HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

zühlke
empowering ideas

Bachelorarbeit

Offerten-Generator

Hochschule für Technik Rapperswil

Abteilung Informatik

Herbstsemester 19

Autor:	Muriel Thévenaz
Betreuer:	Prof. Dr. Luc Bläser
Projektpartner:	Christian Moser, Zühlke Engineering AG
Experte:	Dr. Janos Zatonyi, Varian Medical Systems Imaging Laboratory GmbH
Gegenleser:	Prof. Dr. Stefan Keller

Abstract

Um an neue Aufträge zu kommen, muss *Zühlke Engineering AG*, führendes schweiz- und europaweites Beratungsunternehmen im Bereich Software Engineering und Produkt-Innovation, regelmässig offerieren. Ziel der vorliegenden Bachelorarbeit war, einen Generator zu entwickeln, um das Prozess des Offertenschreibens bei *Zühlke* zu beschleunigen. Der Generator wurde als .NET Bibliothek implementiert. Er arbeitet mit übergrossen Power Point Vorlagen und generiert Power Point Dokumente als Output. Die Logik zur Auswahl der passenden Inhalte durch den Generator wird direkt in der Vorlage gekennzeichnet. Dazu wurde eine möglichst schlichte und intuitive Syntax entwickelt. Jede Vorlage kann mit drei Arten von generischen Bausteinen aufbereitet werden: Platzhalter, Kriterien und Verweise auf Datensätze. Wird eine Vorlage ausgewählt, scannt der Generator die Vorlage und sucht nach allen möglichen Bausteinelementen. Diese werden vom Generator also dynamisch erkannt. Gesteuert wird der Generator über eine Webapplikation (Prototyp). In den normalen, einfachen Fällen kann der Generator 60% bis 70% der Schlussofferte automatisch erzeugen. Umso stärker weicht eine Offerte vom Normalfall ab, desto weniger kann der Generator aber automatisch übernehmen - was zu erwarten war.

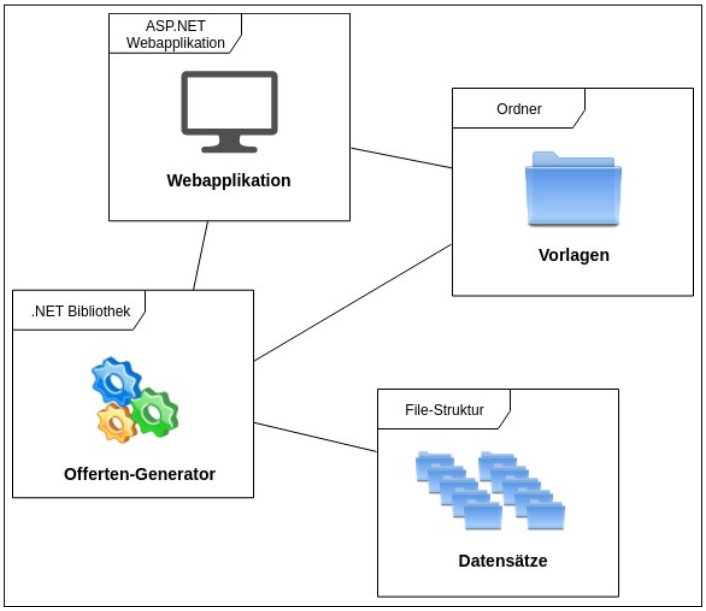
Management Summary

Zühlke Engineering AG ist ein führendes schweiz- und europaweites Beratungsunternehmen im Bereich Software Engineering und Produkt-Innovation. Um an neue Aufträge zu kommen, muss *Zühlke* regelmässig offerieren: Jedes Jahr produziert die Firma über 500 Offerten. Damit jede Offerte den spezifischen Bedürfnissen des Kunden und Produktes gerecht wird, wird auch viel Zeit und Energie in das Verfassen der Offerten investiert. Ziel der vorliegenden Bachelorarbeit war, einen Generator zu entwickeln, um das Prozess des Offertenschreibens bei *Zühlke* zu beschleunigen. Als Output für den Generator wurde eine Power Point Präsentation erwartet. Vom Anfang an wichtig war, dass die automatisch generierte Offerte kein fertiges Dokument ist, sondern nur eine erste Fassung der Offerte darstellt, welche von Mitarbeitenden jeweils manuell angepasst und erweitert wird.

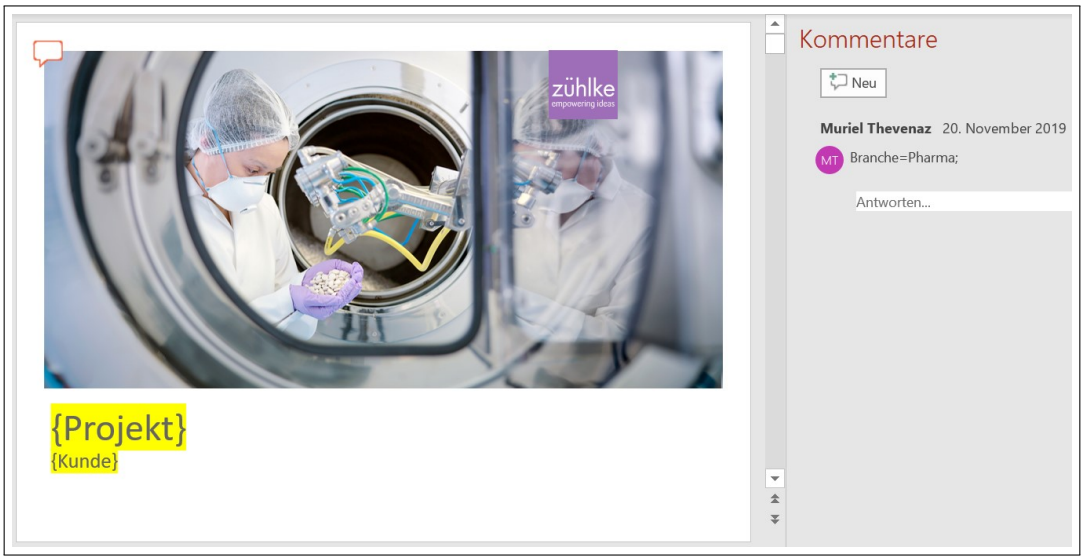
Zu Projektbeginn wurden zwei Hauptentscheide gefällt: Erstens soll der Generator möglichst wenig mit verstreuten Inhalten und stattdessen mit einer übergrossen Power Point Vorlage arbeiten, aus welcher überflüssige Inhalte gelöscht werden; zweitens soll die Logik zur Auswahl der passenden Inhalte direkt in der Vorlage gekennzeichnet werden. Dazu wurde eine möglichst schlichte und intuitive Syntax entwickelt. Jede Vorlage kann mit drei Arten von generischen Bausteinen aufbereitet werden: Platzhalter (direkt im Text), Kriterien (anhand welchen bestimmt wird, welche Inhalte aus einer Vorlage gelöscht werden sollen und welche nicht) und Verweise auf Datensätze (z.B. Informationen über Mitarbeitende, Referenzprojekte, usw.). Wird eine Vorlage ausgewählt, scannt der Generator die Vorlage und sucht nach allen möglichen Bausteinelementen. Diese werden vom Generator also dynamisch erkannt. Die Vorlagen können dementsprechend stetig angepasst, und allfällige Fehler sofort korrigiert werden.

Gesteuert wird der Generator über eine Webapplikation (Prototyp). Der Generator kann in eine Offerte auch Datensätze einfügen. Zurzeit funktioniert dies jedoch nur, wenn die entsprechenden Datensätze in Form von Text- und Bild-Dateien in einer Ordnerstruktur gespeichert sind.

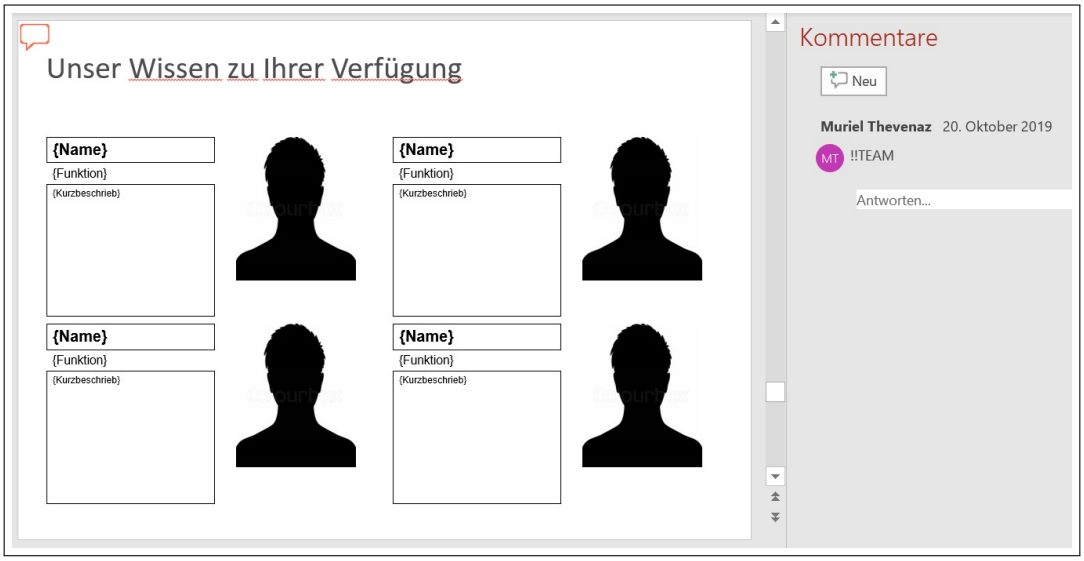
Um den Generator anhand von konkreten Beispielen zu testen, wurden fünf Offerten von *Zühlke* ausgewählt und unter die Lupe genommen. Ziel des Tests war zu ermitteln, welcher Anteil jeder Offerte schätzungsweise (in Prozenten) durch den Generator hätte automatisch generiert werden können. Fasst man die Ergebnisse des Tests zusammen, lässt sich aussagen, dass in den normalen, einfachen Fällen der Generator 60% bis 70% der Schlussofferte automatisch erzeugen kann. Umso stärker weicht eine Offerte vom Normalfall ab, desto weniger kann der Generator aber automatisch übernehmen - was auch zu erwarten war.



***Einfaches Architekturdiagramm**: Der Generator arbeitet mit einer vom/von der BenutzerIn ausgewählten Power Point Vorlage und wird über eine Webapplikation gesteuert. Er ist auch in der Lage, Datensätze aus Text- und Bild-Dateien in die Offerte einzufügen.*



***Beispiel-Slide aus einer Vorlage**: Die Slide enthält zwei Platzhalter im Text (gelb markiert) und ein Kriterium (im Kommentarfeld), anhand welches ausgewählt werden kann, ob die Slide in der Offerte bleibt oder vom Generator gelöscht wird.*



***Beispiel-Slide aus einer Vorlage**: Im Kommentarfeld der Slide wird auf eine Datensatz-Art verwiesen (hier „Team“). Die Slide ist speziell aufbereitet, um mit Informationen aus ausgewählten Datensätzen (Name, Funktion, usw.) gefüllt werden zu können. Werden mehr als vier Datensätze vom/von der BenutzerIn ausgewählt, wird die Slide einfach so oft wie nötig kopiert, damit alle ausgewählte Datensätze in die Offerte eingefügt werden können. Überflüssige Inhalte werden gelöscht.*

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einleitung

Seit über 50 Jahren engagiert sich *Zühlke Engineering AG* im Bereich der Produkt-Innovation und seit mehr als 45 Jahren entwickelt die Firma massgeschnittene Software-Lösungen für Kunden. Heute zählt *Zühlke* über 1000 Mitarbeitende und ist in 14 verschiedenen Standorten - v.a. in der Schweiz und in Deutschland - tätig.[13] Mit etwa 150 realisierten Projekte pro Jahr gehört *Zühlke* zu den führenden Beratungsunternehmen der Schweiz und Europas im Bereich Software Engineering.

Um an neue Aufträge zu kommen, muss *Zühlke* regelmässig offerieren: Jedes Jahr produziert die Firma über 500 Offerten. Damit jede Offerte den spezifischen Bedürfnissen des Kunden und Produktes gerecht wird, wird auch viel Zeit und Energie in das Verfassen der Offerten investiert. Somit gehört das Offertenschreiben zu den wichtigsten Geschäftsprozessen der Firma.

Obwohl jede Offerte am Schluss kunden- und projektspezifisch entwickelt werden muss, gibt es viele Aspekte, welche jede Offerte behandelt (Lösungsvorschlag, Planung, Vorstellung des Teams, Mehrwert von *Zühlke*, Kostenvoranschlag, usw.), wie auch sonst andere zahlreich wiederkehrende Elemente. Heute wird aber vieles jedes Mal neu zusammengestellt. Die Mitarbeitenden fangen von Null an oder verwerten frühere Offerten wieder und passen diese stückweise an. Das verwendete Material ist aber nicht immer aktuell, und es findet wenig Wissenstransfer zwischen den Mitarbeitenden statt. Viel Zeit wird also in Arbeitsschritte investiert, welche aber nicht zu einem tatsächlichen Mehrwert führen, sondern nur sicherstellen, dass eine erste grobe Fassung der Offerte steht, bevor alle Beteiligte an dem Dokument weiterarbeiten. Gerade in dieser ersten Phase des Prozesses liesse sich also einiges optimieren. So kam Christian Moser, Leiter einer „Competence Unit“ bei *Zühlke*, auf die Idee, einen Offerten-Generator zu programmieren, welcher aus zahlreichen, existierenden Vorlagen nur diejenige Elemente zusammenfügen würde, welche für eine bestimmte Offerte überhaupt Sinn machen.

Im vorliegenden Bericht wird verdeutlicht, wie ein solcher Generator im Rahmen einer 16-wöchigen Bachelorarbeit Schritt für Schritt entstanden ist. Zuerst wird die Ausgangslage mit Erläuterungen zu den Anforderungen an den Generator schärfer definiert (Kapitel 2). Danach werden grundsätzliche konzeptionelle Entscheide in Zusammenhang mit der Auswahl von konkreten Inhalten anhand von bestimmten Input-Parametern präsentiert (Kapitel 3). Anschliessend wird erklärt, wie der Generator als .NET Bibliothek konkret implementiert wurde (Kapitel 4). Gestützt auf experimentellen Auswertungen wird schliesslich diskutiert, inwiefern der Generator im Automationprozess des Offertenschreibens die Firma *Zühlke* überhaupt weiterbringt (Kapitel 5).

2. Kontextanalyse

Vor Projektbeginn und in der ersten Projektwoche fanden mehrere Gespräche mit Christian Moser, zuständige Ansprechperson bei *Zühlke*, statt. Diese Gespräche dienten der Klärung der Problem- und Aufgabenstellung. Die folgende Analyse fasst zentrale Erkenntnisse aus diesen ersten Gesprächen zusammen und legt unter anderem den Umfang der Aufgabe sowie die Anforderungen an das Produkt fest.

2.1. Umfang der Aufgabe

Ein Offerten-Generator kann beliebig weiterentwickelt werden, um möglichst viele Arbeitsschritte zu automatisieren und viele Szenarien abzudecken. Da die Zeit für die Entwicklung des Generators aber von vornherein klar begrenzt war, war es wichtig, den Umfang der Aufgabe bis Abgabe der Bachelorarbeit bereits am Anfang des Prozesses festzulegen. Das Ziel war, den Generator mit ein paar generischen Mechanismen auszurüsten, welche *Zühlke* später beliebig erweitern könnte.

Im folgenden Kontextdiagramm (siehe Abbildung 2.1) wird der ursprünglicher Lösungsansatz von Christian Moser veranschaulicht. Seine Idee bestand darin, die Logik zur Auswahl der Inhalte vom tatsächlichen Generator abzulösen. Welche Inhalte unter welchen Voraussetzungen in die Offerte hineinfliessen sollen, wäre als Domain Specific Language (DSL) in Form einer XML-Datei festgelegt (im Kontextdiagramm als einer der beiden Bestandteile des Systems erkennbar). Der Generator (zentrales Objekt im Diagramm) würde mit dieser Datei interagieren, um anhand von bestimmten Input-Parametern herauszufinden, welche Inhalte aus den Vorlagen (als Datenbank ausserhalb des Systems dargestellt) in die Offerte kommen sollen.

Der Vorteil dieser Trennung wäre, dass die XML-Datei von einem/einer AdministratorIn leicht angepasst werden könnte, auch wenn dieser/diese keine Programmiererfahrung hat. Der tatsächliche Code vom Generator müsste dafür nicht geändert werden. Angesichts der Tatsache, dass *Zühlke* sich als Firma stetig weiterentwickelt, könnten allfällige Veränderungen ganz einfach in den Offerten zum Ausdruck kommen.

