Unternehmungen aus Gründen von Geheimhaltung und Datenschutz vermieden wird.

Das Ziel dieser Studienarbeit ist es deshalb, einen Prozess zur verbesserten intelligente Terminfindung zu konzipieren. Dieser soll dann als Prototyp mit Integration in den Microsoft Exchange Servers (allenfalls mit Outlook) implementiert werden.

Der Workflow in einem solchen System könnte beispielsweise wie folgt stattfinden: Unter Angabe von Randbedingung und Zielkriterien des Meeting-Initianten (z.B. „ab jetzt bis in 2 Wochen nur abends von 5 bis 7 in Zürich“) soll die Erweiterung optimale Vorschläge von Terminen unterbreiten. Dabei müssen natürlich Terminüberschneidungen der beteiligten Personen in Bezug auf den Kalender der einzelnen Teilnehmer vermieden werden sowie die Verfügbarkeit der nötigen Räumlichkeiten geprüft werden. Die einzelnen Teilnehmer können anschliessend die Möglichkeit erhalten, die Terminvorschläge noch individuell zu bestätigen oder abzulehnen und zu priorisieren. Schliesslich soll das System basierend auf

[1] Verfasst von: Prof. Dr. Luc Bläser

diesen Informationen möglichst automatisch eine optimalen Termin mit der nötigen Raum-Reservation bestimmen und eintragen.

3.2 Aufgabenstellung

Das Ziel dieser Studienarbeit ist es, ein Konzept und Prototyp zur intelligenten Terminfindung für Microsoft Exchange Server (und Outlook) zu entwickeln. Folgende spezifische Ziele werden vorgegeben:

- Untersuchung der Anforderungen/Ziele der Terminfindung und Meeting-Organisation.
- Konzept zur intelligenten Terminfindung für Meetings (Ablauf, Kritierien, Algorithmen).
- Design und Implementation eines Prototyps der intelligenten Terminfindung mit Integration als Add-On/Plugin in den MS Exchange Server und/oder MS Outlook.
- Validierung des Prototyps mit Tests.

Kapitel 4

Anforderungsspezifikation

Die in diesem Kapitel aufgeführten Anforderungen wurden bei FZAG aufgenommen.

4.1 Funktion

Ziel von SMP ist es, den immer wiederkehrenden Aufwand für das Finden eines geeigneten Termins mit mehreren Teilnehmern zu minimieren. SMP soll dabei intelligente Vorschläge erzeugen und die Möglichkeit bieten, die Teilnehmer aktiv ins Auswahlverfahren einbinden zu können.

Für die Terminfindung soll SMP sowohl die Kalenderinhalte aller Teilnehmer aber auch Ressourcenverfügbarkeiten miteinbeziehen. Dadurch sollen auch Überbuchungen von Meeting Räumen verhindert werden.

4.2 Benutzerrollen

Die folgenden Rollen können von jedem Mitarbeiter der FZAG eingenommen werden.

Bezeichnung	**Beschreibung**
Planer	Person, die eine neue Terminfindung organisiert und auslöst. Termin-Planer kann selbst an dem Termin teilnehmen oder aber für jemand anderen organisieren.
Teilnehmer	Person, die an einem Termin teilnimmt und deren Präferenz bei der Terminfindung miteinbezogen wird. Der Teilnehmer erhält Anfragen und kann entscheiden, ob ein gewisser Termin passend ist oder nicht.

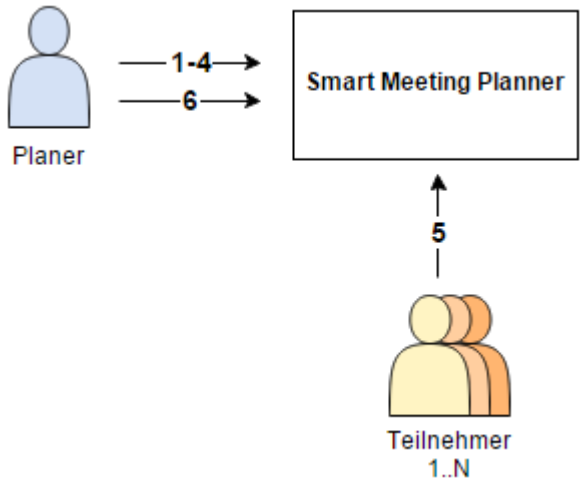
4.3 Use Case “Termin finden”

4.3.1 Main Success Scenario

Der Planer will einen für mehrere Personen passenden Termin finden. Das System gibt anhand der eingegebenen Rahmenbedingungen intelligente Vorschläge. Der Planer kann anschliessend mit einer beliebiger Menge dieser Vorschläge die Präferenzen der Teilnehmer anfordern und entsprechend einen definitiven Termin festlegen.

Schritt	Rolle	Beschreibung
1	Planer	Legt eine Zeitspanne, innerhalb welcher ein Termin gefunden werden soll, fest.
2	Planer	Wählt alle für den Termin benötigte Teilnehmer aus und definiert die Dauer des Termins.
3	Planer	Gibt zusätzliche Informationen wie Titel und Beschreibung an.
4	Planer	Grenzt die vom System generierten Vorschläge in eine beliebige Menge Terminslots ein.
5	Teilnehmer	Präferiert ein oder meherere Terminslots aus der eingegrenzten Menge von Vorschlägen.
6	Planer	Wählt ahnand Teilnehmer-Präferenzen den am meisten geeigneten Termin als Ergebnis des Terminfindungsprozesses aus.

Abbildung 4.1: Main Success Scenario



4.3.2 Alternative Flows

Termin buchen ohne Teilnehmer Präferenzen

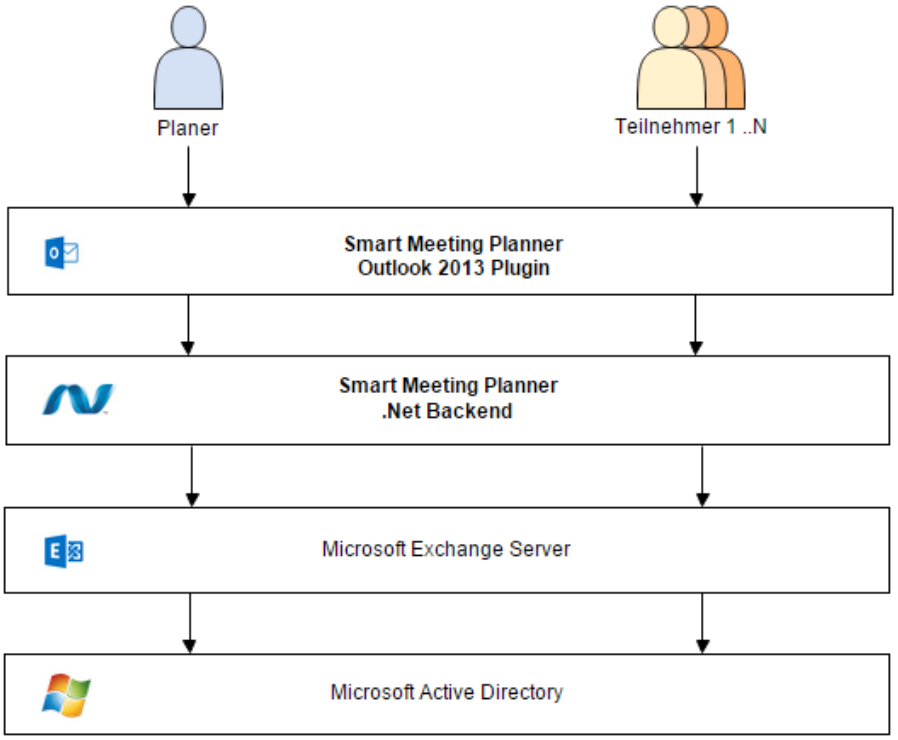
Aus den vom System generierten Terminvorschlägen kann der Planer ohne Miteinbezug der Teilnehmer einen optimalen Termin eruieren und bucht diesen direkt mittels der herkömmlichen Outlook Terminanfrage.

Schritt	Rolle	Beschreibung
4a	Planer	Wählt einen Terminvorschlag aus und legt diesen als definitiven Termin fest.
5-6a	Planer	Bucht definitiven Termin für alle Teilnehmer.

4.4 Integration in die IT-Landschaft von FZAG

SMP soll für die Microsoft basierte Systemlandschaft von FZAG optimiert werden. Konkret ist eine Integration in Outlook als Plugin gefordert. Ein Microsoft Exchange Server kann und soll angebunden werden. Zudem ist ein vom Plugin entkoppeltes .Net Backend erwünscht. Im Rahmen dieser Arbeit wird nicht vor Ort, sondern auf einer möglichst realen Testumgebung an der Hochschule für Technik Rapperswil (HSR) entwickelt und getestet.

Abbildung 4.2: System Kontext



4.5 Non functional requirements

4.5.1 Performance

Der bisherige Betrieb (Standardfunktionalität Outlook) darf durch SMP spezifische Implementierungen insofern nicht beeinträchtigt werden, dass unerwünschte Nebeneffekte auftreten und zu spürbaren Performance-Einbrüchen führen.

Die Ausführungsgeschwindigkeit der Terminfindung selbst, soll die von Outlook gewohnten Reaktionszeiten aufweisen.

Die Effizienz des Benutzers soll nicht durch unnötige Wartezeiten gehemmt werden.

4.5.2 Portability

Die Implementierung der Terminfindung soll kompatibel zu Office 2013 sein und eine Anbindung an Exchange Server 2013 erlauben.

Die Installation der Lösung soll über Verteilung eines Microsoft Installer (MSI) Packages möglich sein.

4.5.3 Security

Das firmeninterne Berechtigungssystem darf nicht umgangen werden. Ein Benutzer von SMP darf nicht mehr Informationen erhalten, als er berechtigt ist einzusehen. Der Zugriff auf Exchange Daten soll mittels Integrated Authentication stattfinden.

4.5.4 Usability

Für FZAG steht die Usability im Vordergrund. Das Design soll sich nahtlos in die Office Umgebung soweit einbringen, dass der Benutzer nicht feststellt, dass es sich hierbei um eine Drittapplikation handelt.

Die einzelnen Ansichten sollen die Benutzer in deren Effizienz unterstützten, in dem das Finden eines Termins in möglichst wenigen Schritten abgehandelt werden kann.

Kapitel 5

Lösungskonzept

5.1 Einleitung

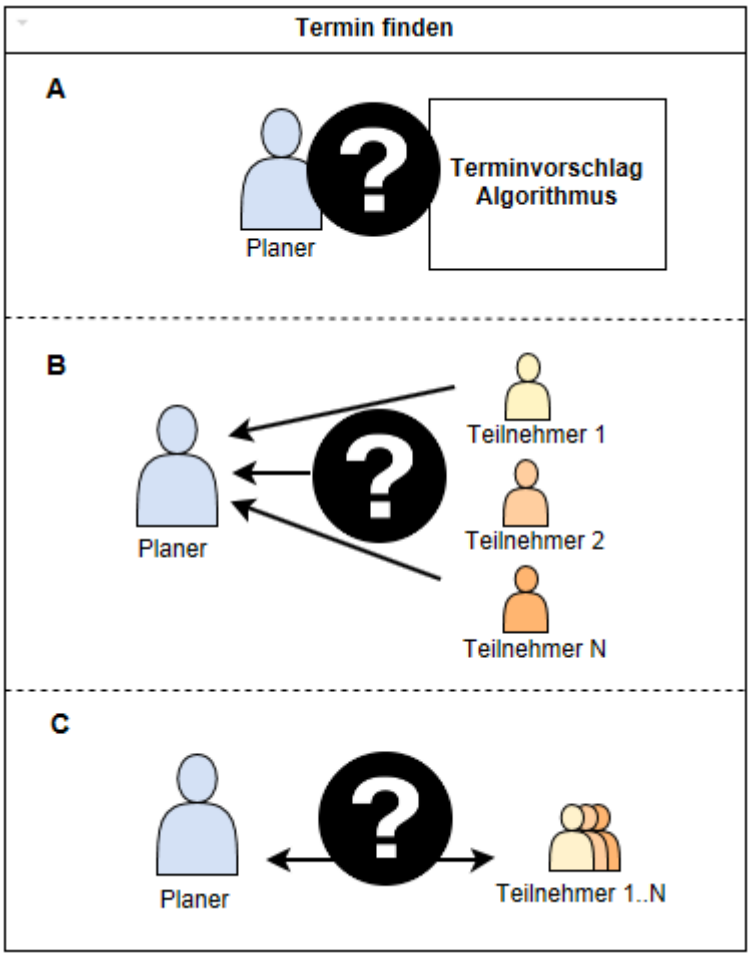
Wie in den Anforderungen spezifiziert, soll sich SMP in die bestehende Systemlandschaft von FZAG integrieren. Konkret geht das im Folgenden beschriebene Lösungskonzept immer von einer Integration in Outlook 2013 als Add-In mit einer Exchange Server Anbindung aus. Verglichen mit anderen Terminfindungs-Ansätzen, wie beispielsweise Doodle, können mit einer integrierten Lösung diverse Vorteile ausgemacht werden. So sind die Teilnehmer durch ihren Exchange Server Account eindeutig identifizierbar. Seitens Benutzer ist nicht zwingend ein zusätzlicher Login nötig. Des Weiteren sind intelligente Terminvorschläge möglich, da die Terminkalender der Teilnehmer eingesehen werden können. Auch Outlook 2013, als bereits bekannte und schnell zugängliche Umgebung ermöglicht besseren Anklang beim Endbenutzer. Seitens Entwickler kann zudem Funktionalität von Outlook wie beispielsweise Adressbuch oder Kalenderansicht wiederverwendet werden. Diese und weitere Aspekte wurden bei der Erarbeitung des Lösungskonzepts miteinbezogen.

5.2 Grundlegende Fragen

Der Use Case *Termin finden* (siehe 4.3) umfasst mehrere Aktionen die bei unterschiedlichen Personen in einer bestimmten Reihenfolge ausgeführt werden. Jeder Schritt in diesem Prozess erfordert Eingaben vom Benutzer und das anschliessende Benachrichtigen einer anderen Person. Es muss ein Workflow definiert werden welcher für jede Terminfindung durchlaufen kann. Für die Definition dieses Terminfindungs Workflows müssen zunächst drei grundlegende Fragen beantwortet werden:

- **A**: Wie kann der Planer bei der Eröffnung einer Terminfindung durch intelligente Vorschläge vom System unterstützt werden?
- **B**: Auf welche Art und Weise können die Präferenzen von Teilnehmern in die Terminfindung miteinbezogen werden?
- **C**: Wie kann der Planer mit den Teilnehmern kommunizieren und umgekehrt?

Abbildung 5.1: Grundlegende Fragen



5.2.1 A: Terminfindung eröffnen

Der Prozess der Terminfindung wird damit eingeleitet, dass der Initiator der Umfrage den Teilnehmern Terminvorschläge unterbreitet. SMP soll die gegebene Microsoft Exchange Umgebung zum Vorteil nutzen und die Einleitung durch intelligente Findung von Termin unterstützen. Dabei sollen die vorhandenen Informationen über die Kalender der Teilnehmer ausgewertet werden, sodass der Initiator anhand dessen möglichst passende Termine den Teilnehmern unterbreiten kann.

Für die Darstellung der Terminvorschläge werden folgende Ansätze in Betracht gezogen:

1. **Erweiterung Outlook Terminplanungsassistent:**
   Der herkömmliche Ansatz für die Findung von Terminen mit mehreren Teilnehmern ist der von Outlook bereitgestellte Terminplanung Assistent. Eine Möglichkeit um den Anforderungen, den Teilnehmern mehrere Termine zu unterbreiten, gerecht zu werden, wäre die Erweiterung des bestehenden Terminplanung Assistenten. Hierbei müsste es ermöglicht werden, die in der separaten Liste aufgeführten Terminvorschläge zu priorisieren und anschliessend eine Mehrfachauswahl durch den Initiator der Umfrage zu erlauben. Dabei würde aktiv in den von Outlook vorgegebenen Workflow eingegriffen werden.

2. **SMP Planungsansicht:**
   Ein weiterer Ansatz ist die Erstellung einer SMP spezifischen, separaten Planungsansicht und somit eine Entkoppelung von bestehenden Outlook Komponenten. Dies ermöglicht eine alternative Gestaltung der Komponenten und erlaubt es, den Workflow gezielt auf eine Mehrfachauswahl auszurichten. Auf diese Weise bietet sich die Möglichkeit der individuellen Darstellung eines Kalenders, bei welchem sich alle Termine der einzelnen Teilnehmer überlagert darstellen lassen. Zusätzlich können Terminvorschläge direkt im Kalender aufgezeigt und hervorgehoben werden.

Aus dem Outlook Planungsassistenten resultiert jeweils ein Termin welcher den Teilnehmern gesendet werden kann. Dies so zu erweitern, dass mehrere Terminmöglichkeiten ausgewählt und zur Auswahl gestellt werden, stellte sich als eine technisch Schwierigkeit dar. Hinsichtlich der Abgrenzung von Outlook interner Prozessen und der freien Gestaltungsmöglichkeiten wurde daher letztere Variante gewählt. Dabei nimmt derjenige User, der als Initiator der Umfrage agiert, die Rolle als Planer ein. Nachdem die Rahmenbedingungen für ein bevorstehendes Meeting definiert wurden, werden dem Planer als Resultat die vorgeschlagenen Termine direkt in einer eigenen Kalenderansicht unterbreitet. Die gewünschten Termine können selektiert werden und anschliessend den Teilnehmern als Vorschläge zur Präferierung übermittelt werden.

