Herbert Groiss

Geschäftsprozessmanagement mit @enterprise

NOREA VERLAG

Bibliographische Informationen der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Alle Rechte vorbehalten.

Das Werk und seine Teile sind urheberrechtlich geschützt. Jede Verwertung in anderen als den gesetzlich zugelassenen Fällen bedarf deshalb der vorherigen schriftlichen Einwilligung des Verlages.

Hinweis zu §52a UrhG:
Weder das Werk noch seine Teile dürfen ohne solche Einwilligung überspielt, gespeichert, kopiert und in ein Netzwerk eingespielt werden. Dies gilt auch für Intranets von Firmen und von Schulen und sonstigen Bildungseinrichtungen.

@enterprise ist eine eingetragene Marke der Groiss Informatics GmbH, andere Namen sind teilweise Markenzeichen der jeweiligen Hersteller. Trotz sorgfältiger Prüfung kann nicht garantiert werden, dass das Dokument fehlerfrei ist. Jegliche Haftung für die Richtigkeit des Inhalts kann nicht übernommen werden.

Gesamtherstellung:
NOREA Druck und Verlag Marija Miksche
9020 Klagenfurt • AUSTRIA
E-Mail: office@norea.at • www.norea.at

Umschlagvorderseite: Heinzelmännchenbrunnen in Köln, 1899, Foto des Autors

Alle Rechte beim NOREA Verlag • Klagenfurt /AUSTRIA
© 2012 NOREA Verlag, Klagenfurt

ISBN 978-3-85312-085-9

Herbert Groiss

Geschäftsprozessmanagement mit @enterprise

NOREA

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Was ist Geschäftsprozessmanagement?

Prozessmanagement beschäftigt sich mit dem Finden, Dokumentieren, Gestalten und Verbessern von Geschäftsprozessen (Wikipedia). Beispiele für betriebliche Prozesse sind: Auftragsbearbeitung, Abwicklung einer Dienstreise oder Rechnungsbearbeitung. Die Beschäftigung mit Prozessen ist zu einer der Kernaufgaben in einem modernen Unternehmen geworden. Wir können dabei mehrere Schritte unterscheiden:

- Erfassen der Prozesse: Der erste Schritt ist die Analyse der bestehenden Prozesse, wie laufen sie ab, wer ist beteiligt, was wird gemacht. Die Prozesse müssen dann entsprechend dokumentiert werden. Wie diese Dokumentation aussieht und welche Notationen Verwendung finden, ist ein wesentlicher Inhalt dieses Buches (siehe Kapitel 2).

- Prozessausführung: Die konkrete Durchführung der Prozesse. Der Bereich Workflow-Management beschäftigt sich mit der IT-unterstützten Prozessbearbeitung, Workflow bedeutet Arbeitsablauf. Die modellierten Prozesse werden in eine Form gebracht, die von einem Workflow-Managementsystem interpretiert werden kann. Dieses kann dann den Ablauf der konkreten Prozesse steuern und so einen koordinierten Ablauf aus manuellen und automatischen Arbeitsschritten (Aktivitäten) sicherstellen.

- Monitoring und Analyse: Die Prozessabläufe werden dokumentiert, sodass eine Kontrolle der Abläufe ermöglicht wird. Die Daten über die Prozessausführungen werden gesammelt und verdichtet, damit ist eine Interpretation dieser Informationen möglich.
- Optimierung der Prozesse: Mithilfe der Daten aus der Analyse werden Optimierungsmöglichkeiten gesucht. Letztendlich werden die Prozesse diesen Erkenntnissen entsprechend angepasst. Für diese Tätigkeit wird oft der Begriff „Business Process Reengineering“ verwendet.

Die Optimierung kann als „Re-Modellierung“ gesehen werden, sodass wir die Schritte in einem Zyklus darstellen können, siehe Abb. 1.1. Wiederkehrende Ausführungen der Analyse- und Optimierungsschritte sollen zu kontinuierlicher Verbesserung der Prozesse führen.

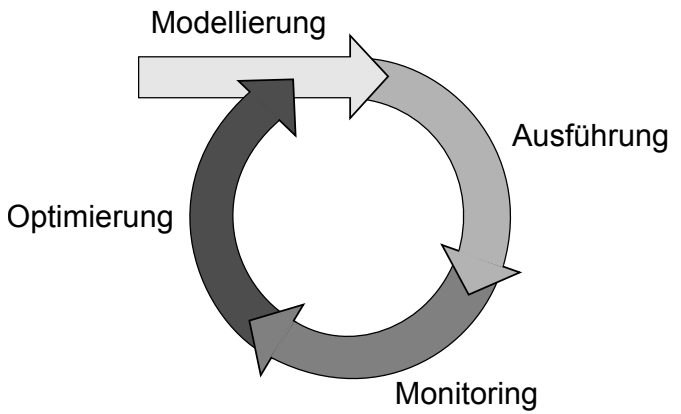


Abbildung 1.1: Der Zyklus des Geschäftsprozessmanagements

Welche Vorteile ergeben sich für ein Unternehmen aus den einzelnen Schritten des Prozessmanagements? Wir betrachten die einzelnen Phasen genauer.

1.1.1 Modellierung der Prozesse

Schriftlich dokumentierte Prozesse legen fest, wie etwas gemacht werden soll. Sie dienen als Richtlinie, erleichtern das Einarbeiten neuer Mitarbeiter und die Kommunikation zwischen den Prozessbeteiligten.

Die Idee der Niederschrift der Abläufe in einer Organisation ist recht alt und ein frühes Beispiel ist die Ordensregel des Benedikt von Nursia aus dem

sechsten Jahrhundert, siehe Abb. 1.2. Warum man damals bereits auf die Idee kam, so etwas niederzuschreiben, ist klar: In einer international agierenden Organisation mit vielen Niederlassungen wie dem Benediktinerorden kann der „Konzernchef“ nicht jedem Novizen einzeln die Abläufe erklären.

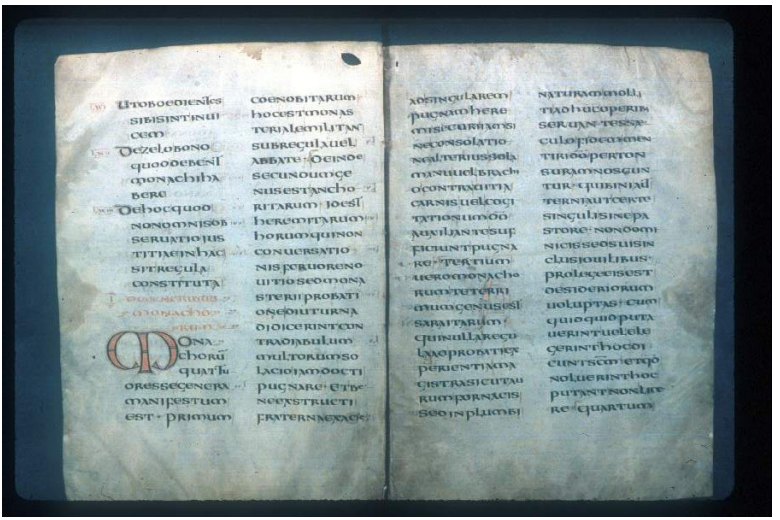


Abbildung 1.2: Regula Benedicti, frühes Beispiel eines Prozesshandbuchs, Abschrift aus dem 8.Jh.

Das Vorhandensein von dokumentierten Prozessen ist oft Voraussetzung von Zertifizierungen, z.B. ISO9000, oder auch von Vorschriften von Regulierungsbehörden, z.B. der US-amerikanischen Food and Drug Administration (FDA), die Regeln für Nahrungsmittel- und Pharma-Erzeuger festlegt.

Die Sammlung der dokumentierten Prozesse in einer Organisation wird als *Prozesshandbuch* bezeichnet. In Kapitel 2 werden wir zeitgemäße Formulierungen von Prozessen kennenlernen.

1.1.2 Ausführung der Prozesse

Der zweite Schritt im Zyklus des Prozessmanagements ist die IT-unterstützte Bearbeitung der Prozesse mit einem sogenannten Geschäftsprozessmanagementsystem oder Workflowsystem.

Zu den Begriffen: Workflowsystem oder Workflowmanagementsystem ist die traditionelle Bezeichnung für Systeme zur Prozessautomatisierung, d.h. für

Systeme, welche die Ausführung von Prozessen unterstützen. Seit 1993 gibt es die Workflow Management Coalition (Webadresse *http://www.wfmc.org*), eine Vereinigung der Hersteller solcher Systeme, die sich vor allem mit der Erstellung von Standards in diesem Bereich beschäftigt.

Der Begriff Geschäftsprozessmanagement ist etwas breiter zu sehen und umfasst die betriebswirtschaftlichen und organisatorischen Aspekte des Themas. Geschäftsprozessmanagementsysteme sind Systeme zur Unterstützung der Bearbeitung von Geschäftsprozessen. Darin ist alles Mögliche enthalten, von Systemen zur Darstellung von Prozessen, über Automatisierung bis Prozessdokumentation und Monitoring. Da sich die Workflow-Produkte zu einer umfassenderen Behandlung des Themas hin entwickeln, spricht man auch hier nun meist von Geschäftsprozessmanagementsystemen [**?**].

Mit der Ausführung von Prozessen in einem BPMS ist gemeint, dass das System die Kontrolle über den Prozess hat: Wenn ein Benutzer einen Prozessschritt abgeschlossen hat, teilt er das dem System mit und dieses wählt anhand der vorgegebenen Ablaufstruktur den nachfolgenden Schritt und Bearbeiter aus.

Der Ausführung von Prozessen in einem BPMS geht in der Regel die Definition der Prozesse voran, diese ist aber nicht unbedingt Voraussetzung dafür. Es ist auch möglich, den Prozessablauf während der Ausführung festzulegen, näheres dazu in Abschnitt 3.5.

Umgekehrt müssen nicht alle Prozesse, die definiert wurden, in einem BPMS ausgeführt werden. Wir möchten drei Kriterien nennen, die die Bearbeitung in einem BPMS sinnvoll erscheinen lassen:

- Der Prozess ist strukturiert: Die Aufgaben werden arbeitsteilig verrichtet, es gibt eine logische oder natürliche Abfolge der Aufgaben und es erfolgt keine unmittelbare Zusammenarbeit der Prozessbeteiligten in einem Raum.

  Das heißt, wenn an der Erfüllung der Aufgabe in der Regel nur eine Person beteiligt ist, ist der Einsatz von BPMS nicht zweckmäßig – falls nicht andere Gründe dafür sprechen. Ebenso ist es nicht sinnvoll, ein BPMS einzusetzen, wenn die Gesamtaufgabe nicht in Teilaufgaben zerlegt werden kann, z.B. wenn eine Aufgabe gemeinsam bei einer Besprechung gelöst wird.

- Der Prozessablauf soll dokumentiert werden: Ist es aufgrund der Nachvollziehbarkeit nötig, feststellen zu können, wer was wann gemacht hat, ist der Einsatz eines BPMS sinnvoll. Und wir möchten betonen: Es ist fast

immer sinnvoll, Geschäftsvorgänge nachvollziehbar zu gestalten.

- Schritte sollen automatisiert werden: Bei der Ausführung von Prozessen in einem BPMS können die Einzelschritte manuell oder durch ein IT-System durchgeführt werden. Das BPMS übernimmt dabei in der Regel nicht die Automatisierung selbst, sondern ist für die Datenübermittlung zu den Systemen, den Aufruf der Systeme und die Reaktion auf die Ereignisse in diesen Systemen zuständig.

Liegen eine oder mehrere dieser Voraussetzungen vor, ergeben sich durch die Prozessausführung in einem BPMS Vorteile. Diese sind:

- Prozessqualität: Die Abarbeitung der Prozesse folgt der Prozessdefinition – und somit den Vorgaben der Prozessdesigner. Abweichungen davon sind zwar möglich, aber nicht unbegrenzt und immer durch das System dokumentiert.
- Geschwindigkeit: Die Geschwindigkeit der Prozessbearbeitung erhöht sich, dies geschieht durch verschiedene Faktoren:
  - Wegfall der Transportzeiten, verglichen mit Papierbearbeitung
  - Möglichkeit der Automatisierung von Schritten oder Teilschritten
  - Möglichkeit der Parallelisierung von Schritten
  - Möglichkeit der laufenden Kontrolle des Arbeitsfortschritts
- Nachvollziehbarkeit: Jede Weiterleitung oder Datenänderung ist im System dokumentiert, sodass festgestellt werden kann, wer wann welchen Schritt durchgeführt hat, eine Bewilligung erteilt hat etc.
- Möglichkeit zur Optimierung: Die aus den Prozessdurchläufen akkumulierten Daten können ausgewertet werden, um Möglichkeiten zur Optimierung zu finden. Folgende Fragen lassen sich beantworten: Welche Schritte dauern lange, wo gibt es Engpässe, wo gibt es häufige Ausnahmen?

Die Entscheidung über den Einsatz eines BPMS ist natürlich immer eine betriebswirtschaftliche. Entweder der erhoffte Nutzen durch die Kombination der obigen Faktoren ist größer als die Einführungskosten oder die Höhe des potentiellen Verlusts durch Nicht-Einsatz rechtfertigt den Einsatz. Wann kann ein solcher Verlust eintreten: Durch nicht nachvollziehbare Abläufe, Übertretung

von Vorschriften durch falsch durchgeführte Prozesse oder liegengebliebene Prozesse ohne Durchführung von Eskalationsaktionen.

In der Literatur gibt es viele eindrucksvolle Beispiele von Einsparungen durch den Einsatz von Workflowsystemen, zum Beispiel in [?] oder [?].

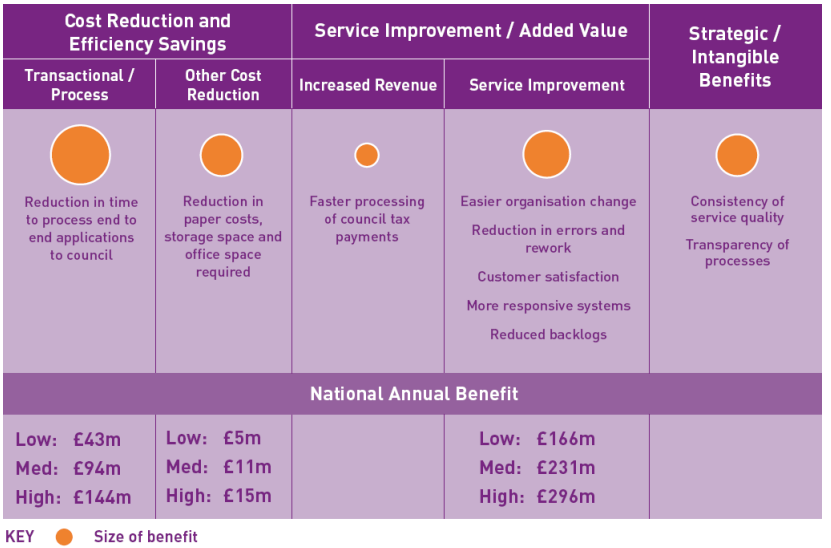


Abbildung 1.3: Vorteile von Workflowmanagement, aus [?]

In einer Studie [?] hat Capgemini im Auftrag der britischen Regierung sechs Workflowprojekte in der öffentlichen Verwaltung untersucht. Wie in Abb. 1.3 ersichtlich, wurden die größten Einsparungen bei der Durchlaufzeit erreicht, aber auch die Servicequalität (Flexibilität, Kundenzufriedenheit) konnte signifikant verbessert werden, die Prozesstransparenz wurde erhöht und weitere Einsparungen bei Papier, Stauraum und Büros waren feststellbar.

Was könnte nach so vielen Vorteilen jemand davon abhalten, Workflowmanagement einzusetzen? Die Bearbeitung von Geschäftsprozessen in Workflowsystemen kann auch zu Problemen führen, in Wikipedia [?] findet sich eine Aufstellung davon. Zusammengefasst enthält sie die folgenden Argumente:

1. „Mitarbeiter verlieren ihre Eigenverantwortung indem sie sich nur an den Workflow halten. Häufig resultiert eine geringere Motivation (Dienst nach Vorschrift).

2. Durch die sich ständig ändernde Geschäftswelt entsteht zwischen Workflow-Modell und der Realität ein permanenter Schlupf. Kreativität und Ideen zur Verbesserung der Geschäftsprozesse werden durch den gegebenen Rahmen eher gebremst.

3. Auf seltene oder nicht vorgesehene Ereignisse kann gar nicht oder nur verzögert reagiert werden.“

Das erste Argument betrifft den Verlust der Kontrolle der einzelnen Mitarbeiter über den Prozess, damit einhergehend Entidentifikation und Motivationsverlust, wie dies bei Fließbandarbeit zu beobachten ist. Es ist einfach eine Managemententscheidung, wer die Kontrolle über einen bestimmten Prozess hat. Und diese Kontrolle muss nicht total sein, da bei der Festlegung des Prozesses Flexibilität bereits eingeplant werden kann, sodass ein Teil der Kontrolle über den Prozess bei den einzelnen Bearbeitern bleibt.

Die Ergonomie der Benutzerschnittstelle des BPMS ist ein wesentlicher Akzeptanzfaktor: Die übersichtliche Präsentation aller für die Bearbeitung notwendigen Daten, die Möglichkeit der Nachverfolgung von Prozessen, die man bearbeitet hat, können Unzufriedenheit mit starrer Prozessabarbeitung aufwiegen. Wichtig ist, den Benutzern wo immer möglich Entscheidungsspielraum zu gewähren: zum Beispiel in der Festlegung der als Nächstes durchzuführenden Aufgabe (wenn mehrere zur Auswahl stehen).

Das zweite Argument betrifft die verlorengehende Agilität. Der Aufwand für Prozessänderungen ist in BPM-Systemen sicher höher als bei der bloßen Festlegung des Prozesses in einem Dokument. Das Argument kann allerdings auch umgedreht werden: Eine Änderung des Prozesses im BPM System bewirkt, dass alle Abläufe *sofort* umgestellt werden – maximale Agilität ist erreicht. Nichtsdestotrotz sind Prozesse, die sich häufig ändern, mit Bedacht zu behandeln. Die oben angesprochene Prozessflexibilität mag auch hier weiterhelfen.

Zu Punkt drei: Das Umgehen mit Ausnahmen und die Notwendigkeit, Abweichungen vom vorgegebenen Prozessweg zu erlauben, sind wichtige Eigenschaften von BPM-Systemen und werden in Abschnitt 3.5 behandelt.

1.1.3 Monitoring und Optimierung

Betrachten wir nun die letzten beiden Schritte des Prozesszyklus, das Monitoring und die Optimierung der Prozesse. Während der Prozessausführung in einem BPMS wird eine große Menge an Daten erzeugt, die erstens der genauen Nachvollziehbarkeit, zweitens der späteren Analyse der Prozesse dienen. Durch Verdichtung dieser Daten können Informationen für Verbesserungen der Prozesse oder der – die Prozesse ausführende – Organisation und Infrastruktur gewonnen werden. Die kontinuierliche Verbesserung der Prozesse ist eine wesentliche Aufgabe des Prozessmanagements.

Soweit die Aufgliederung von Prozessmanagement in Modellierung, Ausführung, Monitoring und Optimierung. Es gibt in der Literatur verschiedene andere Gliederungen der Aufgaben des Prozessmanagements, unter anderem auch eine Reihe von Reifegradmodellen. Wir möchten hier kurz das Business Process Maturity Model (BPMM) [**?**] der Object Management Group vorstellen. Es basiert auf dem älteren, in der Softwareentwicklung verbreiteten, Capability Maturity Model und unterscheidet fünf Ebenen:

1. Initial: Es ist kein Prozessmanagement vorhanden.
2. Managed: Die Prozesse sind lokal definiert und wiederholbar.
3. Standardized: Die Prozesse sind unternehmensweit standardisiert.
4. Predictable: Die Prozesse werden überwacht, Kenndaten gemessen.
5. Innovating: Die Prozesse werden kontinuierlich verbessert.

Vergleichen wir dieses Modell mit dem oben vorgestelltem Zyklus erkennen wir Gemeinsamkeiten: Die Ebenen *Managed* und *Standardized* bedeuten, dass die Prozesse dokumentiert sind (hier etwas feiner aufgegliedert in zwei Ebenen). Die letzte Ebene *Innovating* entspricht der Optimierungsphase. Allein die Ebene *Predictable* ist mit unserer Phase Ausführung nicht ganz deckungsgleich. Im BPMM werden die Ziele und Vorgaben definiert aber keinerlei Aussagen über die konkrete Umsetzung, z.B. IT Unterstützung gemacht. Es wird z.B. gefordert ([**?**], Seite 350):

> Measures of process attributes and performance and quality results emerging from the organization's product and service work are collected on a periodic basis and stored in the organizational measurement repository.

Realistisch sind solche Daten über Prozessabläufe aber nur bei Implementierung des Prozesses in einem BPMS. Somit kommen wir zu einer im Groben ähnlichen Gliederung.

Im folgenden Kapitel sehen wir uns an, welche Prozesse in einem Unternehmen zu finden sind.

1.2 Klassifikationen der Prozesse

Es gibt verschiedene Systeme zur Klassifizierung von betrieblichen Prozessen (siehe [**?**] für eine Übersicht). Gemeinhin werden drei Gruppen von Prozessen unterschieden, siehe Abb 1.4:

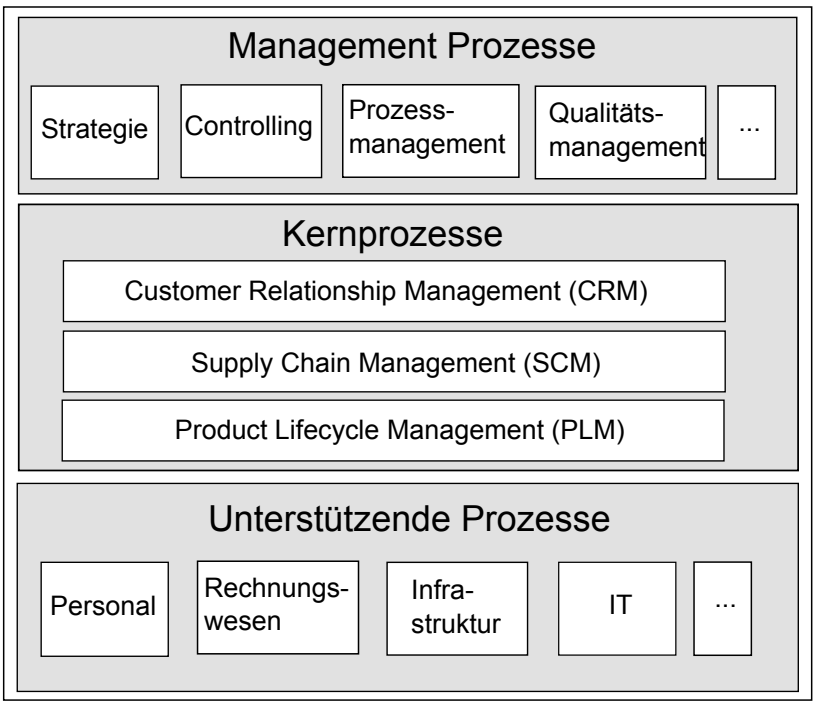


Abbildung 1.4: Die Prozesse im Unternehmen

- Kernprozesse: Diese Prozesse dienen der Erfüllung der Kundenbedürf-

nisse, also der Erreichung des Unternehmenszwecks. Sie sind branchen- und unternehmensspezifisch.

- Managementprozesse sind die Prozesse zur Unternehmensführung, Reorganisation, Qualitätsmanagement.
- unterstützende Prozesse: umfassen die Bereiche Finanzen und Controlling, Personalwesen etc.

Andere Gliederungen unterscheiden zwischen primären Prozessen, das sind die Kernprozesse, und sekundären Prozessen, das sind die Management- und unterstützenden Prozesse. Diese Gliederung vermeidet die etwas unscharfe Unterscheidung zwischen Management- und Unterstützungsprozessen.

Es gibt viele Prozessframeworks, in denen die Prozesse für einen Teilbereich oder eine Branche beschrieben werden, z.B. das ITIL Framework für Prozesse für IT Dienstleistungen [**?**].

Um eine Gesamtübersicht aller Prozesse in einer Organisation zu gewinnen, werden sogenannte Prozesslandkarten, Prozesshandbücher oder Prozessbibliotheken erstellt, die alle definierten Prozesse enthalten.

1.2.1 Kernprozesse

Bei den Kernprozessen geht es um die Prozesse zur Erreichung des Unternehmenszwecks, man nennt sie auch *Primäre Prozesse*. Bei einem Waren produzierenden Unternehmen sind es die Prozesse zu Produktion und Verkauf der Waren. In einem Krankenhaus sind dies die Prozesse zur Behandlung von Patienten.

Eine weitere Untergliederung der Prozesse ist nur möglich, wenn wir uns auf bestimmte Typen von Organisationen einschränken. Zum Beispiel bietet das Siemens Reference Process Framework [**?**] eine Aufstellung von Prozessen für produzierende Unternehmen. Hier unterscheidet man drei Gruppen von Prozessen:

- Product Lifecycle Management (PLM)
- Supply Chain Management (SCM)
- Customer Relationship Management (CRM)

Um Produkte verkaufen zu können, müssen sie zuerst entwickelt werden. Die erste Gruppe von Prozessen beschäftigt sich daher mit der Produktentwicklung (Product Lifecycle Management), wo folgende Teilprozesse zu finden sind:

- ❑ Planung
- ❑ Konstruktion
- ❑ Musterfertigung
- ❑ Produktzulassung

Sind Produktprototypen oder Muster vorhanden, können Produkte für Kunden produziert werden (Supply Chain Management), die Teilprozesse sind folgende:

- ❑ Bestellung
- ❑ Auftragsabwicklung
- ❑ Einkauf
- ❑ Fertigung
- ❑ Lieferung

Die Verwaltung der Kundenbeziehungen (Customer Relationship Management) ist ebenfalls Teil der Kernprozesse:

- ❑ Aquisition von Kunden
- ❑ Angebot/Anfragebearbeitung
- ❑ Marketingaktionen

Für andere Branchen seien einige Beispiele von Kernprozessen in den jeweiligen Organisationen genannt:

- ❑ Banken: Abwicklung von Krediten, Eröffnung eines Kontos oder Sparbuchs
- ❑ Versicherungen: Neuabschluss oder Änderung eines Versicherungsvertrags, Abwicklung eines Schadensfalls

- Behörden: Aktenbearbeitung, Antrag auf einen neuen Pass, Gewerbeanmeldung, Abwicklung eines Bauansuchens

Während die Kernprozesse sehr branchenspezifisch sind, besteht bei den unterstützenden und Managementprozessen größere Übereinstimmung.

1.2.2 Unterstützende Prozesse

Die unterstützenden oder Support-Prozesse können weiter in Gruppen eingeteilt werden:

- Personalprozesse: Verwaltung von Personalangelegenheiten
  - Personalauswahl und Personaleinstellung
  - Vergabe und Entzug von Berechtigungen
  - Urlaubsverwaltung
  - Dienstreise und Reisekosten
  - Spesenabrechnung
  - Abwesenheiten wegen Krankheit, Pflege etc.
  - Projektzeit- und Arbeitszeitabrechnung
  - Ende des Dienstverhältnisses
- Rechnungswesen: Prozesse zur Bearbeitung von Eingangs- und Ausgangsrechnungen, Mahnwesen, Lohnverrechnung.
- Infrastruktur und IT-Prozesse:
  - Incident Management: Behandlung von Reklamationen, Störungsmeldungen
  - Problem Management: Durchführung von Reparaturen, Störungsbehebungen
  - Konfigurationsmanagement
  - Ressourcenbeschaffung und Wartung
  - Lieferantenverwaltung
  - Anwendersupport

1.2.3 Managementprozesse

Die Managementprozesse umfassen die Prozesse:

- Strategie: Festlegung und regelmäßige Überprüfung der Unternehmensstrategie
- Personalplanung: Planen des Personalbedarfs, Definition von Zielen, Schulungsmaßnahmen
- Finanzplanung und Controlling: Finanzbedarf planen, Liquidität sicherstellen, Kapital verwalten und anlegen
- Prozessmanagement: Analyse und Optimierung von Prozessen, Durchführung von Reengineering Maßnahmen
- Risikomanagement: finanzielle Risiken, Risiken in der Produktion etc.
- Qualitätsmanagement: Festlegung von Qualitätskriterien, regelmäßige Überprüfung, Durchführung von qualitätssteigernden Maßnahmen, z.B. Six Sigma Projekten

In dieser Gruppe finden wir auch die Beschäftigung mit den Prozessen selbst. Die Planung, Erfassung und Dokumentation der Geschäftsprozesse sind eine zentrale Managementaufgabe und zwar als abteilungsübergreifende Tätigkeit, die vom Management über die IT bis zu den Mitarbeitern in den einzelnen Fachabteilungen alle betreffen.

