Freie Universität Berlin

Bachelorarbeit am Institut für Informatik der Freien Universität Berlin
Arbeitsgruppe Software Engineering

Automatisierung einer TYPO3-Upgrade-Analyse

Juliane Wundermann
Matrikelnummer: 5107896
wundermaj98@zedat.fu-berlin.de

Erstprüfer: Prof. Dr. Lutz Prechelt
Zweitprüfer: Prof. Dr. Jörn Eichler
Betreuer: Sammy Baghdadi, B. Sc

23.09.2021

Abstract

Deutsch:

Das Ziel der Vorliegenden Arbeit ist zu versuchen, eine TYPO3-Upgrade-Analyse zu automatisieren. TYPO3 ist ein Open Source Content Management System. Im Verlauf der Arbeit werden die Begriffe Upgrade und Update voneinander abgegrenzt. Außerdem wird ein Referenzmodell vorgestellt, nachdem eine TYPO3-Upgrade-Analyse durchgeführt werden kann. Ausgehend von den theoretischen Grundlagen wird ein Interviewleitfaden entwickelt. Dieser dient als Basis bei der Befragung von fünf Experten aus dem Bereich der TYPO3-Upgrade-Analyse. Die Interviews werden mit qualitativer Inhaltsanalyse ausgewertet. Die Aussagen aus den Interviews bilden die Grundlage für die Entwicklung einer Extension, die eine TYPO3-Upgrade-Analyse automatisiert und die zugehörigen Informationen in einem Backendmodul darstellt. Die Arbeit richtet sich vor allem an Personen, die bereits grundlegende Erfahrungen mit TYPO3 haben und ein Hilfsmittel suchen eine Upgrade-Analyse automatisiert durchzuführen.

English:

The goal of this paper is to try to automate a TYPO3 upgrade analysis. TYPO3 is an open source content management system. In the course of the thesis the terms upgrade and update are distinguished from each other. Furthermore, a reference model is presented, according to which a TYPO3 upgrade analysis can be performed. Based on the theoretical basics, an interview guideline is developed. This serves as a basis for interviewing five experts in the field of TYPO3 upgrade analysis. The interviews are evaluated with qualitative content analysis. The statements from the interviews form the basis for the development of an extension that automates a TYPO3 upgrade analysis and presents the related information in a backend module. The work is primarily aimed at people who already have basic experience with TYPO3 and are looking for a tool to perform an upgrade analysis automatically.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel, wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, Datum

Juliane Wundermann

Inhaltsverzeichnis

1 Einleitung ... 1

1.1 Motivation ... 1

1.2 Ziel der Arbeit ... 1

1.3 Aufbau der Arbeit ... 2

2 Grundlagen ... 3

2.1 Open Source ... 3

2.2 TYPO3 ... 3

2.3 Git ... 3

2.3.1 Fork ... 4

2.4 Upgrade oder Update ... 4

2.5 PhpStorm ... 5

2.6 Composer ... 5

2.7 Extbase oder pibase ... 6

2.8 Domain-driven Design ... 6

2.9 Extensions ... 6

2.9.1 Domain Model und Domain Repository ... 7

2.9.2 Hooks ... 7

2.9.3 XCLASSes (Extending Classes) ... 7

2.10 Ns_ext_compatibility ... 8

2.11 Upgrade-Analyse bei familie redlich ... 8

3 Forschungsmethoden ... 13

3.1 Statische Code-Analyse ... 13

3.2 Leitfadengestütztes Experteninterview ... 13

3.2.1 Vorbereitung ... 14

3.2.2 Auswertung ... 16

3.2.3 Abgrenzung qualitativer gegenüber quantitativen Forschungsmethoden ... 17

4 Entwicklung der Extension ... 18

4.1 Anforderungen an das Backendmodul ... 19

4.2 Systeminformationen und Kennzahlen abfragen ... 21

4.3 Scannen der Extensions ... 22

4.4 Entwicklung der Funktion ... 24

4.5 Darstellen der gesammelten Informationen ... 29

5 Abschluss ... 31

5.1 Zusammenfassung ... 31

5.2 Diskussion der Ergebnisse ... 33

5.3 Fazit ... 33

5.4 Ausblick ... 34

6 Literaturverzeichnis ... I

7 Abbildungsverzeichnis III

8 Tabellenverzeichnis III

9 Anhang IV

1 Einleitung

TYPO3 CMS ist ein freies Content Management System, in das mit Hilfe von Erweiterungen (Extensions) neue Funktionen integriert werden können. Viele dieser Erweiterungen stammen von Fremdanbietern. TYPO3 wird beständig weiterentwickelt. Ist die Major-Version zu alt, läuft der Support durch das TYPO3-Team aus. Um den Verlust der Unterstützung zu vermeiden ist in vielen Fällen ein Upgrade der TYPO3-Major-Version problemlos möglich. Im Fall des Upgrades muss für die Fremderweiterungen geprüft werden, ob diese mit der gewünschten Version weiterhin lauffähig sind. Zum Beispiel kann sich die PHP-Version oder die Struktur der Systemkonfiguration ändern. Erweiterungen, die mit den neuen Versionen nicht mehr kompatibel sind, müssen angepasst bzw. ersetzt werden.

1.1 Motivation

Während meiner Arbeit als Werkstudentin bei familie redlich :digital, habe ich einige Upgrade-Analysen durchgeführt. Dabei habe ich mich an dem im internen Wiki enthaltenen Leitfaden zur Durchführung von TYPO3-Upgrade-Analysen bei familie redlich gehalten. Dies war zum einen ein langwieriger Prozess, bei dem ich die Daten der einzelnen Extensions von Hand tabellarisch zusammengeführt und ausgewertet habe. Zum anderen erfuhr ich bei der Nachbesprechung meiner Ergebnisse mit einem erfahrenen Kollegen, dass er in Einzelfällen die Extensions anders kategorisieren würde als im Leitfaden beschrieben. So entstand der Wunsch, diesen Prozess zu automatisieren und dabei die Einteilung der Extensions in Kategorien zu überarbeiten.

Nach Gesprächen mit Kollegen mit viel TYPO3-Erfahrung, stellte sich heraus, dass es für den beschriebenen Prozess keine allgemein gültige Lösung gibt. Sammy Baghdadi berichtete, dass nach Veröffentlichung seiner Bachelor-Arbeit mit dem Thema „Konzeption eines Vorgehens für die Durchführung eines Upgrades von TYPO3“ viele Kommentare aus der TYPO3-Community kamen, wie der Prozess der Analyse am besten durchgeführt werden könnte. Dies bestätigte sich nach einer Online-Suche zu diesem Thema. Als ein Ansatz wird die Extension „ns_ext_compatiblity“ genannt, die einen guten Überblick über den aktuellen Stand eines Systems liefern soll.

1.2 Ziel der Arbeit

Die Dauer des Upgrades wird häufig mit dem Stundensatz des jeweiligen Unternehmens multipliziert und bildet damit einen großen Bestandteil der finalen Kosten für den Kunden. Deswegen soll möglichst vor Beginn der Umsetzung geprüft werden, wie lange das Upgrade dauern wird. Hiermit soll bezweckt werden, dass sowohl die Kosten vor Beginn des Upgrades für den Kunden bekannt als auch die Kosten (Personal, Hardware, Räumlichkeiten, …) nach Fertigstellung des Projekts des beauftragten Unternehmens gedeckt sind.

Die Aufwandsermittlung für die Upgrades der Fremderweiterungen erfolgt durch Sichten der einzelnen Erweiterungen auf Quellcodeebene durch einen Entwickler. Dieser Prozess ist einerseits sehr zeitaufwändig und andererseits in wiederkehrende Arbeitsschritte gegliedert. Das Ziel dieser Arbeit ist es, die einmaligen und wiederkehrenden Arbeitsschritte zu identifizieren, diese wenn möglich zu automatisieren und dadurch die Kosten des Upgrades zu senken.

1.3 Aufbau der Arbeit

Zu Beginn findet eine Heranführung an das Thema statt. Es werden grundlegende Begrifflichkeiten geklärt und genutzte Programme vorgestellt. Im Anschluss werden die Forschungsmethoden und der Umgang mit ihnen erklärt.

Im Hauptteil wird auf Basis der geführten Experteninterviews ein Backendmodul entwickelt, dass eine TYPO3-Upgrade-Analyse automatisiert durchführt.

Im Anschluss werden die Ergebnisse diskutiert. Auf dieser Grundlage wird ein Fazit gezogen und ein Ausblick aufgezeigt.

2 Grundlagen

2.1 Open Source

Gemäß dem Gabler Wirtschaftslexikon ist Open Source ein „Konzept, nach dem Programme mit ihrem Quellcode ausgeliefert werden. Jeder darf den Quellcode einsehen und verändern. Die Open Source Initiative (OSI) definiert Kriterien, die Open Source Software erfüllen soll.“[1] Zu diesen Kriterien gehört unter anderem die Verfügbarkeit des Quellcodes, die freie Weitergabe der Software und, dass eine Software frei von Lizenzbeschränkungen ist.[2]

2.2 TYPO3

TYPO3 ist ein Open Source Content Management System, das in der Scriptsprache PHP implementiert ist.[3]

Die Idee für TYPO3 entstand bei Kasper Skårhøj schon im Jahr 1997. 1998 erschien TYPO3 zum ersten Mal und wurde von der Agentur Superfish.com vertrieben, bei der Skårhøj zu diesem Zeitpunkt angestellt war.

Bis 2000 entwickelt Kasper Skårhøj – jetzt selbstständig – eine Betaversion. Diese wird als Open Source Projekt unter der GPL-Lizenz veröffentlicht.[4] [5]

Der Name „TYPO3“ leitet sich von Typo (Tippfehler) und dem Erfolg der dritten Version ab.

Bis zum Erscheinen des Buchs „Praxiswissen TYPO3 CMS 9 LTS“ wurde TYPO3 fast 9 Millionen Mal heruntergeladen, weltweit auf über 500.000 Webseiten eingesetzt und steht in 51 Sprachen zur Verfügung.[6]

2.3 Git

Bei Git handelt es sich um ein verteiltes Open Source Versionskontrollsystem.[7] Das heißt, es erfasst und verwaltet alle Änderungen an den Dateien und versioniert diese. Damit können die Änderungen von früheren Ständen zu jedem Zeitpunkt wiederhergestellt werden. Git ermöglicht durch das Arbeiten in Branches das gleichzeitige Entwickeln verschiedener Personen an unterschiedlichen Ständen.[8] Git unterstützt die Zusammenführung (Merge) der Stände, sodass der Prozess so einfach und sicher wie möglich ist.[9] Außerdem ermöglicht Git dezentrales Versionsverwaltung. Die Projekte werden von Git lokal auf den Rechnern verwaltet. So kann ein Nutzer lokal arbeiten und die Änderungen später auf den Server übertragen. Dies bietet unter anderem den Vorteil, dass fast alle Operationen ohne Netzwerkzugriff erfolgen und sich die Performance erhöht.[10]

---

[1] (Lackes & Siepermann, Open Source, 2018)

[2] (urheberrecht.de, 2021)

[3] (Trabold, Hasenau, & Niederlag, 2008)

[4] (Meyer & Helmich, 2019, S. 10f)

[5] (Schams, 2021)

[6] (Meyer & Helmich, 2019, S. 10f)

[7] (git, kein Datum)

[8] (IONOS SE, 2020)

[9] (Stachmann & Preißel, 2019, S. 28)

[10] (Stachmann & Preißel, 2019, S. 24f)

Git hat einen Marktanteil von 72%. Alternativen dazu sind Subversion mit 23% und Mercurial und CSV mit je 1%.[11]

2.3.1 Fork

Bei Open Source Projekten soll die Hürde zur Mitarbeit so gering wie möglich sein, um eine aktive Community aufzubauen. Trotzdem soll auch nicht jedem Entwickler das Recht zum Pushen bereitgestellt werden, um eine gewisse Qualität zu erhalten. Dafür bietet Git sogenannte Forks als Hilfsmittel. „Ein Fork ist ein serverseitiger Klon, auf dem unabhängig gearbeitet werden kann, ohne das originale Main-Repository zu verändern.“[12]

Bei einem Fork bleibt die Verknüpfung zum Original bestehen, sodass die Änderungen durch einen Pull-Request zurückgeführt werden können.[13] Durch die Eigenschaft, ein eigenständiges Projekt zu sein, bietet ein Fork den Vorteil, dass der Entwickler uneingeschränkte Rechte (insbesondere pushen) hat.[14]

2.4 Upgrade oder Update

Gemäß dem Gabler Wirtschaftslexikon spricht man in der Informatik von einem Upgrade, wenn Hardware oder Software zu einer höheren Leistungsfähigkeit aufgerüstet wird, ohne dass ein neues Modell erworben werden muss.[15]

Ein Update dagegen ist lediglich eine „erweiterte und/oder verbesserte Version eines Softwareprodukts“.[16]

Gemäß Christian Trabold, Jo Hasenau und Peter Niederlag muss bei einem Update unterschieden werden, um was für einen Versionssprung es sich handelt.[17] Die Version von TYPO3 wird mit drei Zahlen angegeben. Dabei stehen die Zahlen für die Major-, die Minor- und die Patch-Version. Während bei einem Wechsel der Patch-Version ausschließlich Fehler behoben werden dürfen, können mit einem Minor-Versionswechsel neue Funktionalitäten hinzukommen, solange die bestehenden Schnittstellen nicht verändert werden. Verändern sich die Schnittstellen muss gemäß Tom Preston-Werner die Major-Version erhöht werden.[18]

Gemäß Sammy Baghdadi handelt es sich im Umgang mit TYPO3 bei einem Update maximal um einen Wechsel der Minor-Version eher aber um einen Wechsel der Patch-Version. Handelt es sich um einen Wechsel der Major-Version spricht man von einem Upgrade.[19]

---

[11] (Compare Repositories - Open Hub, kein Datum)

[12] (Stachmann & Preißel, 2019, S. 195)

[13] (Stachmann & Preißel, 2019, S. 195)

[14] (Stachmann & Preißel, 2019, S. 196)

[15] (Kenning, Lackes, & Siepermann, 2019)

[16] (Lackes & Siepermann, Update, 2018)

[17] (Trabold, Hasenau, & Niederlag, 2008, S. 96)

[18] (Preston-Werner, kein Datum)

[19] (Baghdadi, Konzeption eines Vorgehens für die Durchfuhrung eines Upgrades von TYPO3, 2018)

2.5 PhpStorm

Bei PhpStorm handelt es sich um eine PHP-IDE (integrated development enviroment, englisch für integrierte Entwicklungsumgebung). Eine IDE ist ein Paket von Programmen, welches das Schreiben, Kompilieren, Ausführen und Debuggen von Quellcode in einer Instanz ermöglicht. In einem Code-Editor dagegen können oft mehrere Programmiersprachen geschrieben werden, allerdings geht deren Funktionalität oft nicht über das Schreiben des Codes hinaus.[20]

PhpStorm unterstützt sämtliche PHP-Sprachfunktionen und bietet unter anderem Codevervollständigung, Codeformatierung, Refactoring und Debugging (Fehlerbehebung). Weiterhin sind verschiedene Entwicklertools, wie Versionsverwaltungssysteme, Befehlszeilentools, Docker oder Composer, integriert.[21]

Zu PhpStorm kann unter anderem alternativ auch Netbeans, Aptana Studio oder Eclipse genutzt werden.[22]

Ich verwende PhpStorm, da es durch die Arbeitsumgebung gegeben war und ich bis jetzt keine Funktionen vermisst habe.

2.6 Composer

Composer ist ein Tool, das die Verwaltung von Abhängigkeiten in PHP übernimmt. In einem Projekt kann so definiert werden, von welchen Paketen oder Bibliotheken es abhängig ist. Composer installiert und aktualisiert diese Pakete und Bibliotheken (inklusive ihrer Abhängigkeiten) im Projektverzeichnis.[23]

Auch in TYPO3 können Extensions mit Composer installiert werden. Seit TYPO3 9 ist die Installation per Composer sogar der offiziell empfohlene Weg. Mit dem Befehl „composer create-project typo3/cms-base-distribution <Zielverzeichnis> <Version>" kann ein grundlegendes TYPO3-Projekt aufgesetzt werden.[24] Wurde ein Projekt mit Composer installiert, befindet es sich im sogenannten Composer-Modus. Sollen weitere Extensions hinzugefügt werden, kann dies ausschließlich über Composer erfolgen.[25] Die entsprechende Extension wird mit „composer require <Paketname>" installiert[26] und in der composer.json als Abhängigkeit aufgelistet.[27] Existiert bereits eine composer.json von einem Projekt, kann dieses mit allen Abhängigkeiten (Extensions) mit dem Befehl „composer install" aufgesetzt werden.[28]

---

[20] (Khan, 2021)

[21] (JET BRAINS, 2021)

[22] (Khan, 2021)

[23] (Introduction - Composer, kein Datum)

[24] (Meyer & Helmich, 2019, S. 21)

[25] (Meyer & Helmich, 2019, S. 256)

[26] (Meyer & Helmich, 2019, S. 261f)

[27] (Meyer & Helmich, 2019, S. 22)

[28] (Meyer & Helmich, 2019, S. 436f)

2.7 Extbase oder pibase

Gemäß Robert Meyer und Martin Helmich stehen für die Entwicklung einer TYPO3 Extension zwei Schnittstellen zur Verfügung: Eine pibase-Schnittstelle und eine Extbase-Schnittstelle.[29] Die neuere Extbase-Schnittstelle gibt es seit der Version 4.3. Bei der pibase-Schnittstelle handelt es sich um eine umfangreiche Funktionsbibliothek, die dem Entwickler freie Hand lässt. Im Vergleich dazu hat die Extbase-Schnittstelle klare Vorgaben zur Architektur einer Erweiterung. Das Framework übernimmt den gesamten Kontrollfluss und der Entwickler muss nur noch definierte Einstiegspunkte bedienen. Dadurch entsteht eine Standardisierung von TYPO3-Erweiterungen und somit auch eine bessere Qualität der Erweiterungen.[30]

Das Framework Extbase arbeitet mit dem Model-View-Controller-Prinzip. Das bedeutet, das die Programmierung in Model (Datenmodel), View (Darstellung) und Controller (Steuerung der Anwendung) getrennt wird. Das Model verwaltet die Daten, die dann durch die View dargestellt werden. Die Steuerung der Aktionen erfolgt über den Controller. Durch die Kapselung der einzelnen Komponenten können diese bei Bedarf leichter ausgetauscht werden und die Komplexität verringert sich.[31]

2.8 Domain-driven Design

Bei dem Domain-driven Design handelt es sich um einen Ansatz, Objekte aus dem realen Leben zu modellieren und deren Verhalten und Eigenschaften zu beschreiben. Ein Modell stammt aus einem Fachgebiet, der sogenannten Domäne. Daher leitet sich der Name Domänenmodell (deutsch für domain model) ab. Ein Modell kann aus mehreren Objekten bestehen. Der Ansatz soll eine Grundlage für eine gemeinsame Sprache bilden und dadurch die Kommunikation zwischen Entwicklern, Anwendern und Auftraggebern erleichtern.[32]

2.9 Extensions

Gemäß des Anjali Bhatt ist eine Extension (egl. Erweiterung) ein „Stück Funktionscode, das in die TYPO3-Website integriert wird, um eine bestimmte Aufgabe zu erfüllen“.[33]

Bei der Klassifizierung der Extensions wird zwischen Core- und Dritt-Extensions unterschieden. Die Core-Extensions enthalten alle Funktionen, die zum TYPO3-Core gehören. Diese werden vom TYPO3-Core-Team (weiter-) entwickelt. Dritt-Extensions können bestehende Funktionen, Objekte oder Konfigurationen überschreiben oder wenn benötigt durch neue erweitern.[34] Dabei wird sowohl Quellcode aus dem TYPO3-Kern als auch aus anderen Dritt-Extensions überschrieben.

