

Studie zur Gestaltung einer „Plattform as a Service“ zur leichtgewichtigen Vorgangsbearbeitung

Gemeinsames Ergebnisdokument

v1.0

Autoren

Richard Hudson

Tim Pommerening

Sven Praum

Thomas Rheil

Mitgeltende Dokumente

Kriterienkatalog zur Bewertung

Bezugnehmende Dokumente

Fachkonzept Förderung Energiesparmaßnahmen

Inhaltsverzeichnis

1 Überblick.....	3
1.1 Motivation.....	3
1.2 Aufbau der Studie.....	3
1.3 Prüfobjekt.....	3
1.3.1 Zustands- und Datenmodell.....	4
1.3.2 Use Cases.....	5
1.4 Bewertung.....	8
2 Gesamtfazit.....	9
3 Erläuterungen zum Kriterienkatalog.....	11
3.1 Nuclos.....	11
3.1.1 Kriterien.....	12
3.1.2 Fazit.....	20
3.2 ProcessMaker.....	21
3.2.1 Kriterien.....	24
3.2.2 Fazit.....	43
3.3 SuiteCRM.....	44
3.3.1 Kriterien.....	44
3.3.2 Fazit.....	48

1 Überblick

1.1 Motivation

In Unternehmen und Behörden entsteht immer wieder der Bedarf für einfache Fachverfahren zur Vorgangsbearbeitung. Diese zeichnen sich aus durch eine kleine Nutzerzahl, geringe fachliche Komplexität, einfache Datenspeicherung und Unterstützung für Ad-Hoc-Workflows. Dagegen haben sie in der Regel keine außergewöhnlichen Qualitätsanforderungen und keine oder geringe Integration mit anderen IT-Systemen. Relevant ist jedoch die Möglichkeit der Trennung in fachliche Konfiguration durch Fachanwender und Betreiber der Lösung im Rechenzentrum.

Für diese Zwecke ist klassische Individualsoftware unwirtschaftlich, da die oben genannten Features durch vielen Standardlösungen abgedeckt werden.

Aus diesem Grund haben die *Landeshauptstadt München* gemeinsam mit der *Fa. msg systems ag* eine Studie mit dem Ziel durchgeführt, passend scheinende Standardlösungen auf ihre Tauglichkeit hin zu untersuchen.

1.2 Aufbau der Studie

Im Rahmen dieser Studie haben die Autoren 3 Standardlösungen betrachtet, von der jede einen anderen Fokus hat.

- | | | |
|----------------------------|---|--------------|
| • Nuclos v4.3 | Enterprise Ressource Planning | (ERP) |
| • ProcessMaker v2.8 | Business Process Management | (BPM) |
| • SuiteCRM v7.1 | Customer Relationship Management | (CRM) |

Nuclos und *SugarCRM* (Vorgänger von *SuiteCRM*) sind bereits in unterschiedlichen Kontexten bei der Landeshauptstadt München im Einsatz. Ihr eigentlicher Schwerpunkt liegt jedoch nicht auf der Vorgangsbearbeitung. Deshalb wurde den beiden *ProcessMaker* gegenübergestellt. Eine Standardlösung, die explizit auf Vorgangsbearbeitung abzielt. Ein wichtiges Kriterium bei der Auswahl war, dass die Produkte Open Source sind. Wegen in neueren Versionen veränderter Lizenzpolitik wurde daher auch von *SugarCRM* auf den Open Source Ableger *SuiteCRM* gewechselt.

1.3 Prüfobjekt

Mit diesen Standardlösungen wurde prototypisch ein Verfahren zur *Förderung von Energiesparmaßnahmen* beim *Referat für Gesundheit und Umwelt* der *Landeshauptstadt München* umgesetzt. Das Verfahren wurde als typischer Kandidat einer leichtgewichtigen Vorgangsbearbeitung gewählt. Der Vorgang umfasst dabei den Prozess von der Anlage eines Förderantrags bis zur Erzeugung eines Bescheids.

1.3.1 Zustands- und Datenmodell

Die folgende Abbildung zeigt die Zustände, in denen sich der Vorgang befinden kann. Die Aufgabe der 3 Standardlösung war es, diesen Ablauf umzusetzen. Dabei sollte stets klar sein, in welchem Zustand sich ein Vorgang gerade befindet. Explizit nicht vorgegeben war, dass der Vorgang wie durch einen Assistenten, also streng geführt abgearbeitet werden muss. Wichtig dabei ist die Rollentrennung zwischen dem Zustand „Antrag erfasst“ und „Kurzprüfung erfolgt“. Hat der Vorgang einmal den Zustand „Kurzprüfung erfolgt“ erreicht, so ist eine andere Bearbeitergruppe zuständig und die bisherigen Bearbeiter dürfen den Vorgang nicht mehr ändern.

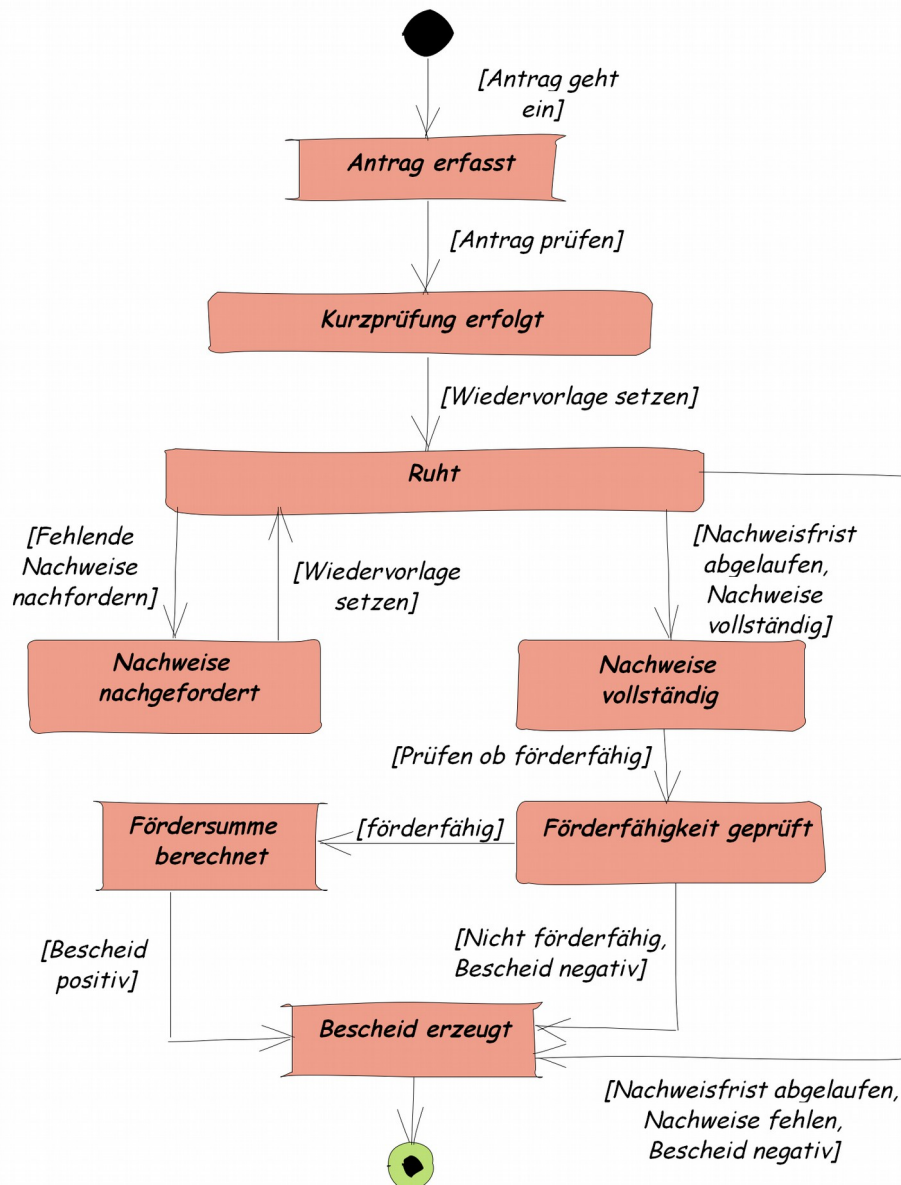


Abbildung 1: Zustandsmodell

1.3.2 Use Cases

Die im Folgenden beschriebenen Use Cases mussten umgesetzt werden können.

1.3.2.1 Antrag erfassen

Ein Mitarbeiter im Bauzentrum gibt die Daten ins System ein, die aus einem Förderantrag in Papierform entstammen. In diesem Zusammenhang wählt er einen dem System schon bekannten Antragsteller aus oder legt ihn neu an. Ebenso verfährt er mit dem Gebäude, für das der Antragsteller den Antrag gestellt hat. Den Antrag verknüpft er mit dem Antragsteller und dem Gebäude und kann eine eingescannte Version des Originals noch als Anhang hinzufügen. Dann übergibt der Mitarbeiter im Bauzentrum den Antrag an einen Sachbearbeiter.

1.3.2.2 Antragsteller pflegen

Mitarbeiter im Bauzentrum und ein Sachbearbeiter können Antragsteller pflegen. Dabei handelt es sich um typische Funktionalität zur Bearbeitung von Stammdaten (suchen, ändern, anlegen, löschen).

1.3.2.3 Gebäude pflegen

Mitarbeiter im Bauzentrum und ein Sachbearbeiter können Antragsteller pflegen. Dabei handelt es sich um typische Funktionalität zur Bearbeitung von Stammdaten (suchen, ändern, anlegen, löschen).

1.3.2.4 Antrag prüfen

Der Sachbearbeiter findet einen neuen Antrag in seiner Arbeitsliste. Er führt eine Kurzprüfung durch und korrigiert bei Bedarf Daten. Danach erstellt er aus dem System heraus eine Eingangsbestätigung. Das System hängt diese automatisch an den Vorgang an. Nun setzt der Sachbearbeiter den Vorgang auf Wiedervorlage.

1.3.2.5 Wiedervorlage

In der ca. einjährigen Ruhephase hat der Antragsteller die Möglichkeit, Belege für Energiesparmaßnahmen einzureichen. Ist der Wiedervorlagezeitpunkt erscheint der Vorgang wieder in der Arbeitsliste des Sachbearbeiters, der nun prüft, ob inzwischen alle zur Bewilligung einer Förderung notwendigen Belege eingegangen sind.

1.3.2.6 Nachweise nachfordern

Sind inzwischen nicht alle zur Bewilligung einer Förderung nötigen Belege eingegangen, so fordert der Sachbearbeiter diese beim Antragsteller durch ein Schriftstück an, das vom System erzeugt und auch direkt am Vorgang angehängt wird. Der Sachbearbeiter setzt den Vorgang erneut auf Wiedervorlage.

1.3.2.7 Förderfähigkeit prüfen

Sind alle Nachweise inzwischen vollständig, kann die eigentliche Förderfähigkeit geprüft werden. Ergibt die Prüfung, dass die Energiesparmaßnahme förderfähig ist, muss der Sachbearbeiter die Fördersumme berechnen.

1.3.2.8 Fördersumme berechnen

Der Sachbearbeiter trägt Ergebnisse ein, die er mit einem anderen System berechnet hat. Das betrachtete System summiert lediglich noch einige Eingaben.

1.3.2.9 Bescheid erzeugen

Am Ende des Vorgangs erzeugt der Sachbearbeiter immer einen Bescheid. Der Bescheid ist positiv, falls alle nötigen Belege vorliegen und die Förderfähigkeit gegeben ist. In dem Fall enthält er auch die berechneten Ergebnisse. Wie zuvor erzeugt das System den Bescheid und hängt ihn direkt wieder an den Vorgang an.

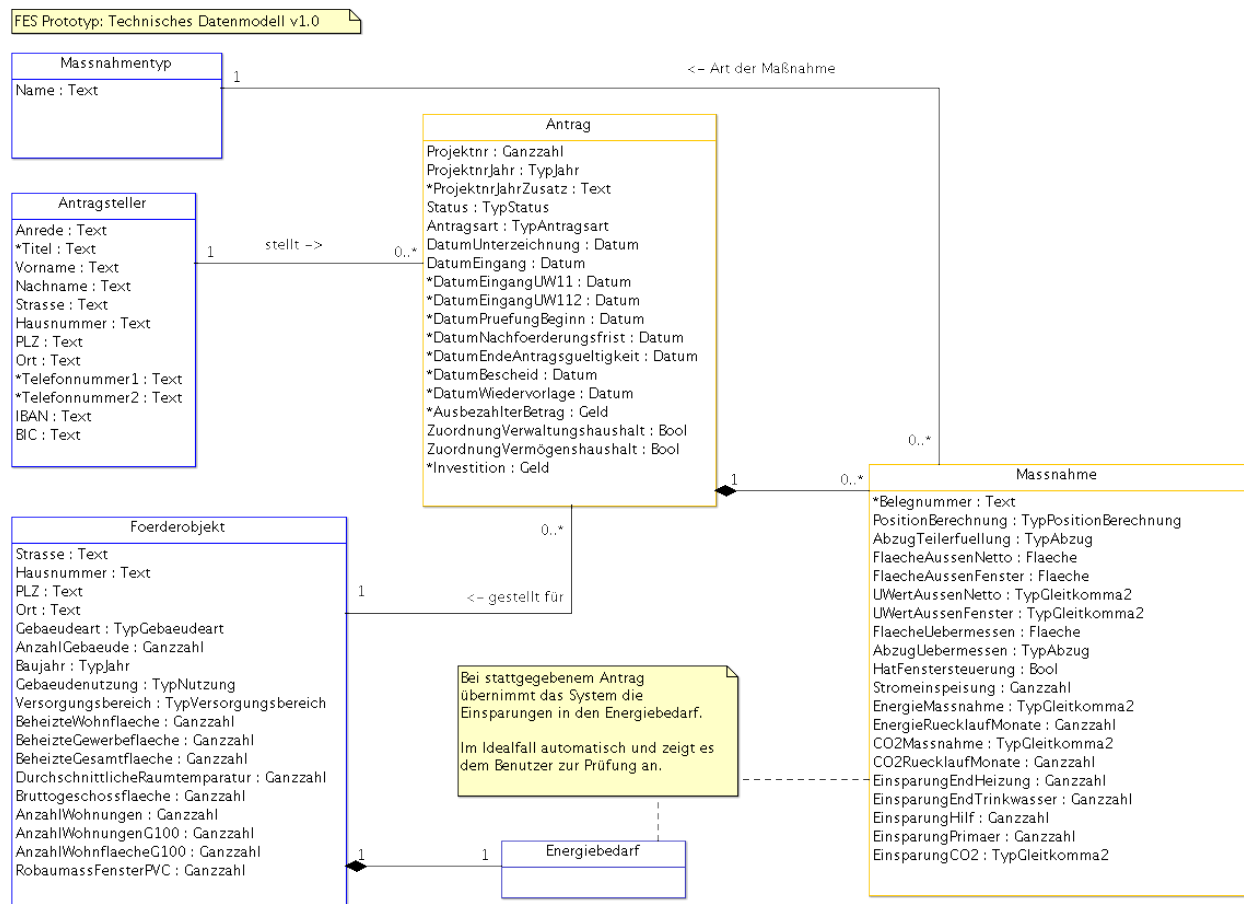


Abbildung 2: Datenmodell

Das Datenmodell des Prototyps unterscheidet sich in Stammdaten (blauer Rahmen) und Bewegungsdaten (gelber Rahmen). Der *Antrag* ist dabei das Vorgangsobjekt. D.h. der Antrag – von einem *Antragsteller* eingereicht – durchläuft den Prozess und hält den Zustands wie im Zustandsmodell beschrieben. Im Antrag können verschiedene *Maßnahmen* zur Energieeinsparung geprüft werden. Die Wirksamkeit der Maßnahmen senken *Energiebedarfe* der *Förderobjekte*. Solche Maßnahmen sind prinzipiell förderbar.

1.4 Bewertung

Die im Vorhinein festgelegten Bewertungskriterien entstammen den Anforderungen des *Fachkonzept Förderung von Energiesparmaßnahmen* und darüber hinaus gemeinsam festgelegten zusätzlichen Kriterien. Die Kriterien wurden in einem Kriterienkatalog gesammelt und in *Kategorien* gegliedert.

Pro Standardlösung wurde jedes Kriterium mit folgenden Punkten bewertet:

- **0** Nicht erfüllt.
- **1** Zum Teil erfüllt.
- **2** Vollständig erfüllt.

Im *Kriterienkatalog* können maximal 1000 Punkte erreicht werden. Die Punkte sind nach Schwerpunkten auf Kriterien verteilt. *Bewertungsschwerpunkte liegen beim Vorgangsmanagement (max. 185 Punkte), den Mitgeltenden Dokumenten (max. 130 Punkte) und den Betriebsaspekten (max. 180 Punkte).*

Jedes in eine Kategorie eingeteiltes Kriterium haben wir ebenfalls ungleichmäßig gewichtet, um auf die Maximalpunktzahl der jeweiligen Kategorie zu kommen. Auf diese Weise haben wir Aspekte hervorgehoben, die uns besonders wichtig waren.

Ein vollständig erfülltes Kriterium hat die volle Punktzahl bekommen. Ein zum Teil erfülltes Kriterium die halbe Punktzahl und ein nicht erfülltes Kriterium 0 Punkte.

Die 3 bewerteten Standardprodukte erreichten die folgenden Punktzahlen:

- **Nuclos** **887**
- **ProcessMaker** **858**
- **SuiteCRM** **851**

Sehr große Unterschiede zwischen den erreichten Gesamtpunkten existieren also nicht.

2 Gesamtfazit

Direkt vorweg: Mit jeder der drei betrachteten Standardlösungen war das Prüfobjekt *Förderung für Energiesparmaßnahmen* umsetzbar. Stärken und Schwächen zeigen die Werkzeuge je nach ihrer Herkunft.