1.3 Die Prozessorientierte Organisation - Ein Fallbeispiel

Wie der erfolgreiche und durchgängige Einsatz von BPM aussehen kann, sei am folgenden Fallbeispiel illustriert: Ein Internet-Provider „SuperSurfer“ setzt BPM flächendeckend ein. Die Kernprozesse des Unternehmen sind ebenso wie die Verwaltungsprozesse im BPMS abgebildet. Alle dazu nötigen Daten werden ebenfalls im BPMS oder in einem angeschlossenen Datenbanksystem verwaltet.

Die Kernprozesse dieses Unternehmens sind folgende:

- CRM: Vom ersten Kundenkontakt bis zur Bestellung. Das im BPMS implementierte CRM umfasst die Verwaltung von Kundendaten, Ansprechpartnern und Kundenterminen. Weiters die Bearbeitung von Angeboten, Bestellungen und den damit verbundenen Bestellprozessen.
- SCM: Die Bestellprozesse sind die Schnittstelle zur Produktion. Je nach Bestellung werden die entsprechenden Prozesse gestartet, zum Beispiel „Herstellung eines Internet-Anschlusses“. Die Prozesse bestehen aus interaktiven und automatisierten Schritten. Bei ersteren ist eine manuelle Bearbeitung durch einen Mitarbeiter nötig. Automatische Schritte werden von einem Programm erledigt, entweder wird eine lokal installierte Applikation aufgerufen oder auch ein Web-Service in einer anderen Organisation. Die Produktionsprozesse enden in der Regel mit der Lieferung und Ausstellung einer Rechnung.
- Service: Die weitere Betreuung des Kunden, einerseits die Bearbeitung von Service-Anfragen und Durchführung von Wartungsaufgaben, andererseits weitere Kontaktaufnahme durch den Verkäufer, womit sich der Kreis zur Akquisition schließt.

Die zweite Gruppe von Prozessen umfasst die unterstützenden Prozesse:

- Personalprozesse: Urlaubsantrag, Sonderurlaub, Zeitausgleich, Dienstreise und Reisekosten, Spesenabrechnung, Krankmeldung, Zeiterfassung und Projektzeitabrechnung, Personalauswahl und Einstellung.
- Infrastruktur und IT: Beschaffung von Hard- und Software, Wartung der Infrastruktur.
- Projektmanagement: Durchführung von Projekten, z.B. in den Bereichen Softwareentwicklung, Qualitätssicherung, Prozess-Reengineering.

Die Bandbreite der Prozesse reicht von hochstrukturierten Prozessen, die eng mit eigenen und externen IT-Systemen verknüpft sind, bis zu einfachen wenig strukturierten Prozessen.

Die Vorteile einer vollständigen Abbildung der Prozesse in einem BPMS lassen sich durch einige Beispielszenarien beschreiben:

- Jeder Mitarbeiter findet alle seine Aufgaben elektronisch abrufbar, auch von unterwegs und am Smartphone. Die Aufgaben können individuell in Ordner gegliedert werden, sodass die Übersicht erhalten bleibt.

- Wer ist zuständig: Jeder laufende Prozess ist immer einem oder mehreren Akteuren zugeordnet (letzteres bei paralleler Bearbeitung). Eine Anfrage eines Kunden kann sofort beantwortet werden: In welchem Status befindet sich der Prozess (Bestellung,...), wer ist damit befasst, wann ist mit Fertigstellung zu rechnen.

- Der Terminkalender wird zentral verwaltet. Die Verknüpfung mit den Prozessen erlaubt es, die verschiedenen Arten von Abwesenheiten (Krankheit, Urlaub oder Dienstreisen) gemeinsam mit Terminen aus laufenden Prozessen und prozessunabhängigen Terminen in einer Ansicht darzustellen.

- Alle kunden- und produktionsrelevanten Daten sind in der zentralen Datenbank enthalten: Somit können Service-Mitarbeiter bei einem Problem auf die Kundendaten, Kontakte, laufende Bestellungen, frühere Serviceeinsätze etc. zugreifen und erhalten jeweils aktuelle und umfassende Informationen.

- Controlling: Die Reportingkomponente liefert alle Fakten für das innerbetriebliche Controlling. Fragen wie: Wie sieht die Auslastung der Mitarbeiter aus? Wieviele Bestellungen sind im aktuellen Monat eingelangt, im Vergleich zum Vormonat? können jederzeit beantwortet werden.

- Agilität: Wenn eine Änderung in einem Prozess, z.B. ein zusätzlicher Prüf- oder Bearbeitungsschritt, nötig ist, wird der Prozess einfach angepasst und ab diesem Zeitpunkt wird der neue Schritt im Ablauf berücksichtigt. Die Organisation kann somit auf geänderte Anforderungen schnellstens reagieren. Ebenso sind Änderungen im Datenschema, z.B. Hinzufügen eines Feldes bei den Kundendaten, leicht durch einen Administrator durchführbar.

In den nächsten Kapiteln werden die wesentlichen Schritte zur Implementierung einer prozessorientierten Organisation behandelt.

Kapitel 2

Modellierung von Prozessen

Die Modellierung eines Prozesses ist die Abbildung der Informationen über den Prozess in ein Modell. Zu den Informationen gehören die Struktur der Daten, die am Prozess Beteiligten, die Definition der einzelnen Prozessschritte und der Ablauf des Prozesses. Verkürzt kann man es mit folgendem Satz ausdrücken:

> *Wer* macht *was wann womit*.

Das heißt, es wird definiert, welche Akteure welche Aktivitäten, mit welchen Daten, in welcher Abfolge ausführen. Bevor wir die konkreten Schritte zur Modellierung erläutern, müssen wir uns fragen, wofür die Modellierung durchgeführt werden soll: Im vorigen Kapitel wurde der Zyklus des Prozessmanagements dargestellt, mit der Modellierung als erstem Schritt. Ziel der Modellierung ist dabei die spätere Ausführung in einem BPMS. Es könnte allerdings auch sein, dass Prozesse modelliert werden, die nicht oder nicht sofort im BPMS ausgeführt werden sollen. Gründe dafür sind z.B.:

- der Prozess eignet sich nicht für die Ausführung in einem BPMS, z.B. weil er Tätigkeiten beschreibt, die eine einzelne Person durchführt.
- der Prozess ist noch nicht genau genug spezifiziert. Die Beschreibung dient einstweilen als Richtlinie für die Prozessbearbeitung, die Details müssen noch formalisiert werden.

In beiden Fällen kann es sinnvoll sein, den Prozess trotzdem zu modellieren,

damit das daraus entstehende Prozesshandbuch eine vollständige Dokumentation der betrieblichen Abläufe darstellt.

Der Unterschied in der Modellierung ist der nötige Detaillierungsgrad: Für ausführbare Prozesse müssen alle Definitionen der Daten, Bedingungen und Programmaufrufe vollständig sein, für nicht ausführbare Prozesse ist dies nicht notwendig.

Bevor wir uns der eigentlichen Prozessmodellierung zuwenden, betrachten wir die Umgebung, in der die Prozesse ablaufen, die Aufbauorganisation bzw. Organisationsstruktur. In den darauffolgenden Abschnitten werden die Prozessabläufe definiert, zuletzt die Prozessdaten.

2.1 Definition der Organisationsstruktur

Um Geschäftsprozesse definieren zu können, ist zunächst die Definition der Elemente der Organisation nötig: die Organisationseinheiten, Personen und Rollen.

2.1.1 Organisationseinheiten

Eine Organisation (Behörde, Firma etc.) ist normalerweise in Einheiten gegliedert. Beispiele sind: Buchhaltung, Personalabteilung, Produktion.

Die Organisationseinheiten können typisiert werden, z.B. gibt es in einer Universität Institute, Abteilungen, Fakultäten etc. Je nach Typ der Organisation sind dort unterschiedliche Rollen belegbar. Zum Beispiel gibt es in einer Fakultät einen Dekan, in einem Institut einen Institutsvorstand. Wir bezeichnen diese Typen von Organisationen als *Organisationsklassen*.

Organisationshierarchie

Die einzelnen Organisationseinheiten sind typischerweise in einer Hierarchie - einer Baumstruktur - angeordnet. Eine Abbildung dieser Baumstruktur im BPM-System ist insofern wichtig, weil an Abläufen oft über- und untergeordnete Organisationen beteiligt sind.

Zwischen der Aufbauorganisation und der Prozessstruktur eines Unternehmens gibt es ein gewisses Konfliktpotential. Wenn Prozesse organisationsübergreifend ausgeführt werden, wer hat dann die Verantwortung für den

Prozess? Als Kompromisslösung wurde die *Matrixorganisation*, in der die Prozesse als Organisationseinheiten definiert werden, vorgeschlagen. In einer reinen *Prozessorganisation* wird die ursprüngliche Aufbauorganisation komplett aufgegeben und durch die Geschäftsprozesse ersetzt. Siehe z.B. [**?**] für eine Diskussion dieser Problematik.

2.1.2 Personen

Die Personen, die das BPMS benutzen sollen, müssen im System abgebildet werden. Im Allgemeinen werden der Name, eine eindeutige Id, sowie weitere Eigenschaften, wie E-Mail, Telefonnummer, Adresse etc. erfasst.

Für die Modellierung der Prozesse ist es nicht nötig, die einzelnen Personen zu erfassen (wohl aber für die Ausführung). Die Prozesse sollen unabhängig von den konkreten Personen sein, als Akteure werden sogenannte *Rollen* verwendet.

2.1.3 Rollen

Rollen definieren eine Funktion oder Fähigkeit, z.B. Sekretär, Abteilungsleiter, Spanischsprechender. Sie können außerdem eine Menge von Rechten zusammenfassen, z.B. erlaubt die Rolle Systemadministrator die Durchführung von verschiedenen privilegierten Operationen.

Neben den in der Organisation bereits vorhandenen Rollen gibt es solche, die für einen Geschäftsprozess oder eine Applikation eigens definiert werden müssen. Erstere sind z.B. Rollen wie Abteilungsleiter, Assistent, Software-Entwickler. Rollen für einen bestimmten Prozess fassen abstrakt die Gruppe von Personen zusammen, die eine bestimmte Aufgabe in einem Prozess wahrnehmen, z.B. steht die Rolle „Genehmiger“ für alle Personen, die den Genehmigungsschritt in einem bestimmten Prozess ausführen dürfen.

Rollen unterscheiden sich weiters in ihrer Beziehung zur Organisation: Es gibt Rollen, die gelten für eine Organisationseinheit, z.B. Institutsvorstand ist jemand für ein bestimmtes Institut. Wir bezeichnen solche Rollen als *lokale* Rollen.

Andere Rollen gelten für eine Organisationseinheit und alle in der Hierarchie darunterliegenden Organisationseinheiten. Der Dekan ist Dekan der Fakultät und somit auch für alle in dieser Fakultät enthaltenen Teilorganisationen. Diese Rollen nennen wir *hierarchische* Rollen. Die dritte Form von Rollen gilt unab-

hängig von der Organisation. „Spanischsprechender“ ist jemand unabhängig von seiner Position. Wir bezeichnen diese Rollen als *globale* Rollen. Wenn Rollen nun Benutzern zugeordnet werden, ist bei lokalen und hierarchischen Rollen eine Organisationseinheit anzugeben, bei globalen Rollen nicht.

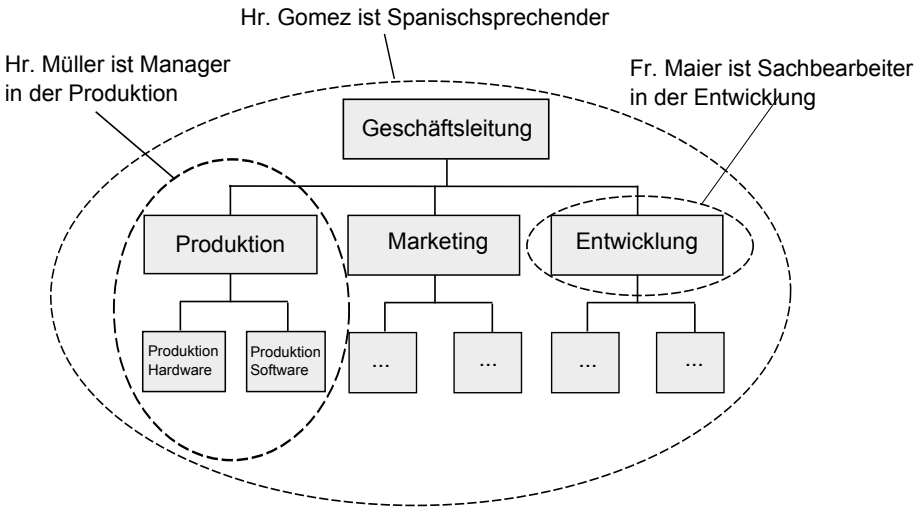


Abbildung 2.1: Organisation und Rollen

Die Abbildung 2.1 zeigt den Gültigkeitsbereich der verschiedenen Rollenzuordnungen. Die Rolle „Spanischsprechender“ ist eine globale Rolle, somit ist der Gültigkeitsbereich der Rollenzuordnung die gesamte Organisation. „Sachbearbeiter“ ist eine lokale Rolle, somit gilt eine Rollenzuordnung zu einer Organisationseinheit nur in dieser. Die Rolle „Manager“ ist hierarchisch, somit gilt sie in der zugeordneten Organisationseinheit und in allen hierarchisch untergeordneten Einheiten.

Abb. 2.2 zeigt ein Schema zur Definition einer einfachen Organisationsstruktur. Wir finden darin die genannten Klassen Benutzer, Rolle und Organisationseinheit. Die Klasse Rollenzuordnung definiert die Zuordnungen von Rollen zu Benutzern, eine 1:n Relation zwischen Organisationseinheiten repräsentiert die Organisationshierarchie.

Benutzer könnnen unterschiedliche Rollen in unterschiedlichen Organisationseinheiten haben, es kann aber Einschränkungen geben:

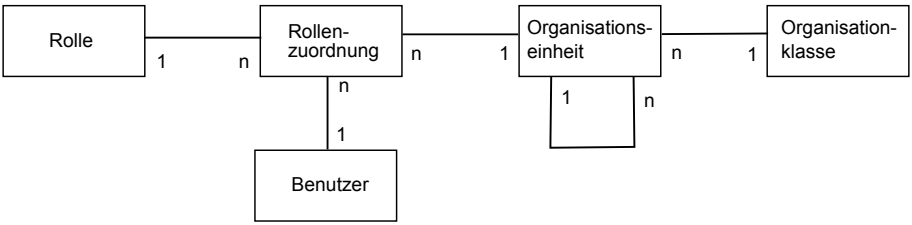


Abbildung 2.2: Schema der Organisationsdaten

- Bestimmte Rollen werden pro Organisationseinheit nur einmal vergeben: Dies ist für Rollen, die im Sinne von *Stellen* verwendet werden, oft der Fall (es kann keine zwei Vorstände für ein Institut geben).

- Ein Benutzer kann eine Rolle nur einmal haben: Dies ist bei globalen Rollen der Fall oder wiederum bei Stellen (jemand kann nicht Vorstand von zwei Instituten sein).

Die Möglichkeit, einem Benutzer Rollen in unterschiedlichen Organisationseinheiten zuzuordnen, erlaubt die Modellierung von sogenannten Matrixorganisationen. Dabei gibt es neben den Elementen der Aufbauorganisation weitere Organisationseinheiten für Projekte, denen Benutzer aus verschiedenen OEs der Aufbauorganisation zugeordnet sind. Zum Beispiel können in einer Projektgruppe für ein Produkt Mitarbeiter vom Marketing, dem Verkauf und der Entwicklung beteiligt sein.

Zwei weitere Aspekte der Organisationsmodellierung, nämlich Vertretungen und Berechtigungen, werden wir im Kapitel 3, Prozessausführung, behandeln.

2.2 Ablaufmodellierung

Bevor wir uns der Prozessmodellierung zuwenden, müssen noch einige Begriffe geklärt werden:

- Prozessdefinition: ist die Abbildung eines Geschäftsprozesses, z.B. Dienstreise-Abwicklung, in ein Modell.

- Prozessinstanz: ist ein einzelner Fall, eine konkrete Ausprägung einer Prozessdefinition, z.B. die Dienstreise von Hrn. Huber am 22. 3. 2010 nach Wien.
- Aktivität: ist ein Schritt in einer Prozessdefinition, dieser kann auch zusammengesetzt, d.h. wieder ein Prozess, sein.
- Task: ist ein elementarer Schritt im Prozess.
- Applikation: Menge von Prozessdefinitionen, die thematisch zusammengehören.

Die Begriffe orientieren sich an dem Glossar der Workflow Management Coalition [**?**].

2.2.1 Auswahl des Formalismus

Die erste Überlegung bei der Modellierung von Prozessen betrifft die Wahl eines Modellierungsformalismus. Die Disziplin ist nicht mehr ganz jung und deshalb gibt es bereits eine Reihe von Formalismen aus unterschiedlichen Quellen, wir wollen nur einige nennen:

- Formale Methoden: Unter den formalen Methoden zur Prozessbeschreibung haben sich vor allem Petri-Netze durchgesetzt, eine Prozessmodellierungssprache, die darauf aufbaut ist z.B. YAWL [**?**].
- Methoden aus Produkten: ARIS, ein sehr weitverbreitetes Tool zur Prozessmodellierung (der Software AG), verwendet die sogenannten Ereignisgesteuerte Prozessketten (EPK).
- Standards: Die Business Process Model and Notation (BPMN) [1] [**?**] der Object Management Group (OMG) hat sich in den letzten Jahren als Standard durchgesetzt und die meisten Produkte unterstützen nun diese Notation - oder eine Teilmenge davon.

  Die sehr umfangreiche Modellierungsmethode UML ermöglicht es mit ihren Activity Diagrams, Prozesse zu modellieren, diese Methode ist allerdings gegenüber BPMN bei der praktischen Verwendung ins Hintertreffen geraten.

[1] BPMN stand urspünglich für Business Process Model*ing* Notation, seit Version 1.2 wird das Model wichtiger genommen und es heißt nun Model *and* Notation.

> Die Business Process Execution Language (BPEL) [**?**] ist eine Sprache zur Definition von Prozessen, die weitgehend ohne interaktive Schritte ablaufen, also hauptsächlich Programme miteinander verbinden („orchestrieren“). Von einigen großen Herstellern propagiert, gegenüber BPMN allerdings in letzter Zeit an Bedeutung verloren, vor allem seit mit BPMN 2.0 eine Ausführungssemantik definiert wurde.
>
> BPEL4People [**?**] ist eine BPEL Ergänzung für interaktive Schritte, die Standardisierung ist seit längerem in Arbeit.

Was alle diese Methoden (außer BPEL) gemeinsam haben, ist, dass die Modellierung grafisch erfolgt, somit das resultierende Prozessmodell eine Grafik ist. Die Vorteile der graphischen Darstellung ist, dass Diagramme relativ leicht zu verstehen und kommunizieren sind.

Ein Nachteil ist, dass die für die Implementierung der Prozesse nötigen Details in der Graphik nicht wiedergegeben werden und nur in zusätzlichen textuellen Repräsentationen aufscheinen. Denn - und das ist das Dilemma der Prozessmodellierung - das Modell soll zwei Zwecken dienen: einerseits eine allen Prozessbeteiligten verständliche Repräsentation des Prozesses darstellen, andererseits eine Ausführung in einem BPMS ermöglichen.

Der zweite Nachteil dieser grafischen Modellierungsmethoden ist, dass die Notationen wenig restriktiv sind und es möglich ist, eine Menge Unsinn damit zu zeichnen: unstrukturierte, niemals terminierende Prozesse oder Prozesse die irgendwo steckenbleiben, weil eine Kante vergessen wurde. Das merkt man aber erst, wenn der Prozess in einem BPMS zum Laufen gebracht werden soll. Bezeichnend ist das Zitat aus der Definition von BPMN ([**?**], Seite 439):

> Not all BPMN orchestration Processes can be mapped to WS-BPEL in a straight-forward way. That is because BPMN allows the modeler to draw almost arbitrary graphs to model control flow, whereas in WS-BPEL, there are certain restrictions such as control-flow being either block-structured or not containing cycles.

Hier wird daher ein etwas anderer Ansatz verfolgt: Die Beschreibung eines Prozessablaufs entspricht der Beschreibung eines Algorithmus. Für die Beschreibung von ausführbaren Algorithmen wurden Programmiersprachen erfunden. Es gibt verschiedene etablierte Sprachfamilien: prozedurale, objektorientierte, funktionale und logische. Für die Definition von Geschäftsprozessen ist der prozedurale Ansatz der naheliegendste, weil eine grafische Darstellung leicht

möglich ist und die einzelnen Kontrollstrukturen am leichtesten zu vermitteln sind.

Einen Geschäftsprozess in einer Programmiersprache zu formulieren ist aber nicht zweckmäßig, weil die Modellierung für alle Beteiligten illustrativ sein soll: Es sollen auch Personen ohne Programmierkenntnisse und ohne Interesse an den technischen Details die Prozesse verstehen und an ihrer Definition mitwirken können. Eine mögliche Lösung ist daher: Die Kontrollkonstrukte von strukturierten Programmiersprachen grafisch darstellen, wie in der grafischen Programmierung (Englisch: Visual Programming Language) üblich [**?**].

Der im folgenden verwendete Ansatz ist daher die Definition einer prozeduralen Scriptsprache für Geschäftsprozesse mit einer korrespondierenden grafischen Darstellung in der Notation BPMN [**?**]. Die textuelle Notation nennen wir Workflow Definition Language (WDL).

Die nötigen Kontrollstrukturen Sequenz, Alternative, Schleife, Parallelität und Ereignisbehandlung sind in BPMN darstellbar, ebenso die Definition der atomaren Aktivitäten. Es werden nicht alle BPMN Konstrukte verwendet, sondern nur eine Teilmenge davon und in einer strikteren Struktur: Jeder modellierte Prozess ist ein korrekter BPMN Prozess, aber nicht jeder in BPMN modellierter Prozess entspricht unseren Strukturvorgaben.

Im folgenden werden wir die BPMN Notation gemeinsam mit den Sprachelementen von WDL einführen. Die atomaren Elemente des grafischen Prozessmodells sind Knoten und Kanten. Bei den Knoten gibt es Aktivitäten, Ereignisse und Gateways. Bei den Kanten wird nur die Kontrollkante der BPMN Notation verwendet.

2.2.2 Aktivitäten

Betrachten wir zuerst die elementaren Aktivitäten, von denen es mehrere Ausprägungen gibt.

Manuelle Aktivitäten

bezeichnen wir als Tasks. Sie werden von Personen (in der Modellierung meist durch Rollen repräsentiert) ausgeführt und in der Prozessmodellierung nicht weiter zerlegt. Jeder Task hat zumindest eine eindeutige Id und einen Namen. Der Name muss nicht eindeutig sein, kann sich also innerhalb des Prozesses oder in anderen Prozessen wiederholen.

Abb. 2.3 zeigt die grafische Darstellung. In dem Rechteck ist links oben ein Personen-Icon dargestellt. Als Text wird der Akteur gefolgt vom Namen des Tasks angegeben.

```
Leiter Genehmigen();
```

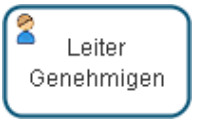


Abbildung 2.3: Manuelle Aktivität

In der WDL Notation wird eine interaktive Aktivität folgendermaßen beschrieben:

```
agent task "(" [ form {"," form } ] ")" [ stepname ] ";"
```

Auf die Definition des Akteurs folgt die Id des Tasks und danach in den Klammern die Liste der verwendeten Prozessdaten. Wie die Akteure und die Prozessdaten definiert werden, darauf wird weiter unten noch genauer eingegangen. Einstweilen nehmen wir hierfür einfach eine Rollen-Id bzw. die leere Liste an.

Automatische Aktivitäten

werden von einem Programm ausgeführt. Wir unterscheiden synchrone und asynchrone Bearbeitung.

(a) Synchron: Während des Ablaufs werden irgendwelche Berechnungen oder Programme ausgeführt, auf eine Antwort von einem Fremdsystem braucht nicht gewartet werden. Abb. 2.4 zeigt die Darstellung in BPMN und WDL.

```
system send.result()
      "Ergebnisse versenden";
```

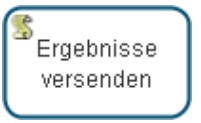


Abbildung 2.4: Systemschritt

Der Unterschied in der Notation zur manuellen Aktivität ist die Schriftrolle statt der Person links oben. In WDL folgt auf das Schlüsselwort `system` ein Name einer Java Methode, gefolgt von 0 bis n String Argumenten. Ein Name des Systemschritts kann optional angegeben werden:

```
"system" methodname([ arg {"," arg } ] ")" [ stepname ] ";"
```

(b) Asynchrone Bearbeitung: Es wird ein Fremdsystem kontaktiert und der Prozess läuft erst weiter, wenn die Kommunikation abgeschlossen ist. In der grafischen Darstellung stehen nun Zahnräder, in WDL wird das Schlüsselwort `batch` verwendet, auf das der Name einer Klasse folgt, welche die Kommunikationsdetails implementiert, siehe Abb. 2.5.

```
batch demo.batch.Main
 "Verarbeitung in externem System";
```

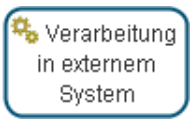


Abbildung 2.5: Batchschritt

Subprozesse

Zur übersichtlicheren Darstellung von größeren Prozessen dienen Subprozesse. Dabei wird ein zusammenhängender Teil eines Prozesses aus diesem entfernt und in einem eigenen Subprozess definiert. Im ursprünglichen Prozess verbleibt ein Aufruf dieses Subprozesses. Subprozesse dienen auch dazu, wiederkehrende Prozessteile nur einmal zu definieren.

Subprozesse werden in WDL mit dem Schlüsselwort `call` eingeleitet.

```
"call" processname ([ arg {"," arg } ] ")" [ stepname ] ";"
```

Im dazugehörigen BPMN Symbol befindet sich unten in der Mitte ein +. Dies soll symbolisieren, dass sich hinter diesem Symbol weitere Struktur verbirgt.

```
call Genehmigung();
```

Abbildung 2.6: Subprozess

2.2.3 Kontrollstrukturen

Die Komposition der Einzelaktivitäten durch Kontrollstrukturen zu einem Ablauf ergibt einen Prozess. In BPMN werden für die Kontrollstrukturen eigene Knotentypen definiert. Die Kanten zwischen den Knoten repräsentieren den Kontrollfluss.

Abb. 2.7 zeigt einen einfachen, kompletten Prozess. Jeder Prozess beginnt mit einem Startknoten und endet mit einem Endknoten – grafisch unterscheiden sie sich dadurch, dass der Endknoten einen dickeren Rand hat.

```
begin
 Role1 Task1();
 Role2 Task2();
end;
```

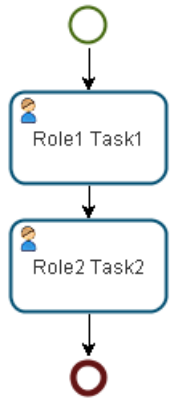


Abbildung 2.7: Sequenz

Gerichtete Kanten verbinden die einzelnen Knoten und definieren damit den Kontrollfluss. Der minimale, leere Prozess besteht aus dem Startknoten, dem Endknoten und eine beide verbindende Kante. Das Einfügen von weiteren Konstrukten erfolgt nun immer so, dass eine Kante ersetzt wird durch ein Konstrukt inklusive der hinführenden und wegführenden Kante. In Abb. 2.8 ist eine solche Transformationsregel dargestellt.

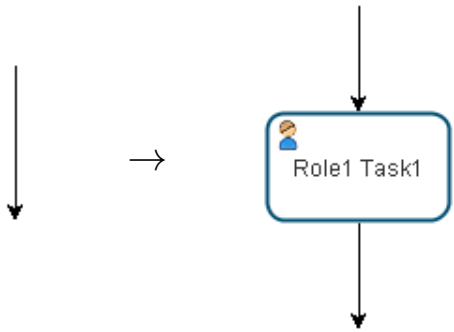


Abbildung 2.8: Regel der Graph-Grammatik

Jede der nun folgenden Kontrollstrukturen wird auf diese Weise in den Prozess eingefügt. Ein gültiger Prozessgraph in unserer Notation ist jeder Graph, der aus dem initialen Graphen durch wiederholte Anwendung dieser Transforma-

tionsregeln erzeugt werden kann. Die Menge dieser Regeln bezeichnet man als Graph-Grammatik. Diese Vorgangsweise hat zur Folge, dass alle erzeugten Graphen in die Scriptsprache WDL übersetzt werden können. Beide Notationen sind daher von ihrer Semantik äquivalent, die grafische Notation enthält aber Zusatzinformationen über Positionierung und Größen von Aktivitäten und Kanten. Eine vollständige Definition dieser Graph-Grammatik erfolgt in Abschnitt 2.2.11. Der Prozesseditor von @enterprise ist gemäß den Regeln der Graph-Grammatik implementiert, sodass es damit nur möglich ist, syntaktisch korrekte Prozessgraphen zu erstellen, siehe Abschnitt 2.7.2.

Im folgenden betrachten wir die einzelnen Kontrollstrukturen.