Die Dritt-Extensions können im öffentlichen Extension-Verzeichnis vom TYPO3, dem sogenannten TYPO3 Extension Repository (kurz TER), bereitgestellt werden.[35]

---

[29] (Meyer & Helmich, 2019, S. 307)

[30] (Meyer & Helmich, 2019, S. 307f)

[31] (Meyer & Helmich, 2019, S. 309f)

[32] (Meyer & Helmich, 2019, S. 309)

[33] (Bhatt, 2020)

[34] (Bhatt, 2020)

[35] https://extensions.typo3.org/

Seit der Einführung von Extbase haben TYPO3-Extensions eine standardisierte Struktur. Alle Extensions enthalten die Ordner „Classes“, „Configuration“ und „Resources“ sowie die Konfigurationsdatei für die Extension (ext_emconf.php). Im Ordner „Classes“ sind alle PHP-Klassen einer Erweiterung enthalten. Die Klassen sind nach ihrem Typ und daraus resultierenden Funktionen sortiert. Der Dateiname einer Klasse leitet sich direkt vom Klassennamen ab. Alle Dateien, die Konfigurationen enthalten (wie zum Beispiel Datenbankkonfigurationen) liegen im Ordner „Configuration“. Der Ordner „Resources“ enthält unter anderem Templates, Icons und Styling-Dateien.[36]

2.9.1 Domain Model und Domain Repository

Eine Klasse des Typen Model bildet ein Domänenobjekt ab. Das Model enthält alle Eigenschaften des Objekts und stellt get- und set-Methoden bereit, um auf die Variablen zuzugreifen.

Zu jedem Model gibt es ein zugehöriges Repository. Das Repository stellt alle CRUD-Operationen bereit, die benötigt werden, um ein Objekt in der Datenbank anzusprechen. Bei den CRUD-Operationen handelt es sich um create (ein Objekt erzeugen), read (auf das Objekt zugreifen), update (alle Änderungen, die an einem Objekt vorgenommen werden) und delete (ein Objekt löschen).

2.9.2 Hooks

Um bestehende Funktionen zu erweitern, gibt es verschiedene Möglichkeiten. Eine davon sind sogenannte Hooks. Hooks sind definierte Stellen im Quellcode, in die man sich mit seinem eigenen Code einhängen kann, um Einfluss auf die Verarbeitung der Daten zu nehmen. Der eigene Code wird im Ordner Classes/Hooks abgelegt.[37]

In der Datei ext_localconf.php wird die entsprechende Funktion für den Hook definiert. Wird ein Hook ausgeführt, werden alle definierten Funktionen auch ausgeführt.

Dadurch, dass die überschreibenden Funktionen direkt durch die Ursprungsfunktion aufgerufen werden, ist die Verwendung von Hooks außerhalb der definierten Stellen nicht möglich. In solchen Fällen kann die Ursprungsklasse mit XCLASSes erweitert werden.[38]

2.9.3 XCLASSes (Extending Classes)

Im Vergleich zu Hooks kann fast jede Klasse mit dem Prinzip „XCLASS“ beliebig erweitert werden. Dabei ist es unwichtig, ob die zu erweiternde Klasse aus dem Core oder einer anderen Extension stammt.[39]

```
$GLOBALS['TYPO3_CONF_VARS']['SYS']['Objects'][\NITSAN\NsExtCompatibility\Controller\nsextcompatibilityController::class] = [
    'className' => \CPSIT\UpgradeAnalysis\Controller\UpgradeAnalysisController::class
];
```

Abbildung 1 - Registrierung einer Erweiterung
Screenshot aus PhpStorm ext_localconf.php

Die erweiternde Klasse entspricht dem Typ der erweiterten Klasse und wird dementsprechend in einem Unterordner von „Classes“ abgelegt.

---

[36] (Meyer & Helmich, 2019, S. 334ff)

[37] (Trabold, Hasenau, & Niederlag, 2008, S. 772)

[38] (Trabold, Hasenau, & Niederlag, 2008, S. 772)

[39] (Trabold, Hasenau, & Niederlag, 2008, S. 772)

Ein Nachteil der Erweiterung ist, dass die Erweiterungen in der Ladereinfolge der Extensions greifen. Es kann nicht sichergestellt werden, in welcher Reihenfolge die Extensions in fremden Systemen geladen werden. Dadurch besteht die Gefahr, dass die eigene Extension durch andere Extensions überschrieben wird und nicht mehr funktionsfähig ist. Aufgrund dieser Erkenntnisse sollte bei geplanter Veröffentlichung der Extension auf XCLASSes verzichtet werden.[40]

Ein weiterer Nachteil gegenüber Hooks ergibt sich daraus, dass die Extensions nicht an definierten Stellen erweitert werden: Ändert sich die Syntax der ursprünglichen Extension an der Stelle, an der diese überschrieben wurde, kann dies zum Verlust der Funktionsfähigkeit der überschreibenden Extension führen. Deswegen muss nach jeder Versionsänderung der ursprünglichen Extension die eigene Extension geprüft und gegebenenfalls angepasst werden.[41]

XCLASSes bieten also den Vorteil, dass sie fast beliebig verwendet werden können, Hooks stellen aber vermutlich eine nachhaltigere Lösung da.

2.10 Ns_ext_compatibility

Die Extension ns_ext_compatibility unterstützt bei der Prüfung der Machbarkeit und Komplexität für ein Upgrade auf eine höhere TYPO3-Version. „ns_ext_compatibility“ stellt ein Backendmodul zur Verfügung, das verschiedene Informationen über das bestehende System darstellt.

Unter anderem wird über grundlegende Systeminformationen wie die aktuellen TYPO3-, PHP- und MySQL-Versionen informiert. Weiterhin gibt das Backendmodul Auskunft über die Anzahl an in der Website enthaltenen Seiten und Domains. Ein wichtiger Bestandteil ist die Auflistung aller im System installierten Extensions. Da das Anpassen der Extension an die Ziel-TYPO3-Version den größten Teil der Zeit eines Upgrades in Anspruch nimmt, ist es sinnvoll, so viele Informationen wie möglich über die enthaltenden Extensions zu haben. Die Extension ns_ext_compatibility analysiert die Konfigurationsdatei der zu prüfenden Extensions und liefert daraus Informationen, wie den Namen, die eindeutige Kennung (Extension-Key) oder die Beschreibung. Zusätzlich werden die Abhängigkeiten und die vorhandenen Versionen der jeweiligen Extensions dargestellt. So beschreibt es Sanjay Chauhan.[42]

ns_ext_compatibility bildet eine Grundlage für die im Rahmen dieser Arbeit entwickelten Extension, da sie zum einen benötigte Daten generiert und zum anderen umfassende Templates zur Visualisierung bereitstellt.

2.11 Upgrade-Analyse bei familie redlich

„familie redlich steht für integrierte Kommunikation durch das effiziente Wirken von Spezialisten aller Disziplinen.“[43] Dazu gehört neben Event-Management, Produktentwicklung und Marketingberatung auch das Erstellen von Onlineauftritten. Dies geschieht hauptsächlich durch den Einsatz von TYPO3. Im Jahr 2015 hat sich familie redlich mit CPS IT zusammengeschlossen. CPS IT betreut, gemäß eigener Aussage, anspruchsvolle Websites mit dem CMS TYPO3 und engagieren sich in der Open Source Community. Gegenwärtig beschäftigt familie redlich ungefähr 240 Mitarbeiter. Davon arbeiten 65

---

[40] (TYPO3 Documentation Team, 2021)

[41] (TYPO3 Documentation Team, 2021)

[42] (Chauhan, 2018)

[43] (familie redlich, 2018)

Personen im Bereich :digital. familie redlich entwickelt vielfältige Extensions für eigene Projekte und darüber hinaus projektunabhängige Extensions für die Community. Seit Oktober 2018 arbeite ich als Werkstudentin im Bereich Backendentwicklung bei familie redlich :digital.

Den Prozess der Upgrade-Analyse bei familie redlich zu formalisieren, wurde von Sebastian Kreideweiß begonnen. Julian Fastnacht hat diese Arbeit fortgesetzt, indem er die Daten zur Unterfütterung der Analyse erweiterte. Sammy Baghdadi untersuchte den Prozess in seiner Bachelor Arbeit mit dem Thema „Konzeption eines Vorgehens für die Durchführung eines Upgrades von TYPO3“. Auch die Erkenntnisse aus dieser Arbeit flossen in den aktuellen Ablauf einer TYPO3-Upgrade-Analyse ein. Weiterhin hat Julian Fastnacht Feedback von Kollegen in den Prozess eingegliedert.[44]

Aktuell sieht die Vorlage für die Durchführung einer TYPO3-Upgrade-Analyse bei familie redlich vor, dass zunächst Informationen über das bestehende System gesammelt werden: Es wird geprüft, wie der Zugriff auf das System abgedeckt ist. Dazu gehören unter anderem, die Zugänge zum Live-System, der vollständige Datenbankauszug und der vollständig Dateiauszug. Weiterhin werden Basiskonfigurationen wie die TYPO3-, PHP- oder MySQL-Version (siehe Abbildung 2) erhoben.

Einrichtungstatus der Basiskonfiguration

TYPO3 Version	x.x.xx
Datenbank	
Bildgenerierung	IM/GD
Schreibrechte der Verzeichnisse	
Mail-Anbindung	
Produktionsmodus	
UTF-8	
OS	
Web-Server	
PHP Version	x.x.x
MySQL	x.x.x

Abbildung 2 - Aufnahme der Systemkonfiguration (Baghdadi & Fastnacht, Ist-Analyse & Aufwandsschätzung, 2020)

[44] Interview mit Julian Fastnacht (Interview 3)

Außerdem werden Kennzahlen (Benutzergruppen, Menüs, Content-Elemente, etc.) festgehalten, um einen besseren Überblick über den Umfang des Systems zu erhalten. Ein Beispiel zeigt die Abbildung 3.

Ergebnisse der Benutzerprüfung

	Backend-Benutzer	davon Admin
Anzahl		
Aktiv		
Inaktiv und/oder Entfernt		

	Backend-Benutzergruppen
Anzahl	
Aktiv	
Versteckt und/oder entfernt	

Abbildung 3 - Beispiel für Kennzahlen
(Baghdadi & Fastnacht, Ist-Analyse & Aufwandsschätzung, 2020)

Ein wesentlicher Bestandteil der Aufwandschätzung ist die Analyse der Extensions. Bei den TYPO3-Kern-Extensions wird davon ausgegangen, dass sie mit geringem Zeitaufwand aktualisiert werden können. Hierbei genügt eine einfache Aktualisierung des TYPO3-Kerns. Alle übrigen Extensions werden nach vorgegebenen Kriterien kategorisiert (siehe Abbildung 4).

Art	TYPO3-Scanner	PHP-Scanner	LOC %	Beschreibung
UPDATEN	Keine Auswirkung	Keine Auswirkung	0	Es steht eine 9.x.-Extension zur Verfügung, die eingespielt werden kann.
ENTFERNEN	Keine Auswirkung	Keine Auswirkung	0	Extension kann vom Server deinstalliert und gelöscht werden. Ggf. keine Anpassung.
ANPASSEN	Strong & Breaking = 0 Strong & Deprecated = 0 Weak & Breaking = 0 Weak & Deprecated > 0	Warnings > 0 Errors = 0	> 10%	Extension muss angepasst werden; Aufwand gering.
ERWEITERN	Strong & Breaking = 0 Strong & Deprecated > 0 Weak & Breaking = 0 Weak & Deprecated > 0	Warnings > 0 Errors = 0	> 20%	Extension muss erweitert werden; Aufwand mittel.
ANPASSEN	Strong & Breaking > 0 Strong & Deprecated > 0 Weak & Breaking > 0 Weak & Deprecated > 0	Warnings > 0 Errors > 0	> 50%	Extension muss erweitert werden; Aufwand Hoch bis nicht einschätzbar
TP	Strong & Breaking > 0 Strong & Deprecated > 0 Weak & Breaking > 0 Weak & Deprecated > 0	Warnings > 0 Errors > 0	> 70%	Auslagerung in ein eigenes Teilprojekt möglich

Abbildung 4 - Kriterien zur Gruppierung der Extensions
(Baghdadi & Fastnacht, Ist-Analyse & Aufwandsschätzung, 2020)

Jeder Kategorie ist eine Zeitschätzung zugeordnet (siehe Abbildung 5). Für die Extensions werden Informationen wie Bezeichnung, Beschreibung und eindeutige Kennung aufgenommen. Diese Informationen dienen hauptsächlich dem Kunden zur besseren Übersicht. Für die Bestimmung der Kategorien werden die Extensions mit den Open Source Tools „TYPO3 scanner“[45] und „phpcs“[46] gescannt. Zusätzlich wird die Anzahl an Codezeilen (Lines of Code) bestimmt. Diese Vorgehensweise erweckt den Eindruck, dass sie keine speziellen TYPO3-Kenntnisse erfordert.

Kategorie	Anzahl	Multiplikator / Faktor	Ungefährer Aufwand pro Ext.	Maßnahmen
0		Kein Multiplikator	~ 0,25 h	System-Extensions Ersatzlos ENTFERNEN oder UPDATEN ohne Anpassung
1		Anzahl über 5 = Aufwand * 2	~ 2 h	Ersatzlos ENTFERNEN oder UPDATEN ohne Anpassung
2		Anzahl über 2 = Aufwand * 2 Wenn Tech-Doku vorhanden * 1,5	~ 4 h	Extension ANPASSEN oder
3		Anzahl über 2 = Aufwand * 2 Wenn Tech-Doku vorhanden * 1,5	~ 1 AT = 8 H	Extension ERWEITERN oder
4		Anzahl über 2 = Aufwand * 2 Wenn Tech-Doku vorhanden * 1,5	~ 2,5 AT = 20 H	Extension ANPASSEN und ERWEITERN
5		Anzahl über 3 = Aufwand * 2 Wenn Tech-Doku vorhanden * 1,5	~ 8 AT = 64 H	Mit höherem Aufwand ANPASSEN bis hin zur Neuerstellung Durchschnittsannahme

Abbildung 5 - Kategorien mit zugehöriger Zeitschätzung (Baghdadi & Fastnacht, Ist-Analyse & Aufwandsschätzung, 2020)

Trotzdem wird bei familie redlich die Kategorisierung der Extensions nicht ohne das Hinzuziehen eines erfahrenen TYPO3-Entwicklers durchgeführt. Um Gründe dafür zu finden, habe ich die Vorgehensweise und die Tabellen auf mögliche Fehlerquellen analysiert.

[45] https://github.com/Tuurlijk/typo3scan

[46] https://github.com/FriendsOfPHP/PHP-CS-Fixer

Zum Beispiel beruhen die konstanten Faktoren in der Zeitschätzung auf Erfahrungen und können so zur Fehlerquelle werden. Eine weitere Fehlerquelle können die Kriterien zur Einteilung in die Kategorien sein. Bei einer als strong und breaking markierten Funktion im TYPO3-Scanner wird die Extension in die Kategorie „Anpassen“ (5) einsortiert. Dies würde einen kalkulierten Zeitaufwand von 64 Stunden bedeuten. Oft können diese Funktionen aber aufgrund vorhandener TYPO3-Kern-Dokumentation in kürzerer Zeit abgelöst werden. Dies Einschätzung erfordert die Erfahrung eines TYPO3-Entwicklers. Außerdem werden Faktoren wie Zeit zum Testen oder zur Übergabe an den Kunden nicht berücksichtigt.

Für die Kalkulation der gesamt benötigten Zeit, wird für jede Extension die Zeitschätzungen („Ungefährer Aufwand pro Ext.“) mit dem Faktor („Multiplikator/Faktor“) aus Abbildung 5 multipliziert. Dabei hängt der Faktor davon ab, ob für die jeweilige Extension eine Tec-Doku vorhanden ist. Der Gesamtzeitaufwand ergibt sich aus der Summe der einzelnen Werte.

3 Forschungsmethoden

Im Verlauf der Arbeit werde ich Interviews führen, um Expertenwissen zum Vorgehen der Upgrade-Analyse festzuhalten und dieses in einer Funktion zu automatisieren. Weiterhin wird in einzelnen Fällen die statische Codeanalyse zum Einsatz kommen, um Funktionalitäten der Extension ns_ext_compatibility zu bewerten.

3.1 Statische Code-Analyse

Bei der statischen Code-Analyse werden Artefakte des Codes untersucht, ohne dass sie zur Ausführung kommen. Artefakte können hierbei Architektur- oder Entwurfsmodelle oder der Quellcode selbst sein. Voraussetzung ist, dass der Code statisch vorliegt (White-Box-Test). Der Code dient als Basis um Kenntnisse über tatsächliche Strukturen und Eigenschaften der Software zu gewinnen.[47]

Die statische Code-Analyse kann manuell oder durch Programme erfolgen.

In Kontext der vorliegenden Arbeit liefert die statische Code-Analyse Einblick in einzelne Funktionen der Extension ns_ext_compatibility, um zu entscheiden, ob ich diese Funktionen verwenden kann und ob diese im Zweifelsfall angepasst werden müssen.

3.2 Leitfadengestütztes Experteninterview

Nach Gesprächen mit erfahrenen TYPO3-Entwicklern stellte sich heraus, dass es für den beschriebenen Prozess keine allgemein gültige Lösung gibt. Sammy Baghdadi berichtete, dass nach Veröffentlichung seiner Bachelor-Arbeit viele Kommentare aus der TYPO3-Community kamen, wie der Prozess der Analyse am besten durchgeführt werden könnte.

Folgende Ergebnisse meiner Online-Recherche verstärken diesen Bericht. Zu dem übergeordneten Thema „Upgrade eines TYPO3-Systems“ existieren Anleitungen zur Durchführung. Hier wird oft benannt, dass es sinnvoll ist den derzeitigen Zustand des Systems zu erfassen und sich insbesondere einen Überblick über die installierten Extensions zu verschaffen. Allerdings wird nicht definiert, wie dieser Überblick aufgebaut sein soll oder wie daraus ermittelt werden kann, was das für den zeitlichen Aufwand bedeutet. Lediglich in einem Blockeintrag von Sanjay Chauhan auf NITSAN wird darauf hingewiesen, dass es sinnvoll sein kann, den Quellcode der Extensions gegen die Changelog der aktuellen Version zu prüfen, um damit eine Aussage über den Umfang der Anpassungen an den jeweiligen Extensions treffen zu können.

Das Vorgehen zur Durchführung einer Upgrade-Analyse scheint also je nach Person und Rolle (im TYPO3-Kontext) zu variieren. Da es sich also um spezielles Domänenwissen handelt, bietet es sich an, Experten zu befragen. Dafür wird zur Strukturierung der Interviews ein Leitfaden ausgearbeitet.[48] Der Leitfaden dient der besseren Vergleichbarkeit der Interviews.[49]

---

[47] (Bommer, Spindler, & Barr, 2008, S. 215)

[48] (Gläser & Laudel, 2010, S. 11)

[49] (Meuser & Nagel, 2013, S. 466)

3.2.1 Vorbereitung

Der Interviewleitfaden gliedert sich in drei Bereiche.[50] Zu Beginn stelle ich Fragen über die Rolle des Interviewpartners, um die Qualität der Antworten einordnen zu können. Im Hauptteil stelle ich Fragen, die den Prozess der Analyse aufnehmen und Kriterien zur Zeitschätzung festhalten. Ich beende das Interview mit einem Ausblick und Wünschen des Interviewpartners zu einer Visualisierung.