Ebenfalls zentral im Kontextdiagramm ist natürlich die Offerte. Dabei illustriert die Visualisierung zwei wichtige Punkte. Erstens soll die Offerte eine Power Point Präsentation sein. Zweitens soll der Generator nur einen ersten Entwurf der Offerte erzeugen, und nicht ein fertiges Dokument. Danach arbeiten mehrere Mitarbeitende und Abteilungen an der Offerte weiter. Diese Bearbeitung bis zum fertigen Dokument (welches dem Kunden

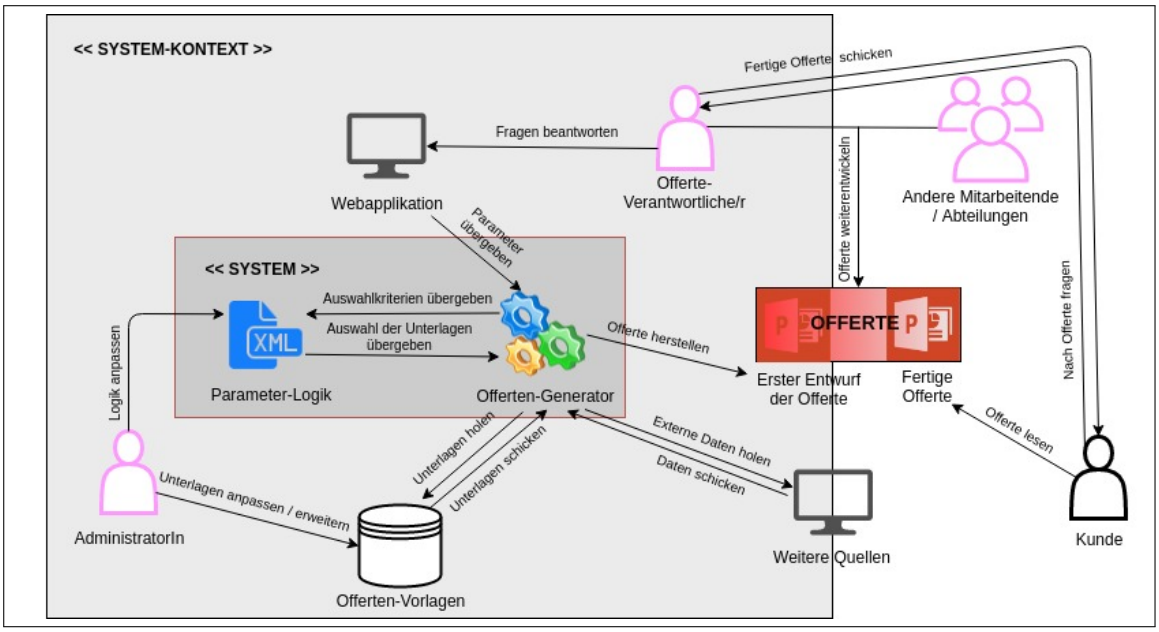


Abbildung 2.1.: *Kontextdiagramm des Offerten-Generators*

dann zugestellt wird) soll wie bisher manuell stattfinden und hat mit dem Generator nichts mehr zu tun. Diese Arbeit lässt sich inhärent nicht automatisch generieren, denn sie macht den individuellen Anteil einer Offerte aus.

Der/die Mitarbeitende bei *Zühlke*, der/die für den ersten Entwurf der Offerte zuständig ist, soll mit dem Generator über eine Webapplikation kommunizieren und dort Präferenzen bezüglich einer konkreten Offerte eingeben. Im Rahmen der Arbeit soll aber nur ein einfacher Webapplikation-Prototyp entwickelt werden, damit die Interaktion mit dem Generator sinnvoll getestet werden kann. Die tatsächliche Einbettung in ihr internes System wird *Zühlke* selber nach Abgabe der Arbeit durchführen.

Der Generator soll ausserdem in der Lage sein, neben den bestehenden Vorlagen weitere Quellen in die Power Point Präsentation einzubetten. Diese Funktionalität soll jedoch nur ansatzweise entwickelt werden, denn auch hier ist *Zühlke* nachher für die Einbindung an ihr internes System zuständig.

2.2. Anforderungen von Zühlke

Am Anfang der Arbeit stellte *Zühlke* folgende Anforderungen an das Produkt.

2.2.1. Functional Requirements

Der/die BenutzerIn gibt ein paar strukturierte Input-Daten ein und übergibt diese dem Generator. Gestützt auf diese Parameter wählt der Generator aus einem festen Satz von Unterlagen alle passende Elemente (Texte, Bilder, Grafiken) aus. Er/sie stellt diese zusammen und generiert eine Power Point Präsentation.

Die automatisch generierte Offerte ist dann noch nicht fertig und muss/kann manuell angepasst und erweitert werden. Die Power Point Präsentation soll aber alle Grundelemente einer Offerte beinhalten - mit tendenziell lieber mehr Slides als weniger, da es für den/die BenutzerIn einfacher ist, Slides zu löschen, als neue einzufügen. Wo noch keine konkrete Informationen vorhanden sind (z.B. bezüglich der Zusammensetzung des Teams), sollen passende Platzhalter sicherstellen, dass die Informationen noch hinzugefügt werden können. Ein erstes Ziel ist, dass der Generator bei herkömmlichen Offerten - d.h. nicht zu spezifischen Fällen - ca. 50% der Schlussofferte automatisch erzeugt.

Die Unterlagen, auf welche der Generator zugreift, sind von einem/einer AdministratorIn einfach einzugeben, zu ersetzen und zu aktualisieren. Dieser kann auch die Input-Parameter anpassen sowie neue hinzufügen und dazu eine passende Logik festlegen, die die Zusammensetzung der Unterlagen neu ordnet. Dafür muss er/sie den internen Code des Generators aber nicht ändern.

2.2.2. External Interface Requirements

Der Generator soll von einer firmeninternen Webapplikation abgerufen werden können. Die Parameter werden in der Applikation eingegeben und dem Generator übergeben. Die Programmierung der Webapplikation ist aber nicht Teil des Projekts: Der Generator soll als selbstständige .NET Bibliothek implementiert werden.

Die Unterlagen sollen zentral gelagert werden: Der Generator muss also nicht überall nach Informationen suchen, sondern nur an einem spezifischen Ort (je nach Standort von *Zühlke* kann dieser Ort aber anderswo sein). Es soll aber für den/die BenutzerIn möglich sein, eine externe Informationsquelle einzugeben, um weitere Unterlagen in die Offerte einzubetten. Somit soll eine spätere Integration in eine bereits existierende firmeninterne Webapplikation ermöglicht werden, auf welche Informationen über Mitarbeitende, alte Projekte, usw. abgerufen werden können.

2.2.3. Performance Requirements

Im Laufe der ersten Gesprächen mit *Zühlke* wurde das Thema der Performanz nie angeschnitten. Es ist denkbar, dass der Generator mehrere Sekunden braucht, um die Offerte aus allen Teilelementen zu erstellen. Mehr als eine Minute soll es jedoch nicht dauern.

2.2.4. Safety, Reliability und Security Requirements

Der Generator wird als einfache Bibliothek programmiert. Die Integration in eine Webapplikation wird von *Zühlke* übernommen. Dementsprechend gibt es auch kaum Anforderungen an den Generator bezüglich Sicherheit und Reliabilität. Die Bibliothek soll aber natürlich so programmiert werden, dass grundlegende Sicherheitsstandards eingehalten werden (Inputs prüfen, v.a. wenn eine Quelle als Pfad eingegeben wird, usw.).

2.3. Betroffene

Bei *Zühlke* sind im Prozess des Offertenschreibens jeweils mehrere Mitarbeitende aus unterschiedlichen Abteilungen beteiligt; bei grossen Offerten sind sogar sehr viele involviert. Dementsprechend sind vom Offerten-Generator folgende Akteure direkt oder indirekt betroffen.

2.3.1. Bid Team

Eine zentrale Rolle bei der Herstellung von Offerten spielt das Bid Team. Vom ersten Kundenkontakt bis zum allfällig unterzeichneten Vertrag zieht dieses Team über alle Schritte hinweg die Fäden. Es behält den Überblick und ist für die fertige Offerte zuständig. So läuft es auf jeden Fall bei grösseren Offerten; bei kleineren kann es auch sein, dass operationelle Abteilungen den ganzen Prozess mehr oder weniger allein abwickeln.

Wichtig für das Bid Team ist es, dem Kunden hochqualitative, kundenspezifische Lösungen anzubieten. Es hat grosses Interesse daran das Offertenschreiben bis zu einem gewissen Grad zu automatisieren. So würde viel kleine und zeitraubende Korrekturarbeit (wie z.B. die Vereinheitlichung von Schriftgrössen oder Policen innerhalb eines Dokuments) wegfallen. Die Offerten würden auch immer das aktuellste Layout verwenden, statt auf alte Vorlagen zurückzugreifen. Ausserdem könnte ein erstes Grundgerüst für die Offerte jeweils rasch erzeugt werden. Es wäre also insgesamt mit einer markanten Effizienzsteigerung zu rechnen.

Die Bid Team-Mitarbeitenden wären sicher die HauptbenutzerInnen des Generators. Mit dem Generator würden sie erste Offerten-Entwürfe erzeugen, an welchen andere Mitarbeitende und Abteilungen nachher weiter arbeiten würden. Das Bid Team wäre vermutlich auch dafür zuständig, die Inhalte für den Generator zu verwalten und stetig zu aktualisieren.

2.3.2. Solution Center

Mit jeder Offerte versucht *Zühlke* ihren (potenziellen) Kunden eine Lösung für ein konkretes Problem zu verkaufen. Dementsprechend steht diese Lösung im Zentrum der

Offerte - und diese wird im Solution Center entwickelt. Offerten zu schreiben ist aber nicht die Kernkompetenz des Solution Centers: In erster Linie ist das Team dafür zuständig, die Lösungen ganz konkret zu entwickeln und zu implementieren. Mit dem Offertenschreiben hat nicht jeder Mitarbeitende gleich viel Übung. Dazu kommt, dass die Abläufe intern nicht klar definiert sind, und dass jeder mit anderen Unterlagen arbeitet.

Für das Team vom Solution Center wäre es natürlich sehr interessant in kurzer Zeit anhand von ein paar Input-Parametern ein erstes Gerüst für eine Offerte generieren zu können, statt das Rad jedes Mal neu erfinden zu müssen. So könnte sich das Team auf die tatsächliche Entwicklung ihres Lösungsvorschlags konzentrieren. Dies würde auch sicherstellen, dass nichts Wichtiges vergessen geht.

Bei kleineren Offerten kommt es regelmässig vor, dass der erste Entwurf einer Offerte von einem Mitarbeitenden des Solution Centers verfasst wird. Auch sie wären also direkte BenutzerInnen des Generators. Inhalte, die das Solution Center für Offerten produziert und nicht zu stark den Bedürfnissen eines spezifischen Kunden zugeschnitten sind, könnten ausserdem über die zentralisierte Dokumentenablage in neue Offerten hineinfliessen. Für diesen Schritt wäre aber das Bid Team zuständig.

2.3.3. Marketing

Das Marketing Team pflegt u.a. die Corporate Identity *Zühlke*'s.

An den Offerten-Generator hätte die Abteilung in erster Linie Interesse in Zusammenhang mit dieser Corporate Identity. Die Offerten würden vom Layout her einheitlicher werden, und das verwendete Material wäre immer auf dem neuesten Stand.

2.3.4. Competence Center

Für jede Offerte muss ein Team zusammengestellt werden, welches den Auftrag nach einem Vertragsabschluss auch abwickeln könnte. Das Competence Center ist für die Zusammenstellung des Teams zuständig. In der Offerte wird das Team dann vorgestellt: Jedes Teammitglied mit Namen, Funktion und einem kurzen Text. Manchmal kommt noch weiteres Material wie ein passendes Zitat des Projektleiters, usw. dazu.

Der Generator könnte die Informationen über die verschiedenen Teammitglieder ohne weiteres hinzufügen. Im besten Fall müsste das Competence Center nur Namen anklicken und schon wären die entsprechenden Texte und Bilder in die Offerte eingefügt. Diese Funktionalität ist aber fortgeschrittener Art und wird nicht als erste implementiert.

2.3.5. Juristische Abteilung

Jede Offerte enthält auch legale Aspekte, welche von der juristischen Abteilung der *Zühlke* geprüft werden.

Der Generator würde zu einer Vereinheitlichung der Offerten führen - also auch im Bezug auf die rechtlichen Aspekte. Für die juristische Abteilung würde die Arbeit also etwas einfacher werden.

2.3.6. Leitung

Wird der Auftrag mit dem Kunden nach einer Offerte abgeschlossen, hat sich das Offertenschreiben für *Zühlke* gelohnt. Geht der Kunde zur Konkurrenz, sind die investierten Mittel aber hingegen verloren. Für die Leitung ist es dementsprechend wichtig die notwendigen Mittel für die Herstellung von Offerten zu minimisieren, denn die Ratio zwischen Offertenkosten und eingeholten Aufträgen muss stimmen.

Mit dem Offerten-Generator könnte die Zeit, die in der ersten Phase des Offertenschreibens investiert wird, kürzer werden, während die Qualität der Offerten womöglich steigen würde. Das trägt zu einer stärkeren Konkurrenzfähigkeit bei, liegt also im Interesse der Leitung.

2.3.7. Kunden

Vom Offerten-Generator ist der Kunden nicht direkt betroffen: Die Offerte, die für ihn spezifisch geschrieben wird, soll hochqualitativ sein, unabhängig davon, wie sie hergestellt wird. Um die Qualität sicherzustellen darf natürlich nicht der ganze Prozess automatisiert werden. Gedacht ist nur, am Anfang Zeit zu gewinnen. Die Offerte kann einerseits dem Kunden schneller zugestellt werden und die gewonnene Zeit wird anderseits in die Verbesserung der Offerte investiert.

Das Interesse der Kunden von *Zühlke* ist, weiterhin hochqualitative Offerten zu erhalten. Der Generator soll auf keinen Fall dazu führen, dass die Offerten weniger spezifisch werden. Da der Generator aber nicht den ganzen Prozess abwickeln soll, ist das Risiko für den Kunden gleich Null.

3. Parametrisierung des Generators

Jede Offerte ist anders - und doch enthalten alle einige wiederkehrende Elemente. Für die Implementierung des Generators war die Identifizierung dieser Elementen absolut zentral. Dabei war der Kerngedanke folgender: Es soll einfach sein, das Material und die Kriterien für die Herstellung der Offerten anzupassen und zu erweitern, ohne etwas an dem Code des Generators zu ändern. Eine Analyse von mehreren existierenden Offerten von *Zühlke* sowie Gespräche mit Mitarbeitenden der Firma und mit dem Betreuer der Arbeit, Prof. Dr. Luc Bläser, dienten der Identifizierung von drei unterschiedlichen generischen Bausteinen, die den Inhalt einer konkreten Offerte direkt beeinflussen. Im vorliegenden Kapitel werden diese Bausteinen und deren jeweiligen Anwendung präsentiert. Zuerst soll aber ein erster grundlegender Design-Entscheid erläutert werden.

3.1. Kernentscheid: Alles in Einem

Vom Anfang an war eine der Hauptanforderungen an den Generator, die Unabhängigkeit der Vorlagen gegenüber dem Kern des Generators zu gewährleisten. Anders gesagt: Der Generator sollte nicht an konkreten Inhalten gebunden sein, sondern die Konfiguration der Inhalte sollte ausserhalb des Generators stattfinden (Abschnitt 2.2). Die ursprüngliche Idee von Christian Moser, zuständige Ansprechperson bei *Zühlke*, sah vor, zu diesem Zweck eine Domain Specific Language (DSL) zu entwickeln. Die Logik zur Auswahl von konkreten Inhalten wäre in einer XML-Konfigurationsdatei festgehalten worden, die der/die AdministratorIn parallel zu den inhaltlichen Vorlagen hätte warten müssen, damit der Generator funktioniert (Abschnitt 2.1). In Absprache mit Christian Moser wurde auf diese Idee jedoch verzichtet und ein anderer Lösungsansatz entwickelt: Inhalte und Logik sollten nun beide zusammen in einer übergrossen Power Point Vorlage gespeichert werden.

3.1.1. Eigenständige Power Point Vorlage

Um die Arbeit des/der AdministratorIn vom Generator einfach und übersichtlich zu halten, und um eine möglichst stabile Lösung zu entwickeln, wurden zwei Hauptentscheide getroffen. Erstens soll der Generator möglichst wenig mit verstreuten Inhalten arbeiten, um diese in einer Power Point Präsentation zusammenzufügen, sondern die Inhalte sollten in eine übergrosse Power Point Vorlage zusammengetragen und richtig gestaltet werden. Der Generator ist dann dafür zuständig, die überflüssigen Inhalte zu löschen. Zweitens soll

die Logik zur Auswahl der passenden Inhalte nicht in einer separaten Konfigurationsdatei gespeichert, sondern direkt in der Vorlage gekennzeichnet werden. Dazu soll eine möglichst schlichte und intuitive Syntax entwickelt werden, damit die Wartung der Vorlage einfach bleibt. Um sicherzustellen, dass diese Syntax mit den gestalterischen Ansprüchen an der Offerte nicht kollidiert, soll ausserdem mit der Kommentar-Funktion von Power Point gearbeitet werden.

Diese Lösung bringt mehrere Vorteile mit sich:

- Die Syntax ist einfach anzuwenden, auch ohne Programmierkenntnisse.
- Keine externe XML-Konfigurationsdatei muss verwaltet werden: Alles Nötige steht direkt in der Vorlage.
- Es kann sichergestellt werden, dass eine Vorlage über alle Slides hinaus das richtige Layout hat (und somit auch die richtige Corporate Identity). Soll das Grundlayout verändert werden, können die Masterslides einfach angepasst werden.
- Mit dieser Lösung ist die Chance grösser, dass beim Anpassen des Inhalts nichts vergessen geht, weil es anderswo in einem anderen Dokument liegt.
- Diese Lösung ist sehr stabil, denn das Füllen von Platzhaltern mit Inhalten ist ein komplexes Prozess, welches schnell zu Schwierigkeiten führen kann (der Text ist zu lang und läuft über den vorgesehenen Rand, die Bilder werden nicht richtig registriert, usw.), während bereits existierende Slides vom Anfang an ordentlich aussehen und es relativ einfach ist, die überflüssigen Slides zu löschen.
- Die Vorlage kann jederzeit mit neuem Inhalt ergänzt werden. Wird im Rahmen einer Offerte eine besonders gute neue Slide hergestellt, kann diese sehr einfach in die Vorlage kopiert werden, um in späteren Offerten wieder verwertet zu werden.
- Kommentare sind am Ende des Auswahlprozesses einfach zu löschen.

3.1.2. Mehrere Vorlagen zur Auswahl

Ob eine Offerte auf Deutsch oder auf Englisch geschrieben wird, oder ob sie eher kurz oder lang sein soll, wird bereits am Anfang des Herstellungsprozesses entschieden. Es sind relativ einschneidende Unterschiede. Aus diesem Grund können/sollen mehrere Power Point Vorlagen vorbereitet werden. Der/die BenutzerIn kann am Anfang des Prozesses entscheiden, mit welcher Vorlage er/sie weitermachen möchte. Die Vorlagen können sich hinsichtlich Struktur und Layout natürlich gleichen - oder auch bewusst anders aussehen. Wichtig ist, dass jede Vorlage in sich konsistent bleibt. Genau das stellt sicher auch die grösste Anforderung an den/die AdministratorIn des Generators dar.