2.2.4 Sequenz

Eine Sequenz ist die Abfolge von zwei (oder mehr) Schritten hintereinander. In WDL schreibt man einfach zwei Aktivitäten hintereinander. In der Grafik sind die Aktivitäten durch eine gerichtete Kante verbunden.

In der Abbildung 2.7 ist ein ganzer Prozess dargestellt, beginnend mit einem Anfangsknoten, gefolgt von zwei Aktivitäten, die hintereinander ausgeführt werden, gefolgt vom Endknoten. In WDL sind Anfangs- und Endknoten durch die Schlüsselwörter `begin` und `end` repräsentiert.

2.2.5 Alternativen

Es wird zwischen zwei oder mehreren alternativen Ausführungspfaden *einer* ausgewählt. Wir unterscheiden, ob diese Auswahl durch das System oder manuell erfolgt. Ersteres nennen wir *If*, zweiteres *Choice*.

If

Die Auswahl eines Ausführungspfades erfolgt durch Auswertung einer Bedingung. Die WDL-Notation mit if-then-else entspricht der aus Programmiersprachen bekannten. In der grafischen Notation führen zwei Kanten vom Bedingungsknoten (einer Raute) weg, eine Kante führt zu dem Zweig, der ausgeführt wird, wenn die Bedingung erfüllt ist, die zweite, jene mit dem 45° Querstrich, zum „else“ Zweig (Im BPMN Standard bedeutet dieser Querstrich, dass dies der Default-Weg ist).

In einer farbigen Darstellung wird erstere Kante grün geeichnet (Bedingung erfüllt), die zweite in rot. Die beiden Zweige werden wieder in einer Raute zusammengeführt.

```
if (f.x = 1) then
  Rolle1 Task1();
else
  Rolle2 Task2();
end;
```

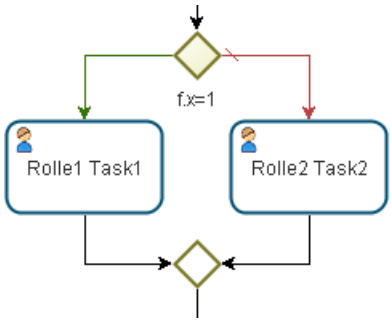


Abbildung 2.9: If

Generell gilt: Von einem Knoten führen entweder ein oder mehrere schwarze Kanten *oder* eine grüne *und* eine rote Kante weg. Im zweiten Fall ist der Ausgangsknoten ein Bedingungsknoten und je nach Auswertung der Bedingung wird einer der beiden Wege gewählt.

Mehrfachverzweigungen können mit elsif modelliert werden, in der grafischen Darstellung laufen alle Zweige zu einem Endknoten zusammen, siehe Abb. 2.10.

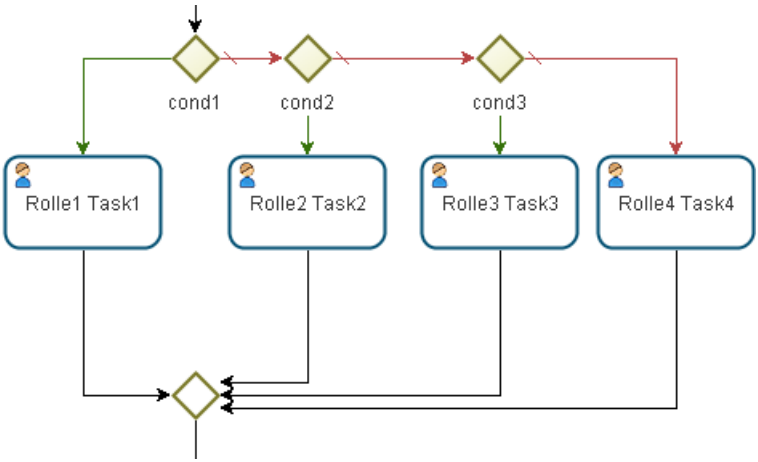


Abbildung 2.10: if .. elsif .. else

Um ein *if* vollständig zu spezifizieren, sodass der Prozess ausführbar wird, ist die Formalisierung der Bedingung nötig. Dazu verwenden wir eine einfache Expression-Syntax oder XPath, eine genaue Spezifikation erfolgt in Kapitel 2.5.

Choice

Die zweite Möglichkeit einen alternativen Pfad auszuwählen, ist die manuelle Auswahl zur Ausführungszeit: Der Bearbeiter des aktuellen Schritts legt beim Weiterleiten fest, welchen Pfad der Prozessablauf beschreitet. Es folgen also dem choice Knoten *n* Nachfolger. Jeder Zweig kann optional mit einer Bedingung versehen werden. Der Benutzer kann unter den Zweigen ohne Bedingung und jenen, deren Bedingung erfüllt ist, auswählen.

In der grafischen Notation, siehe Abb. 2.11, wird für die Choice der BPMN Zwischenereignis-Knoten verwendet, grafisch dargestellt durch einen Kreis mit doppeltem Rand. Der Knoten, der das Konstrukt einleitet, enthält ein Fünfeck, dies symbolisiert, dass auf eines von mehreren Ereignissen gewartet wird. Die einzelnen Pfade beginnen mit einem Knoten, der ein Dreieck enthält. Das bedeutet, es wird auf *ein* Ereignis gewartet. Unter diesen Knoten stehen die Bedingung und die Auswahltexte. Die einzelnen Pfade werden wieder in einem als Raute dargestellten Knoten zusammengeführt.

```
choice
 "direkt bestellen",
  f.amount < 500:
  Assistent Bestellen(f);
 "nochmals prüfen":
   Manager Prüfen(f);
 "nicht bestellen":
  system test.ablegen()
    "Ablegen";
end;
```

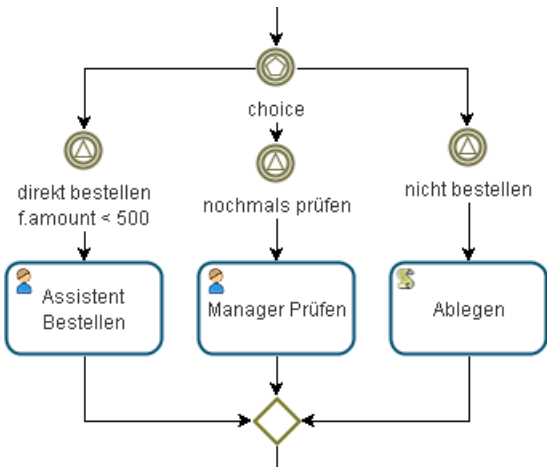


Abbildung 2.11: Choice

Zur Laufzeit können folgende Sonderfälle auftreten: Kein Choice Zweig ist auswählbar, weil alle Bedingungen *false* ergeben: In diesem Fall tritt beim Erreichen des Choice ein Fehler auf. Wenn nur eine Auswahlmöglichkeit vorhanden ist, erscheint die Auswahlmaske nicht und es wird sofort zum Nachfolgeschritt weitergeleitet.

2.2.6 Wiederholungen

Zur Darstellung von sich wiederholenden Abläufen dienen Schleifen. Dabei werden die Aktionen in der Schleife (Schleifenrumpf) immer wieder ausgeführt, solange oder bis eine Bedingung erfüllt ist. Es gibt mehrere Typen von Schleifen, siehe Abb. 2.12. Bei der *While* Schleife wird die Bedingung überprüft, bevor die in der Schleife vorkommenden Anweisungen einmal durchgeführt werden. Das Ende des Schleifenrumpfs zeigt wieder zum Bedingungsknoten - es gibt also bei diesem Konstrukt keinen eigenen Endknoten.

Bei der *Repeat* Schleife werden die Anweisungen zumindest einmal ausgeführt: Der erste Knoten, durch eine Raute dargestellt, verzweigt in den Schleifenrumpf, nach diesem erfolgt die Bedingungsprüfung. Wenn die Bedingung wahr ist, wird die Schleife verlassen, sonst wird der Weg zurück zum Schleifenbeginn verfolgt.

```
while (f.x = 0) do
  Rolle1 Task1();
  Rolle2 Task2();
end;
```

```
repeat
  Rolle3 Task3();
until (f.x = 0);
```

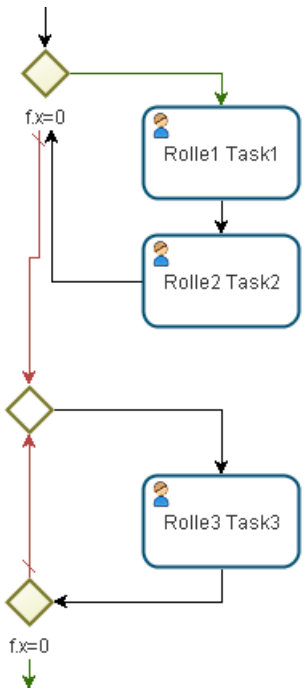


Abbildung 2.12: While- und Repeat-Schleife

Bei der dritten Schleifenvariante, dargestellt in Abb. 2.13, wird die Bedingung weder am Anfang noch am Ende sondern irgendwo im Schleifenrumpf über-

prüft. Die grafische Repräsentation ist wie bei einer Repeat-Schleife, nur dass die Kante von der Bedingung zum Loop-Knoten nicht leer ist. In WDL steht an der Stelle der Bedingungsprüfung ein `exit when`.

```
loop
  Rolle1 Task1();
  exit when (f.finished = 1);
  Rolle2 Task2();
end;
```

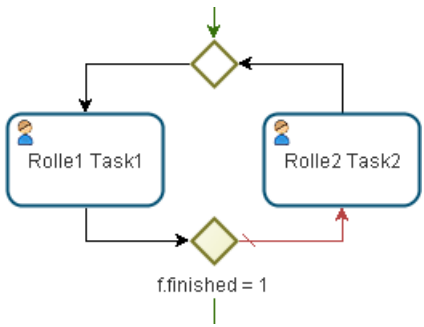


Abbildung 2.13: Allgemeine Schleife

2.2.7 Parallelität

Der Prozess verzweigt sich in beliebig viele parallele Zweige, welche wieder zu einem Knoten zusammenlaufen. Hier gibt es mehrere Varianten:

1. Die Ausführung nach der Parallelität wird fortgesetzt, wenn *alle* Zweige beendet sind. Wir nennen dies *Andpar*.
2. Die Ausführung wird fortgesetzt, wenn der *erste* Zweig beendet ist, dies bezeichnen wir als *Orpar*.
3. Die Ausführung wird fortgesetzt, wenn m aus n Zweigen beendet sind. $1 \leq m \leq n$. n ist die Anzahl der Zweige, m wird vom Modellierer angegeben.
4. Die Ausführung wird fortgesetzt, wenn m aus n Zweigen beendet sind, wobei das m erst zur Laufzeit ermittelt wird.

Bei den Fällen 2 bis 4 muss zusätzlich festgelegt werden, was mit den Aktivitäten in den noch nicht abgeschlossenen Zweigen passiert: abbrechen oder weiterlaufen lassen.

Abbildung 2.14 zeigt die Syntax und grafische Darstellung des Andpars, Abb. 2.15 zeigt das Orpar. Der erste Knoten der Parallelität wird durch eine Raute mit einem + dargestellt. Dies bedeutet, dass alle ausgehenden Pfade weiterverfolgt

werden. Beim Knoten mit dem * laufen die Zweige wieder zusammen, es wird die Bedingung für die Fortführung des Ablaufs nach der Parallelität angegeben. In der WDL werden die einzelnen Zweige durch einen vertikalen Strich | getrennt.

```
andpar
    Rolle1 Task1();
 |  Rolle2 Task2();
end;
```

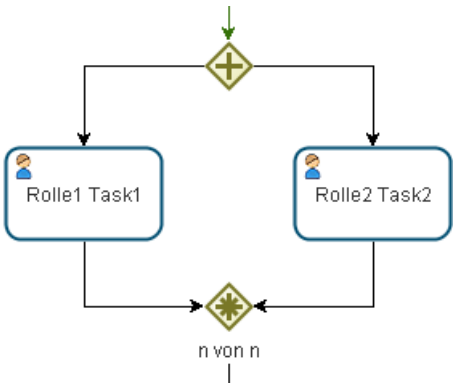


Abbildung 2.14: Andpar

```
orpar
    Rolle1 Task1();
 |  Rolle2 Task2();
end;
```

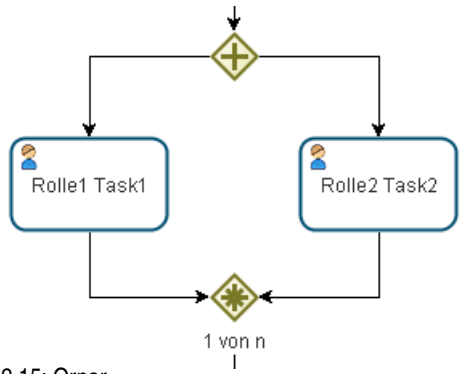


Abbildung 2.15: Orpar

Die Fälle 3 und 4 werden wie Andpar modelliert, allerdings wird nach dem end eine Methode angegeben, die entscheidet, ob die Parallelität beendet ist oder ob auf das Fertigstellen weiterer Zweige gewartet werden muss. Die Methode wird nach jeder Beendigung eines Zweiges aufgerufen.

Der Fall einer dynamischen Berechnung der Anzahl der zur Beendigung nötigen Zweige ist für die Praxis relevanter als die Fälle 2 und 3. Man betrachte die folgenden Beispiele:

- Ein Antrag wird an drei Entscheider weitergeleitet, alle müssen zustim-

men. Sobald einer eine negative Entscheidung abgibt, können die anderen Zweige abgebrochen werden.

- Sobald die Hälfte der Akteure entweder dafür oder dagegen ist, kann der Prozess so oder so fortgesetzt werden (einfache Mehrheit).

In der graphischen Notation ist die Bedingung nicht sichtbar und muss entweder im Label des Parallelität-Endknotens oder durch eine Annotation vermerkt werden.

Parfor

Der *Parfor* Konstrukt ist eine weitere Variante der Parallelität, und zwar für die parallele Ausführung von n gleichartigen Zweigen, wobei die Anzahl der Zweige erst zur Ausführungszeit bekannt ist.

Es gibt dabei zwei Möglichkeiten: 1. Die Anzahl der Zweige wird in den Daten modelliert oder 2. die Anzahl der Zweige wird dynamisch durch einen Funktionsaufruf ermittelt.

```
parallel for review in mainform.1 do
   review.agent Task1(review);
end;
```

Abbildung 2.16: Parfor

Das in Abb. 2.16 dargestellte Parfor ermittelt zuerst die Anzahl der Datenelemente (etwa Zeilen in einem Formularblock). Danach werden so viele gleichartig strukturierte parallele Zweige gleichzeitig erzeugt und aktiviert. In jedem der Zweige wird eines der Dokumente bearbeitet.

In der grafischen Notation wird das Parfor als ein Knoten dargestellt, der einen Teilprozess enthält. Die parallele Ausführung wird durch die drei Striche am unteren Rand des Rahmens dargestellt.

Die Prozessausführung wird fortgesetzt, wenn alle Zweige beendet sind. Durch Angabe einer Funktion kann aber auch wie bei Parallelität, Fall 4, dynamisch über die Prozessfortsetzung entschieden werden.

Abzweigung

Eine weitere Form der Parallelität ist die Abzweigung (engl.: *branch*). Vom Prozessfluss spaltet sich ein Ausführungszweig ab, der unabhängig vom Rest des Prozesses abläuft und endet, d.h. es kommt zu keiner Synchronisation mehr. Verwendet wird diese Kontrollstruktur z.B. für Informationstasks, eine Person wird über den Stand des Prozesses informiert, greift in den Ablauf aber nicht ein.

```
branch
   Role1 Task1();
end;
```

Abbildung 2.17: Branch

Die graphische Darstellung beginnt mit einem Gateway-Knoten (wie bei Andpar und Orpar). Aus diesem Knoten führen zwei Kanten. Die Kante in den Branch ist blau eingefärbt (in einer farbigen Darstellung), außerdem ist der Endknoten, der den Branch abschließt, blau dargestellt.

2.2.8 Synchronisation durch Ereignisse

Wenn mehrere parallele Ausführungspfade in einem Prozess vorkommen, ist es oft nötig diese zu synchronisieren, z.B. wenn ein Schritt nicht beginnen darf, bevor ein anderer fertig gestellt ist.

Um solche Prozesse zu modellieren, gibt es die beiden Konstrukte *RaiseEvent* und *Sync*.

Mit RaiseEvent wird ein Ereignis ausgelöst, mit Sync kann auf ein Ereignis gewartet werden. Abb. 2.18 zeigt ein Beispiel dazu: In einer Parallelität soll der `task3` erst begonnen werden, wenn der dazu parallele Task `task1` abgeschlossen ist.

Die Anweisung RaiseEvent hat die folgenden Parameter: Der erste definiert eine Event-Id, diese Id muss beim Sync ebenfalls angegeben werden. Das zweite Argument gibt die Transaktion an, in der die Event-Handler abgearbeitet

```
andpar
 role1 task1();
 raiseEvent(e1,
   current_tx, process)
   "task1 fertig";
 role2 task2();
|
 role3 task3();
 sync(e1,
  com.groiss.event.EventHandler,
  process) "warte bis task1 fertig";
 role4 task4();
end;
```

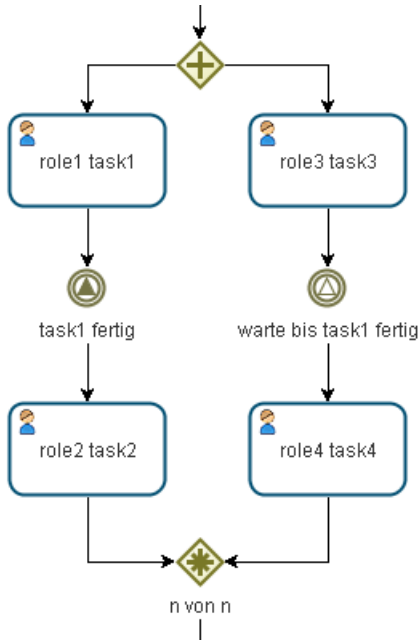


Abbildung 2.18: Synchronisation

werden, `current_tx` ist die aktuelle Transaktion. Das dritte Argument gibt einen Kontext an, das ist entweder das Schlüsselwort `process` für die aktuelle Prozessinstanz oder ein Datenelement (Formular oder Formularfeld).

Beim Sync wird neben der Event-Id ein Event-Handler und ein Kontext angegeben. Der Event-Handler ermöglicht es dem API Programmierer, auf das Empfangen des Ereignisses beliebig zu reagieren. Grafisch wird raiseEvent als BPMN Ereignisknoten mit schwarz gefülltem Dreieck (steht für Sende ein Ereignis) dargestellt, das Sync als Ereignisknoten mit ungefülltem Dreieck (Empfange ein Ereignis).

Damit ist die Definition der Kontrollstrukturen abgeschlossen. Im folgenden werden einige Kombinationen der obigen Elemente vorgestellt.

Parallelität in Schleifen

Wenn die Zweige der Parallelität nicht abgebrochen werden, ergibt sich folgendes Problem: Gegeben sei die in Abb. 2.19 dargestellte Situation mit einer Parallelität in einer Repeat-Schleife. Der Prozess wird nach der Parallelität

fortgesetzt, wenn ein Zweig abgeschlossen ist. Es wird dann die Parallelität neuerlich aufgerufen und einer der beiden Zweige ist unter Umständen doppelt instantiiert. Es können also in den parallelen Zweigen die Aktivitäten zu unterschiedlichen Schleifendurchläufen gehören. Dies kann zu Verwirrung der Anwender führen, deshalb wird folgende Vorgangsweise gewählt:

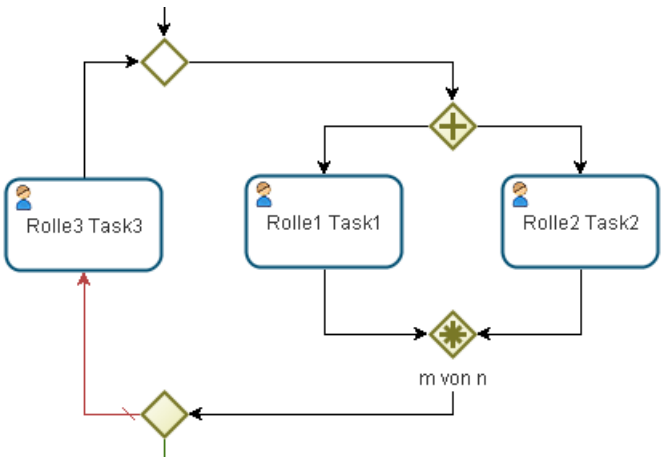


Abbildung 2.19: Parallelität in einer Schleife

Zur Synchronisation der Schleifendurchläufe wird der Eventmechanismus verwendet: Wenn alle Zweige einer Parallelität abgeschlossen sind, wird beim Andpar und Orpar von der Workflow-Engine automatisch ein Ereignis erzeugt. Auf dieses Ereignis kann gewartet werden, bevor wieder in die Schleife eingetreten wird. Abb. 2.20 zeigt die Implementierung. Der Sync Knoten wartet auf den Abschluss aller parallelen Zweige der rechten Parallelität (Task1 und Task2). Der Sync Knoten muss zu Task3 in einer Parallelität stehen, damit kein Ereignis verloren geht, während Task3 bearbeitet wird.

Diese Situation gibt es auch beim parfor-Konstrukt, wo auf ähnliche Weise damit umgegangen werden kann.

2.2.9 Workflow Patterns

Ausgehend von den Basiskonstrukten kommen in der Geschäftsprozess-Modellierung immer wieder ähnliche Teilprozesse (Muster, Patterns) vor. Im Artikel „Workflow Patterns“ [**?**] hat Van der Aalst 20 solche Muster vorgestellt,

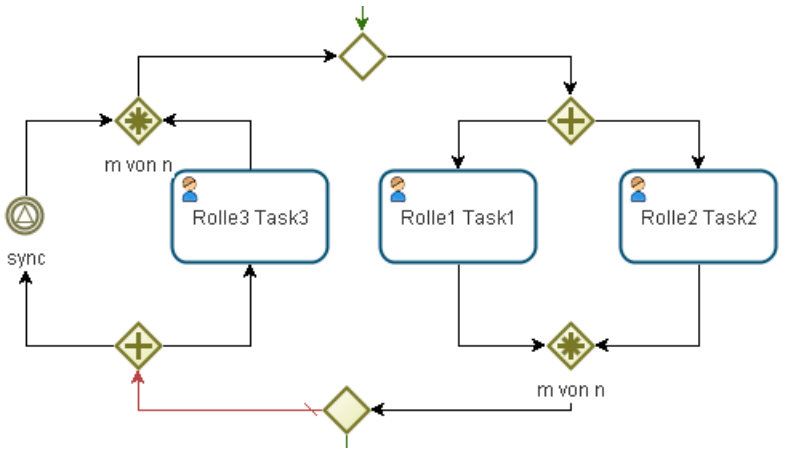


Abbildung 2.20: Parallelität in Schleife mit Synchronisation

später wurden in einer weiteren Arbeit der Gruppe um Van der Aalst die Patterns auf 43 Stück ergänzt [**?**], im Wesentlichen handelt es sich dabei um Verallgemeinerungen der ursprünglichen Patterns. Diese Patterns eignen sich gut um eine Sprache für Prozessdefinitionen zu evaluieren und mit anderen Sprachen zu vergleichen.

Die Patterns werden dabei mit Petri-Netzen dargestellt. Dieser Formalismus unterscheidet sich von der Prozessdefinition, wie sie hier vorgestellt wurde, vor allem dadurch, dass es keine expliziten Kontrollstrukturen, wie Ifs, Andpars oder änliches gibt, sondern nur die elementaren Bausteine davon: also die einzelnen Knoten und bedingten Kanten im Prozessgraph. Viele der Patterns sind daher nicht eins zu eins abbildbar, oft entsprechen zwei Patterns einer Kontrollstruktur, z.B. wird das If Konstrukt durch Pattern 4 - Exclusive Choice, das ist der If Knoten, und Pattern 5 - Simple Merge, dem Endknoten nach dem If, abgebildet.

Wir haben in einer früheren Arbeit gezeigt, wie diese Patterns mit dem hier vorgestellten Formalismus realisierbar sind [**?**] und verzichten hier auf die vollständige Darstellung aller Patterns. Wir sehen uns nur einige davon an und zeigen, wie sie realisierbar sind.

Persistent Trigger

Zwei Prozesszweige sollen mit Ereignissen synchronisiert werden. Das Ereignis soll gespeichert werden, bis der Prozess an der Stelle des Syncs ist. Das Pattern ist ohne Unterstützung persistenter Trigger einfach modellierbar, wie wir im vorigen Beispiel (Abb. 2.20) gesehen haben. Das Speichern des Ereignisses wird modelliert, indem das Sync an die Stelle des Prozesses gestellt wird, ab der ein Ereignis empfangen werden kann. Der eigentliche Prozess läuft parallel dazu (Andpar), die Parallelität vereinigt sich, wo das Sync bei persistenten Ereignissen stehen würde.

Multi-Choice

Eine Parallelität mit n Zweigen, wovon nur m instantiiert werden. Dies ist durch eine Kombination von Andpar und Ifs modellierbar. In Abb. 2.21 ist ein Beispiel mit zwei Zweigen dargestellt.

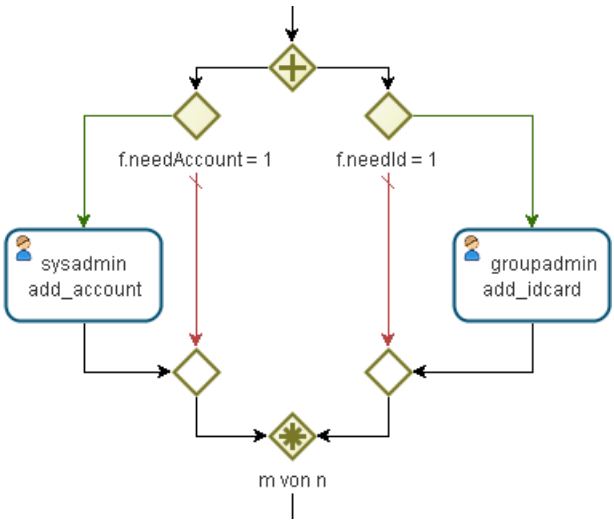


Abbildung 2.21: Multi-Choice

Multiple Instances without a Priori Run-Time Knowledge

Bei einem Parfor sollen zur Laufzeit zusätzliche Zweige hinzugefügt werden können. Prinzipiell ist dies wie in Abb. 2.22 dargestellt abbildbar. Im linken Zweig der Parallelität können durch Einfügen von Subformularen und Ausführen der Funktion *addParforSteps* weitere Zweige hinzugefügt werden. In

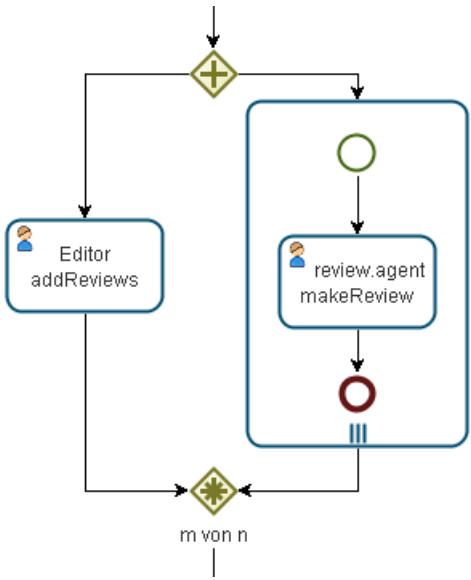


Abbildung 2.22: Parfor mit Hinzufügen von Zweigen zur Laufzeit

der Modellierung ist dies nicht sichtbar - erst bei der Modellierung der Daten, Funktionen und der Zugriffsrechte sieht man, dass im parallelen Schritt ein Einfügen von Subformularen möglich ist.

Ein Problem bei dieser Lösung ist, dass Parfor Zweige nur hinzugefügt werden können, solange mindestens ein Zweig noch aktiv ist. Dies könnte man durch eine Synchronisation sicherstellen, siehe Abb. 2.23 für die BPMN und Abb. 2.24 für das zugehörige WDL.

Wenn der linke parallele Zweig noch aktiv ist, wartet das Parfor mit dem Beenden. Wenn der linke Zweig abgeschlossen wird, wird das Ereignis ausgelöst, welches alle darauf wartenden Sync Schritte beendet. Das Sync muss in einem If stehen, um das Warten zu verhindern, wenn der linke Zweig bereits beendet ist, d.h. kein Ereignis mehr ausgelöst wird (Ereignisse werden nicht gespeichert).