Ich habe die Fragen ausformuliert, um sicherzustellen, dass allen Befragten dieselben Fragen gestellt werden.[51]

Schon zu Beginn der Interview-Phase gab es zwei Leitfäden. Dies resultierte daraus, dass ich einen meiner Interviewpartner zusätzlich zu den methodischen noch technische Fragen stellen wollte. Vladimir Falcon Piva hat bereits einen Ansatz, wie die Tools „TYPO3 scanner“ und „phpcs“ in eine Extension eingebunden werden können. Ziel ist es herauszufinden, ob seine Extension für meine Zwecke verwendet werden kann. Ist dies nicht der Fall, soll erfragt werden, welche technischen Bestandteile für die Einbindung dieser Tools notwendig sind.

Während der ersten beiden Interviews betonten meine Interviewpartner, dass sie sich nicht vorstellen können, dass die Analyse komplett ohne eine menschliche Komponente auskommt. Die Aussage fand ich interessant und wollte wissen, wie meine anderen Interviewpartner diesen Aspekt sehen. Deswegen habe ich unter Frage 11 die Nachfrage, „denkst du, dass man die menschliche Komponente zusätzlich braucht?“ eingefügt.

Insgesamt wurde für das Interview ein Zeitrahmen von einer Stunde eingeplant. Alle Interviews wurden im Video-chat mit der Software „Jitsi“ geführt.

Gemäß Jochen Gläser und Grit Laudel schränkt paralleles Mitschreiben der Antworten den Interviewenden darin ein, während des Interviews zu analysieren ob die Fragen vollständig beantwortet wurden und gegebenenfalls Nachfragen zu stellen.[52] Deswegen habe ich während des Interviews nur stichpunktartig mitgeschrieben und zusätzlich die Interviews mit dem Crome-Browser-Plugin „Nimbus Capture“ aufgezeichnet. Die Aufzeichnungen[53] habe ich bei der Auswertung herangezogen, wenn die Stichpunkte nicht aussagekräftig genug waren.

Bei der Auswahl der Experten stellte sich zunächst die Frage, was einen Experten auszeichnet. Gemäß Jochen Gläser und Grit Laudel ist ein Experte eine Person, die Wissen über die eigenen Arbeitsprozesse hat.[54] Robert Kaiser stimmt dem zu, grenzt aber den Experten vom Spezialisten ab. Gemäß Kaiser zeichnet den Experten aus, dass er nicht nur über Wissen oder Kompetenzen verfügt, sondern dass er darüber hinaus die Autonomie besitzt seine Problemlösungen zu bewerten. Weiter spezialisiert Kaiser Experten als „Träger des für die wissenschaftliche Analyse relevanten Funktionswissen“.[55] In der vorliegenden Analyse des Vorgehens der Upgrade-Analyse bezieht sich das relevante Funktionswissen auf Kenntnisse von TYPO3 und mehrfache Erfahrung mit dem Prozess der Upgrade-Analyse. Entsprechend dem oben genannten Autonomiekriterium habe ich solche Personen ausgewählt, die die Zeiteinschätzung selbstständig vorgenommen haben und anschließend die Differenz zwischen geschätzten und realen Kosten verantwortet haben. Um möglichst verschiedene Sichtweisen auf das Thema zu erhalten, bot es sich an, Entwickler,

---

[50] (Gläser & Laudel, 2010, S. 147ff)

[51] (Gläser & Laudel, 2010, S. 144)

[52] (Gläser & Laudel, 2010, S. 157f)

[53] https://git.imp.fu-berlin.de/wundermaj98/bachelor-arbeit/-/commit/29b4b2bc10e41a9784ed024cef844f658217601d

[54] (Gläser & Laudel, 2010, S. 11f)

[55] (Kaiser, 2014)

Integratoren und einen Requirements Engineer zu befragen. Nach einem Gespräch mit den betreuenden Personen und weiteren Recherchen stellte sich heraus, dass drei bis fünf Interviewpartner eine angemessene Zahl sein könnte.

Person	Erfahrung mit TYPO3	Funktion	Durchgeführte Upgrade-Analysen
Erik Rauchstein	8 Jahre	Backend-Entwickler und Integrator	6
Dirk Wenzel	13 Jahre	Backend-Entwicklung in der Vergangenheit auch Integration und Frontend	Etwas über 10
Julian Fastnacht	5 Jahre	Requirement Engineer	5-10
Anna Schneider	17 Jahre	Backend-Entwicklerin und Integratorin	10-15
Vladimir Falcon Piva	14 Jahre	Backend-Entwickler, Integrator, Frontend-Entwickler	100 Upgrades (viele davon waren Relaunches, für diese wurden keine klassischen Upgrade-Analysen durchgeführt)

Tabelle 1 - Experten

Die Tabelle zeigt, dass alle Befragten einen langjährigen Kontakt mit TYPO3 haben, verschiedene Funktionen ausführen und schon mehrere Upgrade-Analysen durchgeführt haben.

3.2.2 Auswertung

Die Interviews werden mit der qualitativen Inhaltsanalyse ausgewertet. Dabei handelt es sich um „ein Verfahren, das sich frühzeitig vom Ursprungstext trennt, versucht die Informationsfülle zu reduzieren und entsprechend dem Untersuchungsziel zu strukturieren“[56] Die qualitative Inhaltsanalyse behandelt das auszuwertende Interview (das Protokoll liegt in Textform vor) als Material, in dem die Daten enthalten sind. Die Daten werden dem Protokoll entnommen, indem die Rohdaten extrahiert, aufbereitet und ausgewertet werden.[57] Die Extraktion der Informationen erfolgt mithilfe eines Suchrasters, das auf Grundlage der theoretischen Vorüberlegungen konstruiert wurde.[58] Anschließend werden die Informationen aufbereitet, indem diese zusammengefasst und auf Widersprüche und Redundanzen überprüft werden.[59] Dafür werden die Informationen aus allen Interviews thematisch sortiert in einer Tabelle zusammengetragen.[60] Sowohl die Extraktion als auch die Aufbereitung unterliegen dem Interpretationsprozess.[61] Die folgende Abbildung stellt den Prozess noch einmal dar.

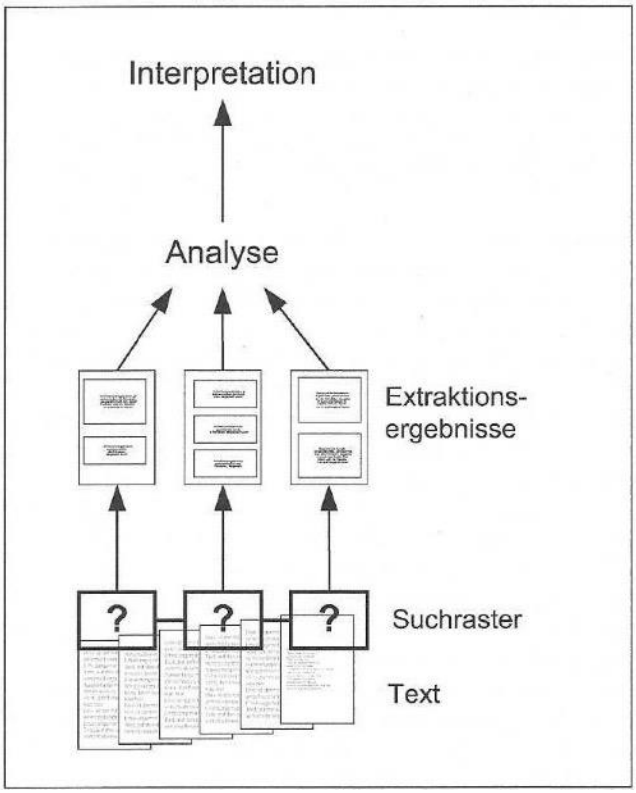

Abbildung 6 - Prinzip der qualitativen Inhaltsanalyse (Gläser & Laudel, 2010, S. 200)

[56] (Gläser & Laudel, 2010, S. 200)

[57] (Gläser & Laudel, 2010, S. 199)

[58] (Gläser & Laudel, 2010, S. 200)

[59] (Gläser & Laudel, 2010, S. 202)

[60] (Meuser & Nagel, 2013, S. 467)

[61] (Gläser & Laudel, 2010, S. 201f)

3.2.3 Abgrenzung qualitativer gegenüber quantitativen Forschungsmethoden

Das leitfadengestützte Experteninterview gehört zu den qualitativen Forschungsmethoden. Letztere werden vor allem in noch wenig erforschten Gebieten angewendet. Hierbei sollen Informationen generiert werden, um einen Bereich zu verstehen.[62] Ziel ist es, die Kausalmechanismen zu finden, die einen bestimmten Effekt erzielen. Dabei können Bedingungen identifiziert werden, eine Aussage über ihre Verbreitung ist jedoch nicht möglich.[63] Bei den qualitativen Forschungsmethoden handelt es sich um eine Interpretation sozialer Sachverhalte in verbaler Beschreibung. Die Komplexität der Sachverhalte wird erst bei der Auswertung schrittweise reduziert.[64]

Im Gegensatz dazu stehen die quantitativen Forschungsmethoden. Hierbei werden standardisiert mit statischen Tests Daten erhoben. Die quantitativen Forschungsmethoden identifizieren Zusammenhänge zwischen sozialen Phänomenen und den dazugehörigen Bereichen. Im Vergleich zu den qualitativen Forschungsmethoden ist klar definiert, in welchem Bereich bestimmte Phänomene gelten. Merkmale und ihre Häufigkeiten sind quantifizierbar. Allerdings werden nur Aussagen über die Korrelation von Ereignissen getroffen, nicht jedoch über die Kausalität; Ursache und Wirkungen sind nicht identifizierbar.[65]

Sowohl die qualitativen als auch die quantitativen Forschungsmethoden gehören zu der empirischen Sozialforschung. Hierbei wird ein bestimmter Abschnitt der sozialen Welt beobachtet. Die Beobachtung wiederum trägt zur Weiterentwicklung einer Theorie bei.[66] In vielen Fällen können die qualitativen und quantitativen Forschungsmethoden nicht sauber getrennt werden und finden oft sogar ergänzend zueinander Anwendung.[67]

---

[62] (Springer, Koschel, Fahr, & Pürer, 2015)

[63] (Meuser & Nagel, 2013, S. 26)

[64] (Meuser & Nagel, 2013, S. 27)

[65] (Meuser & Nagel, 2013, S. 26f)

[66] (Meuser & Nagel, 2013, S. 24)

[67] (Meuser & Nagel, 2013, S. 25)

4 Entwicklung der Extension

Neue Funktionalitäten werden, wie in den Grundlagen beschrieben, mittels Extensions in ein TYPO3-System integriert. Bei der Entwicklung stellt sich die Frage, gegen welches Framework die Extension entwickelt werden soll. Ich habe meine Extension aufgrund der oben genannten Gründe mit Extbase entwickelt. Die Extension trägt den Namen „typo3_upgrade_analysis“ und ist open source auf GitHub zu finden.[68]

Die Extension typo3_upgrade_analysis (im Folgenden TUA) erweitert die Extension ns_ext_compatibility (im Folgenden NEC). Das bedeutet, dass in der Konfigurationsdatei (composer.json) der Extension TUA, NEC als Abhängigkeit definiert wird (siehe Abbildung 7) und alle benötigten Klassen wie oben beschrieben als XCLASSes erweitert werden.

```
],
"require": {
  "typo3/cms-core": "^7.6",
  "nitsan/ns_ext_compatibility": "^5.0",
  "symfony/filesystem": "^5.3",
  "phpcompatibility/php-compatibility": "^9.3",
  "squizlabs/php_codesniffer": "*",
  "michielroos/typo3scan": "^1.7"
},
```

Abbildung 7 - Abhängigkeiten der Extension typo3_upgrade_analysis
Screenshot aus PhpStorm composer.json

Als nächstes stellte sich die Frage für welche TYPO3-Versionen die Extension TUA kompatibel sein soll. Zum Zeitpunkt der Entwicklung ist 7 die älteste noch unterstützte TYPO3-Version (siehe Abbildung 8).[69] Deswegen habe ich TUA initial für TYPO3 7 entwickelt. Zukünftig soll die Extension bis zur aktuellen TYPO3-Version kompatibel sein. Diese Abdeckung zu gewährleisten, war im zeitlichen Rahmen dieser Arbeit nicht möglich.

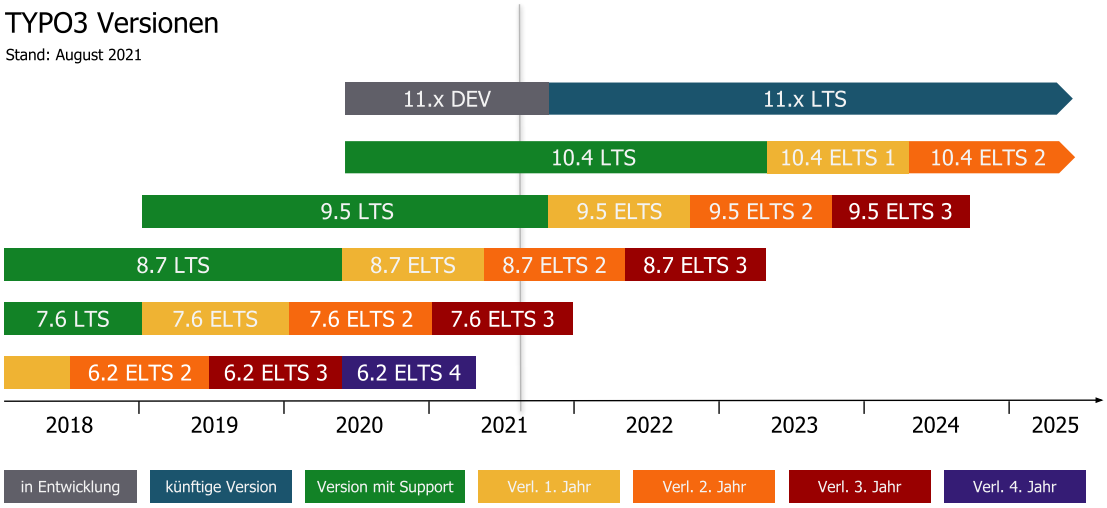

Abbildung 8 - TYPO3 Versionen 6.2 bis 11.x Stand 2021 (jweiland.net, 2021)

[68] https://github.com/CPS-IT/typo3_upgrade_analysis/commit/d5d7cb278f7e96f0b81c7800307e00b0797e31bf

[69] (jweiland.net, 2021)

4.1 Anforderungen an das Backendmodul

Im Interview habe ich nach den Anforderungen an eine Backendmodul gefragt, welches den Interviewpartnern die Durchführung der Upgrade-Analyse erleichtert oder abnimmt. Dabei wurden verschiedene Aspekte deutlich.

Drei Interviewpartner forderten, dass das Backendmodul Informationen über das aktuelle System abbildet. Dabei wurde auf die Informationen verwiesen, die bei dem Verfahren von familie redlich (siehe Kapitel 2.11) erhoben werden. Es sollen also die oben genannten Serverkonfigurationen und Kennzahlen aufgelistet werden. Erik Rauchstein merkte an, dass über das bekannte Vorgehen hinaus folgende Informationen hilfreich sein könnten: Memory-Limit, Auflistung der installierten PHP-Erweiterungen und der PHP-Libraries. Welche dieser Informationen (oder darüber hinaus) sinnvoll sind, müsste nach seiner Aussage noch einmal im Detail geprüft werden. Zusätzlich empfahl er, auch Kennzahlen auf Codeebene festzuhalten. Er nannte als Beispiele die Auflistung der Klassen, welche Core-Klassen erweitern, und der verwendeten Hooks. Das letztere Beispiel wurde von Dirk Wenzel ebenfalls als wünschenswert genannt. Vladimir Falcon Piva traf keine Aussage dazu, ob und welche Informationen über das System für ihn hilfreich wären. Auf Nachfrage sagt er aber, dass Informationen immer gut seien. (Je mehr desto besser.)

Alle Interviewpartner waren sich einig, dass eine Auflistung der installierten Extensions mit den zugehörigen Informationen wichtig sei. Bei der Feststellung, welche Informationen dies seien, gingen die Meinungen zum Teil wieder auseinander. Auch für die Extensions wurden zuerst die Abbildung der Informationen aus dem bisherigen Vorgehen (siehe Kapitel 2.11) gefordert. Dirk Wenzel ergänzte, dass die Abhängigkeiten der jeweiligen Extensions für die Auswertung ebenfalls sinnvoll sein. Ausgehend von der Annahme, dass eine automatisierte Schätzung der Upgrade-Dauer vorhanden ist, schlug Erik Rauchstein vor, diese auch auszugeben. In dem Fall würde es sich seiner Meinung nach anbieten, die Extensions nach den Kategorien (siehe Kapitel 2.11) zu sortieren. Fast alle der befragten Personen wünschten sich eine Anzeige der Ergebnisse der Scanning-Tools, um eine Überprüfung der Zeitschätzungen zu vereinfachen. Zusätzlich wurde von drei Interviewpartnern angemerkt, dass die Suche nach einer für die jeweilige Extension kompatible TYPO3-Zielversion viel Zeit in Anspruch nimmt. Daraus ergab sich, das Ergebnis dieser Prüfung ebenfalls auszugeben.

Dirk Wenzel verfolgt bei der Upgrade-Analyse einen anderen Ansatz. Er erwartet vom Backendmodul Auskunft über die Codequalität der jeweiligen Extensions. Ebenso sind für ihn das Wissen über die Testabdeckung und die Vollständigkeit der Dokumentation der jeweiligen Extensions Faktoren, die in die Zeitschätzungen mit einfließen.

Erik Rauchstein kann sich nicht vorstellen, dass die Schätzung des Aufwands am Ende der Upgrade-Analyse ohne eine menschliche Prüfung auskommt. Er schlug vor, dass die Extensions bei Erfüllung folgenden Kriterien markiert werden sollten: Falls der Versionssprung einer Extension größer als 1 ist (falls schon eine mit der TYPO3-Zielversion kompatible Version existiert); falls im Ergebnis des TYPO3-Scanners mehr als 20% der Veränderungen als „breaking“ markiert wurden; falls die Extension generell in eine hohe Upgrade-Kategorie einsortiert wurde. Bei den so markierten Extensions empfiehlt er die zusätzliche Prüfung durch einen Entwickler. Auf Nachfrage bestätigen die anderen Interviewpartner, dass es sinnvoll ist, die Analyse zumindest von einem Entwickler prüfen zu lassen.

Laut Erik Rauchstein wäre es gut, wenn Entwickler eigene Zeitschätzungen im Backendmodul hinterlegen könnten. Besonders in den Fällen, in denen Extensions als zu kontrollieren gekennzeichnet wurden, kann dies sinnvoll sein.

Zusätzlich wurde von zwei Interviewpartnern gewünscht, dass man die gesammelten Informationen exportieren kann, um sie zum Beispiel Kunden bereitzustellen. Dabei soll der Export als CSV-Datei oder PDF erfolgen.

Weiterhin wurde eine Volltextsuche als potenzielle Anforderung genannt. Eine Anwendungsmöglichkeit wäre das Auffinden mehrfach enthaltene Funktionen, um gegebenenfalls den kalkulierten Aufwand des Upgrades zu korrigieren.

Für die Referenzierung in den weiteren Kapiteln listet die nachfolgende Tabelle die genannten Anforderungen noch einmal auf.