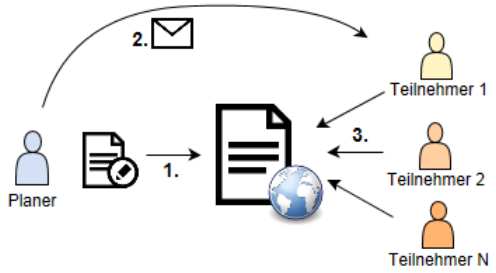
5.2.2 B: Miteinbezug von Teilnehmern

Als Lösung für das Einfordern von Präferenzen der Teilnehmer wurde der Ansatz eines Umfrage-Workflows erarbeitet. Der Planer löst dabei eine Anfrage mit möglichen Terminvorschlägen aus und die Teilnehmer haben daraufhin die Möglichkeit eine beliebige Anzahl der zur Verfügung gestellten Terminvorschläge zu präferieren oder abzulehnen.

Es wurden zwei grundlegende Ansätze für die Lancierung einer Umfrage in Betracht gezogen:

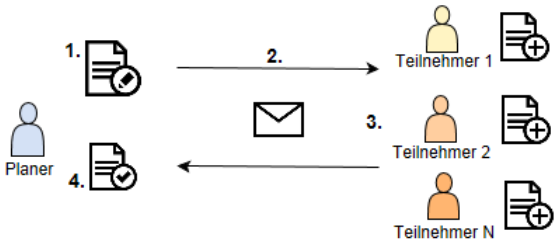
1. **Ansatz einer klassischen Web-Umfrage:**
   Planer erstellt eine Umfrage und stellt diese zentral auf einem System zur Verfügung (1). Anschliessend werden an die gewünschten Teilnehmer lediglich Einladungen mit dem Hinweis, wie auf die Umfrage zugegriffen werden kann, verschickt (2). Die Einladung kann beispielsweise via Mail geschehen, sie enthält einen Hyperlink auf eine spezifischen Web Seite auf welcher die Umfrage ausgefüllt werden kann. Die Teilnehmer gehen schliesslich auf diese Webseite und geben Ihre Präferenzen an (3). Die Eingaben werden direkt im Web persistiert und können jederzeit vom Planer abgerufen werden.

Abbildung 5.2: Konzept Web Umfrage



2. **Ansatz einer klassischen Papier Umfrage:**
   Planer definiert eine Umfrage (1) und verschickt diese mit allen Informationen und Auswahlmöglichkeiten an die gewünschten Teilnehmer (2). Diese können direkt innerhalb der verschickten Nachricht eine Auswahl treffen (3) und wieder zurück senden. Der Planer wartet, bis die einzelnen Umfragen ausgefüllt zurück geschickt werden und kann diese auswerten (4).

Abbildung 5.3: Konzept Papier Umfrage



In Bezug auf das Integrieren als Add-In in Outlook wurde der erste Ansatz als weniger optimal eingestuft. Es wäre denkbar, dass der Planer über das Plugin eine Umfrage starten kann und dadurch entsprechende Nachrichten mit Hyperlinks auf eine Umfrage-Webseite generiert und den Teilnehmern automatisch schickt. Es kann auch in Betracht gezogen werden, diese externe Seite direkt innerhalb von Outlook in einem Frame einzubinden. Das Outlook Add-In wäre in diesem Fall aber nur für das Starten einer neuen Umfrage von Nutzen. Für die restlichen Schritte des Workflows würde ein zusätzliche Web-Applikation benötigt werden. Dies erhöht die Komplexität des System als Ganzens unnötig und ist keine intuitive Outlook Integration, welche die potentiellen Vorteile ausschöpft.

Bei der Konzeption von SMP entschied man sich für das grundlegende Prinzip des zweiten Ansatz. Konkret heisst dies, es gibt keine zentrale Umfrage (-Plattform), auf die zugegriffen wird. Anstatt dessen werden die Auswahlmöglichkeiten dem Teilnehmer direkt mitgeschickt, worauf dieser eine Auswahl treffen und dies dem Planer zurückschicken kann.

Auf die Problematik, wie mit einem Outlook Add-In benutzerdefinierte Daten, gar Formulare, an beliebige Teilnehmer geschickt und anschliessend vom Planer als Gesamtergebnis aggregiert werden, wird in den nachfolgenden Kapiteln eingegangen.

5.2.3 C: Kommunikation zwischen Planer und Teilnehmer

Das gewählte Prinzip für die Handhabung einer Umfrage fordert, dass der Planer die Termin Auswahl-Möglichkeiten dem Teilnehmer schicken und dieser eine Menge davon direkt präferieren und zurückschicken kann. Es wurden verschiedene Möglichkeiten zur Realisierung dieser Interkommunikation zwischen den involvierten Personen, innerhalb von Outlook, in Betracht gezogen:

1. **Terminanfrage pro Auswahlmöglichkeit:**
   Dem Teilnehmer wird für jede Auswahlmöglichkeit eine Standard Outlook Terminanfrage geschickt. Diese kann über bekannte Funktionen zugesagt oder abgelehnt werden. Der Planer sieht schliesslich via SMP Add-In eine Gesamtübersicht der erstellten Termine und den jeweiligen Antworten der Teilnehmer.

2. **Mailanfrage pro Umfrage:** Für jede gestartete Umfrage wird allen Teilnehmern eine Mail geschickt. Das Mail selbst beinhaltet eine Menge von Terminvorschlägen. Durch Antworten des Mails kann der Teilnehmer seine Präferenzen zu den Auswahlmöglichkeiten angeben und dem Sender, in der Rolle als Umfrageinitiator, zurückschicken.

Ein Vorteil der ersten Lösung ist, dass nur der Planer das SMP Add-In benötigt. Die restlichen Schritte des Umfrage-Workflows sind dabei innerhalb der Standard Funktionen von Outlook. Zu Bedenken gibt jedoch die Tatsache, dass bei n Terminauswahlmöglichkeiten auch n Terminanfragen generiert werden. Es fehlt die Möglichkeit, die einzelnen Vorschläge in einem einzelnen Umfrage Objekt zu kapseln. Dies ist Usability technisch schlecht, da dadurch die Teilnehmer lediglich mit einer eine Menge von Terminanfragen zugeschüttet werden. Es kann nur schwer ein Bezug zu einer Umfrage hergestellt werden.

Die zweite Lösung erscheint zunächst ebenfalls nicht benutzerfreundlich, da in einem Standard Mail lediglich Text verschickt wird. Auch bietet die Default Ansicht für Mails nicht die nötige Funktionalität für diesen Teilprozess der Umfrage. Bei der Erarbeitung des Konzepts für SMP konnten jedoch Lösungen für diese Problematik erarbeitet werden:

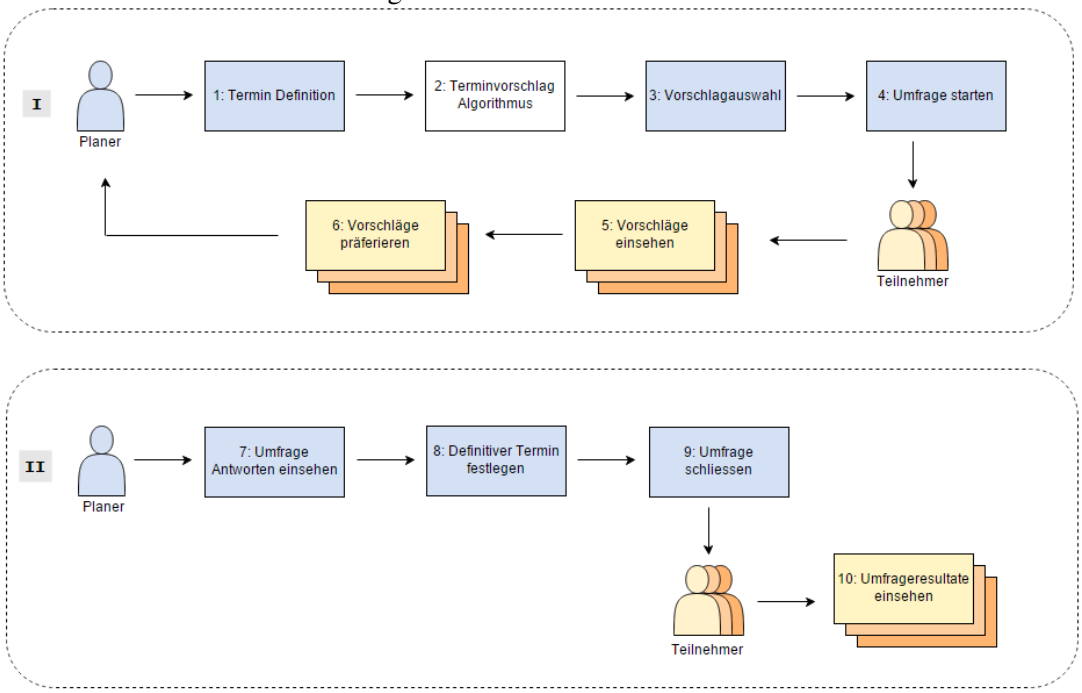
- Die Eigenschaften eines Mail-Objekts (im Folgenden MailItem genannt) können mit benutzerdefinierten Properties (im Folgenden UserProperties genannt) ergänzt werden. UserProperties unterstützen jedoch nur vordefinierte Datentypen, mithilfe Serialisierung ist es aber möglich, dem Mail trotzdem beliebige Datenstrukturen anzuhängen. Konkret sind dies Terminvorschlag-Objekte, die dem Teilnehmer zur Auswahl stehen.
- Damit der Teilnehmer beim Öffnen eines vom SMP Add-In generierten Mails nicht die Standard Ansicht für Mailnachrichten, sondern ein eigens definierte Umfrage View anzeigt, können die internen Eigenschaften eines MailItems so angepasst werden, dass statt der normalen Ansicht für Mails eigene Formular angezeigt werden können (siehe Message Class, Architektur 6.3.1).

5.3 SMP Workflow

Durch Miteinbezug der grundlegenden Fragen wurde der konzeptionelle SMP Workflow wie folgt definiert:

Nr	Aktor	Aktion
Teil I: Umfrage starten		
1	Planer	Definiert Rahmenbedingungen gemäss Anforderungsspezifikation (siehe 4.1)
2	System	Generiert intelligente Terminvorschläge anhand Planungseingaben.
3	Planer	Wählt eine beliebige Menge von Terminvorschlägen aus.
4	Planer	Eröffnet eine Umfrage und schickt diese mit den ausgewählten Terminvorschlägen zur Auswahl an die Teilnehmer.
5	Teilnehmer	Öffnet Umfrage und sieht die Auswahlmöglichkeiten der Umfrage ein.
6	Teilnehmer	Wählt eine Präferenz pro Auswahlmöglichkeit und sendet die Umfrage zurück an den Sender.
Teil II: Umfrage schliessen		
7	Teilnehmer	Sieht Umfrage Antworten der Teilnehmer ein.
8	Planer	Legt einen Terminvorschlag als definitiver Termin fest.
9	Planer	Schliesst Umfrage ab.
10	Teilnehmer	Erhält Umfrageresultat und sieht dieses ein.

Abbildung 5.4: SMP Main Success Workflow

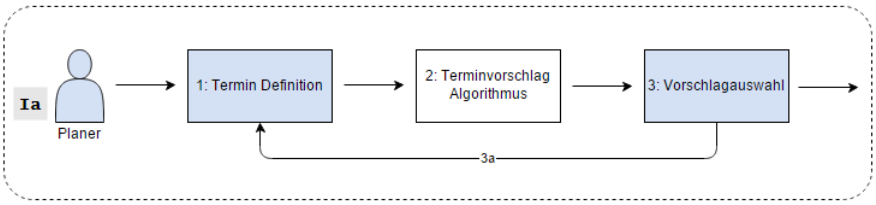


Während der eigentlichen Umfrage, Workflow Teil I, gibt es alternative Flows. Sobald Teil II des Workflows beginnt, haben diese jedoch keinen Einfluss mehr.

Änderung der Termin Definition bevor Umfrage startet:

Nr	Aktor	Aktion
3a	Planer	Ist mit dem Resultat des Terminvorschlag Algorithmus nicht zufrieden. Er wechselt zurück auf Schritt 1, ändert die Termin Definition und durchläuft den Workflow von neuem.

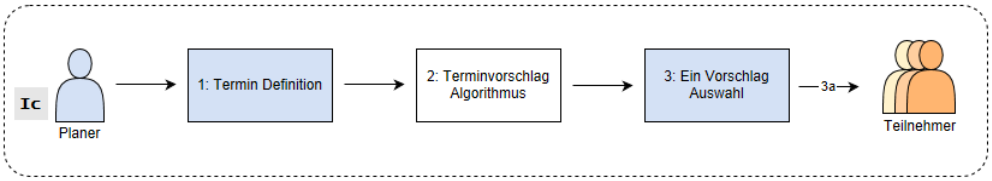
Abbildung 5.5: SMP Alternative Flow a



Termin buchen ohne Teilnehmer Präferenzen

Nr	Aktor	Aktion
3a	Planer	Wählt nur einen der generierten Vorschläge aus und wünscht keine Umfrage. Dadurch wird ein Standard Outlook Termin abgefüllt und es kann eine normale Terminanfrage an die Teilnehmer statt finden. In diesem Fall wird SMP nur für die Generierung von Terminvorschlägen verwendet.

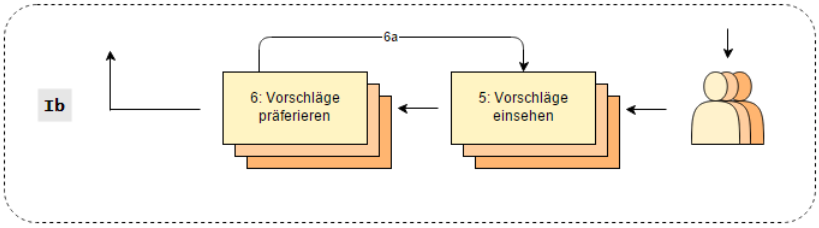
Abbildung 5.6: SMP Alternative Flow c



Änderung der bereits definierten Präferenz:

Nr	Aktor	Aktion
6a	Teilnehmer	Ändert Präferenz nachträglich. Der Planer wird dachurch erneut informiert. Die alte Präferenz wird überschrieben.

Abbildung 5.7: SMP Alternative Flow b



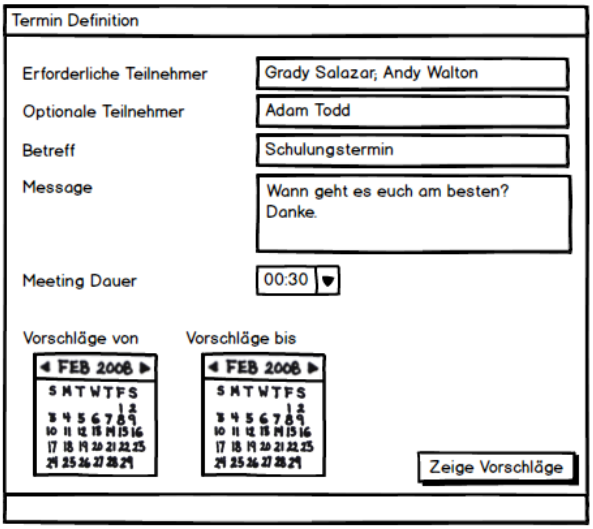
5.3.1 Schritt 1: Termin-Definition

Eine Terminfindung beginnt mit der Termin-Definition. Als anzugebende Anforderung werden nebst der **Meeting Dauer** (auf 15 Minuten genau) und einer **Zeitspanne** in welcher ein Termin gefunden werden soll (Von und Bis), folgende **Ressourcen** geltend gemacht:

- Organisator (i.d.R. der Planer selbst)
- Erforderliche Teilnehmer
- Optionale Teilnehmer

Als einzige Voraussetzung, um eine Terminanfrage auslösen zu können gilt, dass die Terminkalender aller Ressourcen freigegeben sind.

Abbildung 5.8: Mockup - Termin-Definition



Der Planer kann der Umfrage zudem ein Betreff und textuelle Nachricht hinterlegen. Das Aktualisieren der Zeitspanne oder der Teilnehmer führt dazu, dass eine Anfrage zum Terminvorschlag Algorithmus im Hintergrund ausgelöst wird.

5.3.2 Schritt 2: Terminvorschlag Algorithmus

Um den Organisator bestmöglich bei der Suche nach geeignete Terminvorschlägen zu unterstützen, bedarf es einem Algorithmus, welcher über alle Teilnehmer, deren Verfügbarkeiten auswertet und die freie Terminmöglichkeiten als Resultat liefert. Der Algorithmus soll dabei die in der Termin Definition beschriebenen Angaben miteinbeziehen. Das Ziel ist, für eine beliebige Anzahl von Users, freie Termin Slots mit einer bestimmten Termindauer, in einer vorgegebenen Zeitspanne und in einem verfügbaren Raum zu finden.

Aufgabe

In einem ersten Schritt soll der Algorithmus Informationen vom Exchange Server so auswerten, damit er anschliessend eine Verfügbarkeitsmatrix erstellen kann. Diese zeigt alle Verfügbarkeiten der Benutzer innerhalb der angegebenen Zeitspanne. Die Zeilen der Matrix sind dabei durch Time Slots von 15 Minuten Blöcke aufgeteilt und chronologisch von Start bis Ende der gewählten Zeitspanne aufgeführt. Als Spalten werden die jeweiligen Teilnehmer aufgeführt.