Rekursion

Mit Subprozessaufrufen können auch rekursive Prozesse definiert werden. Beispiel: Für einen Kreditantrag wird die Genehmigung in einem eigenen Subprozess definiert, der aus den Schritten „Genehmigen“, „Gutachten erstellen“

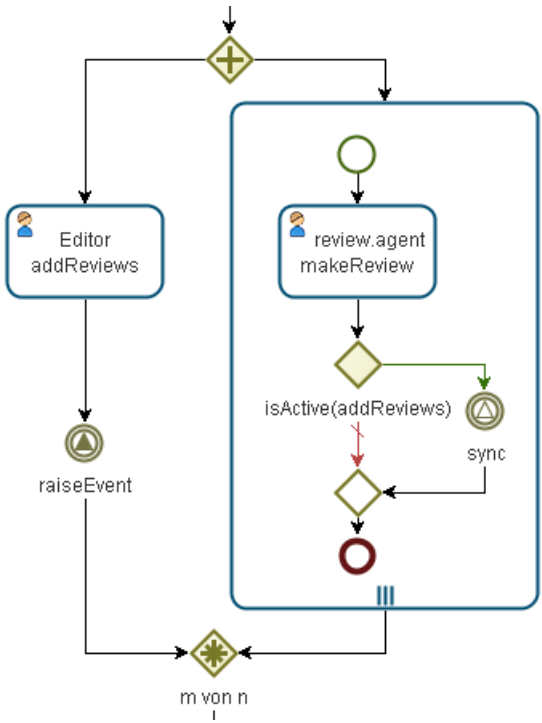


Abbildung 2.23: Parfor mit Hinzufügen von Zweigen zur Laufzeit und Synchronisation

und „an Vorgesetzten weitergeben“ besteht. Der letzte Schritt ist optional und ein Subprozessaufruf, da der Vorgesetzte wiederum die gleichen Möglichkeiten vorfinden soll - solange es einen Vorgesetzten gibt.

2.2.10 Mächtigkeit und Vollständigkeit

Zum Abschluss der Prozessmodellierung möchten wir nochmals auf den gewählten Ansatz zu sprechen kommen. Da nur ein Teil der Sprachkonstrukte von BPMN verwendet wurde, stellt sich die Frage, ob sich alle möglichen Prozesse damit modellieren lassen. Die Antwort ist Ja, jedes mögliche Programm und somit jeder mögliche Prozess lässt sich mit den vorgestellten Kontrollstrukturen ausdrücken (Turing vollständig). Die Frage ist nur, wie umständlich die Formulierung ausschließlich auf Basis der strukturierten Konstrukte ist. Bei den Programmiersprachen wurde diese Diskussion bereits in den 60er

```
andpar
 Editor addReviews(f);
 raiseEvent(finishParfor, current_tx, f);
|
 parallel for x in f.1 do
  review.agent makeReview(x);
  if com.groiss.wf.SystemAction.isActive("addReviews") then
   sync(finishParfor, com.groiss.event.EventHandler, f);
  end;
 end;
end;
```

Abbildung 2.24: Parfor mit Hinzufügen von Zweigen zur Laufzeit und Synchronisation - WDL

Jahren geführt, siehe z.B. Edsger W. Dijkstra, „Go To Statement Considered Harmful“, [**?**], und es ist dort mittlerweile Konsens, dass die Beschränkung auf strukturierte Kontrollstrukturen die Nachteile deutlich überwiegt.

Bei der Geschäftsprozessmodellierung sind wir noch nicht so weit. Freund und Rücker bringen in [**?**] ein Beispiel eines Prozesses, der sich nicht einfach strukturiert umformulieren lässt, siehe Abb. 2.25. Eine Umformulierung kann in die in obiger Abb. 2.18 dargestellte Struktur erfolgen - und ist gar nicht so kompliziert.

Die bereits erwähnte BPEL verwendet ebenfalls nur strukturierte Konstrukte, ist allerdings eher als Ausführungssprache gedacht und nicht als Werkzeug für den Prozessanalytiker. Für das „schnelle“ Zeichnen eines Prozesses, vor allem in der Analysephase, ist eine unstrukturierte Darstellung oft einfacher und leichter verständlich. Die endgültige Formulierung eines Geschäftsprozesses sollte dann allerdings strukturiert erfolgen.

2.2.11 Welche Prozessdiagramme sind korrekt?

Wie entscheiden wir nun, was ein syntaktisch korrekter Prozessgraph ist? Dazu definieren wir die bereits genannte Graph-Grammatik [**?**]. Diese definiert einen Start-Graphen, sowie eine Reihe von Transformationsregeln. Alle Graphen, die aus dem Startgraphen unter fortgesetzter Anwendung dieser Regeln erstellt werden können, sind gültige Graphen im Sinne dieser Graph-Grammatik. Abb. 2.26 zeigt die Graph-Grammatik zur Prozessdefinition.

Für die Definition der Grammatik müssen wir Kanten- und Knotentypen festlegen. Kantentypen gibt es die beiden bereits bekannten: Die Standardkante

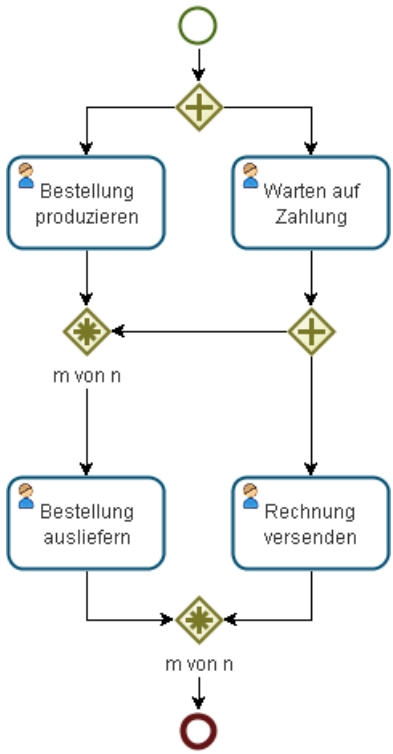


Abbildung 2.25: Ein unstrukturierter Prozess

und die Default-Kante nach Bedingungsauswertungen.

Die Knoten sind ebenfalls die bereits bekannten. Wir fassen für eine einfache Formulierung der Grammatik aber einerseits mehrere Knotentypen zusammen (zum *activity* Knoten), andererseits werden einige in BPMN gleich aussehende Knoten mit unterschiedlichen Typbezeichnungen versehen. Die Knoten im einzelnen: *activity* bezeichnet alle elementaren Aktivitäten (Task, Subprozess, Batch, Systemschritt). *If, while, exit_when* sind die Bedingungsknoten in den entsprechenden Kontrollstrukturen. *par* und *branch* der Beginn der Parallelitäten; *andjoin* und *orjoin* das Ende einer Parallelität; *end_if*, *end_choice*, *loop*: die Knoten zur Zusammenführung des Ablaufs aus zwei Zweigen; *begin*, *end*, *end_branch*: Prozessbeginn und -ende.

In der Graph-Grammatik wird zuerst der initiale Graph definiert, ein Begin-

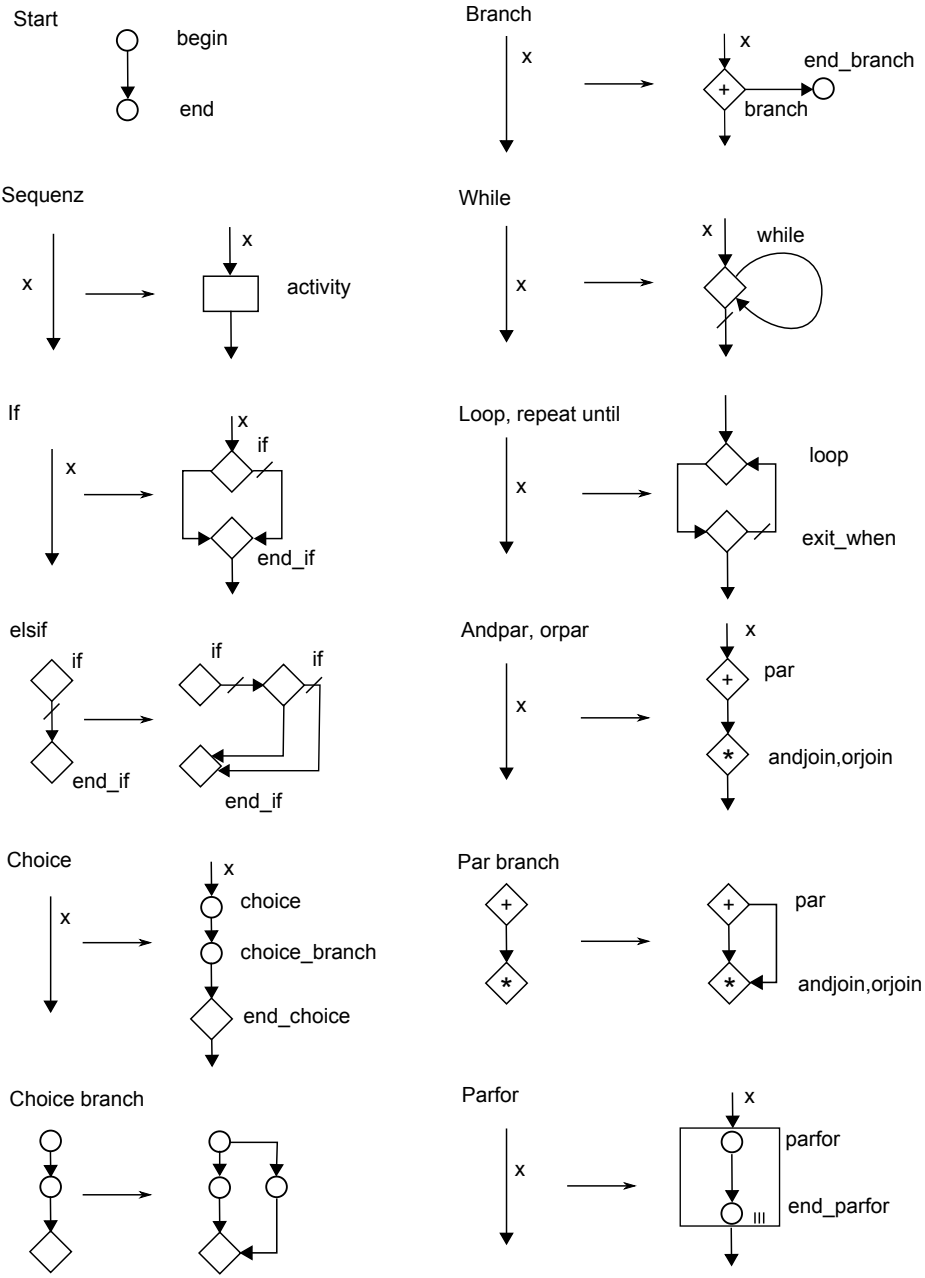


Abbildung 2.26: Graph-Grammatik

Knoten und ein End-Knoten verbunden mit einer Standardkante. Danach folgen die einzelnen Regeln. Die meisten ersetzen eine Kante durch eine Kontrollstruktur, in der eingesetzten Struktur wird der Kantentyp (Variable x) für jeweils eine Kante übernommen. Es können alle Kanten auf diese Weise ersetzt werden, außer der Kante zwischen dem *choice* und *choice_branch* Knoten. Die Regeln für *choice branch* und *par branch* fügen eine Kante zu den entsprechenden Kontrollstrukturen hinzu.

Normalerweise braucht man sich über die syntaktische Korrektheit eines Prozessdiagramms keine Gedanken zu machen, da es mit einem Werkzeug erstellt wird, welches die Einhaltung der Syntaxregeln gewährleistet, z.B. der @enterprise Prozesseditor.

2.2.12 Ausnahmebehandlungen

Bei der Ausführung eines Prozesses können verschiedene Ausnahmen auftreten, die wir bereits in der Modellierung berücksichtigen müssen.

Zeitliche Ausnahmen

werden durch Eskalationsaktionen behandelt. Es gibt verschiedene Situationen, in denen eine Ausnahmebehandlung nötig ist:

- ❑ die Prozesssollzeit wurde überschritten,
- ❑ die Sollzeit bei einem Task wurde überschritten,
- ❑ die Wartezeit auf ein Ereignis wurde überschritten,
- ❑ mit der Bearbeitung eines Tasks wurde nicht rechtzeitig begonnen.

Mit zeitlichen Ausnahmen können Eskalationsaktionen verknüpft werden, die jeweils mit einem zeitlichen Abstand zum (potentiellen) Ereignis gestartet werden, also z.B. proaktiv 4 Stunden vorher oder reaktiv 2 Tage nachher.

Als Aktion kann ein Task, ein Prozess oder ein Systemschritt gestartet oder eine Nachricht versandt werden.

Fehlerbehandlung

Tritt in einem automatischen Schritt (Batch-Schritt oder einem bisher nicht behandelten Web-Service Aufruf) ein Fehler auf, kann dieser mit einer definierten

Aktion behandelt werden. Ein sogenannter Eskalations-Zweig kann im Prozess definiert werden, der vom Schritt, in dem der Fehler auftritt, wegführt und entsprechende Aktionen enthält.

Alle anderen Fehler treten bei Aktionen auf, die durch eine Benutzeraktion getriggert werden. Dort behandelt die Laufzeitumgebung den Fehler entsprechend: Rückgängigmachen (rollback) der Änderungen der Aktion, Anzeige einer Fehlermeldung für den Benutzer.

2.2.13 Was wird nicht modelliert?

Bei der Modellierung eines Prozesses ist es sinnvoll, zuerst den Standardfall zu betrachten, um eine grobe Struktur des Prozesses zu modellieren. Die Frage ist nun, inwieweit Sonderfälle, seltene Abweichungen, ad-hoc Änderungen modelliert werden sollen.

Nehmen wir als Beispiel einen einfachen Problembehebungsprozess (Abb. 2.27). Nach dem Schritt Lösen folgt der Schritt Test. Es ist sicherlich häufig so, dass der Test nicht erfolgreich ist und der Prozess an den Entwickler zurückgeht. Dies kann explizit mit einer Schleife modelliert werden. Meist ist es jedoch besser, die Schleife nicht zu modellieren und festzulegen, dass bei der Prozessausführung Zurückgehen zu einem früheren Schritt möglich ist. In Kapitel 3 wird auf die verschiedenen Abweichungen vom Prozessweg noch näher eingegangen.

2.2.14 Definition mit Regeln

Es gibt eine Reihe von Prozessen, die sich mit den zuvor beschriebenen Konstrukten nur schwer definieren lassen. Meist sind dies nicht komplett strukturierte Abläufe. In einem einfachen Fall hat man drei Tasks, die alle gemacht werden müssen, die Reihenfolge ist aber beliebig oder erst zur Ausführungszeit vom Bearbeiter zu bestimmen. Derartige „semistrukturierte“ Prozesse lassen sich mit den oben genannten Methoden nicht modellieren.

Ein Beispiel zeigt die Praxisrelevanz solcher Prozesse:

Elektronischer Akt - Hier gibt es eine Reihe von unterschiedlichen, mehr oder weniger genau definierten Aktivitäten, z.B. Genehmigen, Bearbeiten, aber keine klar definierte Reihenfolge zwischen den Aktivitäten. Es gibt jedoch ein Regelwerk, wann eine Aktivität durchgeführt werden darf und wann nicht. Zum Beispiel sind nach einer Genehmigung bestimmte Aktivitäten zulässig, andere nicht.

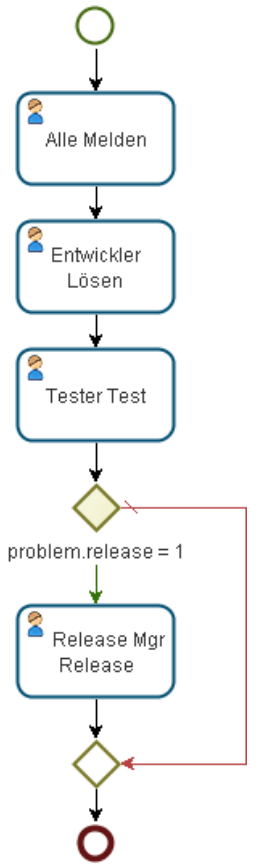


Abbildung 2.27: Problembehebungsprozess

Prozesse wie der Obige lassen sich am besten durch ein Regelwerk beschreiben. Die Prozessdefinition besteht dabei aus einer Menge von Regeln. Jede Regel hat einen Bedingungsteil und einen Aktionsteil:

Die Bedingung beschreibt einen Prozesszustand, z.B.: was ist der aktuelle Schritt, welche Schritte wurden bereits durchgeführt, wie sehen die Prozessdaten aus etc.

Der Aktionsteil enthält einen Vorschlag über die möglichen Nachfolgeschritte, seine Akteure und Datenänderungen.

Die Ausführung eines solchen Prozesses funktioniert folgendermaßen: Beim

Weiterleiten wird der aktuelle Prozesszustand in einen Regelinterpreter geladen, die Regeln werden ausgewertet und die Kandidaten für den Nachfolger ermittelt. Existiert kein Nachfolger, ist der Prozess beendet.

Abschließend sei zur Modellierung noch erwähnt, dass es Prozesse gibt, die sich mit allen diesen Mitteln nicht modellieren lassen - wenn nämlich weder der genaue Ablauf noch Regeln zur Modellierungszeit bekannt sind. In diesem Fall wird der Prozess „ad-hoc“, zu Beginn der Ausführung oder aber während der Laufzeit, modelliert. Wir werden dies in Abschnitt 3.5 näher behandeln.

2.3 Definition der Akteure

Bisher wurde die Abfolge der einzelnen Aktivitäten betrachtet. Für jede interaktive Aktivität muss auch festgelegt werden, wer die Aktivität ausführt. Dazu stehen die folgenden Möglichkeiten zur Verfügung:

Benutzer

Die Aktivität soll von einem bestimmten Benutzer ausgeführt werden und dies wird in die Prozessdefinition eingetragen. Das macht kaum jemals Sinn, auch wenn es nur eine Person gibt, die diesen Schritt ausführen wird, so kann sich dies im Laufe der Zeit ändern. In Prozessdefinitionen sollten demgemäß Personen nie direkt als Akteur angegeben werden [2].

Rolle

Die Angabe einer Rolle ist die gängige und wichtigste Form der Akteursdefinition.

Rolle in Organisationseinheit

Die kombinierte Angabe von Rolle und Organisationseinheit schränkt den Kreis der Akteure auf diejenigen ein, welche die betreffende Rolle in der angegebenen Organisationseinheit haben, z.B. Leiter der Organisationseinheit Marketing.

[2] Sinn machen kann eine Angabe aber zur Laufzeit für eine bestimmte Prozessinstanz

Akteur eines früheren Schritts

Referenz auf den Akteur eines bereits abgeschlossenen Workflowschritts. Dies ist z.B. nötig, wenn in einem späteren Schritt eine Information an den Antragsteller aus der aktuellen Prozessinstanz gehen soll.

Akteur aus Prozessdaten

Eine ad-hoc Festlegung des Akteurs ist möglich, wenn die Akteursangabe aus den Prozessdaten (einem Formularfeld) gelesen wird. In einem früheren Schritt wird der Akteur eines Folgeschrittes in ein Formularfeld geschrieben, der Akteur ist dann wieder entweder ein Benutzer oder eine Rolle (optional in einer Organisationseinheit).

Akteur wird von Programm bestimmt

Eine weitere Möglichkeit ist, den Akteur zur Laufzeit durch einen Funktionsaufruf zu ermitteln. Beispiele, wo dies vorkommen kann, sind:

- Rotationsprinzip: Die Aufgaben werden nacheinander an die Mitglieder einer Rolle vergeben.
- Vergabe nach Last: Bei der Zuordnung wird der Inhalt des Arbeitskorbs analysiert. Die Aktivität geht an den Benutzer mit dem kürzesten Arbeitskorb.
- Vergabe nach einer Regel aus den Daten: etwa Aufteilung nach Anfangsbuchstaben des Nachnamens oder nach Wohnort.

Es ist also auch eine Kombination vorstellbar: In der Prozessmodellierung wird eine Rolle angegeben, bei der Erzeugung der Aktivitätsinstanz wird durch ein Programm ein bestimmter Akteur mit dieser Rolle ausgewählt.

Leere Akteursangabe

Wird bei einem interaktiven Schritt kein Akteur angegeben, muss zur Laufzeit ein Akteur bestimmt werden. Dies geschieht durch den Akteur des Vorgängerschritts im Zuge des Weiterleitens.

Das Beispiel in Abb. 2.28 soll einige dieser Möglichkeiten veranschaulichen.

Jeder Benutzer soll einen Auftragsprozess starten können, die Aufträge sollen von beliebigen, anderen Personen ausgeführt werden. Der Auftraggeber

```
begin
 <order>
 Alle order(form);
 loop
   form.agent dojob(form);
   exit when form.finished = 1;
 end;
 order:user inform(form);
end
```

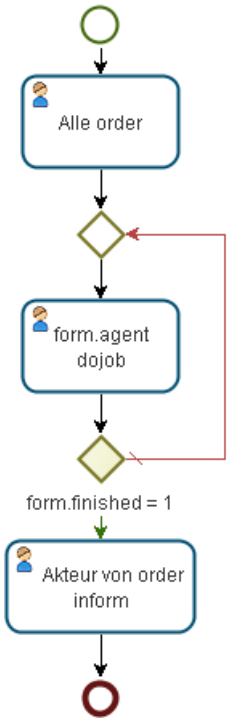


Abbildung 2.28: Aufgabenbearbeitung

startet den Prozess und füllt ein Antragsformular aus. In eines der Felder wird der nächste Akteur eingetragen. Nach dem Weiterleiten erhält dieser den Prozess. Wenn er den Prozess nicht selbst bearbeiten will, kann er das Akteurfeld wieder ändern und den Prozess an den nächsten Akteur weiterschicken. Führt er die gewünschte Tätigkeit durch, dann vermerkt er dies im Formular. Der Prozess gelangt dann beim Weiterleiten zurück zum Auftraggeber (Prozessstart-Akteur), der über die Durchführung informiert wird.

Die verwendeten Akteursdefinitionen sind also Rolle im ersten Schritt, Akteur aus Formular im zweiten Schritt und Akteur eines vorangegangenen Tasks im dritten Schritt.

Ein weiteres Beispiel, dass einige der unterschiedlichen Akteursdefinitionen zeigt, findet sich in Abschnitt 2.8.

2.4 Modellierung der Daten

Eine wesentliche Eigenschaft von BPM-Systemen ist, dass bei der Bearbeitung von Prozessen Daten erzeugt und verändert werden. Daten, die direkt zu einem Prozess gehören, nennt man *Prozessdaten*. Daten, die prozessübergreifend sind, aber ebenfalls im System verwendet werden, heißen *prozessrelevante Daten*. Die Prozessdaten werden für gewöhnlich im BPMS selbst verwaltet. Die prozessrelevanten Daten können auch in externen Systemen verwaltet werden und, wenn nötig, gelesen und geschrieben werden (Die Kommunikation mit anderen Systemen wird in Abschnitt 3.8 behandelt).

Um zu einem im BPMS lauffähigen Prozess, der Daten manipuliert, zu gelangen, müssen wir diese modellieren. Dabei sind zwei Aspekte zu unterscheiden: das logische Schema der Daten und die Präsentation der Daten. Auf beides wird in den folgenden Abschnitten eingegangen. Für das Schema der Daten verwenden wir den Konstrukt des Formulars, für die Präsentation den Standard XForms.

2.4.1 Modellierung der Daten mit Formularen

Betrachten wir zuerst, in welcher Form die Daten vorliegen: Grob kann man zwischen strukturierten und unstrukturierten Daten unterscheiden.

Strukturierte Daten

Die Definition der Struktur erfolgt üblicherweise in einem konzeptuellen Schema mit einem Entity-Relationship Diagramm oder einem ähnlichen Formalismus. Hat man ein solches Datenschema vorliegen (die Definition liegt außerhalb des Rahmens dieses Buches, siehe z.B. [**?**]), nimmt man zuerst die Trennung zwischen den Prozessdaten und prozessrelevanten Daten vor.

Das folgende einfache Beispiel soll das illustrieren. In Abb. 2.29 ist das Schema der Daten zu einem Bestellprozess abgebildet. Die Prozessinstanz ist ebenfalls als Entity eingezeichnet. Die Bestellung selbst und die einzelnen Bestellpositionen sind die Prozessdaten. Pro Prozessinstanz gibt es genau eine Bestellung (1:1 Relation). Die Bestellungen haben einen Kunden und die Bestellpositionen ein Produkt, beides prozessrelevante Daten.

Unstrukturierte Daten

Unstrukturierte Daten sind Daten, deren Bestandteile im BPMS nicht weiter zerlegt werden, z.B. Bilder oder Textdokumente. Jedes dieser Datenobjekte hat

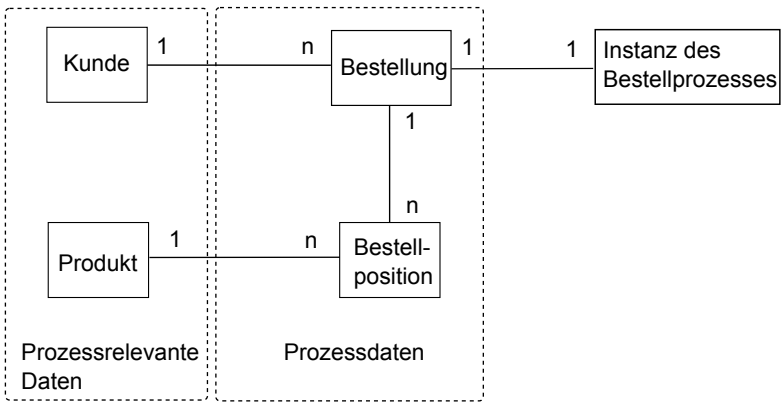


Abbildung 2.29: Datenschema für einen Bestellprozess

neben dem Inhalt auch eine Reihe Metadaten, die im System verwaltet werden müssen.

Die Art der Metadaten ist abhängig vom Typ des Inhalts. Handelt es sich z.B. um ein eingescanntes Schriftstück, ist die Angabe des Absenders, des Empfängers, des Eingangsdatums etc. sinnvoll, bei einem Foto etwa die Aufnahmezeit und der Aufnahmeort.

Formulare und Dokumente

Zur Speicherung von strukturierten Daten verwenden wir sogenannte *Formulare*, die Definition erfolgt in *Formulartypen*. Ein Formulartyp dient zur Speicherung und Darstellung von Daten eines bestimmten Typs. Zur Definition des Formulartyps gehören: Die Definition der einzelnen Felder mit Id, Namen und Typen; die Definition der Darstellung des Formulars zur Erfassung und Bearbeitung von Daten. Die einzelnen Felder sind skalar, d.h. keine zusammengesetzten Typen.

Zur Repräsentation tabellarischer Formularbereiche verwenden wir sogenannte *Subformulare*. Dabei wird bei einem Formulartyp – dem Hauptformular – definiert, dass ein zweiter mit ihm in einer abhängigen 1:n Beziehung steht. In der Bearbeitungsmaske des Hauptformulars ist es dann möglich, Subformulare anzulegen, zu ändern und zu löschen. In Abb. 2.30 ist ein solches Formular mit Subformular dargestellt.

Für unstrukturierte Daten verwenden wir sogenannte *Dokumente*. Da Doku-

Abbildung 2.30: Beispiel der Darstellung eines Formulars

mente Metadaten enthalten können, repräsentieren wir sie als Formulare, mit einem zusätzlichen „content“ Feld zur Aufnahme des unstrukturierten Inhalts, analog zu dem Formulartypen sprechen wir bei der Definition von *Dokumenttypen*.

Eine weitere Spezialform von unstrukturierten Daten sind *Notizen*. Notizen können zu Prozessinstanzen oder zu Dokumenten hinzugefügt werden.

Dokumente können in *Ordnern* abgelegt werden, diese haben ebenfalls Formularcharakter (Metadaten, Bearbeitungsmaske), die Definitionen nennen wir *Ordnertypen*.

Datentypen

Für die einzelnen Felder der Formularattribute müssen die Typen festgelegt werden, dabei kommen die Standard-Typen, wie Boolean, String, Integer, Float, Datum etc. zum Einsatz. Weiters kann der Typ eines Feldes auch ein Formular sein oder ein Stammdatenobjekt des BPMS, z.B. Benutzer, Rolle oder Organisationseinheit.

Eine weitere Art von Typen sind Aufzählungstypen, d.h. Typen deren Werte in einer vorgegebenen Menge enthalten sind (z.B. Sprachen, Länder, Branchen). Die Werte solcher Aufzähltypen können sich in der Praxis durchaus oft verändern. Deshalb gibt es die Möglichkeit, die Werte aus einem eigenen *Wertelisten-Formular* zu lesen, in dem die Mengenelemente flexibel definiert werden können.

Jede Werteliste hat eine eindeutige Id und eine Menge von Werten, wobei für jeden Wert ein unveränderlicher gespeicherter Wert sowie ein veränderbarer Anzeigename erfasst werden kann.

Verbindung von Daten und Prozessen

Die Verbindung zwischen den Prozessen und den Formularen erfolgt bei der Definition der Prozesse. In WDL wird im Header des Prozesses eine entsprechende Zeile eingefügt, die Syntax sei an einem Beispiel erläutert:

```
forms order orderform, del deliveryform;
```

Nach den Schlüsselwort `forms` folgt eine Menge durch Beistrich getrennter Paare von Formularname und Formulartyp. Jeder Formularname kann dann im Prozessablauf (z.B. in Bedingungen) wie ein Variablenname verwendet werden, auf die Formularfelder kann wie üblich mittels einer Punkt-Notation zugegriffen werden (z.B. `order.customer`). Beim Prozessstart wird für jede der Formularvariablen eine Instanz angelegt. Die Systemumgebung stellt die Benutzerschnittstelle für die Bearbeitung zur Verfügung.