Kennung	Anforderung
a1	Auflistung der Systeminformationen
a11	TYPO3-, PHP-, MySQL-Version
a12	Betriebssystem, Web-Server, Datenbank
a13	Weitere PHP-Informationen
a14	Kennzahlen
a15	Core-Klassen, die erweitert wurden
a16	Verwendete Hooks
a2	Extension-Übersicht erstellen
a21	Allgemeine Informationen
a22	Kompatible Version der Extensions (falls vorhanden)
a23	Zeitschätzung bzw. Kategorie (durch die Upgrade-Analyse bestimmt)
a24	Ergebnisse der Scanning-Tools
a25	Sortierung der Extensions nach Kategorien
a26	Feld für Entwicklerschätzung
a27	Codequalität
a28	Testabdeckung
a3	Extensions markieren für zusätzliche Entwicklerprüfung
a4	Suche nach Funktionen
a5	Report exportieren

Tabelle 2 - Anforderungskatalog Backendmodul

4.2 Systeminformationen und Kennzahlen abfragen

Im zeitlichen Rahmen dieser Arbeit konnte nur ein Teil der genannten Anforderungen umgesetzt werden. Folgende Schwerpunkte habe ich gesetzt. Bei der Aufnahme der Systeminformationen habe ich mich dafür entschieden zuerst die Anforderungen a11, a12 (Serverinformationen) und a14 (Kennzahlen) umzusetzen. Diese Entscheidung begründet sich darin, dass die genannten Informationen den Systeminformationen entsprechen, die aktuell im Referenzmodell erhoben werden. Demnach wurden die Anforderungen a13, a15 und a16 nicht umgesetzt.

In einem TYPO3-Backend hat man die Möglichkeit, sich die Serverinformationen anzeigen zu lassen (siehe Abbildung 9). Das Ziel dieser Arbeit ist allerdings, alle Informationen zentral an einer Stelle darzustellen. Deswegen habe ich die Stellen, an der die Serverinformationen dargestellt werden, mit statischer Codeanalyse analysiert. Die für mich relevanten Codestellen habe ich in der Funktion „getServerInfos“ zusammengetragen. In einigen Fällen habe ich die erhaltenen Informationen mit sogenannten Regulären Ausdrücken gefiltert, um eine komprimierte Ausgabe zu erhalten.

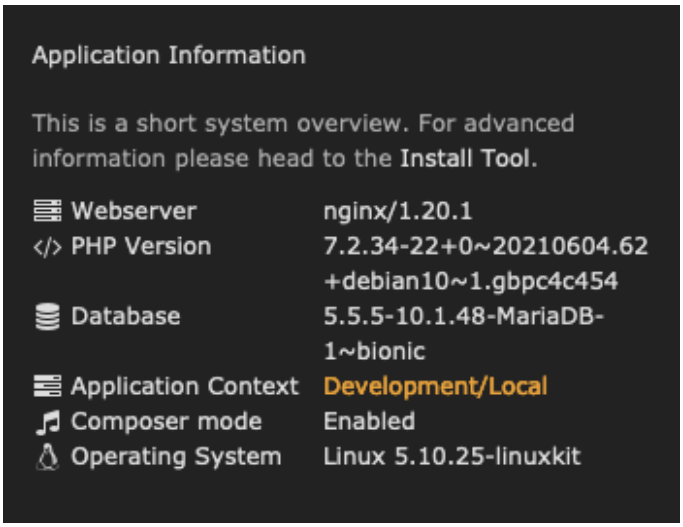

Abbildung 9 – Serverinformationen
Screenshot aus dem TYPO3-Backend

Die gewünschten Kennzahlen müssen aus der Datenbank abgefragt werden. Dabei bestand die Schwierigkeit in der Kompatibilität mit verschiedenen TYPO3 Versionen. Beim Versionssprung von 7 auf 8 wurde die Schnittstelle für Datenbankabfragen als veraltet markiert und in Version 9 entfernt. Seit TYPO3-Version 8 gibt es eine neue Schnittstelle für Datenbankabfragen.[70] Deswegen habe ich eine Abfrage implementiert, welche die TYPO3-Version des aktuellen Systems prüft und abhängig davon die jeweiligen Datenbankabfragen erstellt.

Um eine klare Schnittstelle zu definieren, die von der weiteren Verarbeitung unabhängig ist, werden sowohl die Serverinformationen als auch die Kennzahlen in Form eines PHP-Arrays zurückgegeben.

[70] (TYPO3, 2021)

4.3 Scannen der Extensions

Für die Übersicht der im System enthaltenen Extensions liefert NEC bereits allgemeine Informationen zu den jeweiligen Extensions (a21).

Die Anforderung a22 bezieht sich auf die Dritt-Extensions, die aus dem TER (siehe Grundlagen Extensions) geladen werden. Denn nur für diese kann geprüft werden, ob für die TYPO3-Zielversion bereits eine kompatible Version existiert. Dafür habe ich die Service-Klasse „TerApiService“ geschrieben. Diese führt alle Anfragen an die Schnittstelle vom TER aus. Im Moment enthält die Klasse nur die Funktion „getVersionsForExtensionFromTer“. Als Parameter wird ein Extension-Key übergeben. Die Funktion gibt ein Array zurück, in dem für jede TYPO3-Version ab 7 die letzte kompatible Extension-Version vermerkt ist. Damit die Anfrage ausgeführt werden kann, muss für das jeweilige System ein Access Token generiert werden. Dieser wird im Backend in den Einstellungen der Extension TUA hinterlegt. Der Token wird abgefragt und als Variable innerhalb der Klasse „TerApiService“ vorgehalten, wenn letztere initialisiert wird. Dies ist für die Nutzer in der sogenannten README (Dokumentation der Extension) beschrieben.

Damit die Ergebnisse der Scanning-Tools ausgegeben werden können (a24), müssen mehrere Schritte ausgeführt werden.

Zunächst werden die Extensions gescannt. Ursprünglich verfolgte ich den Ansatz, einen von Vladimir Falcon Piva erarbeiteten Lösungsansatz zu erweitern. Im Interview stellte sich jedoch heraus, dass seine Lösung nicht vollständig funktionsfähig ist. Vladimir Falcon Piva wies im Interview darauf hin, dass es sich nicht um eine klassische Extension handelt. Seine Lösung bildet nach Installation ein eigenes System, welches über einen Konsolenbefehl den Scan startet. Daraus resultierte die Entscheidung seine Lösung nicht zu erweitern, sondern die benötigten Klassen in TUA zu übernehmen.

Mittels statischer Codeanalyse habe ich geprüft, welche Bestandteile der genannten Lösung relevant zum Scannen der im System installierten Extensions sind. Die für den Scan zentralen Klassen sind hierbei „ExtensionScanService“, „PhpCsScanService“ und „LinesOfCodeService“. Die Aufgabe dieser Klassen ist das Ausführen des TYPO3-Scans, das Ausführen des PHP-Scans bzw. das Zählen der Code-Zeilen in jeder Extension. Diese Klassen habe ich übernommen und wie im Folgenden beschrieben überarbeitet. Darin eingeschlossen sind die Klassen, die Vladimir Falcon Piva in den jeweiligen Service-Klassen verwendet. Zur Abgrenzung meiner eigenen Entwicklungsarbeit sind diese weiteren Klassen im Anhang „Auflistung der verwendeten Klassen von Vladimir Falcon Piva“ aufgelistet. Außerdem habe ich die Abhängigkeiten aus der Lösung von Vladimir Falcon Piva als Abhängigkeiten für TUA übernommen.

In der Klasse „LinesOfCodeService“ habe ich der Funktion „scanDirectory“ überarbeitet. Diese Funktion hat ursprünglich den Scan durchgeführt und das Ergebnis in einer Datei abgelegt. Um auch hier eine Unabhängigkeit von der weiteren Verarbeitung zu gewährleisten (vgl. Kapitel 4.2), habe ich die Funktion so umgeschrieben, dass das Ergebnis als PHP-Array zurückgegeben wird. Außerdem habe ich einen boolschen Parameter „outputInFile“ hinzugefügt. Damit haben nachfolgende Benutzer weiterhin die Möglichkeit, die Ausgabe in eine Datei zu nutzen. Abbildung 10 zeigt am Beispiel der Extension „add_wizard“ wie der Rückgabeparameter „linesOfCodeinFiles“ aussieht. Ebenfalls ist der Parameter „outputInFile“ zu sehen. Die Abbildung stammt aus dem Breakpoint-Feature von PhpStrom, das dem Entwickler ermöglicht während der Durchführung an beliebigen Stellen zu stoppen und Parameter einzusehen.

```
$directoryToScan = "/var/www/html/app/web/typo3conf/ext/add_wizard"
> $file = {Symfony\Component\Finder\SplFileInfo} [4]
> $files = {Symfony\Component\Finder\Finder} [20]
$findFileMask = "*.php"
> $finder = {Symfony\Component\Finder\Finder} [20]
> $linesInfile = {array} [2]
v $linesOfCodeInFiles = {array} [5]
    totals = {int} 195
    executionTime = {int} 0
  v 0 = {array} [2]
      lines = {int} 140
      path = "/var/www/html/app/web/typo3conf/ext/add_wizard/Classes/Controller/Wizard/AddController.php"
  v 1 = {array} [2]
      lines = {int} 21
      path = "/var/www/html/app/web/typo3conf/ext/add_wizard/Configuration/Backend/Routes.php"
  v 2 = {array} [2]
      lines = {int} 34
      path = "/var/www/html/app/web/typo3conf/ext/add_wizard/ext_emconf.php"
$outputInFile = false
$outputPath = "/var/www/html/app/web/fileadmin/scan/add_wizard/"
```

Abbildung 10 - Parameter der Funktion scanDirectory in der Klasse LinesOfCodeService
Screenshot aus PhpStorm LinesOfCodeService

In der Klasse „ExtensionScanService“ bestand das Problem, dass Klassen aus einer TYPO3-Kern-Extension verwendet wurden, die erst ab TYPO3 8 zur Verfügung steht. Da die Extension TUA ab Version 7 kompatibel sein soll, musste hier eine Lösung gefunden werden. Dafür habe ich die TYPO3-Version des Systems abgefragt und abhängig davon den Scan ausgeführt. Für die Version 7 muss der Scan über einen Shell-Befehl ausgeführt werden. Das heißt, der Befehl wird abhängig von den Parametern generiert und über den PHP-Befehl „exec“ ausgeführt. Dies hat den Nachteil, dass die Ergebnisse der Scans in Dateien abgelegt werden, was die weitere Verarbeitung verkompliziert. Eine Lösung dafür bietet die Symfony-Console, die es erlaubt, Konsolenabfragen im PHP-Kontext auszuführen und direkt auf das Ergebnis zuzugreifen. Die Umsetzung dieser Lösung war im zeitlichen Rahmen dieser Arbeit nicht möglich. Ist die TYPO3-Version aktueller als 8.0 wurde das Scan-Ergebnis in ein JSON-Array umgewandelt. Das Array wurde in eine Datei geschrieben und anschließend als Rückgabewert zurückgegeben. Ich habe die Konvertierung in ein JSON-Array und das Schreiben in eine Datei entfernt, da das Ergebnis vorher schon als PHP-Array vorlag. Wie bereits erwähnt, vereinfacht das die weitere Verarbeitung.

Das Ergebnis des TYPO3-Scans enthält unter anderem eine detaillierte Liste über die zu aktualisierenden Funktionen. Dazu gehören auch Verlinkungen auf die TYPO3-Dokumentation, in der jeweils erklärt wird, wie die Funktionen auszutauschen sind. Um diese dem Nutzer einfach zugänglich zu machen, sollen die Ergebnisse visuell aufbereitet werden. Hierdurch wird die Analyse für den Backendbenutzer besser nachvollziehbar, so dass die Zeitschätzung gegebenenfalls anpassen werden kann. Zusätzlich kann hier geprüft werden, wie die Funktionen einer Extension im Rahmen eines nachfolgenden Upgrades migriert werden müssen. Dafür bietet die Ausführung des TYPO3-Scans mit dem Befehl „exec“ Möglichkeit, die Ergebnisse in einer HTML-Datei abzulegen. Diese Funktionalität erspart mir das Anlegen und Befüllen eines zusätzlichen Templates. Die Dateien werden im sogenannten Fileadmin (Dateiablagesystem des Backends) abgelegt. In Zukunft soll es dem Backendbenutzer möglich sein, durch Klicken auf einen Link in der Detailansicht einer Extension, das gerenderte Scan-Ergebnis anzeigen zu lassen (vergleiche Abbildung 11).

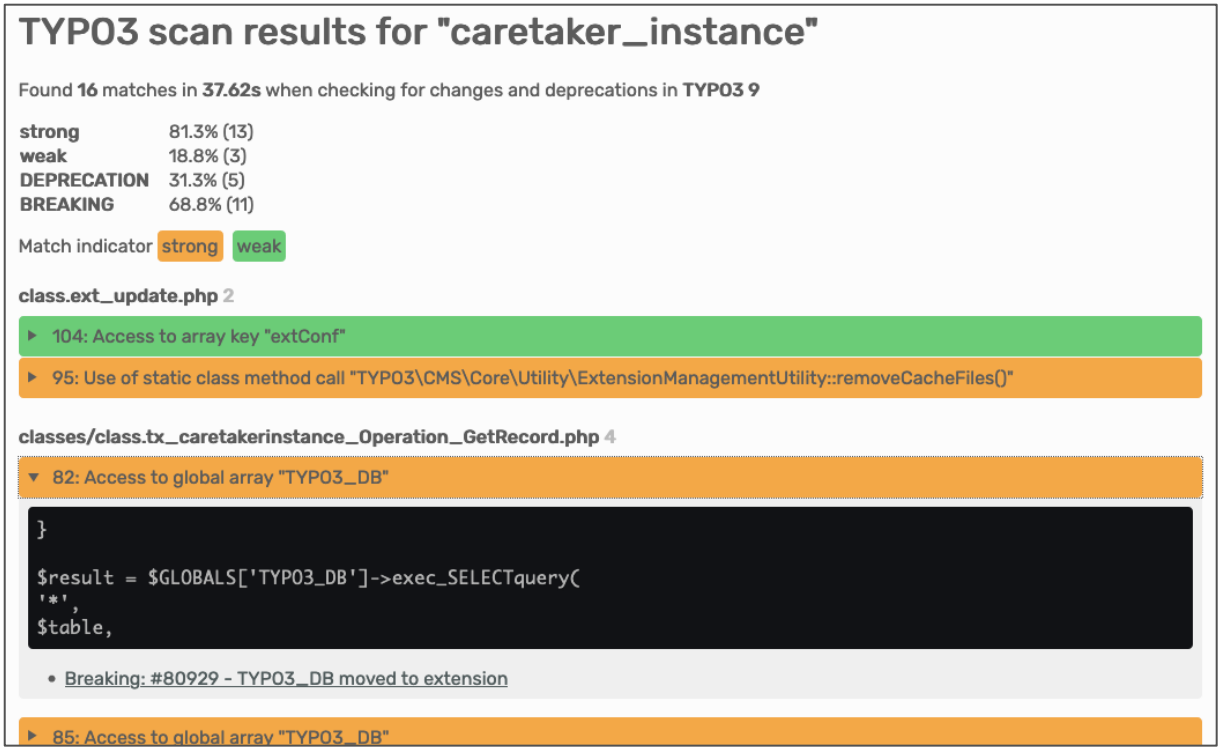

Abbildung 11 - TYPO3-Scan am Beispiel caretaker_instance
Screenshot aus Firefox, HTML-Datei liegt im Projektverzeichnis

Der zweite Punkt umfasst das Verwalten der gewonnenen Daten. Die Ergebnisse der Scans werden in der Datenbank abgelegt, da so die Extensions nur einmal initial gescannt werden müssen. Beim erneuten Aufrufen des Backendmoduls können die Ergebnisse aus der Datenbank geladen werden.

Für das Verwalten der Daten habe ich ein neues Domain Model „Analysis“ erzeugt. Dieses enthält alle Attribute, die für die Einteilung der Extensions in die Upgrade-Kategorien benötigt werden. Dazu gehört, ob es eine mit der TYPO3-Zielversion kompatible Version der Extension gibt oder ob die Extension deinstalliert werden kann. Sind diese Abfragen nicht erfüllt, werden sieben weitere Bedingungen geprüft (siehe Kapitel 2.11, Seite 10): vier im TYPO3-Scanner, zwei im PHP-Scanner und Lines of Code (LOC). Zusätzlich enthält Analysis als Referenz den Extension-Key. Zu dem Model gehört die Datenbanktabelle „tx_typo3upgradeanalysis_domain_model_analysis“ mit der gleichnamigen Konfigurationsdatei. Nach Installation der Extension TUA muss der Nutzer einmal die Datenbank aktualisieren, um die Tabelle hinzuzufügen. Dies ist in der README dokumentiert.

4.4 Entwicklung der Funktion

Nach Anforderung a23 soll die Upgrade-Kategorie der Extension bestimmt werden, aus der die Zeitschätzung abgeleitet wird.

Drei der fünf Interviewpartner haben keine allgemeingültigen Verfahren, um die Extensions in Upgrade-Kategorien einzuteilen. Sie beziehen sich ausschließlich auf ihre Erfahrungen als TYPO3-Entwickler. Dieses ausschließlich implizit vorhandene Wissen ist für mich bei der Automatisierung nicht nutzbar.

Julian Fastnacht und Erik Rauchstein gaben im Interview an, die in Kapitel 2.11 (siehe Seite 10) dargestellten Kategorien zu verwenden. Julian Fastnacht merkte an, dass er keinen Grund sehe, während der Analyse von den Kriterien abzuweichen, da diese über längere

Zeit entwickelt wurden. Nach seiner Meinung sollte bei Bedarf lieber Feedback gegeben werden, um die Regeln generell zu überarbeiten.

Um aus den in Kapitel 4.3 gewonnenen Informationen die Kategorien abzuleiten, habe ich die Kriterien aus Abbildung 4 (Seite 10) verwendet. Da sich die Eigenschaften des Models „Analysis“ mit den zu prüfenden Werten aus dem Referenzmodell deckt (siehe Kapitel 4.3), kann daraus direkt die Kategorie abgeleitet werden.

Beim Übertragen der Bedingungen aus den Tabellen in Code, habe ich festgestellt, dass es keine mathematischen Kriterien für die Kategorie 4 gab. Laut der Tabelle „Kategorien“, gehört eine Extension in die Kategorie 4, wenn sie erweitert und angepasst werden muss. Dies ließ sich auf Basis der Tabelle Kriterien nicht eindeutig abbilden. Beim Prüfen der Kriterien fiel auf, dass sich bei den gelben Maßnahmen (entspricht den Kategorien 2 und 3) je nur eine Bedingung verändert (siehe Abbildung 12; rote Kreise). Im Vergleich dazu verändern sich beim Sprung auf die rote Maßnahme (Kategorie 5) zwei Bedingungen.