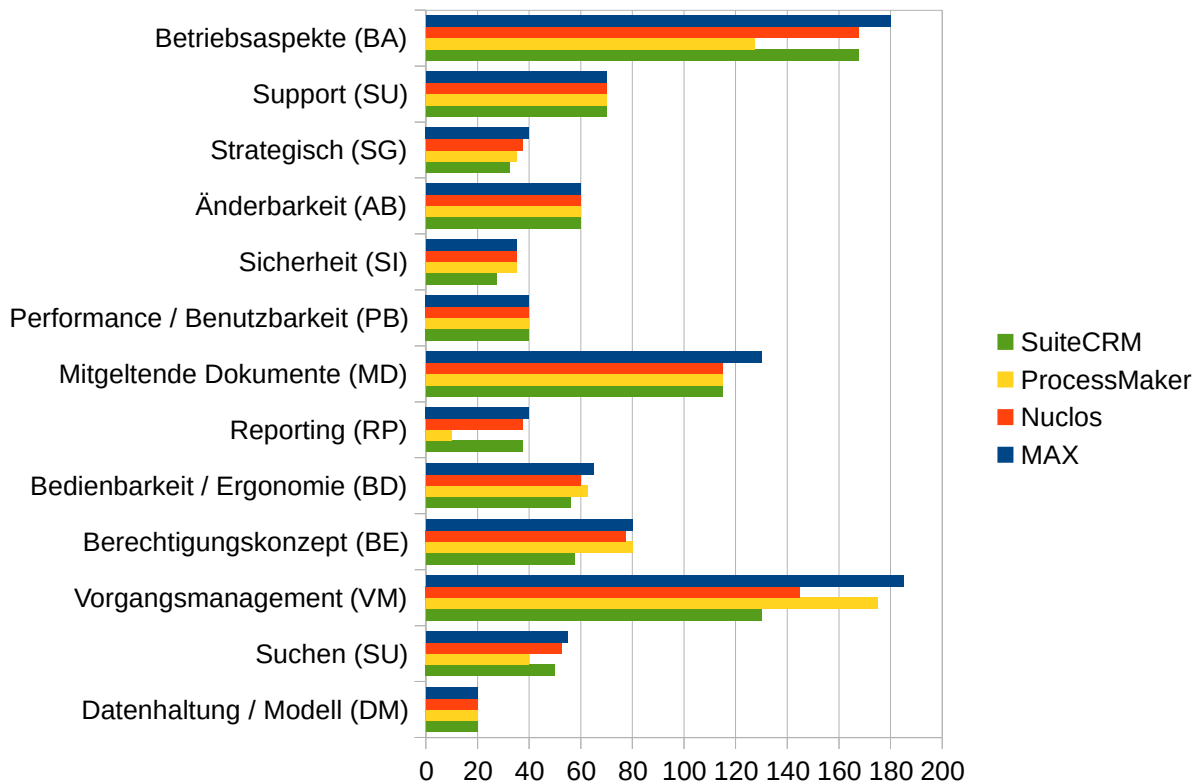


Abbildung 3: Gesamtergebnis: Vergleich der Kategorien

Nuclos (rot) ist der Allrounder. Die Lösung fällt im konkreten Testaufbau in keiner der betrachteten Kategorien gegenüber den anderen beiden zurück und sticht in keiner besonders heraus. Nuclos liegt in der Gesamtpunktzahl vorne. Bei den Betriebsaspekten besitzt es nach der Bewertung der Studie gegenüber den beiden anderen Lösungen einige Vorteile, wie die strikte Kapselung und Trennung von Anpassungen gegenüber dem Produkt, sowie die Integration in ein Versionskontrollsystem. Mit den anderen beiden Kandidaten ist eine Trennung der Konfigurationsartefakte ebenfalls realisierbar, erfolgt aber manuell. Eine Unterstützung durch die Produkte selbst konnte hier nicht festgestellt werden. Nuclos ist als einziger Kandidat keine Webanwendung, was der IT-Strategie mancher Kunden entgegenstehen dürfte. Als Java-WebStart Anwendung sind mit Nuclos gebaute Anwendungen allerdings über Betriebssysteme hinweg einsetzbar und auch zentral aktualisier- und verteilbar. Nuclos basiert auf einer REST-API. Für die nächste Hauptversion ist eine Webapp geplant.

ProcessMakers (gelb) Stärken liegen im Vorgangsmanagement, und das Produkt übernimmt im Verlauf der Studie dort die Führung. Beim Reporting hingegen fällt Processmaker mit Abstand hinter die anderen beiden zurück. Es sind lediglich Rohdatenexporte möglich, die allerdings in der Praxis mit einem Drittanbieter-Werkzeug ausgewertet werden können. Auch bei den Betriebsaspekten sahen die Studienteilnehmer Process Maker hinter den anderen beiden Produkten, da sie keine Auto-Updatefunktion bei Versionsupdates finden konnten. Werden Anwendungsfälle etwas komplexer, sind bei ProcessMaker schnell Programmierkenntnisse erforderlich. In der Gesamtwertung reiht sich ProcessMaker knapp vor SuiteCRM ein.

SuiteCRM (grün) hinkt ähnlich wie Nuclos im Testaufbau in fast keiner Kategorie den anderen Produkten deutlich hinterher, hebt sich gleichzeitig auch in keiner Kategorie positiv von den anderen Produkten ab. In den Kategorien Vorgangsmanagement und Berechtigungskonzept liegt das CRM-Tool in der Studie weit hinten, was sicherlich auf den Fokus des Produkts zurückzuführen ist. Eine Modellierung strikter Prozessabläufe etwa mit Benutzerwechseln ist nur mit Programmieraufwand möglich. Weniger strikte, offene Abläufe lassen sich mit SuiteCRM schnell und ohne Programmierung umsetzen.

Alle drei Tools können ihre jeweiligen Wurzeln aus den Bereichen ERP, BPM und CRM nicht verleugnen. Je nach Anwendungsfall eignet sich also das eine oder andere besser und die Tools sollten je nach ihrer Stärke für die jeweiligen Anwendungsfälle eingesetzt werden. Ohne Festlegung auf einen der Anwendungsfälle oder bei Mischformen, sollte Nuclos als der Allrounder zuerst betrachtet werden. Auch weil Nuclos sich wegen der direkt integrierten Versionskontrolle und der starken Kapselung der Anpassungen als guter Kandidat im Betrieb herausstellt, sofern die Strategie nicht eine Webanwendung fordert. Auch Prozesse sind abbildbar und darstellbar. Für kleine, strikte Prozesse wie das klassische Beispiel eines Urlaubsantrags wirkt der Start eines dedizierten Clients schon wieder zu schwergewichtig. Hier kommt ProcessMaker als Webanwendung ins Spiel, die solch simple interne Prozesse noch ohne Programmierung umsetzen kann, bei komplexeren Vorgängen jedoch schnell eine steile Lernkurve aufweist. SuiteCRM bietet sich bei etwas komplexeren Vorgängen an, die nicht strikt prozessorientiert Schritt für Schritt ablaufen müssen, sondern freier gestaltet sein können. In solchen Fällen, lassen sich Vorgänge einfach und ohne Programmierung umsetzen.

3 Erläuterungen zum Kriterienkatalog

Dieser Abschnitt beschreibt die Stärken und Schwächen der 3 ausgewählten Standardlösungen in Bezug auf die zuvor ausgewählten Kriterien. Die Beschreibung richtet sich dabei nach der Gliederung des Kriterienkatalogs. D.h. zu jeder Kategorie von Kriterien erklärt der folgende Abschnitt, ob und worin sich hier das jeweilige Standardprodukt besonders ausgezeichnet hat oder ob und warum es in dieser Kategorie nur schlecht punkten konnte.

3.1 Nuclos

Nuclos ist eine freie Enterprise-Resource-Planning-Software (ERP-Software), die der GNU Affero General Public License unterliegt. Es handelt sich bei Nuclos um einen Softwarebaukasten, mit dem Unternehmen ERP-Software erstellen können. Nuclos wird seit 2003 entwickelt. Seit Oktober 2009 ist Nuclos Open Source, seit Juli 2010 steht es zum Download zur Verfügung. Der Name Nuclos leitet sich ab aus einer Kombination der Begriffe Nucleus (lat. Kern) und dem Kürzel OS für Open Source. Nuclos wird von der Firma Novabit entwickelt, die seit Mitte 2010 Mitglied der Open Source Business Alliance ist.

Softwareerstellung mit Nuclos

Nuclos abstrahiert in seiner Eigenschaft als Softwarebaukasten von technischen Details, insbesondere auch von Programmiersprachen. Der Ersteller einer Businessapplikation in Nuclos benötigt im Gegensatz zum Softwareentwickler in anderen Systemen in der Regel keine Kenntnisse über Programmiersprachen mehr, um Geschäftsprozesse mit Nuclos abzubilden. Da ein Verständnis der Geschäftsprozesse typischerweise eine Kenntnis von Zusammenhängen im Unternehmen voraussetzt und oftmals auch eine Koordination von mehreren Beteiligten erfordert, sollte der Applikationsersteller in Nuclos über eine gewisse disziplinarische Befugnis oder Führungsverantwortung und einen umfassenderen Überblick über das Unternehmen verfügen.

Iterative Realisierung

Diese Art der Herangehensweise an Softwareentwicklung, bei welcher der Betroffene eines Geschäftsprozesses in die Lage versetzt wird, seinen Geschäftsprozess in Nuclos selbst zu definieren, erleichtert und unterstützt insbesondere eine iterative Umsetzung, da Kommunikations- und Abstimmungsbedarf zwischen Unternehmensfunktion und IT reduziert wird.

Einerseits entfällt der Aufwand für alle an der Umsetzung beteiligten Personen, Anforderungen in Form von Spezifikationen zu Papier zu bringen, die dann als Grundlage eines Verständnisses der Anforderungen durch den Softwareentwickler bzw. Programmierer dienen. Andererseits werden in der Mehrheit der Fälle schriftliche Spezifikationsunterlagen zum Bestandteil von Werkverträgen zwischen Unternehmen und Softwareentwicklern gemacht. Damit wird die nachträgliche Änderung von Anforderungen während der Projektlaufzeit und später zumindest erschwert und ist oft mit preislichen Aufschlägen verbunden. Diese entfallen, wenn eine direkte Umsetzung der Anforderungen durch die Unternehmensfunktionen selbst geschehen kann.

In diesem Zusammenhang hebt Nuclos die Trennung zwischen Design-Time (Entwurfszeit), Compile-Time (Übersetzungszeit) und Laufzeit auf. Die Erstellung von Businessapplikationen in Nuclos erfolgt innerhalb derselben Benutzeroberfläche wie die Nutzung der erstellten Applikation selbst. Eine Organisation des Applikationserstellungsprozess geschieht in Nuclos über eine entsprechende Benutzerrechtesteuerung.

Baukastenprinzip

Im Gegensatz zu proprietären ERP Systemen und anderen Open Source ERP Systemen geschieht die Realisierung von ERP (Enterprise Resource Planning) Software basierend auf Nuclos unter

Nutzung generischer Mechanismen, die allen Geschäftsprozessen und typischen Anforderungen an datenverarbeitende Systeme gemein sind. Für einen konkreten, individuellen Geschäftsprozess nimmt der Applikationsersteller in Nuclos eine Konfiguration bzw. Parametrisierung der generischen Mechanismen vor, um diese für die einzelnen Anwendungsfälle einzustellen. Er erstellt dabei Entitäten, Maskenlayouts, Gesamt- und Teilprozesse, Geschäftsregeln, Workflows und Reports und fügt diese wie Bausteine zusammen.

Nuclets

Die daraus resultierende Businessapplikation wird nicht zu einem festen Bestandteil von Nuclos, sondern bleibt ein von Nuclos getrenntes Softwareartefakt (sog. Nuclet), das sich nur aus Konfigurationsinhalten und Parametern zusammensetzt und damit unabhängig von einer Softwarekompilierung jederzeit zur Laufzeit änderbar ist. Die technische Struktur von Nuclos erzwingt eine strikte Trennung von Nuclos und den darauf aufsetzenden Nuclets. Dies ermöglicht insbesondere auch einen einfachen Austausch von Nuclets zwischen Unternehmen bzw. Anwendern.

Architektur

Nuclos ist Spring-basierend. Zum Einsatz kommt ein Apache Tomcat, grundsätzlich kann Nuclos jedoch auf beliebigen Java Servlet Containern betrieben werden. Entsprechend setzt Nuclos client- und serverseitig Java voraus und ist auf jedem Betriebssystem lauffähig, für das eine Java-Distribution existiert. Nuclos abstrahiert Datenbankzugriffe und kann daher prinzipiell auf beliebigen relationalen SQL-Datenbanken betrieben werden.

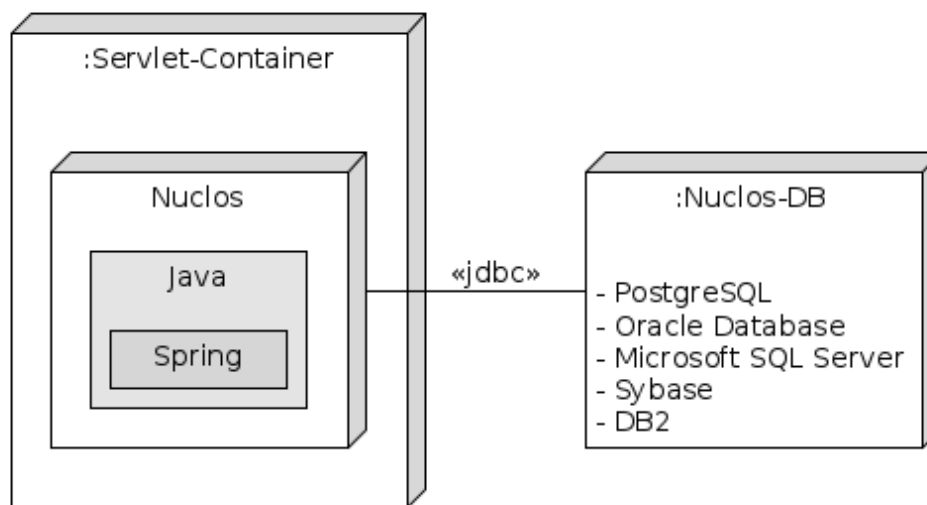


Abbildung 4: Architekturschaubild Nuclos

Quelle (Text): wikipedia (<https://de.wikipedia.org/wiki/Nuclos>)

3.1.1 Kriterien

3.1.1.1 Datenhaltung / Modell (DM)

Die beschriebenen Kriterien gehören zu den Basics eines ERP-Systems. Nuclos erfüllt erwartungsgemäß die Kriterien zu 100% mit folgenden Einschränkungen:

- Zu häufiges Löschen von Attributen eines Businessobjekt (BO) führte in wenigen Fällen dazu, dass sich das BO nicht mehr bearbeiten lies. Das reine Hinzufügen von Attributen ist unproblematisch
- In der untersuchten Version von Nuclos ließen sich nur flache BO-Strukturen importieren. Die im Wiki beschriebene Möglichkeit des XML-Imports konnte nicht erfolgreich getestet werden.

Besonders gut wurde die Historisierung und die Wiederherstellung von Datensätzen umgesetzt. Bei der Konfiguration eines BOs kann entschieden werden, ob für das Objekt eine Historie angelegt werden soll. Wenn diese Option gewählt wird, wird beim Speichern der vor der Änderung gültige Datensatz historisiert.

Über das „Extras“-Menü in der Detailansicht lässt sich die komplette Historie tabellarisch auflisten. Durch einen Doppelklick auf eine Zeile öffnet sich der zu diesem Zeitpunkt gültige Datensatz. Zwar erfolgt keine Gegenüberstellung der Werte aber die geänderten Felder sind farblich markiert. Soll der Datensatz wieder hergestellt werden, so geschieht das über das Speichern-Symbol in der Detailansicht.

¹ Ein Datenimport außerhalb von Nuclos ist m.E. zu empfehlen.

3.1.1.2 Suchen (SU)

Mit der Version 3.12 (aktuell 4.3.5) wurde die leistungsfähige Apache Lucene Suchtechnologie in Nuclos integriert. Sie ermöglicht eine benutzerfreundliche Volltextsuche in Daten und Dokumenten sowie die systemweite Suche nach bestimmten Begriffen.

Apache Lucene liefert qualitativ hochwertige Suchergebnisse und eine gute Systemperformance (Die Datenbank wird durch die Integration von Apache Lucene von komplexen Suchanfragen entlastet).

Folgende Eigenschaften übertreffen die Erwartungen:

- Komplexe Suchen sind durch die Kombination logischer Ausdrücke zusammenklickbar. Durch Rechtsklick kann auf Feldebene der Vergleichsoperator bestimmt werden.
- Getätigte Suchen lassen sich mit wenigen Klicks als Filter² speichern.
- Die Globale Suche erfolgt inkrementell über alle Objekte

Folgende Einschränkungen wurden festgestellt:

- Trotz der Verwendung von Apache Lucene ist eine Volltextsuche in Writer-Dokumenten nicht möglich (ob alle OpenOffice-Dokumente betroffen sind, wurde nicht untersucht)
- In den Suchfiltern lassen sich keine dynamischen Datumsangaben speichern. Es ist war möglich als Suchkriterium „HEUTE+14T“ (Heute in 14 Tagen) anzugeben, jedoch wandelt Nuclos dies in ein festes Datum um.

² Das Speichern von eingegebenen Suchkriterien wird in Nuclos als Filter bezeichnet. Sie sind benutzerspezifisch und dann von Vorteil, wenn häufig nach den gleichen Kriterien gesucht wird.

3.1.1.3 Vorgangsmanagement (VM)

In diesem Kriterienbereich ist Nuclos schwächer aufgestellt und erreicht nur gut Dreiviertel der Maximalpunktzahl. Ursächlich hierfür ist, dass Nuclos ein ERP-Baukasten und keine Workflow-Engine ist. So lassen sich Anforderungen wie „Automatisches Eskalationsmanagement“, „4-Augen-

Prinzip“, ... nicht durch 2 bis 3 Maus-Klicks konfigurieren, sondern müssen teilweise durch Programmierarbeiten (sogenannte Regeln) umgesetzt werden.

Nuclos bringt auch von Haus aus keine „gruppenübergreifenden“ Aufgaben mit. Sollen Aufgabenlisten wie zum Beispiel „offene Anträge“ erstellt werden, so geschieht dies über selbst definierte SQL-Abfragen. Im entsprechenden Konfigurationsmenü gibt es eine grafische Oberfläche mittels derer sich via Drag&Drop die betroffenen Tabellen zu einer SQL-Abfrage zusammenklicken lassen. Für komplexere Abfragen gibt es auf der gleichen Maske einen weiteren Reiter mit einem SQL-Editor. Mit diesem Editor lassen sich die generierten Abfragen ad hoc bearbeiten und testen, sowie die ursprünglich genierte SQL-Abfrage wieder herstellen.

Um z.B. Vorgänge einem bestimmten Benutzer zuordnen zu können muss zuerst aus den Systemtabellen eine Benutzerliste erstellt und dem BO zugeordnet sein. Anschließend muss das Objekt aus der Aufgabenliste durch Doppelklick geöffnet werden. Erst jetzt kann das BO einem Benutzer zugewiesen werden. Das Zuweisen von Objekten an Benutzergruppen erfolgt analog.

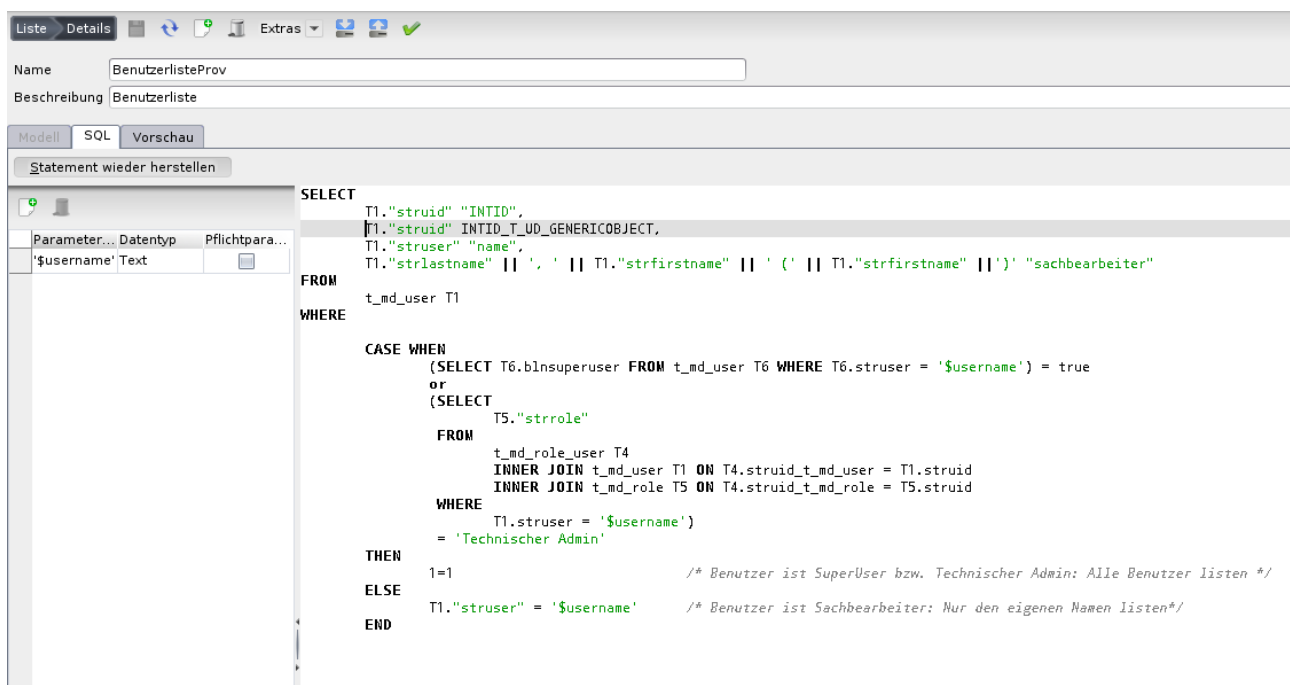


Abbildung 5: Benutzerliste als Valuelist Provider im SQL-Editor

Statt einer Workflow-Engine bietet Nuclos ein so genanntes Statusmodell. Businessobjekte lassen sich mit einem Statusmodell verknüpfen, dass sich in seiner Mächtigkeit mit einem UML Zustandsdiagramm (State Diagram) vergleichen lässt. Hierbei sind Bedingungen/Aktionen an den Zustandsübergängen immer programmatisch selber zu kodieren. Das System selber erlaubt per Konfiguration nur das Rechtemanagement, also z.B. wer einen Zustandswechsel überhaupt durchführen darf. Die Statuswechsel selber können dann entweder durch den Benutzer oder automatisch erfolgen.