3.2. Generische Bausteine

Eine Vorlage für den Generator kann mit drei Arten von generischen Bausteinen aufbereitet werden. Genaue Informationen dazu befinden sich im Dokument mit Erklärungen zur Aufbereitung der Vorlagen im Anhang (Anhang A). Im vorliegenden Abschnitt werden diese drei Arten aber konzeptuell erklärt. Von der Grundidee her funktionieren alle drei Bausteine gleich: Wird eine Vorlage ausgewählt, scannt der Generator die Vorlage und sucht nach allen möglichen Bausteinen. Die gefundenen Elemente werden dann der Webapplikation übergeben, damit der/die BenutzerIn des Generators sinnvolle Inhalte für die Offerte eingeben/auswählen kann. Änderungen in einer Vorlage werden beim nächsten Scannen vom Generator sofort dynamisch erkannt. Die Vorlagen können dementsprechend stetig angepasst und allfällige Fehler sofort korrigiert werden. Jede Vorlage kann ausserdem ihre eigenen Bausteine (d.h. Platzhalter, Kriterien, oder Verweise auf Datensätze) enthalten.

3.2.1. Platzhalter

In jeder Offerte gibt es projektspezifische Informationen, die jedes Mal aber gleicher Natur sind: Es handelt sich um Kundenname, Projekttitel, Verfasser der Offerte, usw. Diese projektspezifischen Daten kommen oft auch direkt im Text innerhalb der Offerte vor. Aus diesem Grund schien es sinnvoll, für solche Informationen mit Platzhaltern zu arbeiten, welche bei der Generierung der Offerte durch die Angaben ersetzt werden, welche der/die BenutzerIn über die Webapplikation dem Generator übermittelt hat.

In der folgenden Abbildung (Abbildung 3.1) wird eine Slide aus einer Vorlage dargestellt, welche zwei Platzhalter beinhaltet: {Projekt} und {Kunde}. Wird eine Vorlage vom/von der BenutzerIn ausgewählt, erkennt der Generator alle vorhandene Platzhalter in der Vorlage. Dies findet generisch statt, so dass beliebige (neue) Platzhalter ganz einfach in einer Vorlage eingesetzt werden können, ohne dass Änderungen am Generator nötig wären.

Ein Platzhalter kann natürlich auch mehrmals in einer Vorlage vorkommen und wird dann jedes Mal durch den entsprechenden Input vom/von der BenutzerIn ersetzt. Zwei Vorlagen können auch unterschiedliche Platzhalter beinhalten.

Mit einer Schwierigkeit ist der Generator aber nicht in der Lage umzugehen: Die Deklination von gewissen Platzhaltern. Dies ist gerade in der deutschen Sprache ein Thema. Aus diesem Grund empfiehlt es sich, jeden Platzhalter in der Vorlage gelb zu markieren. Die Markierung bleibt nach der Generierung der Offerte vorhanden, so dass der Leser die Platzhalter schnell wieder identifiziert und deren Deklination prüfen kann. Die Markierung beeinflusst die Arbeit des Generators jedoch nicht.

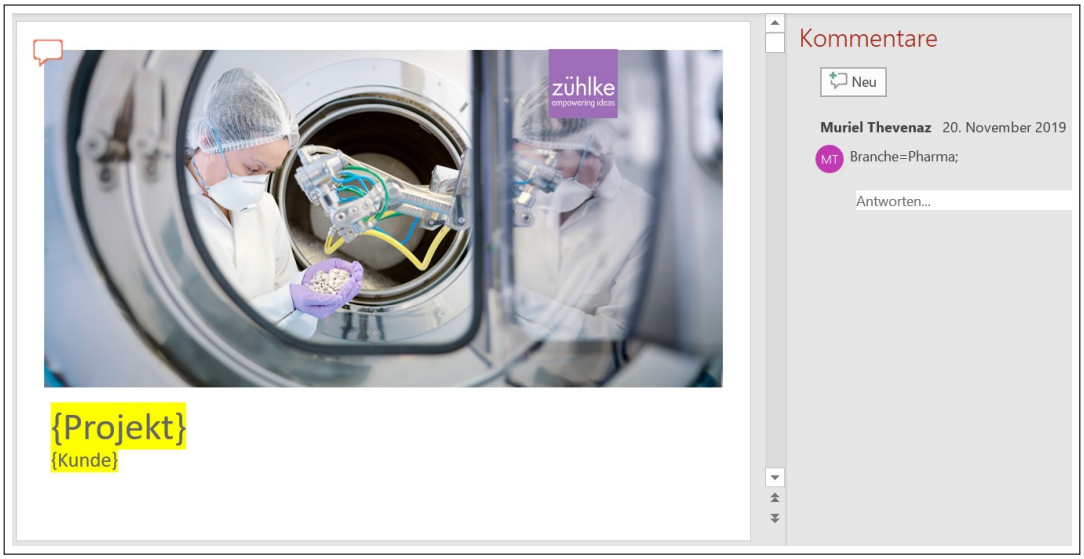


Abbildung 3.1.: *Slide mit Platzhaltern und Kriterium im Kommentar*

3.2.2. Kriterium

Damit bestimmt werden kann, welche Inhalte aus einer Vorlage gelöscht werden sollen, und welche nicht, werden Auswahlkriterien definiert, anhand welcher der Generator die Inhalte selektiert. Kriterien enthalten zwei Komponenten: eine Bedingung (z.B. „Branche“), und mindestens einen Wert (z.B. „Industrie“). Eine Bedingung fasst alle Werte gleicher Art zusammen. Der/die BenutzerIn kann dann für jede Bedingung den passenden Wert auswählen. Alle Werte einer Bedingung schliessen sich also gegenseitig aus.

Um die Gestaltung der Slides nicht aufs Spiel zu setzen, werden Kriterien in den Vorlagen innerhalb des Kommentarfelds einer Slide gespeichert - wie ersichtlich auf der Abbildung 3.1, bei welcher das Kriterium „Branche=Pharma;“ in einem Kommentar steht. Wie bei den Platzhaltern werden alle in einer Vorlage vorkommende Bedingungen und deren verschiedene Werte während dem Scannen der Vorlage registriert und der Webapplikation übergeben. Auch da können die Kriterien vom/von der AdministratorIn also stetig angepasst werden.

Wird bei einer Slide keines der für sie definierten Kriterien ausgewählt, wird die Slide gelöscht. Wenn nur ein Kriterium aber passt, bleibt sie in der Offerte. Die Auswahl der Inhalte folgt also dem sogenannten ODER-Prinzip. Dieses Vorgehen wurde dem UND-Prinzip bevorzugt, weil es tendenziell besser ist, wenn Slides in der Offerte bleiben, als wenn sie gelöscht werden. Der/die BenutzerIn kann überflüssige Inhalte im Nachhinein einfacher löschen, als neue Inhalte suchen und in die Offerte einfügen. Heute kann nur eine Art UND-Prinzip verwendet werden, indem Werte innerhalb eines Kriteriums kombiniert werden („Inhalt=Agiles Vorgehen und viel Bilder;“ und „Inhalt=Agiles Vorgehen und viel

Text;“). Wenn es sich herausstellen würde, dass dies nicht reicht, könnte in der Zukunft die Syntax allenfalls so erweitert werden, dass der Generator auch das UND-Prinzip unterstützt. Dies würde aber ohne Änderungen am Code des Generators nicht gehen. Ausserdem wurde absichtlich auf eine komplizierte Syntax verzichtet.

Wichtig zu erwähnen ist noch, dass wenn eine Slide gar kein Kriterium enthält, bleibt sie sowieso in der Offerte. Dies wurde so entschieden, weil viele Inhalte jedes Mal in einer Offerte vorkommen.

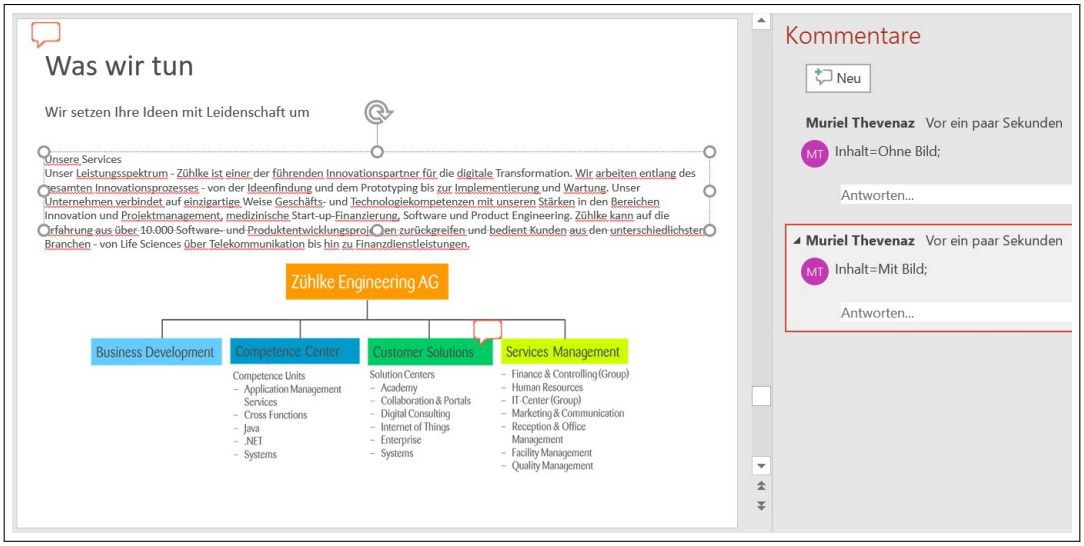


Abbildung 3.2.: *Auswahlkriterien zum lokalen Löschen einzelner Elemente*

Mit den Kriterien ist es auch möglich, Elemente innerhalb einer Slide selektiv zu löschen. Dazu muss der Kommentar auf das Element - also auf ein Bild, eine Grafik, ein Textfeld, usw. - zeigen. Diese Selektion setzt jedoch voraus, dass die betroffene Slide für die Offerte sonst ausgewählt wurde. In der Abbildung 3.2 wird dazu zwei Mal die gleiche Bedingung, aber mit unterschiedlichen Werten, verwendet: Wird „Inhalt=Ohne Bild“ gewählt, bleibt die Slide in der Offerte, das Bild wird aber gelöscht; wird „Inhalt=Mit Bild“ gewählt, bleibt die Slide ebenfalls in der Offerte, dieses Mal mit dem Bild. Es ist aber durchaus möglich, mit unterschiedlichen Bedingungen (eine für die ganze Slide und eine für das selektive Löschen) zu arbeiten.

Wie bereits angekündigt, findet die Selektion von Elementen innerhalb einer Slide über die Position des Kommentars statt. Der Kommentar ist aber keinem Element auf der Slide direkt zugeordnet (siehe Unterabschnitt 4.3.2). Das heisst, dass wenn ein Kommentar gleichzeitig auf zwei Elemente zeigt (zwei Bilder übereinander - oder ein vergessenes leeres Textfeld und ein normales Textfeld), nur das eine Element gelöscht wird. Die Berechnungen zur Positionierung der Kommentare und Elementen auf der Slide sind ausserdem leider nur approximativ. Aus diesem Grund kann das selektive Löschen für

sehr präzise Arbeit nicht angewendet werden. Für solche Fällen ist es dann besser, die Slide zu kopieren und so anzupassen wie gewünscht. Die Auswahl der für eine Offerte passenden Slide findet dann wie üblich über Kommentare auf Slide-Ebene - und nicht über selektives Löschen - statt.

3.2.3. Datensatz

Datensätze sind strukturierte Inhalte, welche ausserhalb der Vorlage dem Generator zur Verfügung stehen - z.B. Informationen über Mitarbeitende, Referenzprojekte, usw. Wie bei den anderen generischen Bausteinen können Verweise auf neue Arten von Datensätzen relativ einfach in die Vorlage hinzugefügt werden. Existiert eine Datenbasis, muss nur eine Slide in der Vorlage korrekt aufbereitet (mehr dazu in der Gebrauchsanleitung im Anhang A) und in ihrem Kommentarfeld auf die richtige Quelle verwiesen werden.

Abbildung 3.3 ist ein Beispiel für eine solche Slide: Sie enthält vier genau gleiche Gruppen von Elementen (Textfelder und ein Bild) sowie die Markierung „!!TEAM“ in einem Kommentar. Jede Gruppe kann mit Informationen aus einem ausgewählten Datensatz gefüllt werden. Das Layout der Slide - inkl. Textgrösse, usw. - bleibt dabei erhalten.

Werden vom/von der BenutzerIn mehr Datensätze ausgewählt, als in der vorbereiteten Slide Platz vorhanden ist, wird die Slide vom Generator automatisch kopiert und mit den zusätzlichen Inhalten gefüllt. Überflüssige Elemente werden gelöscht.

Die Platzhalter in den Textfeldern verweisen auf bestimmte Arten von Informationen im Datensatz. Diese Platzhalter werden also nicht als „normale Platzhalter“ betrachtet. In Slides mit Verweis auf Datensätze prüft der Generator beim Scannen der Vorlage, ob ein Platzhalter auf eine konkrete Quelle in der Datenbasis verweist. Existiert die Quelle nicht, wird der Platzhalter erst dann als sonstiger Platzhalter (siehe Unterabschnitt 3.2.1) betrachtet.

In jeder Gruppe ist auch ein Bild vorhanden. Mehr als eins darf es pro Gruppe nicht geben. Dies wurde so entschieden der Einfachheit halber (im Hinblick auf die Implementierung, aber auch damit die Syntax für die Aufbereitung und Wartung der Vorlagen möglichst schlicht bleibt). Bei Bildern kann nicht wie bei Textfeldern einfach mit Platzhaltern gearbeitet werden, so dass ein weiteres Konstrukt hätte eingebaut werden müssen, damit der Generator entscheiden kann, welches Bild in welchem Feld eingefügt werden muss.

Der gleiche Ansatz wie bei den Datensätzen wurde ausserdem für die Registrierung der Kapitel im Inhaltsverzeichnis der Offerte verwendet. Dabei verweist die Slide jedoch nicht auf externe Quellen, sondern auf andere Slides innerhalb der Offerte (siehe Unterabschnitt 4.3.2).

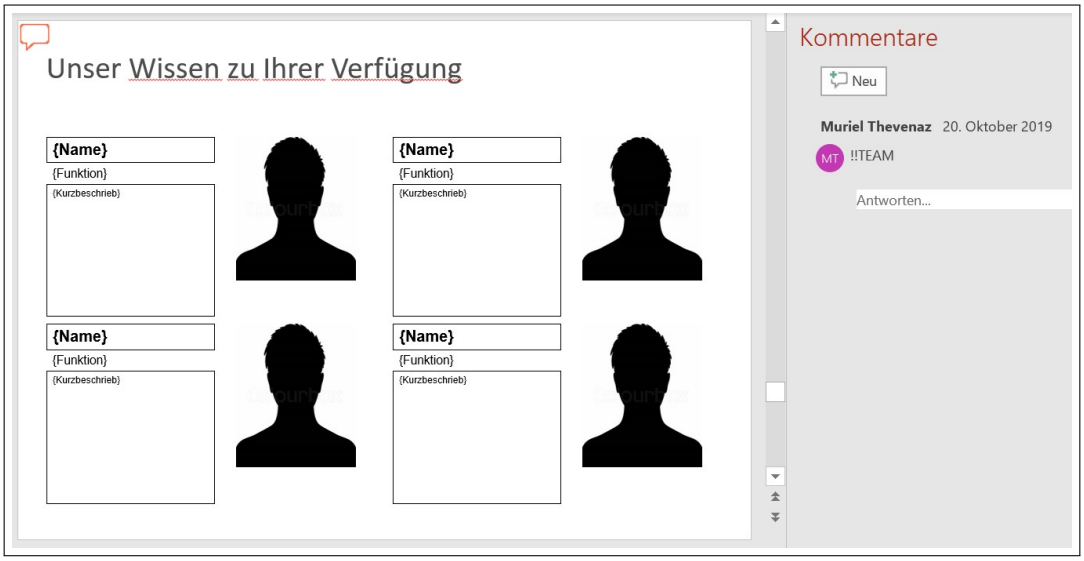


Abbildung 3.3.: *Slide zum Ausfüllen mit Informationen zu den Teammitgliedern*

4. Implementierung

Im vorliegenden Kapitel wird die Implementation des Generators diskutiert. Dabei wird Schritt für Schritt erläutert, wie der Generator konkret arbeitet. Zuerst werden aber allgemeine Informationen zum Projekt und zu den eingesetzten Technologien präsentiert.

4.1. Übersicht

Der Offerten-Generator ist ein kleines Projekt, welches als Einzelarbeit in ca. 20 bis 30 Arbeitstagen entwickelt wurde (es handelt sich um eine Schätzung; die Zeit, welche in das Verfassen der Bachelorarbeit investiert wurde, wurde dabei nicht mitberechnet). Architektur und Code deuten auf diesen begrenzten Projektumfang.

4.1.1. Einfache Architektur

Abbildung 4.1 zeigt alle vier Komponente der entwickelten Lösung. Der Generator ist ein zustandsloses System und wurde als .NET Bibliothek implementiert. Gesteuert wird er über eine Webapplikation: Ein Prototyp, entwickelt mit dem ASP.NET Framework, erlaubt den Generator zu testen. Wählt ein/eine BenutzerIn über die Webapplikation eine der Power Point Vorlagen aus, welche im dafür reservierten Ordner gespeichert ist (im Diagramm oben rechts dargestellt), wird ein *Generator*-Objekt instanziert und dabei der Pfad zur ausgewählten Vorlage mitgegeben. Somit ist jede *Generator*-Instanz in der Lage mit einer bestimmten Vorlage direkt zu interagieren (es ist zu beachten, dass der Generator nicht thread-safe entwickelt wurde: Wird eine Vorlage vom/von der AdministratorIn bearbeitet, während ein/eine BenutzerIn genau mit dieser Vorlage eine Offerte generieren will, können beim Generieren der Offerte unvoraussehbare Fehler passieren).