Art	TYPO3-Scanner	PHP-Scanner	LOC %	Beschreibung
UPDATEN	Keine Auswirkung	Keine Auswirkung	0	Es steht eine 9.x.-Extension zur Verfügung, die eingespielt werden kann.
ENTFERNEN	Keine Auswirkung	Keine Auswirkung	0	Extension kann vom Server deinstalliert und gelöscht werden. Ggf. keine Anpassung.
ANPASSEN	Strong & Breaking = 0 Strong & Deprecated = 0 Weak & Breaking = 0 Weak & Deprecated > 0	Warnings > 0 Errors = 0	> 10%	Extension muss angepasst werden; Aufwand gering.
ERWEITERN	Strong & Breaking = 0 Strong & Deprecated > 0 Weak & Breaking = 0 Weak & Deprecated > 0	Warnings > 0 Errors = 0	> 20%	Extension muss erweitert werden; Aufwand mittel.
ANPASSEN	Strong & Breaking > 0 Strong & Deprecated > 0 Weak & Breaking > 0 Weak & Deprecated > 0	Warnings > 0 Errors > 0	> 50%	Extension muss erweitert werden; Aufwand Hoch bis nicht einschätzbar
TP	Strong & Breaking > 0 Strong & Deprecated > 0 Weak & Breaking > 0 Weak & Deprecated > 0	Warnings > 0 Errors > 0	> 70%	Auslagerung in ein eigenes Teilprojekt möglich

Abbildung 12 - Kriterien zur Einteilung in Upgrade-Kategorien (Baghdadi & Fastnacht, Ist-Analyse & Aufwandsschätzung, 2020)

Daher habe ich mich dazu entschieden die Kriterien für Kategorie 4 wie folgt zu definieren:

- TYPO3-Scanner: alle Werte außer „Strong & Breaking“ dürfen größer als 0 sein
- PHP-Scanner: die Kriterien bleiben wie bei Kategorie 2 und 3
- LOC %: >35%

So ändert sich im TYPO3-Scanner je eine Bedingung, wenn die Kategorie um 1 erhöht wird. Der Anteil LOC orientiert sich mittig zwischen den Werten für die Kategorien 3 und 5. In Abbildung 13 ist die daraus resultierende Funktion zu sehen.

```php
/**
 * @param $extKey
 * @throws \TYPO3\CMS\Extbase\Persistence\Exception\IllegalObjectTypeException
 * @throws \TYPO3\CMS\Extbase\Persistence\Exception\UnknownObjectException
 */
protected function getUpgradeCategoryForExtension($extKey)
{
    /** @var Analysis $analysis */
    $analysis = $this->analysisRepository->getAnalysisByExtensionKey($extKey);

    //Extensions with category 0 (system extensions) are loaded directly from the database
    //for the entire calculation
    if ($analysis->isCompatibleVersion() == 1 || $analysis->isDeactivated()) {
        $category = 1;
    } elseif ($analysis->getPhpErrors() == 0 && $analysis->getExtensionScanStrongBraking() == 0
        && $analysis->getExtensionScanStrongDeprecated() == 0
        && $analysis->getExtensionScanWeakBraking() == 0) {
        $category = 2;
    } elseif ($analysis->getPhpErrors() == 0 && $analysis->getExtensionScanStrongBraking() == 0
        && $analysis->getExtensionScanStrongDeprecated() == 0) {
        $category = 3;
    } elseif ($analysis->getPhpErrors() == 0 && $analysis->getExtensionScanStrongBraking() == 0) {
        $category = 4;
    } else {
        $category = 5;
    }

    $analysis->setCategory($category);
    $this->analysisRepository->update($analysis);
    $this->persistenceManger->persistAll();

}
```

Abbildung 13 - Funktion zur Ermittlung der Upgrade-Kategorie einer Extension
Screenshot aus PhpStorm UpgradeAnalysisController

Die nächste Abbildung zeigt beispielhaft in Welche Kategorie die Extension cartaker_instance einsortiert wurde. Die Ergebnisse des TYPO3-Scans sind zum Vergleich in Abbildung 11 (Seite 24) zu sehen.

```
$analysis = {CPSIT\Typo3UpgradeAnalysis\Domain\Model\Analysis} [18]
    linesOfCode = {int} 6911
    extKey = "caretaker_instance"
    phpWarnings = {int} 1
    phpErrors = {int} 0
    extensionScanStrongBraking = {int} 11
    extensionScanWeakBraking = {int} 0
    extensionScanStrongDeprecated = {int} 2
    extensionScanWeakDeprecated = {int} 3
    category = {int} 5
    compatibleVersion = false
    deactivated = false
    uid = {int} 49
    _localizedUid = {int} 49
    _languageUid = {int} 0
    _versionedUid = {int} 49
    pid = {int} 1
```

Abbildung 14 - Attribute eines Objekts der Klasse Analysis, nach Bestimmung der Upgrade-Kategorie
Screenshot aus PhpStorm UpgradeAnalysisController

Kategorie 1 enthält Extensions, für die bereits eine kompatible Version existiert oder die deinstalliert werden können. In Kapitel 4.3 wurde bereits erläutert, wie bestimmt wird, ob eine kompatible Version vorhanden ist. Nun gilt noch zu klären, wann eine Extension deinstalliert werden sollte. Diese Frage habe ich meinen Interviewpartnern gestellt.

Aus Anna Schneiders und Erik Rauchsteins Sicht braucht man dafür Wissen über das System. Extensions können nur entfernt werden, wenn diese vom Kunden nicht mehr benötigt werden. Lediglich bei den bereits deaktivierten Extensions ist diese Frage beantwortet. Hier ist also eine automatisierte Entscheidung nicht möglich und es muss individuell mit dem Kunden gesprochen werden.

Dirk Wenzel und Vladimir Falcon Piva entfernen Extensions, wenn die Codequalität zu schlecht ist. Dafür wurden allerdings bis jetzt keine Kriterien definiert. Die entfernten Extensions müssten in diesem Fall ersetzt werden.

Julian Fastnacht verfolgt einen dritten Ansatz. Er prüft, wann das letzte Mal an einer Extension gearbeitet wurde. In einem nachfolgenden Gespräch gab er an, dass er dafür prüft, wann der letzte Commit gemacht wurde. Ist der letzte Stand zu alt, sollte die Extension aus seiner Sicht deinstalliert und durch eine neue ersetzt werden. Auf Nachfrage, welchen Zeitraum er empfehlen würde, wurde als Beispiel ein Projekt genannt, in dem die Zeitspanne 8 Monate war.

Dadurch, dass die Kommunikation mit dem jeweiligen Kunden individuell ist und die Aussagen über die Codequalität nicht eindeutig sind, kann im Moment nicht automatisiert geprüft werden, ob eine Extension deinstalliert werden soll. Es müsste zunächst Kriterien zur Messung der Codequalität definiert werden.

Im Anschluss wird aus den ermittelten Kategorien der gesamte Zeitaufwand kalkuliert. Dafür habe ich eine Formel aus dem Vorgehen für Upgrade-Analysen bei familie redlich (siehe Kapitel 2.11, Seite 12) verwendet. Da TUA noch nicht in der Lage ist, automatisiert zu prüfen, für welche Extensions eine Dokumentation vorhanden ist, nehme ich an, dass keine Tec-Doku vorhanden ist und verwende die entsprechende Formel. Diese Formel ergänze ich um ein Budget für Testen und Projektleitung. Diese beiden Faktoren wurden von Erik

Rauchstein und Vladimir Falcon Piva (Testen) sowie Anna Schneider (Projektleitung) als bisher nicht berücksichtigte Faktoren genannt. Erik Rauchstein gab an, dass das Testvolumen mindestens 20% des Projektvolumens sein sollte. Über den für die Projektleitung einzukalkulierenden Anteil hatte ich keine Aussage. Aus Gesprächen mit Projektleitern der Firma ergab sich auch hier ein Anteil von 20% des Projektvolumens. Demnach habe ich die mit der Formel kalkulierte Zeitschätzung mit dem Faktor 1,4 multipliziert.

```php
/**
 * @return float|int|mixed
 */
protected function doEffortCalculation()
{
    // $effort = ['category' => 'estimated time per extension (hours)']
    $effort = [
        0 => 0.25,
        1 => 2,
        2 => 4,
        3 => 8,
        4 => 20,
        5 => 64,
    ];

    $extensions = $this->analysisRepository->findAll();
    $estimatedTime = 0;
    foreach ($extensions as $extension) {
        $category = $extension->getCategory();
        if ($category == 0) {
            $estimatedTime += $effort[$category];
        } else {
            $estimatedTime += $effort[$category] * 2;
        }

    }

    return $estimatedTime * 1.4;
}
```

Abbildung 15 - Funktion zur Berechnung des Zeitaufwands des Upgrades
Screenshot aus PhpStrom UpgradeAnalysisController

4.5 Darstellen der gesammelten Informationen

Um die Daten aus den vorhergehenden Kapiteln auszugeben, habe ich den Controller „nsextcompatibilityController“ mit dem Controller „UpgradeAnalysisController“ erweitert.

Für die Darstellung der Systeminformationen und Kennzahlen musste ich die „listAction“ überschreiben. Da ich die Informationen aus der ursprünglichen listAction nicht verlieren wollte, habe ich sie aus der überschreibenden Funktion heraus aufgerufen.

Im Anschluss prüft die neue listAction ob im Backend eine TYPO3-Zielversion ausgewählt ist. In diesem Fall wird die Funktion „scanExtensions“ aufgerufen, welche die im System enthaltenen Extensions (nicht die TYPO3-Kern-Extensions) scannt. Darin wird zunächst geprüft, ob es im TER neuere kompatible Extension-Versionen für die jeweiligen TYPO3-Versionen gibt. Da der Scan sehr zeitaufwändig ist, wird er grundsätzlich nur dann ausgeführt, wenn noch kein Objekt der Klasse „Analysis“ mit dem entsprechenden Extension-Key existiert. Wurde der Scan ausgeführt, wird die Funktion „getUpgradeCategoryForExtension“ ausgeführt. Der ermittelte Wert wird ebenfalls im zugehörigen Analysis-Objekt gespeichert. Die Funktion „scanExtensions“ enthält bereits die Möglichkeit die Aktualisierung der bestehenden Analysis-Objekte zu erzwingen. Das Backendmodul stellt diese Funktionalität dem Nutzer bislang noch nicht zur Verfügung.

Im Anschluss ruft die listAction die Funktionen zum Ermitteln der Serverinformationen und Kennzahlen auf. Die Ergebnisse werden an das View-Objekt übergeben, welches das zur Action gehörige Template „List“ rendert. Die Ausgabe erfolgt wie in Abbildung 16 zu sehen.

System Information	
Current TYPO3 Version	7.6.39
Target TYPO3 Version	
Database	MariaDB
Image Rendering	im6
Application Context	warning
UTF8	
Operating System	Linux 5.10.25
Web Server	nginx/1.20.1
PHP Version	7.2.34
MySQL version	10.3.29

Abbildung 16 - Ausgabe der Systeminformationen im Backendmodul
Screenshot Backendmodul der Extension typo3_upgrade_analysis

Um die Ansicht der Informationen im Backend übersichtlich zu halten, habe ich mich dafür entschieden, die Ergebnisse der Scanning-Tools (Kapitel 4.3 und 4.4) nicht in der allgemeinen Übersicht darzustellen. NEC verfügt schon über eine detailAction, welche die Informationen zu einer Extension darstellt, wenn der Backendbenutzer die entsprechende Extension aus der Liste anklickt. In dieser Einzelübersicht sollen die Ergebnisse des Scans angezeigt werden. Im zeitlichen Rahmen dieser Arbeit war es nicht möglich die Darstellung

der Scan-Ergebnisse komplett umzusetzen. Dafür hätte die detailAction wie bei der listAction überschrieben werde müssen (siehe Seite 29).

Damit die zusätzlichen Variablen aus der listAction auch ausgegeben werden, musste ich das zugehörige Template überschreiben. So wie das Backendmodul aus der Extension NEC konfiguriert ist, war es mir nicht möglich die Templates aus der Extension TUA als alternative Templates zu definieren. Deswegen musste ich ein eigenes Backenmodul registrieren. Damit im Backend nicht zwei fast identische Module existieren, habe ich die Konfiguration des ursprünglichen Moduls mit einem leeren Array überschrieben.

Die Menge der formulierten Anforderungen und die zeitliche Beschränkung im Rahmen dieser Arbeit erforderte eine Priorisierung. Die minimalen Anforderungen für ein lauffähiges Programm umfassten zunächst die automatisierte Analyse und die direkt daraus abzuleitende Zeitschätzung. Die Anforderungen a25 bis a28, sowie a3 bis a5 gehen darüber hinaus. Sie sind für ein erstes lauffähiges Programm nicht zwingend erforderlich.

5 Abschluss

5.1 Zusammenfassung

Die Arbeit hat sich mit der Problemstellung befasst, den Prozess der TYPO3-Upgrade-Analyse, wie er bei familie redlich durchgeführt wird, zu automatisieren.

Das erste Kapitel der Arbeit hat die Motivation und die grundlegende Problemstellung beschrieben, die zu dieser Arbeit führten. Auch der grobe Aufbau der Arbeit wurde dargestellt.

Im Anschluss wurden die Grundlagen für diese Arbeit erläutert. Zu Beginn wurden die Begriffe Open Source, TYPO3 und Upgrade für eine bessere Einordung der Problemstellung geklärt. Auch wurde auf den Prozess der Upgrade-Analyse eingegangen. Im Weiteren wurden verwendete Programme und deren Vorteile erläutert. Außerdem wurde ein grundlegendes Wissen für die Entwicklung einer Extension im Kontext von TYPO3 generiert.

In Kapitel drei wurde die statische Codeanalyse und das leitfadengestützte Experteninterview vorgestellt und erklärt, für welche Zwecke die jeweilige Methode verwendet werden sollte. Die statische Codeanalyse wurde herangezogen, um fremden Code zu verstehen. Die Experteninterviews bilden die Grundlage, auf der später der Code entwickelt wurde. Außerdem wurde aufgezeigt, wie die Interviewleitfäden entwickelt und nach welchen Kriterien die Experten ausgewählt worden sind.

Das vierte Kapitel stellte die Forschungsergebnisse dar. Hier wurden die Interviews ausgewertet und gegenübergestellt. Zunächst wurde aus den Interviews ein Anforderungskatalog an eine Extension formuliert, die eine TYPO3-Upgrade-Analyse automatisieren sollte. Auf Grundlage dieser Anforderungen wurde eine TYPO3-Extension entwickelt. Die Extension wurde gegen das Extbase-Framework entwickelt und ist ab der TYPO3-Version 7 kompatibel.

Bei der Auswertung der Interviews habe ich dargestellt, dass die Vorgehen der einzelnen Personen bei der Durchführung einer TYPO3-Upgrade-Analyse zum Teil sehr verschieden sind. Hier bestand die Schwierigkeit darin, die Verfahren zusammenzuführen und in Code zu übertragen. Eine weitere Schwierigkeit bestand darin, eine Kompatibilität für mehrere TYPO3-Verisonen sicherzustellen.

Die nachfolgende Tabelle stellt noch einmal da, welche Anforderungen im Rahmen der Arbeit inwieweit umgesetzt wurden.

Kennung	Anforderung	Erfüllt
a1	Auflistung der Systeminformationen	Zum Teil
a11	TYPO3-, PHP-, MySQL-Version	ja
a12	Betriebssystem, Web-Server, Datenbank	ja
a13	Weitere PHP-Informationen	nein
a14	Kennzahlen	ja
a15	Core-Klassen, die erweitert wurden	nein
a16	Verwendete Hooks	nein
a2	Extension-Übersicht erstellen	Zum Teil
a21	Allgemeine Informationen	ja
a22	Kompatible Version der Extensions (falls vorhanden)	Darstellung fehlt, Funktion ist umgesetzt
a23	Zeitschätzung bzw. Kategorie (durch die Upgrade-Analyse bestimmt)	Darstellung fehlt, Funktion ist umgesetzt
a24	Ergebnisse der Scanning-Tools	Darstellung fehlt, Funktion ist umgesetzt
a25	Sortierung der Extensions nach Kategorien	nein
a26	Feld für Entwicklerschätzung	nein
a27	Codequalität	nein
a28	Testabdeckung	nein
a3	Extensions markieren	nein
a4	Suche nach Funktionen	nein
a5	Report exportieren	nein

Tabelle 3 - Anfoderungskatalog mit Beschreibung des Umsetzungsstandes

5.2 Diskussion der Ergebnisse

Die durchgeführten Interviews haben sich als gute Quelle erwiesen, um den Ablauf der Analyse zu verstehen und potentielle Schwachstellen am Referenzmodell aufzudecken. Dabei wurde unter anderem auf fehlende Faktoren hingewiesen, die ein Upgrade verzögern können. Ein weiteres Problem bestand bei der Einteilung der Extensions in Kategorien. Für die Upgrade-Kategorie 4 waren keine klaren Kriterien definiert. Ich habe mich dafür entschieden, die Kriterien so zu definieren, dass sie das Mittel zwischen den Kennwerten für die Kategorien 3 und 5 bilden.

Ich habe die Extension ns_ext_compatibility erweitert, indem ich sie als Abhängigkeit in der Extension typo3_upgrade_analysis definiert habe. Eine Alternative dazu wäre der Fork (siehe Kapitel 2.3.1, Seite 4) gewesen. Trotz der oben beschriebenen Vorteile habe ich mich gegen diese Methode entschieden, da ich die gesamten Namespaces (gibt an zu welcher Klasse eine Extension gehört) auf den Namen meiner Extension hätte anpassen müssen. Weiterhin habe ich so eine klare Trennung zwischen dem Code aus der Extension ns_ext_compatibilty und meiner Eigenleistung geschaffen.

Betrachtet man die Funktion getUpgradeCategoryForExtension fällt auf, dass ich ein umfangreiches if-else-Konstrukt verwendet habe. Da die Werte vieler verschiedener Variablen geprüft werden mussten, habe ich mich dafür entschieden. Die Alternative Switch habe ich verworfen, da dort nur verschiedene Werte derselben Variablen betrachtet werden.

Für das Zusammentragen der Systeminformationen habe ich unter anderem Funktionen aus dem Framework Extbase verwendet. Dies geschah zunächst unter der Prämisse, eine bereits existierende Funktion nicht noch einmal zu schreiben. Außerdem gehören die verwendeten Funktionen zum Core. Dadurch bleiben die Schnittstellen häufig auch dann konstant, wenn sich die Funktionalität oder Zugriff auf einzelne Variablen verändert. Die Lauffähigkeit meiner Funktion bleibt weiterhin erhalten.

Aufgrund der Menge an Anforderungen, musste ich die Entscheidung treffen, welche Anforderungen im Rahmen dieser Arbeit umgesetzt werden sollten. Bei der Aufnahme der Kennzahlen habe ich die Anforderungen umgesetzt, die sich am Referenzmodell orientieren. Bei der Umsetzung der automatisierten TYPO3-Upgrade-Analyse habe ich die Anforderungen als minimal definiert, ohne die Kategorisierung der Extensions und die daraus resultierende Zeitschätzung nicht umsetzbar wäre.

5.3 Fazit

Die Arbeit beschäftigte sich mit der Problemstellung der Automatisierung einer TYPO3-Upgrade-Analyse. Die vorliegende Arbeit kommt zu dem Ergebnis, dass die gesamte nach dem Referenzmodel durchgeführte TYPO3-Upgrade-Analyse, automatisierbar ist. Alle Anforderungen aus dem Referenzmodel sind umgesetzt, auch wenn die Ausgabe noch nicht vollständig und die Funktion noch nicht komplett lauffähig ist. Die Experteninterviews haben aber gezeigt, dass es vielfältige Anforderungen an eine sinnvolle TYPO3-Upgrade-Analyse gibt, die über das Referenzmodel hinaus gehen. Diese müssen noch umgesetzt und zum Teil auch erst noch konkretisiert werden. Ein weiteres Ergebnis dieser Arbeit ist die Formulierung dieser Anforderungen.

5.4 Ausblick

Im Interview wurde die Frage gestellt, welche Anforderungen ein Backendmodul erfüllen müsste, damit es den Interviewpartnern die Durchführung der Upgrade-Analyse erleichtert oder abnimmt. Die Frage wurde bewusst offen formuliert und hat deutlich mehr Ergebnisse erzielt, als im zeitlichen Rahmen dieser Arbeit möglich war. Die nicht umgesetzten Anforderungen können einen Leitfaden bei der weiteren Entwicklung der Extension typo3_uprade_analysis bilden.

Einige der befragten Personen äußerten Kritik am Referenzmodell, die in Zukunft bei der automatisierten Upgrade-Analyse berücksichtigt werden könnte. Erik Rauchstein gab an, dass bei Extensions von Dritten die Versionssprünge berücksichtigt werden sollten. Wird mindestens eine Version übersprungen, sollte die Extension in die nächst höhere Upgrade-Kategorie eingeteilt werden. Hier gilt zu prüfen, inwieweit diese Eingruppierung sinnvoll ist. Eventuell ist hier ein Faktor sinnvoller, mit dem die Zeitschätzung multipliziert wird. Ebenfalls sollte sich gemäß Erik Rauchstein das Testvolumen dieser Extension je Versionssprung um 20% erhöhen. Außerdem überlegte er, ob eine Metrik zur Kalkulation des allgemeinen Testvolumens möglich wäre.