Die Verfügbarkeiten der einzelnen Teilnehmer werden anschliessend wie folgt eingetragen:

- **0 (false):** User verfügbar
- **1 (true):** User nicht verfügbar

Folgendes Beispiel zeigt eine Verfügbarkeitsmatrix mit 3 Teilnehmern mit der Zeitspanne zwischen 15:00 und 16:00, wobei Teilnehmer 1 nur teilweise verfügbar ist und Teilnehmer 2 gar nicht:

$$\begin{array}{c} \\ 0\text{min} \\ 15\text{min} \\ 30\text{min} \\ 45\text{min} \end{array} \begin{array}{c} \begin{array}{ccc} T1 & T2 & T3 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

In einem zweiten Schritt soll der Algorithmus aus der Verfügbarkeitsmatrix schliessen, zu welcher Zeit alle oder die meisten Teilnehmer verfügbar sind. Daraus lassen sich schlussendlich effektive Terminvorschläge, die durch eine Start und Endzeit definiert sind, generieren.

Lösung

Bei der Erarbeitung einer Lösung für obige Aufgabe wurden die Möglichkeiten der Exchange Web Service Schnittstelle analysiert. Die Methode **GetUserAvailability**[6] dient zur Abfrage der Verfügbarkeiten von beliebigen Teilnehmer. Diese liefert Informationen über die Termine der Teilnehmer bezüglich einer mitgegebenen Zeitspanne. Das Resultat kann als Grundlage für die Erstellung der Verfügbarkeitsmatrix benutzt werden.

Bei der Bewertung der resultierenden Terminvorschläge unterscheidet SMP zwischen drei Qualitätseigenschaften, welche bei der Darstellung der Vorschlag-Auswahl mit einfliessen:

Fair: Ein erforderlicher Teilnehmer hat eine Terminüberschneidung von max. 15 Minuten.

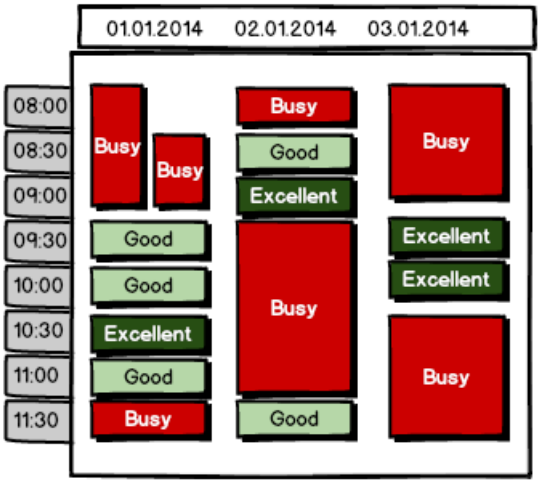
Good: Alle erforderlichen Teilnehmer sind verfügbar. Mindestens ein optionaler Teilnehmer ist nicht verfügbar.

Excellent: Alle Teilnehmer sind verfügbar.

5.3.3 Schritt 3: Vorschlag-Auswahl

Die vom Algorithmus generierten Daten werden in einer Planungsansicht dargestellt. Die Verfügbarkeitsmatrix ist so aufbereitet, dass die gesamte Ansicht wie ein Outlook Kalender, mit einer horizontalen Datum und vertikalen Zeit Achse versehen, dargestellt werden kann. In diesen Kalender werden sowohl die besetzten Termin Slots (rot gekennzeichnet) als auch die eigentlichen Vorschläge (grün gekennzeichnet) als Blöcke eingetragen. Überlappende Blöcke, wie sie zum Beispiel bei mehreren besetzten Slots zur gleichen Zeit vorkommen, werden nebeneinander dargestellt. Der Planer kann nun durch Auswählen von Blöcken eine beliebige Menge der Terminvorschläge, die er den Teilnehmern zur Auswahl stellen will, auswählen. Es muss jeweils mindestens ein Terminvorschlag ausgewählt werden, damit anschliessend eine Umfrage gestartet werden kann.

Abbildung 5.9: Mockup - SMP Planungsansicht



5.3.4 Schritt 4: Umfrage starten

Sobald der Planer alle gewünschten Terminvorschläge selektiert hat, ist die Umfrage bereit ausgelöst zu werden. Dabei werden auf dem Server grundsätzlich zwei Tasks ausgeführt:

1. **Präferenz für den Organisator hinzufügen:** Der Organisator wird ebenfalls am zukünftigen Termin teilnehmen und ist somit ebenfalls Teilnehmer bei dieser Umfrage. Da er die Termine selber auswählt und diese als Terminvorschläge den Teilnehmern versendet, kann davon ausgegangen werden, dass diese Termine aus Sicht des Organisators passend sind. Folglich wird automatisiert und für jeden Terminvorschlag die Präferenz als Zusage gesetzt.

2. ***Tentative Appointments* beim Organisator eintragen:**

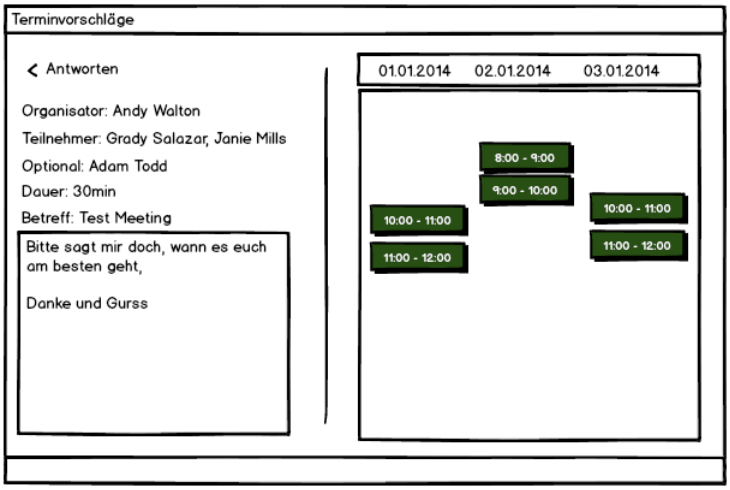
   Mit dem Versenden der Terminvorschläge ist der Organisator potentiell an den vorgeschlagenen Daten nicht mehr verfügbar. Um Terminkonflikte bei späteren Terminumfragen und manuellen Terminbuchungen vorzubeugen, wird im Kalender des Planers für jeden Terminvorschlag ein Tentative Appointment (provisorischer Termin) eingetragen.

Sobald die Tasks auf dem Server abgearbeitet wurden, wir auf dem Client die Umfrage an die Teilnehmer versendet.

5.3.5 Schritt 5: Vorschläge einsehen

Der Teilnehmer erhält die Umfrage mit den vom Planer zur Auswahl gestellten Terminvorschlägen. Diese beinhaltet Informationen zur Umfrage selbst und führt in einer Kalender Ansicht die zur Auswahl stehenden Terminvorschläge.

Abbildung 5.10: Mockup - Vorschläge einsehen



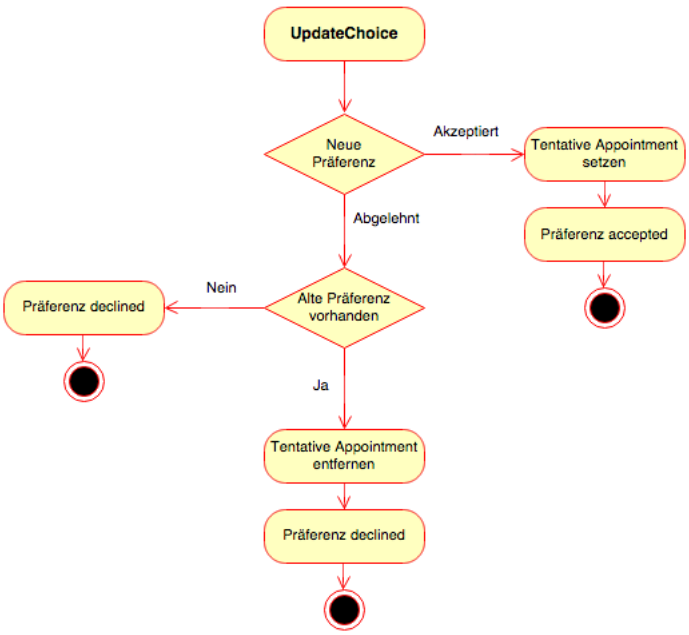
5.3.6 Schritt 6: Vorschläge präferieren

Der Teilnehmern kann, sobald er dafür bereit ist, für die Terminvorschläge eine gewünschte Präferenz angeben. Wird ein Terminvorschlag selektiert, so spricht der Teilnehmer seine Verfügbarkeit für den entsprechenden Zeitraum zu.

Ist der Teilnehmer fertig mit der Eingabe seiner Präferenzen kann er die Umfrage an den Planer zurück leiten. Kurz zuvor wird das Eintragen der Kalendereinträge getätigt.

Das Ausfüllen einer Umfrage soll mehrmals durchgeführt werden können, da sich die Präferenzen der Teilnehmer leicht innert kürzester Zeit ändern können. Folglich liegt die Schwierigkeit im Grunde darin, die Konsistenz der Präferenzen der Teilnehmer zu bewahren. Es muss also geprüft werden, ob die Umfrage bereits ausgefüllt wurde und ob es sich bei der neuen Präferenz um eine Ab- oder Zusage für den Terminvorschlag handelt. Dieser Entscheidungsprozess ist im folgenden Entscheidungsfluss visualisiert:

Abbildung 5.11: Entscheidungsprozess



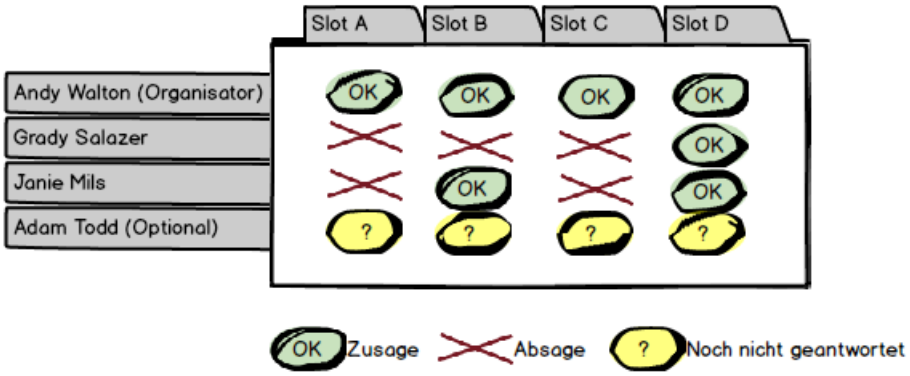
Sind alle Überprüfungen getätigt wird die Umfrage zurück an den Planer geleitet, damit dieser über das Ausfüllen der Umfrage informiert wird und sich bereits einen Überblick bezüglich den Verfügbarkeiten verschaffen kann.

5.3.7 Schritt 7: Umfrage Resultate einsehen

Planer erhält pro Teilnahme ein ausgefüllte Umfrage und kann die Präferenzen des jeweiligen Teilnehmer einsehen. Hat der Teilnehmer seine Präferenz nachträglich angepasst sieht der Planer mehrere Eingänge derselben Person. Diese werden jedoch pro Umfrage gruppiert und chronologisch sortiert.

Der Planer hat jederzeit die Möglichkeit, eine Übersicht anzeigen zu lassen. Diese stellt den aktuellen Stand der Umfrage über alle Teilnehmer in einer tabellarischen Ansicht dar. Der Planer kann dabei einsehen, welche Teilnehmer bereits ihre Präferenzen angegeben haben und welche nicht. Aus technischer Sicht ist hier wichtig, dass die einzelnen Präferenzen der Teilnehmer zuvor in einer Art und Weise gespeichert werden, damit diese jederzeit als Gesamtergebnis abgerufen werden können.

Abbildung 5.12: Mockup - Übersicht der Umfrage Resultate



5.3.8 Schritt 8: Definitiver Termin festlegen

Haben genügend Teilnehmer die Umfrage ausgefüllt und abgeschickt, so kann der Planer die Umfrage abschliessen in dem er einen definitiven Termin festlegt. Der Zeitpunkt für das Abschliessen einer Umfrage liegt voll im Ermessen des Planers. Dazu steht ihm wiederum, eine Gesamtübersicht, wie sie bereits in Schritt 7 zum Einsatz kam, zur Verfügung. Es sind alle Teilnehmer mit den jeweiligen Präferenzen aufgeführt. Die besten Termine sind diejenigen, welche die meisten Präferenzen erhalten haben. Diese werden farblich hervorgehoben. Zum Abschliessen der Umfrage wählt der Planer den gewünschten Termin aus und klickt anschliessend auf *Abschliessen*. Im Hintergrund finden folgende Arbeiten statt:

Tentative Appointments löschen

Die während der Umfrage erstellten *Tentative Appointments*, welche beim Ausfüllen der Präferenzen der Teilnehmer entstehen, sollen gelöscht werden. Dabei ist zu beachten, dass die Termine wirklich zur Umfrage gehören und nicht anderweitige Termin sind.

Festgelegter Termin eintragen

Der vom Planer festgelegte Termin wird nun bei allen Teilnehmern in den Kalender auf Status *Busy* gesetzt.

5.3.9 Schritt 9: Umfrageresultat einsehen

Beim Festlegen des definitiven Termins durch den Planer wird allen Teilnehmern eine finale Nachricht versendet. Durch Öffnen dieser Nachricht kann der Teilnehmer schliesslich einsehen, welcher Termin als Gewinner der Umfrage definiert wurde. Dieser Schritt ist rein informell, der effektive Termin wurde bereits bei Schritt 8 in den Kalendern der Teilnehmern eingetragen. Es ist keine weitere Interaktion von Seiten des Planers oder Teilnehmers mehr möglich.

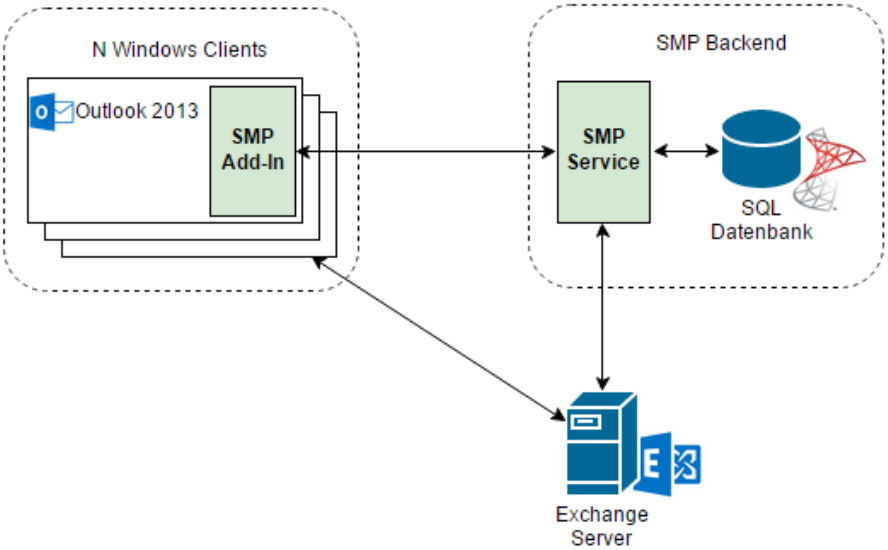
Kapitel 6

Architektur

6.1 Einleitung

Für die Terminfindung wird das Abfragen von Daten von einem Microsoft Exchange Server verlangt. Um diese Anbindung vollständig vom Client, dem Outlook Add-In, zu entkoppeln, wurde eine WCF Service Komponente entwickelt. Es wird zwischen zwei verschiedenen Services unterschieden, um die beiden Schritte “Planung” und “Umfrage” logisch getrennt zu halten (siehe auch 6.2.3). Eine Service Komponente greift auf die Kalenderdaten des Exchange Servers zu und persistiert Umfrage Daten in einer SQL Datenbank. Die Entscheidung für eine separate Datenbank wurde getroffen, um eine möglichst optimierte Datenstruktur ausarbeiten zu können (siehe 6.2.2) und der Entkoppelung vom Exchange Server nach zu kommen. Outlook selbst, behält die bestehende Kommunikation zum Exchange Server bei. Das SMP Add-In jedoch, kommuniziert ausschliesslich mit dem Service via Request-Reply Prinzip.

Abbildung 6.1: SMP Systemarchitektur



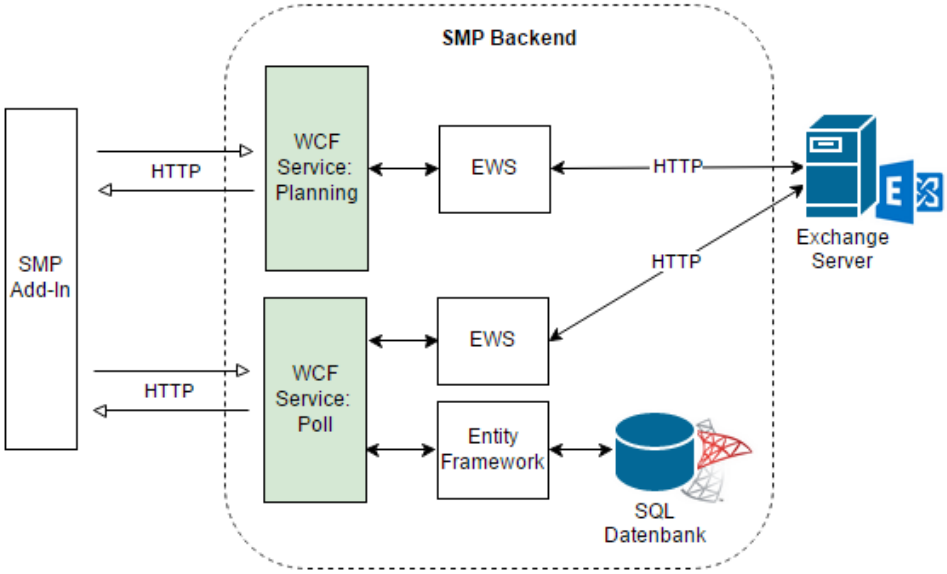
6.2 SMP Backend

Wie in der Architektur Einleitung beschrieben, wurde das SMP Add-In mit einer Service Schnittstelle vom Exchange Server entkoppelt. Gründe dafür liegen zunächst einmal im Security Bereich. Ein Outlook Add-In lässt sich auf beliebigen Workstations installieren und folglich sollen die Zugriffe via dem SMP Service kontrolliert und eingegrenzt werden können.
Für die Auslagerung der gesamten Exchange Zugriffslogik gibt es mehrere Gründe. Nicht nur will man möglichst leichtgewichtige Clients anbieten, auch sollen die Anfragen vom SMP Add-In durch eine Zwischenschicht verarbeitet werden. Auf diese Weise, dem klassischen Client-Server Prinzip, bedarf es nur eine Datenbankanbindung und die Exchange Zugriffe und deren Verarbeitung können mit gewährleisteter Rechenleistung des Servers erfolgen.
Ein weiterer Aspekt für die Einführung eines Service Layers ist die Datenhaltung der Umfragen. Das Lösungskonzept besagt, dass Planer und Teilnehmer im Grunde via Mails kommunizieren. In diesen Mails werden Informationen wie Terminvorschläge oder Teilnehmer Präferenzen mitgeschickt. Damit während einer laufenden Terminfindung keine Daten, durch beispielsweise Löschen von Mails, verloren gehen, sollen diese zentral persistiert werden. Es wird als sehr wichtig eingestuft, dass eine Umfrage zu jedem Zeitpunkt einen klaren und eindeutigen Status hat. Das SMP Add-In kommuniziert daher bei jeder Statusänderung einer Umfrage mit dem Service um die Datenhaltung zu aktualisieren.