Der Generator kann in eine Offerte auch Datensätze einfügen (siehe Unterabschnitt 3.2.3). Zurzeit funktioniert dies jedoch nur, wenn die entsprechende Datensätze in Form von Text- und Bild-Dateien in einer Ordnerstruktur gespeichert sind. Alle Ordner, die Datensätze gleicher Art enthalten, müssen sich in einem übergeordneten Ordner befinden. Der Pfad zu diesem Ordner muss in der Konfigurationsdatei des Generators angegeben werden. Die Webapplikation erhält vom Generator alle nötige Informationen darüber, welche Datensätze in Zusammenhang mit einer ausgewählten Vorlage für den/die BenutzerIn zur Auswahl stehen. Wird die „Generate“-Methode des Generators über die Webapplikation

aufgerufen, müssen nur die Namen der aus allen Datensätzen ausgewählten Ordner mitgegeben werden damit das Einbinden der passenden Informationen in die Offerte stattfindet.

Später soll die Verarbeitung von Informationen aus Datensätzen erweitert werden, damit der Generator mit anderen Konstrukten als Ordnerstrukturen umgehen kann - beispielsweise mit Informationen, welche aus der firmeninternen Webapplikation von *Zühlke* stammen.

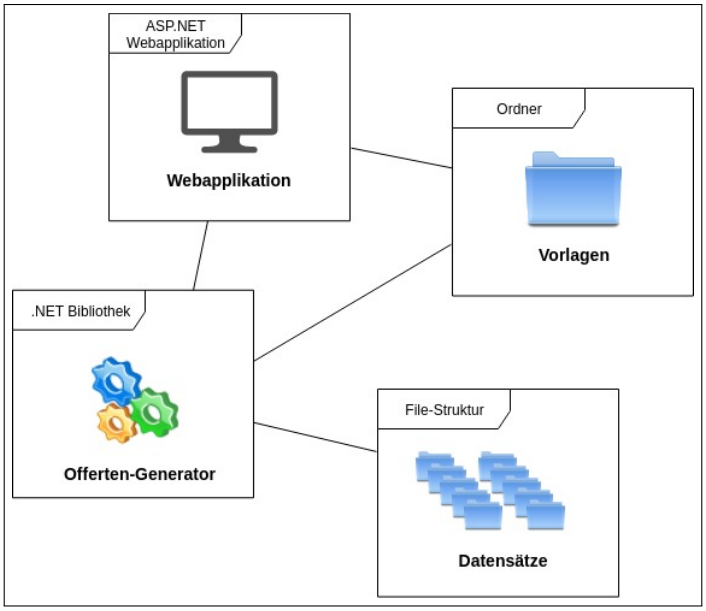


Abbildung 4.1.: *Architekturdiagramm der implementierten Lösung*

4.1.2. Einige Eckdaten zum Code

Wie bereits mehrmals erwähnt, wurde der Generator als selbstständige .NET Bibliothek entwickelt. Dies war eine Anforderung von *Zühlke* (Abschnitt 2.2). Konkret wurde mit .NET Standard 2.0 gearbeitet.[6] Für den Prototyp der Webapplikation - welcher nicht direkt Teil der Arbeit war - wurden ASP.NET MVC 5.2.7.0[4] und ASP.NET Razor 3.2.7[5] eingesetzt. Dabei wurde als Server das von Microsoft empfohlene Internet Information Services (IIS) verwendet.[9] NUnit 3.12.0[12] wurde für das Testing eingesetzt.

Wie in Tabelle 4.1 ersichtlich, wurden bei der Implementierung des Generators ca. 1400 Codezeilen, verteilt auf 16 Klassen, eine abstrakte Klasse und eine Schnittstelle geschrieben. Dazu kommen ca. 1700 Zeilen für die Tests und ein bisschen weniger als 450 Zeilen für den Prototyp der Webapplikation. Insgesamt handelt es sich also um über 3500 Codezeilen und 1260 ausführbare Codezeilen (davon etwa 40% im Generator selbst).

Generator	Webapplikation	UnitTests
.NET Standard 2.0	ASP.NET MVC 5.2.7.0	NUnit 3.12
16 Klassen, 1 abstrakte Klasse und eine Schnittstelle	5 Models, 6 Views und 1 Controller	13 Klassen
1437 Codezeilen	444 Codezeilen	1736 Codezeilen
521 ausführbare Codezeilen	121 ausführbare Codezeilen	624 ausführbare Codezeilen
97% Code Coverage	Manuelles Testing	129 Testfälle

Tabelle 4.1.: *Eckdaten zum Code-Projekt*

Die UnitTests sind so strukturiert, dass für fast jede Klasse des Generators eine entsprechende Testklasse existiert (nur die *GeneratorException*-Klasse und die beiden Helper-Klassen wurden nicht in einer eigenen Klasse getestet). Laut dem Testabdeckung-Tool dotCover[3] decken die insgesamt 129 UnitTests 97% des Codes vom Generator ab (die tiefste Abdeckung liegt bei 90%). Die Webapplikation wurde ausschliesslich manuell getestet. Es handelt sich um einen einfachen Prototyp mit 5 „Models“, 6 „Views“ und einem „Controller“.

4.2. Open XML SDK

Zentral für die Implementierung des Generators war der Entscheid, mit welchem Programmierwerkzeug die Bearbeitung der Power Point Präsentationen (sowohl Input wie auch Output des Generators) stattfinden soll. Es standen drei Möglichkeiten zur Auswahl: Microsoft.Office.Interop.PowerPoint[11], Spire.Presentation[2] oder Open XML SDK[10]. In Rücksprache mit Prof. Luc Bläser, Betreuer der Arbeit, und Christian Moser, Ansprechperson bei *Zühlke*, wurde entschieden mit Open XML SDK (Version 2.9.1)[10] zu arbeiten. Im Gegensatz zu Microsoft.Office.Interop ist dieser SDK aktuell, recht verbreitet und gut dokumentiert. Anders als Spire.Presentation ist er ausserdem kostenlos.

Word, Excel wie auch Power Point sind XML-Dokumente. Die Struktur eines Power-Point-XMLs unterscheidet sich aber von derjenigen eines Word- oder Excel-XMLs:

> *A PresentationML document is not stored as one large body in a single part. Instead, the elements that implement certain groupings of functionality are stored in separate parts. For example, all comments in a document are stored in one comment part, while each slide has its own part. A separate XML file is created for each slide.*[8]

Anders gesagt, es werden die Informationen betreffend einem einzelnen Dokument je nach Art von Information (Layout, Bild, Text, usw.) anderswo gelagert. Damit am Ende aber ein einheitliches Dokument entsteht, müssen die Beziehungen zwischen den verschiedenen Elementen aber auch logisch gespeichert werden. Dies geschieht in sogenannten „parts“, die alle von der Klasse *OpenXmlPart* erben. Zu jeder Art von Element existiert also ein

Gegenstück, in welchem die Beziehungen zu den relevanten anderen Elementen gespeichert werden: *Presentation* und *PresentationPart*, *Slide* und *SlidePart*, usw. Die Elemente werden anhand von IDs referenziert.

Auf jeder XML-Stufe existieren auch „properties"-Elemente, in welchen Informationen über ein konkretes XML-Element gespeichert werden. So wird auf Präsentation-Stufe z.B. festgelegt, wie ein Dokument aussehen soll, wenn dieses „normal" bearbeitet wird (*NormalViewProperties*) oder wenn der Präsentationsmodus ausgewählt wurde (*SlideViewProperties*). Ein Text (*TextBody*) kann auch spezielle Eigenschaften besitzen (*BodyProperties*): Er kann z.B. seine Grösse automatisch anpassen, um innerhalb eines vordefinierten Felds zu bleiben, oder sie behalten und allenfalls über den Rand des Feldes hinausgehen.

Auch wichtig bei Power-Point-XMLs sind Listen von Elementen: *SlideIdList* oder *SlideMasterIdList* auf Präsentation-Stufe, *CommentList* auf Slide-Stufe. In diesen Listen werden nicht nur alle Elemente einer bestimmten Sorte registriert, sondern auch die Reihenfolge dieser Elemente. Um die Position einer Slide in einer Präsentation zu ändern, muss also beispielsweise nur die Liste aller Slides in dem Dokument (nämlich *SlideIdList*) überarbeitet und deren Reihenfolge dort angepasst werden.[7]

4.3. Arbeitsschritte des Generators

Zwei Haupt-Prozesse sind mit dem Generieren einer Offerte durch den Generator verbunden: Zuerst wird die vom/von der BenutzerIn ausgewählte Vorlage gescannt, damit alle im Dokument vorhandenen Platzhalter, Kriterien und Verweise auf Datensätze (siehe Abschnitt 3.2) dynamisch erkannt werden; hat der/die BenutzerIn die passende Informationen und Kriterien eingegeben/ausgewählt, kann der tatsächliche Generierungsprozess angestossen werden.

4.3.1. Scannen der Vorlage

In Abbildung 4.2 ist die Beziehung zwischen allen im Scan-Prozess involvierten Klassen dargestellt. Sobald ein *Generator*-Objekt instanziert wird, startet das Scannen der Vorlage. Für jedes relevante Power Point Element in der Vorlage (Präsentation, Master-Slide, Slide oder Kommentar) wird ein entsprechendes Template-Objekt instanziert, in welchem alle nötige Informationen über das Element gespeichert werden. Die Beziehung zwischen den verschiedenen Template-Objekten ist baumartig: Jede *Generator*-Instanz hält die Referenz zu genau einer *TemplatePresentation*-Instanz, welche selber eine Liste von *TemplateMasterSlide*-Instanzen und eine von *TemplateSlide*-Instanzen referenziert. *TemplateSlide*-Instanzen verweisen ihrerseits je auf eine Liste von *TemplateComment*-Instanzen.

Jedes Template-Objekt sammelt auch Informationen über alle für sich relevanten und

im Prozess angetroffenen Bausteine. In der Vorlage werden Kriterien in Kommentaren geschrieben. Jede *TemplateComment*-Instanz ist also dafür zuständig, ihre eigenen Kriterien zu identifizieren. Jede *TemplateSlide*-Instanz sammelt die von den *TemplateComment*-Instanzen identifizierten Kommentare, und die *TemplatePresentation*-Instanz fügt ihrerseits alle Kommentare, welche in der Vorlage angetroffen wurden, zusammen. Ein ähnliches Vorgehen erlaubt es, die Platzhalter in den Slides und Masterslides zu finden (mit den Platzhaltern haben Kommentare nichts zu tun, da Platzhalter direkt im Text sind). Auch hier werden alle in den Slides und MasterSlides identifizierte Platzhalter schliesslich in der *TemplatePresentation*-Instanz zusammengetragen.

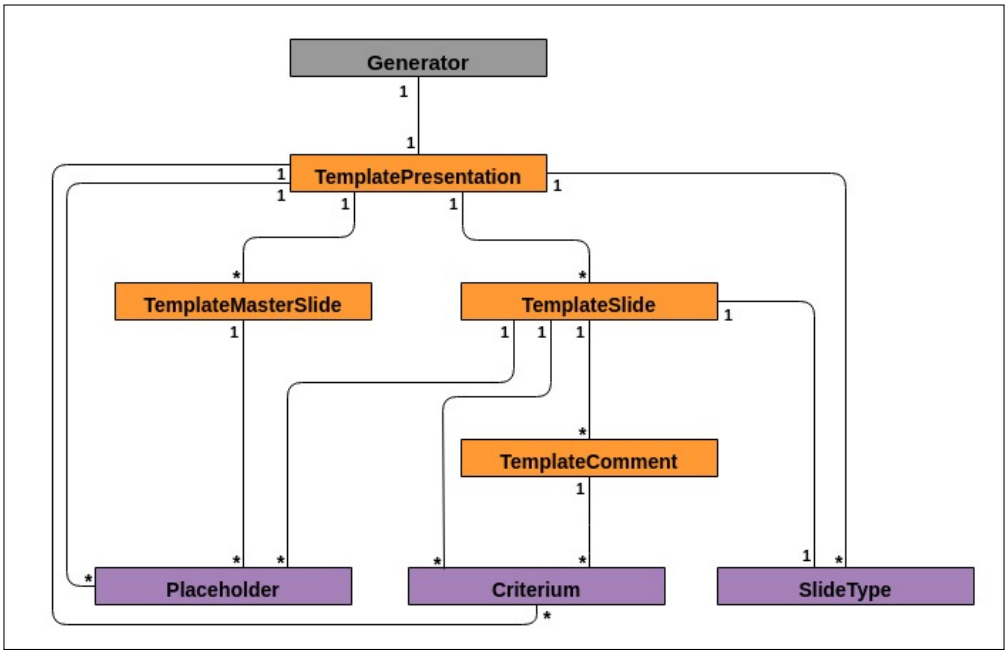


Abbildung 4.2.: *Beteiligte Klassen im Scan-Prozess*

Etwas speziell hingegen ist das Prozedere nach welchem der Typ der untersuchten Slide jeweils festgelegt wird. Jede *TemplateSlide*-Instanz wird zuerst als normale Slide („COMMON“-Typ) gekennzeichnet. Wird in einem Kommentar die passende Syntax angetroffen (siehe das Dokument mit Erklärungen zur Aufbereitung der Vorlagen im Anhang A), wird der Typ der Slide entsprechend aktualisiert und in der *TemplateSlide*-Instanz gespeichert. Handelt es sich um einen Verweis auf Datensätze (siehe Unterabschnitt 3.2.3), prüft das instanzierte *SlideType*-Objekt im verwiesenen Ordner, welche Textdateien im ersten vorhandenen Datensatz überhaupt existieren, und speichert den Namen jeder dieser Datei. Wenn die *TemplateSlide*-Instanz später nach Platzhaltern innerhalb der Slide sucht, prüft sie jeweils, ob der identifizierte Platzhalter dem Namen einer Datei in der *SlideType*-Instanz gleicht. Ist dies der Fall, wird der Platzhalter nicht als sonstiger Platzhalter (siehe Unterabschnitt 3.2.1), sondern als Verweis auf eine Infor-

mationsquelle betrachtet. In der *TemplatePresentation*-Instanz werden ausserdem alle in der Vorlage vorhandenen Slide-Typen gespeichert, die auf externe Daten verweisen (d.h. nicht die reservierten Typen „COMMON“, „TOC“ und „CHAPTER“).

Werden im Scan-Prozess Probleme angetroffen (die Vorlage hat eine falsche Dateiendung oder enthält gar keine Bausteine, usw.), wirft der Generator eine *GeneratorException* mit passender Meldung. Kann der Prozess erfolgreich abgeschlossen werden, kann die Webapplikation alle in der Vorlage gefundenen Platzhalter, Kriterien und Verweise auf Datensätze abrufen. Kriterien werden erst dann nach Bedingungen sortiert. Innerhalb des Generators existiert jedes Kriterium als unabhängiges Element.

4.3.2. Erstellung der Offerte

Abbildung 4.3 illustriert die wichtigsten Schritte im Kernprozess des Generators: die tatsächliche Erstellung der generierte Offerte. Zuerst wird die Vorlage kopiert und eine temporäre Datei für die Offerte kreiert (am Ende des Prozesses gibt der Generator den Pfad zu dieser Datei zurück). Danach wird bei jeder Masterslide geprüft, ob sie einen Platzhalter enthält (bzw. ob ihre entsprechende *TemplateMasterSlide*-Instanz beim Scan-Prozess Referenzen auf Platzhalter gespeichert hatte). Ist dies der Fall, werden alle Platzhalter durch die passende Angabe des Benutzers/der Benutzerin ersetzt, bevor die nächste Masterslide geprüft wird.

Wurden alle Masterslides bearbeitet, geht der Generator weiter zum eigentlichen Kern des Prozesses: Die Bearbeitung aller Slides aus der für die Abwicklung des Prozesses kopierten Vorlage. Dabei geht der Generator, Slide nach Slide, nach dem gleichen Prozedere vor. An allererster Stelle wird geprüft, ob die Slide überhaupt in der Offerte bleiben soll oder nicht. Passt kein Kriterium, wird die Slide gelöscht. Ist mindestens ein passendes Kriterium vorhanden oder wurde die Slide gar nicht mit Kriterien gekennzeichnet, bleibt die Slide hingegen in der Offerte (siehe Unterabschnitt 3.2.2) und wird weiterverarbeitet.

Spezielle Slide mit Datensätzen füllen

Bei jeder Slide, die in der Offerte bleibt, wird zuerst geprüft, inwiefern es sich um eine „spezielle Slide“ handelt - also eine Slide mit Verweis auf Datensätze (sei bemerkt, dass bei diesem Schritt ebenfalls geschaut wird, ob die Slide als Inhaltsverzeichnis oder Kapitel markiert wurde, damit der Generator sich allenfalls merken kann, welche Slides im Inhaltsverzeichnis registriert werden sollen). Soll die Slide mit externem Material gefüllt werden, wird sie kopiert. Anschliessend wird eine Gruppe von Elementen nach der andere mit allen passenden Inhalten zu einem bestimmten Datensatz gefüllt. Ist die Slide voll, wird die ursprüngliche Slide ein weiteres Mal kopiert und wie die erste gefüllt - bis alle vom/von der BenutzerIn gewählten Datensätze verarbeitet werden konnten. Umgekehrt werden alle Gruppen von Elementen, welche am Schluss nicht mit Datensätzen gefüllt wurden, einfach gelöscht.

Alle neu kreierten Slides müssen in der Präsentation registriert werden. Dazu muss eine neue ID für die Slide erzeugt werden (was auch heisst, dass geprüft werden muss, welche IDs in der Präsentation bereits existieren). Damit die neue Slide von der Reihenfolge her direkt nach der letzten mit Datensätzen gefüllten Slide kommt, muss sie in der *SlideIdList* der Präsentation am richtigen Ort eingetragen werden (siehe Abschnitt 4.2). Auch die Bilder müssen auf Slide-Ebene richtig registriert werden, damit sie am Ende zu sehen sind: Ein neuer *ImagePart* muss dazu kreiert werden und eine Referenz zu diesem „Part" muss im *Picture*-Element selber innerhalb der Slide eingetragen werden.