Zukünftige Wissenschaftliche Arbeiten könnten untersuchen inwieweit die automatisierte Zeitschätzung die Realität abbildet und die Funktionen eine Unterstützung bei der Durchführung einer TYPO3-Upgrade-Analyse sind.

6 Literaturverzeichnis

Atlassian. (2021). *Sourcetree | Free Git GUI for Mac and Windows*. SourceTree. https://www.sourcetreeapp.com

Baghdadi, S. (2018, Februar). *Konzeption eines Vorgehens für die Durchfuhrung eines Upgrades von TYPO3*. https://doi.org/10.13140/RG.2.2.27003.62240

Baghdadi, S., & Fastnacht, J. (2020, 12. August). *Ist-Analyse & Aufwandsschätzung* [Intranet]. CPS-IT Confluence. https://confluence.cps-projects.de/pages/viewpage.action?pageId=50900237

Bhatt, A. (2020, 17. Juni). *Was ist eine TYPO3-Erweiterung? Eine Einführung*. T3TERMINAL. https://t3terminal.com/blog/de/typo3-erweiterungs-anleitung

Bommer, C., Spindler, M., & Barr, V. (2008). *Softwarewartung: Grundlagen, Management und Wartungstechniken.* (1. Aufl.). dpunkt.verlag.

Chauhan, S. (2018, 19. Juni). *[T3EXT DAY] TYPO3 Extensions Compatibility Report*. NITSAN. https://www.nitsan.in/de/blog/t3ext-day-typo3-extensions-compatibility-report/

Composer. (o.D.) *Introduction - Composer*. Abgerufen am 20.September 2021, von https://getcomposer.org/doc/00-intro.md

familie redlich. (2018, 8. März). *familie redlich : Startseite*. familie redlich. https://www.familie-redlich.de/startseite/

git. (o. D.). *Git*. Abgerufen am 17. September 2021, von https://git-scm.com/

Gläser, J., & Laudel, G. (2010). *Experteninterviews und qualitative Inhaltsanalyse : als Instrumente rekonstruierender Untersuchungen* (4. Aufl.). VS Verlag für Sozialwissenschaften.

IONOS SE. (2020, 15. Juni). *Git vs. SVN – Von verteilter und zentralisierter Versionsverwaltung*. IONOS Digitalguide. https://www.ionos.de/digitalguide/websites/web-entwicklung/git-vs-svn-versionsverwaltung-im-vergleich/

JET BRAINS. (2021, 2. Juni). *PhpStorm: die blitzschnelle und smarte PHP-IDE von*. JetBrains. https://www.jetbrains.com/de-de/phpstorm/

jweiland.net. (2021, 19. August). *TYPO3 Versionen 6.2 bis 11.x Stand 2021* [Grafik]. https://jweiland.net/fileadmin/images/typo3/version/2021-08-19_TYPO3_Versionen.svg

Kaiser, R. (2014). *Qualitative Experteninterviews: Konzeptionelle Grundlagen und praktische Durchführung (Elemente der Politik)* [E-Book]. Springer VS.

Kenning, P., Lackes, R., & Siepermann, M. (2018, 19. Februar). *Upgrade*. Gabler Wirtschaftslexikon. https://wirtschaftslexikon.gabler.de/definition/upgrade-47384

Khan, A. (2021, 14. Juni). *Top Code Editors and IDE for PHP Development of 2021* . Cloudways. https://www.cloudways.com/blog/top-ide-and-code-editors-php-development/

Lackes, R., & Siepermann, M. (2018, 19. Februar). *Open Source*. Gabler Wirtschaftslexikon. https://wirtschaftslexikon.gabler.de/definition/open-source-43032

Lackes, R., & Siepermann, M. (2018, 19. Februar). *Update*. Gabler Wirtschaftslexikon. https://wirtschaftslexikon.gabler.de/definition/update-49022

Meuser, M., & Nagel, U. (2013). Experteninterviews - wissenssoziologische Voraussetzungen und methodische Durchführung. In A. Langer, B. Friebertshäuser, & A. Prengel (Hrsg.), *Handbuch Qualitative Forschungsmethoden in der Erziehungswissenschaft* (4. Aufl., S. 457-471). Beltz Juventa.

Meyer, R., & Helmich, M. (2019). *Praxiswissen TYPO3 CMS 9 LTS.* O`Reilly.

Preston-Werner, T. (o. D.). *Semantic Versioning 2.0.0*. Semantic Versioning. Abgerufen am 15. September 2021, von https://semver.org/lang/de/

Schams, M. (2021, 12. Februar). *TYPO3 Extbase Moderne Extension-Entwicklung mit Extbase und Fluid*.

Springer, N., Koschel, F., Fahr, A., & Pürer, H. (2015). *Empirische Methoden der Kommunikationswissenschaft* (1. Aufl.) [E-Book]. UTB GmbH.

Stachmann, B., & Preißel, R. (2019). *Git* [E-Book]. Beltz Verlag.

Synopsys, Inc. (2021) *Compare Repositories - Open Hub*. Synopsys. https://www.openhub.net/repositories/compare

The History of TYPO3. (o. D.). TYPO3. Abgerufen am 12. Juli 2021, von https://typo3.org/project/history abgerufen

Trabold, C., Hasenau, J., & Niederlag, P. (2009). *TYPO3 Kochbuch.* O`Reilly Verlag GmbH & Co. KG.

TYPO3. (3. September 2021). *Deprecation: #78244 - Deprecate TYPO3_DB and Prepared Statement class – TYPO3 CMS Core ChangeLog Master documentation*. TYPO3. https://docs.typo3.org/c/typo3/cms-core/master/en-us/Changelog/8.5/Deprecation-78244-DeprecateTYPO3_DBAndPreparedStatementClass.html

TYPO3 Documentation Team. (2021, 9. August). *XCLASSes (Extending Classes) - TYPO3*. TYPO3. https://docs.typo3.org/m/typo3/reference-coreapi/master/en-us/ApiOverview/Xclasses/Index.html

Urheberrecht.de. (2021, 13. August). *Open Source – Welche Regeln gelten bei der freien Software?* Urheberrecht.de. https://www.urheberrecht.de/open-source/

Weiland, J. (2021, 19. August). *TYPO3 Versionen:Infos und Tipps zu Updates – jweiland.net*. Copyright (c) 2021 jweiland.net – Jochen Weiland. https://jweiland.net/typo3/versionen-und-updates.html

7 Abbildungsverzeichnis

Abbildung 1 - Registrierung einer Erweiterung Screenshot aus PhpStorm ext_localconf.php 7

Abbildung 2 - Aufnahme der Systemkonfiguration (Baghdadi & Fastnacht, Ist-Analyse & Aufwandsschätzung, 2020) 9

Abbildung 3 - Beispiel für Kennzahlen (Baghdadi & Fastnacht, Ist-Analyse & Aufwandsschätzung, 2020) 10

Abbildung 4 - Kriterien zur Gruppierung der Extensions (Baghdadi & Fastnacht, Ist-Analyse & Aufwandsschätzung, 2020) 10

Abbildung 5 - Kategorien mit zugehöriger Zeitschätzung (Baghdadi & Fastnacht, Ist-Analyse & Aufwandsschätzung, 2020) 11

Abbildung 6 - Prinzip der qualitativen Inhaltsanalyse (Gläser & Laudel, 2010, S. 200) 16

Abbildung 7 - Abhängigkeiten der Extension typo3_upgrade_analysis Screenshot aus PhpStorm composer.json 18

Abbildung 8 - TYPO3 Versionen 6.2 bis 11.x Stand 2021 (jweiland.net, 2021) 18

Abbildung 9 – Serverinformationen Screenshot aus dem TYPO3-Backend 21

Abbildung 10 - Parameter der Funktion scanDirectory in der Klasse LinesOfCodeService Screenshot aus PhpStorm LinesOfCodeService 23

Abbildung 11 - TYPO3-Scan am Beispiel caretaker_instance Screenshot aus Firefox, HTML-Datei liegt im Projektverzeichnis 24

Abbildung 12 - Kriterien zur Einteilung in Upgrade-Kategorien (Baghdadi & Fastnacht, Ist-Analyse & Aufwandsschätzung, 2020) 25

Abbildung 13 - Funktion zur Ermittlung der Upgrade-Kategorie einer Extension Screenshot aus PhpStorm UpgradeAnalysisController 26

Abbildung 14 - Attribute eines Objekts der Klasse Analysis, nach Bestimmung der Upgrade-Kategorie Screenshot aus PhpStorm UpgradeAnalysisController 27

Abbildung 15 - Funktion zur Berechnung des Zeitaufwands des Upgrades Screenshot aus PhpStrom UpgradeAnalysisController 28

Abbildung 16 - Ausgabe der Systeminformationen im Backendmodul Screenshot Backendmodul der Extension typo3_upgrade_analysis 29

8 Tabellenverzeichnis

Tabelle 1 - Experten 15

Tabelle 2 - Anforderungskatalog Backendmodul 20

Tabelle 3 - Anfoderungskatalog mit Beschreibung des Umsetzungsstandes 32

9 Anhang

Allgemeiner Leitfaden

Leitfaden für das Interview mit Vladimir Falcon Piva (technischer Bestandteil)

Interviewprotokolle

Gegenüberstellung der Protokolle

Auflistung der verwendeten Klassen von Vladimir Falcon Piva

Allgemeiner Leitfaden

Länge: 1 Stunde

Befragte: 4 Personen

Eröffnung des Gesprächs:

- Aktueller Stand
  - Ich habe bereits zwei Upgrade-Analysen durchgeführt. Dabei bin ich nach dem Firmeninternen Plan vorgegangen. Ich habe meine Ergebnisse hinterher jeweils mit einem erfahrenen Entwickler ausgewertet.
- Einordnung des Interviews
  - Das Interview wird im Rahmen meiner Bachelor Arbeit geführt. Ich verfolge das Ziel den Prozess der TYPO3-Upgrade-Analyse zu automatisieren.
  - Die im Interview gewonnenen Informationen sollen als Grundlage für die Entwicklung einer Funktion dienen, die den Aufwand des Upgrades kalkulieren soll.
- Ist es für dich ok, wenn ich das Interview aufzeichne und im Rahmen meiner Bachelor Arbeit analysiere und veröffentliche?

Einstiegsfragen

1. Wie lange arbeitest du schon mit TYPO3?
2. Was ist deine Rolle, wenn du mit TYPO3 arbeitest? (Entwickler, Integrator, …)
3. Wie viele Upgrades hast du ungefähr durchgeführt?
4. Hast du schon von Anfang an Upgrade-Analysen durchgeführt? Wenn nicht, seit wann machst du Upgrade-Analysen?

Hauptfragen

5. Welche Phasen siehst du bei der Durchführung einer Analyse? (Auf die Analyse der Extensions gehen wir gleich noch mal genauer ein.)
   - Wie fließen die Kennzahlen über zum Beispiel Benutzer und Content Elemente in die Aufwandsschätzung ein?
6. Wie gehst du vor, wenn du die Extensions analysierst?
   - Welche Informationen werden dafür erhoben?
   - Was machst du mit diesen Informationen? (im Hinblick auf Zeitschätzung)
   - Gibt es Situationen, in denen du von diesem Vorgehen abweichst? Warum ist das dort der Fall?
7. Wie erkenne ich Extensions die nicht mehr verwendet werden (sollten)?
   - Auf welcher Grundlage triffst du die Entscheidungen?
     - Eigene Erfahrungen
   - Gibt es dafür Faktoren?
     - Verhältnis Changes zu „Lines of Code“
     - Kein Support mehr
     - Nicht kompatibel mit der TYPO3 Version
8. Gibt es weitere Faktoren, die den Aufwand des Upgrades vergrößern?

9. Gibt es spezielle Extensions, die deutlich mehr Zeit benötigt haben oder waren alle zu knapp kalkuliert?
   - Wenn ja, was waren das für Extensions?
10. Wenn du an vergangene Projekte denkst, für die du Upgrade-Analyse gemacht hast. Was würdest du anders machen, wenn du die Analysen nochmal machen könntest?
11. Kannst du daraus Regeln für nächste Upgrades formulieren?
    - Wäre ein Faktor sinnvoll, der berücksichtigt wie viele Versionen übersprungen werden?

- Denkst du, dass man die menschliche Komponente zusätzlich braucht?

Abschlussfragen

12. Angenommen wir haben ein Tool, dass die Analyse durchführt und zusätzlich ein BE Modul zur Visualisierung hat. Was wünschst du dir von diesem Modul?
13. Möchtest du noch etwas zu dem Thema sagen, das deiner Meinung nach im Interview nicht ausreichend berücksichtigt wurde?

Leitfaden für das Interview mit Vladimir Falcon Piva

Länge: 1 Stunde

Befragte: 1 Person

Eröffnung des Gesprächs:

- Aktueller Stand
  - Ich habe bereits zwei Upgrade-Analysen durchgeführt. Dabei bin ich nach dem Firmeninternen Plan vorgegangen. Ich habe meine Ergebnisse hinterher jeweils mit einem erfahrenen Entwickler ausgewertet.
- Einordnung des Interviews
  - Das Interview wird im Rahmen meiner Bachelor Arbeit geführt. Ich verfolge das Ziel den Prozess der TYPO3-Upgrade-Analyse zu automatisieren.
  - Die im Interview gewonnenen Informationen sollen als Grundlage für die Entwicklung einer Funktion dienen, die den Aufwand des Upgrades kalkulieren soll.
- Ist es für dich ok, wenn ich das Interview aufzeichne und im Rahmen meiner Bachelor Arbeit analysiere und veröffentliche?

Einstiegsfragen

1. Wie lange arbeitest du schon mit TYPO3?
2. Was ist deine Rolle, wenn du mit TYPO3 arbeitest? (Entwickler, Integrator, …)
3. Wie viele Upgrades hast du ungefähr durchgeführt?
4. Hast du schon von Anfang an Upgrade-Analysen durchgeführt? Wenn nicht, seit wann machst du Upgrade-Analysen?

Hauptfragen

5. Welche Phasen siehst du bei der Durchführung einer Analyse? (Auf die Analyse der Extensions gehen wir gleich noch mal genauer ein.)
   - Wie fließen die Kennzahlen über zum Beispiel Benutzer und Content Elemente in die Aufwandsschätzung ein?
6. Wie gehst du vor, wenn du die Extensions analysierst?
   - Welche Informationen werden dafür erhoben?
   - Was machst du mit diesen Informationen? (im Hinblick auf Zeitschätzung)
   - Gibt es Situationen, in denen du von diesem Vorgehen abweichst? Warum ist das dort der Fall?
7. Wie erkenne ich Extensions die nicht mehr verwendet werden (sollten)?
   - Auf welcher Grundlage triffst du die Entscheidungen?
     - Eigene Erfahrungen
   - Gibt es dafür Faktoren?
     - Verhältnis Changes zu „Lines of Code“
     - Kein Support mehr
     - Nicht kompatibel mit der TYPO3 Version
8. Gibt es weitere Faktoren, die den Aufwand des Upgrades vergrößern?

9. Gibt es spezielle Extensions, die deutlich mehr Zeit benötigt haben oder waren alle zu knapp kalkuliert?

   - Wenn ja, was waren das für Extensions?
10. Wenn du an vergangene Projekte denkst, für die du Upgrade-Analyse gemacht hast. Was würdest du anders machen, wenn du die Analysen nochmal machen könntest?
11. Kannst du daraus Regeln für nächste Upgrades formulieren?
    - Wäre ein Faktor sinnvoll, der berücksichtigt wie viele Versionen übersprungen werden?
    - Denkst du, dass man die menschliche Komponente zusätzlich braucht?

Z1. Du hattest schon eine Extension programmiert, die einen Teil der Analyse der installierten Extensions übernimmt. Kannst du mir kurz zusammenfassen was die Funktion dieser Extension ist?

Z2. Kann man deine Extension in einem System integrieren (oder siehst du vor, dass man die Funktionen außerhalb des zu prüfenden Systems aufruft)?

- Ja
    - Gibt es Hooks, die ich verwenden kann, um die Funktionen aufzurufen?
- Nein
    - Worauf muss ich achten, wenn ich den Aufruf von externen Scripten in eine Extension einbinden will?
    - Was für Strukturen sollte ich implementieren?
        - Services
        - Klassen die ich erweitern sollte

<u>Abschlussfragen</u>

12. Angenommen wir haben ein Tool, dass die Analyse durchführt und zusätzlich ein BE Modul zur Visualisierung hat. Was wünschst du dir von diesem Modul?
13. Möchtest du noch etwas zu dem Thema sagen, das deiner Meinung nach im Interview nicht ausreichend berücksichtigt wurde?

Interview 1

Einstiegsfragen

1. Wie lange arbeitest du schon mit TYPO3?

   8 Jahre

2. Was ist deine Rolle, wenn du mit TYPO3 arbeitest? (Entwickler, Integrator, …)

   Backendentwickler und Integrator

3. Wie viele Upgrades hast du ungefähr durchgeführt?

   6

4. Hast du schon von Anfang an Upgrade-Analysen durchgeführt? Wenn nicht, seit wann machst du Upgrade-Analysen?

   ja

Hauptfragen

5. Welche Phasen siehst du bei der Durchführung einer Analyse? (Auf die Analyse der Extensions gehen wir gleich noch mal genauer ein.)
   - Wie fließen die Kennzahlen über zum Beispiel Benutzer und Content Elemente in die Aufwandsschätzung ein?

   Bestandsaufnahme: Erfassen aller Extensions inklusive der vorhandenen Versionsnummer; TYPO3 Version; Konfiguration (Inhaltselemente, Funktionen, Update Wizards, Scheduler Tasks); Serverkonfiguration

   Aufbereitung der Informationen: Auflisten der Informationen in Confluence; Einschätzen des Systemzustands, Extensions und Content-Zustand

   Kennzahlen: je mehr Content-Elemente vorhanden, desto komplexer das System; je mehr Benutzergruppen, desto mehr müssen geprüft werden -> komplex; aber wenig Benutzergruppen und viele Content-Elemente spricht für eine schlechte Gruppenaufteilung

6. Wie gehst du vor, wenn du die Extensions analysierst?
   - Welche Informationen werden dafür erhoben?
   - Was machst du mit diesen Informationen? (im Hinblick auf Zeitschätzung)
   - Gibt es Situationen, in denen du von diesem Vorgehen abweichst? Warum ist das dort der Fall?

   Welche Version der Extension ist aktuell installiert und gibt es für die TYPO3-Zielversion schon eine kompatible Version der Extension + automatisierter Updatepfad

   Wie viele Lines of Code hat die Extension

   Codequalität wird geprüft: Wie gut kann man den Code lesen und verstehen (Programmatisches- und tiefergehendes TYPO3-Wissen)

   Typo3 scanner: Informationen über (zukünftig) nicht mehr unterstützte Funktionen

   Auswertung: Kategorisierung nach Tabelle (siehe Kapitel 2.11, Seite 11)

7. Wie erkenne ich Extensions die nicht mehr verwendet werden (sollten)?
   - Auf welcher Grundlage triffst du die Entscheidungen?
     - Eigene Erfahrungen
   - Gibt es dafür Faktoren?
     - Verhältnis Changes zu „Lines of Code“

- Kein Support mehr
- Nicht kompatibel mit der TYPO3 Version

Absprache mit Kunden: Extensions die nicht mehr verwendet werden

Deinstallierte Extensions im System

Wenn keine kompatible Version vorhanden ist muss der Dienstleister die Extension weiterentwickeln oder im Zweifelsfall neu schreiben

8. Gibt es weitere Faktoren, die den Aufwand des Upgrades vergrößern?

   Anzahl der Versionssprünge der TYPO3-Version

   Testing 20% des Projektvoulmens, das ist aber eigentlich zu wenig

   Gut wäre eine Metrik, anhand der das Testvolumen angepasst wird (Anzahl schwierige Extensions, Anzahl TYPO3-Versionssprünge, …)

9. Gibt es spezielle Extensions, die deutlich mehr Zeit benötigt haben oder waren alle zu knapp kalkuliert?
   - Wenn ja, was waren das für Extensions?