Dem Prozess zugeordnete Dokumente werden in einem Ordner abgelegt, der zur Prozessinstanz gehört (1:1 Beziehung).

Für prozessrelevante Daten müssen in der Modellierung die „Einstiegspunkte" festgelegt werden, das sind entweder bestimmte Ordner im prozessunabhängigen Dokumentenmanagementsystem, in denen die Formulare und Dokumente enthalten sind oder Links zu Tabellen von Formularen.

Der nächste Schritt ist die Abbildung eines konzeptuellen Schemas auf Formulare: Die einzelnen Entities werden als Formulartypen definiert. Die Abbildung von Relationen zwischen den Entities kann folgendermaßen durchgeführt werden:

- 1:n Relationen: Es gibt mehrere Möglichkeiten, je nach Existenzabhängigkeit der Objekte:
  - Subformulare: sind für Beziehungen zu n abhängigen Objekten gedacht, z.B. eine Bestellung enthält n Bestellpositionen oder eine Firma hat n Adressen. Die Definition der Verknüpfung erfolgt bei der Erstellung des Formulartyps.
  - Speichern des Fremdschlüssels: Beziehungen zu unabhängigen Objekten: z.B. Ein Buch hat genau einen Autor, der Schlüssel des Autorenobjekts wird also beim Buch als Fremdschlüssel gespeichert.
  - Ordner: Diese Art eignet sich vor allem bei heterogenen abhängigen Entitites bzw. unstrukturierten Daten (Dokumenten) sowie für den Aufbau von Hierarchien. Ein Entity wird als Ordner angelegt, andere Entities werden in diesem Ordner eingefügt.

- n:m Relationen: Für jede solche Relation muss ein eigener Relations-Formulartyp angelegt werden. Dieser kann wiederum als Subformular einer der beteiligten Datenobjekte definiert werden. Die andere Klasse wird über Fremdschlüssel referenziert. Die Entscheidung, welches der beiden Formulare das Hauptformular wird, hängt von der bevorzugten Bearbeitung ab. Beispiel: Die Zuordnung von Rollen zu Benutzern wird in der Regel vom Benutzer aus gewartet, d.h. man würde ein Subformular UserRole zum Benutzerformular hinzufügen. Im UserRole Formular gibt es dann ein (Auswahl-)Feld für die Rolle - und weitere optionale Felder.

Mehrwertige Relationen (n:m:k) werden wie n:m Relationen behandelt, d.h. es wird ein Relations-Formulartyp angelegt, die anderen Klassen (Formulartypen) werden über Fremdschlüssel referenziert.

Für das obige Beispiel eines Bestellprozesses (Abb. 2.29) bedeutet das folgendes: Kunde, Produkt, Bestellung und Bestellposition werden als Formulartypen definiert. Bestellposition ist ein Subformular von Bestellung, der Kunde wird im Bestellformular referenziert, das Produkt in der Bestellposition.

Im Dokumentenmanagementsystem gibt es einen Ordner für Kunden (in dem die Kunden-Formulare liegen) und einen Ordner für Produkte (in dem die Produkt-Formulare liegen).

Mit der eben dargestellten Vorgangsweise ist es möglich, jedes beliebige Datenschema mit unseren Grundkonstrukten zu realisieren und die Verbindung zwischen Daten und Prozessen herzustellen.

2.4.2 Präsentation der Daten - XForms

Neben dem Schema der Daten ist auch deren Darstellung Bestandteil der Datenmodellierung. In diesem Abschnitt behandeln wir die Präsentation der Formulardaten mithilfe des XForms Standards des World Wide Web Consortiums (W3C) [**?**]. Der Standard beschreibt eine Reihe von Formularelementen und die zugehörige Logik.

Die Formularelemente können in eine Trägersprache, die XML basiert sein muss, eingebettet werden. Die natürliche Trägersprache ist XHTML (die XML Version von HTML). Die Benennung der Formularelemente ist an die entsprechenden HTML Elemente angelehnt: Im folgenden ein Beispiel für ein Auswahlfeld (aus [**?**]):

```
<select1 ref="method">
  <label>Select Payment Method:</label>
  <item>
    <label>Cash</label>
    <value>cash</value>
  </item>
  <item>
    <label>Credit</label>
    <value>cc</value>
  </item>
</select1>
```

Das Element dient zur Auswahl einer von mehreren angebotenen Möglichkeiten. Zu beachten ist dabei:

- Es wird nicht spezifiziert, ob die Auswahl mit Radio-Buttons oder einer Dropdownlist erfolgt. Die konkrete Darstellung kann über das optionale Attribut `appearance` festgelegt werden.
- Labels sind Teil der Formularelemente, dies vereinfacht die Behandlung von Sichtbarkeiten, Eingabehilfen etc.
- Es gibt kein `value` Attribute. Das Element verweist mit `ref` auf ein Element in einer XML Struktur, welche die Instanzdaten enthält. Der Verweis wird als XPath Ausdruck angegeben. XPath [**?**] ist eine Sprache zur Navigation in XML Dokumenten.

Die Struktur des Formulars und der Inhalt (Instanzdaten) sind also voneinander getrennt. Die Instanzdaten sind in einem XML Dokument repräsentiert, dessen Struktur frei definierbar ist. Eingebettet werden die Daten im Formular in einem `model` Element, das neben den Instanzdaten weitere Informationen enthält. Das folgende XML zeigt solch ein `model` Element (aus [**?**]):

```
<model>
  <instance>
    <payment>
      <method>cc</method>
      <number>1234539299549</number>
      <expiry></expiry>
    </payment>
  </instance>
  <submission action="http://example.com/submit" method="post"/>
</model>
```

Die Daten sind im `instance` Element enthalten. Mit dem `submission` Element wird angegeben, wohin die Daten des Formulars nach dem Ausfüllen geschickt werden sollen. Ein gesamtes XForm besteht also aus einem Dokument in einer Trägersprache, z.B. XHTML, in dem ein `model` Element im Header enthalten ist und die einzelnen Formularelemente an beliebiger Stelle im XHTML Dokument aufscheinen.

Teil der Präsentationsschicht der Daten ist die mit der Eingabe und Veränderung verbundene Logik. Im Wesentlichen umfasst diese die folgenden Aspekte:

- Eingabeüberprüfung und -unterstützung: Sicherstellung, dass nur gültige Werte eingegeben werden können. Einfache Eingabe von zusammengesetzten Werten, z.B. Datum und Uhrzeit. Auswahlmöglichkeit von bestehenden Datenobjekten (z.B. Kunden, Produkten).

- Dynamische Darstellung: Ein- und Ausblenden von Teilen des Formulars je nach Kontext (abhängig z.B. vom aktuellen Schritt im Prozess, von Werten anderer Felder oder Rechten des Benutzers).

- Berechnung von abhängigen Werten, wie z.B. Summen.

In XForms kann diese Logik mit `bind` Elementen und XPath Ausdrücken abgebildet werden. Ein `bind` Element kann die folgenden Attribute haben, bis auf `type` jeweils XPath Ausdrücke:

- `type`: Angabe des Datentyps. Damit kann die Eingabe auf gültige Werte eingeschränkt werden und Eingabehilfen, z.B. eine Datumsauswahl bei Datumsfeldern, können automatisch in das Formular hinzugefügt werden.

- `constraint`: Weitere Einschränkungen des Wertebereichs können als constraint definiert werden. Damit können auch Abhängigkeiten zwischen Feldern definiert werden.

- `required`: Ein Ausdruck, der angibt, ob das Feld ein Mussfeld ist.

- `readonly`: Legt fest, ob das Feld editierbar ist.

- `relevant`: Legt fest, ob das Feld dargestellt wird.

- `calculate`: Damit kann der Wert eines Feldes berechnet werden.

Die Besonderheit von XForms ist, dass diese Definitionen deklarativ erfolgen, ohne die sonst für diese Zwecke übliche Verwendung von Script-Sprachen, z.B. Javascript bei Web-Applikationen.

Die Formularelemente zur Bearbeitung der Daten sind die in Benutzerschnittstellen-Frameworks üblichen:

- ❑ Eingabefelder: einzeilig, mehrzeilig, für Passwörter
- ❑ Auswahlfelder für einfache Auswahl: Radio-Buttons, Select-Listen
- ❑ Auswahlfelder für mehrfache Auswahl: Select-Listen, Checkboxen
- ❑ Schieberegler (Slider) zur Auswahl numerischer Werte

Die Laufzeitkomponente zur Darstellung von XForms kennt die Typen der Felder und kann somit die Eingabefelder entsprechend anpassen. So wird z.B. ein Eingabefeld für einen Boole'schen Wert als Checkbox dargestellt, bei einem Datum wird beim Eingabefeld ein Kalender als Eingabehilfe hinzugefügt.

Eine weitere Bearbeitungsmöglichkeit bietet der `repeat` Konstrukt. Damit können im Formular eingebettete Tabellen bearbeitet (eingefügt, verändert und gelöscht) werden (vergleiche Subformulare in Abschnitt 2.4.1).

Die obige Abb. 2.30 zeigt ein Formular mit verschiedenen solcher Elemente. Die Felder MitarbeiterIn und Projekt haben die Datentypen Benutzer bzw. Projekt, deshalb wird jeweils eine Auswahlliste (mit Suchfunktion) dargestellt. Der Zeitraum wird mit einer Auswahlliste für den Monat (mit fixem Wertebereich) und mit einem Eingabefeld für das Jahr eingegeben. Die einzelnen Tabellenzeilen befinden sich innerhalb eines `repeat` Elements, die Tabelle kann mit den Schaltflächen darunter bearbeitet werden.

XForms Laufzeitkomponente

Wie erfolgt nun die Bearbeitung einer Formularinstanz im BPMS: Der Ablauf bei der Anzeige eines XForms sieht folgendermaßen aus:

1. Das XForm Template für den Formulartyp wird von seinem Speicherort geladen und geparst.
2. Im `model` Element wird ein `instance` Element mit den Instanzdaten des Formulars und den Kontextdaten hinzugefügt. Die Felder der Formulare sind über XPath ansprechbar, z.B. über den Pfad `/data/form/`*feldname*.

3. Die festgelegten Sichtbarkeiten werden als `bind` Elemente ebenfalls ins `model` eingefügt.
4. Je nach Art der Anzeige werden entsprechende Submit-Schaltflächen und die dazugehörigen URLs zu den Aktionen eingefügt.
5. Aus dem XForm wird eine HTML Seite erzeugt: Dabei werden die einzelnen XForms Formularelemente in die HTML Äquivalente übersetzt, mit den Daten aus dem `model` befüllt und entsprechend der Sichtbarkeiten dargestellt.

Das folgende Beispiel zeigt das `model` einer Formularinstanz mit den Feldern `name`, `country` und `amount` im Kontext eines Prozesses zur Genehmigung von Dienstreisen:

```
<xf:model>
  <xf:instance>
    <data xmlns="">
      <form object="com.dec.avw.appl.wiztest_1:1000074412">
        <transactionId>4</transactionId>\
        <avwcreatedby>herbert groiss</avwcreatedby>
        <avwcreatedat>2009-04-06T07:05:22Z</avwcreatedat>
        <avwchangedby>herbert groiss</avwchangedby>
        <avwchangedat>2009-04-07T08:28:22Z</avwchangedat>
        <name>Max Huber</name>
        <destination>Wien</destination>
        <cost>40011</cost>
      </form>
      <context>
        <activityinstance oid="42420">158</activityinstance>
        <processinstance oid="42417">158</processinstance>
        <task oid="10185" id="request">Anfordern</task>
        <processdefinition oid="10090" id="hr_businesstrip">
         Dienstreise</processdefinition>
        <viewmode>view_text</viewmode>
      </context>
    </data>
  </xf:instance>
  <xf:bind nodeset="/data/form/name" required="false()"
    type="string" />
  <xf:bind nodeset="/data/form/country" required="false()"
    type="string" />
  <xf:bind nodeset="/data/form/amount" required="false()"
```

```
    type="decimal" />
  <xf:submission replace="instance" validate="false"
    action="com.groiss.storegui.FormWrapper.updateNoAction"
    id="submit0" method="post" />
  <xf:submission method="post" id="submit1"
    action="com.groiss.storegui.FormWrapper.updateAndAction?\
    javaAction=finish&afterSubmit=top.right.location=comingFrom"
    />
  <xf:submission afterSubmit=parent.parent.changeTab()"
    action="com.groiss.storegui.FormWrapper.updateAndAction?\
    method="post" id="submit2" validate="false"/>
</xf:model>\
```

Neben den Formularfeldern sind die folgenden Kontextdaten enthalten:

- `activityinstance`: die Objekt-Id (oid) und textuelle Repräsentation der aktuellen Aktivität
- `processinstance`: die oid der Prozessinstanz und die Id des Prozesses
- `task`: die oid, die Id und der Name des Tasks
- `processdefinition`: die oid, die Id und der Name der Prozessdefinition
- `viewmode`: Der Anzeigemodus gibt an, ob das Formular zum Ändern, zum Ansehen oder zum Drucken dargestellt wird.

Im folgenden sollen einige Beispiele die Verwendung von XForms schlaglichtartig an Fragmenten von verschiedenen Formularen illustrieren. Die konkreten Pfade, URLs, Zugriffe auf Wertelisten und Konfigurationsdaten entsprechen der @enterperise XForms Implementierung und können sich in anderen Systemen davon unterscheiden.

Beispiel: Summe von Subformularen. In einem Rechnungsformular gibt es ein Subformular mit den einzelnen Positionen. Am Hauptformular soll die Summe angezeigt werden. Dazu kann man ein `bind` Element verwenden, das mit dem `calculate` Attribut die Summe berechnet:

```
<xf:bind nodeset="/data/form/totalamount"
   calculate="sum(/data/form/subform/form/total)"/>
```

Beispiel: Ausblenden eines Feldes. Abhängig von den Daten eines anderen Formularfeldes sollen andere Felder dargestellt oder ausgeblendet werden:

Das Eingabefeld für Transportart soll nur angezeigt werden, wenn bei den vorgegebenen Auswahlmöglichkeiten „Sonstiges“ ausgewählt wurde. Im `model` wird dies durch das folgende `bind` Element definiert:

```
<xf:bind nodeset="/data/form/transpmisc"
   relevant="/data/form/transporttype = 'misc'"/>
```

Im Formular ist für `transporttype` ein Ausgabefeld definiert, für `transpmisc` ein Eingabefeld:

```
<xf:select1 ref="/data/context/transporttype">
  <xf:item><xf:label>Zug</xf:label>
     <xf:value>zug</xf:value></xf:item>
  <xf:item><xf:label>Auto</xf:label>
     <xf:value>auto</xf:value></xf:item>
  <xf:item><xf:label>Sonstiges</xf:label>
     <xf:value>misc</xf:value></xf:item>
</xf:select1>
<xf:input ref="/data/form/transpmisc">
  <xf:label>Art:</xf:label>
</xf:input>
```

Beispiel: Setzen eines Feldes auf read-only. Das Feld `curecost` ist nur editierbar, wenn das Feld `reason` auf den Wert `cure` gesetzt ist.

```
<xf:bind nodeset="/data/form/curecost"
   readonly="/data/form/reason != 'cure'"/>
```

Beispiel: Verwendung von Wertelisten (siehe Abschnitt 2.4.1). Hier wird gezeigt, wie in XForms Wertelisten verwendet werden können. Die verschiedenen Typen eines Urlaubs (Erholungsurlaub, Pflegeurlaub etc.) sind in einer Werteliste verwaltet. Die Werteliste kann man im Dokumentenmanagementsystem verwalten. In einem Ordner kann man eine neue Werteliste anlegen, die Id festlegen und die einzelnen Werte und Labels dazu definieren. Im XForm gibt man für die verwendeten Wertelisten ein eigenes `model` Element an:

```
<xf:model id="valuelist">
  <xf:instance
   src="com.groiss.wf.html.ValueList.show?id=holidaytype"/>
</xf:model>
```

Als `src` Attribut muss die dargestellte URL angeführt werden, das Attribut `id` referenziert die Id der Werteliste. Bei Verwendung mehrerer Wertelisten, werden deren Ids durch Kommas getrennt.

Im `body` des XForms gibt es ein Auswahlelement, das auf die Werteliste referenziert:

```
<xf:select1 ref="/data/form/type"><xf:label>Typ:</xf:label>
   <xf:itemset model="valuelist"
     nodeset="/valuelists/list[@id='holidaytype']/item">
      <xf:label ref="label"/>
      <xf:value ref="value"/>
   </xf:itemset>
</xf:select1>
```

In obigem Beispiel werden im model Element die Daten nicht explizit eingetragen sondern es wird eine URL angegeben, mithilfe derer die Daten gelesen werden können.

Beispiel: Konfigurationsdaten. Im Formular soll das in der Konfiguration eingestellte Währungssymbol dargestellt werden. Weiters soll das Kilometergeld mithilfe des in der Konfiguration definierten Werts berechnet werden.

Zur Verwendung von Konfigurationsparametern im Formular ist es nötig, im `model` Element ein `instance` Element mit einem `configuration` Element einzufügen. Darin werden alle benötigten Parameter als property Elemente mit ihrem Namen angegeben. Die Namen bestehen aus der Applikations-Id als Präfix und dem Parameternamen. Für @enterprise Parameter wird kein Präfix angegeben. Zur Laufzeit werden die aktuellen Werte automatisch eingefügt.

```
<xf:instance>
  <data xmlns="">
    <configuration>
      <property name="staffprocs:kmallowance" />
      <property name="staffprocs:dailyallowance.domestic" />
      <property name="staffprocs:currency.symbol" />
    </configuration>
  </data>
</xf:instance>
...
<xf:bind id="currency"
  nodeset="//property[@name='staffprocs:currency.symbol']"/>
<xf:bind nodeset="/data/form/kmamount"
```

```
calculate="//property[@name='staffprocs:kmallowance'] *
           /data/form/km"/>
```

Fassen wir die Datenmodellierung zusammen: Aus dem konzeptuellen Schema werden die einzelnen Formulartypen und ihre Beziehungen zueinander definiert. Zu jedem Formular gibt es eine XHTML Seite mit den eingebetteten XForms Elementen – Eingabefelder und Bind-Elemente. Die Erstellung von solchen Formularen mit @enterprise werden wir in Abschnitt 2.7.4 darstellen.

2.4.3 Formularsichtbarkeiten

Die Definition von Sichtbarkeiten von Formularfeldern mittels der drei Attribute `relevant`, `readonly` und `required` mithilfe von XPath Ausdrücken ist aufwändig und unübersichtlich, wenn die Sichtbarkeiten für jedes Feld in jedem Prozessschritt festgelegt werden müssen. Wir bilden daher die vier sinnvollen Kombinationen der Attribute wie folgt in vier Modi ab:

Modus	relevant	readonly	required
unsichtbar (inv)	false	false	false
nur lesbar (ro)	true	true	false
beschreibbar (rw)	true	false	false
Pflichtfeld (man)	true	false	true

Für Subformulare (eingebettete Tabellen) gibt es einen zusätzlichen Modus: Formulare änderbar, aber Hinzufügen von neuen Subformularen oder Löschen von Subformularen nicht möglich.

Diese Modi können für alle Formularvariablen pro Formularfeld und Prozessaktivität in einer Tabelle festgelegt werden.

Abb. 2.31 zeigt die Sichtbarkeiten eines Formulars einer Prozessdefinition. Die Spalten der Tabelle sind die einzelnen Prozessaktivitäten, die Zeilen die Formularfelder. In jeder Tabellenzelle ist einer der oben angeführten Modi zu finden.

2.5 Bedingungen

Bei der Modellierung der Prozessabläufe hatten wir bereits mehrfach die Notwendigkeit, Bedingungen zu formulieren. Da diese naturgemäß auf die Pro-

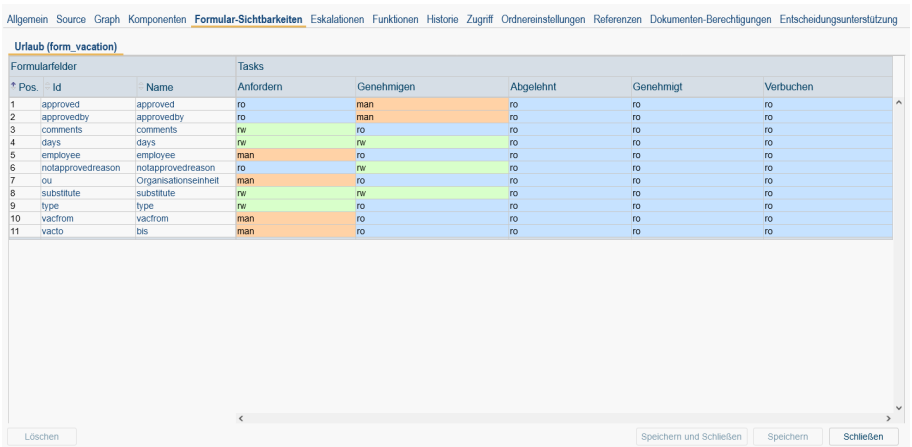


Abbildung 2.31: Formularfeld-Sichtbarkeiten

zessdaten zugreifen, wurde die Definition von Bedingungen verschoben und soll hier nachgeholt werden. Im Prozessablauf treten Bedingungen auf bei:

- ❑ Verzweigungen
- ❑ Schleifen
- ❑ Auswahl (Choice)

Ebenfalls Teil der Prozessdefinition sind Postconditions von Tasks: Bedingungen, die nach der Bearbeitung eines Tasks gelten müssen. Diese Bedingungen können Daten aus mehreren Formularen wie auch andere Daten beinhalten.

Bedingungen können auf drei Arten formuliert werden. WDL erlaubt es, einfache Bedingungen auf Formulardaten direkt zu formulieren, die WDL Beispiele zu If, Choice und Schleifen enthielten bereits solche Bedingungen (vgl. Abschnitt 2.2.5).

Die zweite Möglichkeit ist die Formulierung mit XPath Ausdrücken, welche bereits bei XForms erwähnt wurden. Bei Bedingungen im Prozesskontext stehen die Daten aller Prozessformulare sowie weitere Kontextinformation zur Verfügung. Auf diese Daten wird mittels Variablen zugegriffen, in @enterprise sind folgende Variablen definiert:

- `pi`: die Prozessinstanz
- `ai`: die Aktivitätsinstanz
- `user`: der aktuelle Benutzer
- `form_{id}`: die Formularvariable mit der angegebenen Id
- `configuration`: die Systemkonfiguration
- `configuration_{applid}`: die Konfiguration der Applikation mit der gegebenen Id

Mithilfe dieser Variablen können nun Bedingungen definiert werden.

Beispiel: Der Wert im Feld Gesamtsumme muss größer als 3000 sein:

```
$form_order/totalamount > 3000
```

Die Variable `form_order` verweist auf des Prozessformular `order`, welches ein Feld mit dem Namen `totalamount` (Gesamtsumme) haben muss.

Beispiel: Der Wert des Felds `toagent` im Formular `order` ist gleich dem Akteur, der den Prozess gestartet hat.

```
$form_order/toagent = $pi/agent
```

Wenn komplexe Anforderungen die Möglichkeiten von XPath bezüglich der Formulierung von Bedingungen übersteigen (z.B. Zugriff auf Fremdsysteme), kann man eine beliebige (Java-)Methode zur Definition der Bedingung verwenden. In dieser Methode kann man natürlich auch in einfacher Weise auf Formulardaten und Kontextinformationen zugreifen.

2.6 Funktionen

Interaktive Tasks erfordern vom Bearbeiter die Durchführung von Tätigkeiten, z.B. im einfachsten Fall das Ausfüllen eines Prozessformulars oder das Hinzufügen eines Dokuments. Im Zuge der Bearbeitung können aber auch Programme aufgerufen werden, sodass der Task teilweise automatisiert wird: Zum Beispiel Daten von einem Fremdsystem oder einem anderen Prozess in den aktuellen Prozess übernehmen.

Bei der Modellierung soll dementsprechend auch festgelegt werden, welche Funktionen in welchen Tasks zur Verfügung gestellt werden sollen, damit die Akteure bestmögliche Unterstützung erhalten und von Routinetätigkeiten entlastet werden. Die Funktionen sind hinsichtlich Umfang und Schnittstellen zu System und Benutzer zu definieren.

Neben den Funktionen, die zur Ausführung eines Tasks gehören, gibt es auch globale Funktionen, die unabhängig vom Prozesszustand ausgeführt werden können. Meist sind dies Funktionen zur Bearbeitung von prozessrelevanten Daten, z.B. das Anstoßen einer Datenübernahme in ein externes System.

2.7 BP-Modellierung mit @enterprise

In diesem Abschnitt beschreiben wir, wie die konkrete Modellierung von Prozessen mit @enterprise durchgeführt wird.

Der erste Schritt ist der Zugang zu einer @enterprise Installation. Dafür muss man sich auf der Homepage der Groiss Informatics Gmbh (*https://www.groiss.com*) registrieren (ist unverbindlich und kostenlos). Danach kann man das System herunterladen und lokal installieren oder eine Online-Demo-Instanz erstellen.

In beiden Varianten wählt man bei der Konfiguration eine Benutzer-ID und ein Passwort, mit dem man sich einloggen kann. Nach dem Einloggen ist man in der Arbeitskorbumgebung von @enterprise (vgl. Abb. 3.7). Ein Klick auf den Administrationslink (im Toolbar rechts außen die Benutzerdetails öffnen, dort Administration auswählen 🖳) öffnet die Administrationskonsole, die in Abb. 2.32 dargestellt ist.

Von hier sind alle Funktionen zur Modellierung erreichbar. Eine ausführliche Beschreibung findet sich im @enterprise Administrationshandbuch [**?**]. Auf der linken Seite des Fensters befindet sich die Navigation mit Links zu den einzelnen Funktionen der Administration. Oben befindet sich der Toolbar, der die Funktionen, die auf die gerade ausgewählte Seite anwendbar sind, anzeigt.

2.7.1 Modellierung der Organisation

Die erste Gruppe der Navigationslinks dient zur Erfassung der Organisationsstrukturen. Die Bearbeitung der einzelnen Objekte funktioniert für alle Objektklassen gleich. Beim Klick auf den Navigationslink wird die Tabelle angezeigt. Mit Doppelklick auf einen Tabelleneintrag wird dieser geöffnet. Im

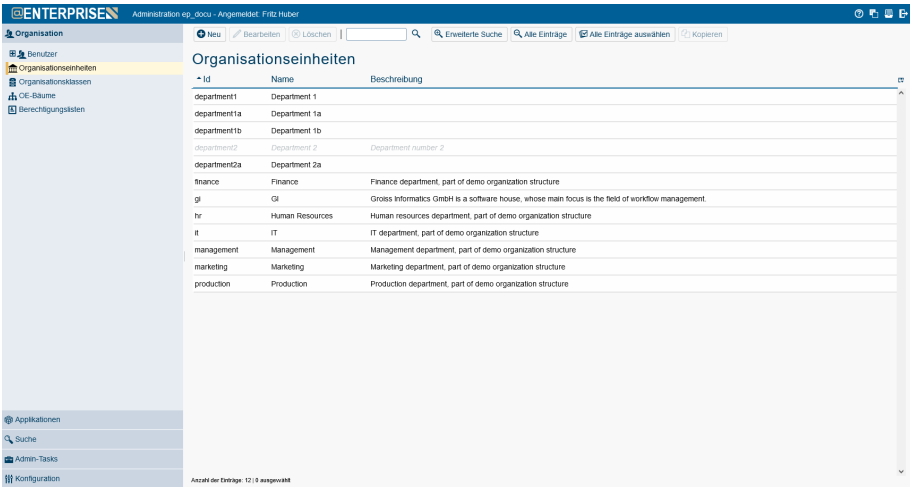


Abbildung 2.32: @enterprise Administration

Toolbar sind die Funktionen Neu, Bearbeiten (entspricht Doppelklick), Löschen und Kurzsuche verfügbar.

Um ein neues Objekt, z.B. einen Benutzer, anzulegen, klickt man im Toolbar auf *Neu*. Im sich öffnenden Fenster sind dann die Mussfelder (kenntlich durch Beschriftung in Fettschrift) auszufüllen. Die Eingabe wird mit Klick auf *OK* oder *Übernehmen* abgeschlossen.

Jedes Objekt hat ein Schlüsselfeld, die sogenannte Id. Dieses Feld darf nicht leer sein und darf keine Sonderzeichen oder Leerzeichen enthalten. Es dient zur eindeutigen Identifikation des Objekts. Bei den Organisationsdaten gibt es als weiteres Mussfeld noch den Namen, beim Benutzer den Nachnamen.

Am besten beginnt man mit der Erfassung der Organisationseinheiten. Dies kann entweder in der beschriebenen Weise in der Liste der Organisationseinheiten geschehen oder aus der Organisationshierarchie erfolgen: Im Detailfenster des Default OE-Baums auf den Reiter Hierarchie klicken und dort die neuen OEs direkt unter die übergeordneten einhängen.