Statusmodelle können auch dann noch geändert werden, wenn das Statusmodell bereits mit einem Businessobjekt verknüpft ist. Dabei verhindert das System automatisch das Erzeugen von „Zustandsleichen“, da nur solche Zustände gelöscht werden können die aktuell keinem Objekt zugeordnet sind. Ansonsten kann das Statusmodell auch nachträglich um weitere Zustände und Zustandsübergänge angereichert werden

Besonders elegant löst Nuclos die Nachvollziehbarkeit des Bearbeitungsstands eines Vorgangs bei Businessobjekten mit Statusmodell. Bei der Konfiguration des Statusmodells lassen sich Standardstatusübergänge definieren, die dem Benutzer in der Detailmaske angezeigt werden. Mit ei-

dem Klick auf einen Status, werden alle Statuswechsel und ggf. hinterlegte Statusregeln ausgeführt.

Abbildung 6: Statusliste in Detailmaske

3.1.1.4 Berechtigungskonzept (BE)

Nuclos besitzt ein ausgeklügeltes Berechtigungskonzept. Dabei wird das Minimalprinzip von Haus aus zu 100% umgesetzt (Neue Benutzer dürfen die Anwendung nicht starten).

Bei der Definition der Businessobjekte kann für jede Benutzergruppe konfiguriert werden, welche Rechte diese erhalten soll. Folgende Rechte können vergeben werden:

- keine
- Lesen
- Lesen/Schreiben
- Lesen/Schreiben/Löschen

Bei Businessobjekten mit Statusmodell kann o.g. Konfiguration für einzelne Felder definiert werden. Dazu müssen die einzelnen Felder sogenannten „Attributgruppen“ zugeordnet werden. Anschließend können in der Statusmodell-Konfiguration für jeden einzelnen Status folgende Rechte auf diese Attributgruppe festgelegt werden:

- keine Sichtrechte (Feld und Bezeichner verschwinden komplett aus der Maske)
- nur Sichtrechte (Feld wird auf readonly gesetzt)
- Sicht- und Schreibrechte

Die Rechte sind dabei für jede Benutzergruppe separat konfigurierbar. Gleiches gilt für den Statusübergang. Hier kann definiert werden, welche Benutzergruppe welchen Statusübergang ausführen darf. Auf diese Weise lässt sich z.B. sehr komfortabel ein Genehmigungsverfahren abbilden.

Der große Schwachpunkt bei diesem Konzept liegt allerdings im Berechtigungsmodul selbst. So lassen sich die Rechte für die Benutzerverwaltung nicht einschränken/erweitern. Der große Nachteil daran ist, dass Benutzer mit dem Zugriffsrecht auf die Benutzerverwaltung sich und alle anderen Benutzer zum sogenannten „Super-User“ machen können. Eine Rollentrennung „Fachlicher Admin“ vs. „root“ ist nur auf Vertrauensbasis möglich.

3.1.1.5 Bedienbarkeit / Ergonomie (BD)

Nuclos wird mithilfe der Java Web Start-Technologie gestartet und zeichnet sich durch folgende Charakteristika aus:

- Sorgt für eine schnelle Aktivierung von Anwendungen mit nur einem Mausklick
- Gewährleistet, dass Sie immer die neueste Version der Anwendung ausführen
- Macht komplizierte Installations- bzw. Upgradeverfahren überflüssig

Quelle: Oracle (https://www.java.com/de/download/faq/java_webstart.xml)

Nuclos ist keine Webanwendung.³ Die Oberfläche erinnert stark an Swing. Die Verwendung von Swing birgt 2 entscheidende Nachteile:

- Die Schriftgröße ist nur systemweit (durch Einstellungen im Betriebssystem) skalierbar.
- Das übliche Kontextmenü innerhalb von editierbaren Feldern in Webanwendungen (mit den Optionen Kopieren, Ausschneiden, Einfügen...) nach Rechtsklick ist nicht vorhanden und wurde auch nicht nachgebaut.

Diesen Nachteilen stehen folgende Vorteile entgegen:

- In Nuclos gibt es viele Maskengebundene Kontextmenüs. So lässt sich z.B. in der Auflistung eines BOs mit Statusmodell über einen Rechtsklick der nächste Status setzen.
- Das Layout lässt sich personalisieren. Das heißt jeder Benutzer kann für sich entscheiden welche Tabellenspalte an welcher Stelle stehen soll...

³ In der untersuchten Version 4.3.5 bringt Nuclos zwar eine alpha Version eines Webclients mit, jedoch wurde diese nicht genauer untersucht.

3.1.1.6 Reporting (RP)

Nuclos ist zwar kein Reporting-Tool, genügt aber den definierten Anforderungen voll und ganz. Ob ein zeitgesteuertes Erstellen von Reports möglich ist, konnte nicht abschließend geklärt werden. Einen entsprechenden Task Scheduler⁴ bringt Nuclos von Haus aus nicht mit. Diese Anforderung ließe sich aber ggf. eine Job-Regel realisieren. Dieser Lösungsansatz ist jedoch umständlich und mit einem gewissen Programmieraufwand verbunden.

⁴ Nuclos hat zwar einen Scheduler im Bauch, jedoch lassen sich mit diesem nur Regeln und Datenbankprozeduren ausführen.

3.1.1.7 Mitgeltende Dokumente (MD)

In dieser Kategorie zeigen sich Stärken und Schwächen. Während die Erstellung von Dokumenten ohne änderbaren Inhalt (PDF) mit iReport leicht von der Hand geht, so gestaltete sich die Erstellung von Dokumenten mit änderbaren Inhalten sehr schwierig.

Im Nuclos-Wiki ist die Erstellung von Dokumenten aus doc- bzw. docx-Vorlagen beschrieben. Die ersten Versuche mit docx-Vorlagen, die zum einen mit OpenOffice 3.2.x und zum anderen mit LibreOffice 4.1.6.x erstellt wurden, scheiterten. Es wurden zwar Dokumente generiert, jedoch wurde maximal eine Textmarke befüllt.

Noch unbefriedigender war das Ergebnis beim Erstellen eines Dokumentes aus einer doc-Vorlage. Hier wurde der Prozess systemseitig nach einer gewissen Zeit durch einen Timeout abgebrochen.

In weiteren Versuchen eine änderbare Vorlage zu Erstellen ist die Validierungsmeldung genauer untersucht wurden. Wird in der Reportkonfiguration eine Vorlage (z.B. doc) mit einem Ausgabeformat (z.B. docx) kombiniert, die nicht kompatibel sind, gibt Nuclos folgende Fehlermeldung aus:

„Es sind Validierungsfehler aufgetreten.

Bitte geben Sie für das Ausgabeformat DOCX eine MS Word Vorlage (.docx) an“

Daraufhin wurde eine docx-Vorlage mit MS Word 2003 erstellt. Mit dieser Vorlage konnte anschließend ein docx-Dokument mit befüllten Textmarken generiert und anschließend auch mit LibreOffice geöffnet werden. Das Verhalten von Nuclos beim Erstellen von doc-Dokumenten aus einer MS Word Vorlage ist identisch zu doc-Vorlagen, die mit anderen Office-Produkten erstellt wurden. Das Abbrechen durch einen Timeout liegt vermutlich am fehlenden MS Word.

Out of the box bietet Nuclos keine bequeme Möglichkeit einem Dokument bei der Erstellung einen Freitext hinzuzufügen. Um dies dennoch realisieren zu können wurden folgende Workarounds gefunden:

Download / Upload

Der Benutzer kann das erstellte Dokument öffnen, ändern und lokal speichern. Der Nachteil an diesem Workaround ist, dass das Dokument anschließend händisch wieder hochgeladen werden muss. Ein automatischer Upload⁵ wie er bei DMS-Systemen üblich ist, ist nicht vorhanden. Des Weiteren ist der Workaround für statische Dokumente (z.B. PDFs) nicht anwendbar.

Freitextfeld(er) im Businessobjekt

Eine Möglichkeit Freitexte automatisch in ein generiertes Dokument einzufügen ist das Definieren von Freitextfeldern im Businessobjekt selbst. Diese Felder müssen dem Benutzer auf der Maske zur Verfügung gestellt und in der Vorlage als Textmarke definiert werden. Bei der Erzeugung des Dokumentes werden diese Texte dann an der entsprechenden Stelle eingefügt. Der Nachteil an diesem Workaround ist, dass das Objekt vor dem Erstellen des Dokumentes gespeichert werden muss und somit die Inhalte der Freitextfelder dauerhaft persistiert sind. Ggf. lassen sich die Inhalte der Freitextfelder nach der Dokumenterstellung durch eine eigens erstellte Regel löschen. Diese Option wurde während der Studie jedoch nicht näher untersucht.

⁵ Ob und wie eine Schnittstelle zu einem DMS realisierbar ist, wurde nicht untersucht. Da aber bereits Schnittstellen zu Drittanbietern existieren, könnte sich der Autor gut vorstellen, dass eine solche Schnittstelle als Nuclet umsetzbar wäre.

3.1.1.8 Performance / Benutzbarkeit (PB)

Die vergebenen Punkte beruhen auf den Erfahrungswerten bereits bestehender mit Nuclos realisierter Anwendungen (z.B. bei der LHM: Grip).

Als Performancestellschraube bietet Nuclos im Businessobjekt-Wizzard die Möglichkeit an für einzelne Attribute einen Index anzulegen. Komplexere Indizes müssen jedoch über die Datenbank selbst erstellt werden.

3.1.1.9 Sicherheit (SI)

Alle Kriterien werden zu 100% erfüllt. Überraschende bzw. einschränkende Eigenschaften wurden nicht festgestellt.

3.1.1.10 Änderbarkeit (AB)

Alle Kriterien werden zu 100% erfüllt.

Reine Wertelisten wurden mit Version 4.x abgeschafft. Sie sind zwar im Konfigurationsmenü noch vorhanden, jedoch führt ihre Verwendung zu einem Verarbeitungsfehler beim Speichern des BO.

Sollen Wertelisten erstellt werden, so muss hierfür ein weiteres BO erstellt und anschließend dem BO, welches eine Wertelisten beinhalten soll, verknüpft werden. Die Verknüpfung erfolgt als sog. Referenzfeld (Fremdschlüsselbeziehung) und weist folgende Charakteristika auf:

- Nuclos geniert auf Wunsch einen Verwaltungsvorgang
- Änderung eines Wertes in der Werteliste wirkt sich auf die Werte im verknüpften BO aus
- Löschen eines Wertes in der Werteliste nur möglich, wenn keine Referenz zu einem BO vorhanden ist

Nuclos bringt ein leicht verständliches Schritt-für-Schritt-Wizzard zur Erstellung/Änderung von Businessobjekten und dessen Attributen mit. (Einfache) Änderungen am Datenmodell sind so sehr schnell umsetzbar.

3.1.1.11 Strategisch (SG)

Strategisch ist Nuclos sehr gut aufgestellt. Nuclos selbst ist in Java unter Verwendung diverser Third-Party-Framworks (u.a. Spring, Apache Lucene, ...) implementiert. Eigene Regeln werden in Java (Serverregeln) und Groovy (Clientregeln) implementiert.

Die Anwendung ist mit folgenden Datenbanken lauffähig:

- PostgreSQL Server (ab Version 8.4)
- Oracle Database Server (ab Version 10)
- Microsoft SQL Server (ab Version 2005)
- Sybase SQL Anywhere (ab Version 10)
- IBM DB2 Express-C (ab Version 10)

Der Administrator wird im Installations-Wizard gefragt, welche Datenbank verwendet werden soll. Ob und wie ein Datenbankwechsel nach der Installation stattfinden kann, wurde nicht untersucht.

In den folgenden Abschnitten wird auf einzelne Kriterien vertieft eingegangen.

Layouterstellung/Layouteditor

In Nuclos lassen sich Layouts über die Businessobjekt-Konfiguration generieren und es bringt einen eigenen grafischen Editor für die Erstellung des Layouts mit, jedoch offenbaren diese Funktionalitäten kleine Unschönheiten.

Layoutgenerierung

Bei der Layoutgenerierung hat der Administrator zwar die Option die einzelnen Felder anhand ihrer Attributgruppe zu gruppieren, jedoch hat er keine Handhabe über die genaue Anordnung (Reihenfolge, 2 Eingabefelder nebeneinander, ...). Nuclos schreibt die Felder in einer Art Tabellenform (links der Bezeichner, rechts das Eingabefeld) untereinander.

Layouterstellung im Editor

Nuclos verwendet für die Anordnung der Felder ein Gridlayout. Die einzelnen Felder lassen sich mittels drag and drop aus der „Palette“ in die gewünschte „Spalte“ ziehen. Es ist von Vorteil wenn der Administrator vor der Layouterstellung eine klare Vorstellung vom Layout hat, denn das nachträgliche Verschieben von Felder ist teilweise etwas knifflig.

Mandatenfähigkeit

In der untersuchten Version (4.3.2) ist die Mandatenfähigkeit von Nuclos eher rudimentär. Es lassen sich Mandaten erstellen und Benutzer können Mandaten zugeordnet werden. Weitergehende Mandatenfunktionalitäten konnten jedoch nicht festgestellt werden.

Nichts desto trotz wurde in dieser Kategorie die volle Punktzahl vergeben, da die Mandantenfähigkeit aller Voraussicht nach mit Version 4.5 umfangreich ausgebaut wird. Zukünftig soll unter anderem dem Benutzer für die jeweiligen Mandanten andere Benutzergruppen zu geordnet werden können.

Weitere Mandatenfeatures sind bereits im Nuclos-Wiki beschrieben. Siehe hierzu <http://wiki.nuclos.de/display/Konfiguration/Erweiterung+Mandantenwesen+mit+4.5>

Lizenz

Einen Punktabzug erhält Nuclos aufgrund dessen, dass es unter einer GNU Affero General Public License veröffentlicht ist. Das heißt, dass Erweiterungen und Verbesserungen an der Software der Öffentlichkeit zur Verfügung gestellt werden müssen. Davon sind nicht die erstellten Nuclets oder aber eigens programmierte Regeln betroffen.

3.1.1.12 Support (SU)

Alle Kriterien werden zu 100% erfüllt. Novabit selbst bietet kostenpflichtigen Support an.

3.1.1.13 Betriebsaspekte (BA)

Bis auf die Aspekte BA6, BA8 und B12 sind alle Kriterien zu 100% erfüllt. Im folgenden wird auf diese Aspekte näher eingegangen.

System unterstützt beim Test von Änderungen

Nuclos selbst unterstützt nicht beim Test von Änderungen. Auch die Internetrecherche zum Thema Testautomatisierung verlief ergebnislos. Ggf. ist hier eine Nachfrage bei Novabit zielführend.

Sobald der Web-Client ausgereift ist, lassen sich mithilfe von Selenium automatisierte Tests erstellen.

System bietet Debugger

Der nuclosinterne Regeleditor bietet die Option „Debug?“ an. Wird diese Option aktiviert, so lassen sich fehlerhafte Regeln speichern und Nuclets zuweisen. Es handelt sich hierbei aber nicht um einen echten Debugger. So lassen sich z.B. keine Breakpoints setzen, keine Variablen auf die watchlist setzen...

Novabit bietet jedoch ein kostenpflichtiges Eclipse-Plugin für Nuclos an. Mit der Verwendung dieses Plugins steht der Eclipse-Debugger zur Verfügung und es besteht die Möglichkeit Junit-Tests zu erstellen.

System erlaubt mehrere Fachanwendungen isoliert in einer Instanz

Die Umsetzung der Anforderung wäre technisch möglich, aber ist nicht praktikabel, da sie vom Installer nicht unterstützt wird. Das heißt, dass die Konfiguration/Wartung des Applikationsservers selbst gemacht werden muss. Dabei ist auch noch zu beachten, dass Nuclos Dokumente und Ressourcen im /data Verzeichnis ablegt. Wenn also mehrere Nuclos Instanzen über einen Applikationsserver gehandelt werden sollen, muss auch dafür gesorgt werden, dass unterschiedliche data-Verzeichnisse verwendet werden (gilt auch für das config und log

3.1.2 Fazit

Die Studie hat ergeben, dass es sich bei Nuclos um einen wahren Allrounder handelt. In allen Kategorien wurde eine ansehnliche Punktzahl erreicht. Selbst bei Kategorien, bei denen Workflow-Engines oder CRM-Systeme klar im Vorteil sind, musste sich Nuclos nur knapp geschlagen geben.

Dafür überzeugte Nuclos auf ganzer Linie in der Kategorie „Betriebsaspekte (BA)“ in puncto Releasemanagement. Die einzelnen Artefakte sind aufgrund ihrer xml-Struktur für jeden verständlich und Änderungen einfach nachvollziehbar.

Aufgrund ihres „Nuclet-Systems“ lassen sich Artefakte einfach Versionieren und in verschiedene Umgebungen übertragen. Im Nuclet Manager sind die einzelnen Nuclets und ihre Bestandteile gelistet. Mit wenigen Klicks lassen sich einzelne Artefakte zu einem Nuclet zusammenfügen.

Das Import-Wizard analysiert jedes Nuclet und zeigt eine Vorschau der DB- Schemaänderungen sowie der Nuclet-Parameter.

Ein weiteres großes Plus ist das sehr durchdachte und sauber umgesetzte Berechtigungskonzept. Angefangen mit der 100%igen Einhaltung des Minimalprinzips über die Rechtevergabe auf Vorgangsebene bis hin zur Einschränkung der Rechte auf Attributebene.

Die größten Schwächen hat Nuclos bei der Erstellung von Dokumenten mit änderbarem Inhalt. So können doc- und docx-Dokumente nur aus Vorlagen heraus erstellt werden, welche mit Microsoft Office erstellt wurden. Vorlagen, die mit anderen Textverarbeitungsprogrammen erstellt wurden, können nicht verarbeitet werden.

Weitere wesentliche Schwachpunkte hat Nuclos bei einigen Anforderungen aus dem Komplex Betriebsaspekte. Hier ist wohl der größte Schwachpunkt, dass es empfehlenswert ist jede Nuclos-Installation auf einer eigenen Tomcat-Instanz laufen zu lassen. Dies bläht die Unternehmensinfrastruktur im Fall von vielen ähnlichen Nuclosanwendungen extrem auf.

Trotz weniger Schwachpunkte ist Nuclos in puncto „leichtgewichtige Vorgangsbearbeitung“ ein absolut konkurrenzfähiges Produkt. Entgegen der anfänglichen Antipathie und diversen „Schimpfattacken“ möchte der Autor festhalten, dass „je tiefer man in Nuclos einsteigt um so mehr muss man feststellen, dass das Tool eine ganze Menge mitbringt und 'gar nicht mal so schlecht ist'“.