Beim Einfügen der Text-Inhalte wird jeweils ein neues Element (*TextBody*) mit passendem Text kreiert. Dieses ersetzt das alte Text-Element in der Slide. Damit der neue Text aber das ursprüngliche Layout vom alten Element behält, werden alle passende „properties" (siehe Abschnitt 4.2) zum neuen Element kopiert. Es wird auch sichergestellt, dass zu lange Texte ihre Grösse automatisch anpassen, um innerhalb des vordefinierten Feldes zu bleiben. Leider findet dieses Anpassen aber erst richtig statt, wenn ein/e BenutzerIn mit dem Text interagiert. Aus diesem Grund muss der/die BenutzerIn nach dem Generieren der Offerte alle betroffene Textfelder manuell anklicken, damit der Text nicht mehr über den Rand seines Felds hinausgeht. Dieses Problem hat anscheinend mit Open XML SDK selber zu tun.

Platzhalter ersetzen

Als nächster Schritt bei der Verarbeitung der Slide wird geprüft, ob diese Platzhalter im Text beinhaltet, welche ersetzt werden sollten. Dieser Schritt kommt erst nachdem allfällige Datensätze in die Slide eingefügt wurden, um zu verhindern, dass Platzhalter, welche für das Einfügen externer Daten benutzt werden, mit den sonstigen Platzhaltern kollidieren. Ausserdem werden beim Einfügen von Datensätzen teilweise neue Slides kreiert. Indem die Slides zuerst kreiert und nachher weitervearbeitet werden, wird sichergestellt, dass alle Slides in der schliesslich erstellten Offerte sicher durch alle Anpassungsschritte gegangen sind.

Speziell, bezüglich dem Ersetzen von Platzhaltern, ist, dass Texte in Slides nicht mit normalen Zeilenumbrüchen arbeiten, sondern mit dem XML-Element *Break*. Texte können also nicht wie normale *Strings* behandelt werden: Der Generator muss bei jedem *TextBody* prüfen, ob dieses ein *Break*-Element beinhaltet, um letzteres allenfalls am richtigen Ort zu behalten. Ansonsten würden Zeilenumbrüche beim Ersetzen der Platzhalter plötzlich verschwinden.

Elemente innerhalb einer Slide selektiv löschen

Bei jeder Slide muss der Generator auch prüfen, ob Kommentare mit Kriterien direkt auf Textelemente, Bilder oder Grafik platziert wurden. Dazu wird zuerst geprüft, ob Kommentare in der Slide Kriterien beinhalten, welche vom/von der BenutzerIn eigentlich

nicht gewählt wurden. Ist dies der Fall, wird bei jedem dieser Kriterien geschaut, ob das entsprechende Kommentar über ein Element platziert wurde, um dieses Element allenfalls zu löschen.

Die Implementierung dieser Funktionalität erwies sich als relativ schwierig, weil Kommentare in Power Point Präsentationen - im Gegensatz zu Kommentaren in Word-Dokumenten - nicht einem Element im Dokument zugewiesen werden. Nur die Position des Kommentars auf der Slide wird registriert (wird ein Element innerhalb der Slide versetzt, bleibt der Kommentar also am alten Ort stehen und muss allenfalls manuell versetzt werden). Die Messeinheit für die Koordinaten beim Kommentar ist aber nicht die gleiche wie die, welche bei Elementen innerhalb der Slide verwendet wird. Konkret wird innerhalb der Slide mit English Metric Units (oder EMUs) gearbeitet[1], während die bei den Kommentaren benutzte Einheit trotz langen Recherchen nicht identifiziert werden konnte. Damit das selektive Löschen funktioniert, wurde zur Umwandlung der Koordinaten in EMUs eine magische Zahl kalkuliert, welche sich nach mehreren, auch teilweise extremen Tests aber als stabil erwies.

Kommentare löschen und weitere Schritte

Bei jeder Slide, die in der Offerte bleibt, müssen am Schluss alle Kommentare gelöscht werden. Dies geschieht auf eine sehr einfache Art indem das XML-Element, welches die Referenzen auf Kommentare innerhalb einer Slide beinhaltet (nämlich *SlideCommentsPart*), einfach gelöscht wird.

Enthält eine Offerte ein Inhaltsverzeichnis, wird am Schluss die passende spezielle Slide mit dem Titel - und allenfalls dem Bild - jeder Slide gefüllt, welche während dem bisherigen Prozess als Kapitel registriert wurde. Dies geschieht auf ähnlicher Art wie das Füllen von speziellen Slides mit externen Datensätzen. Damit die Kapitel im Inhaltsverzeichnis aber in der richtigen Reihenfolge erscheinen, wird in der *SlideIdList* auf Präsentation-Ebene - d.h. die Liste aller Slides, welche im Dokument vorhanden sind, inkl. Reihenfolge - geschaut, wann überhaupt welche Kapitel-Slides in der Offerte vorkommen.

Am Ende des Gesamtprozesses werden schliesslich alle Referenzen auf Kommentare auf Präsentation-Ebene gelöscht.

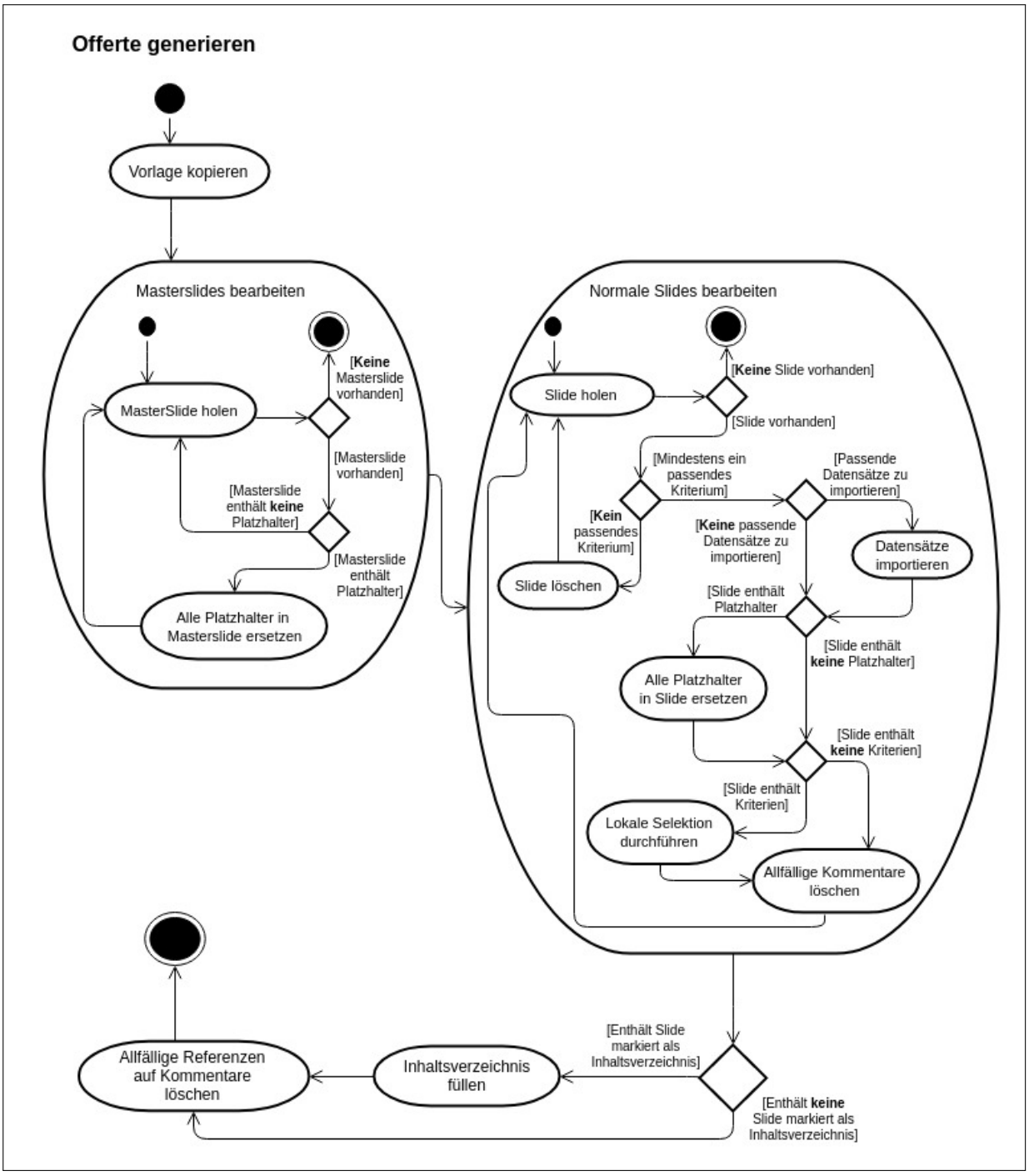


Abbildung 4.3.: *Aktivitätsdiagramm des Erstellungsprozesses der Offerte*

4.4. Prototyp der Webapplikation

Als Prototyp für die Webapplikation wurde ein relativ einfacher „Model-View-Controller“ entwickelt. Als Grundlage dazu wurden die Struktur und das Layout benutzt, welche beim Erzeugen eines neuen ASP.NET-Projekt in Visual Studio automatisch mitgegeben werden. Sechs „Views“ wurden insgesamt programmiert: Eine zur Auswahl der Vorlage, eine zur Eingabe der passenden Inhalte anstelle von Platzhaltern, eine zur Auswahl der richtigen Kriterien, eine zur Auswahl von Datensätzen, eine um am Schluss die Offerte zu erstellen - und eine zur Darstellung allenfälliger Fehlermeldungen.

Die „Views“ referenzieren vier verschiedene Modelle: Vorlagen, Platzhalter, Kriterien und Datensätze. Ein fünftes Modell wurde als Hilfekonstrukt entwickelt: Die Offerte. Jede/r BenutzerIn der Webapplikation arbeitet mit einem *Offer*-Objekt, welches jeweils in der Session des/der BenutzerIn gespeichert ist.

Im einzigen Controller, welcher den ganzen Prozess verwaltet, wird beim Starten der Webapplikation zuerst im passenden Ordner nach allen verfügbaren Offerten-Vorlagen gesucht. Ab dem Moment, wo der/die BenutzerIn eine Vorlage auswählt und zum nächsten Schritt geht, wird ein *Generator*-Objekt instanziert. Dieser scannt die Vorlage, sodass die Webapplikation schliesslich alle für die Erstellung der Offerte nötige Informationen Schritt für Schritt darstellen kann. Dabei werden alle Angaben vom/von der BenutzerIn zuerst gesammelt, und am Ende - konkret: beim Klicken des Knopfs „Offerte erstellen“ - werden sie der *Generator*-Instanz mitgegeben, damit diese die richtige Offerte erstellen kann. Ist dieser Prozess abgeschlossen, wird das neu kreierte Dokument automatisch heruntergeladen (es ist nicht trivial, Power Point Dokumente im Browser darzustellen - auf das wurde bei diesem einfachen Prototyp deshalb verzichtet).

5. Ergebnisse

In den rund dreieinhalb Monaten, die für die Entwicklung des Generators als Bachelorarbeit vorgesehen waren, wurde eine funktionsfähige Lösung entwickelt. Interessant wäre jedoch, wenn *Zühlke* diese Lösung auch in Betrieb nehmen würde. Klar müsste der Generator dazu noch an das firmeninterne System von *Zühlke* angebunden werden und gewisse Anpassungen am Code wären bestimmt ebenfalls nötig (u.a. damit der Generator in der Lage wäre mit anderen Arten von Datensätzen als mit einfachen Informationen in Form von Text- und Bild-Dateien umzugehen). Damit das Inbetrieb nehmen aber überhaupt in Frage kommt, müsste der Mehrwert des Generators für *Zühlke* jedoch eindeutig sein. Gestützt auf experimentellen Auswertungen und auf Rückmeldungen von Mitarbeitenden bei *Zühlke* wird im vorliegenden Kapitel dieser Mehrwert diskutiert.

5.1. Allgemeine Ergebnisse

Die entwickelte Lösungen entspricht allen ursprünglichen Anforderungen von *Zühlke* an den Generator (siehe Abschnitt 2.2):

- Der Generator ist eine selbstständige .NET Bibliothek, welche von einer Webapplikation abgerufen werden kann.
- Alle Unterlagen für die Erstellung der Offerte können zentral gespeichert und verwaltet werden.
- Anhand von Input-Parametern wird eine Power Point Präsentation mit Inhalten generiert, welche den Angaben des/der BenutzerIn entsprechen.
- Die generierte Präsentation kann und muss bis zur fertigen Offerte manuell weiterbearbeitet werden.
- Inhalte sowie Logik für die Auswahl dieser Inhalte können einfach von einem/einer AdministratorIn angepasst werden, ohne dass etwas am Code des Generators verändert werden muss.
- Der Generator ist auch in der Lage, mit externen Informationsquellen umzugehen - zurzeit jedoch nur wenn die externe Datenbasis in Form von Text- und Bild-Dateien in einer Ordnerstruktur gespeichert ist.
- Mit den getesteten Vorlagen hat die Generierung des Dokuments nie länger als 20 Sekunden gedauert (der Generator wurde jedoch nie mit Vorlagen getestet, die

mehr als 70 Slides und 15 unterschiedliche Bedingungen enthielten, bei grösseren und komplexeren Vorlagen steigert sich diese Erstellungszeit sicher).

5.2. Mündliche Rückmeldungen

Gegen Ende der Entwicklung wurde die Lösung mit zwei Mitarbeitenden von *Zühlke* diskutiert: Christian Moser, Initiant des Projekts und Ansprechperson bei *Zühlke* während dem ganzen Prozess, und Martina Duerr, Leiterin des Bid Teams (siehe Abschnitt 2.3). Im Allgemeinen waren die Rückmeldungen positiv: Der Generator würde einen guten Teil von dem abdecken, was beim Offertenschreiben überhaupt automatisiert werden könnte. Ausserdem würde die Lösung eine gute Grundlage für weitere Entwicklungen und Verfeinerungen bieten. Mehrere interessante (Kritik-)Punkte wurden angesprochen.

Erstens erwies sich der Name Generator als verwirrend: Damit es allen Mitarbeitenden, welche den Generator benutzen, klar ist, dass die generierte Offerte kein fertiges Produkt, sondern erst einen ersten Entwurf ist, müsste den Namen allenfalls angepasst werden. Diese wichtige Eingrenzung müsste ausserdem bei der Einführung des Generators klar kommuniziert werden, um die Qualität der Offerten sicherzustellen.

Zweitens stellte es sich heraus, dass es durchaus interessanter sein könnte, mit (relativ) vielen unterschiedlichen mittelgrossen Vorlagen zu arbeiten als mit einer kleinen Menge grossen und womöglich auch gleichartigen Vorlagen. Dies würde auch bedeuten, dass man bei jeder Vorlage mit tendenziell weniger Kriterien arbeiten würde. Die Komplexität innerhalb jeder Vorlage würde also sinken, dafür müssten mehr Vorlagen - mit teilweise auch redundanter Arbeit - verwaltet werden. Interessant dabei wäre aber, dass die Vorlagen weniger genereller Natur sein müssten, und dass das generierte Dokument der fertigen Offerte somit sogar näher kommen könnte. Ausserdem könnten verschiedene Leute aus unterschiedlichen Abteilungen die Verantwortung für eine oder mehrere Vorlagen tragen, statt dass ein oder eine AdministratorIn die ganze Arbeit übernimmt.

Drittens kam zwischen den Zeilen die Frage zum Ausdruck, inwiefern es tatsächlich zu einer Zeitersparnis führen würde, mit dem Generator statt wie heute zu arbeiten. Klar könnten Dokumente anhand von wenigen Klicks erstellt werden. Damit der Generator sinnvoll eingesetzt werden könnte, müssten die Vorlagen aber sauber verwaltet werden, was natürlich viel Zeit in Anspruch nehmen würde - vermutlich vom Bid Team. Vielleicht würde jeder, der sich bis jetzt Zeit nahm, um einen halbwegs guten ersten Offerten-Entwurf zu verfassen, neu den Generator benutzen, mit einem aber weniger fortgeschrittenen Dokument am Ende des Prozesses. Die restliche Arbeit bis zur fertigen Offerte müsste also stärker von anderen getragen werden - vielleicht auch hier in erster Linie vom Bid Team. Klar hat die Frage nach dem tatsächlichen Zeitersparnis nicht direkt mit dem Generator, sondern mehr mit interner Arbeitsverteilung zu tun. Im Rahmen der Bachelorarbeit wurde aber tatsächlich erst eine kleine und einfache Lösung entwickelt und die Frage ist berechtigt, inwiefern der Generator nicht in der Lage sein müsste, mehr zu automatisieren, damit er *Zühlke* im täglichen Betrieb einen tatsächlichen Mehrwert bringt. Ist dieser

Mehrwert für die Mitarbeitende nicht offensichtlich, wird der Generator nicht breit genug benutzt, damit eine saubere Wartung der Vorlagen Sinn macht. Werden die Vorlagen nicht regelmässig aktualisiert, fällt auch einer der Hauptvorteile des Generators, nämlich dass beim Offertenschreiben alle Mitarbeitende von *Zühlke* die gleichen aktuellen und hochqualitativen Vorlagen benutzen.

Im Rahmen der Diskussion mit Christian Moser und Martina Duerr entstand die Idee, klein zu anfangen: Der Generator könnte zuerst nur für die Erstellung von einfachen Offerten für ein oder zwei spezifischen Stammkunden gebraucht werden. Gerade bei diesen Kunden könnte eine halb Automatisierung des Offertenschreibens nämlich mit eindeutigem Zeitgewinn verbunden sein, denn die Aufträge sind oft klein und einfach, und vieles wiederholt sich von Mal zu Mal. Ausserdem gäbe es nur eine oder zwei relativ einfache Vorlagen zu verwalten. Erste Erfahrungen mit dem Generator könnten somit gesammelt werden.