Im nächsten Schritt definieren wir Benutzer und ordnen sie Organisationseinheiten zu. Dazu gehen wir in der Navigation zu Benutzer und klicken dann im Toolbar auf *Neu*. Nach Eingabe von Id, Vorname und Nachname kann mit *Übernehmen* gespeichert werden. Klickt man nun auf den zweiten Reiter der

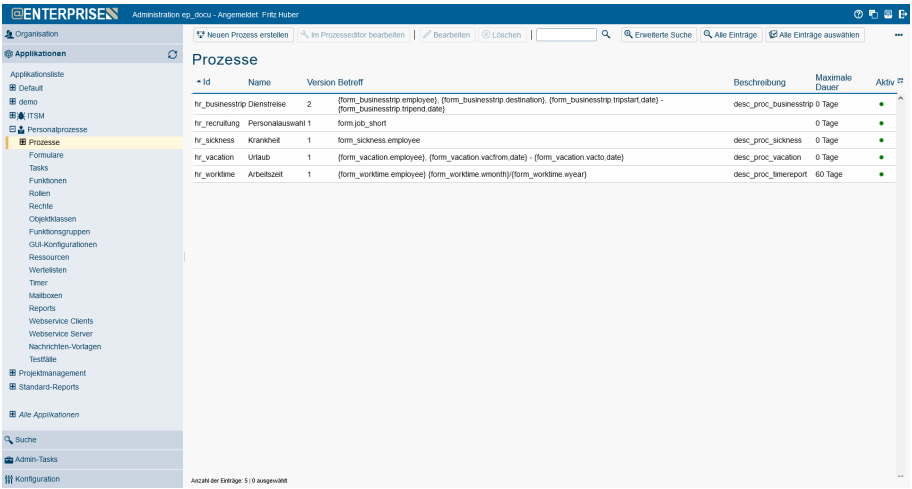


Abbildung 2.33: Liste der Prozesse einer Applikation

Benutzermaske - Rollenzuordnungen, kann man dem Benutzer mit der *home* Rolle eine Organisationseinheit zuordnen.

Für weitere Rollenzuordnungen müssen zuerst die Rollen definiert werden, dies ist weiter unten beschrieben.

2.7.2 Modellierung der Prozesse

Die zweite Gruppe der Navigationslinks ist für die Applikationen bestimmt. Unter einer Applikation fassen wir eine Gruppe von Prozessen und ihren Bestandteilen zusammen. Um einen Prozess anlegen zu können, muss zuerst eine Applikation angelegt werden. Beispiele für Applikationen sind z.B. die Prozessgruppen aus Abschnitt 1: Personalprozesse, IT-Prozesse.

Nachdem man eine Applikation neu angelegt hat, erscheint eine neue Gruppe von Navigationslinks unter dem Namen der Applikation im Baum. Der erste Link führt zur Liste der Prozesse, siehe Abb. 2.33. Prozesse werden mit dem Prozesseditor erstellt, der über den ersten Link im Toolbar aufgerufen werden kann.

Der Prozesseditor

Der Prozesseditor dient dazu, den Prozessablauf grafisch zu modellieren. Die dabei benötigten Komponenten, Tasks und Rollen, können ebenfalls im Prozesseditor erstellt werden.

Mit dem Prozesseditor kann ein Prozess definiert werden, der nach der Definition sofort mit der Engine ausgeführt werden kann. Es ist aber auch möglich, den Prozesseditor in einer früheren Projektphase zur Modellierung zu verwenden, wo der Prozess noch nicht so detailliert ausformuliert ist, dass eine unmittelbare Ausführung möglich wäre. Zum Beispiel wenn die Prozessdaten noch nicht modelliert sind oder die Bedingungen in Verzweigungen und Schleifen nur verbal beschrieben sind.

Wenn man einen neuen Prozess erstellt, vergibt man zuerst eine Id und einen Namen, indem man im Prozess-Menü Eigenschaften auswählt und in der folgenden Maske beides einträgt.

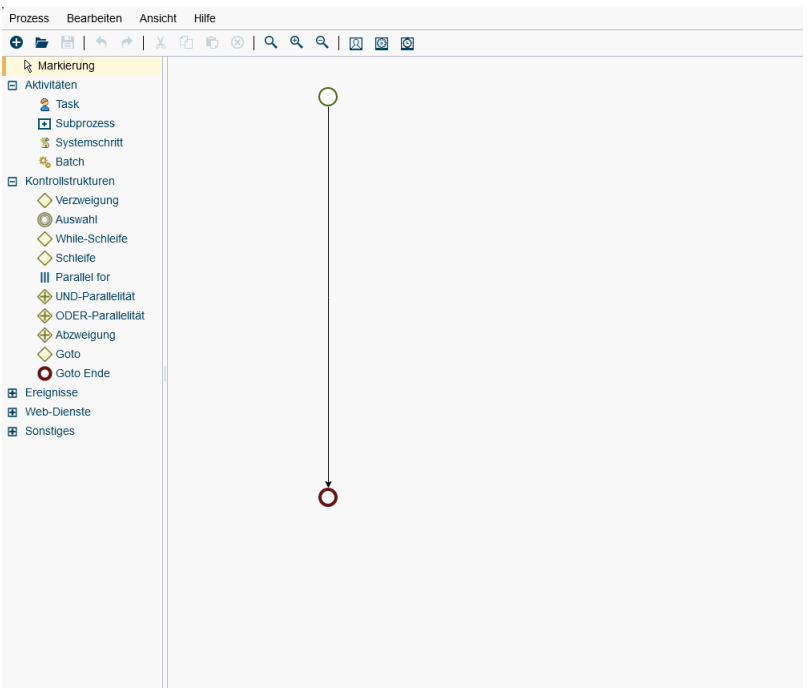


Abbildung 2.34: Prozesseditor

Ein neuer, leerer Prozess besteht aus Anfangsknoten, Endknoten und der sie verbindenden Kante (siehe Abb. 2.34). Prozessknoten werden eingefügt, indem man links eine der Knotentypen anklickt und dann auf eine Kante klickt. Ein Knoten oder eine Kontrollstruktur vom ausgewählten Typ wird dann an dieser Stelle eingefügt, die ursprüngliche Kante wird entfernt. Beim Löschen wird der umgekehrte Vorgang ausgeführt.

Im Prinzip kann jede Kante das Ziel einer solchen Ersetzung sein, außer der Kante zwischen Choice Knoten und der Bedingung.

Bei den meisten Knotentypen ist es nach dem Einfügen erforderlich, die Eigenschaften festzulegen. Bei Doppelklick auf den Knoten öffnet sich das Eigenschaftenfenster, das je nach Knotentyp verschiedene Eingaben erlaubt.

Falls der Prozess (noch) nicht ausführbar sein muss, sind bei den Knoten keine Eingaben verpflichtend. Man kann also nichts oder nur eine Beschreibung (die in der Grafik aufscheint) eingeben. Ansonsten sind die folgenden Informationen nötig:

- Task-Knoten: ein interaktiver Schritt (Task) kann ausgewählt oder neu erstellt werden. Weiters kann ein Akteur bestimmt werden.
- Subprozess: Es muss ein Prozess als Subprozess ausgewählt werden.
- Bedingungen: Für Schleifen und Verzweigungen müssen die Bedingungen definiert werden.
- Choice-Pfad: Jeder Pfad ist mit einem Namen zu versehen, die Bedingung ist optional.

Weitere Informationen über die Bedienung des Prozesseditor kann dem Administrationshandbuch entnommen werden [**?**].

2.7.3 Tasks und Rollen

Tasks und Rollen können während der Prozessdefinition im Prozesseditor angelegt werden, können aber auch unabhängig von Prozessen in der jeweiligen Tabelle verwaltet werden.

Vor allem Rollen wie Abteilungsleiter oder Sachbearbeiter, die unabhängig von einem konkreten Prozess sind, werden so angelegt. Es empfiehlt sich, die prozessunabhängigen Rollen in einer Applikation, z.B. der Default-Applikation zusammenzufassen.

2.7.4 Formulare

Die Modellierung der Daten erfolgt unter dem Navigationslink *Formulare* in der Navigationsgruppe der Applikation (siehe Abb. 2.33).

Ein Klick auf den ersten Toolbareintrag „Neues Formular erstellen“ öffnet den Formular-Editor (siehe Abb. 2.35), in dem sowohl das Layout als auch die Datentypen definiert werden können:

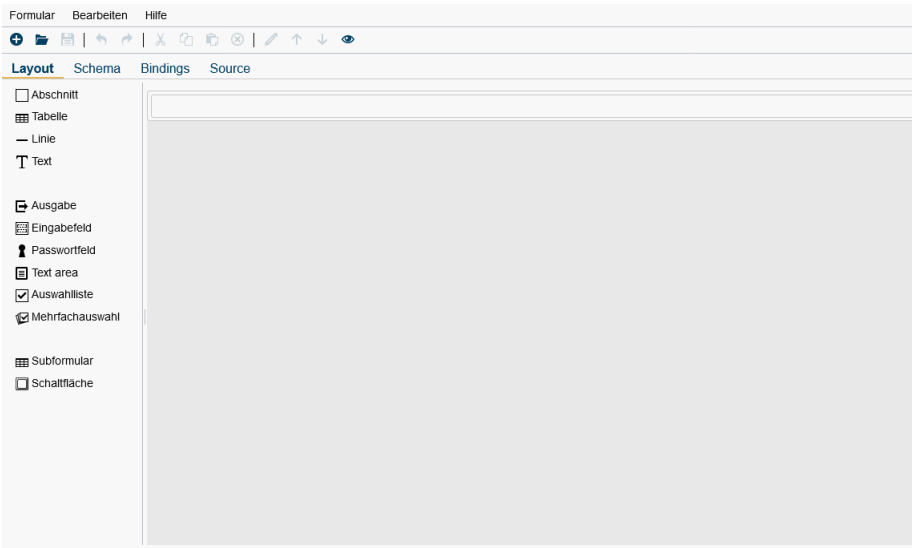


Abbildung 2.35: Formularerstellung

- Im ersten Reiter wird das HTML Layout festgelegt. Die einzelnen Eingabeelemente (siehe den linken Bereich in der Abbildung) können mit drag-and-drop zum Formular hinzugefügt werden.
- Im zweiten Reiter - Schema - können die einzelnen Datenbankfelder definiert werden: Namen, Typen, Länge der Felder in der Datenbank etc., siehe Abb. 2.36. Das Layout kann auch aus diesen Schemainformationen erstellt werden (Menüfunktion: Standard-Layout erstellen).
- Im Reiter Bindings können XForms-Bindings für die Felder definiert werden.

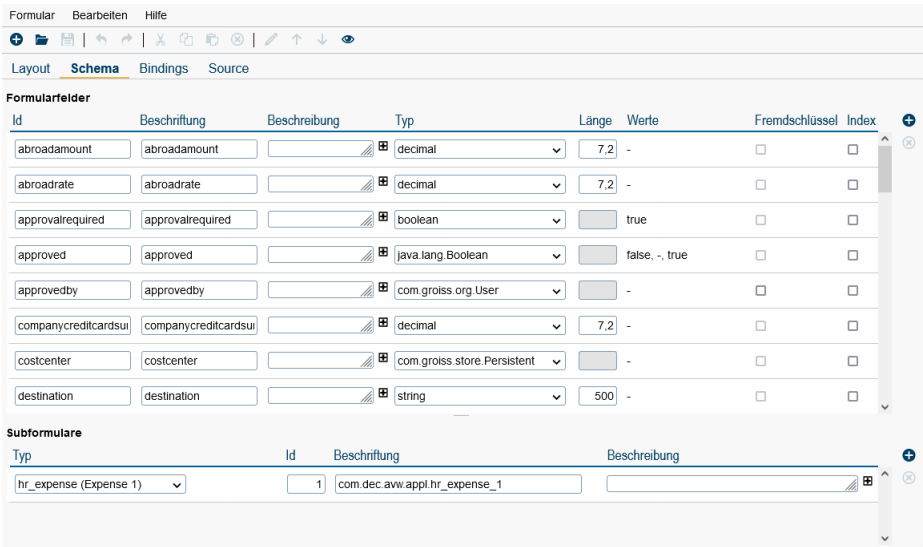


Abbildung 2.36: Formularerstellung - Festlegung der Felder und Typen

- ❑ Im letzten Reiter, Source, kann das HTML direkt editiert werden.

Beim Speichern muss man einen Namen und eine Id vergeben, danach wird die Java-Klasse erstellt, in der Datenbank eine Tabelle angelegt und das HTML im Filesystem abgelegt.

2.7.5 Prozessdokumentation

Mit @enterprise modellierte Prozesse können mit der Funktion „Prozessübersicht“ dargestellt werden. Dabei werden in einer HTML-Seite oder einem PDF-Dokument alle Aspekte des Prozesses dargestellt: graphische Definition, WDL, Tasks, Rollen, Formulare, Fomularsichtbarkeiten. Abb. 2.37 zeigt die HTML Darstellung (mit dem Icon rechts oben läßt sich das PDF Format erstellen).

Die Prozessübersicht dient als Dokumentation und zur Diskussion mit Prozessbeteiligten. In Abschnitt 4.1.3 werden wir das Prozess-Cockpit vorstellen, eine Darstellung der Menge der definierten Prozesse mit zusätzlichen Laufzeitinformationen.

Urlaub

1. Allgemein

Name	Urlaub
Applikation	Personalprozesse
Beschreibung	desc_proc_vacation
Id (Version)	hr_vacation (1)
Übersetzungen	Englisch: Vacation, Deutsch: Urlaub
Formulare	Id: form_vacation, Name: Urlaub, Typ: Urlaub
Start	Manuell: Alle
Eskalation	Eskalationstyp Zeitabstand Aktion Aktionsdetails Beschreibung Prozessfälligkeit 1 Tag E-Mail senden

2. WDL

3. Graph

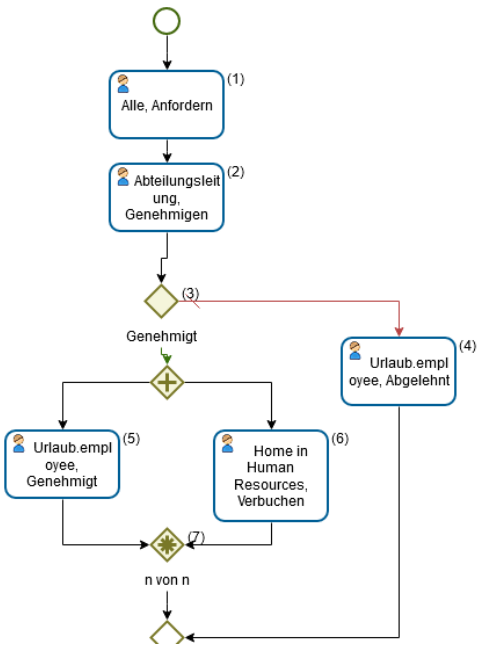


Abbildung 2.37: Prozessdokumentation

2.8 Ein vollständiges Beispiel

Als Abschluss des Modellierungskapitels möchten wir noch ein vollständiges Beispiel präsentieren. Die Aufgabenstellung ist folgende:

Es soll ein Prozess zur Beantragung, Genehmigung und Vergabe von Berechtigungen an Mitarbeiter erstellt werden. Unter einer Berechtigung ist die Zuordnung einer Ressource zu einem Mitarbeiter zu verstehen. Dies kann in verschiedenen Funktionen (oder Rollen) erfolgen. Beispielsweise kann für die Ressource „SAP“ die Funktion „Superuser“ vergeben werden.

Jeder Benutzer soll für sich selbst Berechtigungen beantragen können. Der Antrag muss danach von einem Abteilungsleiter genehmigt werden, wobei dieser auch einzelne Berechtigungen aus dem Antrag entfernen kann. Nach der Genehmigung erfolgt die Berechtigungsvergabe. Dies wird für jede Ressource von einem für diese Ressource verantwortlichen Mitarbeiter durchgeführt. Schließlich wird der Antragsteller von der Erledigung informiert.

2.8.1 Daten

Wir entwerfen zuerst die Datenstrukturen. Abb. 2.38 zeigt das Datenschema.

Die Ressourcen haben einen Namen und einen Verantwortlichen und eine 1:n Beziehung zu Funktionen. Für die Funktionen könnten die @enterprise Rollen direkt verwendet werden, flexibler für spätere Erweiterungen ist allerdings eine eigene Funktionsklasse.

Der Berechtigungsantrag erhält ein Feld Antragsteller, die Organisationseinheit, eine 1:n Beziehung zu den Berechtigungen, ein Boole’sches Feld für die Genehmigung und eine Begründung. In einem Berechtigungsantrag werden n Berechtigungen beantragt, diese Berechtigungen enthalten eine Ressource und eine Funktion.

Der nächste Schritt ist die Abbildung auf Formulare: Aus jeder der Entities wird ein Formular, die Funktion wird als Subformular von Ressource und Berechtigung als Subformular des Berechtigungsantrags modelliert.

Die Formulare können nun mit dem Form-Wizard erzeugt werden. Die Ablage der Ressourcen erfolgt dann in einem Dokumentenordner. Die Abbildungen 2.39 und 2.40 zeigen den Formularquelltext und die Darstellung im Browser.

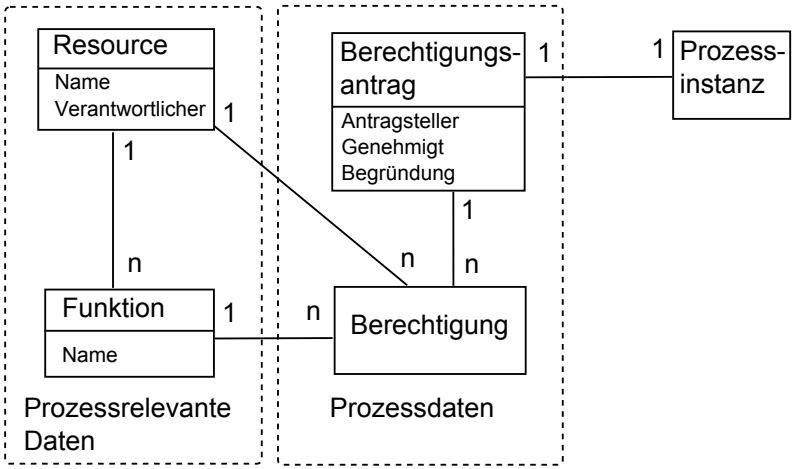


Abbildung 2.38: Datenschema zum Berechtigungsprozess

2.8.2 Prozess

Der Prozess ist in Abb. 2.41 dargestellt, das WDL-Script in Abb. 2.42.

Man beachte die Definition der Akteure. Der erste Schritt ist der Rolle `all` zugeordnet, jeder der diese Rolle hat, darf den Prozess starten. Danach genehmigt der `leiter`. Die Vergabe der Rechte erfolgt im `parfor` für jede Berechtigung parallel. Der Akteur ist der für die jeweilige Ressource Verantwortliche. Da dies in WDL nicht direkt ausdrückbar ist, fungiert hier eine Funktion als Akteursangabe. Im letzten Schritt wird der Akteur des ersten Schritts verständigt.

Wenn Prozess und Datenstruktur definiert sind, können die Feldberechtigungen der einzelnen Schritte festgelegt werden, siehe Abb. 2.43.

In jeder Zeile ist das typische Muster so, dass ein Feld zuerst unsichtbar ist (in diesem Schritt noch nicht relevant), dann schreibbar (`rw`) oder ein Mussfeld (`man`), danach nur mehr lesbar (`ro`). Normalerweise wird das Prozessformular von oben nach unten ausgefüllt, d.h. zuerst kommen die Felder, die im ersten Schritt ausgeführt werden, dann die vom zweiten etc. Die Matrix der Feldberechtigungen enthält damit meist einen diagonalen Balken mit Beschriftungen `rw` oder `man` von links oben nach rechts unten.

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:xf="http://www.w3.org/2002/xforms">
 <head>
  <link rel="stylesheet" type="text/css"
    href="../html/avw.css"></link>
  <xf:model></xf:model>
 </head>
 <body id="theform">
  <div id="title" class="title">Berechtigungen</div>
  <table id="formtab">
   <tr><td>
     <xf:input ref="/data/form/applicant">
       <xf:label class="label100">Antragsteller</xf:label>
     </xf:input>
   </td></tr>
   <tr><td>
     <xf:repeat formtype="com.dec.avw.appl.demo_permisson_1"
       subformid="1">
      <xf:label class="label100">Berechtigungen</xf:label>
      </xf:repeat>
   </td></tr>
   <tr><td>
     <xf:input ref="/data/form/approved">
      <xf:label class="label100">genehmigt</xf:label>
     </xf:input>
   </td></tr>
  </table>
 </body>
</html>
```

Abbildung 2.39: Formular Berechtigungsantrag

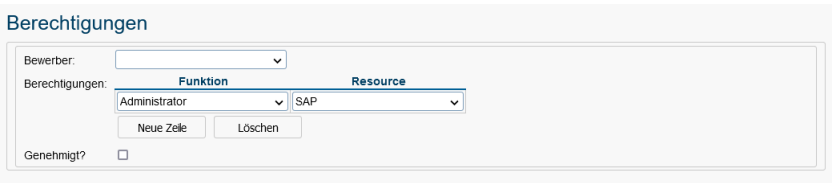


Abbildung 2.40: Berechtigungsantrag im Browser

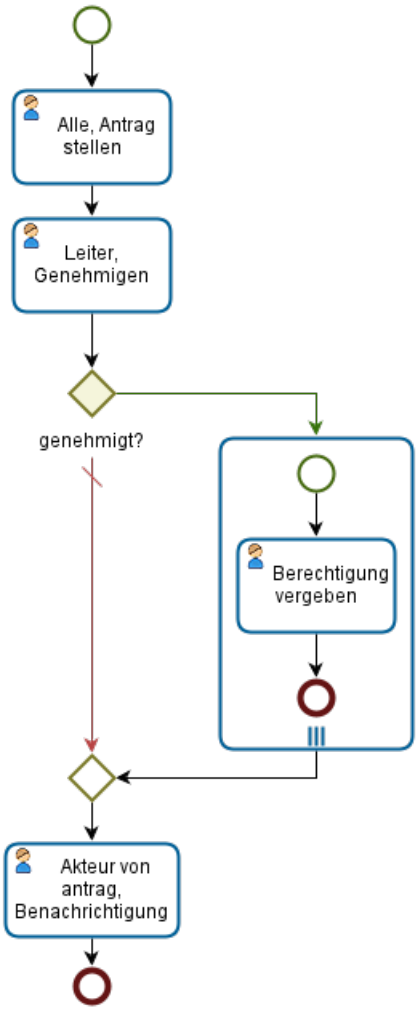


Abbildung 2.41: Berechtigungsprozess grafisch

```
process br_assignpermissions()
version 5;
name "Berechtigungsvergabe";
forms form br_permissions "Form";
timeoutaction none;
application permissions;
begin
   <antrag> all br_antrag_stellen(form);
   br_leiter br_genehmigen(form);
   if com.groiss.wf.SystemAction.getFormFieldValue(
        "form.approved") then "genehmigt?"
     parallel for permission "Berechtigung" in form.1 do
       com.groiss.wf.SystemAction.getFormFieldValue(
       "permission.resource.responsible")
       br_berechtigung_vergeben(form) "Berechtigung vergeben";
      end;
   end;
   antrag:user br_inform(form);
end
```

Abbildung 2.42: Berechtigungsprozess WDL

Formularfelder	Antrag stellen	Genehmigen	Benachrichtigung
Antragsteller	man	ro	ro
Berechtigungen	rw	rw	ro
genehmigt	inv	man	ro
genehmigt von	inv	man	ro

Abbildung 2.43: Sichtbarkeiten

Kapitel 3

Prozessausführung

Nachdem wir uns im vorigen Kapitel ausführlich mit der Modellierung beschäftigt haben, geht es in diesem Kapitel um die Ausführung von Prozessen in einem BPM-System. Wir beschreiben zuerst die Architektur eines BPM-Systems und einer Workflow-Engine, stellen danach die Benutzerschnittstelle des Systems, das Berechtigungssystem und die Systemintegrationen vor.

3.1 Architektur

Ein BPMS enthält alle Komponenten, die zur Modellierung, Ausführung und Laufzeit-Analyse von Prozessen nötig sind. Die bereits erwähnte Workflow Management Coalition hat im Dokument Terminology and Glossary [**?**] ein Referenzmodell präsentiert. Unser Modell in Abb 3.1 ist daran angelehnt. Es enthält die folgenden Komponenten:

- Modellierung: Die Modellierungskomponente dient zur Erstellung von Prozessdefinitionen, Datenstrukturen, Organisationsstrukturen sowie zur Definition aller weiteren Applikationskomponenten.
- Laufzeitkomponente: Für die Ausführung von Prozessen ist die Workflow-Engine verantwortlich, zur Verwaltung der Dokumente das Dokumentenmanagementsystem, für zeitgesteuerte Aktionen der Timer.

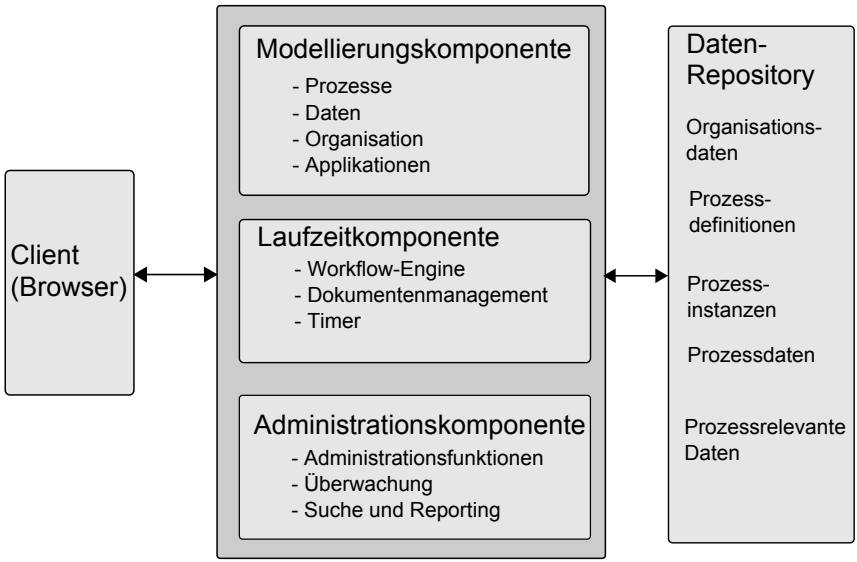


Abbildung 3.1: Architektur eines BPMS

- Administrationskomponente: Verwaltung der Benutzer, Organisationsstrukturen, Such- und Reportingfunktionen.
- Daten-Repository: Enthält die Definitions- und Laufzeitdaten.
- Client: Als Client kommt im Normalfall ein herkömmlicher Web-Browser zum Einsatz, eventuell angereichert mit Plugins, z.B. Java. Alternativ kann auch ein spezieller BPM Client zum Einsatz kommen.

Die Funktion der einzelnen Komponenten wird in den folgenden Abschnitten erläutert.

3.2 Die Workflow-Engine

Die Workflow-Engine ist für den Ablauf der Prozesse zuständig. Sie interpretiert die Prozessdefinition und erzeugt Prozessinstanzen und Aktivitätsinstanzen.

Die Arbeit der Workflow-Engine kann mit zwei Funktionen beschrieben werden: `startActivity` und `finishActivity`. Die Funktionen sind verkürzt in Pseudo-Code in Abb. 3.2 dargestellt.

Wenn ein Prozess gestartet wird, wird die Funktion `startActivity` mit dem initialen Knoten des Prozessgraphen aufgerufen. Als erstes wird ein Objekt vom Typ Aktivitätsinstanz angelegt. Das Verhalten der Funktion hängt nun vom Typ des Prozessknotens ab: Ist der Typ `begin`, `par`, `loop`, `end_if` oder `end` wird sofort die Funktion `finishActivity` aufgerufen (siehe unten). Ist der Knoten eine Bedingung (`if`, `while`, `exit_when`), wird die zugehörige Bedingung ausgewertet und je nach Ergebnis entweder der `then` oder `else` Zweig weiterverfolgt, d.h. `finishActivity` wird mit dem Nachfolgeknoten auf dem entsprechenden Zweig aufgerufen. Die beiden Knoten, die eine Parallelität beenden, `orjoin` und `andjoin` werden folgendermaßen behandelt: Beim `orjoin` wird sofort mit dem Nachfolgeknoten fortgefahren, beim `andjoin` passiert nichts, solange nicht alle Zweige beendet sind. Wenn der Knoten ein Task-Knoten ist, wird die optionale Preprocessing Methode ausgeführt, der Akteur zugeordnet und die Funktion beendet. Das heißt die Funktion läuft solange, bis der erste interaktive Schritt erreicht ist. Bei einem Systemschritt wird die beim Schritt angegebene Prozedur ausgeführt und dann wieder `finishActivity` aufgerufen. Bei einem Subprozessaufruf wird der erste Schritt des Subprozesses ermittelt und danach die Methode rekursiv mit diesem Schritt als Parameter aufgerufen.

Wenn der Benutzer eine Aktivität beendet, wird die Funktion `finishActivity` aufgerufen (Funktion *Weiterleiten* im Arbeitskorb). Diese sucht alle Nachfolger des aktuellen Knotens auf den Kanten des angegebenen Typs (`normal`, `then`, `else`) und ruft für diese die Funktion `startActivity` auf.

Es sei nochmals erwähnt, dass die Funktionen hier etwas vereinfacht sind, die Konstrukte `batch`, `parfor` und `event` wurden weggelassen.

3.2.1 Struktur der Laufzeitdaten

Wenn ein Prozess gestartet wird, wird ein Prozessinstanz-Objekt in der Datenbank angelegt. Dieses Objekt erhält eine eindeutige Prozess-Id. Für jede Aktivität, die gestartet wird (d.h. jedesmal wenn die Prozedur `startActivity` aufgerufen wird), wird eine Aktivitätsinstanz angelegt. Diese Objekte (und weitere folgende) nennen wir Laufzeitdaten (run-time data), da sie zur Laufzeit der Prozessinstanz erzeugt werden. Im Gegenzug dazu heißen die bei der Modellierung angelegten Objekte Modelldaten (build-time data).