Daraus kristallisieren sich zwei Hauptaufgaben für das SMP Backend:

- Kapselung von Exchange Zugriffen
- Sicherstellung der Umfragepersistenz

Abbildung 6.2: SMP Backend im Detail



6.2.1 EWS Integration

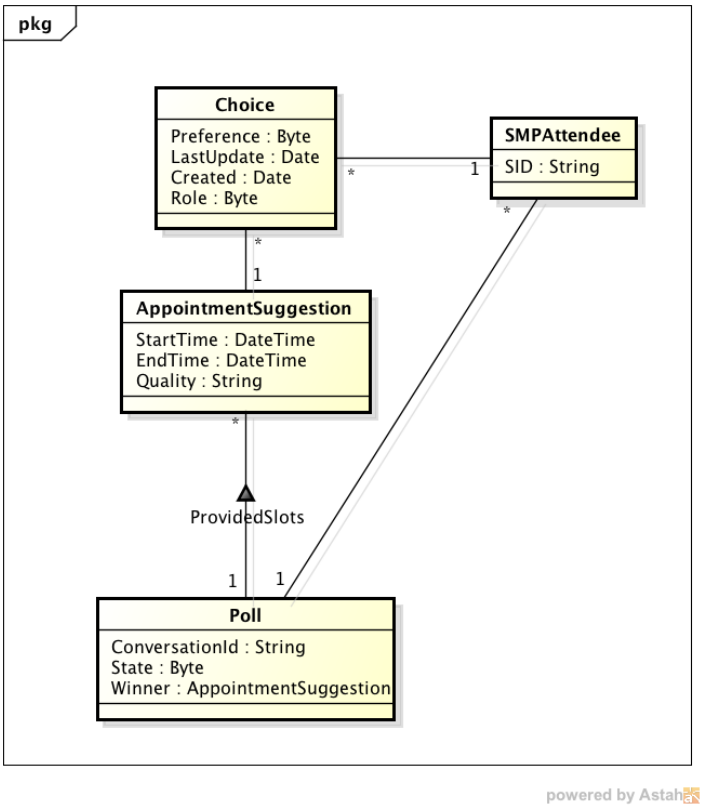
Die Kommunikation mit dem Exchange Server findet ausschliesslich mit der Exchange Web Services (EWS) statt. Diese ermöglicht nicht nur effizientes Bearbeiten der Kalendereinträge, sondern erlaubt es auch, durch Impersonation, Aktionen aus Sicht eines beliebigen Users vorzunehmen. Im Falle der SMP Anwendung ist dies ein zwingendes Feature, um automatisiert Terminanfragen vornehmen zu können, ohne dabei die User in den ganzen Prozess miteinbeziehen zu müssen.

6.2.2 Persistierung

Für das Verwalten von Kalendereinträgen und Zusammenfassen der ausgewählten Termine ist eine Datenbank zur Persistenz der EWS Referenzen erforderlich. Zusätzlich zu den EWS Referenzen werden Informationen zur Umfrage-Durchführung und den einzelnen Teilnehmer sowie ihre Auswählen gespeichert. Parallel dazu werden so weitgehend als möglich, relevante Daten im Outlook-Item persistiert, was eine Offline-Ansicht ermöglicht.
Für eine spätere Mobile-Integration oder eine Web-Seite für externe Teilnehmer wäre damit also gesorgt. Diese Funktionalität liegt aber ausserhalb des Umfangs der Studienarbeit (SA).

Das Domain Modell wurde wie folgt umgesetzt:

Abbildung 6.3: Domain Model - PollService.



Es wurde bewusst in Kauf genommen, dass die Teilnehmer (`SMPAttendee`) für jede neue Umfrage, bei welcher diese eine Präferenz (`Choice`) eintragen, neu angelegt werden. Zwar entstehen Duplikate durch die nicht vorhandene Normalisierung, doch der Vorteil der dadurch verringerten Komplexität der Implementierung übersteigt dies:

Kein Lookup der SMPAttendee
: Beim Ausfüllen der Umfrage (`UpdateChoice`) kann direkt mit der mitgelieferten Email Adresse gearbeitet werden, ohne ein Lookup über die bereits vorhandenen `SMPAttendee` durchzuführen.

Auswirkungen bei Änderungen an User Accounts sind überschaubar
: Es muss damit gerechnet werden, dass die Email Adresse eines Users sich ändern kann. (Z.b. durch Heirat, Scheidung, etc). Wäre nun der `SMPAttendee` normalisiert, so müsste die hinterlegt Email Adresse durch die neue ersetzt werden. Diese Überprüfung ist auf Grund der wenigen Informationen nicht nur schwer realisierbar, sondern müsste auch bei jeder Eintragung der Präferenz eines Teilnehmers (`UpdateChoice`) durchgeführt werden.

Ein weiterer Nachteil ist, dass die Assoziation der alten `SMPAttendee` mit dem aktuellen Exchange User nicht möglich ist.

Die Konsistenz der gespeicherten Appointment ID kann nicht gewährt werden, da es unvermeidlich ist, dass ein User manuell die Appointments, ob tentative oder nicht, in seinem Kalender ändert. Aus diesem Grund müssen die gespeicherten Appointment ID als inkonsistent betrachtet werden und geben lediglich Informationen darüber, welches das letzte vom System eingetragene Appointment war. Diese Tatsache wird in der Businesslogik sorgfältig behandelt.

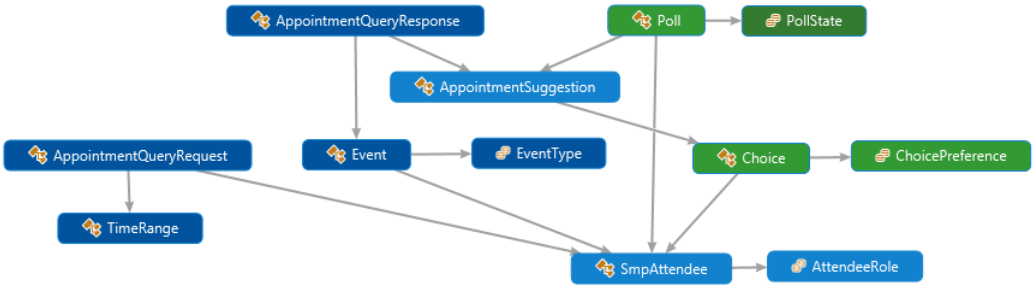
6.2.3 WCF Services

Die Schnittstelle zum SMP Add-In wird mit WCF realisiert. Wie in Abbildung 6.2 ersichtlich, wurden zwei unabhängige Services entwickelt. Im Folgenden sind die Gründe für diese Aufteilung aufgeführt:

1. **Logische Trennung der Business Logik**
   Ein wichtiger Schritt im Terminfindung Workflow ist die Eruierung von Terminvorschlägen. Der Algorithmus benötigt mehrere Exchange Zugriffe und die Kalenderdaten der Teilnehmer einzusehen. Er wurde daher in die Service Schicht ausgelagert und bildet einen unabhängigen Teil, der nichts mit der Umfrage-Persistenz zu tun hat und somit auch keinen Datenbankzugriff benötigt. So wurde ein logische Trennung in zwei Services erarbeitet. Der *Planning Service* beinhaltet dabei die Schnittstelle für den Terminvorschlag Algorithmus und der *Poll Service* wird bei Änderung eines Umfrage Status aufgerufen.

2. **Entkopplung von DTO (Data Transfer Objects)**

Abbildung 6.4: DTO's Service

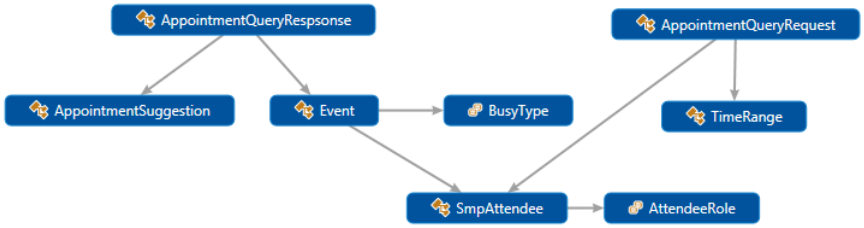


Wie in der Abbildung (6.6) zu erkennen ist, lässt sich die logische Trennung zwischen *Planning* (dunkelblau) und *Poll* (grün) auch in den DTO's widerspiegeln. Dabei ist aber auch zu erkennen, dass es DTOs's gibt, bei welchen es unumgänglich ist, diese in beiden Services zu verwenden (hellblau). Auf Grund der vorliegenden Komplexität und hinsichtlich der Austauschbarkeit der beiden Workflow Prozessabschnitte (Planung und Umfrage), macht eine Entkoppelung und somit eine Aufteilung der in die beiden Services Sinn.

6.2.4 Planning Service

Wie bereits in der Einleitung (siehe 6.1) erwähnt, dient der Planning Service der Planung und somit grundsätzlich der Findung von Terminvorschlägen (`AppointmentSuggestion`). Ebenfalls werden mit diesem Service die Verfügbarkeiten der Teilnehmer (`Event`) wiedergegeben, um folglich eine Kalender-artige Ansicht, wie in Kapitel beschrieben, darstellen zu können.

Abbildung 6.5: DTOs Planning Schnittstelle



Dafür zuständig ist die Methode `GetAppointmentProposals`. Als Argument wird ein `Appointment-QueryRequest` erwartet und der Rückgabewert ist vom Typ `AppointmentQueryResponse`:

Listing 6.1: IPlanningService – GetAppointmentProposals

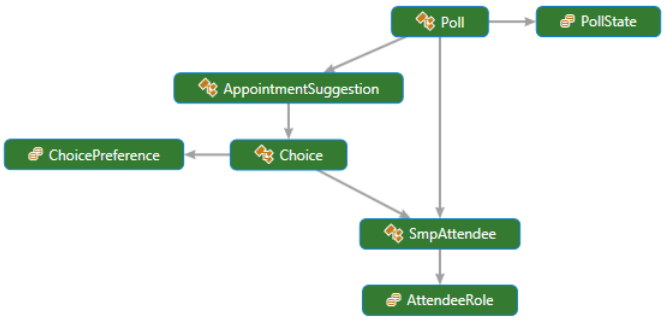
```
1 [OperationContract]
2 AppointmentQueryResponse GetAppointmentProposals(AppointmentQueryRequest ↩
      appointmentQuery);
```

Mit dem `AppointmentQueryRequest`, welcher zusammengesetzt ist aus einer Zeitspanne, der Meeting Dauer und einer Liste von SMPAttendee, können nun Anfragen an der Exchange Server gestellt werden. Dazu wird die EWS Methode `GetUserAvailability`[6] verwendet. Das erwartete Resultat ist vom Typ `GetUserAvailabilityResults`[5] und enthält die effektiven Verfügbarkeiten der Teilnehmer, also deren in der mitgegebenen Zeitspanne gefundenen Termine, sowie konkrete Terminvorschläge. Hierbei dienen mitgelieferten `AvailabilityOptions`[4] für die gewünschte Konfiguration, um beispielsweise sicherzustellen, dass nur jene Termine vorgeschlagen werden, bei denen alle nicht-optionalen Teilnehmer keine überlappenden Termine haben. In diesem konkreten Fall spricht man von der Option: `MinimumSuggestionQuality = SuggestionQuality.Excellent`.
Anschliessend werden die Resultate für die weitere Rückgabe vom Typ `AppointmentQueryResponse` aufbereitet. Nicht brauchbare Informationen, wie die Termine mit Vorbehalt oder Vorschläge ausserhalb den Bürozeiten werden dabei vernachlässigt.

6.2.5 Poll Service

Ebenfalls wurde der Poll Service in der Einleitung (siehe 6.1) bereits erwähnt. Dieser dient zur Unterstützung des kompletten Umfrage-Prozesses. Wurde aus dem `AppointmentQueryResponse`, das Resultat der Planning Service Methode `GetAppointmentProposals`, eine Auswahl an `AppointmentSuggestions` getroffen, so dienen diese beim Poll Service nun als Ausgangslage.

Abbildung 6.6: DTOs Planning Schnittstelle



CreatePoll

Anfangs soll eine Umfrage mittels der Methode `CreatePoll` erstellt werden. Als Argumente werden eine auf dem Client generierte Conversation-ID, eine Liste der `SMPAttendee` und die Liste von `AppointmentSuggestion`, mitgeliefert.

Listing 6.2: IPollService – CreatePoll

```
1 [OperationContract]
2 bool CreatePoll(string conversationId, List<SmpAttendee> attendees, List<↩
    AppointmentSuggestion> appointmentSuggestions);
```

Hierbei erfolgt als erstes das Persistieren eines neuen Poll Objektes. Anschliessend soll, wie im Workflow (siehe 5.3.4) beschrieben, für den Planer bereits Zusagen für die jeweiligen Terminvorschläge gesetzt werden. Dafür wird ein `Choice` Objekt mit der `Preference Accepted` für jede `Appointment-Suggestion` eingetragen. Die tentative Appointments im Kalender des Planers, werden mittels der EWS Schnittstelle angelegt. Dies geschieht durch die Erzeugung von neuen `Appointment` [3] Objekte mit der Option `appointment.LegacyFreeBusyStatus = LegacyFreeBusyStatus.Tentative`.

UpdateChoice

Die Verfügbarkeiten der Teilnehmer werden mit der Methode `UpdateChoice` kund gegeben. Der Client hält jeweils bereits die Conversation-ID, `AppointmentSuggestion` Globally Unique Identifier (GUID) und das entsprechende `Choice` Objekt. Folglich wird für jede `Choice` ein separater Aufruf getätigt.

Listing 6.3: IPollService – UpdateChoice

```
1 [OperationContract]
2 bool UpdateChoice(string conversationId, Choice choice, Guid ↩
    appointmentSuggestionGuid);
```

Es wird zuerst geprüft, ob ein `Choice` Objekt bereits peristiert wurde ist bzw. ob der Teilnehmer bereits ein Umfrage ausgefüllt hat. Dies kann mittels der `appointmentSuggestionGuid` und der `Attendee.Email` ermittelt werden. Ist dies der Fall, so wird das Attribut `Preference` überschrieben und falls der neue Wert nicht dem alten Wert überein stimmt, wird das tentative Appointment, wiederum abhängig von der `Preference`, durch die EWS Schnittstelle erstellt bzw. gelöscht. Ist dies jedoch nicht der Fall und die `Choice` ist noch nicht vorhanden, dann wird ein neues `Choice` Objekt angelegt und persistiert, sowie den Termin mit Vorbehalt im Kalender wiederum für den entsprechenden `Attendee` eingetragen.

CompletePoll

Zum erfolgreichen Abschliessen der Umfrage wird die Methode `CompletePoll` beansprucht. Vom Client mitgegeben wird die GUID des finale Termins (`definitiveAppointmentId`) sowie die `conversationID` zur Identifikation des `Poll` Objekts.

Listing 6.4: IPollService – CompletePoll

```
1 [OperationContract]
2 bool CompletePoll(string conversationId, Guid definitiveAppointmentId);
```

Dem entsprechenden Poll Objekt wird die *definitiveAppointmentId* zugewiesen und persistiert. Anschliessend müssen Aufräumarbeiten ausgeführt werden. Für jede Referenz vom `Poll` auf ein `Choice` Objekt müssen die allenfalls bestehenden tentative Appointments gefunden und gelöscht werden. Dabei muss fehlertolerant vorgegangen werden, da es durchaus möglich ist, dass Teilnehmer die Appointments manuell aus deren Kalender gelöscht haben. Zum Schluss wird ein Appointment bei allen Teilnehmern, welche ihre Zusage für den finalen Terminvorschlag zugesprochen haben, mit dem Status *LegacyFreeBusyStatus.Busy* eingetragen.

StopPoll

Auch dem Szenario eines Abbruchs der Umfrage muss gedient werden. Dazu wird die Methode *StopPoll* mit der entsprechenden *conversationId* aufgerufen.