5.3. Experimentelle Auswertungen

Um den Generator anhand von konkreten Beispielen zu testen, wurden fünf Offerten von *Zühlke* aus den vergangenen zwölf Monaten ausgewählt und unter die Lupe genommen. Es wurden möglichst aktuelle Beispiele genommen, damit der Test aussagekräftig ist. Die Auswahl der Offerten erfolgte durch Christian Moser, Ansprechperson bei *Zühlke*. Dabei achtete er darauf, Beispiele unterschiedlicher Natur, Länge und Komplexität zu nehmen (siehe Tabelle 5.1):

- Offerten für Kunden aus vier verschiedenen Branchen;
- Eine einfache und zwei komplexen Offerten, aber auch zwei Offerten durchschnittlicher Komplexität;
- Offerten für Aufträge in der Höhe von 250'000 bis 5'000'000 Franken (mit allerdings nur ein Beispiel für einen Auftrag über eine Million);
- Offerten von knapp zehn bis fast 50 Slides.

Es ist an dieser Stelle noch wichtig hervorzuheben, dass *Offerte 5* dem einfachen Fall entspricht, welcher im Rahmen des Gesprächs mit Christian Moser und Martina Duerr als möglicher Einstieg für den Einsatz des Generators diskutiert wurde (Abschnitt 5.2), nämlich eine einfache Offerte für einen Stammkunden.

Ziel des Tests war zu ermitteln, welcher Anteil jeder Offerte schätzungsweise (in Prozenten) durch den Generator hätte automatisch generiert werden können. Der Test erfolgte also rückblickend, mit fertigen Offerten als Messgegenstände. Massgebend war nur der Inhalt jeder Offerte und nicht die Zeit, welche zur Erstellung dieser Inhalte in Anspruch genommen wurde. Dieses Vorgehen wurde ausgewählt, weil es ansonsten zu aufwändig gewesen wäre, im Nachhinein zu rekonstruieren, wie viele Stunden und Minuten in welchen Inhalten investiert wurden. Dies hat jedoch zur Folge, dass die geschätzten

	Branche	**Datum**	**Komplexität der Offerte**	**Kosten-voranschlag (Näherungswert)**	**Anzahl Slides**
Offerte 1	Öffentlich-rechtlicher Betrieb	23.10.2019	Mittel	300'000 CHF	46
Offerte 2	Versicherung	07.11.2019	Hoch	5'000'000 CHF	43
Offerte 3	Fachorganisation	04.10.2019	Hoch	500'000 CHF	42
Offerte 4	Versicherung	12.12.2018	Mittel	250'000 CHF	29
Offerte 5	Bank	19.09.2019	Tief	250'000 CHF	9

Tabelle 5.1.: *Grundlegende Informationen über die analysierten Offerten*

Prozentzahlen nichts darüber aussagen, wie viel Zeit durch den Einsatz des Generators hätte gewonnen werden können. Die letzten 30% oder 40% einer Offerte sind nämlich vermutlich komplexer und zeitaufwändiger zu schreiben als einfache Inhalte wie Titel eines Kapitels oder Texte über Mitarbeitende, die mit einfachem Copy/Paste in die Offerte eingefügt werden können.

Damit der Generator sinnvolle Ergebnisse liefern kann, muss er auch mit qualitativ hochwertigen und in sich konsistenten Vorlagen arbeiten können (siehe Kapitel 3). Die Qualität des Generators hängt also direkt mit der Qualität der Power Point Vorlagen zusammen. Eine gute Vorlage aufzubereiten und weiterzuentwickeln ist aber eine anspruchsvolle und langwierige Arbeit. Es wäre zeitlich nicht möglich gewesen, für die Durchführung des Tests für jede analysierte Offerte eine passende Vorlage vorzubereiten - v.a. da diese Arbeit in erster Linie durch Mitarbeitende von *Zühlke* hätte erledigt werden sollen. Aus diesem Grund wurde zur Durchführung des Tests angenommen, dass für jede Offerte eine solche Vorlage zur Verfügung stehen würde. Jede Offerte wurde unter dieser Annahme Slide für Slide analysiert. Entschieden wurde jeweils:

- Ob die Slide in der passenden Vorlage überhaupt vorhanden gewesen wäre, und somit durch den Generator automatisch in die Offerte hätte eingefügt werden können oder nicht.
- Wenn die Slide in der generierten Offerte vorhanden gewesen wäre, ob Teile davon noch manuell hätten überarbeitet werden sollen oder nicht.
- Wenn eine Slide noch hätte überarbeitet werden sollen, welcher Anteil bereits vorhanden gewesen wäre, und welcher hätte noch angepasst werden müssen.

Um sicherzustellen, dass diese Fragen nicht willkürlich beantwortet werden, wurden zwei der fünf Offerten zusammen mit Christian Moser getestet. Es ist aber trotzdem klar, dass die Ergebnisse des Tests nicht sehr genau sind. Ziel war eher, eine Grundlage zu schaffen, um über den Mehrwert des Generators nachzudenken.

	Anzahl Slides in der Offerte	**Anzahl voll automatisch generierte Slides**	**Anzahl halb automatisch generierte Slides**	**Automatisch generierte Inhalte (Schätzung)**
Offerte 1	46	11	22	45%
Offerte 2	43	12	15	40%
Offerte 3	42	19	16	60%
Offerte 4	29	18	10	70%
Offerte 5	9	4	5	70%

Tabelle 5.2.: *Ergebnisse der experimentellen Auswertungen*

Tabelle 5.2 fasst die Ergebnisse des Tests zusammen:

- Mit 40% automatisch erzeugten Inhalte schneidet die Offerte für den grössten Auftrag (*Offerte 2*) am wenigsten gut ab. Es handelt sich um eine lange und komplexe Offerte. Nur ca. 1 von 4 Slides könnte mit dem Generator voll automatisch erzeugt, und ca. 1 von 3 müsste nachher noch manuell überarbeitet werden. Fast 4 Slides von 10 wären nach der Generierung gar nicht in der Offerte vorhanden und müssten nachher manuell hinzugefügt werden.
- Etwas überraschend ist, dass *Offerte 1* - eine nur durchschnittlich komplexe Offerte für einen nicht so grossen Auftrag - mit 45% automatisch erzeugten Inhalte am zweit schlechtesten abschneidet. Dies lässt sich vielleicht dadurch erklären, dass *Zühlke* weniger oft Offerten für Kunden im öffentlich-rechtlichen Bereich schreibt, und dass für eine solche Offerte aus diesem Grund mehr neue Inhalte als für Kunden aus anderen Branchen produziert werden müssen. Auch bei dieser Offerte könnte ca. 1 von 4 Slides voll automatisch erzeugt werden. Fast jede zweite Slide müsste aber manuell überarbeitet/ergänzt, dafür aber „nur“ 3 Slides von 10 im Nachhinein hinzugefügt werden.
- Mit 60% automatisch erzeugten Inhalte schneidet *Offerte 3* trotz ihrer Komplexität nicht schlecht ab. Dies hat vielleicht damit zu tun, dass die ausgewählte Lösung verschiedene Vorgehen kombiniert, welche an sich nicht kundenspezifisch sind. Das Material zur Erklärung jedes Vorgehens kann in einer Vorlage also eins zu eins vorhanden sein. Aus diesem Grund wäre im Fall von *Offerte 3* fast jede zweite Slide nach der Generierung voll automatisch in der Offerte vorhanden gewesen. Ca. 1 von 3 Slides hätte dazu manuell noch angepasst und 2 von 10 Slides hätten im Nachhinein manuell hinzugefügt werden müssen.
- Sehr gut schneidet *Offerte 4* ab (70% automatisch erzeugten Inhalte). Dies lässt sich dadurch erklären, dass diese Offerte in vielen Hinsichten den Durchschnittswerten entspricht (durchschnittlich komplex, mittellang, klein- bis mittelgrossen Auftrag).

In anderen Worten weicht sie nur wenig von den Inhalten ab, welche in einer guten, sauberen Offerten-Vorlage vorhanden wären. Nur eine einzige Slide wäre nach der Generierung gar nicht in der Offerte gewesen. Fast 2 von 3 Slides könnten voll automatisch erzeugt werden und ca. 1 von 3 müssten noch manuell angepasst werden.

- Vielversprechend ist, dass *Offerte 5* ebenfalls sehr gut abschneidet (auch hier mit schätzungsweise 70% automatisch erzeugten Inhalte). Es handelt sich um eine kurze und einfache Offerte für einen Stammkunden. Vorlagen könnten also für einen solchen Kunden sehr spezifisch entwickelt werden, damit Offerten sehr rasch generiert werden können, ohne dass allzu viel noch angepasst werden müsste. In dem Beispiel hätte keine Slide hinzugefügt werden müssen. Ca. die Hälfte der Slides wäre nach der Generierung voll automatisch vorhanden und die andere Hälfte müsste noch mit spezifischen Inhalten ergänzt werden.

Fasst man die Ergebnisse des Tests zusammen, lässt sich aussagen, dass in den normalen, einfachen Fällen der Generator über 50% der Schlussofferte automatisch erzeugen kann - was das ursprüngliche Ziel war (siehe Abschnitt 2.2). Umso stärker weicht eine Offerte vom Normalfall ab, desto weniger kann der Generator aber automatisch übernehmen, was auch zu erwarten war. Wie ersichtlich in Abbildung 5.1 spielen dabei Komplexität und Länge der Offerte eine besonders ausprägende Rolle (vermutlich hängen diese beide Faktoren auch direkt voneinander ab). So sind *Offerte 5* und *Offerte 4* einfach bis durchschnittlich komplex und enthalten weniger als 30 Slides. 70% deren Inhalte könnten vom Generator automatisch generiert werden. *Offerte 1*, *Offerte 2* und *Offerte 3* enthalten hingegen über 40 Slides und sind eher komplexer. Schätzungsweise könnte ein weniger grosser Anteil an deren Inhalten (60% und weniger) anhand vom Generator automatisch erzeugt werden.

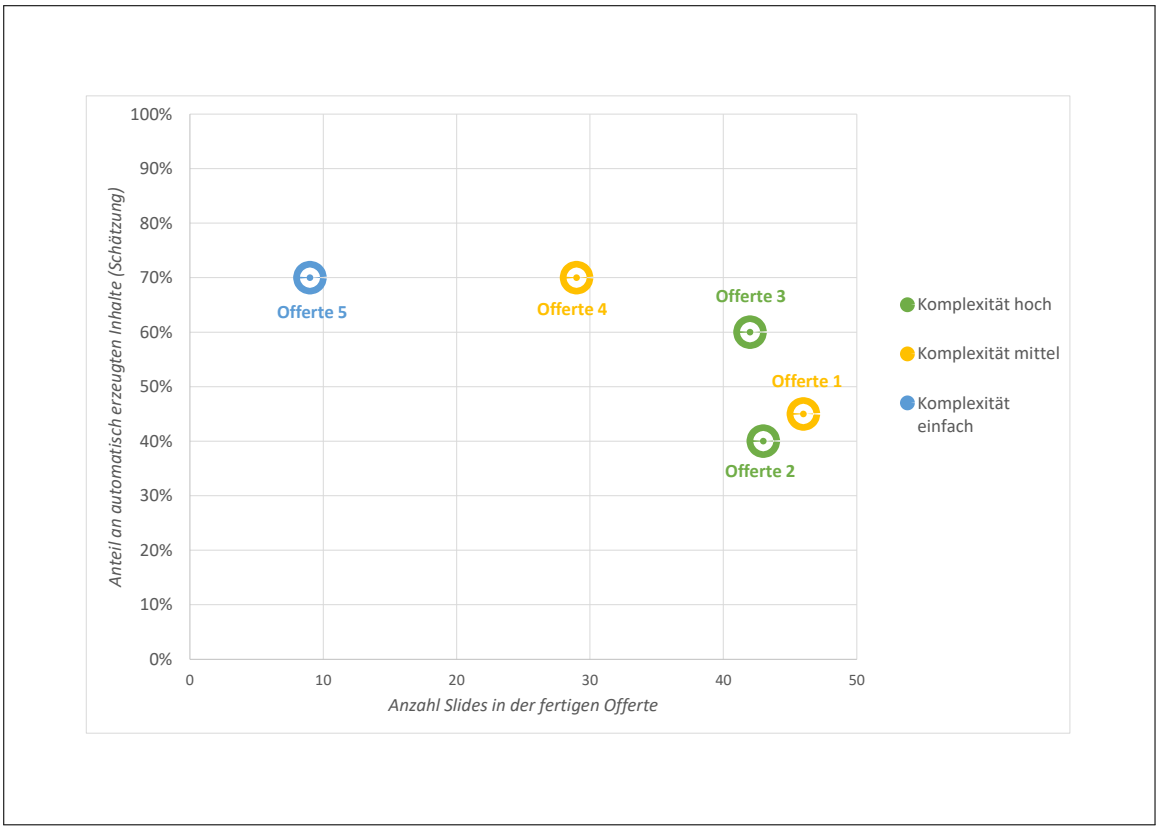


Abbildung 5.1.: *Anteil automatisch erzeugten Inhalten im Verhältnis zur Anzahl Slides*

6. Schlussfolgerung

Seit über 50 Jahren engagiert sich *Zühlke Engineering AG* im Bereich der Produkt-Innovation und seit mehr als 45 Jahren entwickelt die Firma massgeschnittene Software-Lösungen für Kunden. Mit etwa 150 realisierten Projekte pro Jahr gehört *Zühlke* zu den führenden Beratungsunternehmen der Schweiz und Europas im Bereich Software Engineering. Um an neue Aufträge zu kommen, produziert die Firma jedes Jahr über 500 Offerten.

Mit dem Ziel, die erste Phase des Offertenschreibens bei *Zühlke* zu beschleunigen, wurde in den letzten dreieinhalb Monaten ein Offerten-Generator als einfache .NET-Bibliothek entwickelt. Zu Projektbeginn wurden zwei Hauptentscheide gefällt: Erstens soll der Generator möglichst wenig mit verstreuten Inhalten und stattdessen mit einer übergrossen Power Point Vorlage arbeiten, aus welcher überflüssige Inhalte gelöscht werden; zweitens soll die Logik zur Auswahl der passenden Inhalte direkt in der Vorlage gekennzeichnet werden. Dazu wurde eine möglichst schlichte und intuitive Syntax entwickelt.

Jede Vorlage kann mit drei Arten von generischen Bausteinen aufbereitet werden: Platzhalter (direkt im Text), Kriterien (anhand welchen bestimmt wird, welche Inhalte aus einer Vorlage gelöscht werden sollen und welche nicht) und Verweise auf Datensätze (z.B. Informationen über Mitarbeitende, Referenzprojekte, usw.). Wird eine Vorlage ausgewählt, scannt der Generator die Vorlage und sucht nach allen möglichen Bausteinelementen. Diese werden vom Generator also dynamisch erkannt. Die Vorlagen können dementsprechend stetig angepasst und allfällige Fehler sofort korrigiert werden. Gesteuert wird der Generator über eine Webapplikation (Prototyp). Der Generator kann in eine Offerte auch Datensätze einfügen. Zurzeit funktioniert dies jedoch nur, wenn die entsprechenden Datensätze in Form von Text- und Bild-Dateien in einer Ordnerstruktur gespeichert sind.

Welcher Anteil aller bei *Zühlke* produzierten Offerten-Inhalten in der Zukunft vom Generator schliesslich automatisch sollen erstellt werden, ist heute noch undefiniert. Im besten Fall wird die Zulassung von *Zühlke* in Schlieren die implementierte .NET-Bibliothek rasch an ihrem System einbinden, so dass erste einfache Offerten für Stammkunden von einzelnen immer dafür zuständigen Mitarbeitenden schnell generiert werden können. Werden in diesem kleinen Rahmen gute Erfahrungen gemacht, könnte der Generator Schritt für Schritt immer breiter benutzt werden - unter der Voraussetzung, dass intern auch sinnvoll geregelt wird, wer für die Verwaltung welcher Vorlagen zuständig ist, denn ohne gute, regelmässig aktualisierte Vorlagen ist der Generator unnütz.

Sicher ist, dass die entwickelte Lösung erst in einem frühen Stadium ist. Der Generator

weist gewisse Kernkompetenzen auf, hoffentlich ist die Lösung aber solid genug, um in einem immer breiteren Umfang gebraucht zu werden. Der Generator hat auf jedem Fall einen grossen Vorteil im Hinblick auf seine zukünftige Nutzung: Er muss nicht fertige Offerten erstellen, sondern nur ersten (grobe) Fassungen, welche jeweils weiter verarbeitet werden. Da der Output nicht perfekt sein muss, sind die Anforderungen an den Generator auch nicht so hoch. Das Verhältnis zwischen der Zeit, welche in der Verwaltung der Vorlagen investiert wird, und der Zeit, welche bei der Erstellung der Offerten tatsächlich gespart wird, muss einfach stimmen - und sonst nichts.

Literaturverzeichnis

[1] Corneliussen (2010). Points, inches and Emus: Measuring units in Office Open XML. https://startbigthinksmall.wordpress.com/2010/01/04/points-inches-and-emus-measuring-units-in-office-open-xml/. Abgerufen am 22.12.2019.

[2] E-Iceblue (2019). Spire.Presentation for .NET. https://www.e-iceblue.com/Introduce/presentation-for-net-introduce.html#.Xf4Ku1RKg5k. Abgerufen am 20.12.2019.

[3] JetBrains (2019). dotCover: The .NET Unit Test Runner and Code Coverage Tool. https://www.jetbrains.com/dotcover/. Abgerufen am 21.12.2019.

[4] Microsoft (2019). Getting started with ASP.NET MVC 5. https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/getting-started. Abgerufen am 16.12.2019.

[5] Microsoft (2019). Microsoft AspNet Razor. https://www.nuget.org/packages/Microsoft.AspNet.Razor/. Abgerufen am 16.12.2019.

[6] Microsoft (2019). .NET Standard. https://docs.microsoft.com/en-us/dotnet/standard/net-standard. Abgerufen am 16.12.2019.

[7] Microsoft (2019). Open XML SDK: Move a slide to a new position in a presentation . https://docs.microsoft.com/en-us/office/open-xml/how-to-move-a-slide-to-a-new-position-in-a-presentation. Abgerufen am 16.12.2019.

[8] Microsoft (2019). Open XML SDK: Structure of a PresentationML document. https://docs.microsoft.com/en-us/office/open-xml/structure-of-a-presentationml-document. Abgerufen am 16.12.2019.

[9] Microsoft (2019). The Official Microsoft IIS Site. https://www.iis.net/. Abgerufen am 16.12.2019.