   Nein, meistens werden die für den Kunden individualisierten Extensions zum Problem oder Unterschiede indem wie früher gepflegt wurde und jetzt gepflegt werden müsste

   Das einzige war die Umstellung RTEditor zum CKEditor

10. Wenn du an vergangene Projekte denkst, für die du Upgrade-Analyse gemacht hast. Was würdest du anders machen, wenn du die Analysen nochmal machen könntest?

    Höhere Zeiten schätzen

    Bsp: TYPO3 Verison 7 -> Version 9 neues Routing wurde vorab angeschaut und sah mit einem neuen System gut aus, aber es zeigte sich, dass sich das Verhalten wie Urls generiert werden komplett geändert hat. Problem für Systeme, die stark auf den alten Paradigmen basierten

    Höhere TYPO3-Versionssprünge als 1 benötigen mehr Zeit

    Größere Sprünge in den Extension-Versionen wurden nicht berücksichtigt; sollten in Zukunft in eine höhere Version eingruppiert werden und das Testvolumen sollte erhöht werden

11. Kannst du daraus Regeln für nächste Upgrades formulieren?
    - Wäre ein Faktor sinnvoll, der berücksichtigt wie viele Versionen übersprungen werden?

    TYPO3-Versionssprung > 1: Produktvolumen * 1,5

    Extension-Versionssprung > 1: + Extension-Zeitaufwand * 0,2 je Versionssprung; weiterhin sollte diese Extension durch einen Entwickler betrachtet werden

    Wenn Regeln sich wiedersprechen, muss ein Entwickler die Extension prüfen: (Bsp.) Versionssprung > 2, aber der Upgradepfad ist komplett beschrieben, dewegen ist es eine einfache Kategorie. Es stellt sich dann aber heraus, dass zwei Hauptfunktionen aus dieser Extension entfernt wurden, die für das System wichtig waren, wodurch die Extension neu geschrieben/durch eine andere ersetzt werden muss -> höchste Kategorie

Entwickler muss prüfen: Extension-Versionssprung > 1; >20% der Changes einer Extension als breaking markiert wurden

Es wird schwer komplett automatisiert eine Zeitschätzung durchzuführen (fehlende Regeln oder Regeln die sich gegenseitig aufheben)

Abschlussfragen

12. Angenommen wir haben ein Tool, dass die Analyse durchführt und zusätzlich ein BE Modul zur Visualisierung hat. Was wünschst du dir von diesem Modul?

    Übersicht über die installierten Extensions: (Sortierung nach) Kategorisierung der jeweiligen Extension, Detaillink (Lines of Code, Typo3 scanner), vorhandene Extension-Versionen

    Systeminformationen: TYPO3 Version, PHP-Version, installierte PHP-Extensions (enthalten in der PHP-Info), Memory-Limit, Bildrendering, Content-Elemente, Benutzergruppen, registrierte Hooks

    Suchfunktion nach (deprecated/breaking) Funktionen

    CSV-, Pdf-Export

    Anpassungen der Zeitschätzungen der Extensions möglich (Entwicklerschätzung)

13. Möchtest du noch etwas zu dem Thema sagen, das deiner Meinung nach im Interview nicht ausreichend berücksichtigt wurde?

    Kein Projekt, das kalkuliert wurde, konnte in der geschätzten Zeit umgesetzt werden

    Kategorien müssen noch verfeinert werden (nicht nah genug an der Realität)

    Wenn keine Serverinformationen vorhanden -> Risikofaktor: + 0,25 * Projektvolumen

    Zusammenspiel der Abteilungen (IT, Projektleitung, Testing)

    Wenn automatisierte Tests vorhanden, könnte man diese im Backend in der Extensionconfiguration hinterlegen mit einem Zeitfaktor

Interview 2

Einstiegsfragen

1. Wie lange arbeitest du schon mit TYPO3?

   13 Jahre

2. Was ist deine Rolle, wenn du mit TYPO3 arbeitest? (Entwickler, Integrator, …)

   Letzten Drei Jahre Backend, jetzt auch Beratung, davor Frontend + Integration

3. Wie viele Upgrades hast du ungefähr durchgeführt?

   Etwas über 10, davon 5 große Upgrades

4. Hast du schon von Anfang an Upgrade-Analysen durchgeführt? Wenn nicht, seit wann machst du Upgrade-Analysen?

   Am Anfang nicht formalisiert

Hauptfragen

5. Welche Phasen siehst du bei der Durchführung einer Analyse? (Auf die Analyse der Extensions gehen wir gleich noch mal genauer ein.)
   - Wie fließen die Kennzahlen über zum Beispiel Benutzer und Content Elemente in die Aufwandsschätzung ein?

   Bestandsaufnahmen (welche Komponenten sind verbaut)

   Komponenten (Extensions) bewerten

   Statische Codeanalyse: Vollständigkeit der Dokumentation; Features, die verwendet werden; werden die Features noch benötigt

   Kennzahlen als Warnzeichen: viele Benutzergruppen sprechen für ein komplexes System oder schlechte Einteilung der Gruppen; ab mehr als 9 Benutzergruppen wird es schwierig

   Testabdeckung und/oder Dokumentation vorhanden: erleichtert das Upgrade

6. Wie gehst du vor, wenn du die Extensions analysierst?
   - Welche Informationen werden dafür erhoben?
   - Was machst du mit diesen Informationen? (im Hinblick auf Zeitschätzung)
   - Gibt es Situationen, in denen du von diesem Vorgehen abweichst? Warum ist das dort der Fall?

   Bekannte Extensions, die immer wieder Schwierigkeiten bei Upgrades machen (powermail, realurl)

   Bekannte Extensions, die bis jetzt keine Probleme bei Upgrades gemacht haben (tx_news)

   Genaueres Prüfen von Custom Extensions: sind meistens nicht so gut mit Tests und Dokumentationen abgedeckt wie bei weit verbreiteten Open-Source-Extensions

   Genaueres Prüfen von unbekannte Extensions

   Genaueres Prüfen: prüfen der Versionshistorie; prüfen der Dokumentation; prüfen des Codes

   Zeitschätzung: muss der Entwickler etwas ablösen, wenn ja prüfen wie komplex das wird (z.B.: komplexe Datenbankabfragen, müssen Schnittstellen abgelöst werden)

   Hat keine Kategorien, in die die Extensions eingruppiert werden können um eine Zeitschätzung abzuleiten

Könnte (je Extension) nach Lines of Code, Anzahl der Abhängigkeiten, vorhandene Hooks abgrenzen

7. Wie erkenne ich Extensions die nicht mehr verwendet werden (sollten)?
    - Auf welcher Grundlage triffst du die Entscheidungen?
        - Eigene Erfahrungen
    - Gibt es dafür Faktoren?
        - Verhältnis Changes zu „Lines of Code“
        - Kein Support mehr
        - Nicht kompatibel mit der TYPO3 Version

    Wenn nicht verstanden wird was die Extension macht: rausschmeißen, wenn sie wichtig war, wird jemand schreien (Risiko besteht)

    Wenn Codequalität schlecht ist: entferne die Extensions und ersetze sie durch eine neue -> damit der Aufwand beim Upgrade nicht zu groß wird

    Codequalität als Entscheidungsgrundlage

8. Gibt es weitere Faktoren, die den Aufwand des Upgrades vergrößern?

    Vorhandenes/nicht vorhandenes Wissen beim Kunden

    Erfahrungen der Entwickler

9. Gibt es spezielle Extensions, die deutlich mehr Zeit benötigt haben oder waren alle zu knapp kalkuliert?
    - Wenn ja, was waren das für Extensions?

    Meistens werden schwierige Extensions auch in zukünftigen Upgrades schwierig, da der Entwickler hinter der Extension meistens gleichbleibt und sich die Arbeitsweis nicht doll ändert

    Library-Sammlungen können komplex werden

    Extensions, die ein komplexes Thema unterkomplex darstellen wollen (Bsp: Bootstrap)

10. Wenn du an vergangene Projekte denkst, für die du Upgrade-Analyse gemacht hast. Was würdest du anders machen, wenn du die Analysen nochmal machen könntest?

    Große Menge unterschätzt, Bsp.: Viele Templates und Partials, die Ursachen waren viele Seitentypen -> komplex

    Varnish und Solr erhöhen die Komplexität eines Systems

    Risiko höher bei einem Versionssprung größer als 1

11. Kannst du daraus Regeln für nächste Upgrades formulieren?
    - Wäre ein Faktor sinnvoll, der berücksichtigt wie viele Versionen übersprungen werden?

    Schätzungen nur für bereits vorhandene Komponenten

<u>Abschlussfragen</u>

12. Angenommen wir haben ein Tool, dass die Analyse durchführt und zusätzlich ein BE Modul zur Visualisierung hat. Was wünschst du dir von diesem Modul?

Codequalität

Je Extensions: Wenn viele Klassen, Templates oder Codezeilen enthalten sind

Hooks, Abhängigkeiten

Abdeckung durch Tests und Dokumentationen

13. Möchtest du noch etwas zu dem Thema sagen, das deiner Meinung nach im Interview nicht ausreichend berücksichtigt wurde?

    Nichts zur Analyse

Interview 3

Einstiegsfragen

1. Wie lange arbeitest du schon mit TYPO3?

   5 Jahre

2. Was ist deine Rolle, wenn du mit TYPO3 arbeitest? (Entwickler, Integrator, …)

   Nimmt Anforderungen für (hauptsächlich) Extensions auf, mit dem Hintergrund, dass die Extension in TYPO3 entwickelt wird.

3. Wie viele Upgrades hast du ungefähr durchgeführt?

   Keine, aber 5 – 10 Upgrade-Analysen (Grund: Funktion im Umgang mit TYPO3)

4. Hast du schon von Anfang an Upgrade-Analysen durchgeführt? Wenn nicht, seit wann machst du Upgrade-Analysen?

   Frage gestrichen, da in Frage 3 bandwortet

Hauptfragen

5. Welche Phasen siehst du bei der Durchführung einer Analyse? (Auf die Analyse der Extensions gehen wir gleich noch mal genauer ein.)
   - Wie fließen die Kennzahlen über zum Beispiel Benutzer und Content Elemente in die Aufwandsschätzung ein?

   Systemzustand: TYPO3-Version; Befüllen einer Vorlage für die Ist-Analyse (Siehe Kapitel 2.11, Seite 9f)

   Zu viele Benutzer (-gruppen) Zeit einplanen fürs aufräumen

   Upgrade nicht nur Systemupdate machen, sondern auch als Möglichkeit sehen Altlasten zu beseitigen

   Fazit (wäre es vielleicht besser das System neu zu machen) + Kostenschätzung

   Nutzung der Kennzahlen über CTypes, listType, MenuType (auf Basis der Extensionkeys): Gibt Auskunft darüber ob eine Extension noch genutzt wird

6. Wie gehst du vor, wenn du die Extensions analysierst?
   - Welche Informationen werden dafür erhoben?
   - Was machst du mit diesen Informationen? (im Hinblick auf Zeitschätzung)
   - Gibt es Situationen, in denen du von diesem Vorgehen abweichst? Warum ist das dort der Fall?

   Informationen: Kategorie, Name, Extensionkey, state (Entwicklungszustand; nicht gut standardisiert, deswegen bei der Analyse nicht hilfreich), Extensionbeschreibung

   Informationen aus TYPO3 scanner und php scanner

   Zeitaufwand wird aus den Informationen (Extension und Tools) abgeleitet (Kapitel 2.11, Seite 11); konkrete Zeitschätzung pro Extension ist massiver Aufwand und die Einteilung in Kategorien bildet einen Mittelweg zwischen konkret und pi-mal-Daumen

   Wenn Tecdoku vorhanden, ist die Zeitschätzung pro Extension geringer (Da man Informationen über die Extension nicht aus dem Code gewinnen muss)

   Sollte von dem Vorgehen nur in begründeten Fällen abweichen, da das Vorgehen begründet ist: Start mit einem Vorgehen (Ursprung unbekannt), Julian (Interviewpartner) formal verfeinert, Sammy hat seine Bachelor Arbeit über dieses Thema geschrieben, Erweitert um Tools,

7. Wie erkenne ich Extensions die nicht mehr verwendet werden (sollten)?
    - Auf welcher Grundlage triffst du die Entscheidungen?
        - Eigene Erfahrungen
    - Gibt es dafür Faktoren?
        - Verhältnis Changes zu „Lines of Code“
        - Kein Support mehr
        - Nicht kompatibel mit der TYPO3 Version

   Wenn Extension über einen gewissen Zeitraum nicht weiterentwickelt wird

   Fall Upgrade TYPO3-Version 7.6.x auf 9.5.x, Analyse durchgeführt 13.08.18, markieren aller Extensions, die nach dem 01.01.18 nicht weiterentwickelt wurden -> 8 Monate

8. Gibt es weitere Faktoren, die den Aufwand des Upgrades vergrößern?

   Kein Zugriff aufs System, kann man nur sehr schlecht abschätzen (sollte nicht angenommen werden)

   Wenn zusätzlich Erweiterungen am System vorgenommen werden (nicht empfehlenswert)

9. Gibt es spezielle Extensions, die deutlich mehr Zeit benötigt haben oder waren alle zu knapp kalkuliert?
    - Wenn ja, was waren das für Extensions?

   Nicht bekannt, fehlt Information aufgrund von Fehlenden Auswertungen auf Extensionebene nach dem Upgrade

10. Wenn du an vergangene Projekte denkst, für die du Upgrade-Analyse gemacht hast. Was würdest du anders machen, wenn du die Analysen nochmal machen könntest?

    Hat der Interviewpartner bereits gemacht: In jeder Iteration von Upgrade-Analysen wurde Feedback in das genannte Vorgehen eingearbeitet und zu dem aktuellen Stand geführt

11. Kannst du daraus Regeln für nächste Upgrades formulieren?
    - Wäre ein Faktor sinnvoll, der berücksichtigt wie viele Versionen übersprungen werden?
    - Denkst du, dass man die menschliche Komponente zusätzlich braucht?

    Wurde mit Frage 10 bandwortet

    Menschliche Komponente ist nicht zu vernachlässigen (aus dem Nachgespräch)

<u>Abschlussfragen</u>

12. Angenommen wir haben ein Tool, dass die Analyse durchführt und zusätzlich ein BE Modul zur Visualisierung hat. Was wünschst du dir von diesem Modul?

    Übersicht über vorhandene Versionen einer Extension (aus TER, Gitlab, GitHub, …)

    Genannten Informationen aus Frage 5

13. Möchtest du noch etwas zu dem Thema sagen, das deiner Meinung nach im Interview nicht ausreichend berücksichtigt wurde?

Nichts zur Analyse

Interview 4

Einstiegsfragen

1. Wie lange arbeitest du schon mit TYPO3?

   17 Jahre

2. Was ist deine Rolle, wenn du mit TYPO3 arbeitest? (Entwickler, Integrator, …)

   Entwicklerin + Integratorin

3. Wie viele Upgrades hast du ungefähr durchgeführt?

   10 - 15

4. Hast du schon von Anfang an Upgrade-Analysen durchgeführt? Wenn nicht, seit wann machst du Upgrade-Analysen?

   Ja, aber zu Beginn noch nicht so formalisiert

Hauptfragen

5. Welche Phasen siehst du bei der Durchführung einer Analyse? (Auf die Analyse der Extensions gehen wir gleich noch mal genauer ein.)
   - Wie fließen die Kennzahlen über zum Beispiel Benutzer und Content Elemente in die Aufwandsschätzung ein?

   Es muss unterschieden werden ob es sich um ein reines Upgrade oder Relaunch mit Upgrade handelt

   Feststellen: was ist das Ausgangssystem und was ist das Zielsystem

   Sind die Extensions für die Zielversion im TER vorhanden (*Vorgriff auf Frage 6*)

   Wie ist der Stand der eigenen Extensions

   Serverarchitektur prüfen

   Keine Kennzahlen, da jede Upgrade-Analyse einzigartig ist

   Prüfen in wie weit Upgrade-Wizzards schon vorhanden sind, die genutzt werden können

   Benutzergruppen und Content-Elemente spielen keine Rolle für das Upgrade; die Kennzahlen spiegeln hier nur ein bisschen die Komplexität des Upgrades (interessant zur Übersicht für den Kunden)

   Fokus bei der Analyse liegt hier auf dem Code (Begründung: viele einfach CTypes können im Zweifelsfall schneller abgelöst werden, als wenige komplexe)

6. Wie gehst du vor, wenn du die Extensions analysierst?
   - Welche Informationen werden dafür erhoben?
   - Was machst du mit diesen Informationen? (im Hinblick auf Zeitschätzung)
   - Gibt es Situationen, in denen du von diesem Vorgehen abweichst? Warum ist das dort der Fall?

   Hängt davon ab ob die Extension bekannt ist

   Bei fremden Extensions ist es oft wie mit einer Blackbox und die Analyse ist schwierig. Da wird eine Pauschale genommen

   Typo3Scanner wird genutzt: gibt Überblick über breaking changes und deprecated functions

   Keine festen Werte an denen die Zeitschätzung festgemacht wird

Wenn breaking changes öfter auftreten ist die Fehlerbehebung meistens überall ähnlich und benötigt bei allen anderen Vorkommen (außer dem ersten) weniger Zeit

Es gibt auch Fehler, die nicht durch einen Scanner gefunden werden; z.B.: fehlende Frontendausgabe

(z.B.) Erforderliche Migrationsschritte für die Zielversion kann man meistens erst genauer überblicken, wenn man schon mal ein Upgrade auf diese Zielversion gemacht hat

50 mal der gleiche breaking change ist schneller behoben, als wenn es 10 verschiedene breaking changes sind.

7. Wie erkenne ich Extensions die nicht mehr verwendet werden (sollten)?
    - Auf welcher Grundlage triffst du die Entscheidungen?
        - Eigene Erfahrungen
    - Gibt es dafür Faktoren?
        - Verhältnis Changes zu „Lines of Code“
        - Kein Support mehr
        - Nicht kompatibel mit der TYPO3 Version

Braucht Wissen übers System

Extensions die in ähnlicher Form in den Core aufgenommen wurden können entfernt werden (benötigt Erfahrung in der Backendentwicklung)

Feedback von Kunden oder der Redaktion, dass eine Funktion (Extension) schlecht zu nutzen ist/nocht mehr benötigt wird

8. Gibt es weitere Faktoren, die den Aufwand des Upgrades vergrößern?

Interne Absprachen benötigen viel Zeit; je mehr Teams, die miteinander arbeiten, desto mehr Zeitaufwand

Onboarding-Zeit benötigt, wenn Projekt unbekannt

Wenn das Projekt größer ist, wird auch mehr Zeit für die Kommunikation benötigt

Onboarding-Zeit/Kommunikationszeit muss am Ende für das gesamte Projekt kalkuliert werden

Projektleitung

9. Gibt es spezielle Extensions, die deutlich mehr Zeit benötigt haben oder waren alle zu knapp kalkuliert?
    - Wenn ja, was waren das für Extensions?

Eigentlich nicht, vielleicht „multicolumn“ (dafür wird mehr Zeit eingeplant)

10. Wenn du an vergangene Projekte denkst, für die du Upgrade-Analyse gemacht hast. Was würdest du anders machen, wenn du die Analysen nochmal machen könntest?

nichts

11. Kannst du daraus Regeln für nächste Upgrades formulieren?
    - Wäre ein Faktor sinnvoll, der berücksichtigt wie viele Versionen übersprungen werden?
    - Denkst du, dass man die menschliche Komponente zusätzlich braucht?