```
function startActivity(node)
   ActivityInstance ai := create_instanceof(node);
   if typeOf(node) in {begin, par, loop, end_if, end} then
      finishActivity(ai, "normal");

   elsif typeOf(node) in {if, while, exit_when} then
      if executeExpression(node)
         then finishActivity(ai,"then");
         else finishActivity(ai,"else");
      end if;

   elsif typeOf(node) = orjoin then
      if this is the first finished branch then
         finishActivity(ai, "normal");
      end if;

   elsif typeOf(node) = andjoin then
      if this is the last finished branch then
         finishActivity(ai, "normal");
      end if;

   elsif typeOf(node) = task then
      executeProcedure(ai);
      assignAgent(ai);

   elsif typeOf(node) = process then
      node1 := initial node of process node
      startActivity(node1);

   elsif typeOf(node) = system then
      executeProcedure(ai);
      finishActivity(ai, "normal");
   end if;
end;

function finishActivity(ai, edgetype)
   if no successors of ai then
      finishActivity(parent(ai));
   else
      for all successor nodes succ of the ai node with type edgetype do
         startActivity(succ);
      end do;
   end if;
end;
```

Abbildung 3.2: Zentrale Funktionen der Workflow-Engine

Die Zustände von Prozessinstanzen und Aktivitätsinstanzen sind in den Abbildungen 3.3 und 3.4 dargestellt. Nachdem ein Prozess gestartet wird, hat er den Zustand *gestartet*. Wenn er beendet wurde (der letzte Schritt beendet wurde) erhält er den Zustand *beendet*. Aus diesem Zustand kann er reaktiviert werden. Ein laufender Prozess kann jederzeit abgebrochen werden, er erhält den Zustand *abgebrochen*. Aus diesem Zustand ist ebenfalls ein Reaktivieren möglich.

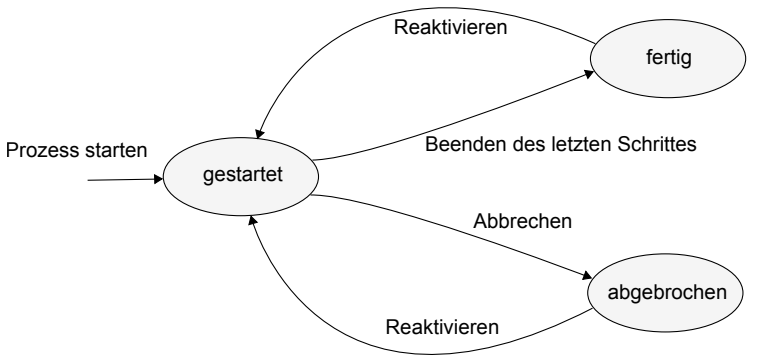


Abbildung 3.3: Prozesszustände

Wenn eine Aktivitätsinstanz erzeugt wird, ist sie zunächst im Zustand *gestartet* oder *aktiv*, je nachdem, ob der Akteur eine Rolle ist (Status gestartet) oder ein Benutzer (Status aktiv). Der Zustand *suspendiert* bedeutet, dass die Bearbeitung der Aktivität unterbrochen wurde. Das Weiterleiten führt (im Normalfall) zum Zustand *beendet*. Wenn eine Choice-Weg Auswahl oder Akteursauswahl nötig ist, bleibt die Aktivitätsinstanz nach dem Weiterleiten im Zustand *wartend* bis diese Aktionen abgeschlossen sind. Eine bereits beendete Instanz kann durch Zurückgehen in den Zustand *kompensiert* gelangen. Ein Abbruch der Aktivitätsinstanz, z.B. bei Prozessabbruch, führt zum Zustand *abgebrochen*.

Wie hängen nun Prozessinstanzen und Aktivitätsinstanzen zusammen? Beim Prozessstart wird für den Prozess eine Prozessinstanz erzeugt, bei jedem Start einer Aktivität wird eine neue Aktivitätsinstanz als „Kind“ der Prozessinstanz erzeugt. Ist einer der Prozessschritte ein Subprozessaufruf, ist die zugehörige Aktivitätsinstanz gleichzeitig eine Prozessinstanz und hat selbst wieder „Kinder“ in Form weiterer Aktivitätsinstanzen. Abb. 3.5 zeigt den Graphen der Laufzeitdaten, wobei pi1 die Prozessinstanz ist. Der Prozess hat vier Schritte, von denen einer ein Subprozess ist (pi2). Ai1, ai2, und ai2 sind die Aktivitätsin-

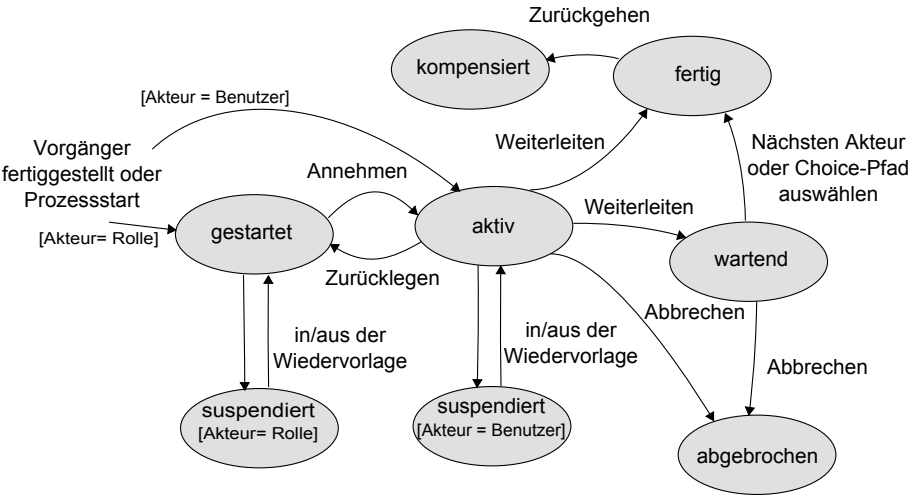


Abbildung 3.4: Taskzustände

stanzen von pi1. Die Objekte ai4, ai5, und ai6 sind die Aktivitätsinstanzen des Subprozesses pi2.

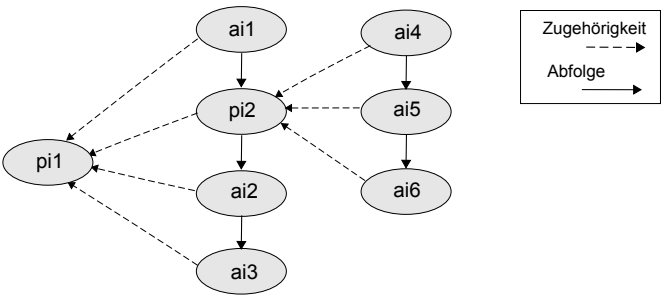


Abbildung 3.5: Laufzeitdaten

Nun betrachten wir, wie die Modelldaten und Laufzeitdaten zusammenhängen. Abb. 3.6 stellt das Schema dar. Der Prozessgraph ist durch die beiden Entitäten Step (die Knoten) und Flow (die Kanten) repräsentiert. Ein Prozessschritt kann auf einen Task (= interaktiver Schritt) oder eine Prozessdefinition (= Subprozess) verweisen.

Die Entität Aktivitätsinstanz bildet die Laufzeitinformation eines Prozesses ab, es repräsentiert auch die Prozessinstanzen. Die Relation zu sich selbst beschreibt die Verbindung zwischen Prozessinstanz und den zugehörigen Aktivitätsinstanzen (die strichlierten Linien aus Abb. 3.5).

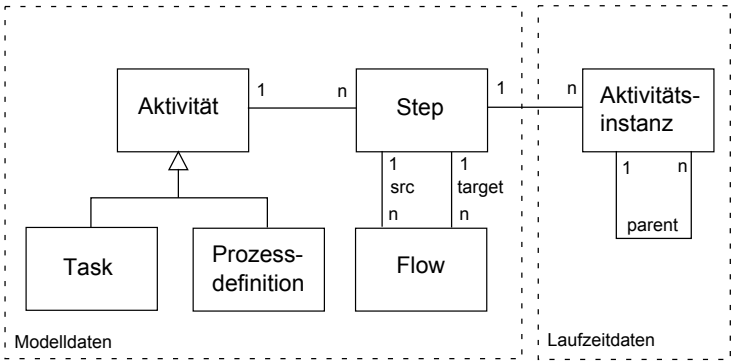


Abbildung 3.6: Schema von Prozessdefinition und Instanz

3.3 Die Benutzerschnittstelle

Die Benutzerschnittstelle eines BPM-Systems dient dazu, die zu bearbeitenden Tasks anzuzeigen und Werkzeuge zur Bearbeitung der Prozessdaten zur Verfügung zu stellen. Abb. 3.7 zeigt die Benutzerschnittstelle von @enterprise. Im Rest dieses Abschnitts wird diese exemplarisch beschreiben.

Die einzelnen Funktionen erreicht man über eine Navigation, in der im Wesentlichen die folgenden Punkte zu finden sind:

- Arbeitskörbe
- Liste der startbaren Prozesse
- Liste von Funktionen der Applikation
- Suchfunktionen und Reports
- Dokumentenmanagement
- Kalender

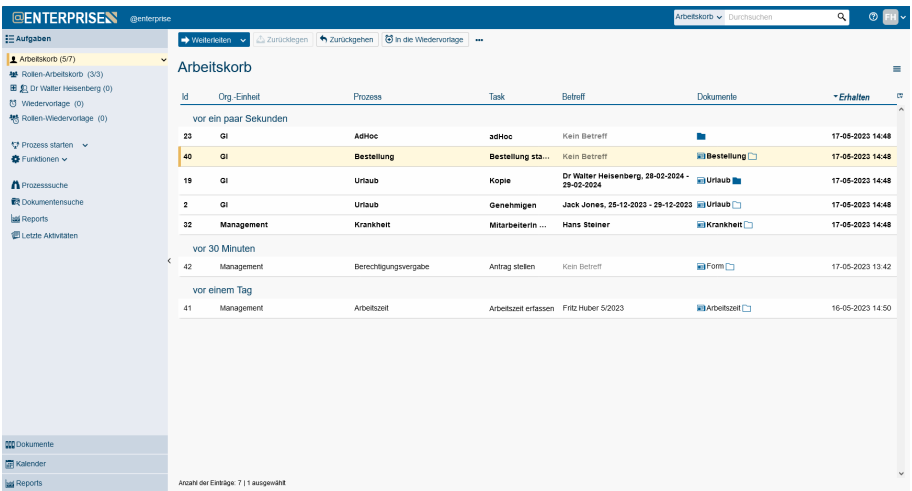


Abbildung 3.7: Benutzerschnittstelle

- ❑ Benutzerspezifische Einstellungen
- ❑ Dashboard
- ❑ Ansicht der Organisation und der Prozesse

Welche Funktionen konkret angeboten werden, hängt von den Rollen des Akteurs und von der verwendeten Konfiguration des Clients ab.

3.3.1 Arbeitskorb

Zentraler Punkt der Benutzerschnittstelle ist der Arbeitskorb (engl. Worklist), der eine Ansicht der zu bearbeitenden Aktivitätsinstanzen enthält. Meist ist dieser in mehrere Listen strukturiert:

- ❑ persönlicher Arbeitskorb: persönlich zugeordnete Aktivitätsinstanzen, entweder direkt an einen Benutzer adressiert oder aus dem Rollenarbeitskorb angenommen
- ❑ Rollen-Arbeitskorb: Aktivitätsinstanzen, die einer Rolle, welche der Benutzer innehat, zugeordnet sind

- Wiedervorlage: persönlich zugeordnet, aber derzeit nicht bereit zur Bearbeitung
- Rollen-Wiedervorlage: einer Rolle zugeordnet und derzeit nicht für die Bearbeitung bereit.

Die Aufteilung entspricht den vier Zuständen gestartet, aktiv, suspendiert/Benutzer und suspendiert/Rolle aus dem Zustandsdiagramm für Aktivitäten (Abb. 3.4).[1]

Darüberhinaus kann es weitere, auch individuelle Strukturierungen geben, z.B.: mache ich heute, morgen, dringend, wenn Kurt wieder da ist etc. Der Arbeitskorb enthält ja die Aufgaben, die von einem Mitarbeiter in einem Zeitraum zu erledigen sind. Bei einem hohen Automatisierungsgrad, wenn viele verschiedene Aufgaben über das BPMS bearbeitet werden, können sich hundert oder mehr Einträge in einem Arbeitskorb befinden. Der Möglichkeit zur individuellen Strukturierung kommt daher hohe Bedeutung zu.

Neben der Strukturierung in unterschiedliche Listen gibt es die Möglichkeit die Listen nach bestimmten Inhalten zu filtern, z.B.:

- nur Prozesse eines Typs
- keine Support Prozesse
- heute gekommen

Die Arbeitskörbe sind Tabellen mit Aktivitätsinstanzen, die einzelnen Spalten stellen deren wesentliche Eigenschaften dar, diese sind:

- Prozessinstanz-Id: die eindeutige Identifikation der Prozessinstanz
- Organisationseinheit: die OE, der die Aktivitätsinstanz zugeordnet ist
- Akteur: der aktuelle Akteur; diese Spalte ist im persönlichen Arbeitskorb verzichtbar, da sie dort nur den Benutzernamen anzeigt
- Prozessname: Name der Prozessdefinition
- Taskname: Name des Tasks

[1] Aktivitäten mit den Zuständen kompensiert, fertig und abgebrochen können nicht bearbeitet werden. Aktivitäten mit dem Zustand wartend werden - falls vorhanden - im Arbeitskorb mit einem Hinweis, welche Aktion noch auszuführen ist, angezeigt.

- Betreff: eine inhaltliche Identifikation des Prozesses
- Funktionen: Liste der für diese Aktivität verfügbaren spezifischen Funktionen
- Formulare: Verweise (Links) zu den Prozessdaten
- Erhalten: wann wurde die Aktivität gestartet
- Fertig bis: wann soll die Aktivität beendet sein
- Priorität: Priorität des Prozesses

In der Listenansicht müssen nicht alle diese Informationen angezeigt werden, oder werden nur in einer Kurzform - als Verweis, der zur ausführlichen Information führt - angezeigt. Weitere Informationen, die vom Arbeitskorbeintrag verlinkt sind:

- wo bin ich im Prozess: grafische Darstellung des Prozesses mit Kennzeichnung des aktuellen Schritts
- bisheriger Ablauf: die Historie des Prozesses: wer hat ihn gestartet, bisher bearbeitet, inklusive der Zeitstempel
- die Daten des Prozesses: Formulare, Notizen und Dokumente

Abb. 3.8 zeigt die Detailansicht eines Arbeitskorbeintrags, die einzelnen Funktionen werden im folgenden Abschnitt erläutert.

3.3.2 Funktionen des Arbeitskorbs

Der Arbeitskorb dient zur Anzeige und der Bearbeitung von Aktivitäten. Die wesentlichen Bearbeitungsfunktionen sind einerseits die Aktionen der Workflow-Engine, wie Prozess starten oder Weiterleiten, andererseits die Bearbeitung der Daten (Formulare und Dokumente).

Aktionen zur Steuerung von Prozessen und Aktivitäten

- Weiterleiten: Der Schritt wird beendet, indem die entsprechende Funktion der Workflow-Engine aufgerufen wird (die Funktion `finishActivity` aus Abschnitt 3.2).

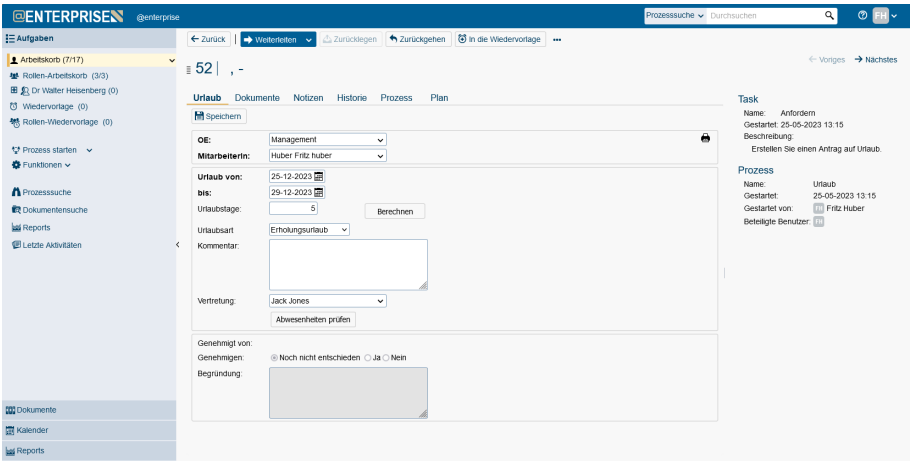


Abbildung 3.8: Detailansicht eines Arbeitskorbeintrags

- Weiterleiten und Auswahl: Wenn der Prozess nach dem aktuellen Schritt einen weiteren interaktiven Schritt enthält, der an eine Rolle adressiert ist, ist es oft wünschenswert, einen konkreten Benutzer aus der Menge der Rolleninhaber auszuwählen. Bei Ausführung dieser Funktion erfolgt nach dem Weiterleiten die Benutzerauswahl.
- Zurückgehen: Wenn der Prozess vor dem aktuellen Schritt bereits von anderen Personen bearbeitet wurde, gibt es die Möglichkeit, den Prozess zu einem der früheren Schritte zurückzuschicken. Die Implikationen dieser Funktion werden in Kapitel 3.5.1 behandelt.
- Akteur ändern: Abtreten der Bearbeitung des Prozesses an eine andere Person. Der Prozess bleibt im aktuellen Schritt, es wird nur der Akteur des aktuellen Schrittes geändert. In der Historie des Prozesses ist diese Aktion nachvollziehbar.
- Kopie an: Verschicken einer Read-only Kopie des Prozesses an eine andere Person, z.B. zu Informationszwecken.
- Zurücklegen: Wenn die Aktivität aus dem Rollenarbeitskorb angenommen wurde, kann sie auch dahin zurückgelegt werden.
- In die Wiedervorlage: Kann oder will die Aktivität zurzeit nicht bearbeitet werden, kann sie in die Wiedervorlage gelegt werden. Dabei wird

ein Datum vermerkt, an dem die Aktivität wieder automatisch in den Arbeitskorb zurückkehrt.

- ❑ Prozess abbrechen: Der Prozess wird willentlich aber außerplanmäßig beendet.

Funktionen zur Bearbeitung der Daten

Diese Funktionen stehen in der Detailansicht der Aktivitätsinstanz zu Verfügung und sind neben der Hauptfunktion der Workflow-Engine, dem Weiterleiten, die zentralen Tätigkeiten beim Bearbeiten einer Prozessinstanz.

- ❑ Formulare bearbeiten: In der Detailansicht zeigen die ersten Reiter die Formulare. Sie werden dabei mit den in der Prozessdefinition für diesen Schritt vorgesehenen Sichtbarkeiten angezeigt.

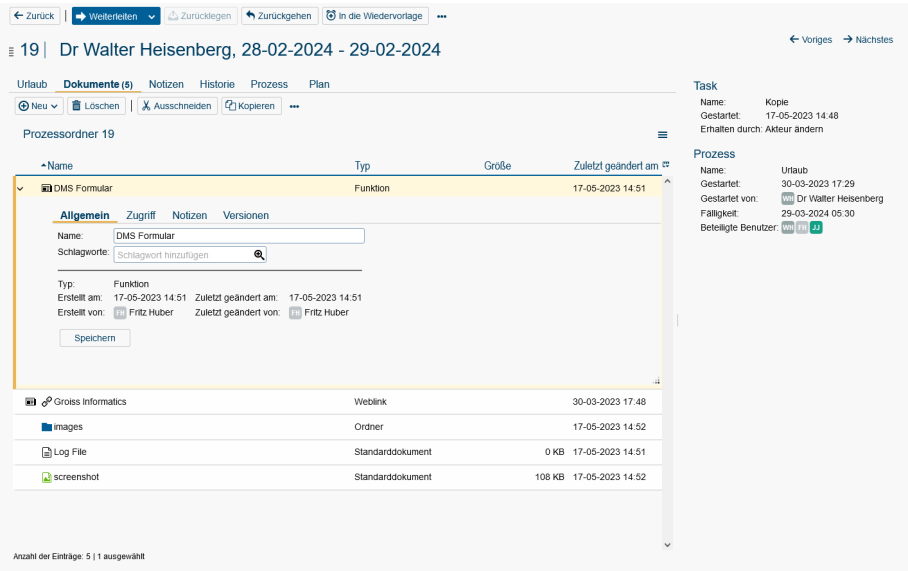


Abbildung 3.9: Dokumente zum Prozess

- ❑ Die Dokumente des Prozesses: Der auf die Formulare folgende Reiter in der Detaildarstellung zeigt die Dokumente des Prozesses, siehe Abb. 3.9. Jede Prozessinstanz repräsentiert einen Ordner, in welchem unstrukturierte Daten zu diesem Prozess abgelegt werden können, z.B. Bilder oder

Textdateien. Die verfügbaren Bearbeitungsfunktionen sind die eines Dokumentenmanagementsystems: Hochladen von Dokumenten, Bearbeiten, Löschen, Versionieren, Kopieren, Einfügen etc.

- ❑ Es ist möglich, Notizen zum Prozess hinzuzufügen, eine Tabelle mit den Notizen wird in einem eigenen Reiter verwaltet.
- ❑ Prozessdaten ändern: Einige Daten der Prozessinstanz sind zur Laufzeit änderbar: die Priorität, das Soll-Fertigstellungsdatum, die Prozessinstanzbeschreibung.

Weitere Funktionen

Applikationsspezifisch können weitere Funktionen ausführbar sein, vergleiche Abschnitt 2.6.

3.3.3 Versionierung und Nachvollziehbarkeit

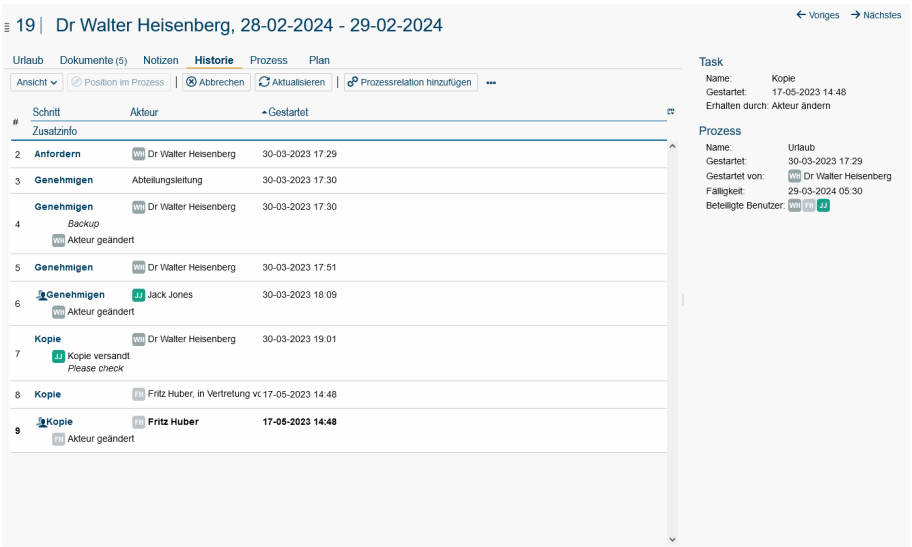


Abbildung 3.10: Prozesshistorie

Die Nachvollziehbarkeit der durchgeführten Aktionen ist eine der Hauptgründe für den Einsatz von BPMS. Zu jeder Prozessinstanz gibt es daher eine

Darstellung des bisherigen Ablaufs, genannt *Historie*, siehe Abb. 3.10. Die wesentlichen Informationen in dieser Ansicht sind:

- Der Prozessablauf: eine tabellarische Darstellung der durchlaufenen Schritte; welcher Akteur hat wann welchen Prozessschritt ausgeführt. Optional ist die Auflistung der nicht interaktiven Schritte, also der If, End, While und sonstigen Prozesselemente.
- Die Daten: Jede Formularänderung wird ebenfalls mitprotokolliert. Bei jedem Schritt können die in diesem Schritt aktuellen Formularversionen angesehen werden. Die Änderungen zur Vorversion werden auf Feldebene genau dargestellt.

Die Prozesshistorie ist sowohl während der Bearbeitung interessant (von wem kommt das, wer hat diese Daten eingetragen?) als auch nach Beendigung des Prozesses für spätere Recherchen.

3.3.4 Suche

Suchfunktionen sind Teil der Kernfunktionalität eines Workflowsystems. Jede Prozessinstanz, die in Bearbeitung ist oder bearbeitet wurde, muss aufgrund verschiedener Kriterien wiedergefunden werden können. Wir unterscheiden drei Suchfunktionen.

Kurzsuche

Es gibt ein Eingabefeld für ein Suchwort, damit kann im Betreff oder nach der Prozess-Id gesucht werden. Das Suchfeld ist in der Navigation immer sichtbar, somit kann diese Suche schnell abgesetzt werden.

Standardsuche

Mehrere Suchfelder stehen zur Verfügung. Gesucht kann z.B. nach einem Zeitraum, dem Prozesstyp, den Beteiligten oder nach Daten im Prozess werden. [2]

[2] Warum benötigen wir eine komplexe Suchmaske, wenn die Suche in Google auch mit einem Feld auskommt? Der Grund ist, dass es bei Prozessinstanzen sehr viele gleichförmige Daten gibt, z.B. 500 Bestellungen des gleichen Artikels im laufenden Jahr. Eine Einschränkung über mehrere Kriterien ist daher nötig. Heuristiken (wie oft wurde der Prozess bereits angeklickt, o.ä.) helfen hier wenig.

Das Ergebnis von Kurzsuche und Standardsuche ist immer eine Liste von Prozessinstanzen, wobei in der Ergebnisliste die folgenden Spalten dargestellt werden:

- Prozess-Id
- Prozesstyp
- bei welchen Tasks und Akteuren befindet sich der Prozess aktuell
- Betreff
- Beginn und - falls vorhanden - Fertigstellungszeit des Prozesses

Erweiterte Suche

Die erweiterte Suche erlaubt es, Suchkriterien aus allen verfügbaren Daten zu verwenden und flexibel zu kombinieren. Die Darstellung des Ergebnisses kann vielfältig beeinflusst werden. Verdichtungen und Aggregationen wie z.B. mehrstufiges Zählen, Mittelwert- oder Summenbildung über mehrere Prozessinstanzen sind möglich und stellen ein wesentliches Hilfsmittel bei der Analyse von Prozessinstanzen dar (vgl. Kapitel 4).

Die Erstellung von solchen Abfragen erfordert aber Wissen über die Datenstrukturen und ist daher nicht für normale Benutzer gedacht. Vielmehr werden Administratoren häufig wiederkehrende Suchen zusammenstellen und abspeichern. Diese als *Reports* bezeichneten Suchmuster können zur Ausführung durch Inhaber von Rollen freigegeben werden. Dadurch können auch normale Benutzer ohne Kenntnis des Datenmodells Suchen effektiv und einfach verwenden. Eine Liste der für den jeweiligen Benutzer freigegebenen Suchen ist unter dem Navigationslink *Gespeicherte Anfragen* verfügbar.

Bei allen Suchen müssen die Berechtigungen beachtet werden, denn nicht jeder darf jeden Prozess finden oder jedes Detail eines Prozesses ansehen, mehr dazu in Kapitel 3.6.

3.3.5 Anpassung der Benutzerschnittstelle

Eine wesentliche Eigenschaft einer Benutzerschnittstelle eines BPMS ist die Anpassbarkeit. Warum ist dies gerade bei BPMS wichtig:

- Darstellung von Prozessdaten: Der Arbeitskorb sollte nicht nur allgemeine Informationen darstellen, sondern für den schnellen Überblick

auch prozessspezifische. Welche Daten angezeigt werden, ist von der konkreten Anwendung abhängig.

- Funktionen abhängig von Rollen und Prozessfortschritt: Die ausführbaren Funktionen sind abhängig von den Rollen des Akteurs und dem Bearbeitungsschritt des Prozesses.

- Abteilungsleiter sehen mehr Reports, Administratoren haben zusätzliche Funktionen, z.B. zur Wartung von applikationsspezifischen Stammdaten.

Wir haben also die Notwendigkeit, die Benutzerschnittstelle abhängig (1) von den Rechten und Rollen des Benutzers anzupassen, (2) abhängig von der Applikation, da unterschiedliche Applikationen unterschiedliche Informationen benötigen. Wie die Adaption von Benutzerschnittstellen in @enterprise erfolgt, sehen wir in Abschnitt 3.9.1.

3.3.6 Mobiles Arbeiten

Bislang wurde ein Web-Interface vorgestellt, das für normale Bildschirmarbeitsplätze geeignet ist. Mit der immer stärkeren Verbreitung von Smartphones (Mobiltelefonen mit Internet-Browser und anderen Zusatzfunktionen) werden auch diese Geräte als Benutzerschnittstellen interessant. Mehrere Faktoren kommen hier zusammen:

- Die Geräte sind in großer Zahl vorhanden, mehrere 100 Millionen Smartphones wurden 2009 weltweit verkauft.

- Schnelle Internetverbindungen sind fast überall verfügbar, sei es durch UMTS oder über WiFi Hotspots.

- Die Anwender nutzen diese Infrastruktur für mobiles Arbeiten. Laut einem Bericht von Optus [**?**] arbeiten 6% der Mitarbeiter ständig von zu Hause aus, 36% regelmäßig außerhalb des Büros (zu Hause oder unterwegs), 39% greifen mit einem mobilen Gerät auf das Firmennetzwerk zu.

Bis vor Kurzem bedeutete die Integration von mobilen Geräten noch, dass nur eine rudimentäre oder eingeschränkte Funktionalität zur Verfügung gestellt wird: Empfangen von Nachrichten, Senden von Statusinformationen oder Informationsabfrage.

Mittlerweile ist die vollständige oder weitgehende Abdeckung des Funktionsumfangs am mobilen Gerät gefordert. Die Herausforderungen dabei sind: Sicherheit, Benutzerfreundlichkeit und Integration der Applikation mit den anderen Funktionen des Smartphones.

Die Sicherheitsaspekte werden in Abschnitt 3.8.7 behandelt, betrachten wir die beiden anderen Themen.

Benutzerschnittstelle am Smartphone