Listing 6.5: IPollService – StopPoll

```
1 [OperationContract]
2 bool StopPoll(string conversationId);
```

Die Implementierung ist simpel. Es wird lediglich auf dem Poll Objekt der Status *PollState.Stopped* gesetzt. Alle weiteren Objekte behalten deren Referenzen auf das Poll Objekt und werden auch nicht gelöscht. Würden diese verloren gehen, so würden Folgefehler beim Öffnen von Umfrage-Mails sowohl beim Teilnehmer als auch beim Planer auftreten.

6.3 SMP Outlook Add-In

In der Regel werden Office Add-Ins entwickelt um Applikationen wie Outlook in kleinerem Mass zu erweitern. Ein Beispiel dafür wäre die Einbettung einer Google Maps Ansicht in den Kontakt Daten. Auf der Karte wäre der Wohnort des jeweiligen Kontakts ersichtlich. Add-In's können unterschiedlich entwickelt werden:

Outlook Web Access (OWA):
HTML5, JavaScript und CSS3 Programmierung. Benötigt weniger Arbeitsspeicher und kann sowohl im Web (Outlook.com) als auch auf dem Desktop (ab Outlook 2013) verwendet werden.

Klassisches Outlook Add-In:
Können nur auf dem Desktop ausgeführt werden und müssen installiert werden. Dies erschwert in der Regel zwar den Support es ist jedoch eine viel stärkere Integration in Office Applikationen möglich. Verwendung des .Net Frameworks und Zugriff auf Registerkarten und Steuerelemente von Outlook.

Um die Komplexität des SMP Workflows (siehe Abbildung 6.9) abzubilden, wird eine gute Integration in Outlook benötigt. Im Rahmen dieser Studienarbeit hat man sich für die Entwicklung eines klassischen Outlook Add-In's entschieden. Nicht zuletzt, war der Bedarf für die Verwendung des .Net Frameworks ausschlaggebend. Es ist jedoch nicht auszuschliessen, dass einzelne Schritte des Workflows, wie zum Beispiel das Einsehen von Umfrage Antworten, auch in einer OWA Lösung realisiert werden können. Die WCF Service Schnittstelle kann von beliebigen Clients angesprochen werden.

6.3.1 Outlook Modell

Für die Integration in Outlook musste zunächst analysiert werden wie Outlook Funktioniert und wie eigene Benutzeroberflächen eingebunden werden können.

MailItem Objekt
Das MailItem Objekt repräsentiert eine Mail-Nachricht in Outlook. Die Standard Properties wie Absender, Empfänger oder Betreff können beliebig mit eigenen UserProperties, ergänzt werden. Ein MailItem kann mit Programmcode zum Beispiel erstellt, gesendet und beantwortet werden.

Message Class
Jedes OutlookItem, so auch das MailItem, beinhaltet eine Message Class Property. Dieses gibt den Namen eines Outlook Formulars an, welches für das Öffnen oder Bearbeiten des Items angezeigt werden soll. Der Wert der Message Class Property beginnt immer mit 'IPM' und ist gefolgt vom einem Typ, auf welchem das Formular basiert. Der Standard Wert für Mails ist beispielsweise 'IPM.Note'. Somit weiss Outlook beim Öffnen eines MailItems, dass das Standard Formular für Mails geladen werden soll.

Für Benutzerdefinierte Formulare, wie sie auch bei SMP zum Einsatz kommen, kann die Message Class eines Typs spezialisiert werden. So verwenden MailItems, die von SMP generiert werden,

eine eigene Message Class (zum Beispiel 'IPM.Note.SMP'). Dadurch wird beim Öffnen eines solchen MailItems, zwar ein Fenster mit den bekannten Funktionen und Toolbars für Mails angezeigt, das eigentliche Formular jedoch (FormRegion), ist benutzerdefiniert.

FormRegions

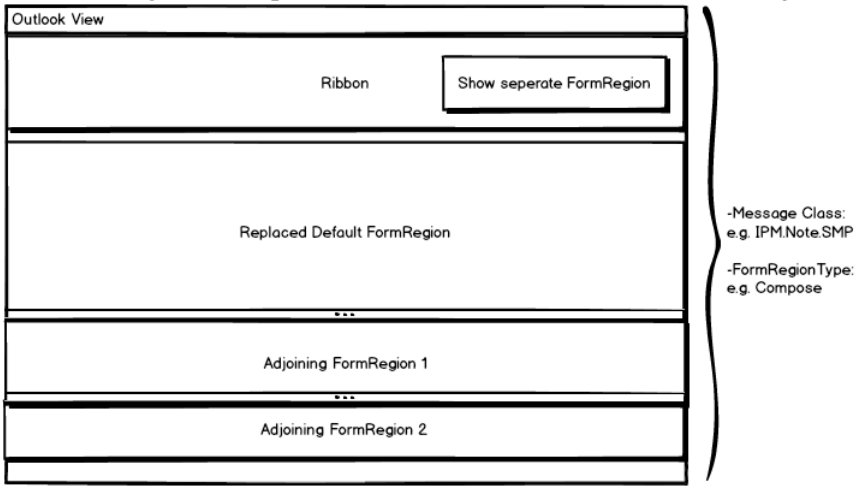
FormRegions sind Teile der Benutzeroberfläche, die zum Anpassen der Standardformulare von Outlook verwendet werden können. Ein solches Standardformular besteht jeweils aus einem Ribbon (Toolbar oben) und einer Default FormRegion (Formular unten). Ein solches Formular kann belibig mit weiteren FormRegions erweitert werden. Folgende Möglichkeiten stehen zur Auswahl:

- Adjoining: FormRegionwird unterhalb der DefaultForm Region angehängt.
- Seperate: FormRegionist eine zusätzliche Seite die dem Standardformular eingefügt wird. Der User kann über entsprechende Buttons im Ribbon zwischen den Seiten wechseln.
- Replacement: FormRegion Ersetzt die Default FormRegion eines Formulars komplett (nur möglich für Items mit spezialisierter MessageClass)

FormRegions werden anhand eines konfigurierbaren Manifest von Outlook geladen und dargestellt. Über dieses Manifest muss die FormRegion einer MessageClass zugewiesen werden. Somit entsteht eine Verlinkung zwischen einem OutlookItem welches ebenfalls ein MessageClass Property besitzt. Outlook benötigt nun noch die Information in welchem der folgenden Modi die FormRegion angezeigt werden soll (ebenfalls in Manifest konifgurierbar):

- Compose: FormRegion wird beim Erstellen oder Bearbeiten eines OutlookItems angezeigt.
- Read: FormRegion wird beim Öffnen (i.d.R. via Doppelklick aus dem Outlook Explorer) angezeigt.
- ReadingPane: FormRegion wird in der Reading Pane (Vorschau Ansicht) vom Outlook Explorer angezeigt.

Abbildung 6.7: Beispiel Outlook Formular mit Custom FormRegion



6.3.2 Verwendung von VSTO Contrib

Die Architektur des SMP Outlook Add-In basiert auf dem Model-View-ViewModel (MVVM) Pattern. Dies wurde mit der Verwendung der Open Source Library VSTO Contrib [1] realisiert. Mit VSTO Contrib lassen sich Office-Add-Ins im MVVM Stil entwickeln sowie Unit-Test und IoC-Container / DI verwenden. Es unterstützt Outlook, Word, Excel und Powerpoint 2007, 2010, 2013.

ViewModels

VSTO Contrib führt einen Ribbon Manager pro Formular Typ (Explorer, Read, Compose) ein. Via dem zugehörigen Ribbon XML können beliebig Buttons ein- oder ausgeblendet werden. Öffnet sich ein neues Fenster via Ribbon Manager, instanziert dieser einen DataContext in Form eines ViewModels. Im Falle SMP wird dem ViewModel direkt das von Outlook erzeugte Mail mitgegeben. Um diese Logik im Ribbon jedoch vom eigentlichen MVVM zu entkoppeln wird mit DI gearbeitet. IoC-Containerr werden voll unterstützt (Package Autofac [**?**]).

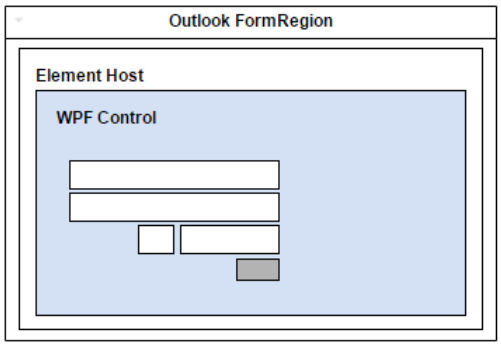
Listing 6.6: FormRegionFactory Konfiguration in WPF

```
1  public class ReadRibbon : OfficeViewModelBase, IRibbonViewModel
2  {
3    public void Initialised(object context)
4    {
5      item = ItemAdapter.FromObject(context).GetAsMail();
6      var containerBuilder = new ContainerBuilder();
7      containerBuilder.Register(c => item.As<MailItem>().SingleInstance();
8      container = RegisterViewModel<IPollViewModel, PollViewModel>(ref ↩
           containerBuilder);
9      Globals.ThisAddIn.ViewModelContainers.Add(context, container);
10   }
11 }
```

View

Wie in vorherigem Kapitel beschrieben lassen sich in Outlook die Standardformulare durch FormRegions bearbeiten. FomsRegions basieren standardgemäss auf der Windows Forms API für grafische Benutzeroberflächen. Um den Anforderungen des SMP Workflows gerecht zu werden und den Einsatz des MVVM Pattern zu ermöglichen, wird WPF als User Interface (UI) Technologie eingesetzt. Mithilfe des Element Host Controls kann WPF Code innerhalb einer Form Region geladen und angezeigt werden:

Abbildung 6.8: UIs mit WPF



Dieser Aufbau wird für jede eigene Ansicht benötigt und bringt viel generierten Code mit sich. Um dies zu verhindern wurde eine Factory für die Erzeugung einer WPF fähigen Form Region entwickelt. Es werden nur noch WPF Files generiert, bei deren Initialisierung ebenfalls ein FormRegion Manifest konfiguriert werden kann. Die Factory generiert anhand der schliesslich zur Laufzeit eine entsprechende Form Region die wie oben beschrieben die WPF View einbindet.

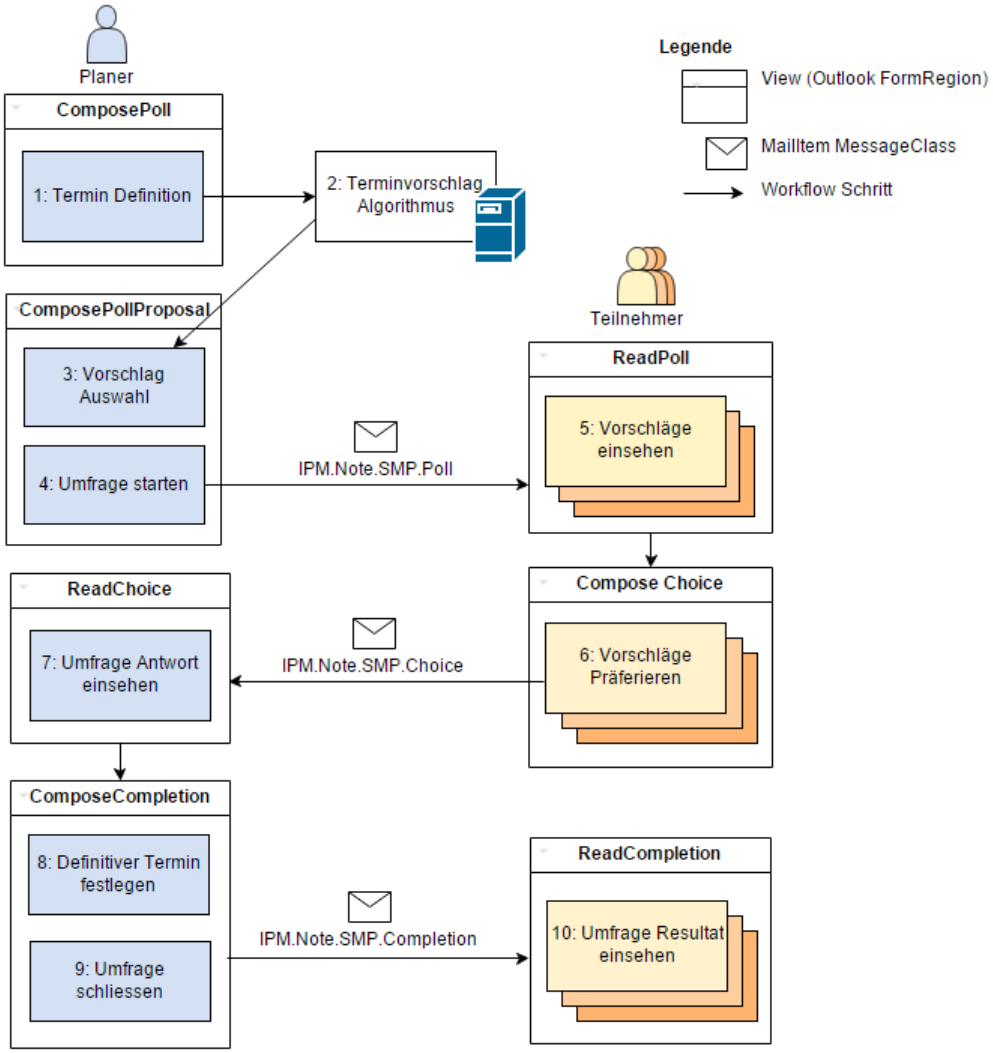
Listing 6.7: Konfiguration des FormRegion Manifests via eigener FormRegion Factory

```
1 public class ComposePoll
2 {
3   [FormRegionMessageClass("IPM.Note.SMP.Poll")]
4   [FormRegionName("Smp.Client.Outlook.Features.Poll.ComposePoll")]
5   public class FormRegionFactory : OutlookFormRegionFactory<ComposePoll, ↩
        IPollViewModel>
6   {
7     ..
8   }
9 }
```

6.4 SMP Workflow

Für die einzelnen Schritte im SMP Workflow werden unterschiedliche Benutzeroberflächen gefordert. In der Grafik 6.9 wird aufgezeigt welche FormRegions entwickelt wurden und welche Schritte darin abgebildet werden. Wie in Kapitel 5.2.3 beschrieben, wird bei personenübergreifenden Steps via Mail kommuniziert. Jede dieser Benutzeroberflächen unterliegt also einem *MailItem* und zwar mit einer von drei SMP spezifischen Message Classes. Die Namen der FormRegions geben zudem Auskunft in welchem Modus diese verwendet werden (Compose oder Read).

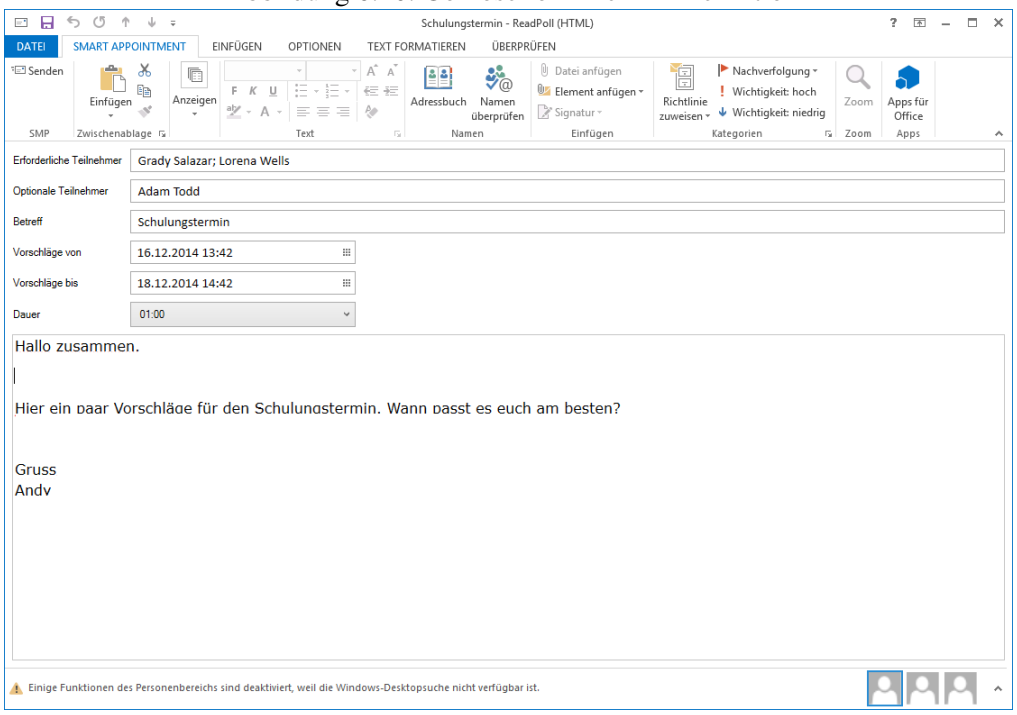
Abbildung 6.9: SMP Workflow mit Views



Schritt 1: Termin Definition

Das Starten einer Terminfindung ist direkt in Outlook integriert. Als Einstiegspunkt dient die Schaltfläche *New Smart Meeting* im Explorer Ribbon, der oberen Toolbar im Hauptmenü von Outlook. Beim Klick auf diesen Button wird ein neues MailItem der Message Class IPM.Note.SMP.Poll erzeugt und die entsprechende Bearbeitungsansicht ComposePoll wird sichtbar. Der Planer kann hier die im Lösungkonzept definierten Rahmenbedingungen für die Terminfindung eingeben. Bei der Eingabe von Teilnehmern wird für das Auflösen von Name zu Email-Adresse Outlook Funktionalität wiederverwendet. So ist es auch möglich, Namen via Adressbuch hinzuzufügen oder zu entfernen.