Der/Die Entwickler, die das Upgrade durchführen, sollten auch an der Analyse beteiligt sein

Abschlussfragen

12. Angenommen wir haben ein Tool, dass die Analyse durchführt und zusätzlich ein BE Modul zur Visualisierung hat. Was wünschst du dir von diesem Modul?

    Zusätzlich zu den Systeminformationen Informationen über veraltete Datenbank-Connections

    Information ob RTE oder CKEditor

    Informationen ob die Extensions für die Zielversion schon existieren (Existieren die Dritt-Extensions noch nicht kann man keine Aussage machen)

    Typo3Scanner Informationen

13. Möchtest du noch etwas zu dem Thema sagen, das deiner Meinung nach im Interview nicht ausreichend berücksichtigt wurde?

    Wichtig Kenntnisse über das System zu haben

    Teilung in der Vorlage zwischen mit/ohne Tec-Doku ist nicht realistisch, da man in bekannten Systemen auch ohne Dokumentation nicht mehr Zeit benötigt

    Vorhandene Produktkenntnis ist ein Faktor, der bis jetzt nicht berücksichtigt wird

    Wenn neue Architekturen/Hilfsmittel/Software eingesetzt/entwickelt werden soll, muss jeweils genug Zeit eingeplant werden

Interview 5

Einstiegsfragen

1. Wie lange arbeitest du schon mit TYPO3?

   14 Jahre

2. Was ist deine Rolle, wenn du mit TYPO3 arbeitest? (Entwickler, Integrator, …)

   Full-Stack-Entwickler (Entwickler, Integrator, Frontend)

3. Wie viele Upgrades hast du ungefähr durchgeführt?

   Ca 100 (inklusive Relaunches)

4. Hast du schon von Anfang an Upgrade-Analysen durchgeführt? Wenn nicht, seit wann machst du Upgrade-Analysen?

   Nicht immer, für Relaunches wurden keine klassischen Upgrade-Analysen gemacht

Hauptfragen

5. Welche Phasen siehst du bei der Durchführung einer Analyse? (Auf die Analyse der Extensions gehen wir gleich noch mal genauer ein.)
   - Wie fließen die Kennzahlen über zum Beispiel Benutzer und Content Elemente in die Aufwandsschätzung ein?

   Systemuntersuchen mit Tools

   Analyse aller Extensions

   Prüfen ob das System manipuliert oder erweitert wurde

   Gespräch mit dem Kunden

   Custom-Elemente werde erst im Rahmen der Extensions geprüft (nur auf Codeebene; die Kennzahlen werden nicht berücksichtigt)

   Benutzerrechte machen meistens keine Probleme

6. Wie gehst du vor, wenn du die Extensions analysierst?
   - Welche Informationen werden dafür erhoben?
   - Was machst du mit diesen Informationen? (im Hinblick auf Zeitschätzung)
   - Gibt es Situationen, in denen du von diesem Vorgehen abweichst? Warum ist das dort der Fall?

   Zuerst wird die sogenannte Sitepackage (-Extension; hier sind alle Systemkonfigurationen drin) geprüft

   Es wird geprüft ob Datenmigrationen nötig sind (Nötig, wenn eine Extension durch eine andere ersetzt wird, oder wenn sich Datenbanktabellen ändern)

   Tools zum Scannen der Extension: Typo3Scanner

   Keine festen Zahlen für die Zeitschätzung. Die Zeitschätzung erfolgt aus der Erfahrung.

7. Wie erkenne ich Extensions die nicht mehr verwendet werden (sollten)?
   - Auf welcher Grundlage triffst du die Entscheidungen?
     - Eigene Erfahrungen
   - Gibt es dafür Faktoren?
     - Verhältnis Changes zu „Lines of Code“
     - Kein Support mehr
     - Nicht kompatibel mit der TYPO3 Version

Im Installtool sichtbar, wenn die Extension nicht kompatibel ist oder Fehler hat. (muss aber ersetzt werden)

Grid_elements ist nicht so gut zu verwenden (persönliche Meinung)

8. Gibt es weitere Faktoren, die den Aufwand des Upgrades vergrößern?

   Testen

   Zeit für die Abnahme des Kunden (Testen und eventuelle Schulungen)

9. Gibt es spezielle Extensions, die deutlich mehr Zeit benötigt haben oder waren alle zu knapp kalkuliert?
   - Wenn ja, was waren das für Extensions?

   Mittlerweile nicht mehr (vorausgesetzt man hat eine halbwegs aktuelle Version)

   Man benötigt mehr Aufwand, wenn die TYPO3 Versionen (Start und Ziel) weiter auseinander sind

10. Wenn du an vergangene Projekte denkst, für die du Upgrade-Analyse gemacht hast. Was würdest du anders machen, wenn du die Analysen nochmal machen könntest?

    Es gibt immer Raum für Verbesserungen, aber nichts Konkretes

    Technologisch aktuell bleiben: prüfen welche Tools es gibt, um die Analysezeit zu reduzieren und die Qualität zu steigern

11. Kannst du daraus Regeln für nächste Upgrades formulieren?
    - Wäre ein Faktor sinnvoll, der berücksichtigt wie viele Versionen übersprungen werden?
    - Denkst du, dass man die menschliche Komponente zusätzlich braucht?

    Mach eine gute Analyse

    Drauf achten, wenn sich der Aufbau von Elementen geändert hat

Z1. Du hattest schon eine Extension programmiert, die einen Teil der Analyse der installierten Extensions übernimmt. Kannst du mir kurz zusammenfassen was die Funktion dieser Extension ist?

    Die Extension wurde nicht bis zu Ende entwickelt. Sie kam nie zum Einsatz

    Müsste aktualisiert werden

    Lauffähig mit Docker und Ddev

Z2. Kann man deine Extension in einem System integrieren (oder siehst du vor, dass man die Funktionen außerhalb des zu prüfenden Systems aufruft)?

    - Ja
      - Gibt es Hooks, die ich verwenden kann, um die Funktionen aufzurufen?
    - Nein
      - Worauf muss ich achten, wenn ich den Aufruf von externen Scripten in eine Extension einbinden will?
      - Was für Strukturen sollte ich implementieren?
        - Services
        - Klassen die ich erweitern sollte

Ist zu lange her, um darüber Auskunft geben zu können

Extension ist nur schlecht dokumentiert, deswegen nicht verwenden

Abschlussfragen

12. Angenommen wir haben ein Tool, dass die Analyse durchführt und zusätzlich ein BE Modul zur Visualisierung hat. Was wünschst du dir von diesem Modul?

    Reports der Extensions (mögliche Fehler, deprecaded/breaking Code)

    Report Exportieren (Informationen sind immer gut. Je mehr je besser)

    Super wäre TypoScript Scanner

13. Möchtest du noch etwas zu dem Thema sagen, das deiner Meinung nach im Interview nicht ausreichend berücksichtigt wurde?

    Nein

Frage	Erik Rauchstein	Dirk Wenzel	Julian Fastnacht	Anna Schneider	Vladimir Falcon Piva
Phasen der Analyse	Bestandsaufnahme: Erfassen aller Extensions, TYPO3 Version, Serverkonfiguration, Konfigurationen (Inhaltselemente, Funktionen, Update Wizards, Scheduler Tasks) Aufbereitung der Informationen: Auflisten, Einschätzen des Systemzustands/der Extensions/Content	Bestandsaufnahme: welche Komponenten (Extensions) sind verbaut Bewerten der Extensions Statische Codeanalyse: Vollständigkeit der Dokumentation, welche Features werden verwendet und noch benötigt Ist eine Testabdeckung und/oder Dokumentation vorhanden	Systemzustand: TYPO3 Version, Serverzustand, Kennzahlen (Benutzergruppen, Content-Elemente, … siehe Protokoll) Fazit: Soll das System neu gemacht werden oder soll ein Upgrade gemacht werden? Kostenschätzung	Feststellen: Eigenschaften Ausgangssystem und Zielsystem Stand der systemspezifischen Extensions prüfen Serverarchitektur prüfen Prüfen welche Upgrade-Wizzards vorhanden sind	Systemuntersuchen mit Tools Analyse aller Extensions Prüfen ob das System manipuliert oder erweitert wurde Gespräch mit dem Kunden
Aussage der Kennzahlen	Je mehr Content-Elemente, desto komplexer das System Viele Benutzergruppen -> komplex Wenige Benutzergruppen + viele Contentelemente -> schlechte Aufteilung der Gruppen	Warnzeichen: viele Benutzergruppen -> komplexes System oder schlechte Einteilung der Gruppen Mehr als 9 Benutzergruppen -> Warnzeichen	Viele Benutzergruppen -> Zeit einplanen fürs aufräumen Kennzahlen über CTypes, listTypes, menüTypes, … (in Abhängigkeit pro Extension) gibt Auskunft ob eine	Keine Kennzahlen, da jede Upgrade-Analyse einzigartig ist Kennzahlen spiegeln nur grob die Komplexität, werden nicht in der Zeitschätzung berücksichtigt Fokus liegt auf dem Code (viele einfache	Custom-Elemente werde erst im Rahmen der Extensions geprüft (nur auf Codeebene; die Kennzahlen werden nicht berücksichtigt) Benutzerrechte machen meistens keine Probleme

			Extension noch genutzt wird.	CTypes werden im Zweifelsfall schneller abgelöst als wenige komplizierte)	
Analyse der Extensions (Dritt-Extensions)	Prüfen der aktuell installierten Version Gibt es schon eine kompatible Version mit der Zielversion Ist der Upgrade-Pfad definiert/automatisert Sonst: Wie viele Lines of Code hat die Extension Prüfen der Codequalität (wie gut kann man den Code lesen und verstehen) Typo3Scanner: Informationen über (zukünftig) nicht mehr unterstützte Funktionen	Bekannte Extensions wie powermail, realurl die Probleme machen Bzw wie tx_news die gut laufen Risiko bewusst sich darauf zu verlassen Unbekannte Extensions genau prüfen (siehe systemspezifische Extensions)	Festhalten der Informationen einer Extensions (Kategorie, Name, …) Informationen aus Typo3Scanner und php scanner	Sind die Extensions in einer kompatiblen Version mit dem Zielsystem vorhanden? Wenn nicht: oft wie Blackbox, Planung mit Pauschale (hohes Risiko)	Zuerst wird die sogenannte Sitepackage (-Extension; Systemkonfigurationen) geprüft Es wird geprüft ob Datenmigrationen nötig sind (Nötig, wenn eine Extension durch eine andere ersetzt wird, oder wenn sich Datenbanktabellen ändern) Tools zum Scannen der Extension: Typo3Scanner
Analyse der Extensions (systemspezifische Extensions)		Genaues Prüfen: meistens nicht gut mit Dokumentation/Tests abgedeckt Prüfen Versionshistorie, Dokumentation, Code		Typo3Scanner wird genutzt: Überblick über breaking changes und deprecated functions Oft kennt man die nötigen Migrationsschritte nur, wenn man schonmal ein Upgrade auf die Zielversion durchgeführt hat	
Zeitschätzung für Extensions	Zeitaufwand, wird mit den Informationen aus	Keine Werte an denen die	Zeitaufwand, wird mit den	Keine Werte an denen die	Keine festen Zahlen für die Zeitschätzung. Die

	einer Tabelle (siehe Protokoll) abgeleitet	Zeitschätzung festgemacht wird Zeitschätzung erfolgt nach Erfahrung (gibt es komplexe Datenbankabfragen, sind Schnittstellen vorhanden, …) Potenziell möglich: Eingruppierung nach Lines of Code, Anzahl der Abhängigkeiten, vorhandene Hooks	Informationen aus einer Tabelle (siehe Protokoll) abgeleitet Wenn Tec-Doku vorhanden -> geringere Zeitschätzung (Dokumentation bedeutet besseres Verständnis für die Extension)	Zeitschätzung festgemacht wird Treten breaking changes an mehreren Stellen auf, ist die erste aufwändig, die folgenden können schneller behoben werden (fast gleicher Lösungsansatz) Viele gleiche braking changes, benötigen im Zweifelsfall weniger Zeit beim Beheben als wenige verschiedene	Zeitschätzung erfolgt aus der Erfahrung.
Extensions, die nicht mehr verwendet werden sollen	Absprache mit den Kunden – Extension wird nicht mehr verwendet Deinstallierte Extensions im System Zu schlechte oder nichtkompatible Extensions müssen ersetzt werden	Wenn nicht verstanden wird wofür die Extension ist, dann entfernen; wenn sie wichtig war, wird sich der Kunde melden Wenn die Codequalität zu schlecht ist, wird die Extension entfernt, muss aber durch eine neue ersetzt werden (Aufwand hoffentlich kleiner)	Wenn eine Extension über einen gewissen Zeitraum nicht weiterentwickelt wurde Beispielfall: Upgrade TYPO3-Version 7.6.x auf 9.5.x von 2018, nicht älter als 8 Monate	(Man braucht Wissen über das System) Wenn Extensions in ähnlicher Form in den Core aufgenommen werden Feedback von Kunden oder der Redaktion, dass eine Extension schlecht ist oder nicht mehr verwendet wird	Im Installtool sichtbar, wenn die Extension nicht kompatibel ist oder Fehler hat. (muss aber ersetzt werden)

Weitere Faktoren die ein Upgrade verlängern	Anzahl der Sprünge der TYPO3-Version Testing 20% des Projektvolumens (müsste noch mehr sein) Gut wäre eine Metrik anhand der das Testvolumen erhöht wird (Anzahl schwierige Extensions, Anzahl TYPO3-Versionssprünge, ...)	Fehlendes Wissen beim Kunden, was dadurch nicht an die Entwickler gegeben werden kann Fehlende Erfahrungen bei den Entwicklern	Wenn kein Zugriff auf das System besteht Wenn während des Upgrades neue Erweiterungen am System vorgenommen werden (beides sollte nicht gemacht werden)	Interne Absprachen (je mehr Teams miteinander arbeiten, desto mehr benötigt) Onboarding-Zeit Kommunikation Projektleitung	Testen Zeit für die Abnahme des Kunden (Testen und eventuelle Schulungen)
Extensions, die immer länger dauern beim Upgrade	Meistens werde die systemspezifischen Extensions zum Problem Unterschiede bei der Pflege von Elementen zwischen den Versionen	Meistens werden schwierige Extensions auch in zukünftigen Upgrades schwierig Library-Sammlungen können komplex werden Extensions, die ein komplexes Thema unterkomplex darstellen wollen	Nicht bekannt Fehlende Auswertungen auf Extension-Ebene nach dem Upgrade	Eigentlich nicht multicolumn	Grid_elements ist nicht so gut zu verwenden (persönliche Meinung) Mittlerweile nicht mehr (vorausgesetzt man hat eine halbwegs aktuelle Version)
Was würdest du anders machen, wenn du manche Upgrades nochmal machen könntest	Höhere Zeiten schätzen Höhere TYPO3-Versionssprünge als 1 benötigen mehr Zeit	Große Mengen (z.B. Templates, SeitenTypen) wurden unterschätzt	Wurde bereits gemacht In jeder Iteration von Upgrade-Analysen wurde Feedback	nichts	Es gibt immer Raum für Verbesserungen, aber nichts Konkretes Technologisch aktuell bleiben: prüfen welche

	Größere Sprünge in den Extension-Versionen sollten berücksichtig werden: Eingruppierung in höhere Version und das Testvolumen sollte erhöht werden	Varnish und Solr erhöhen die Komplexität eines Systems Risiko wird größer, wenn der TYPO3-Versionssprung größer als 1 ist Jeweils mehr Zeit	eingearbeitet. Das hat zu dem aktuellen Stand geführt		Tools es gibt, um die Analysezeit zu reduzieren und die Qualität zu steigern
Regeln für nächste Upgrades	TYPO3-Versionssprung > 1: Produktvolumen * 1,5 Extension-Versionssprung > 1: + Extension-Zeitaufwand * 0,2 je Versionssprung; weiterhin sollte diese Extension durch einen Entwickler betrachtet werden Extra Prüfung durch Entwickler wenn: Extension-Versionssprung > 1; >20% der Changes einer Extension als breaking markiert wurden	Schätzungen können nur für bereits vorhanden Komponenten gemacht werden	Die menschliche Komponente ist wichtig	Die Entwickler, die das Upgrade durchführen, sollten auch an der Analyse beteiligt sein	Man benötigt mehr Aufwand, wenn die TYPO3 Versionen (Start und Ziel) weiter auseinander sind Mach eine gute Analyse Drauf achten, wenn sich der Aufbau von Elementen geändert hat
Anforderungen an ein Backendmodul	Übersicht über die installierten	Auskunft über Codequalität	Übersicht über vorhandene	Zusätzlich zu den Systeminformationen,	Reports der Extensions (mögliche Fehler,

	Extensions: (Sortierung nach) Kategorisierung der jeweiligen Extension, Detaillink (Lines of Code, Typo3 scanner), vorhandene Extension-Versionen Systeminformationen: TYPO3 Version, PHP-Version, installierte PHP-Extensions (enthalten in der PHP-Info), Memory-Limit, Bildrendering, Content-Elemente, Benutzergruppen, registrierte Hooks Suchfunktion nach (deprecated/breaking) Funktionen CSV-, Pdf-Export Anpassungen der Zeitschätzungen der Extensions möglich (Entwicklerschätzung)	Je Extensions: Wenn viele Klassen, Templates oder Codezeilen enthalten sind Hooks, Abhängigkeiten Abdeckung durch Tests und Dokumentationen	Versionen einer Extension (aus TER, Gitlab, GitHub, …) Informationen der Bestandsaufnahme	Informationen über veraltete Datenbank-Connections Information ob RTE oder CKEditor Informationen ob die Extensions für die Zielversion schon existieren (Existieren die Dritt-Extensions noch nicht, kann man keine Aussage machen) Typo3Scanner Informationen	deprecaded/breaking Code) Report Exportieren (Informationen sind immer gut. Je mehr je besser) Super wäre TypoScript Scanner
Zusätzliche Infos	Die bekannten Projekte, konnten nicht in der kalkulierten Zeit umgesetzt werden	Nichts zur Analyse	Nichts zur Analyse	Wichtig Kenntnisse über das System zu haben Teilung in der Vorlage zwischen mit/ohne Tec-Doku ist nicht	Nichts

	Kategorien müssen noch verfeinert werden Wenn keine Serverinformationen vorhanden Risikofaktor: + 0,25 * Projektvolumen Zusammenspiel der Abteilungen (IT, Projektleitung, Testing) Wenn automatisierte Tests vorhanden, könnte man diese im Backend in der Extensionconfiguration hinterlegen mit einem Zeitfaktor und einkalkulieren			realistisch, da man in bekannten Systemen auch ohne Dokumentation nicht mehr Zeit benötigt Vorhandene Produktkenntnis ist ein Faktor, der bis jetzt nicht berücksichtigt wird Wenn neue Architekturen, Hilfsmittel/Software eingesetzt/entwickelt werden soll, muss jeweils genug Zeit eingeplant werden	

Tabelle Anhang 1 - Gegenüberstellung der Protokolle

Auflistung der verwendeten Klassen von Vladimir Falcon Piva

- Classes/DataProvider/DirectoryDataProvider.php
- Classes/DataProvider/ScanDirectoryDataProviderInterface.php
- Classes/DataProvider/Typo3ExtensionsDataProvider.php
- Classes/Service/ExtensionScanService.php
- Classes/Service/LinesOfCodeService.php
- Classes/Service/PhpCsScanService.php
- Classes/Service/ScannerService.php
- Classes/Utility/Utility.php