Unter dem Gesichtspunkt der untersuchten Aspekte eignet sich Nuclos besonders gut für folgende Systeme:

- Bestellsysteme
- (Lager-) Verwaltungssysteme
- „Antrag-Genehmigungsverfahren“ mit Standardprozessen wie z.b. FES

Bei folgenden Anforderungen würde der Autor von einer Verwendung von Nuclos abraten:

- (mehrstufige) Genehmigungsverfahren mit Eskalationsmanagement
- individuelle Fallbearbeitungen (mit individuellen Dokumenten)

3.2 ProcessMaker

ProcessMaker ist ein Produkt für Prozessverarbeitung. Es wird durch die US-amerikanische Firma Colosa Inc. betrieben; die Entwicklung findet verteilt zwischen den USA und Südamerika statt. Das Produkt ist open-source: Es werden drei kostenpflichtige Enterprise-Ausprägungen und eine kostenfreie Community Edition angeboten. Dieses Dokument beschreibt die kostenfreie Community Edition, die zwar die wichtigsten Features für leichtgewichtige Use-Cases enthält, gleichzeitig aber einige relevante Einschränkungen mit sich bringt (siehe 3.2.1.13 - Betriebsaspekte (BA)).

ProcessMaker läuft primär auf dem LAMP-Technologie-Stack (Linux, Apache, Mysql und PHP). Eine Installation auf Windows mit Apache, Mysql und PHP (WAMP) ist ebenfalls möglich, wobei die Windows-Version unter mangelhafter Performanz mit langsamen Antwortzeiten leidet und eher für Demonstrations- und Entwicklungszwecke geeignet ist (siehe 3.2.1.8 - Performance / Benutzbarkeit (PB)).

ProcessMaker hat mit zahlreichen anderen Tools gemeinsam, dass Geschäftsprozesse mit einem graphischen BPM-Editor modelliert und anschließend die einzelnen Prozess-Tasks ausentwickelt werden. Das Besondere an ProcessMaker ist, dass die Inhalte der Prozess-Tasks auf standardisierten Formulardialogen basieren, die optional um Javascript- und PHP-Code-Snippets erweiterbar sind. (Das BPM-Tool Bizagi beinhaltet zwar auch ähnliche Funktionalität; hier wird aber keine kostenfreie Version angeboten.) Das Produkt ist zwar weniger flexibel, als herkömmliche BPM-Tools. Soweit es sich aber um solche Geschäftsprozesse handelt, die formularbasiert realisiert werden können, lässt sich selbst ein komplexes Prozessdesign schnell zu einer lauffähigen Anwendung ausarbeiten.

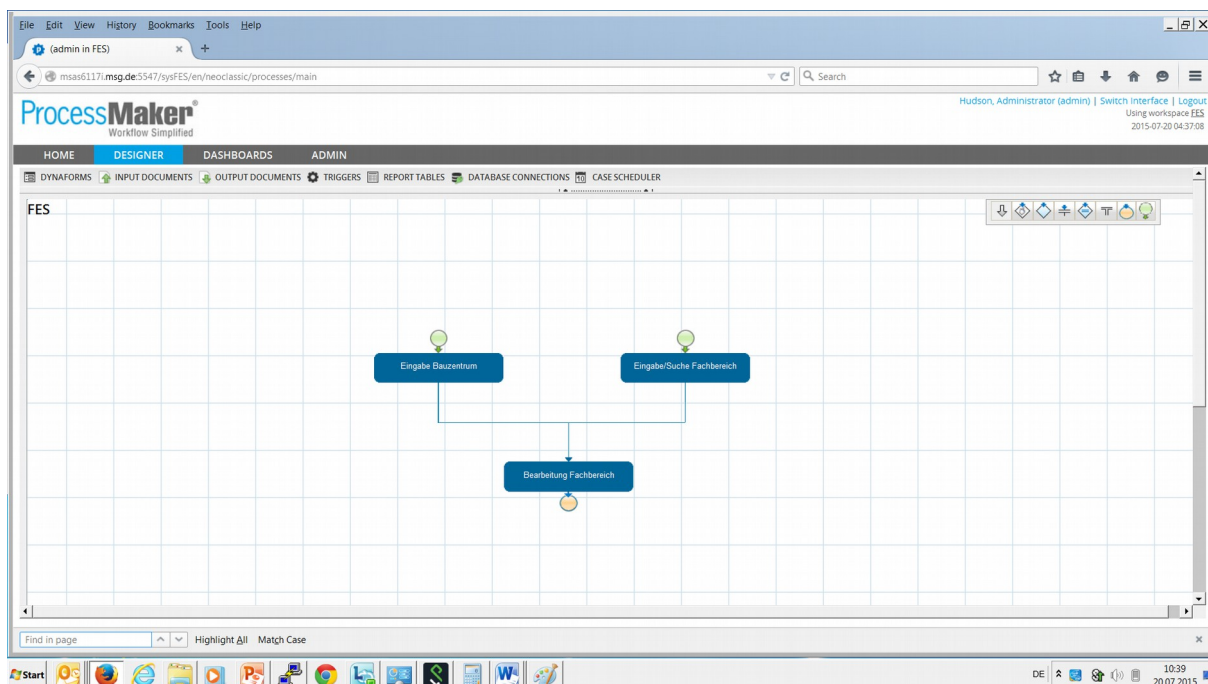


Abbildung 7: ProcessMaker Prozesssicht

Wie in anderen BPM-Tools wird der Kontrollfluss zwischen den einzelnen Prozess-Tasks definiert und die Rechte konfiguriert, die zur Ausführung eines jeden Prozess-Tasks nötig sind. Bei ProcessMaker besteht jeder dieser Prozess-Tasks aus einem oder mehreren HTML-Formularen (im ProcessMaker **Dynaforms** genannt), die nacheinander angezeigt werden.

Eine Variable, die in einem Formular definiert und gesetzt wurde, gilt fortan für die laufende Prozessinstanz (im ProcessMaker **Case** genannt) und darf in späteren Formularen bzw. Prozess-Tasks aufgegriffen, angezeigt oder zur Errechnung neuer Variablen verwendet werden. Die Dyna-

forms lassen sich in einem WYSIWYG-Editor erstellen und konfigurieren, so dass sich ein einfacher Use Case komplett realisieren lässt, ohne dafür Code schreiben zu müssen.

Bei komplexeren Anforderungen gibt es zwei Stellen, an denen sich Code in die Prozess-Tasks injizieren lässt. Einmal kann zu jedem Dynaform Javascript-Code definiert werden, die ausgeführt wird, wenn das Dynaform gerendert bzw. submitted wird; und einmal können vor oder nach der Anzeige jedes Dynaforms PHP-Code-Snippets ausgeführt werden (im ProcessMaker **Triggers** genannt).



Abbildung 8: ProcessMaker: Schematische Darstellung eines einfachen BPM-Tasks

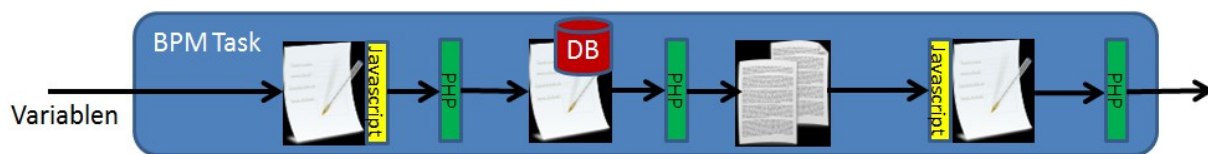


Abbildung 9: ProcessMaker: Schematische Darstellung eines komplexeren BPM-Tasks

Grundsätzlich darf in einem Trigger alles gemacht werden, was sich mit PHP realisieren lässt – auch Anrufe an externe Schnittstellen – wobei ein übermäßiger Einsatz dieser Möglichkeiten sicher darauf hinweisen würde, dass die Problemstellung zum Produkt nicht besonders gut passt. Üblicherweise sind die sinnvollen Einsatzgebiete für Trigger das Setzen, Verändern oder Löschen von Variablen und die Beeinflussung des Weges, den ein Case durch das Prozessdiagramm sowie durch die Formulare innerhalb der einzelnen Prozess-Tasks nimmt.

In Triggern stehen zwei Merkmale zusätzlich zur herkömmlichen PHP-Sprache zur Verfügung. Einmal können Case-Variablen über eine spezielle Syntax referenziert werden (siehe 2.2.1.1); und einmal können ProcessMaker-eigene-Methoden aufgerufen werden, wie beispielsweise `executeQuery()` um eine SQL-Abfrage abzusetzen.

Da mittels Trigger zwischen den Formularen innerhalb eines Prozess-Tasks hin- und hergesprungen werden kann, ergibt sich in vielen Fällen eine Designentscheidung bezüglich des logischen Pfades durch die Anwendung: inwieweit soll es auf der BPM-Ebene und inwieweit innerhalb der einzelnen Prozess-Tasks mit PHP realisiert werden? Soll eine Anwendung komplett ohne Programmierung auskommen, scheidet die PHP-Möglichkeit aus; wenn ohnehin programmiert werden muss, ist die Entscheidung etwas komplizierter. Im Pilotprojekt hat sich der Grundsatz bewährt, dass ein Prozess-Task die Funktionalität enthalten soll, die in der Regel von einem einzelnen Nutzer in einer einzelnen Sitzung verwendet wird. Meistens ist dann PHP-Code nötig, um zwischen den einzelnen Dynaforms im Prozess-Task hin- und herzuspringen und dadurch eine effiziente und benutzerfreundliche Dateneingabe zu ermöglichen.

ProcessMaker hat sich während des Pilots als stabiles Tool erwiesen. Die wenigen auftretenden Probleme mit der Plattform waren schnell reproduzierbar und mit passenden Workarounds lösbar. Es besteht zu ProcessMaker eine große Menge an Dokumentation, beantworteten Fragen und Hinweisen im Netz, so dass zu den meisten Herausforderungen schnell eine Antwort zu finden ist.

Problematisch sind aber einerseits das zum Teil überraschendes Verhalten der Plattform, sowohl bei der Ausführung von Prozessen als auch bei der Erstellung der Masken und PHP-Triggers, was eine relativ steile Lernkurve ergibt, wenn man zum ersten Mal mit ProcessMaker arbeitet; und andererseits die Tatsache, dass die Dokumentation, die als Wiki angeboten wird, keinerlei Qualitätssicherung unterliegt. So werden Javascript- und PHP-Code-Snippets dargestellt, die gar nicht

syntaktisch korrekt sind, und Lösungen zu Backend-Problemen angeboten, die zwar mit einem Nutzer gut funktionieren würden, gleichzeitig aber nicht threadsicher und für Race-Conditions anfällig sind. Zusammenfassend findet man schnell Antworten zu den meisten Fragen in der Dokumentation, braucht aber selber das Wissen, um passende von unpassenden Beiträgen auseinanderzuhalten.

Installation

Eine **für den produktiven Betrieb geeignete Installation** von ProcessMaker setzt vorhergehende, unabhängige Installationen von Apache, PHP und MySQL voraus. Dies erlaubt zwar dem Nutzer die Kontrolle über die Konfiguration dieser Technologien, die häufig bereits für andere Produkte auf dem gleichen Server im Einsatz sein werden, bedeutet aber auch, dass sie getrennt installiert und konfiguriert werden müssen. Für Apache und PHP sind außerdem verschiedene vorinstallierte Module vorgeschrieben, die je nach Betriebssystem bzw. Linux-Variante aus unterschiedlichen Installationspaketen zusammengesammelt werden müssen.

Die ProcessMaker-Installationsdokumentation ist aber detailliert und umfasst neben allgemeinen Anweisungen konkrete Angaben zu einzelnen Linux-Versionen wie OpenSUSE und Ubuntu, so dass die Installation auch der vorausgesetzten Technologien ohne größere Probleme möglich ist. ProcessMaker selbst lässt sich dann einfach aus einer Archivdatei entpacken und nach einigen wenigen Konfigurationsgriffen ausführen. Wenn die Applikation zum ersten Mal ausgeführt wird, erscheint ein ‚Pre-Installation Check‘, der sämtliche Voraussetzungen prüft.

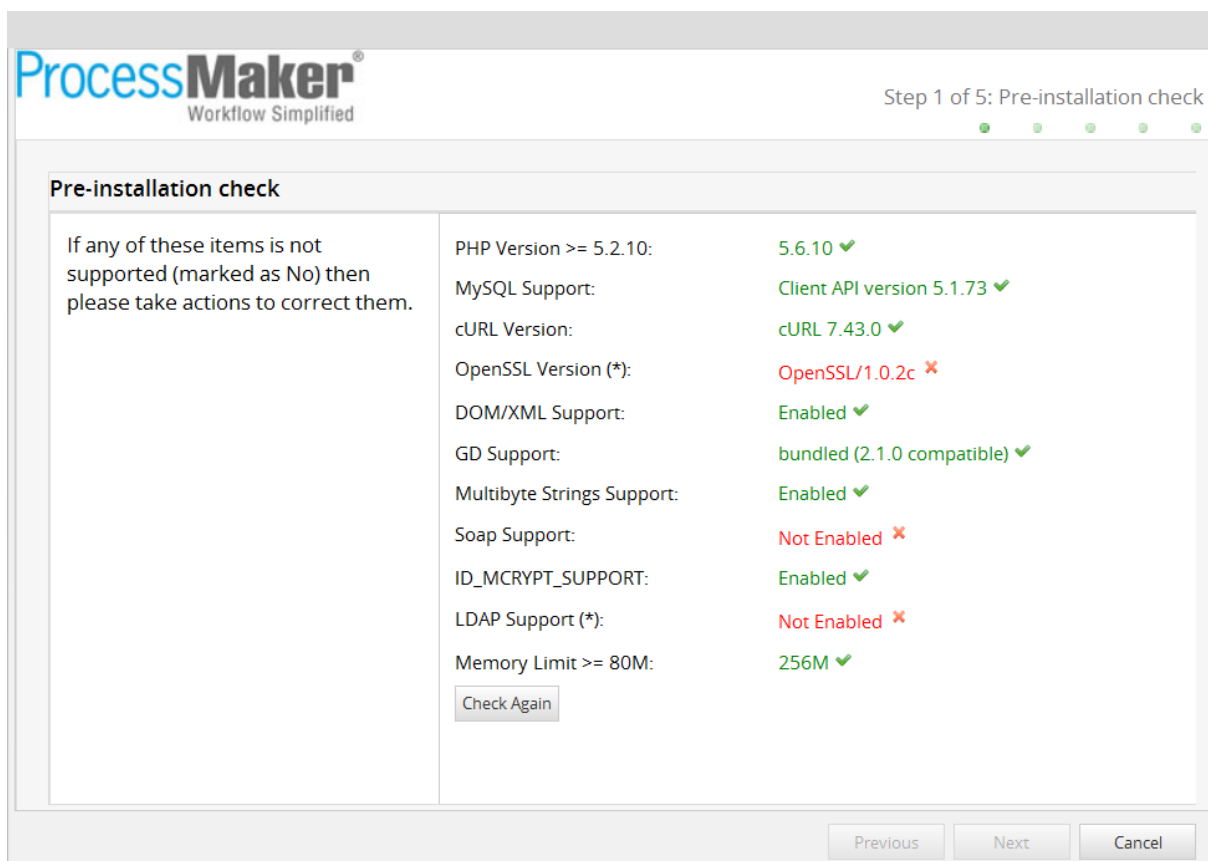


Abbildung 10: ProcessMaker Pre-Installation Check

Zu **Demonstrations- und Entwicklungszwecken unter Windows** gibt es ein umfassendes Installationspaket, das die vorausgesetzten Technologien mit installiert und für ProcessMaker konfiguriert. Dies ist eine äußerst benutzerfreundliche und komfortable Installationsmöglichkeit für Maschinen, auf denen die vorausgesetzten Technologien ausschließlich für ProcessMaker eingesetzt

werden. Im Produktionskontext wird von ihr aber ausdrücklich abgeraten (siehe http://wiki.processmaker.com/index.php/3.0/ProcessMaker_Windows_Installation), wobei dies in abgeschwächter Form auch grundsätzlich für den Betrieb unter Windows gilt.

Es gibt zwei Besonderheiten, die zwar nach der Installation beachtet werden müssen, die aber nicht hinreichend dokumentiert sind. Beide haben vermutlich nicht die nötige Aufmerksamkeit der Dokumentierenden in den USA erhalten, weil sie mit Internationalisierungsanforderungen zusammenhängen:

- Die Standard-Installation von ProcessMaker bietet ausschließlich eine englischsprachige Benutzeroberfläche an. Es werden eine Reihe von zusätzlichen Sprachen unterstützt, unter anderem Deutsch, für die zusätzliche Sprachdateien aus dem Internet heruntergeladen und anschließend über die Admin-View der laufenden ProcessMaker-Installation hochgeladen werden können.

Hier ist unbedingt dafür Sorge zu tragen, dass der ProcessMaker-Nutzer Leserechte auf die Sprachdatei hat, was normalerweise nicht ohne Weiteres der Fall ist, da man meistens nicht als der ProcessMaker-Nutzer Downloads aus dem Internet durchführt. Ist dies nämlich nicht gegeben, bleibt zwar beim Versuch, die Sprachdatei hochzuladen, die Prozessbar aktiv. Der Hintergrundthread hat sich aber in Wirklichkeit gleich aufgehängt.

- Der Datei `install-directory/workflow/engine/config/env.ini` muss die Zeile
`time_zone = "Europe/Berlin"`

hinzugefügt werden, da ansonsten alle Uhrzeiten in New-York-Zeit angezeigt werden.

3.2.1 Kriterien

3.2.1.1 Datenhaltung / Modell (DM)

Case-Variablen

Case-Variablen werden hauptsächlich an zwei Stellen definiert und referenziert: implizit in Dynaforms und explizit in PHP-Triggern.

Der Feldname, der in einem Dynaform für ein GUI-Objekt, beispielsweise ein Textfeld, angegeben wird, entspricht der Case-Variable. (Dieser Name muss dem im GUI angezeigten Label nicht gleich sein.) Ist diese Case-Variable bereits definiert, bevor das Formular angezeigt wird, wird dem GUI-Objekt der bestehende Wert zugewiesen; und nach Abschicken des Formulars wird der Case-Variable der im Formular eingegebene Wert zugewiesen.

In PHP-Triggern werden Case-Variablen mit einer besonderen PHP-Syntax referenziert. Der Case-Variablenname wird mit einem Präfix versehen, der ihn zum einen als Case-Variable auszeichnet und zum anderen eine Typverwandlung angibt, die durchgeführt wird, bevor der Wert dem PHP-Interpreter übergeben wird:

@@variable:	wird in einen String verwandelt
@@%variable:	wird in einen Integer verwandelt
@@#variable:	wird in einen Float verwandelt
@@?variable:	wird mit <code>encodeURIComponent()</code> verwandelt
@\$variable:	wird für den Einsatz in einer SQL-Query vorbereitet, indem Single-Quotes escaped werden (d.h. ' wird \')
@=variable:	es findet keine Verwandlung statt

Leider können Case-Variablen ausschließlich direkt in Trigger-Definitionen verwendet werden: eine Verwendung in Funktionen funktioniert nicht. Case-Variablen können ansonsten wie vollwertige PHP-Variablen eingesetzt werden, so dass beispielsweise die syntaktischen Ausdrücke `&@@variable` (eine Referenz zu einer Case-Variable) sowie `@@@variable` (um Fehler zu unterdrücken) einwandfrei funktionieren. Dies ist aber innerhalb eines Strings nicht der Fall, was eine notwendige Einschränkung ist, um Kompatibilität mit allgemeinem PHP-Code zu gewährleisten:

```
$test1 = "Frühling";
@@test2 = "Sommer";
echo "Im $test1 und im @@test2"
ergibt als Output Im Frühling und im @@test2.
```

Diese Einschränkung schließt praktisch die Verwendung von Heredocs mit Case Variablen aus. Bei einer solchen Herausforderung empfiehlt es sich, die Case-Variablen in standardmäßige PHP-Variablen zu kopieren, die dann standardmäßig im Heredoc referenziert werden können (siehe auch 2.2.1.7):

```
$eingangsdatum = @@AN_DATUMEINGANG;
$foerdersnummer = @@AN_FOERDERNUMMER;
@@BR_TEXT = <<<Separator
wir bestätigen den Eingang ihres Antrags am $eingangsdatum.
Ihr Antrag wurde unter folgender Fördernummer registriert:
$foerdersnummer.
```

Die Bearbeitung der Anträge erfolgt durch das Referat für Gesundheit und Umwelt, Team "FES", E Mail: fes.rgu@muenchen.de, Nachrichten AB: 089/233-47754.