Abbildung 6.10: ComosePoll - Termin Definition



Das zugrundeliegende MailItem wird wie folgt abgefüllt:

Eingabe	**Mail Property**	
Organisator	Standard Mail Property	From
Geforderte Teilnehmer	Standard Mail Property	To
Optionale Teilnehmer	Standard Mail Property	CC
Subject	Standard Mail Property	Subject
Zeitspanne (von, bis)	UserProperty	StartTime, EndTime
MeetingDauer	UserProperty	Duration
Meeting Text	UserProperty	Body

Schritt 2: Terminvorschlag Algorithmus

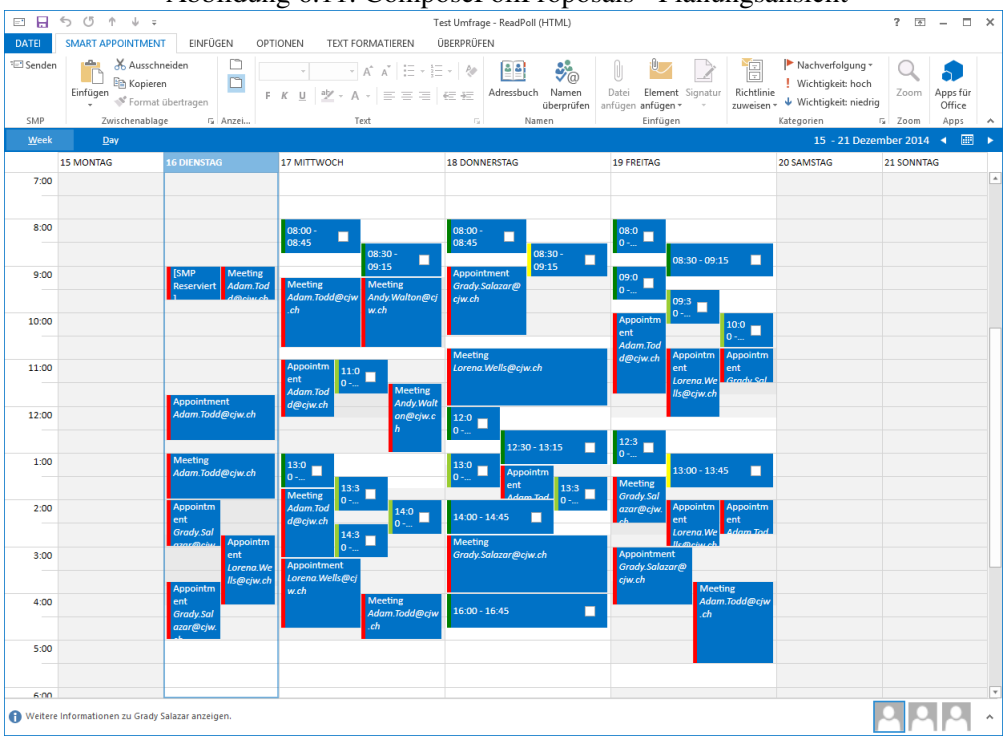
Der SMP Terminvorschlag Algorithmus wurde so definiert, dass er anhand den Termin Definitionen aus Schritt 1 (gekapselt in einem Objekt `AppointmentQueryRequest`) eben diese Schnittstelle ansteuert. Anschliessend werden die relevanten Informationen aus dem Resultat ausgelesen, optimiert und in ein eigenes Datenmodell abgebildet. Dieses Datenmodell beinhaltet eine Liste von Terminvorschlägen (namentlich `AppointmentSuggestions`) und eine Liste von sogenannten `Event`'s. `Event`'s repräsentieren einen besetzten Termin Slot eines Teilnehmers. Gekapselt werden alle Resultate wiederum im Objekt `AppointmentQueryResponse`.

Sobald also in der `ComposePoll`-Ansicht die Teilnehmer, Meeting-Dauer und Zeitspanne angegeben sind, wird im Hintergrund der WCF Aufruf `GetAppointmentProposal` auf den Planning-Service ausgelöst. Die Resultate des Requests werden im ViewModel Property `MeetingProposals` abgelegt sobald der Response einkommt. Für jede Änderung an einer der drei Anforderungen wird der WCF Call erneut ausgelöst und die Daten im ViewModel entsprechend aktualisiert. Somit hat der Planer die Möglichkeit die den Terminvorschlag Algorithmus nochmals zu justieren (Alternative Flow gemäss Lösungskonzept).

Schritt 3: Vorschlag-Auswahl

Der vorherige Schritt läuft asynchron im Hintergrund ab, der User nimmt den WCF Call nur indirekt wahr. Sobald der Planer die Termin-Definition bei Schritt 1 erledigt hat, kann er innerhalb des gleichen Fensters auf die separierte FormRegion `ComposePollProposals` wechseln. Sobald der asynchrone Calls zum Plannings-Service eine Rückgabe gibt, wird die Planungsansicht mit den erhaltenen Informatioenen aktualisiert. Die Planungsansicht selbst, ist wie im Lösungskonzept beschrieben, ein Kalender, welcher besetzte und vorgeschlagene Slots in Form von Blöcken anzeigt. Anstatt diese Kalenderansicht komplett von Grund auf zu programmieren, wurde das UI-Element `RadScheduleView` von der Telerik DevCraft Library [8] als Grundlage verwendet. Ein weiterer Grund für die Verwendung dieser Library ist die ausgereifte Usability der verwendeten Controls. Durch entsprechendes Anpassen der Themes konnte man diese Ansicht auch dem Outlook Style anpassen.

Abbildung 6.11: ComposePollProposals - Planungsansicht



Das Auswählen einer Menge dieser Terminvorschläge wurde mit Checkboxen gelöst. Dadurch ist schnell ersichtlich, dass mit den jeweiligen Blöcken interagiert werden kann. Sobald ein Terminvorschlag zu Umfrageauswahl gewählt wurde, wird im Hintergrund das `UserProperty - AppointmentSuggestions` im MailItem abgefüllt. Dies sind schliesslich die effektiven Terminblöcke die den Teilnehmern zur Auswahl geschickt werden. Alle anderen Terminvorschläge werden nach Schritt 3 nicht mehr verwendet und verworfen (da sie nur auf ViewModel gesichert sind). Es ist zudem möglich für mehr Übersicht auf eine Tagesansicht zu wechseln.

Der Planer hat während der Planungsphase jederzeit die Möglichkeit, über die Toolbar, zurück zur `ComposePoll` Ansicht zu wechseln und Workflow Schritte 1-3 zu wiederholen.

Schritt 4: Umfrage starten

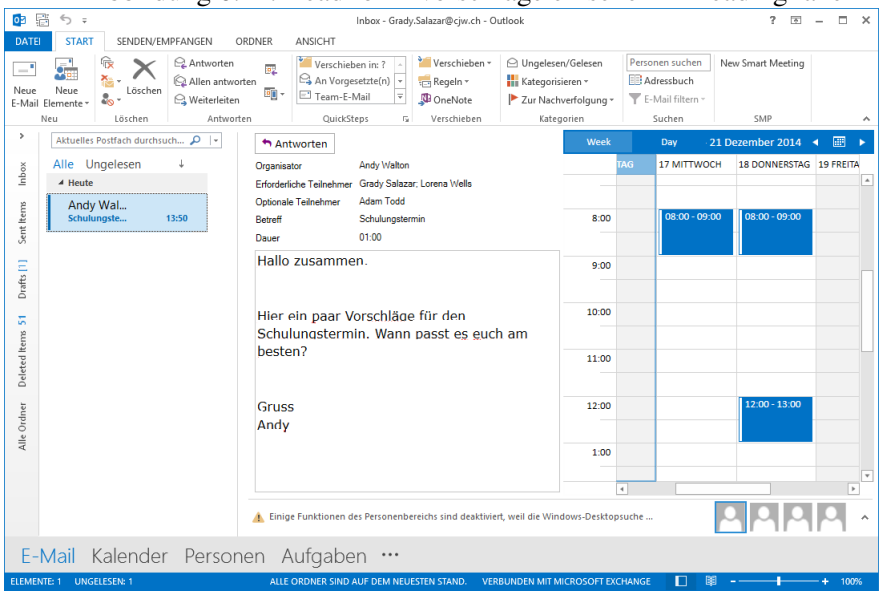
Die Umfrage wird ausgelöst, indem der Organisator auf den Button mit der Aufschrift “Senden” klick. Dabei wird im Hintergrund die Methode `CreatePoll` auf dem WCF Service aufgerufen.

Wie im Lösungskonzept beschrieben, werden dabei die Präferenzen für den Organisator hinzugefügt. Dies wird insofern erledigt, indem für jede `AppointmentSuggestion` eine `Choice` mit dem Organisator als `SMPAttendee` erstellt wird. Konkret wird dazu die Methode `CreateChoice` mit den entsprechenden Parametern aufgerufen. Ein weiterer Nebeneffekt ist die Eintragung der tentative Appointments, welche mittels der Methode `CreateTentativeAppointment` erstellt werden. Dabei wird die EWS Schnittstelle benutzt, in dem ein neues `Appointment` Objekt erstellt wird.

Schritt 5: Vorschläge einsehen

Nachdem der Planer eine Umfrage gestartet hat, erscheint bei allen Teilnehmern ein entsprechendes Mail im Posteingang. Anhand der MessageClass kann Outlook wiederum schliessen, dass es sich um ein SMP Umfrage Mail handelt. Somit wird beim Öffnen des MailItems sowohl in der ReadingPane durch einfaches Klicken oder aber in einem neuen Fenster durch Doppelklick, die ReadPoll Ansicht angezeigt.

Abbildung 6.12: ReadPoll - Vorschläge einsehen in ReadingPane

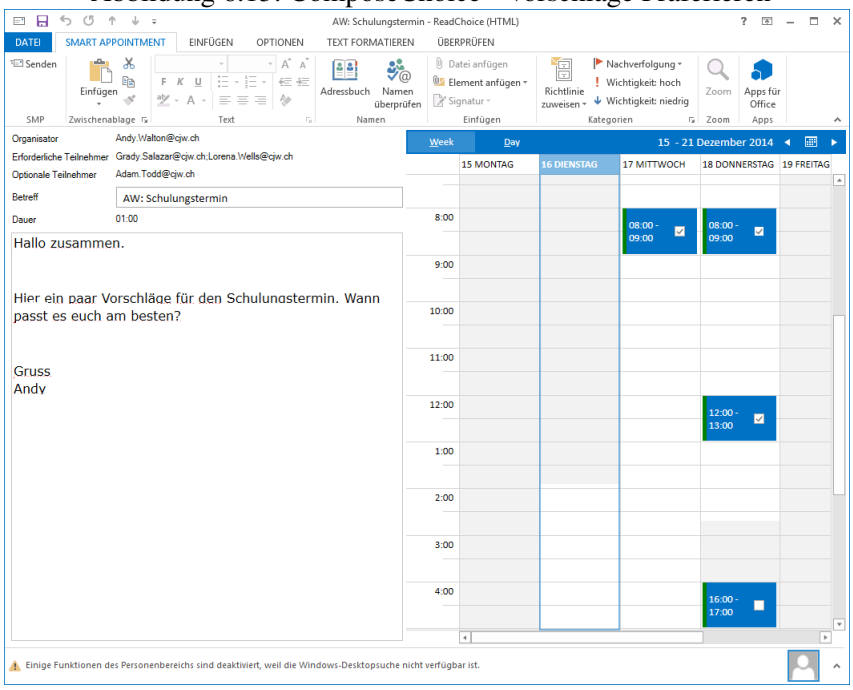


Die angezeigten Daten wurden direkt mit dem MailItem in UserProperties mitgeschickt. Der Teilnehmer kann in dieser Ansicht jedoch noch keine Auswahl tätigen.

Schritt 6: Vorschläge präferieren

Die Umfrage kann vom Teilnehmer nur dann ausgefüllt werden, wenn auf die Schaltfläche *Antworten* geklickt wird. Der Grund dafür ist, dass ein herkömmliches Mail Item in Verwendung ist, welches immer eine Read- und Compose-Ansicht vorgibt. Folglich ist man an diesen Workflow gebunden, sodass beim Klick eine Reply Message erzeugt mit der Message Class `Choice`.

Abbildung 6.13: ComposeChoice - Vorschläge Präferieren

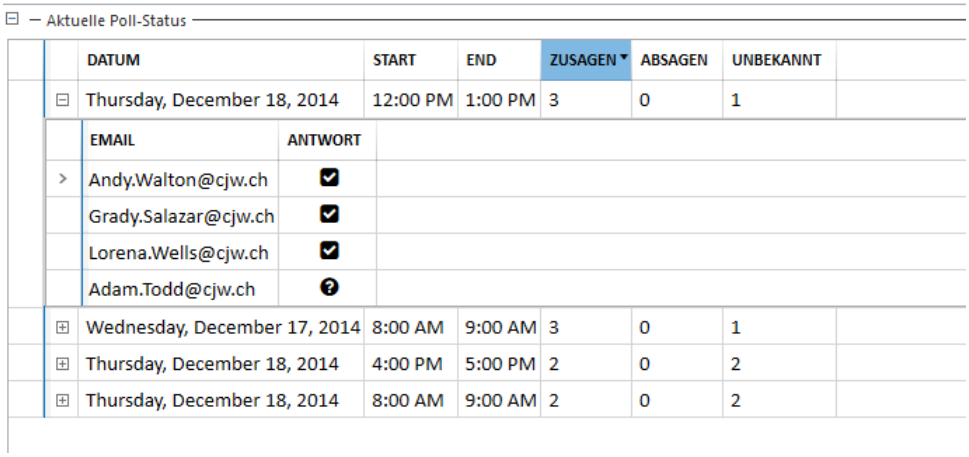


Nach dem die aufgeführten `AppointmentSuggestions` durch eine Selektion präferiert wurden, kann die Message abgeschickt werden. Dabei wird die WCF Service Methode `UpdateChoice` aufgerufen, welche wie im Lösungskonzept erwähnt, die Eintragung der tentative Appointments vornimmt und die Präferenzen persistiert.

Schritt 7: Umfrage Resultate einsehen

Einzelne Antworten erscheinen als neue Mails im Posteingang des Planers. Absender ist dabei jeweils der Teilnehmer welcher die Umfrage ausgefüllt hat. Der Planer sieht die Präferenzen der Teilnehmer für die jeweiligen Terminvorschläge. Die Mail ist ebenfalls eine Instanz der Message Class `Choice` und sieht somit optisch identisch derjenigen Nachricht aus, welche der Teilnehmer zuvor ausgefüllt hat. Der Unterschied liegt darin, dass die Präferenzen fix eingelockt sind und nicht geändert werden können vom Planer.

Abbildung 6.14: ReadChoice - Auschnitt der Gesamtübersicht

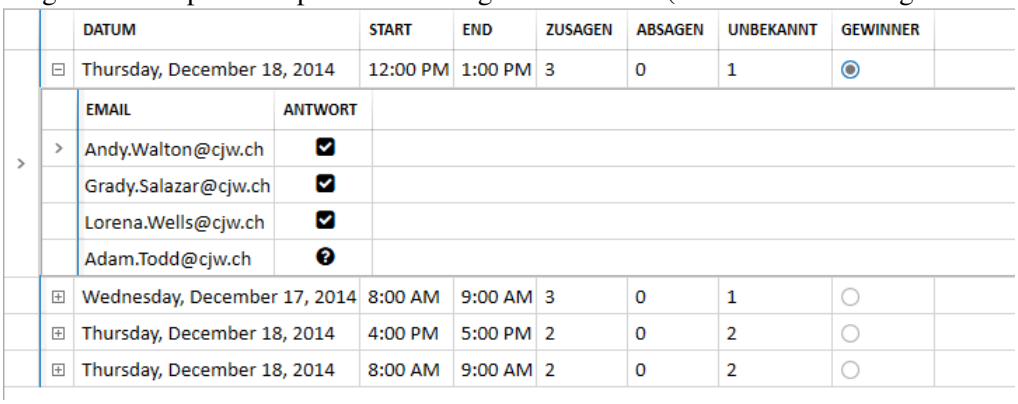


Anzumerken ist hierbei, dass es sich um eine statische Nachricht handelt und unter Umständen nicht den aktuellen Daten entsprechen muss. Es ist gut möglich, dass der Teilnehmer in der Zwischenzeit seine Präferenzen bereits wieder geändert hat. Alle eingegangenen Resultate laufen jedoch unter der gleichen Outlook Unterhaltungs-Id und werden somit gruppiert. Dabei ist zu oberst immer die aktuellste Antwort.

Schritt 8: Definitiver Termin festlegen

Um auf die Übersicht zu gelangen, wo der definitive Termin ausgewählt werden kann, muss eine Ausgefüllte Terminumfrage Nachricht (Message Class `Choice`) geöffnet werden. Dort findet man die Schaltfläche mit der Aufschrift *Umfrage Abschliessen*. Die dann erscheinende Übersicht ist wiederum eine Mail mit der Message Class `Completion`.

Abbildung 6.15: ComposeCompletion - Umfrage abschliessen (Gewinner kann ausgewählt werden)



Wird ein Termin ausgewählt und somit die Umfrage abgeschlossen, so wird die Service Methode `CompletePoll` aufgerufen, welche zuständig für das Löschen der tentative Appointments und Eintragen des festgelegten Termins ist. Um sicherzustellen, dass lediglich die von SMP erstellten tentative Appointments gelöscht werden, reicht es nicht aus, nur nach den Appointments während einem Zeitraum mit der entsprechenden Dauer des Appointments zu suchen. Stattdessen wurden die von Exchange generierten ID's der einzelnen Termine persistiert und können nun zur Überprüfung verwendet werden. Wie im Kapitel Persistierung (siehe 6.2.2) jedoch genannt, ist dies noch immer kein Gewähr, dass das Appointment noch besteht. Ist dies nicht der Fall, so muss davon ausgegangen werden, dass das Appointment gelöscht oder verschoben wurde und keine weiteren Massnahmen mer getroffen werden können. Ein Warnhinweis ist zwar denkbar, wurde aber in diesem Falle unterlassen.