[10] Microsoft (2019). Welcome to the Open XML SDK 2.5 for Office. https://docs.microsoft.com/en-us/office/open-xml/open-xml-sdk. Abgerufen am 16.12.2019.

[11] Microsoft (2019). Welcome to the PowerPoint 2010 Primary Interop Assembly Reference. https://docs.microsoft.com/en-us/previous-versions/office/developer/office-2010/ff759900(v=office.14). Abgerufen am 20.12.2019.

[12] NUnit (2019). NUnit. https://nunit.org/. Abgerufen am 16.12.2019.

[13] Zühlke (2019). Facts and Figures. https://www.zuehlke.com/ch/de/ueber-uns/facts-figures/. Abgerufen am 18.09.2019.

Glossar

Bedingung

Eine Bedingung stellt einen Sammelbegriff für verschiedene Werte gleicher Art dar. Für jede Bedingung kann ein konkreter Wert ausgewählt werden, welcher dann den Inhalt der Offerte tatsächlich beeinflusst. Ein Beispiel dafür wäre die Bedingung „Branche“. Die für die Offerte zutreffende Branche kann als Wert ausgewählt werden, damit in der Offerte nur die Inhalte erscheinen, welche für eine bestimmte Branche passen.

Datensatz

Ein Datensatz enthält ein paar strukturierte Informationen für einen konkreten Inhalt, also eine Menge von Attributen, die eingesetzt werden. Beispiele dafür wären alle Informationen über einen Mitarbeiter oder ein Referenzprojekt.

Kriterium

Ein Kriterium stellt ein mögliches Merkmal für die Auswahl eines Inhaltes dar. Es besteht aus einer Bedingung und einem ausgewählten Wert.

Platzhalter

Ein Platzhalter ist ein Textelement, welches direkt im Text einer Offertenvorlage steht und in der tatsächlichen Offerte durch einen anderen Wort ersetzt wird. Ein Beispiel dafür wäre das Wort „Kunde“, welches dann durch den tatsächlichen Name des Kunden ersetzt wird.

Vorlage

Eine Vorlage ist ein Dokument, welches als Grundgerüst für die Herstellung einer Offerte funktioniert.

Wert

Ein Wert kann einer Bedingung zugewiesen werden, um den Inhalt der Offerte zu beeinflussen. Alle Werte, die bei einer Bedingung zur Auswahl stehen, schliessen sich gegenseitig aus. Für die Bedingung „Branche“ wären mögliche Werte „Banken“, „Industrie“, usw.

Anhang

A. Dokument mit Erklärungen zur Aufbereitung der Vorlagen

Am Ende der Arbeit wurde folgendes Dokument mit Erklärungen zur Aufbereitung der Vorlagen - zusammen mit dem Generator (.NET Bibliothek) - der Auftraggeber-Firma *Zühlke* übergeben.

Offerten-Generator

Wie bereitet man eine Power Point Vorlage korrekt vor?

Eine grundlegende Voraussetzung dafür, dass der Generator qualitativ hochwertige Offerten (bzw. erste Entwürfe zum Weiterverarbeiten) herstellt, ist die sorgfältige Aufbereitung und Wartung der Offerten-Vorlagen. Damit die Inhalte für eine Offerte vom Generator jeweils korrekt ausgewählt und eingefügt werden, müssen die Vorlagen sehr konsistent sein. Alle wichtige Informationen zur richtigen Vorbereitung der Vorlagen befinden sich im vorliegenden Dokument. Eine Beispiel-Vorlage mit Kommentaren (in blauen Kästchen) steht ausserdem am Ende des Dokuments zur Illustration aller angeschnittenen Themen.

1. Prämissen

- Alle Vorlagen müssen in einem **gemeinsamen Ordner** gespeichert werden. Die Webapplikation, welche zur Interaktion mit dem Generator entwickelt wird, soll auf diesen Ordner verweisen. Der Pfad zu diesem Ordner muss in der Konfigurationsdatei der **Webapplikation** festgelegt werden.

- Die Vorlagen können **beliebig benannt werden**. Wichtig ist jedoch, dass es sich um Power Point Präsentationen handelt (.pptx-Dateien, wobei auch diese Datei-Endung in der Konfigurationsdatei des Generators angepasst werden kann – es dürfte aber nicht plötzlich ein Word-Dokument sein).

- Die Vorlagen können jederzeit angepasst/ergänzt werden. Der Generator liest jedes Mal alle Platzhalter, Kriterien sowie Verweise auf Datensätze aus der Vorlage dynamisch heraus. Wird ein neuer Platzhalter oder Kriterium eingefügt, oder eine neue Slide aufbereitet, welche mit externem Inhalt gefüllt werden kann, **nimmt der Generator diese Änderungen sofort auf**.

- Die Vorlagen können sich hinsichtlich Struktur und Layout natürlich gleichen – oder auch **bewusst anders aussehen**.

2. Platzhalter

- Platzhalter kommen **direkt in den Slides** oder in den **Masterslides** einer Offerten-Vorlage vor (siehe Beispiel-Slide 1).

- Zur Kennzeichnung eines Platzhalters wird **im Text** eine geschweifte Klammer benutzt: **{Platzhalter}**

- Zwischen den Klammern können alle Buchstaben – auch mit Akzenten – wie auch Lehrzeichen und Unterstriche gebraucht werden. Spezielle Zeichen wie “@” oder “/” dürfen hingegen nicht verwendet werden. Ein erlaubter Platzhalter wäre also **{Platz Hälter_}**, aber **NICHT {Platz@halter.com}**.

- Derselbe Platzhalter kann im Dokument **mehrmals vorkommen**. Er wird dann jedes Mal durch die Eingabe des/der BenutzerIn ersetzt.

- Bei der **Eingabe durch den/die BenutzerIn** sind **alle Zeichen erlaubt**, um den Platzhalter zu ersetzen.

- Es wird **empfohlen**, die Stellen im Text, wo ein Platzhalter vorkommt, **gelb einzufärben** (siehe Beispiel-Slide 1). Der Generator ist nicht in der Lage, allfällige Inputs im Text richtig zu deklinieren. Das Einfärben ermöglicht es, im Nachhinein kurz zu prüfen, ob der Text nach Einfügen der Inputs stimmt. Das Einfärben beeinflusst die Arbeit des Generators allerdings nicht.

3. Kriterien

- Kriterien kommen in den **Kommentarfeldern** einer Offerten-Vorlage vor (siehe Beispiel-Slide 3).

- Zur Kennzeichnung eines Kriteriums in einem **Kommentar** wird folgende Syntax verwendet: **Bedingung = Wert;**

- Wie beim Platzhalter können für Bedingung und Wert alle Buchstaben – auch mit Akzenten – wie auch Lehrzeichen und Unterstriche gebraucht werden. Spezielle Zeichen wie "@" oder "/" dürfen hingegen nicht verwendet werden. Wichtig ist es, nach dem Wert das **Semikolon nicht zu vergessen**.

- Eine Bedingung stellt einen Sammelbegriff für verschiedene Werte gleicher Art dar. Alle Werte, die bei einer Bedingung zur Auswahl stehen, schliessen sich gegenseitig aus. Ein Beispiel wäre **"Branche" als Bedingung** mit **"Industrie" und "Banken" als mögliche Werte**.

- Für jede Bedingung muss **mindestens einen Wert** zur Auswahl stehen.

- Ein Kriterium kann so oft wie gewünscht in einer Vorlage vorkommen. Wird das Kriterium vom/von der BenutzerIn ausgewählt, **bleiben dann alle passende Inhalte in der Offerte**.

- Alle Slides, die durch **KEIN Kriterium** gekennzeichnet werden, **bleiben** automatisch in der Offerte (siehe Beispiel-Slide 1).

- Die Auswahl der Inhalten anhand der Kriterien funktioniert nach dem **"ODER"-Prinzip**. Das heisst, dass wenn mindestens ein Kriterium für einen Inhalt stimmt, bleibt dieser Inhalt in der Offerte. Wird also eine Slide durch "Branche = Industrie;" und "Methode = agil;" gekennzeichnet, und wählt ein/e BenutzerIn die Kriterien "Branche = Banken" und "Methode = agil" aus, wird die Slide **NICHT** gelöscht.

- Wird bei einer Bedingung **KEIN** Wert vom/von der BenutzerIn ausgewählt, werden **alle Inhalte** in der Offerte **gelöscht**, bei welchen die Bedingung vorkommt (entspricht einem leeren Wert für die Bedingung).

- Um eine “UND”-Logik einzuführen, kann durchaus mit Werten wie “**Industrie und agil**” und “**Banken und agil**” gearbeitet werden. In diesem Fall muss der/die BenutzerIn eine dieser Kombinationen auswählen, damit die passende Inhalte in der Offerte vorkommen.

- Es können auch **Inhalte innerhalb einer Slide** filtriert, d.h. gelöscht oder in der Slide gelassen werden (siehe Beispiel-Slide 5). Damit das funktioniert, müssen folgende Regel eingehalten werden:

  - Als Element, welches selektiert werden kann, gilt ein **Bild**, ein **Textfeld**, eine **Grafik**, usw. Das Element soll nicht zu klein sein.

  - Der Kommentar, welcher das passende Kriterium enthält, muss **direkt auf das Element** innerhalb der Slide platziert werden. Entscheidend dabei ist die obere linke Ecke des Kommentarzeichens auf der Slide. Am Besten wird der Kommentar **in der Mitte** des Elements platziert.

  - Wird das entsprechende Kriterium **NICHT** ausgewählt, wird das Element aus der Slide **gelöscht** – unter der Voraussetzung, dass die ganze Slide sonst in der Offerte bleibt.

  - Wenn ein Kommentar **gleichzeitig auf zwei Elementen** zeigt (zwei Bilder übereinander – oder ein vergessenes leeres Textfeld und ein normales Textfeld), wird **nur eines gelöscht**.

  - Das selektive Löschen ist **nicht sehr exakt**. Ist Präzision erfordert, empfiehlt es sich, die Slide zu kopieren und so anzupassen, wie gewünscht. Die Auswahl der passenden Slide findet dann wie übrig über Kriterien auf Slide-Ebene – und nicht über selektives Löschen – statt.

- Es wird **empfohlen**, Kriterien die eine **ganze Slide** betreffen im oberen Ecken links zu platzieren (siehe Beispiel-Slide 5) , damit es klar ersichtlich ist, welche Kriterien nur eine lokale Wirkung und welche eine “ganzslidige” Wirkung haben. Dies hat jedoch keinen Einfluss auf die Arbeit des Generators.

4. Datensätze

- Datensätze sind **strukturierte Inhalte**, welche **ausserhalb der Vorlage** zur Verfügung stehen. Ausgewählte Inhalte können in eine speziell dafür gestaltene Slide innerhalb der Vorlage eingefügt werden (siehe Beispiel-Slide 8).

- Um auf die passende Datenbasis (z.B. Mitarbeitende oder Projekt-referenzen) zu veweisen, wird die vorbereitete Slide anhand eines **Kommentars** wie folgt gekennzeichnet: **!!DATENBASIS**

- Zugelassen für den Verweis werden nur **Grossbuchstaben** und **Unterstriche**, beispielweise !!TEAM oder !!PROJEKTE_ALS_REFERENZEN

- Verwiesen wird auf einen Ordner mit gleichem Namen. Beim Ordner muss der Name aber **klein geschrieben sein**, beispielweise “team” oder “projekte_als_referenzen”.

- Ein neuer Verweis auf eine Datenbasis kann in eine Vorlage **jederzeit eingefügt werden**, solange ein passender Ordner dazu aufbereitet wird.

- Alle Ordner, die mehrere Datensätze gleicher Art enthalten, müssen sich in einem übergeordneten Ordner befinden. Der Pfad zu diesem Ordner muss in der Konfigurationsdatei des **Generators** festgelegt werden.

- Bei der **Gestaltung** der betroffenen Slide-Vorlage müssen **einige Regel** eingehalten werden, damit bei der Erstellung einer Offerte alle Informationen richtig in die Slide eingefügt werden:

  - Alle Textfelder und das optionale Bild, welche zu **EINEM** Datensatz gehören, müssen zusammen **gruppiert** werden (Gruppierung-Funktion von Power Point). Gibt es pro Datensatz nur ein Textfeld und kein Bild, muss der Textfeld natürlich nicht gruppiert werden (denn das wäre nicht möglich).

  - In jedem Textfeld muss ein **Platzhalter** eingefügt werden. Dieser Platzhalter wird vom Generator aber nicht als sonstiger Platzhalter (siehe Punkt 2) betrachtet. Wenn z.B. der Name eines Mitarbeitenden in einem Textfeld eingefügt werden soll, soll im passenden Textfeld **{Name}** geschrieben werden.

  - In jedem Textfeld darf **nur EIN Platzhalter** – und kein weiterer Inhalt – vorhanden sein. Ansonsten könnte das Einfügen in die Slide unerwartete Wirkungen haben (Inhalte falsch eingefügt, Gruppen unerwartet gelöscht, usw.).

  - Der Platzhalter kann wie gewünscht **gestaltet werden** (Schriftart, Schriftgrösse, usw.). Der hinzugefügte Text behält dann diese Gestaltung.

  - Es dürfen für jeden Datensatz **so viele Textfelder wie nötig** kreiert werden.

  - Pro Datensatz darf auch **ein Bild** – aber nicht mehr – vorhanden sein. Der Bild-Vetreter soll von der Grösse her so gestaltet werden wie gewünscht und wird später durch das richtige Bild ersetzt.

  - In einer Slide darf es auch **mehrere Datensätze gleicher Art** geben. Eine Gruppe von Textfeldern und Bild darf also mehrmals kopiert werden, damit z.B. mehrere Teammitglieder zusammen in einer Slide vorkommen. Theoretisch darf auch jede Gruppe anders gestaltet werden, solange die Platzhalter die gleiche bleiben.

- Im jetzigen Stadium des Generators müssen alle Informationen zu einem Datensatz in einem Ordner gespeichert werden (z.B. ein Ordner für den Mitarbeitenden Hans Muster). Damit beim Einfügen die passende Inhalte gefunden werden, müssen folgende Regel eingehalten werden:

- Zu jedem Platzhalter in der Slide muss im Ordner eine **Textdatei (.txt)** vorhanden sein, welche nach dem Platzhalter genannt ist (z.B. "**Name.txt**"). Wird der Datensatz gewählt, wird der Inhalt dieser Datei am richtigen Ort in die Slide eingefügt.

- Wird mit einem Bild gearbeitet, soll im Datensatz-Ordner eine **Bilddatei (.jpg)** vorhanden sein. Wird der Datensatz gewählt, wird dann das Bild aus der Slide mit dem Bild aus dem Ordner ersetzt.

- Zurzeit arbeitet der Generator mit **".txt"** und **".jpg"**-Dateien. Dies kann in der Konfigurationsdatei des Generators aber angepasst werden.

➢ Eine bestimmte Slide-Vorlage muss in einer Offerte-Vorlage **nur einmal** vorhanden sein. Werden mehr Datensätze eingefügt, als auf der Slide Platz vorhanden ist, wird die Slide-Vorlage einfach **kopiert** und mit dem zusätzlichen Inhalt gefüllt. **Überflüssige Elemente werden gelöscht**.

➢ Sind in der Slide-Vorlage Platzhalter vorhanden, welche **NICHT** auf Dateien verweisen, werden diese als **sonstige Platzhalter** (siehe Punkt 2) betrachtet.

➢ Wird beim Einfügen eines Textes in ein Textfeld **mehr Text eingefügt, als dafür Platz vorhanden ist**, passt sich der Text **nur halb-automatisch** an. Konkret heisst das, dass im Nachhinein eine kleine Änderung im Text manuell vorgenommen werden muss, damit der Text sich – und erst dann – automatisch anpasst. Leider kann die Power Point Präsentation nur in der Interaktion mit einem/einer BenutzerIn erkennen, dass der Text anders formatiert werden sollte.

➢ Es wird **empfohlen**, Slide-Vorlage für Datensätze **NICHT mit Kriterien** zu ergänzen. Da der Generator zuerst alle BenutzerIn-Inputs sammelt, und erst dann die Offerte generiert, kann es sonst plötzlich vorkommen, dass Datensätze für eine Slide-Vorlage zwar ausgewählt wurden, die Slide-Vorlage beim Generieren aber gelöscht wird. In einem solchen Fall kann die Offerte zwar erstellt werden, die Slide mit Datensätzen fehlt dann einfach.

5. Inhaltsverzeichnis

➢ Der Generator ist in der Lage, allfällige **Kapitel** in ein **Inhaltsverzeichnis** zu **registrieren**.

➢ Damit eine Slide vom Generator als **Inhaltsverzeichnis** erkannt wird (siehe Beispiel-Slide 2), muss die entsprechende Slide im **Kommentarfeld** wie folgt beschriftet werden: **!!TOC** (diese Bezeichnung kann in der Konfigurationsdatei des Generators angepasst werden).

➢ Die Slide muss ausserdem **speziell aufbereitet** werden. Die allgemeine Regel zur Aufbereitung der Slide sind die gleiche wie bei den Slide-Vorlagen für Datensätze (siehe Punkt 4 dieser Anleitung). Konkret heisst das:

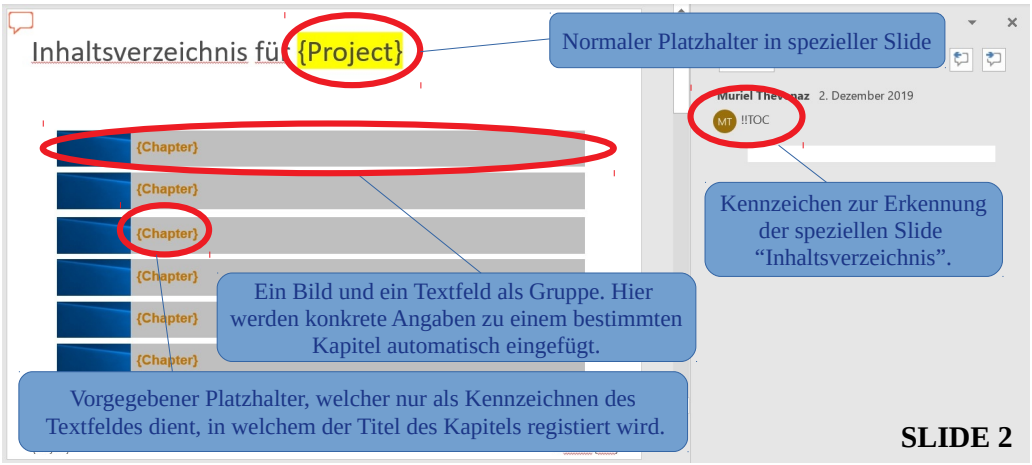
- Jede Gruppe muss genau ein Textfeld enthalten, welches mit dem Platzhalter **{Chapter}** markiert wird. Das Wort “Chapter” kann in der Konfigurationsdatei des Generators geändert werden.
- Es darf auch ein Bild zum Textfeld geben.
- Sind Text UND Bild vorhanden, müssen beide Elemente **gruppiert** werden (Gruppierung-Funktion von Power Point), damit der Generator weiss, dass sie zusammengehören.