Separator;

Es gibt einige anderen Situationen, in denen Case-Variablen referenziert werden können: leider sind die Namenskonventionen alles andere als konsistent, was vermutlich auf die organische Art und Weise zurückzuführen ist, in die ProcessMaker als Open-Source-Projekt gewachsen ist:

- Es kann mittels einer Bedingung entschieden werden, ob ein GUI-Objekt angezeigt wird oder nicht. Es dürfen andere Case-Variablen in der Bedingung mit `@#variable` referenziert werden. Innerhalb der PHP-Trigger hat `@#variable` aber eine ganz andere Bedeutung.
- Bei der Erzeugung eines Dokumentes (siehe 2.2.1.7) werden Case-Variablen normalerweise mit `@#variable` referenziert; werden sie mit `@@variable` referenziert, wird der Wert in Gänsefüßchen ausgegeben.
- In der Feldlisten-Sicht eines Dynaforms (siehe 2.2.1.5) kann eine SQL-Query definiert werden, um den Wert einer Case-Variable zu setzen. Andere Case-Variablen dürfen in die Query eingebaut werden und werden mit `@@variable` referenziert. Hier findet ein automatisches Escaping von Single-Quotes statt, was in PHP-Triggers mit `@$variable` erreicht wird.

Somit ergibt sich ein Wirrwarr von Variablensemantik, die eher an die historische Linguistik erinnert als an ein durchgeplantes System:

	Keine Verwandlung	Verwandlung in String	Verwandlung in Integer	Verwandlung in Float	encodeURIComponent()	SQL Escaping	Anzeige in Gänsefüßchen
PHP-Trigger	@=	@@	@%	@#	@?	@\$	-
Anzeige- bedingung	@#	-	-	-	-	-	-
SQL-Query in Dynaform	-	-	-	-	-	@@	-
Output- Dokument	@#	-	-	-	-	-	@@

In Dynaforms dürfen in Anzeigebedingungen und SQL-Queries nur solche Case-Variablen referenziert werden, die Feldern im gleichen Dynaform entsprechen. Es ist zwar möglich, nicht sichtbare Felder zu definieren. Die Unsichtbarkeit wird aber dadurch erreicht, dass die Felder gerendert und anschließend gleich per Javascript entfernt werden, was je nach Browser manchmal sichtbar sein kann. Solange es keinen Sicherheitsgrund gibt, warum die Felddaten nicht an den Client weitergegeben werden sollten, ist dies zwar etwas gewöhnungsbedürftig, stellt aber kein praktisches Problem dar.

Die voreingestellten Optionen von Dropdowns und Listboxes werden grundsätzlich entweder hartkodiert oder über SQL-Queries geladen, obwohl es auch eine etwas gewundene Möglichkeit gibt, die Inhalte einer PHP-Array so abzuspeichern, dass sie über eine Pseudo-Datenbankverbindung abgerufen werden können (siehe

http://wiki.processmaker.com/index.php/2.0/Dropdown_Boxes_and_Listboxes). Eine SQL-Query, die die Daten für solche Objekte lädt, muss stets zwei Werte zurückgeben: der eine wird im GUI-Objekt angezeigt, der zweite, der für die Nutzdaten vorgesehen ist, wird als zusätzliche Case-Variable gesetzt, wenn der Wert ausgewählt wird. Die zweite Case-Variable enthält den normalen Feldnamen, die erste den normalen Feldnamen plus `_label`, z.B. `@variable` und `@variable_label`.

Zur SQL-Query in einem Dynaform kann eine SQL-Verbindung ausgewählt werden. Überraschenderweise entspricht der ProcessMaker-internen Datenbank, in denen PM-Tables abgelegt werden, die Auswahl (none).

The screenshot shows a 'Data' section with two main components: a 'Sql Connection' dropdown menu currently set to '(none)', and a 'Sql' text area containing the SQL query 'SELECT ID, WERT FROM PMT_ANREDE'. The text area has a scrollbar on the right side.

Abbildung 11: ProcessMaker: Angabe einer SQL-Query in einem Dynaform

Relationale Datenbanken

ProcessMaker kann auf alle gängigen relationalen Datenbanken zugreifen. Die Datenbankverbindungen werden im Designer-View definiert und gepflegt, in der auch Dynaforms und Triggers erstellt werden. Die Zuweisung eines Wertes **aus einer relationalen Datenbank an eine Case-Variable** kann entweder in einem Trigger oder als getrennte SQL-Query in der Feldlisten-Sicht eines Dynaforms (siehe 2.2.1.5) erfolgen. Die Dynaform-Option ist eine bequeme Möglichkeit, wenn nur einzelne Werte übertragen werden sollen. Wenn aber beispielsweise mehrere Werte aus einer Datenbankreihe in die Case-Variablen einer Prozessinstanz zu kopieren sind, macht es mehr Sinn, die Operation in einem Trigger auszuführen, da ansonsten für jede Variable eine getrennte SQL-

Query ausgeführt werden muss, was problematisch ist sowohl aus Performanz-Sicht als auch auf Grund der mangelnden Threadsicherheit.

Es besteht die Möglichkeit, sogenannte **PM-Tables** in ProcessMaker selbst – im Admin-View - zu definieren. Diese werden dann in der gleichen internen MySQL-Datenbank abgelegt, in der auch die Case-Variablen gemanagt werden, und zwar mit einem allgemein gültigen, definierbaren Präfix: der Default-Wert lautet PMT_, so dass eine Tabelle, die logisch NAME heißen würde, als PMT_NAME in der Datenbank abgelegt würde.

Der Einsatz von PM-Tables hat folgende Vorteile gegenüber einer getrennten relationalen Datenbank:

- ☒ Die Erstellung und Pflege der Datenbankstrukturen sowie der Daten ist benutzerfreundlich im Admin-View der ProcessMaker-Oberfläche möglich und somit auch von einem fachlichen Admin mit entsprechenden Berechtigungen.
- ☒ Während das Schreiben **aus Case-Variablen in externe relationale Datenbanken** ausschließlich über PHP-Triggern möglich ist, lassen sich spezielle PM-Dynaforms erstellen, die automatisch mit entsprechenden PM-Tabellen verlinkt sind. Hier werden Werte automatisch in die PM-Tabelle geschrieben, wenn das Formular submitted wird.

Es gibt aber auch folgende Nachteile:

- ☒ Separation of Concerns wird insofern verletzt, dass die meist kurzlebigen Daten zu den einzelnen Prozessinstanzen mit meist längerfristigen relationalen Daten in einem einzigen Store gespeichert werden. Inwiefern dies zutrifft und ob es ein tatsächliches Problem bildet, hängt stark vom Use Case und von den Daten ab.
- ☒ Nicht alle Datenbank-Features werden im PM-Dialog angeboten: es fehlen beispielsweise relationale Integrität sowie sekundäre Indizes. Solche Merkmale könnten zwar nachträglich direkt auf der MySQL-Datenbank definiert werden, was sich aber kaum empfiehlt, da dann das gleiche Objekt in zwei verschiedenen Stellen gewartet werden müsste. Dieser Nachteil relativiert sich dadurch, dass der Mangel an Indizes vor allem bei großen Datenmengen ein Problem darstellen würde. Diese sind im Kontext des leichtgewichtigen Einsatzzwecks aber eher untypisch.

Zu einer existierenden PM-Tabelle lässt sich ein gelinktes PM-Dynaform automatisch erzeugen: für jede Spalte in der PM-Tabelle wird je nach Typ ein entsprechendes GUI-Objekt angelegt (wo bei eine Boolean-Spalte ein Textfeld ergibt und nicht, wie es zu erwarten wäre, eine Checkbox). Die Werte eines PM-Dynaforms werden aus der entsprechenden Tabelle vor befüllt und Änderungen automatisch nach dem Submit in die Tabelle zurückgeschrieben.

Die PM-Dynaform-Funktionalität ist nur bedingt mit der Administrationsoberfläche der PM-Tabellen selbst integriert: nachträgliche Änderungen in der Tabellenstruktur führen dazu, dass die PM-Tabellenreferenzen stillschweigend aus der Dynaform-Definition entfernt werden, was zu tückischen Bugs führen kann; und die automatischen Verbindungen zwischen GUI-Objekten und Datenbankspalten können zwar in der XML-Ansicht eines Dynaforms nachträglich gepflegt werden, aber nicht in der benutzerfreundlicheren Feldansicht.

Sonderzeichen wie Umlaute in Mysql-Spaltennamen werden von ProcessMaker nicht unterstützt und führen zu einem internen PHP-Fehler, wenn man versucht, zu einer PM-Tabelle ein Dynaform zu erzeugen.

PM Table

Table Name (Auto Prefix "PMT"):

Description:

☒ Keep the records of the table

[Add Field](#) [Edit Field](#) [Remove Field](#)

Field Name	Field Label	Type	Size	Null	Primary Key	Auto Increment
AS_ID	AS_ID	BIGINT		No	Yes	Yes
ANREDE	ANREDE	CHAR	100	No	No	No
TITEL	TITEL	CHAR	100	Yes	No	No
VORNAME	VORNAME	CHAR	100	No	No	No
NACHNAME	NACHNAME	CHAR	100	No	No	No
STRASSE	STRASSE	CHAR	100	No	No	No

Abbildung 12: ProcessMaker: Administrationssicht einer PM-Tabelle

Bei der Definition einer PM-Table muss einen Primärschlüssel bestimmt werden, der auch als Integer mit auto-increment definiert werden kann. Hier entsteht schnell die Frage, wie ein entsprechendes PM-Dynaform zu konfigurieren ist, damit neue Einträge automatisch mit dem nächsten automatisch erzeugten Primärschlüssel abgespeichert werden. Dies lässt sich leider nur mit dem folgenden undokumentierten Workaround erreichen:

- 1) Beim Erzeugen der PM-Dynaform wird nach einer Variablen gefragt, deren Wert den Wert des Primärschlüssels bilden wird. Hier wird eine Variable eingegeben, die gleich heißt, wie die Primärschlüsselspalte der PM-Table.
- 2) Als Default-Wert für das entsprechende Feld in der Dynaform wird null eingegeben.
- 3) Da die ID meistens nicht angezeigt werden soll, wird das Feld auf unsichtbar gesetzt.

Mit dieser Technik wird die automatisch erzeugte ID **nicht** in die entsprechende Case-Variable kopiert; sie behält den Wert null. Wird sie im weiteren Prozess benötigt, muss sie zu einem späteren Zeitpunkt mit einer SQL-Query in einem PHP-Trigger erfragt werden.

3.2.1.2 Suchen (SU)

ProcessMaker bietet die Möglichkeit, laufende sowie abgeschlossene Prozessinstanzen aufzulisten. In der Version 2.8, die hier untersucht wurde, lassen sich die Ergebnisse ausschließlich nach Datum und Nutzer filtern. In der inzwischen verfügbaren Version 3.0 kommt eine vollwertige Volltextsuche hinzu, was eine extrem wichtige Verbesserung darstellt. Diese wird vom Open-Source-Produkt Sphinx realisiert und bietet fortgeschrittene Merkmale wie Wildcards und einen SQL-ähnlichen Suchsyntax.

Zu abgeschlossenen Prozessinstanzen gibt es verschiedene anzeigbare Informationen wie z.B. den Pfad, den eine Prozessinstanz durch das Prozessdiagramm genommen hat bzw. wer wann welche Feldwerte geändert hat. Während der Ausführung der Prozessinstanz verwendete Dynaforms werden mit den Werten der Case-Variablen angezeigt, die beim Abschluss der Prozessinstanz galten. In komplizierteren Fällen, wo ein Dynaform mehrfach für verschiedene Daten verwendet wurde, etwa um mehrere Einträge eines Arrays zu bearbeiten (siehe 3.2.1.5 - Bedienbarkeit / Ergonomie (BD)), führt dies leider zu keinem sinnvollen Ergebnis, da immer nur der Eintrag angezeigt werden kann, der zuletzt bearbeitet wurde.

Die Rechte zur Besichtigung der Informationen zu abgeschlossenen Prozessinstanzen werden getrennt nach Informationstyp vergeben. So ist es beispielsweise möglich, einer Nutzergruppe die Besichtigung der allgemeinen Prozessinformationen zu erlauben und gleichzeitig die Besichtigung der verwendeten Dynaforms zu verbieten.

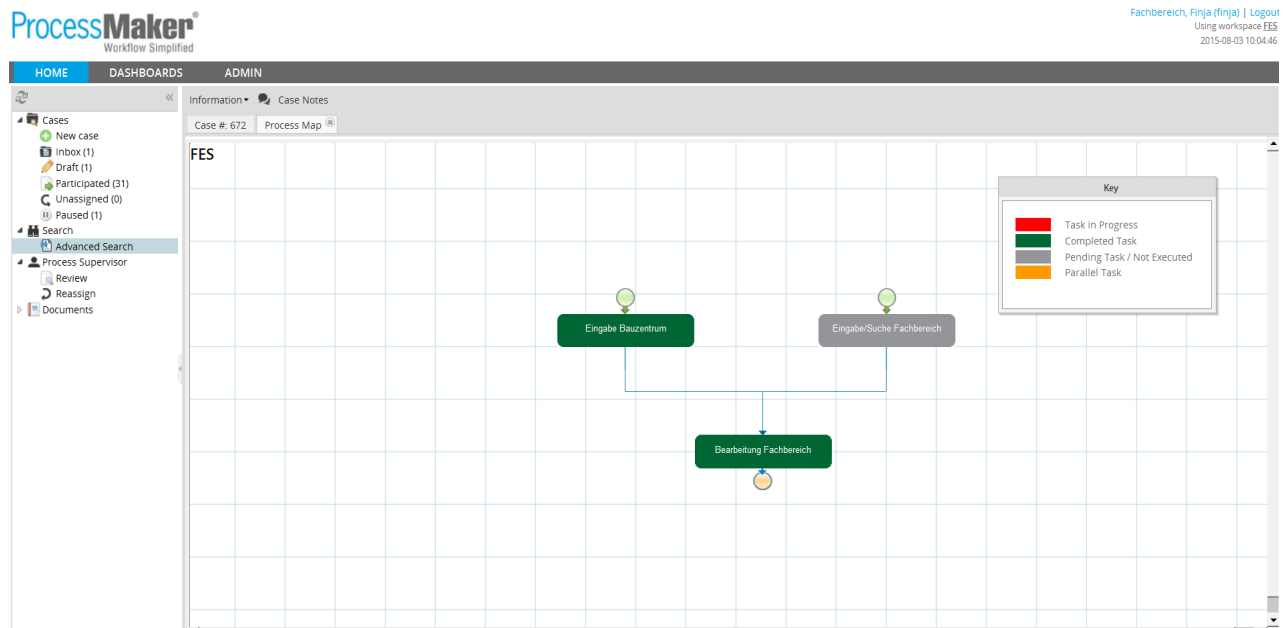


Abbildung 13: ProcessMaker: Sicht auf eine abgeschlossene Prozessinstanz

3.2.1.3 Vorgangsmanagement (VM)

In der Version 2.8, die hier untersucht wurde, war die Vorgangsmodellierung in ProcessMaker bereits sehr stark an BPMN angelehnt. Mit der neuen, bereits verfügbaren Version 3.0 ist nun eine offizielle Unterstützung des gesamten Feature-Sets von BPMN 2.0 erreicht.

Die Sicht eines einzelnen Nutzers enthält unter anderem einen Inbox mit Prozessinstanzen, die ihm zugewiesen wurden. Da typischerweise mehrere Nutzer die Berechtigung haben, einen Prozess-Task auszuführen, können im Designer-View verschiedene Regeln definiert werden, um zu bestimmen, welchem Nutzer ein gegebener Prozess-Task zugewiesen wird:

- Am einfachsten werden eingehende Tasks reihum zugeteilt.
- Eine explizite Zuteilung in PHP-Triggern ist auch möglich.
- Es existiert ebenfalls ein Self-Service-Modell, bei dem Nutzer sich selbst Aufgaben aus einem Pool zuweisen. Diese Variante ist etwas komplizierter, weil Regeln definiert werden müssen für den Fall, dass Aufgaben zu lange im Pool verweilen, ohne einem konkreten Bearbeiter zugewiesen zu werden.

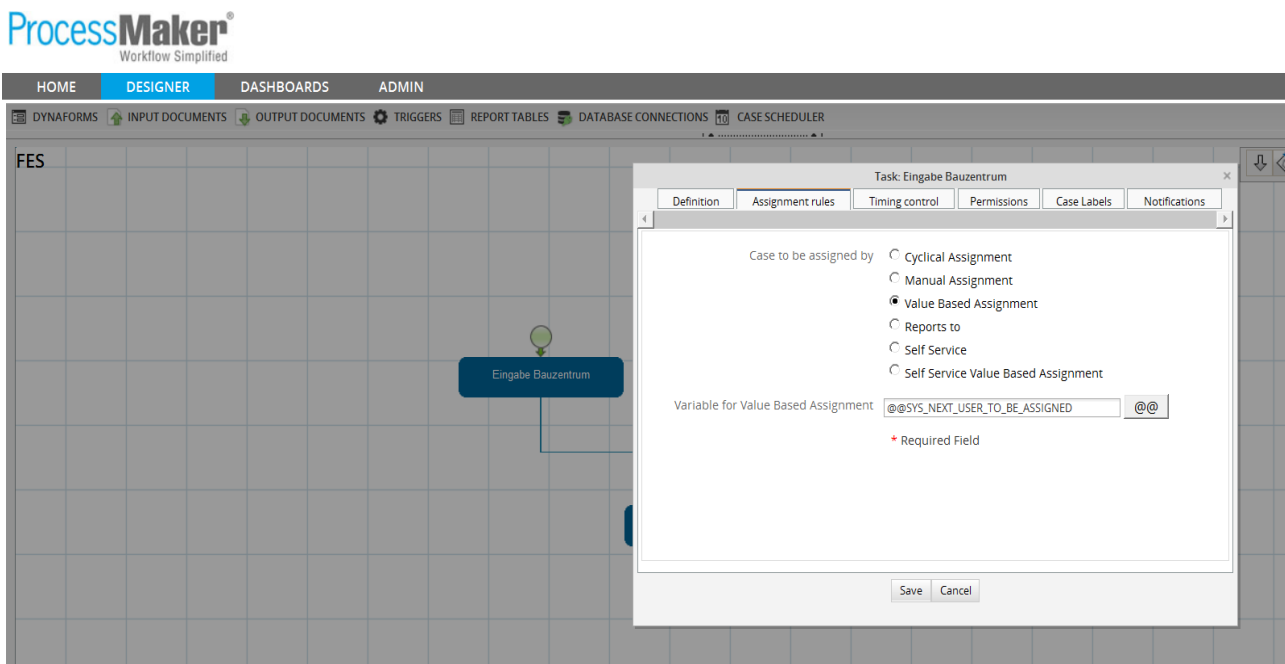


Abbildung 14: ProcessMaker: Einstellung der Zuweisungsregeln eines Tasks

ProcessMaker bringt eigene nützliche Zusatzfeatures um die Verwaltung von Prozessinstanzen mit:

- Ein Nutzer kann eine Prozessinstanz auf Wiedervorlage setzen; sie bleibt weiterhin sichtbar, aber unter einem anderen Tab, und erscheint erst dann wieder im Inbox, wenn das Wiedervorlagedatum erreicht wird.
- Einzelne Nutzer lassen sich als Supervisor berechtigen (siehe 3.2.1.4 - Berechtigungskonzept (BE)); diese können dann generell auf alle Prozessinstanzen lesend zugreifen, die sich gerade in der Bearbeitung durch andere Gruppenmitglieder befinden.
- Ein Nutzer mit entsprechenden Berechtigungen kann eine von ihm bearbeitete Vorgangsinstanz einem anderen Nutzer manuell weiterleiten. Diese Funktionalität eignet sich am besten dann, wenn die Weiterleitung aus fachlichen Gründen erfolgt, z.B. an einen fachlichen Experten.
- Für den Urlaubs- bzw. Krankheitsfall kann der Administrator einen Nutzer als Vertretung eines zweiten Nutzers definieren; der erste Nutzer sieht dann automatisch alle Aufgaben des zweiten Nutzers, wenn dessen Status von ‚Active‘ auf ‚Vacation‘ verändert wird.