Zum Schluss wird ein Mail der Message Class `Completion` an alle Teilnehmer versendet, wo die festgelegte `AppointmentSuggestion`, also der nun stattfindende Termin, vermerkt ist.

Schritt 9: Umfrageresultat einsehen

Die erhaltene Mail der Message Class `Completion` verfügt über einen statischen Inhalt. Die Umfrage ist hiermit abgeschlossen.

6.5 Nebenläufigkeits-Konzept

6.5.1 SMP Add-In

Damit das Add-In GUI bei Aufrufen zum SMP Service nicht einfriert und die Wartezeit für den Benutzer nicht unnötig erhöht wird, werden alle WCF Calls von einem vom GUI Thread unabhängigen, asynchronen Thread ausgelöst. Im Rahmen dieser Arbeit wird dabei immer davon ausgegangen, dass der Service beim Eingang mehrere Calls in kurzer Zeit, diese auch in der Aufruf-Reihenfolge abarbeitet. In einem weiteren Schritt könnte hier beispielweise mit einer Cancellation Token Struktur willkürem Verhalten in der Nebenläufigkeit entgegen gesetzt werden. Als weitere Ausgangslage wird angenommen, dass Service Aufrufe nur innerhalb des Firmen-Netzwerks ausgeführt werden und somit der SMP Service verfügbar ist. Es wäre eine zusätzliche Verfügbarkeitsprüfung vor dem Senden des eigentlichen Aufrufs denkbar.

Eingaben vom Benutzer werden clientseitig und synchron validiert. Ein Grund hierfür ist, dass die Response Time zum Server in der Regel zu viel Zeit in Anspruch nimmt. Insbesondere bei WCF Calls die mit dem Senden eines MailItems verknüpft sind (CreatePoll, UpdatePoll, etc.) soll der Client aus Performance Gründen nicht mehr auf eine serverseitige Validierung warten müssen. In diesem Fall werden vor dem eigentlichen WCF Aufruf die Eingaben überprüft und Fehlerfall den Sendevorgang abgebrochen (cancel = true). Sind die Eingaben gültig wird der WCF Call Asynchron ausgelöst. bei denen der Client nicht auf eine Response wartet

Listing 6.8: Vereinfachtes Beispiel der clienseitigen Validierung beim Senden eines MailItems

```
1  private void OnSend(ref bool cancel)
2  {
3      if (suggestions.Count == 0)
4      {
5        cancel = true;
6        MessageBox.Show("Keine Vorschlaege ausgewaehlt!")
7        return;
8      }
9    pollService.CreatePollAsync(args);
10 }
```

6.5.2 WCF Service-Verhalten

Um eine parallele Verarbeitung von mehrere Requests via WCF zu ermöglichen, wurde auf dem Service folgende Konfiguration vorgenommen.

Listing 6.9: Zugriff auf Benutzeridentität

```
1 [ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Multiple, ↩
      InstanceContextMode = InstanceContextMode.PerCall)]
2 public class Service : IService {
3   ...
4 }
```

CurrencyMode Multiple : Parallele Verarbeitung

InstanceContextMode PerCall : Erzeugt eine Instanz per Aufruf

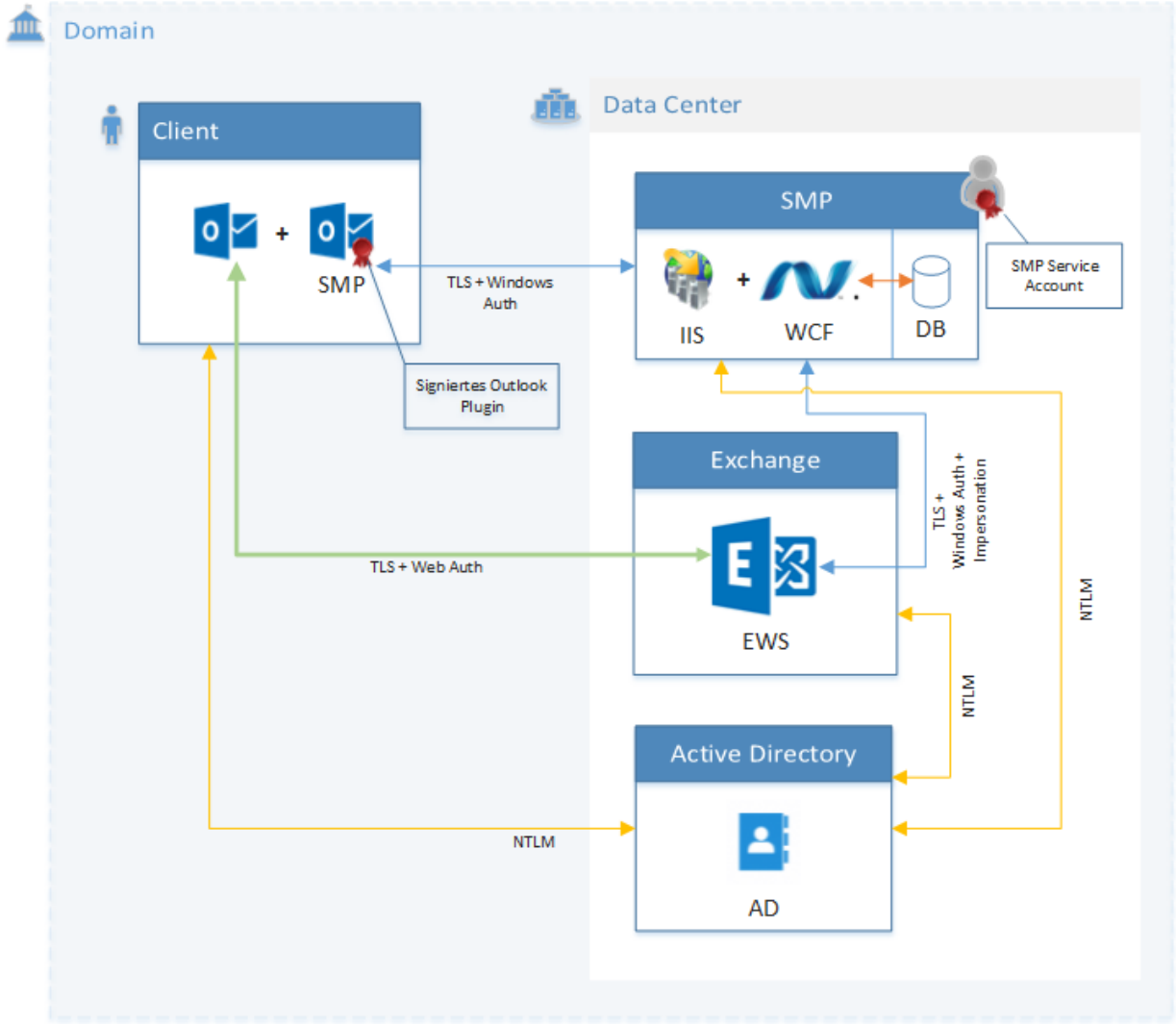
Via DI werden Abhängigkeiten wie z.B. DbContext verwaltet, sodass sie nicht bei jedem Aufruf neu erstellt werden müssen.

6.6 Security

In erster Linie muss eine gültige Authorizierung vom SMP Outlook Plugin bis hin zum Exchange-Server garentiert sein. Es dürfen keine *unintended data leakages* auftreten. Zudem soll die Kommunikation zwischen dem Client und dem Exchange-Service abhörbarsicher sein.

In Anbetracht des technischen Ecosystems an der FZAG liegt eine Integration mit bestehenden und bewährten Windows Authentisierungs- und Authorizierungkomponenten nahe. Die Integration verlangt einen hohen Initialaufwand wegen der Komplexität, sowie Interoperibilität. Im Gegenzug werden dafür die Schulungs-, Wartungs- und Operationsaufwände reduziert.

Abbildung 6.16: Übersicht der Sicherheitsmassnahmen



6.6.1 Client

Die Installation des Outlook-Plugins kann über das integrierte Microsoft Package-Management-System erfolgen. Dadurch sind auch automatische Installationen oder Updates an mehrere Personen bzw. Gruppen möglich.

Das Outlook-Plugin wird zudem mit einem Zertifikat signiert, sodass dieses vom Outlook geladen wrid. Das Zertifikat kann von jeden Trusted Certificate Authority bezogen werden (inkl. interne CA) [2].Der Einsatz des Plugins erfordert keine zusätzliche Berechtigungen des Users. Zwischen dem Client und der API wird die Kommunikation mittels TLS-Verschlüsselung geschützt.

6.6.2 API / WCF Services

Die Authentisierung der WCF Services erfolgt durch das integrierte Windows-Authentication Modul vom Internet Information Services (IIS), NT LAN Manager (NTLM): Authentisierung und Authorisierung erfolgt dabei automatisch, indem IIS den vom Client erhaltenen Token mit dem Active Directory abgleicht. Mit dieser Methodik erhält die API schliesslich die Windows-Credentials des aufzurufenden Users.

Listing 6.10: Zugriff auf Benutzeridentität

```
...
  var identity = ServiceSecurityContext.Current.WindowsIdentity;
  var sid = identity.User.Value;
...
```

Zugriffe auf die EWS Schnittstelle erfolgen über den SMP Service-Account mit Impersonation statt via Pass-Through Authentication. So können fehlende Zugriffsrechte besser im Applikationslogik abgehandelt werden.

Listing 6.11: Initialisierung des ExchangeService

```
  using (HostingEnvironment.Impersonate())
  {
    var exchangeService = new ExchangeService(ExchangeVersion.Exchange2013)
    {
      UseDefaultCredentials = true,
      Url = new Uri("http://example.com/"),
            ImpersonatedUserId = toImpersonate
    };

    return exchangeService;
  }
```

Mit der Impersonation des HostingEnvironment's werden die Credentials des ExchangeService mit jenen der Application Pool Identity gesetzt (hier: SMP Service Account). Durch Angabe des ImpersonationUserid werden nur jene Daten vom EWS geliefert, die der User gemäss der Exchange Rollen und Bentuzerverwaltung einsehen und ausführen darf.

6.6.3 Exchange-Server

Die Verbindung zum Exchange-Server ist hier ebenfalls mit einer Transport Layer Security (TLS v3)-Verschlüsselung geschützt. Da beide Servern im gleichen Domain befinden wird ein `NetworkCredential` (Authorizierung und Authentisierung via Active Directory) übergeben.
Im Active Directory muss ein User erstellt werden, der als Service Account dient. Es ist zu empfehlen den User keinen Profil zuzuweisen und kein Mailbox auf dem Exchange einzurichten.

Auf dem Exchange Server muss für den Service Account eine neue Rolle angelegt und Impersonationsrechte zugeteilt werden. Dies wird mit dem Exchange Powershell Toolbox erstellt:

Listing 6.12: Erstellung Benutzergruppe und Zuweisung Impersonationsrolle

```
1 [PS] >New-ManagementRoleAssignment  Name:SMPApplicationImpersonation -Role:↩
      ApplicationImpersonation -User:SMP
```

6.7 Verwendung externer Libraries

VSTO Contrib
: Mit VSTO Contrib lässt sich Office-Add-Ins im MVVM Stil entwickeln sowie Unit-Test und IoC / DI verwenden. Es unterstützt Outlook, Word, Excel und Powerpoint 2007, 2010, 2013.
Lizenz: MIT, Url: https://github.com/JakeGinnivan/VSTOContrib

Autofac
: Autofac ist ein IoC container und DI Manager.
Lizenz: MIT, Url: https://github.com/autofac/Autofac.

Telerik DevCraft
: Eine Sammlung von UI-Steuerelemente für .NET für die Herstellung von Software-Anwendungen. Diverse Controls werden für das UI verwendet.
Lizenz: Proprietär, http://www.telerik.com/

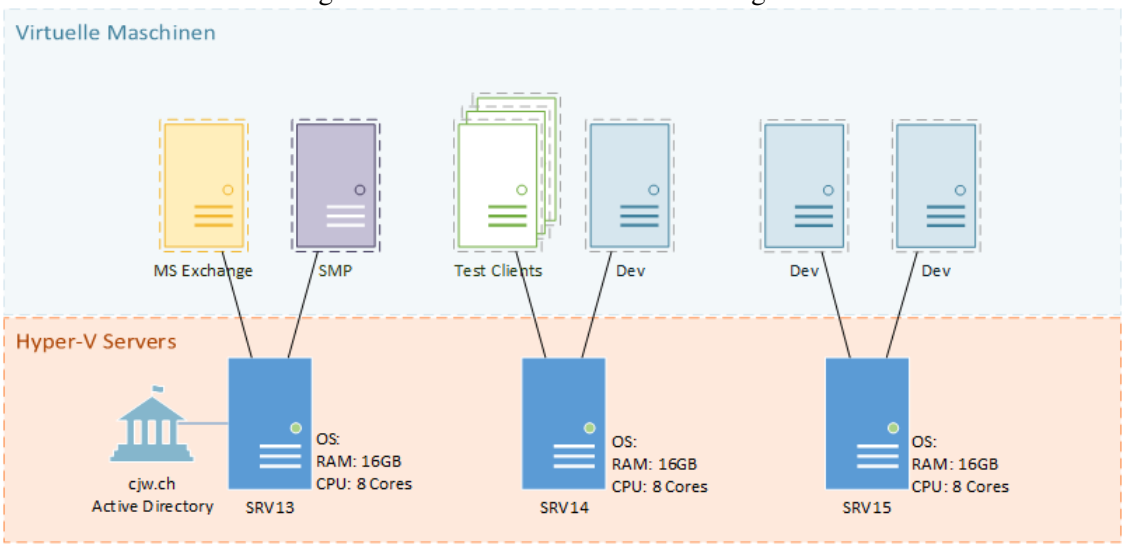
Kapitel 7

Auswertung

7.1 Einleitung

Für die Validierung der Funktionalität und Anforderungen von SMP wurde die Systemlandschaft von FZAG in einer Testumgebung nachgeahmt. Dabei wurden entsprechende Exchange und Applikationsserver mit integrierten Microsoft SQL Server 2014 *(Windows Server 2013, 4GB RAM, DualCore)* sowie drei Test Clients *(Windows 8 Enterprise, 2GB RAM, DualCore)* virtuell aufgesetzt. Für die Entwicklung wurden weitere drei Client konfiguriert *(Windows 8 Enterprise, ca. 6GB RAM, QuadCore)*.

Abbildung 7.1: Virtuelle Test- und Entwicklungsinfrastruktur



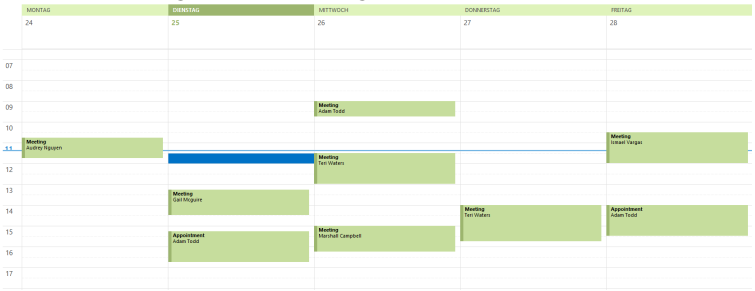
Für eine möglichst reale Umgebung wurde ein Programm entwickelt, welches via EWS Schnittstelle Testdaten in das System einspielt. Als Grundlage für die Generierung wurden die folgenden, von FZAG gelieferten Informationen, verwendet.

Im Schnitt hat die FZAG 48 Meetings pro Woche:

- ein Meeting dauert im Schnitt 1.1 Stunden.
- Normalerweise nehmen zwischen 2 - 8 Teilnehmer an einem Meeting teil.
- 14 Sitzungszimmer > 1 Sitzungszimmer ist für einen bestimmten Benutzerkreis eingeschränkt.

Effektiv generiert das Programm auf der Testinfrastruktur insgesamt 100 Users inklusive Exchange Mailbox. Zusätzlich werden auf zehn Räume als Ressource eingerichtet. In einem weiteren Schritt werden 48 Meetings pro Woche mit jeweils 2-8 Zufallsteilnehmer und Zufallsdauer erstellt. Start- und Endzeit müssen innerhalb von Bürozeiten liegen und zwischen 1200-1300 wird kein Meeting gestartet. Die Räume für die Meetings werden dabei gleichmässig ausgelastet.

Abbildung 7.2: User mit generierten Zufalls Terminen



7.2 Funktionalität und Integration

7.2.1 Workflow

Der Use Case *Termin finden* (siehe 4.3) wurde mit dem SMP Workflow (siehe Lösungskonzept 5.2) konzipiert und als Prototyp (siehe 6.4) abgebildet. Die beiden Komponenten, Planung und Umfrage, wurden auf die selbe Art und Weise in Microsoft Outlook integriert. Es wurde jeweils der Ansatz gewählt, sich auf die Basis eines Mail Items zu stützen. Der Use Case konnte somit erfolgreich nachgebaut werden.

Die Entscheidung, eine eigene Planungsansicht zu entwerfen anstelle auf dem bestehenden Terminplanung Assistenten aufzubauen (beschrieben im Lösungskonzept 5.2.1), wurde nebst den freien Gestaltungsmöglichkeiten auch mit der einfacheren Integration (Mail Item anstelle Calendar Item) begründet. Es zeigte sich, dass die freie Gestaltungsmöglichkeiten mit einem enormen Aufwand verbunden sind. Nicht nur bedarf das Entwickeln von WPF Templates seine Zeit, auch die konzeptionellen Überlegungen, auf welche Art die jeweiligen Ansichten optimal dargestellt werden sollen. Beispielsweise stellt sich im Nachhinein heraus, dass die Ansicht der Terminvorschläge bei einem Teilnehmer optisch sehr anspricht, doch der Kalender diejenigen Termin verdeckt, welche nicht in den Zeitraum der aktuellen Ansicht fallen. Auch in Sachen Funktionalität ist vom Terminplanung Assistenten einiges geboten, dass hätte verwendet werden können und somit zur Qualität der Terminfindung Resultate hätte beitragen können. Folglich kann man sagen, dass wenn die Integration der Planungskomponente in den Terminplanung Assistenten von Outlook insofern realisiert worden wäre, dies hinsichtlich Komfort, Einfachheit und Konzeption die bessere Variante darstellt.