➢ Damit eine Slide vom Generator als **Kapitel** erkannt wird (siehe Beispiel-Slide 3), muss diese Slide im **Kommentarfeld** wie folgt beschriftet werden: **!!CHAPTER** (diese Bezeichnung kann in der Konfigurationsdatei des Generators angepasst werden).

➢ In einer “Kapitel-Slide” wird **nur der Titel** der Slide ausgelesen. Enthält die Slide keinen Titel, funktioniert die Generierung der Offerte nicht, stattdessen kommt eine Fehlermeldung.

➢ Wenn im Inhaltsverzeichnis für jedes Kapitel ein Bild registriert werden soll, darf eine “Kapitel-Slide” **nicht mehr als ein Bild** enthalten. Auch hier funktioniert die Generierung der Offerte ansonsten nicht und eine Fehlermeldung kommt stattdessen.


Kommentare
Neu
Es gibt Kommentare auf anderen Folien in dieser Präsentation.
Verwenden Sie die Schaltflächen "Nächster" und "Vorheriger", um sie anzuzeigen.
Zühlke {Client}
«{Project}»
{Project}
© Zühlke {Jahr}
Platzhalter in Slide
Platzhalter in Masterslide
Kein Kommentar in dieser Slide, die Slide bleibt auf jedem Fall in der Vorlage
SLIDE 1


Inhaltsverzeichnis für {Project}
{Chapter}
{Chapter}
{Chapter}
{Chapter}
{Chapter}
{Chapter}
Normaler Platzhalter in spezieller Slide
2. Dezember 2019
!!TOC
Kennzeichen zur Erkennung der speziellen Slide "Inhaltsverzeichnis".
Ein Bild und ein Textfeld als Gruppe. Hier werden konkrete Angaben zu einem bestimmten Kapitel automatisch eingefügt.
Vorgegebener Platzhalter, welcher nur als Kennzeichnen des Textfeldes dient, in welchem der Titel des Kapitels registiert wird.
SLIDE 2

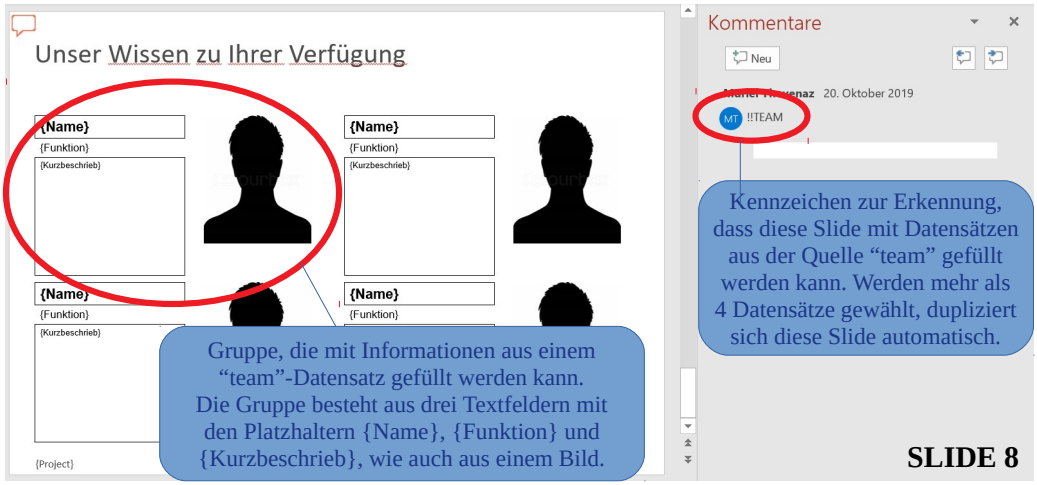

Da es im Inhaltsverzeichnis Platz für ein Bild gibt, wird dieses Bild auch registriert.
20. Oktober 2019
!!CHAPTER
Muriel Thevenaz
Zühlke=kurz;
Muriel Thevenaz
Zühlke=lang;
Kennzeichen zur Erkennung einer Slide, welche im Inhaltsverzeichnis registriert werden soll.
Unsere Firma
{Project}
Titel, welcher im Inhaltsverzeichnis registriert wird.
Kriterien, anhand welchen entschieden wird, ob die Slide in der Offerte bleibt oder nicht.
SLIDE 3


Was wir tun
Wir setzen Ihre Ideen mit Leidenschaft um
Zühlke Engineering AG
Business Development
Competence Center
Customer Solutions
Services Manag
Competence Units
Application Management Services
Cross Functions
Java
.NET
Systems
Solution Centers
Academy
Collaboration & Portals
Digital Consulting
Internet of Things
Enterprise
Systems
Finance & Controll
Human Resources
IT-Center (Group)
Marketing & Communication
Reception & Office Management
Facility Management
Quality Management
{Project}
© Zühlke (Jahr)
Kommentare
Neu
Freitag
Zühlke=lang;
Zühlke=kurz;
Kriterium, damit die Slide ausgewählt wird, wenn viele Informationen über Zühlke in der Offerte vorkommen sollen.
Kriterium, damit die Slide ausgewählt wird, wenn wenige Informationen über Zühlke in der Offerte vorkommen sollen.
SLIDE 4


Allgemein gültiger Kommentar oben links
Zühlke Technology Group AG
Es bleibt nur das Kriterium “Zühlke=lang;”, die Slide bleibt nicht in der Offerte, wenn “Zühlke=kurz” ausgewählt wird.
Unser Leistungsumfang
Wichtigste Zahlen
{Project}
© Zühlke (Jahr)
Kommentare
Neu
Freitag
Zühlke=lang;
Muriel Thevenaz Freitag
Inhalt=Zeichnung;
Lokaler Kommentar: Wird das Kriterium “Inhalt=Zeichnung” nicht ausgewählt, wird die Zeichnung gelöscht – unter der Voraussetzung, dass “Zühlke=lang” ausgewählt wurde, ansonsten würde die ganze Slide gelöscht werden.
SLIDE 5


Zühlke als Partner für Business- & Technologie-Innovation – Alles aus einer Hand und unter einem Dach
Web, Mobile, AR & VR, Blockchain Kompetenz
CX & UX
Agile Consulting
Digital Consulting
Maintenance
{Project}
© Zühlke (Jahr)
Kommentare
Neu
Freitag
Zühlke=lang;
Auch hier bleibt die Slide nicht in der Offerte, wenn “Zühlke=kurz” ausgewählt wird.
SLIDE 6


Kommentare
Neu
Samstag
!!CHAPTER
Neues Kapitel, welches mit Titel und Bild im Inhaltsverzeichnis registriert wird. Da keine Kriterien vorhanden sind, bleibt dieses Kapitel auf jedem Fall in der Offerte.
Unser Team
{Project}
© Zühlke (Jahr)
SLIDE 7


Kommentare
Neu
20. Oktober 2019
!!TEAM
Kennzeichen zur Erkennung, dass diese Slide mit Datensätzen aus der Quelle “team” gefüllt werden kann. Werden mehr als 4 Datensätze gewählt, dupliziert sich diese Slide automatisch.
Unser Wissen zu Ihrer Verfügung
{Name}
{Funktion}
{Kurzbeschrieb}
Gruppe, die mit Informationen aus einem “team”-Datensatz gefüllt werden kann. Die Gruppe besteht aus drei Textfeldern mit den Platzhaltern {Name}, {Funktion} und {Kurzbeschrieb}, wie auch aus einem Bild.
{Project}
SLIDE 8

B. Aufgabenstellung

Folgende Aufgabenstellung hat der Betreuer der Arbeit, Prof. Dr. Luc Bläser, am Anfang des Semesters verfasst.

Aufgabenstellung Bachelorarbeit für Muriel Thévenaz

Offerten-Generator für Engineering-Projekte

1. Auftraggeber und Betreuer

Diese Bachelorarbeit findet in Zusammenarbeit mit *Zühlke Engineering AG* statt.

Ansprechpartner Auftraggeber:

- Christian Moser, Head Competence Unit, Zühlke Engineering AG, Christian.Moser@zuehlke.com

Betreuer HSR:

- Prof. Dr. Luc Bläser, Institut für vernetzte Systeme, lblaeser@hsr.ch

2. Ausgangslage

Zühlke Engineering AG ist ein führendes schweiz- und europaweites Beratungsunternehmen im Bereich Software Engineering und Produkt-Innovation.

Im Rahmen der Gewinnung von Kundenprojekten erstellt Zühlke zahlreiche hochwertige Projekt-Offerten, die auf die spezifische Problemstellung des Projektes, Grösse des Projektes und die verschiedenen technischen und organisatorischen Gegebenheiten des Kunden massgeschneidert sind. In jeder Offerte werden ähnliche Aspekte dargelegt wie Value Proposition, Zielsetzung des Projektes, Vorgehen, Software-Architektur, Cross-Cutting Concerns, Software-Methodik, Team mit Mitarbeiter-Profilen, Projekt-Referenzen, finanzieller Rahmen und so weiter. Die Darlegung dieser Aspekte ist wiederum aber von verschiedenen Parametern abhängig und muss situativ angepasst. Am Schluss müssen alle Teile der Offerte, inkl. Bildsprache, aufeinander abgestimmt werden.

Die aktuelle Erstellung einer Offerte ist sehr zeitintensiv und besteht zu einem gewissen Teil aus wiederkehrenden Mustern und Bauteilen. Den wiederkehrenden Teil der Arbeit soll neu mit einem Offerten-Generator erstellt werden, indem ein möglichst nützlicher Offerten-Entwurf aus den wiederkehrenden Bausteinen und anhand gewissen Input-Parametern zusammengebaut wird. Selbstverständlich muss der Entwurf danach von Mitarbeitern noch spezifisch ausgebaut und abgerundet wird. Mit dem Offerten-Generator soll aber der nicht-kreative Teil der Offerten-Arbeit möglichst weitgehend automatisiert werden.

Der Kern des Generators soll eine .NET Bibliothek sein, welche später von einem Frontend benutzt werden kann. Es basiert auf einem Repository mit den verschiedenen parametrisierbaren Bausteinen (Texte und Bilder), die vom Generator abhängig von den jeweiligen Projektparametern

zusammenkompiliert werden. Um den Generator leicht anpassbar zu halten, würde sich die Benutzung einer geeigneten DSL (Domain Specific Language) anbieten. Als Output des Generators soll ein Powerpoint-Dokument allenfalls über eine Zwischendarstellung (Baumstruktur der Offerte) erzeugt werden. Ein möglicher Architekturansatz ist in Abbildung 1 ersichtlich.

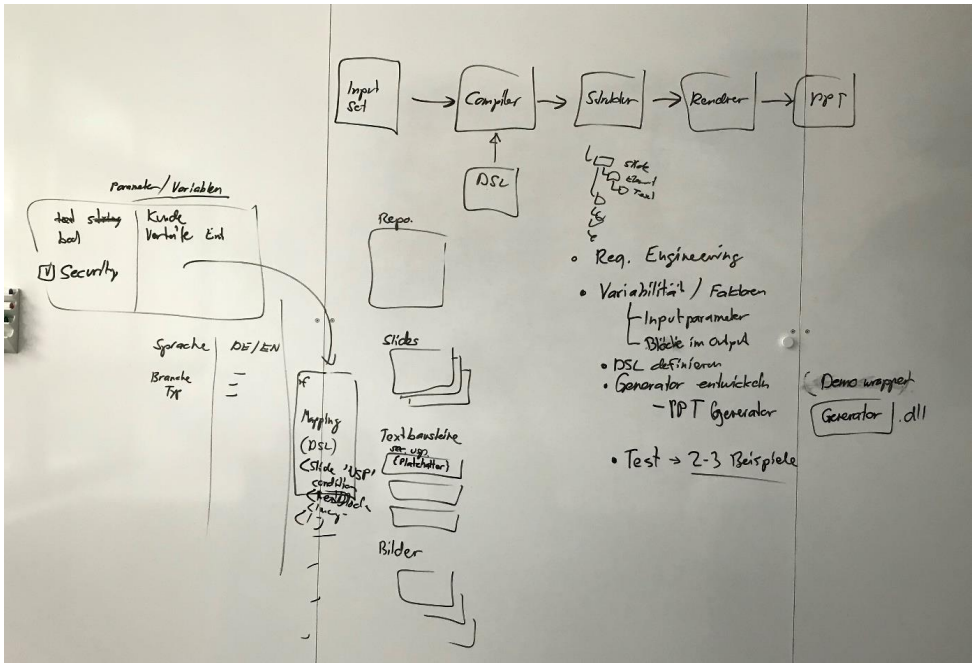


Abb. 1: Skizze einer möglichen Architektur von Christian Moser, Zühlke Engineering AG

3. Ziele und Aufgabenstellung

Das Ziel dieser Arbeit ist es, einen Offert-Generator für die Engineering-Projekte von Zühlke zu entwickeln, der einen möglichst hochwertigen Offerten-Entwurf liefert und so den Anteil der weniger kreativen Arbeit der Offerten-Erstellung soweit wie möglich reduziert.

Die Arbeit hat demnach folgende spezifische Ziele:

- Requirements Engineering: Verstehen und Systematisierung der Anforderungen an eine Offerte und deren Bauteile, Struktur und Aspekte. Zusammenstellung geeigneter anonymisierter Beispielofferten.
- Entwurf einer Software-Architektur, welche die Offert-Generierung möglichst flexibel lässt, um den unterschiedlichen Kundenszenarien gerecht zu werden. Ein besonderer Fokus ist das Modell der Offerten-Bausteine, deren Beziehungen sowie der Parametern, welche die Auswahl der Struktur und der Inhalte einer Offerte beeinflussen. Dabei spielt das Design der Domain-Specific Language eine besondere Rolle.

- Design, Implementation und automatisierte Tests des Generators, allenfalls mit einem kleinen entkoppelten Frontend für Demonstrationszwecke.
- Auswertung des Nutzens des Generators beim Auftraggeber.
- Dokumentation der Software-Architektur, insbesondere mit Spezifikation der Domain Specific Language und des Baustein-/Parameter-Modells.
- Schlusspräsentation der Resultate beim Auftraggeber.

Es empfiehlt sich ein iteratives Vorgehen, bei dem zuerst für ein oder sehr wenige konkrete Beispiele ein Durchstich realisiert und dann ausgewertet wird. Danach kann dieser Prototyp iterativ um weitere Fälle und Bausteine ausgebaut und evaluiert werden.

Die Implementation des Kerns soll in .NET C# erfolgen und als Output soll ein Powerpoint-Dokument geliefert werden.

4. Zur Durchführung

Mit dem HSR-Betreuer finden wöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierende zu veranlassen. Besprechungen mit dem Auftraggeber werden nach Bedarf durchgeführt.

Alle Besprechungen sind von der Studentin mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das dem Betreuer und dem Auftraggeber per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

5. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig digital dem Betreuer, dem Experten/der Expertin, dem Gegenleser/der Gegeleserin sowie dem Auftraggeber abzugeben. Auf Wunsch ist für den Auftraggeber und den Betreuer eine gedruckte Version zu erstellen.

6. Termine

Siehe auch Terminplan des Studiengangs Informatik:

16.9.2019 Beginn der Bachelorarbeit, Ausgabe der Aufgabenstellung durch den Betreuer.

bis 6.1.2019	Erfassung des Abstracts im Online-Tool https://abstract.hsr.ch/ Die Studierenden geben den Abstract für die Diplomarbeitsbroschüre zur Kontrolle an ihren Betreuer/Examinator frei. Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen auf dem Skripteserver zur Verfügung. Der Betreuer/Examinator gibt das Dokument mit dem korrekten und vollständigen Abstract der Broschüre zur Weiterverarbeitung an das Studiengangsekretariat frei.
10.1.2020	**Abgabe des Berichts an den Betreuer bis 17.00 Uhr.** **Hochladen aller Dokumente auf archiv-i.hsr bis 17 Uhr**
bis 14.1.2019	Mündliche BA-Prüfung

7. Beurteilung

Eine erfolgreiche Bachelorarbeit zählt 12 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert.

Für die Beurteilung sind die HSR-Betreuer verantwortlich.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/6
2. Berichte (Abstract, Mgmt Summary, technische u. persönliche Berichte) sowie Gliederung, Darstellung, Sprache der gesamten Dokumentation	1/6
3. Inhalt *)	(1/2)
3.1 Problemanalyse (Vorstudie, Literaturstudium, Anforderungsspezifikation, Anforderungsanalyse, Domainanalyse)	1/6
3.2 Lösungsentwurf (Lösungsvarianten und deren Beurteilung, Variantenentscheid, Konzept, Entwurf)	1/6
3.3 Realisierung und Test	1/6
4. Mündliche Prüfung zur Bachelorarbeit	1/6

*) Die Gliederung des Gesichtspunktes “3. Inhalt” in einzelne Unterpunkte kann den Gegebenheiten der Arbeit angepasst werden. Das Gesamtgewicht des Gesichtspunktes bleibt hingegen bei 50%.

Im Übrigen gelten die Bestimmungen der Abt. Informatik zur Durchführung von Bachelorarbeiten.

Rapperswil, den 12. September 2019

Der verantwortliche Betreuer

Prof. Dr. Luc Bläser
Institut für vernetzte Systeme

Hochschule für Technik Rapperswil