Standardmäßig werden die Schritte in einem Task, die typischerweise aus Dynaforms bestehen, sequentiell abgearbeitet. Es besteht die Möglichkeit, Bedingungen im GUI zu definieren, die darüber entscheiden, ob ein gegebener Schritt samt seinen Triggern ausgeführt werden soll oder nicht.

Es gibt auch zwei ProcessMaker-eigene Methoden, mit denen der Kontrollfluss in PHP-Triggern beeinflusst werden kann: `PMFRRedirectToStep()`, der zu einem anderen Schritt im gleichen Task springt; und `PMFDerivateCase()`, der den Task beendet und dem Prozessmodell gemäß weitermacht. Bei `PMFDerivateCase()`, aber nicht bei `PMFRRedirectToStep()`, wird die neue Funktionalität in einem neuen Thread ausgeführt, so dass der bestehende Task mit `die()` beendet werden muss.

In beiden Fällen – `PMFRedirectToStep()` sowie `PMFDerivateCase()` – werden Änderungen, die im alten Schritt durchgeführt wurden, nicht zuverlässig resp. überhaupt nicht für den weiteren Verlauf sichtbar. Zumindest im Falle von `PMFDerivateCase()` ist dieses Verhalten laut der Dokumentation gewollt, bildet aber einen ernsthaften Mangel. Es gibt zwar eine Methode `PMFSendVariables()`, mit denen Werte explizit persistiert werden können. Hier muss aber jede Variable namentlich aufgeführt werden, was eine solche Stelle extrem brüchig macht: wird der Code zu einem späteren Zeitpunkt verändert, muss auch der `PMFSendVariables()`-Aufruf penibel aktuell gehalten werden. Um dieses Problem zu vermeiden, werden in der Praxis ‚Dummy‘-Schritte benötigt, deren Dynaforms nie angezeigt werden, aus denen aber mittels einem vorgelagerten PHP-Trigger woandershin gesprungen wird.

Insbesondere wenn der gleiche Trigger an verschiedenen Stellen eingesetzt wird, kann es nötig sein, zu ermitteln, in welchem Teil des Prozesses man sich gerade befindet. Es gibt verschiedene ProcessMaker-eigene Prozessvariablen, die Hilfe verschaffen können. Wie das folgende Beispiel zeigt, ist hier die `@@TASK`-Variable am hilfreichsten, die das GUID des gerade ausgeführten Tasks enthält:

```
switch (@@TASK)
{
    case "6159901155530d3363c1865064149660": // task 'Eingabe Bauzentrum'
        //look up all the users in the Fachbereich group
        $aUsers = executeQuery("SELECT USR_UID FROM GROUP_USER WHERE
GRP_UID = '506037340555f23bed2ad27041774420'");
        //get random number to select the user
        $noUser = rand(1, count($aUsers));
        @@SYS_NEXT_USER_TO_BE_ASSIGNED = $aUsers[$noUser]['USR_UID'];
        break;
    case "457057984555f08d9872cc6046386567": // task 'Eingabe / Suche
Fachbereich'
        @@SYS_NEXT_USER_TO_BE_ASSIGNED = @@USER_LOGGED;
        break;
}
```

Vorsicht ist mit der Variable `@%INDEX`, die aussagt, durch wie viele Tasks eine Prozessinstanz bereits gelaufen ist, weil auch das Setzen einer Prozessinstanz auf Wiedervorlage, was jederzeit passieren kann und außerhalb der Kontrolle des Prozessentwicklers liegt, dazu führt, dass `@%INDEX` inkrementiert wird.

Standardmäßig haben Dynaforms Pfeile oben links bzw. rechts, die auf das vorige bzw. das nächste Formular verlinken. Wird eine anders als rein sequentielle Abarbeitung der Dynaforms vorgesehen, müssen diese Pfeile ausgeblendet werden, was mit folgendem Javascript möglich ist:

```
hiddenById("DYN_FORWARD");
hiddenById("DYN_FORWARD[bullet]");
hiddenById("DYN_BACKWARD");
hiddenById("DYN_BACKWARD[bullet]");
```

Es besteht häufig der Wunsch, in einem Dynaform verschiedene Optionen anzubieten und den weiteren Prozessverlauf davon abhängig zu machen, welche Option ausgewählt wurde. Hier muss das ursprüngliche HTML-Formular-Modell, in dem ein Formular über eine einzige Submit-Aktion verfügt, etwas erweitert werden. Es bieten sich die folgenden zwei Lösungen an:

- Es werden mehrere Submit-Buttons in ein Dynaform zusammen mit einem versteckten Textfeld eingebaut. Im Dynaform-Javascript (siehe 2.2.1.5) wird der Wert des versteckten Feldes auf einen anderen Eventnamen gesetzt, je nachdem, welcher der Submit-Buttons gedrückt wird:


```
getField('EI_SUBMIT').onclick = function(){
    getField('EI_EVENT').value = 'submit';
}
```

Der Wert des versteckten Textfelds lässt sich anschließend in einem PHP-Trigger erfragen, um den weiteren Verlauf zu bestimmen.

- Es wird eine Listbox mit den verfügbaren Aktionen angezeigt. Im Dynaform-Javascript wird der Onclick-Handler der Listbox so überladen, dass er das Formular submittet:

```
getField('AN_EVENT').onclick = function() {
    getField('AN_EVENT').form.submit();
};
```

Der ausgewählte Listbox-Wert lässt sich anschließend in einem PHP-Trigger erfragen.

Im Pilotprojekt gab es die Anforderung, zwischen Prozessinstanzen zu springen. Dies ist grundsätzlich über Javascript-Redirects möglich. Die `G::redirect`-Methode (Das G steht für Gulliver, ein ProcessMaker-eigenes Javascript-Framework), führt ein Redirect nur im innersten Frame aus, was nicht ausreicht, weil die Prozessinstanz auch im äußeren Frame referenziert ist. Abhilfe schafft ein Standard-Javascript-Kommando, dass von PHP aus über `echo()` ausgeführt werden kann. Nach dem Redirect muss der alte Thread mit dem `die()`-Kommando beendet werden.

Dieser Ansatz hat zwar funktioniert, es gab aber Probleme im weiteren Verlauf der Session, da die alte Prozessinstanz immer noch referenziert war. Im Pilotprojekt war es akzeptabel, dass die alte Prozessinstanz im Sprungfall gelöscht wurde. Dies wurde über die Methode `removeCase()` erreicht:

```
echo("<script>parent.location='../cases/open?
APP_UID=" .@@CASE_SELECTION."&DEL_INDEX=$delIndex&action=todo'</script>");
$c = new Cases();
$c->removeCase(@@APPLICATION);
die();
```

3.2.1.4 Berechtigungskonzept (BE)

ProcessMaker verfügt über ein vollwertiges und sehr flexibles rollenbasiertes Berechtigungskonzept, das um drei Achsen organisiert ist:

- 1) Einem Nutzer werden **Rollen** zugewiesen, die aussagen, welche der ProcessMaker-Standardfunktionalitäten ihm zur Verfügung stehen. Jeder Rolle wird eine beliebige Menge aus 17 vordefinierten Rechten zugewiesen, wie z.B. `PM_LOGIN`, `PM_FOLDERS_VIEW` und `PM_CANCELCASE`. Es werden standardmäßig drei Rollen angelegt und zwar für Administratoren, Manager und Operator.
- 2) Nutzer werden auch in **Gruppen** organisiert, die bestimmen, an welchem der Prozess-Tasks ein Nutzer teilnimmt. Gruppen werden durch den Vorgangsentwickler angelegt und gewartet.
- 3) Nutzer können ebenfalls in **Abteilungen** und Unterabteilungen organisiert werden. Die resultierende Struktur definiert den Supervisor eines jeden Nutzers (siehe auch 2.2.1.3).

Die kostenpflichtige Enterprise-Edition kommt mit der Möglichkeit, LDAP bzw. Active Directory als

vollwärtige Authentifizierungsquelle anzubinden. In der Community-Edition hingegen werden Nutzer zwangsweise in der ProcessMaker-eigenen MySQL-Datenbank gespeichert und vom ProcessMaker-Administrator als separate Authentifizierungsinstanz gewartet, obwohl auch die Community-Edition die Möglichkeit anbietet, Nutzerdaten aus einer LDAP- oder Active Directory-Quelle zu importieren. Diese Funktionalität könnte verwendet werden, um die ProcessMaker-Nutzerbasis dem LDAP-Stand über einen nächtlichen Cron-Job anzugleichen.

Der Administrator weist einem Nutzer eine Anzeigeart zu. Dies bestimmt, ob der Nutzer eine Desktop- oder eine mobile Oberfläche erhält oder ob er zwischen den zwei wechseln kann. Es lässt sich auch eine vereinfachte Desktop-Sicht einstellen, bei der der Nutzer weder den Inbox noch sonstige erweiterte Funktionalitäten sieht, sondern immer nur die zunächst anstehende Aufgabe.

3.2.1.5 Bedienbarkeit / Ergonomie (BD)

Ein Dynaform hat mehrere Sichten:

- Eine **Vorschau-Sicht**, in der GUI-Objekte hinzugefügt und gelöscht werden können. Die Vorschau-Sicht soll dem eigentlich angezeigten Dynaform soweit als möglich ähneln (WYSIWYG). SQL-Kommandos, beispielsweise um Dropdown-Optionen zu laden, werden in der Vorschau ausgeführt, soweit keine Case-Variablen-Werte dazu nötig sind. Für das Dynaform definierter Javascript-Code wird in der Vorschau zwar ausgeführt. Da die Vorschau aber selbst mit Hilfe von Javascript gerendert wird, gibt es für Dynaforms mit selbst definiertem Javascript Fälle, in denen sich das Verhalten der Vorschau vom echten Verhalten unterscheidet. Zusammenfassend ist die Vorschau-Sicht zwar nützlich, um den ersten Wurf eines Dynaforms zu erstellen; sie kann aber ein strukturiertes Testen des echten Dynaforms nicht ersetzen.
- Eine **XML-Sicht**, die die Repräsentation des Dynaforms zeigt, die ProcessMaker intern verwendet. Im Zusammenspiel der anderen Ansichten kommt es gelegentlich zu Fehlern, bei denen nur noch die XML-Sicht angezeigt werden kann. In solchen Fällen ist das Problem meistens schnell erkennbar und behebbar (z.B. nicht gültiges XML).
- Eine **HTML-Sicht**, in der sich die automatisch erzeugte Darstellung des Dynaforms anpassen lässt. Bei Änderungen in den anderen Sichten scheint ProcessMaker davon auszugehen, dass die HTML-Sicht nicht verändert wurde und dem ursprünglich erzeugten Stand entspricht, so dass schnell Fehler entstehen, wenn bereits Anpassungen in der HTML-Sicht durchgeführt wurden. Deshalb empfiehlt es sich, HTML-Änderungen als letzter Schritt durchzuführen, wenn die restliche Funktionalität des Dynaforms bereits feststeht.
- Eine **Feldlisten-Sicht**, in der Attribute der einzelnen Felder angepasst werden können. Hier kann man beispielsweise angeben, ob ein Feld sichtbar und, wenn ja, ob es nur angezeigt oder auch editierbar sein soll. Manche Angaben wie z.B. die Angabe eines Default-Wertes für einen Checkbox funktionieren in dieser Sicht nicht und müssen in der XML-Sicht gemacht werden. Teilweise ist diese Sicht auch unnötig restriktiv. So darf ein Date Picker nur unter der Angabe zulässiger Werte erzeugt werden, obwohl die Fachlichkeit dies nicht erfordert und die Einschränkung nicht gilt, wenn die Anlage in der XML-Sicht stattfindet.
- Eine **Javascript-Sicht**, in der einer oder mehrere Javascripts eingegeben werden können, die beim Rendern des Dynaforms ausgeführt werden. Hier lässt die Benutzerfreundlichkeit leider zu wünschen übrig: klickt man einfach auf den Javascript-Tag, tippt man unwissend ins Leere! Man muss vielmehr über den Javascript-**Knopf** einen Javascript anlegen, der dann anschließend in der Javascript-Sicht verarbeitet werden kann. Bei der Anlage eines Javascripts wird um einen ‚Field Name‘ gebeten. Dieser Name ist irreführend: es geht um die Benennung eines Javascripts, der für das gesamte Dynaform

gilt.

Ein Dynaform muss aktiv gespeichert werden, wenn es geändert wurde. Wechselt man ohne Speichern in einen anderen Teil von ProcessMaker, werden die Änderungen stillschweigend verworfen.

Ein Dynaform-Feld hat einen technischen Namen, der die verbundene Prozessvariable bestimmt, sowie optional einen oder mehrere Labels. Ist kein Label für ein Feld konfiguriert, fungiert der technische Name als Label. Etwas tückisch ist, dass die Labels sprachspezifisch sind und standardmäßig angelegt werden in der Sprache der Oberfläche zur Entwicklungszeit. Entwickelt man einen deutschsprachigen Dialog mit der englischsprachigen Oberfläche, funktioniert alles wunderbar, bis man die Oberflächensprache zum Testen wechselt: alle Labels werden dann unerwartet durch ihre korrespondierenden technischen Namen ersetzt!

ProcessMaker bietet für individuelle Felder einige Validierungsoptionen (z.B. Inhalt muss numerisch sein) sowie Konvertierungsoptionen (z.B. Kleinbuchstaben in Großbuchstaben konvertieren). Leider berücksichtigt die Standardvalidierung für alphanumerische Inhalte nur die englischsprachigen Buchstaben, so dass die Option für europäische Sprachen nicht nutzbar ist und eine selbstentwickelte Javascript-Validierung nötig ist. Dies ist ein Problem, das am sinnvollsten im ProcessMaker-Quellcode anzupassen wäre, wenn eine Firma das Produkt in Europa einsetzen wollte (siehe aber 2.2.1.11).

Leider funktionieren die Standardvalidierungen bzw. –Konvertierungen nicht, wenn ein Dynaform mit Tastatur (Return- oder Enter-Knopf) anstatt per Mausklick submittet wird. Um dieses Problem zu umgehen, empfiehlt es sich, Tastatur-Submits per Javascript auszuschalten:

```
document.onkeypress = function(evt)
{
    if (evt.keyCode == 13)
    {
        return false;
    }
}
```

Das Überladen von Events in Javascript funktioniert manchmal nicht und kann sogar dazu führen, dass ProcessMaker-eigene Funktionalität ungewollt inaktiviert wird. Der onblur-Event eines Feldes wird zwar in der Vorschau-Sicht erzeugt, bei der echten Verwendung eines Dynaforms aber nicht mehr; und wird der onsubmit-Event eines Dynaforms überladen, werden die Standardvalidierungen und –Konvertierungen nicht mehr ausgeführt. Es gibt zwei Strategien, um mit diesem Problem umzugehen:

- Events können der ProcessMaker-eigenen Leimnud-Framework hinzugefügt werden, z.B.

```
leimnud.event.add(getField('FIELDNAME'), 'blur', function);
```

- Das Leimnud-Framework bietet keine Möglichkeit, die weitere Verarbeitung des Events zu unterdrücken, was beispielsweise bei Validierungen nötig ist. In solchen Fällen empfiehlt es sich, auszuprobieren, welche Events ohne unerwünschte Nebeneffekte funktionieren. Für Formvalidierungen, bei denen der onsubmit-Event nicht zur Verfügung steht, kommt beispielsweise der onclick-Event in Frage:

```
getField("SUBMIT").onclick = function ()
{
    if (nachname == '' || nachname.length < 1){
```

```

        G.alert("Das Feld 'Nachname' darf nicht leer sein.");
        return false;
    }
}

```

Tabellarische Inhalte lassen sich mit Hilfe sogenannter **Grid-Dynaforms** darstellen. Ein Grid-Dynaform wird als eigenständiges Objekt angelegt und gepflegt; zur Anzeige muss es dann in ein Haupt-Dynaform eingefügt werden. Die mit ihm verbundene Prozessvariable ist ein Associative-Array, das etwas ungeschickterweise 1-basiert ist, anstatt 0-basiert wie es in PHP üblich ist. Wird ein solches Array in einem PHP-Trigger angelegt, müssen entweder die Keys explizit angegeben werden, oder das Array als letzter Schritt wie folgend umgebaut:

```

array_unshift($workingArray, "placeholder");
unset($workingArray[0]);

```

Im Pilot-Projekt ergab sich das Problem, dass die Inhalte eines Grid-Dynaforms ausschließlich in waagrechten ganzen Reihen dargestellt werden können. Da jeder Eintrag über um die dreißig Felder verfügte, wäre eine solche Darstellung kaum benutzbar gewesen.

Dieses Problem lässt sich so lösen, dass ein Haupt-Dynaform angelegt wird, das alle Attribute eines Eintrags auf einer Seite darstellt. Die tabellarische Sicht im Grid-Dynaform umfasst hingegen nur einige wenige Attribute, aber dafür ein HTML-Link auf das Haupt-Dynaform samt einem ID-Parameter, um den Eintrag zu identifizieren. PHP-Triggers laden die der ID zugehörigen Attribute aus einem selbst verwalteten Associative Array in die Prozessvariablen der Haupt-Dynaform. Nachdem das Hauptdynaform submitted wurde, werden die Werte wieder zurück aus den Prozessvariablen in den Associative Array geladen. Um Race-Conditions zu vermeiden, empfiehlt es sich, zur Identifizierung der Einträge im selbstverwalteten Associative Array ein GUID zu verwenden (GUIDs lassen sich mit dem Kommando `G::generateUniqueID()` erzeugen) anstatt dem AssociativeArray-Index.

3.2.1.6 Reporting (RP)

Die Community-Edition von ProcessMaker bietet keine eigene Reporting-Funktionalität an, um Statistiken über die Vorgangsbearbeitung zu analysieren und darzustellen. Stattdessen können andere herkömmliche dispositive Werkzeuge verwendet werden. Diese können auf die ProcessMaker-eigene MySQL-Datenbank sowie gegebenenfalls auf andere verwendete Datenquellen zugreifen.

ProcessMaker speichert die Prozessvariablen zu einer gegebenen Prozessinstanz zwar in der Datenbank, aber in JSON, was es für Reporting-Tools schwierig macht, sie zu analysieren. Um dieses Problem zu umgehen, lassen sich in ProcessMaker sogenannte **Report Tables** definieren, in die Prozessvariablen importiert werden können.

3.2.1.7 Mitgeltende Dokumente (MD)

Es können beliebige Dateien auf den ProcessMaker-Server hochgeladen und mit Prozessinstanzen assoziiert werden. Ein Behälter für solche beliebigen Dateien nennt sich ein **Input Document** bzw. **Eingabedokument**. Viel interessanter ist aber die Möglichkeit, strukturierte **Output Documents** bzw. **Ausgabedokumente** zu erzeugen, in die die Werte von Prozessvariablen für die geltende Prozessinstanz eingefügt werden können.