Die Umfrage, also der Miteinbezug von Teilnehmern (beschrieben im Lösungskonzept 5.2.2), nach dem Ansatz einer klassischen Papier Umfrage basierend auf einem Mail Item konnte zufriedenstellend gelöst werden. Der gesamte Workflow konnte soweit integriert werden, dass dieser sich in Outlook natürlich verhält. Auch konzeptionell wurde eine optimierte und effiziente Vorgehensweise ausgearbeitet. Einzig wäre eine Gesamtübersicht aller Umfragen wünschenswert. Aktuell kann nur über eine vom Teilnehmer ausgefüllte Umfrage auf die Übersicht der der Umfrage referenziert werden. Dieser Weg erscheint noch etwas harzig und auf den ersten Blick nicht ersichtlich.

7.2.2 Alternative Flow

Die im Lösungskonzept beschriebenen Alternative Flows (siehe 5.5, 5.6 und 5.7) konnten im SMP Prototypen abgebildet werden. Dafür wurden entsprechende Optionen im GUI zur Verfügung gestellt. Es sind jedoch noch weitere Alternative Flows denkbar, die aus Zeitgründen nicht in den Umfang dieser Arbeit eingeflossen sind:

- Umfrage abbrechen: Planer kann zur jederzeit eine Umfrage abbrechen. Die Teilnehmer werden daraufhin informiert. Die geschickten Präferenzanfragen müssten dabei wieder gelöscht werden können. Serverseitig wurde dieser Flow realisiert (siehe 6.2.5), auf Seiten des Add-Ins wurde jedoch im Rahmen dieser Arbeit noch keine Lösung realisiert.
- Umfrage anpassen nachdem diese ausgelöst wurde: Planer kann währenddem die Umfrage läuft, Änderungen vornehmen. Die Teilnehmer werden entsprechend informiert.

7.2.3 Raumfindung

Weder in der Planungs- noch in der Umfrage Komponente wurde in Rahmen dieser Studienarbeit die Raumfindung implementiert. Die Kombination der Probleme Terminfindung und Raumfindung stellte sich als grösseren Aufwand, als Anfangs angenommen, dar. Es muss eine Lösung gefunden werden für den gegenseitigen Ausschluss von freien Räumen und Terminen der Teilnehmern. Der Mehraufwand um diese Problematik zu lösen, konnte im verfügbaren Zeitbudget nicht realisiert werden.

7.3 Komplexität

Der gesamte SMP Workflow konnte, wie in den Anforderungen beschrieben, mit einer Add-In und Backend Komponente realisiert werden. Es wurden folglich keine weiteren Komponenten verwendet, was das Produkt in einem Microsoft Umfeld gut dastehen lässt.

Die Add-In Komponente in sich selbst weisst jedoch eine etwas erhöhte Codekomplexität auf. Die von Outlook nicht vorgesehenen Funktionalität, welche im Rahmen dieses Projektes implementiert wurden, führten dazu, dass Umfangreiche Änderungen an den Mail Items vorgenommen werden mussten. Durch geschickten Gebrauch von Polymorphismus wurden zwar Massnahmen ergriffen, doch bedarf es noch immer ein überproportialer Umfang an Code für die einzelnen Ansichten im Workflow. Ein weitere Nachteil, ist die Verwendung der kostenpflichtigen Bibliothek *Telerik* [8]. Mit dieser können Anteile

hinsichtlich der Kalenderansichten ausgelagert werden, sollte jedoch für eine allfällige Weiterentwicklung abgesetzt werden.

Durchaus positiv zu Bewerten ist aber das Einhalten der Security Anforderungen, welche mittels dem Windows Authentication Modul realisiert werden konnten. Wie im Kapitel Security (siehe **??**) beschrieben, bringt dies zwar einen erhöhten Initialaufwand mit sich, stellt sich in der darauf folgenden Benutzung als äusserst benutzerfreundlich dar.

7.4 Performance

Die Performance spielt in dieser Applikation insofern eine wichtig Rolle, dass diese sich weder auf die Effizienz der Mitarbeiter auswirken, noch deren Geduld auf die Probe stellen darf. Dazu wurden manuelle Testdurchläufe mit verschiedenen Eingabewerten durchgeführt.

Folgende Probleme sind aktuell vorhanden:

- Erster Start von SMP dauert lange.
- Die Ansichten mit enthaltenen Kalenderansichten (*Telerik*) verhält sich träge.
- Der erste Verbindungsaufbau zum Server dauert lange. Das ist spürbar, wenn zum ersten Mal nach dem Start des Clients eine Umfrage abgeschickt wird. Trotz Asynchronität bleibt das Fenster noch offen bis die Abfrage losgeschickt wurde.
- Es ist eine leichte Verzögerung bei der Abfrage *UpdateChoice* festzustellen. Grund dafür ist, dass pro Choice ein separater Call abgeschickt wird anstelle von einem gebündelten Aufruf aller Choices.
- Allgemein ist eine Einbusse beim öffnen der Fenster spürbar. Vermutlicher Grund dafür wird das Aufsetzen des Data Context durch *VSTO Contrib* [1] sein.

Die Verzögerungen der Benachrichtigungen vom Planer zu Teilnehmer und umgekehrt sind von SMP unabhängig zu betrachten. Dies hängt von der Performance Exchange Server bzw. Netzwerks ab.

Der Terminfindung Algorithmus, bzw. die Exchange Abfragen sind erstaunlich performant. Es wurden folgende 8 Tests durchgeführt und ausgewertet: Die Tests wurden so aufgebaut, dass diese immer mehr

	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**
Anzahl Teilnehmer	3	10	10	10	30	30	10	150
Zeitspanne (Von-Bis)	3d	1w	2w	4w	4w	8w	6m	1w

Last verursachen sollen und folglich abzuwägen ist, in wie fern die Parameter Auswirkungen auf die Antwortzeiten haben. Dabei wurde festgestellt, dass weder die Anzahl der Benutzer noch die Grösse der Zeitspanne eine signifikante Auswirkung auf die Antwortzeiten haben.
Durch die stetige Erhöhung der Werte wurden wir mit den Limiten (T7, T8) der Exchange API (*GetUserAvailabilty*) vertraut gemacht:

- Maximale Anzahl Teilnehmer: 100
- Maximales Time-Window: 62 Tage

Eine weitere Grenze (T6) stellt die limitierte Grösse der Message dar. Ab circa 40 tägiger Timespan kann es zu Problemen mit der MaxReceiveMessageSize *(WCF Setting, Standard 65536 [7])* kommen. Diese kann jedoch in der WCF Konfiguration des Web-Configs erhöht werden.

Eine bereits vorhanden Optimierung stellt sicherlich der Aufruf der Service Schnittstelle direkt nach dem der User die Eingaben an der Zeitspanne vornimmt dar. Dies erhöht die Antwortzeit aus Sicht des Users, kann aber auch verwirrend wirken, da aktuell kein Ladebalken bzw. Information, dass bereits eine Abfrage in Gang ist, angezeigt wird.

Es kann von folgenden Bottlenecks gesprochen werden:

- WCF Verbindungsaufbau
- GUI Telerik Library

Wobei sich das Problem mit dem Verbindungsaufbau durch geschicktere Instanziierung, beispielsweise direkt bei der Öffnung des Clients, umgehen liesse.

7.5 Qualität

Um die Qualität des Systems gewährleisten zu können, wurden folgende auf Seiten des Servers ein Testkonzept erstellt, welches den kompletten Umfrageablauf mit Testdaten abhandelt (siehe Anhang **??**). Dieses Konzept wurde mittels funktionalen Tests implementiert.

Auf Seiten des Clients wäre dank des Einsatz des MVVM-Pattern ebenfalls ein effizientes Testing möglich. Die Realisierung wurde jedoch als weniger Wichtig eingestuft. Die Ausarbeitung eines Prototyps, mit welchem herauszufinden ist, wie die grafische Darstellung der verschiedenen Ansichten für eine Umfrage zu optimieren ist, wurde mit höherer Priorität eingestuft. Als zusätzlicher Ansatz den Client zu testen, wäre eine Simulation von Benutzereingaben sinnvoll.

7.6 Portabilität

Die Möglichkeit einer einfachen Auslieferung des Outlook Add-In's ist eine Anforderung seitens FZAG. Folgende Überprüfungen wurden gemacht:

- Für die Verteilung von Installation-Files und das Eintragen von den benötigten Registry-Keys (pro FormRegion und Add-In ist ein Eintrag erforderlich), kann mit InstallShield ein .msi Package generiert werden.
- Das .msi Package verteilt unter Anderem ein VSTO Deployment Manifest welches für die effektive Installation des Add-Ins benötigt wird.
- Via dem mitgelieferten Konfiguration-File `Smp.Client.Outlook.dll.config` sind nachträglich einfache Änderungen wie der Pfad des SMP Server ohne erneute Kompilierung möglich.

7.7 Usability

Zum aktuellen Zeitpunkt liegen seitens FZAG noch keine Auswertungen der Benutzeroberfläche vor. Es wird jedoch als störend empfunden, dass der Planer, um den Stand der Umfrage einsehen zu können, als Einstieg hierfür jeweils ein Antwort Mail eines Teilnehmers benötigt.

Kapitel 8

Diskussion

8.1 Resultat und Zielerreichung

FZAG ersuchte eine Möglichkeit für mehr Effizienz bei der Findung von Terminen für interne Meetings. Insbesondere sollte dabei das Miteinbeziehen von Präferenzen der Teilnehmer im Zentrum stehen. Durch eine Integration in Outlook soll dabei die Verwendung von externen Tools wie Doodle vermieden werden können.

Im Rahmen dieser Arbeit wurde zunächst ein konzeptioneller Workflow für das Finden eines Termins definiert. Dieser ist zugeschnitten auf die Systemlandschaft von FZAG, es ist jedoch auch denkbar, diesen Workflow in einem anderen technischen Umfeld zu implementieren. Wie im Kapitel Architektur beschrieben, wurde anschliessend ein Prototyp entwickelt welcher die Machbarkeit des SMP Workflows aufzeigt. Dabei wurde besonders auf die nahtlose Integration in Outlook 2013 und einer sicheren Anbindung zum Exchange Server Wert gelegt. Der Prototyp wurde fortlaufend in einer Testumgebung, welche dem IT-Umfeld von FZAG nachgeahmt ist getestet.

Damit zeigt der SMP Prototyp die technische Machbarkeit einer Terminfindung-Lösung, die auf die Systemlandschaft von FZAG zugeschnitten ist, auf. Es kann gezeigt werden, dass mit der entwickelten Lösung das bestehende Berechtigungssystem und Datenschutzbestimmungen eingehalten werden können. Zudem wurde FZAG die Möglichkeiten für eine Outlook 2013 und Exchange Server Integration aufgezeigt. Diese kann auf Rechnern der Mitarbeiter als isoliertes Paket verteilt werden. Das Backend muss auf einem internen Server installiert werden.

8.2 Offene Punkte

Folgende Schritte konnten innerhalb der Arbeit nicht fertiggestellt werden (Gründe dafür siehe Kapitel Auswertung 7):

Zentrale Übersicht für Umfragen
Es muss eine zentrale Ansicht dargeboten werden, in welcher der Planer zu jeder Zeit seine laufenden Umfragen überwachen kann. Da der SMP Workflow komplett auf Basis von Mails abgebildet wurde, kann der Planer aktuell nur durch das Öffnen einer Teilnehmer-Antwortnachricht Einsicht über die jeweilige Umfrage erhalten.

Raumfindung
Die Verfügbarkeit von Räumen wird aktuell nicht in den Workflow mit einbezogen.

Umfrage Stoppen
Der SMP Prototyp bietet dem Planer keine Möglichkeit eine laufende Umfrage abzubrechen.

Ausarbeitung GUI
Die Benutzeroberfläche kann noch nicht als komplett ausgereift erachtet werden. Teilweise ist nicht klar, ob ideale Darstellungsweisen gewählt wurden. Dazu sind weitere Abstimmungen mit FZAG nötig.

Limitierungen bewältigen
Sofern Bedarf vorhanden, müssen die beschriebenen Limitierungen der EWS Schnittstelle durch gezielte Aufteilung von Schnittstellen-Anfragen umgangen werden.

8.3 Ausblick

Das Resultat der Arbeit ist ein Prototyp, welcher eine Möglichkeit zur Terminfindung aufzeigt. Dieser muss zunächst zu einem produktionsreifen Produkt ausgearbeitet werden. Dazu werden als nächste Schritte interne Tests sowohl hinsichtlich Usability und Funktionalität, als auch bezüglich der Integration in die interne Umgebung benötigt. Abhängig davon kann der Prototyp weiter ausgebaut und verbessert werden.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Jake Ginnivan. VSTO Contrib. https://github.com/JakeGinnivan/VSTOContrib. [Online; accessed 30-Sep-2014].

[2] Microsoft Inc. ClickOnce Deployment and Authenticode. http://msdn.microsoft.com/en-us/library/ms172240(v=vs.90).aspx. [Online; accessed 15-Dez-2014].

[3] Microsoft Inc. EWS, Klasse Appointment. http://msdn.microsoft.com/en-us/library/microsoft.exchange.webservices.data.appointment_members(v=exchg.80).aspx. [Online; accessed 15-Nov-2014].

[4] Microsoft Inc. EWS, Klasse AvailabilityOptions. http://msdn.microsoft.com/en-us/library/microsoft.exchange.webservices.data.availabilityoptions_members(v=exchg.80).aspx. [Online; accessed 15-Nov-2014].

[5] Microsoft Inc. EWS, Klasse GetUserAvailabilityResults: User Verfügbarkeit. http://msdn.microsoft.com/en-us/library/microsoft.exchange.webservices.data.getuseravailabilityresults_members(v=exchg.80).aspx. [Online; accessed 15-Nov-2014].

[6] Microsoft Inc. EWS, Methode GetUserAvailability: User Verfügbarkeit. http://msdn.microsoft.com/en-us/library/ee344039.aspx. [Online; accessed 15-Nov-2014].

[7] Microsoft Inc. MaxReceiveMessageSize. http://msdn.microsoft.com/en-us/library/system.servicemodel.basichttpbinding.maxreceivedmessagesize%28v=vs.100%29.aspx. [Online; accessed 13-Dez-2014].

[8] Telerik. Eine Sammlung von UI-Steuerelemente für .NET für die Herstellung von Software-Anwendungen. Lizenz: Proprietär. http://www.telerik.com/. [Online; accessed 01-Dez-2014].

Glossar

Appointment

Kalendareintrag mit einem User.. 64

Autofac

Autofac ist ein IoC container und DI Manager. Lizenz: MIT. 39, 64

DI

Dependency Injection, deutsch: Abhängigkeit Injektion. Reglementiert die Abhängigkeiten eines Objekts zur Laufzeit.. 39, 50, 53, 64, 65

Doodle

Webdienst zur Findung von Terminen. . 13, 64

DTO

Data Transfer Objects, bündelt mehrere Daten in einem Objekt, sodass sie durch einen einzigen Programmaufruf übertragen werden können. 33, 34, 62, 64

EWS

Exchange Web Services. 31, 34, 35, 45, 52–54, 61, 64

FZAG

Flughafen Zürich AG. 2, 9, 11–13, 51, 54, 55, 58, 60, 61, 64

GUID

Globally Unique Identifier. 35, 64

HSR

Hochschule für Technik Rapperswil. 11, 64

IIS

Internet Information Services. 52, 64

InstallShield

Proprietäres Installationsprogramm von Flexera Software für Windows. Einfache Erstellung von Installation-Wizards zur Verteilung von Software.. 58, 64

IoC

Inversion of Control, deutsch: "Steuerungsumkehr". Inversion of Control wird verwendet, um die Modularität des Programms zu erhöhen und macht dies erweiterbar.. 39, 53, 64, 65

Meeting

Kalendareintrag mit mehrere Teilnehmer/User und evtl Räume. Ein Teilnehmer ist als Organizer definiert. Teilnehmer können als optional eingeladen werden.. 64

MSI

Microsoft Installer. 12, 64

MVVM

Model-View-ViewModel. 39, 40, 64

NTLM

NT LAN Manager. 52, 64

OWA

Outlook Web Access. 37, 64

SA

Studienarbeit. 31, 64

SID

Ein Security Identifier, kurz SID, ist ein eindeutiger Sicherheits-Identifikator, den Microsoft Windows NT automatisch vergibt, um jedes System, jeden Benutzer und jede Gruppe dauerhaft zu identifizieren.. 64

SMP

Smart Meeting Planer. 3, 9, 11–13, 15, 17–21, 24, 29–32, 37, 39–41, 43, 45, 48, 49, 51, 53–57, 60–62, 64

TLS v3

Transport Layer Security. 53, 64

UI

User Interface. 40, 53, 62, 64

VSTO

Visual Studio Tools for Office. 3, 39, 58, 64

WCF

Windows Communication Foundation. 3, 29, 32, 37, 43, 45, 46, 50, 52, 58, 64

WPF

Windows Presentation Foundation, ist ein Grafik-Framework und Teil des .NET Frameworks von Microsoft.. 39, 40, 55, 62, 64