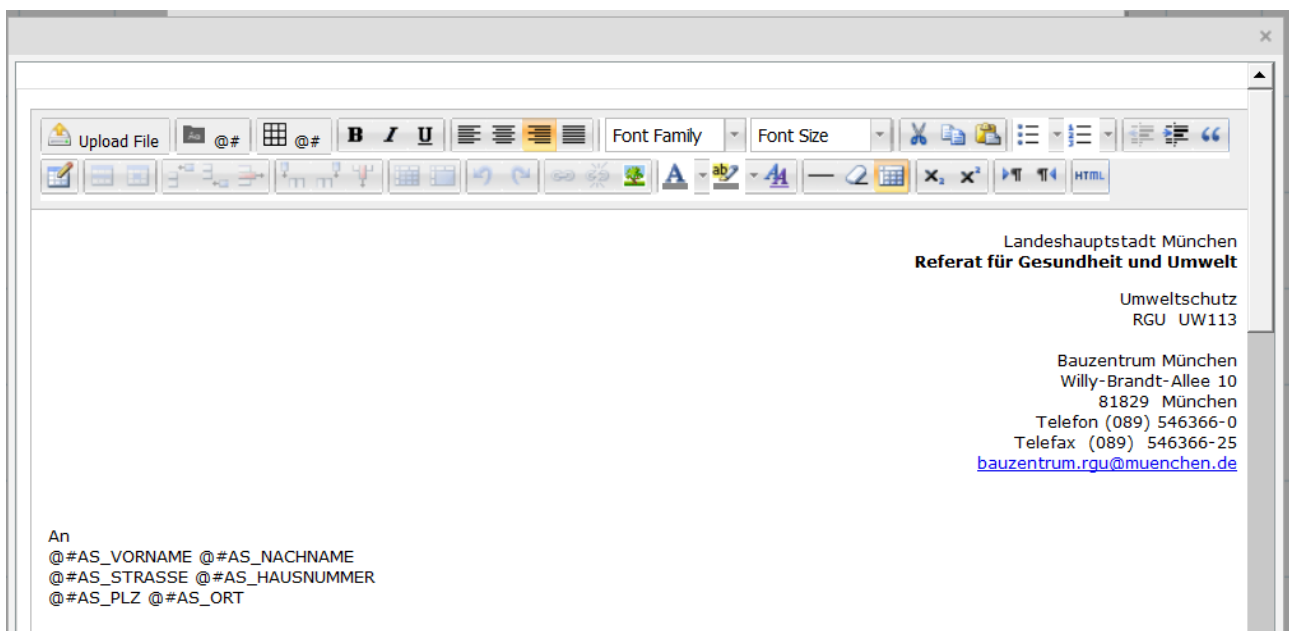


Abbildung 15: ProcessMaker: Editieroberfläche für Ausgabedokumente

Theoretisch besteht für Ausgabedokumente eine Versionierungsmöglichkeit, die aber auf Grund eines Bugs zumindest in der Version 2.8 nicht verfügbar ist (die Versionen werden falsch herum indiziert in der Datenbank abgespeichert.) Dies zwingt den Nutzer dazu, Ausgabedokumente ohne Versionierung zu nutzen, so dass ein bestehendes Ausgabedokument überschrieben wird, wenn für eine Prozessinstanz ein neues Dokument mit Hilfe des gleichen Templates erzeugt wird.

Ausgabedokumente werden standardmäßig in einem speziellen Schritt erzeugt und angezeigt, der anstelle eines Dynaforms in einen Task eingefügt wird. Es besteht aber auch die Möglichkeit, ein Ausgabedokument in einem PHP-Trigger mit der Methode `PMFGenerateOutputDocument()` zu erzeugen und anschließend in einem normalen Dynaform eine Anzeigemöglichkeit anzubieten. Hier ist eine zusätzliche SQL-Query im Trigger nötig, um den Link zu ermitteln:

```
executeQuery("SELECT APP_DOC_UID AS APPDOCUID FROM APP_DOCUMENT WHERE
DOC_UID = '$outputDocumentTemplateId' AND APP_UID = '" .@@APPLICATION .
"'");
```

Die zulässigen Formate für Ausgabedokumente sind `.docx` und `.pdf`. Das ursprünglich erzeugte Dokument wird von ProcessMaker auf dem Server gespeichert und kann nachträglich angesehen werden; wenn der Nutzer die Datei nachträglich ändert, bleibt dies für ProcessMaker unsichtbar. So empfiehlt sich das `.pdf`-Format, da es bei einer `.pdf`-Datei deutlich unwahrscheinlicher ist als bei einer `.docx`-Datei, dass ein Nutzer über die Fähigkeit verfügt, sie nachträglich zu ändern. Muss der Nutzer einen Teil eines Dokumentes frei schreiben, kann der Text in einem vorgelagerten Schritt in eine Prozess-Variable gespeichert werden.

Im Pilot-Projekt waren für die freigeschriebenen Texte Vorlagen nötig. Diese konnten nicht in einer PM-Table gespeichert werden, da hier die maximale Spaltenlänge auf 255 Buchstaben begrenzt ist. In einer produktiven Lösung könnte solche Vorlagen aus einer externen Datenbank bzw. aus einer extern definierten SQL-Tabelle eingelesen werden; im Pilot-Projekt hingegen wurde die Anforderung mit Heredocs verwendet (siehe 2.2.1.1).

Ein Benutzer mit den erforderlichen Rechten darf durch alle Dokumente navigieren, die im Rahmen verschiedener Instanzen eines Prozesses erzeugt wurden. Die Dokumente werden nach einem virtuellen Filesystem geordnet: beim Erzeugen eines Dokumentes wird ein virtuelles Verzeichnis ggf. mit Hilfe von Case-Variablen angegeben, in welchem das Dokument abzuspeichern ist. In diesem virtuellen Verzeichnis findet sich das Dokument in der Dokument-Historie. Es ist wichtig zu verstehen, dass kein Zusammenhang besteht zwischen diesem virtuellen Filesystem und dem echten Ablageort der Dateien auf dem Server, der in der Konfigurationsdatei `sysGeneric.php` festgelegt wird.

3.2.1.8 Performance / Benutzbarkeit (PB)

Im Kontext der Pilotstudie konnte keine Lasttests durchgeführt werden, so dass lediglich die Antwortzeiten bei einer kleinen Anzahl von Nutzern erlebt wurden. Hier schien die Performanz unter Linux völlig akzeptabel, so dass den Hauptteil der Antwortzeiten gefühlt eher die Netzwerkkommunikation als die Anwendung selbst ausmachte. Unter Windows hingegen war die Performanz völlig unakzeptabel mit Antwortzeiten von zum Teil über 10 Sekunden. Es konnte aber nicht geklärt werden, ob dies ein allgemeines Problem darstellt oder ob es durch irgendwelche Besonderheiten am einzelnen System hervorgerufen wurde. Die Windows-Variante ist aber ohnehin nicht für den produktiven Einsatz vorgesehen (siehe Einleitung).

Da Case-Variablen in einer Mysql-Datenbank gespeichert und verwaltet werden, eignet sich die Community Edition von ProcessMaker nicht für Fälle, in denen sich die Gesamtlast einer Anwendung nicht durch eine einzige Mysql-Instanz stemmen lässt. Diese Einschränkung passt aber gut in den leichtgewichtigen, eher klein ausgelegten Kontext, in dem ProcessMaker evaluiert wurde. Für Anwendungsfälle, in denen horizontale Skalierung eine Anforderung ist, wird vermutlich bereits aus grundlegenden Gründen ein anderes Tool eine bessere Wahl sein.

Datenbankzugriffe in PHP-Triggern werden über die ProcessMaker-eigene `executeQuery()`-Methode ausgeführt und erfolgen nichttransaktionell: es wird nach jedem Methodenaufwurf ein automatischer Commit durchgeführt. Es wäre zwar möglich, mit PHP-Bordmitteln Transaktionen nachzuimplementieren. Da dies aber aufwändig ist, empfiehlt es sich für leichtgewichtige Anwendungen, einen Optimistisches-Locking-Mechanismus zu implementieren. Im folgenden Beispiel muss eine eindeutige Zahl pro Prozessinstanz vergeben werden; der herkömmliche auto-increment-Mechanismus kann aber nicht zum Einsatz kommen, weil die Zahl mit jedem Jahreswechsel wieder auf eins gesetzt werden muss. Mit optimistischem Locking lässt sich diese Anforderung auch ohne Transaktionen threadsicher realisieren:

```
$zusatz = null;
$counter = 0;
while ($zusatz == null && $counter++ < 10)
{
    $res = executeQuery("SELECT PROJEKTNUMMER FROM PMT_PROJEKTZAEHLER WHERE JAHR = '"
. @AN_PROJEKTNRJAHRE . "'");
    if (is_array($res) and count($res) > 0)
    {
        $workingZusatz = $res[1]['PROJEKTNUMMER'];
        if (executeQuery("UPDATE PMT_PROJEKTZAEHLER SET PROJEKTNUMMER
= '" . ($workingZusatz + 1) . "' WHERE JAHR =
'" . @AN_PROJEKTNRJAHRE . "' AND PROJEKTNUMMER =
'" . $workingZusatz . "'") > 0)
        {
            $zusatz = $workingZusatz + 1;
        }
    }
    else
    {
        if (executeQuery("INSERT INTO PMT_PROJEKTZAEHLER (JAHR,
```

```

        PROJEKTNUMMER) VALUES ('" . @@AN_PROJEKTNRJAHR . "', '1')") >
        {
            $zusatz = 1;
        }
    }
}
if ($zusatz == null){
    G::SendMessageText('Jahr Projektnummer Zusatz konnte nicht gesetzt werden.',
    'ERROR');
}
@%AN_PROJEKTNRJAHRZUSATZ = $zusatz;

```

3.2.1.9 Sicherheit (SI)

Eine tiefgehende sicherheitstechnische Betrachtung von ProcessMaker konnte in der begrenzten Zeit nicht durchgeführt werden. Auf Grund einiger grundlegender Überlegungen lässt sich aber eine klare Empfehlung aussprechen, ProcessMaker nicht für extern im Internet verfügbare bzw. für sicherheitskritische interne Anwendung einzusetzen. Wenn ein solcher Einsatz doch erwogen würde, müsste auf alle Fälle ein gründlicher Penetrationstest erfolgen. Dies ist zwar schade, aber dennoch mit den üblichen Zielen der leichtgewichtigen Prozessverarbeitung vereinbar:

- Das Prozessdesign mit Dynaforms und Triggers impliziert grundsätzlich, dass Input-Validierung im Frontend im Browser stattfindet.
- Die Art und Weise, in die SQL-Queries in PHP-Triggers zusammengebaut werden, ist ohne große Sorgfalt leicht für SQL-Injections anfällig.

Die ProcessMaker-eigenen Methoden bieten keine Möglichkeit an, Prepared Statements zu erzeugen, obwohl es denkbar wäre, dies mit anderen PHP-Mitteln zu tun. Die Referenzierung einer Processvariable mit @\$variable anstatt @@variable garantiert zwar ein Escaping des Variablenwertes. Der Entwickler muss sich aber daran erinnern, diesen Referenzierungssyntax zu verwenden; und so wie das Escaping beschrieben ist, wird es vermutlich viel zu oberflächlich durchgeführt, um einen entschlossenen Angreifer zu verhindern.

Positiv ist, dass die ProcessMaker-Methode executeQuery() nur die erste Query im Argument ausführt und dass nur die DQL-Methoden (SELECT, INSERT, DELETE und UPDATE) unterstützt werden; es besteht keine Möglichkeit, über die Methode DDL-Befehle wie DROP TABLE auszuführen.

- Da Nutzdaten und Metadaten in der gleichen Datenbank gespeichert werden, hat jeder PHP-Trigger die Rechte, Prozessmetadaten anzupassen. Diese fehlende Defence-In-Depth lässt vermuten, dass gefundene Schwachstellen gnadenlos ausgenutzt werden könnten. Diese Schwachstelle lässt sich etwas abgemildern, indem statt PM-Tables eine externe Datenbank zum Einsatz kommt.

Cross-Site-Scripting-Schwächen (Javascript-Injektion) scheinen zumindest für die ProcessMaker-eigene Funktionalität berücksichtigt und vermieden worden zu sein.

3.2.1.10 Änderbarkeit (AB)

Eine wichtige Anforderung für das Pilot-Projekt war die Möglichkeit, dass ein fachlicher Administrator das Recht erhalten kann, Werteliste zu ändern, ohne für die volle Produktadministration verantwortlich sein zu müssen. Diese Anforderung ist bei ProcessMaker erfüllt. Angenommen, dass Wertelisten typischerweise in PM-Tables gehalten würden (siehe 2.2.1.1), sind die Rechte,

um diese editieren zu können (PM_SETUP und PM_SETUP_ADVANCE) getrennt vom Recht, um Prozesse entwickeln zu können (PM_FACTORY) (siehe 2.2.1.4).

3.2.1.11 Strategisch (SG)

Die Community-Edition von ProcessMaker wird unter der GNU Affero General Public License version 3 (AGPLv3) angeboten. Dies bedeutet, dass Quellcodeänderungen sämtlichen Nutzern einer ProcessMaker-Anwendung verfügbar gemacht werden müssen, die nicht der rechtlichen Entität angehören, die die Anwendung betreibt.

Diese **Bedingungen für eine Quellcodeveröffentlichung** sind ziemlich schnell erfüllt: bereits der Einsatz externer Mitarbeiter an einem fremden Standort könnte reichen. Laut Colosa ist die Veröffentlichung dann zwar für Änderungen zum Produktquellcode Pflicht, aber nicht für die Dynaforms und Triggers, die die Businesslogik enthalten (persönliche Kommunikation). So gesehen ist die Veröffentlichungspflicht vielleicht etwas lästig, aber ohne wesentliche betriebswirtschaftliche Relevanz, da zwar technische Verbesserungen, aber keine Fachlichkeit preisgegeben werden muss.

Die drei Enterprise-Ausprägungen kosten jeweils \$850, \$1.500 bzw. eine nicht veröffentlichte Summe pro Installation und Monat. Neben zusätzlichen Features und Support bringen die Enterprise-Ausprägungen einen unterstützten Upgrade-Pfad bei neuen Plattform-Versionen mit (siehe 2.2.1.13). Es ist aber wichtig, zu verstehen, dass nicht alle Verbesserungen, die die Open-Source-Gemeinschaft für die Community Edition zur Verfügung stellt, gleich mit der Enterprise Edition verwendet werden können. Laut der Wiki-Dokumentation galt dies in der Vergangenheit für die mehrsprachige Oberfläche; ob dies immer noch der Fall ist, konnte nicht abschließend ermittelt werden.

3.2.1.12 Support (SU)

Da die Firma Colosa schon seit dem Jahr 2000 besteht, ist die betriebswirtschaftliche Situation als stabil zu betrachten. Obwohl es zur Community Edition kein offizieller Support gibt, besteht eine große und noch sehr aktiv gepflegte Wiki, in der zu den meisten im Pilot-Projekt auftretenden Problemen schnell und effektiv Lösungen gefunden wurden. Es gibt auch sowohl deutsche als auch internationale Firmen, die bei Bedarf Unterstützung zu ProcessMaker bieten können.

3.2.1.13 Betriebsaspekte (BA)

ProcessMaker bietet für Dynaforms, PHP-Triggers und Javascripts keine Integration mit einem Version Control System wie Subversion oder GIT. Ein solches Tool kann auch nicht einfach extern auf die Installation zugreifen, da PHP-Triggers in der internen MySQL-Datenbank anstatt im Filesystem gespeichert werden. Dieses Problem ließe sich durch ein einfaches Skript lösen, der die PHP-Triggers aus der MySQL-Datenbank auf das Filesystem sowie aus dem Filesystem in die MySQL-Datenbank kopiert. (Dynaforms samt Javascripts werden standardmäßig im Filesystem abgelegt.)

Für Javascript empfiehlt es sich, so viel Funktionalität wie möglich in externe Dateien auszulagern und innerhalb ProcessMaker nur Rumpffunktionen zu schreiben, die die externe Funktionalität einbindet. Innerhalb ProcessMaker besteht nämlich keine Möglichkeit, Javascript-Code zwischen Dynaforms zu teilen, so dass er sonst von Dynaform zu Dynaform kopiert werden muss. Die gleiche Vorgehensweise ist zwar für PHP-Triggers nicht möglich, da Case-Variablen in Funktionen nicht referenziert werden können (siehe 2.2.1.1); für PHP-Trigger besteht aber eine produktinterne Möglichkeit, den gleichen Code an mehreren Stellen einzusetzen.

Eine relative schwere Einschränkung bildet der Mangel an garantierter Rückwärtskompatibilität für existierende Prozesse bei einem Versionsupgrade der Community-Version. Dies bedeutet, dass im schlimmsten Fall nach einem Upgrade auf eine neue Plattform-Version alle Prozesse neu konfiguriert und entwickelt werden müssen. In der Praxis war aber bei den letzten zwei Upgrades die Rückwärtskompatibilität gegeben; und da sie für die Enterprise-Edition vertraglich garantiert ist und die Codebasis größtenteils der Gleiche ist, scheint es unwahrscheinlich, dass sich das Modell so grundlegend ändern würde, um eine Migration unmöglich zu machen. Das Risiko bleibt aber, dass der Plattformverantwortliche einer Community-Installation bei einem Upgrade die nötigen Änderungen selber erforschen und die Migration händisch durchführen müsste, wenn dies für existierende Prozesse unbedingt erforderlich wäre (meistens bleibt ja immer noch die Möglichkeit, verschiedene Plattform-Versionen parallel zu betreiben).

Wird eine Prozessdefinition selbst angepasst, ist es typischerweise unerlässlich, dass bestehende Prozessinstanzen noch mit der alten Definition weiterlaufen, da sie nicht unbedingt den Zustand haben werden, den die neue Definition zum gegebenen Punkt im Prozessdurchlauf erwartet. Dies ist eine unvermeidbare fachliche Einschränkung. Was aber sicherlich bei ProcessMaker verbessert werden könnte, ist die fehlende Unterstützung für die explizite Benennung von Prozessversionen. So muss eine Versionsnummer im Prozesstitel, z.B. „FES v 1.0“, anstatt in einer dafür vorgesehenen Spalte angegeben werden.

Die Backup- und Restore-Funktionalität für einen Prozess wird durch die Kommandos `process - backup` und `process - restore` unterstützt. Die Linux-Versionen der Skripte setzen einiges an der Linux-Umgebung voraus, was nicht unbedingt zutreffen muss: in einem Fall während des Pilotprojektes haben die Skripte nicht funktioniert. Deren Funktionalität ist aber einfach genug, dass die Schritte problemlos per Hand durchgeführt werden können: ein Backup besteht aus einem Dump der internen Mysql-Datenbank, einer Kopie des Verzeichnisses, in dem Dateiobjekte zum Prozess gespeichert werden sowie einer Metadaten-Datei, in der beispielsweise der Mysql-Nutzer benannt wird, unter dem der Prozess läuft.

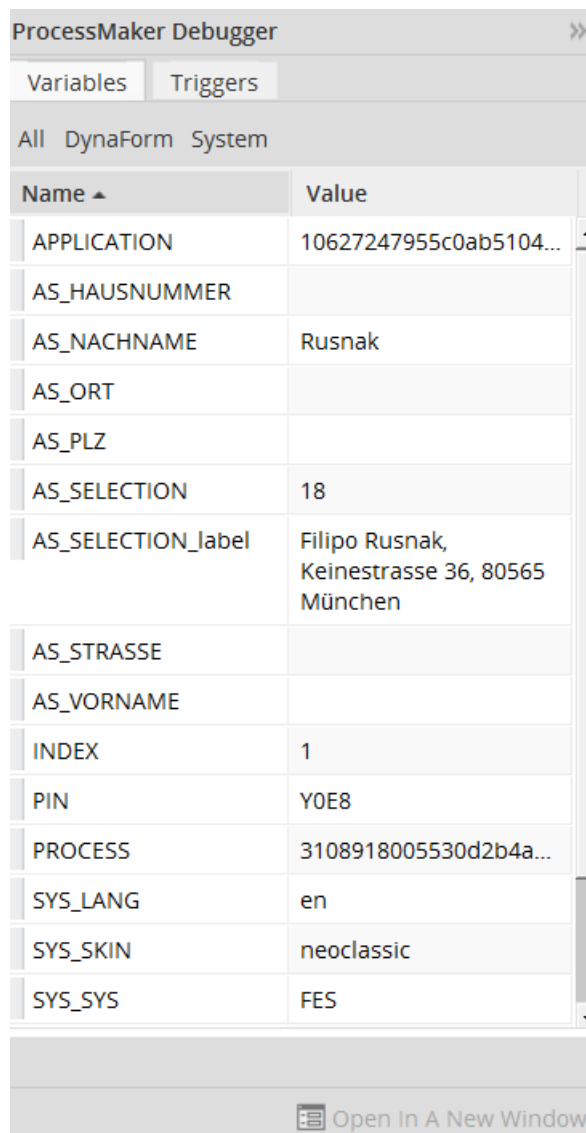
Mit einem manuellen Restore ist es auch möglich, einen unter Linux erzeugten Prozess auf Windows zu kopieren oder umgekehrt. Hierbei müssen folgende Probleme beachtet werden:

- Linux nutzt als Zeilenumbruch `\n`, Windows `\r\n`: Dateiobjekte müssen entsprechend angepasst werden.
- Unter Linux erzeugt Mysql Dumps in UTF-8. Beim Importieren in Windows muss dies explizit beim Start von Mysql mit der Option `--default-character-set=utf8` angegeben werden.
- Unter Windows erzeugt Mysql Dumps, in denen alle Objekte mit Kleinbuchstaben benannt sind. ProcessMaker erwartet aber Großbuchstaben. Hier muss die Option `lower_case_table_names=1` der Konfigurationsdatei `my.cnf` hinzugefügt werden.

Die Backup- und Restore-Funktionalität wäre grundsätzlich dafür geeignet, Prozessdefinitionen zwischen Instanzen wie Test, Integration und Produktion zu kopieren. Ein Problem wäre zwar, dass sowohl der Mysql-Dump als auch das kopierte Verzeichnis eine Mischung aus Metadaten und Daten enthalten, die jeweils zu kopieren bzw. nicht zu kopieren wären. Da die Strukturen relativ übersichtlich sind, könnte aber ohne allzu viel Aufwand ein Skript geschrieben werden, um nur die benötigten Metadata-Objekte mitzunehmen.

Insbesondere während der Entwicklung kann es nötig sein, verschiedene ProcessMaker-interne Caches zu leeren. Dies ist in der Admin-Sicht möglich (Optionen ‚Cases List Cache Builder‘ sowie ‚Clear Cache‘).

ProcessMaker verfügt über einen Debugger, mit dem die Werte der Prozessvariablen während der Ausführung einer Prozessinstanz überwacht werden können. Dies ist unerlässlich, um Probleme während der Entwicklungsphase nachzuvollziehen. In manchen, wenn nicht allen Situationen können ebenfalls über den Debugger Fehler im PHP-Code gefunden werden.



The screenshot shows the 'ProcessMaker Debugger' window with the 'Variables' tab selected. It displays a table of variables for the 'All DynaForm System'. The variables include application ID, user information (AS_HAUSNUMMER, AS_NACHNAME, AS_ORT, AS_PLZ, AS_SELECTION, AS_SELECTION_label, AS_STRASSE, AS_VORNAME), index, PIN, process ID, language, skin, and system type.

Name ▲	Value
APPLICATION	10627247955c0ab5104...
AS_HAUSNUMMER	
AS_NACHNAME	Rusnak
AS_ORT	
AS_PLZ	
AS_SELECTION	18
AS_SELECTION_label	Filipo Rusnak, Keinestrasse 36, 80565 München
AS_STRASSE	
AS_VORNAME	
INDEX	1
PIN	Y0E8
PROCESS	3108918005530d2b4a...
SYS_LANG	en
SYS_SKIN	neoclassic
SYS_SYS	FES

Open In A New Window

Abbildung 16: ProcessMaker: Der Debugger

3.2.2 Fazit

ProcessMaker bietet eine geeignete kostenfreie Lösung für leichtgewichtige Prozessverarbeitung, soweit die verarbeiteten Daten nicht sicherheitskritisch sind und kein öffentlicher Internet-Zugang vorgesehen ist. Bei allen anderen als die simpelsten Use-Cases sind PHP- und Javascript-Kenntnissen nötig, um Prozesse zu entwickeln. Insbesondere für Use-Cases, bei denen die Vorgangslogik bzw. das Benutzerberechtigungsmodell komplexer sind oder bei denen die ProcessMaker-eigene Funktionalitäten wie Inbox und Wiedervorlage nützlich sind, erlaubt ProcessMaker aber eine deutliche Produktivitätserhöhung gegenüber einer Entwicklung auf der grünen Wiese.

Um die Lernkurve für neue ProcessMaker-Entwickler etwas flacher zu halten (siehe Einleitung), sollte beim Ersteinsatz von ProcessMaker zum einen in einer kurzen (vielleicht eintägigen) Schulung auf die Besonderheiten des Produktes eingegangen werden, um den Einstieg zu erleichtern. Es sollte zum anderen ein Kochbuch mit Lösungen für die häufigsten technischen Use Cases erstellt werden, damit sich neue Entwickler auf die Fachlichkeit fokussieren können.

3.3 SuiteCRM

Bei SuiteCRM handelt es sich um eine Software für das Management von Kundenbeziehungen. SuiteCRM ist ein Fork der seit 2004 auf dem Markt befindlichen Software SugarCRM.

Die Software bietet die Möglichkeit, Kunden in Sicht auf verschiedenste Produkte zu verwalten. Hierzu können Prozesse hinterlegt werden, die sich auf die Hauptmodule: Kunden, Produkte, Verträge und Verkäufe beziehen. Sofern diese Hauptmodule verwendet werden, steht der komplette Funktionsumfang der Software zur Verfügung.

Es besteht innerhalb der Software die Möglichkeit vorhandene Module anzupassen (Customizing) oder komplett neue Module zu erzeugen. Dies kann man, in einfachen Fällen ohne Programmierkenntnisse (GUI) durchführen. Mit entsprechenden System- und Programmierkenntnissen in PHP und JavaScript lassen sich auch schwierige Aufgaben lösen.

Zahlreiche kommerzielle und nicht kommerzielle Erweiterungen runden den Funktionsumfang des Produktes ab.

3.3.1 Kriterien

Die kommenden Abschnitte nehmen auf folgende Arten von Benutzern Bezug:

- | | |
|---------------------------|--|
| • Administratoren (Admin) | Technischer Administrator |
| • Fachadministrator | Administrator mit dem Recht Benutzer, Rollen und Gruppenrechte (Securitygroups) einzurichten. |
| • Entwickler | Interner oder Externer Mitarbeiter der sich in der Struktur von SuiteCRM auskennt und die Programmiersprachen PHP und JavaScript beherrscht. |
| • Anwender | Sachbearbeiter der Fachabteilung |
| • Power User | Sachbearbeiter der Fachabteilung mit erweiterten Rechten und Kenntnissen. |

3.3.1.1 Datenhaltung / Modell (DM)

SuiteCRM unterstützt derzeit folgende Datenbank Anwendungen:

- MySQL, Oracle, Microsoft SQL

Entwickler erstellen und erweitern darin Tabellen mit der systemeigenen GUI von SuiteCRM. Referenzen zwischen den Tabellen können die Entwickler ebenfalls direkt in SuiteCRM vornehmen. Die GUI unterstützt dabei folgende Beziehungstypen:

- 1:1, 1:n und n:m

In SuiteCRM ist es auch nachträglich möglich, vorhandene Werte über die GUI weiter beschränken. Z.B. kann die Stringlänge verändert werden.

Für Entwickler ist es also möglich, Datenbankmodelle 1 zu 1 direkt mit SuiteCRM abzubilden.

3.3.1.2 Suchen (SU)

Admin- und Entwicklersicht

Neben der globalen Suche über alle Module und alle Felder existieren für jedes Modul zwei Arten einer Suche:

- Eine einfache Suche, eingeschränkt durch den Admin / Entwickler.
- Eine erweiterte Suche, eingeschränkt durch den Admin / Entwickler.

Beide werden vom Admin / Entwickler per Customizing erstellt. Was in diesem Zusammenhang „einfach“ und „erweitert“ bedeutet, legt der Admin / Entwickler individuell pro Modul fest. Natürlich im Idealfall so, dass sich hinter den Begriffen „einfach“ und „erweitert“ auch das verbirgt, mit dem der Benutzer rechnet.

Gruppenrechte (SecurityGroups) schlagen auf die Möglichkeiten der Suche durch, d.h. die Suchergebnisse enthalten nur Einträge, die die angemeldete Person entsprechend ihrer Berechtigungen auch sehen darf. Jede Gruppe kann hierzu ein eigenes Layout bekommen.

Da SuiteCRM eine Webanwendung ist, ist die Suchfunktion des Browsers (STRG + F) natürlich ebenfalls nutzbar.

Anwendersicht

Der Anwender hat jederzeit die Möglichkeit das globale Suchfeld zu benutzen.

Bei Aufruf eines Moduls werden dem Anwender die möglichen Suchfelder in der Kopfzeile angezeigt und darunter die Liste mit dem letzten Suchergebnis. Nach Ausführung der Suche wird eine Ergebnisliste angezeigt.

Die Auswahl des Suchfilters bleibt für eine weitere Einschränkung der Suche gespeichert und wird dem Benutzer weiterhin in der Kopfzeile angezeigt.

Es besteht auch die Möglichkeit, einen Suchfilter zu erstellen und diesen abzuspeichern. Bei Bedarf kann dieser Filter ausgewählt und genutzt werden.

3.3.1.3 Vorgangsmanagement (VM)

Es gibt zwei Möglichkeiten den Bearbeitungsstand nachzuvollziehen.

1. Der Anwender aktualisiert das Auswahlfeld, das den Zustandsautomat enthält händisch.
2. Der Entwickler kann unter Verwendung der vorhandenen Module die Möglichkeit nutzen, einen gewünschten Ablauf als Prozessmodell zu hinterlegen. Das funktioniert jedoch nicht mit selbst erstellten Modulen.

Der Anwender kann einen Vorgang ruhen lassen, indem er einen Wiedervorlagetermin erstellt. Wiedervorlagetermine tauchen als Listenansicht im Dashboard auf.

Es besteht die Möglichkeit mit vorhandenen Modulen, eine Ad hoc Zuweisung von Vorgängen an Bearbeiter zu ermöglichen. Das ist gleichermaßen an einzelne Nutzer, wie auch an Gruppen möglich.

3.3.1.4 Berechtigungskonzept (BE)

SuiteCRM setzt ein umfängliches Benutzer-, Rollen- und Rechtekonzept um. Die Authentifizierung von Benutzern via LDAP ist standardmäßig möglich. Innerhalb von SuiteCRM sind dem Benutzer weitere Daten (personalisierte Systemeinstellungen für den Benutzer) zugeordnet.

Mit der ausgefeilten Rechteverwaltung können die Administratoren den Benutzerzugriff über Rollen und Teams zentral verwalten und individuell festlegen, welche Funktionen und Datensätze den Anwendern bereitstehen sollen.

Die Rechtevergabe auf vorhandene bzw. selbst erstellte Module (lesen, schreiben, löschen, drucken ..) erfolgt mittels Rollen:

- SuiteCRM hat die Möglichkeit, beliebige Rollen festzulegen.
- Eine Rolle gilt für alle Module. Pro Modul legt sie die Rechte fest.
- Die Software hat die Möglichkeit, verschiedene Security Groups zu erstellen.
 - Jede Security Group ist von einer anderen getrennt.
 - Jede Security Group hat eine Rolle.
 - Jede Security Group hat Benutzer.

Benutzer in mehreren Security Groups können unterschiedliche Rollen haben. Einstellbare Regeln sind für diesen Fall vorhanden. Einzelne Benutzer können in ihren Rechten zusätzlich weiter beschränkt werden.

Die komplette Steuerung dieser Rechtevergabe wird in SuiteCRM über die GUI vorgenommen.

Rechte auf ein einzelnes Datenfeld lassen sich in den verschiedenen Sichten des Moduls mit PHP Code realisieren.

3.3.1.5 Bedienbarkeit / Ergonomie (BD)

Alle Vorteile einer Webanwendung sind vorhanden. Die Bedienung der Anwendung ist mit Maus und Tastatur möglich.

Die Oberfläche kann an die Anforderungen der Fachabteilung angepasst werden. Diese Anpassungen werden ausschließlich in der Administrativen GUI ausgeführt, bewährt hat sich die Anpassung direkt zusammen mit dem Endbenutzer.

Die Gestaltung der GUI ist mit folgenden Elementen möglich:

- Reiter neben und untereinander
- Unterreiter
- Popups
- Listen

Das Customizing der Software erfolgt bis auf eine Ausnahme (Anzahl anzuzeigender Spalten) über die GUI.

Programmatische Erweiterungen erfolgen über PHP und JavaScript.

3.3.1.6 Reporting (RP)

Es besteht die Möglichkeit, Listen im csv-Format zu exportieren.

Zudem bietet die Software umfassende Möglichkeiten für Reporting. Grafische Auswertungen können aus dem Modul Reporting heraus generiert werden. Sie können mittels Konfiguration auf dem Startbildschirm der Software (Dashboard) angezeigt werden.

Das kostenlose Reporting-Tool K-Report ermöglicht es dem PowerUser / Entwickler / Admin CSV Export Dateien oder komplexe Datenbankabfragen zu erstellen.

Mit dem ebenfalls kostenlosen Reporting-Tool ZuckerReport besteht die Möglichkeit Jasper Reports einzubinden, die mit dem frei verfügbaren Tool iReport als JasperXML-Dateien erstellt und dann mittels ZuckerReport in SuiteCRM eingebunden und ausgeführt werden können.

3.3.1.7 Mitgeltende Dokumente (MD)

Die Anwendung bietet die Möglichkeit, mit Metadaten befüllte PDF-Dateien zu erzeugen und in der Anwendung zu speichern.

Mit einer kommerziellen Erweiterung ist es zusätzlich möglich, mit Metadaten befüllte Office Dokumente zu erzeugen, zu öffnen, zu bearbeiten und in der Anwendung zu speichern.

3.3.1.8 Performance / Benutzbarkeit (PB)

Die Software ist die führende Open Source Lösung im Bereich Customer Relationship Management und wird bei mehreren mittelständigen Unternehmen weltweit eingesetzt. Dabei arbeiten bis zu 100 Personen gleichzeitig am System.

Sollten zwei Personen denselben Fall bearbeiten so wird der zweiten Person beim Speichern angezeigt, welche Daten von der ersten Person geändert wurden und nachgefragt, ob diese Daten übernommen werden sollen.

3.3.1.9 Sicherheit (SI)

SugarCRM verfügt über diverse Sicherheitsmechanismen auf der Applikationsebene, um das Login zum System zu validieren und Aktivitäten zu dokumentieren.

3.3.1.10 Änderbarkeit (AB)

Änderungen können jederzeit in Absprache zwischen der Fachabteilung und dem Admin/Entwickler durchgeführt werden.

Es werden zwei Arten von Änderungen unterschieden:

1. Änderung ohne Programmieraufwand (GUI)
2. Änderungen mit Programmieraufwand (GUI / PHP)

In beiden Fällen gilt: Die Datenbankstruktur wird nicht verändert.

Datenfelder können nur in dem Maße verändert werden, dass sichergestellt ist, dass Daten jederzeit weiterverarbeitet werden können. Z.B. kann ein String von 50 auf 150 Zeichen erweitert werden, aber nicht von 150 auf 50 Zeichen verkürzt.

Datentyp-Änderungen (z.B. Int auf String) müssen mittels Programmieraufwand realisiert werden. Hier besteht die Möglichkeit, neue String Felder zu erzeugen und ggf. vorhandene Daten mittels SQL Befehl zu kopieren.

Sind Admin und Entwickler die selbe Person, können Änderung ohne Programmieraufwand innerhalb von Minuten durchgeführt werden. Dazu gehören auch Änderungen am CSV Export. Anlegen neuer User ist durch den Fachadministrator ebenfalls schnell möglich.

Das komplette Layout kann innerhalb von 2 Wochen durch Entwickler vollständig geändert werden.

3.3.1.11 Strategisch (SG)

Suite CRM ist in PHP und Javascript geschrieben. Es nutzt MYSQL und seit neusten Oracle Datenbanken.

Es handelt sich bei Suite CRM um eine Open Source Software, die mit einem grafischen Editor zur Bearbeitung der Abläufe für Endbenutzer und auch der Abläufe für Administratoren ausgestattet ist. Die Software ist gemäß Wiki mandantenfähig.

3.3.1.12 Support (SU)

Für SuiteCRM besteht eine Community seit dem Jahr 2004. Support durch Drittanbieter bieten in Deutschland z.B.

- Code Partners Deutschland
- Delivery Partners Deutschland
- Integration Partners Deutschland

3.3.1.13 Betriebsaspekte (BA)

Je nach Rollen und damit Benutzer können unterschiedliche Aspekte des Systems von verschiedenen Personen administriert werden. Damit ist eine Aufteilung nach Zuständigkeiten etwa zwischen Fach- und IT-Abteilung möglich.

Sämtliche Customizing Einstellungen befinden sich in einem eigenen Pfad und liegen somit weitestgehend getrennt vom Produkt vor. Damit können die Anpassung getrennt gesichert werden.

Ein Austausch zwischen unterschiedlichen Installationen findet über Pakete statt. Sämtliche Updates und die Installation neuer Erweiterungen werden in Form von Paketen vorgenommen. Updates führt das System nach Anstoß durch den Administrator automatisiert durch.

Das System schreibt Logfiles nach Vorgabe des Administrators.

3.3.2 Fazit

Sugar CRM ist eine Anwendung, die durch ihre Möglichkeit der grafischen Benutzeroberfläche und der damit verbunden einfachen Bedienung besticht.

Es besteht mit dem Einsatz der Anwendung die Möglichkeit **auch für Computer interessierte Personengruppen** komplexe Datenbank Anwendungen zu erstellen. Durch die Bereitstellung dieser Anwendung können viele derzeit laufende Vorhaben kostengünstig durchgeführt werden.

Durch diese Anwendung können eigene Anwendungen, programmiert mit eigenen Programmiersprachen der LHM (KOI Anwendungen) oder aber Calc Tabellen, die von mehreren User gleichzeitig/ nacheinander benutzt werden, abgelöst werden.

Das dadurch frei werdende IT-Personal kann dazu genutzt werden, Logik, die im System implementiert werden muss, zu programmieren. Ein weiterer Vorteil, durch den Einsatz von PHP und Javaskript kann jederzeit Know-how von außen eingekauft werden, sofern es keine Kapazitäten innerhalb der städtischen IT gibt.

Durch den Einsatz der Anwendung lassen sich Ausschreibungen vermeiden und somit Beschaffungszeiten verkürzen.

Suite CRM benötigt nur die Bereitstellung der Infrastruktur, sofern diese vorhanden ist, kann ein kleiner Kreis von Personen, innerhalb des Referates, die Administration komplett übernehmen.

Suite CRM ist eine kostengünstige, schnell einsetzbare Alternative zu den jetzt im Einsatz befindlichen Möglichkeiten.