Für die Implementierung eines Smartphone-Clients gibt es zwei Möglichkeiten: Eine Web-Applikation, die den lokalen Browser als Benutzerschnittstelle verwendet oder eine am Smartphone zu installierende Applikation (meist *App* genannt). Beides hat wohl seine Vor- und Nachteile, wie beim normalen Benutzerinterface wollen wir uns hier auf die Betrachtung eines Web-Interfaces beschränken.

Der Hauptaspekt des Interface-Designs ist die Anpassung der Navigation und der Bedienbarkeit an die eingeschränkte Eingabemöglichkeit und die geringere Bildschirmgröße. Abb. 3.11 zeigt die Startseite und den Arbeitskorb in der mobilen Version von @enterprise. Es wird auf Frames, breite Tabellen und Toolbars mit kleinen Icons verzichtet. Die einzige echte Einschränkung ist hier beim Dokumentenmanagement zu finden, wo durch die fehlenden Programme am Client keine Möglichkeit besteht, Dokumente zu bearbeiten.

Integration mit mobilen Funktionen

Der Urahn des Smartphones diente dem Telefonieren, und auch bei den aktuellen Geräten nehmen die Kommunikationsfunktionen eine zentrale Stellung ein: Telefonieren, SMS (Short Message Service), MMS (Multimedia Messaging Service) und E-Mails versenden. In der Bearbeitung eines Geschäftsprozesses ist Kommunikation mit anderen Prozessbeteiligten sicherlich häufig nötig – und eine Integration bei aktuellen Smartphones auch leicht möglich. Aus einer Profil-Seite eines Benutzers können Telefonnummer und E-Mail Adresse vermerkt sein, über spezielle Links gelangt man direkt in die zuständige Anwendung. Ein Benutzerprofil kann z.B. aus der Historie eines Prozesses oder aus einem Formular heraus aufgerufen werden.

Eine weitere auf Smartphones meist vorhandene Funktion ist ein GPS (Global Positioning System) Empfänger, mit dem die aktuelle Position bestimmt werden kann.

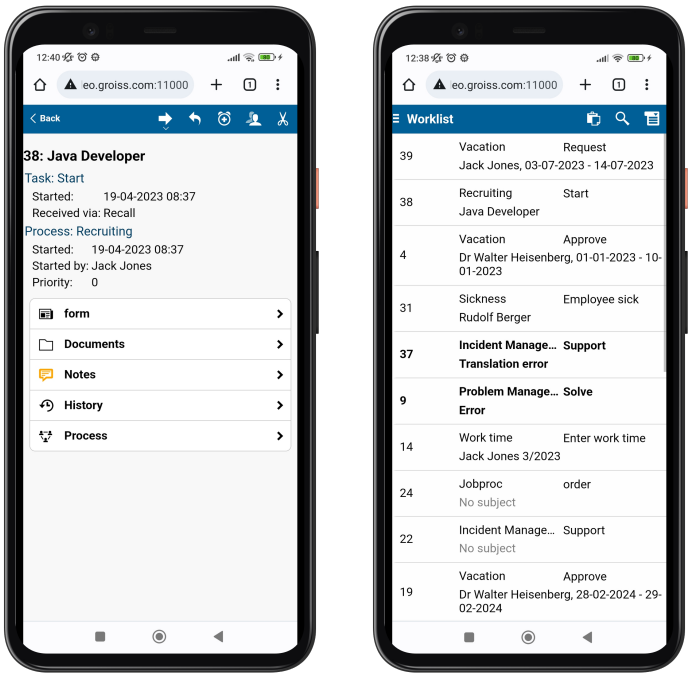


Abbildung 3.11: @enterprise am Smartphone

Abb. 3.12 zeigt ein Anwendungsbeispiel für eine GPS-Integration: Der mobile Außendienstmitarbeiter erhält seinen Auftrag in den Arbeitskorb, am Formular ist die Position vermerkt, mit Klick auf die Schaltfläche *Show on Map* kann diese Position auf der Karte angezeigt werden. Umgekehrt kann bei der Aufnahme einer Störung die aktuelle Position mit Klick auf *Set to Current Loc.* eingegeben werden.

Weitere Integrationen mit den Funktionen des Smartphones sind vorstellbar, z.B. Hinzufügen von Fotos, Videos oder Audioaufnahmen zu Prozessen.

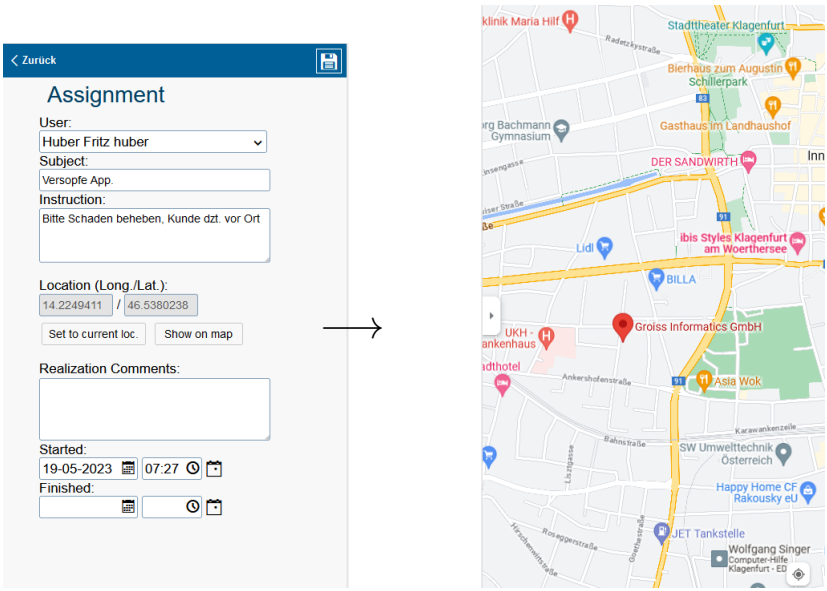


Abbildung 3.12: Formularanzeige und GPS-Integration

3.4 Social BPM

Im vorigen Abschnitt wurde die Integration von Kommunikationswerkzeugen mit der BPM Applikation behandelt. Wenn wir in diese Richtung weiterdenken, kommen wir zu einer allgemeinen Integration von BPM Software mit anderen Kommunikations-Werkzeugen. Das daraus entstehende Gesamtsystem bezeichnet man als Social BPMS [**?**].

Soziale Software ermöglicht die Kommunikation zwischen Personen, der Begriff fasst die unterschiedlichen Systeme, wie Blogs, Wikis, Chats, Verwaltung von Freundschaftsgruppen etc. zusammen. Die wesentlichen Aspekte sind Informationen (über die anderen Teilnehmer) und Kommunikation mit diesen.

Betrachten wir zuerst den Informationsaspekt im Fokus des Geschäftsprozessmanagements. Die Zusammenarbeit in Prozessen formt Gruppen von Personen, die durch diese Prozesse miteinander kommunizieren. Weitere Gruppen ergeben sich durch die Zuordnung zu Abteilungen oder Arbeitsgruppen (über Rollenzuordnungen). Informationen, die im BPM über diese Personen zur Verfügung gestellt werden, können sein:

- Benutzerprofil: Über den Organisations-Browser oder die Prozesshistorie gelangt man zu einem Benutzer-Profil, das die wesentlichen Daten über diesen Benutzer anzeigt: Kontaktdaten, woran arbeitet er gerade, wo befindet er sich.
- Aufenthaltsort auf Karte: Eine andere Ansicht der Aufenthaltsorte kann über eine Karte angeboten werden, gerade bei mobilen Arbeiten kann diese Information nützlich sein.
- Anwesenheitsliste: In die Navigation kann eine Anwesenheitsliste eingeblendet werden: Wer ist gerade online?

Der eigentliche Zweck der Informationsbereitstellung aus obiger Aufzählung ist die Ermöglichung der Kommunikation unter den Teilnehmern. Folgende Mechanismen können angeboten werden:

- Audio: Telefon oder Audiokommunikation über Internet (Skype)
- Chat: Versenden von Kurznachrichten
- Wiki: Erstellen von Texten, die von anderen bearbeitet und erweitert werden können.
- Blog: Texte, die von Benutzern zu einem Thema erstellt werden, im Gegensatz zu Wikis mit sequentiellem Charakter. Blogs werden normalerweise später vom Autor oder anderen Benutzern nicht mehr geändert, wohl aber mit Kommentaren versehen.
- Tagging: In @enterprise ist das Beschlagworten von Dokumenten möglich. Tagging ist eine Erweiterung dieses Konzepts, wo alle Benutzer Inhalte beschlagworten können (auch „Social Tagging“ genannt). Inhalte werden nun nach Suche über diese Schlagworte auffindbar. Sogenannte Tag Clouds dienen zur Visualisierung der Schlagworthäufigkeiten.

Nicht jede dieser Kommunikationsformen muss im BPMS direkt integriert werden. Wichtig ist, dass sie aus dem System heraus „mit einem Klick“ angestoßen werden können. Andere Kommunikationsformen, wie z.B. Wikis, Blogs und Tags lassen sich gut in das Dokumentenmanagementsystem des BPMS integrieren.

Die Kommunikation kann sowohl für gemeinsames Problemlösen während der Prozessbearbeitung als auch während der Prozessdefinition verwendet werden. Vor allem bei mobilem Arbeiten ermöglicht es einfachere Zusammenarbeit und kann Nachteile der Telearbeit kompensieren [**?**].

3.5 Abweichungen vom vorgegebenen Prozessweg

Die Grundidee von Workflow ist, dass vordefinierte Prozesse gleichartig abgearbeitet werden, d.h. dass eine Menge von Geschäftsfällen einem vordefinierten Ablauf oder vordefinierten Regeln folgt. Wo es Regeln gibt, gibt es in der Praxis auch Ausnahmen. Es ist nicht möglich bzw. sinnvoll, alle Ausnahmen vorherzusehen und zu modellieren. Das BPMS muss daher mit diesen Ausnahmen umgehen können. Wir werden im Folgenden eine Reihe von typischen Ausnahmefällen vorstellen, in Abb. 3.13 sind sie zusammenfassend dargestellt.

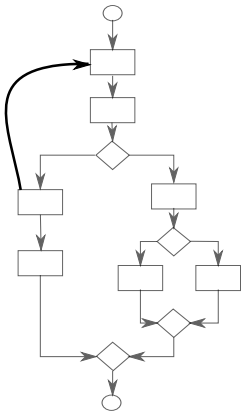
3.5.1 Zurückgehen

Eine gängige Änderung des Prozessweges ist das Zurückgehen (Abb.3.13 a). Der Bearbeiter eines Schrittes sieht sich aufgrund bestimmter Umstände nicht in der Lage, den Schritt weiter zu bearbeiten sondern möchte den Prozess zu einem früheren Schritt bzw. Bearbeiter zurückschicken. Dies kann z.B. geschehen, um Ergänzungen bei den Daten vornehmen zu lassen. Beispiel: Ein Antrag wird vom Genehmiger zurück an den Antragsteller geschickt, weil die Begründung nicht ausreichend ist.

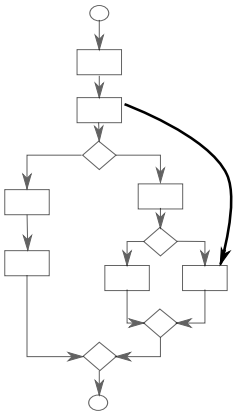
Da während des normalen Weiterleitens am Ende jedes Tasks eine Postcondition ausgewertet werden kann, die im Prinzip eine beliebig komplexe Aktion automatisch ausführen kann, können durch das Zurückgehen unerwünschte Seiteneffekte auftreten. Es müssen daher für solche Schritte Kompensationsaktionen definiert werden, um sicherzustellen, dass der Prozesszustand konsistent bleibt. Wird mehr als einen Schritt zurückgegangen, muss gegebenenfalls eine ganze Kette von Aktionen rückgängig gemacht werden.

Auch bei Verwendung von Parallelitäten gibt es beim Zurückgehen einiges zu beachten: Geht man aus einem Schritt, der sich innerhalb der Parallelität befindet, zu einem Schritt, der sich vor der Parallelität befindet zurück, müssen die übrigen Zweige der Parallelität abgebrochen werden. Da das Abbrechen von Schritten, d.h. Entfernen aus dem Arbeitskorb, eine privilegierte Operation ist, ist hier ein eigenes Recht (Schritt abbrechen) nötig, um im Falle von noch aktiven parallelen Zweigen Zurückgehen zu ermöglichen.

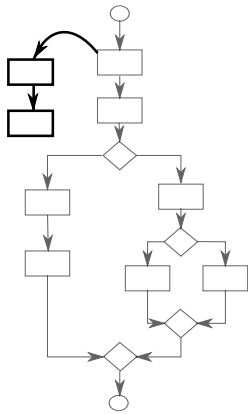
Auch das Zurückgehen in eine bereits abgeschlossene Parallelität ist problematisch. Es stellt sich die Frage nach der Behandlung der anderen parallelen Zweige. Das Hineinspringen in eine Parallelität wird daher in @enterprise nicht unterstützt.



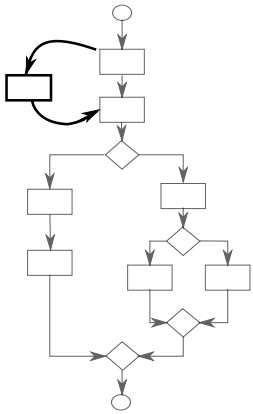
a) Zurückgehen



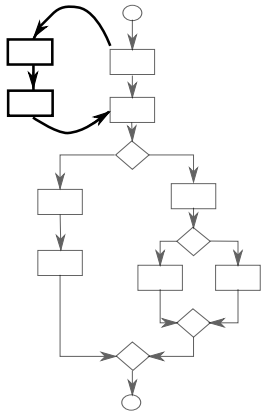
b) Nach vor Gehen



c) Kopie an...



d) Akteur ändern



e) Einfügen von Schritten

Abbildung 3.13: Verschiedene Abweichungen vom Prozessweg

Das Zurückgehen kann natürlich auch als Schleife modelliert werden. Häufig tritt das folgende Muster auf: In einer Sequenz wird im ersten Schritt eine Tätigkeit ausgeführt, im zweiten dann das Ergebnis geprüft und beim Fehlschlagen der Prüfung neuerlich der erste Schritt ausgeführt, bis das Ergebnis passt.

Ein Modellieren als Schleife macht das explizit und ermöglicht es, den Korrekturschritt anders zu behandeln als den initialen Schritt - z.B. mit einer anderen Frist zu versehen. Bei größeren Prozessen können jedoch nicht alle Möglichkeiten zum Zurückgehen ausmodelliert werden - der Prozess würde unnötig komplex und unübersichtlich werden. Es empfiehlt sich die explizite Modellierung also nur, wenn der Korrekturschritt besonders behandelt werden soll bzw. einen integralen Prozessbestandteil darstellt.

3.5.2 Nach vor gehen

Das Springen nach vor (Abb.3.13 b) ist mit noch mehr Problemen verbunden als das Zurückgehen: Man müsste jeweils definieren, welche Schritte übersprungen werden dürfen und welche nicht. In @enterprise wird diese Funktion deshalb nicht unterstützt.

Zu beachten ist, dass mit Zurückgehen im vorigen Abschnitt immer ein Zurückgehen in der Prozessstruktur gemeint ist und nicht in der Prozesshistorie. Ein Zurückgehen in der Historie zu einem früheren Schritt kann nämlich ein „Nach vor gehen“ in der Prozessstruktur sein, wenn der Prozess schon einmal weiter war (z.B. im Falle von Schleifen und nach einem vorher erfolgten Zurückgehen).

3.5.3 Kopie an...

Das Versenden einer Kopie der Prozessdaten (Abb.3.13 c) ist ebenfalls eine Ad-hoc Änderung des Prozessablaufs. Eine anderer Akteur erhält in seinem Arbeitskorb eine read-only Ansicht der Prozessinstanz. Diese Aktion ist immer möglich.

3.5.4 Akteur ändern

Wenn ein Schritt zum Bearbeiter A gelangt und dieser sich aber nicht zuständig fühlt oder nicht in der Lage ist, die Aktivität zu bearbeiten, möchte er sie an einen anderen Bearbeiter abgeben. Wenn dieser neue Bearbeiter dann weiterleitet, folgt der Prozess weiter dem vordefinierten Ablauf (Abb.3.13 d).

Dieses Vorgehen wird häufig genutzt, allerdings ist zu beachten, dass in manchen Prozessen eine Einschänkung besteht, wer einen bestimmten Schritt ausführen darf. Zum Bsp. darf den Antrag nur der Leiter einer Abteilung genehmigen. Hier käme es durch das Ändern des Akteurs zu einer Verletzung der Prozessbedingungen.

In @enterprise kann man diese Funktion für bestimmte Tasks daher verbieten (einstellbar bei der Taskdefinition).

Es sei noch angemerkt, dass man die Umsetzung der Funktion auch durch Modellierung erreicht. Wenn an einer bestimmten Stelle im Prozess eine Akteursänderung zu erwarten ist, kann man eine Schleife einbauen, wo der Akteur des Tasks in jedem Schleifendurchlauf neu aus dem Formular gelesen wird. Beendet wird die Schleife dann z.B. durch die Abfrage auf ein weiteres Formularfeld (vgl. Abb. 2.28).

3.5.5 Einfügen von Schritten

Eine weitere Änderung des Ablaufs ist das Einfügen von zusätzlichen Schritten (Abb.3.13 e). Die Schritte werden nach dem aktuellen und vor dem nächsten Schritt eingefügt, sozusagen als *Umweg*. Um die Integrität des Prozesses zu erhalten, müssen zwei Punkte beachtet werden:

Zulässigkeit von Tasks

Es dürfen nur solche Tasks eingefügt werden, die dem Prozess zugeordnet sind, d.h. entweder im Ablauf vorkommen oder explizit bei der Prozessdefinition als erlaubte, zusätzliche AdHoc-Tasks angegeben sind. Bei allen diesen Tasks sind die Akteure und die Fomularsichtbarkeiten definiert. Ein Problem könnte allerdings sein, dass die Reihenfolge, in der Tasks eingefügt werden, sowie deren Position zur Laufzeit definiert werden: Ein Bearbeitungsschritt kann z.B. nach einer erfolgten Genehmigung wieder eingefügt werden - bei einem Prozesszustand, wo eine nachträgliche Bearbeitung eigentlich nicht mehr möglich sein sollte.

@enterprise bietet dazu die Möglichkeit im Preprocessing solche Bedingungen abzuprüfen. Das Ad-hoc Einfügen von Schritten kann entweder in wenig strukturierten, unkritischen Prozessen verwendet werden oder muss auf Tasks eingeschränkt werden, die - auch was ihre Formularberechtigungen angeht - überall im Prozess zulässig sind.

	PROJEKT	PROZESS
Ablaufbeschreibung	Projektplan: individuell erstellt nach allgemeinen Prinzipien	Prozessdefinition: allgemeines Schema, individuelle Anpassungen
Kontrolle	Soll-Ist Vergleich gewartet vom Projektmanager	Daten aus Historie und Formularen
Weitergabe	manuell, informelle Kommunikation	durch Engine

Abbildung 3.14: Projekt- vs. Prozessmanagement

Zulässigkeit von Kontrollstrukturen

Es ist zu beachten, dass das Einfügen von Schritten vom Endbenutzer durchgeführt wird. Diesem ist es in der Regel nicht möglich bzw. zumutbar, Bedingungen zu formulieren. Somit sind Schleifen und bedingte Verzweigungen aus der Ad-hoc Modellierung auszuschließen.

Die @enterprise Funktion „Senden an“, die das Einfügen von Schritten ermöglicht, unterstützt Sequenzen und Parallelitäten.

3.5.6 Prozessdefinition zur Laufzeit

Ausgehend von einer minimalen Prozessdefinition, der aus einem einzelnen Task besteht, können wir zur Laufzeit mit den beschriebenen Mechanismen fast beliebige Prozesse erstellen. Dies ist dann nötig, wenn die einzelnen Prozessinstanzen so unterschiedlich sind, dass der Ablauf jedesmal anders aussieht. Wir haben also keine vorgegebene Prozessdefinition sondern machen beim Prozessstart einen instanzspezifischen *Plan*. Solche Abläufe nennt man auch Projekte und die Systeme zum Management dieser bezeichnet man als Projektmanagementsysteme (im Gegensatz zu Prozessmanagementsystemen).

Betrachten wir die Unterschiede zwischen Projekt- und Prozessmanagement in Abb. 3.14.

Mithilfe eines Ad-Hoc Prozesses und der Möglichkeit des Einfügens von Schritten zur Laufzeit können mit einem BPMS auch Projekte durchgeführt und kontrolliert werden, unter Beibehaltung der Vorteile bei Fortschrittskontrolle und Weitergabe. Abb. 3.15 zeigt einen solchen allgemeinen Projektprozess.

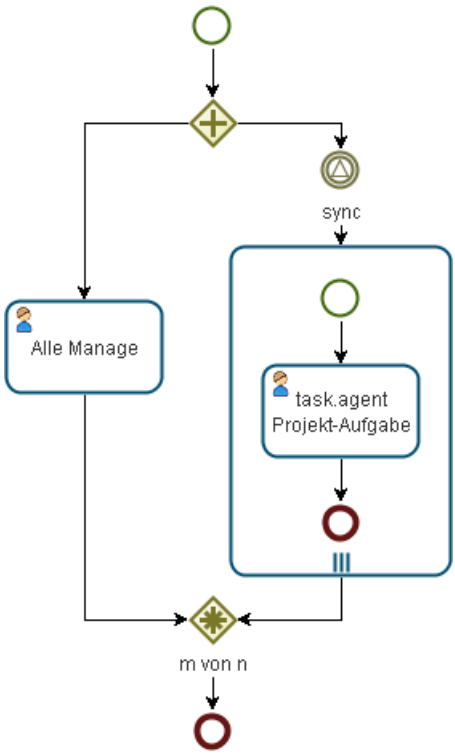


Abbildung 3.15: Projekt-Prozess

Der *Manage* Task ist parallel zur Ausführung der einzelnen Projektaufgaben. Der Projektmanager trägt sukzessive die Aufgaben ein, ordnet sie Mitarbeitern zu und startet sie mit einer eigenen Funktion. Die auf diese Weise eingetragenen Aufgaben werden parallel ausgeführt, an jede Aufgabe können aber gegebenenfalls eine Sequenz weiterer Aufgaben angehängt werden. So ist es möglich, die Bearbeitung komplexerer Projekte adäquat und flexibel zu unterstützen.

Der beschriebene Prozess ist in der @enterprise Demo-Applikation Projektmanagement enthalten, welche von der Homepage der Groiss Informatics heruntergeladen werden kann.

3.6 Berechtigungen

Eine wesentliche Komponente eines BPMS ist das Berechtigungssystem, das ist jener Systembestandteil, der prüft, welcher Benutzer welchen Zugriff auf welche Daten erlangen darf. Bedenkt man, dass bei einem umfassenden Einsatz eines BPMS vielfältigste Daten im System sind, wird es klar, dass es Beschränkungen für den Zugriff geben muss. Beispiele:

- Workflows, die mit Gehältern, Pfändung, Kündigung etc. zu tun haben.
- Suchergebnisse, vor allem verdichtete, die Aufschluß über Bearbeitungszeiten geben.

Der Umgang damit, welche Daten eingesehen werden können, ist in den einzelnen Ländern und Firmen sehr unterschiedlich. Was in den USA oder der Schweiz gängige Praxis ist, kann in Deutschland oder Österreich durch das Datenschutzgesetz verboten sein.

Wir wollen in diesem Abschnitt schrittweise die Elemente des Berechtigungssystems darstellen, beginnend mit den einfachen Konzepten bis hin zu den komplexeren.

Grundsätzlich erfolgt die Berechtigungsvergabe über Rollen, man bezeichnet dies als *Role Based Access Control* [**?**]. Leider kommen wir damit nicht aus und brauchen zusätzliche Regeln, die die Berechtigungen genauer definieren, solche Systeme bezeichent man dann als *Attribute Based Access Control* (ABAC) [**?**].

Ein wesentliches Unterscheidungsmerkmal bei der Ermittlung der Berechtigungen in einem BMPS ist die Trennung zwischen prozessbezogenen und prozessunabhängigen Rechten. Dadurch, dass ein Benutzer aktueller oder gewesener Akteur eines Prozesses ist, bekommt er bereits bestimmte Rechte. Wir betrachten zuerst die prozessunabhängigen Rechte.

3.6.1 Prozessunabhängige Rechte

Die Verwaltung von Berechtigungen basiert auf der folgenden Beziehung: Ein Akteur erhält ein Recht auf ein Objekt, siehe Abb. 3.16.

- Als Akteur kann ein Benutzer oder eine Rolle oder eine Rolle in einer Organisationseinheit angegeben werden. Die Berechtigung gilt in diesem Fall für alle Benutzer, die diese Rolle haben (bzw. diese Rolle in der entsprechenden OE haben).

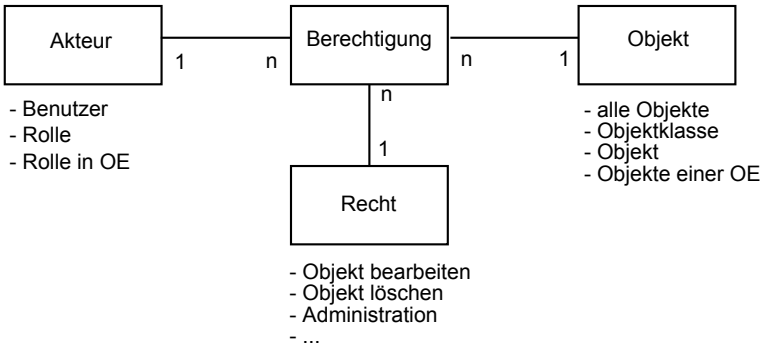


Abbildung 3.16: Berechtigung

- Das Recht ist eines aus der Liste der vordefinierten Rechte oder ein applikationsdefiniertes. Vordefinierte Rechte sind u.a. Objekt erzeugen, bearbeiten, ansehen, ausführen.

- Als Objekt kann ein konkretes Objekt angegeben werden, dann gilt die Berechtigung für dieses Objekt. Oder es wird eine Objektklasse angegeben, dann gilt die Berechtigung für alle Objekte dieser Klasse. Als dritte Möglichkeit wird kein Objekt angegeben, dann gilt die Berechtigung für alle Objekte unabhängig von deren Klasse.

  Weiters gibt es den Gültigkeitsbereich „Objekte der Org.einheit“ in zwei Varianten: die Berechtigung gilt entweder für die angegebene Organisationseinheit oder die Organisationseinheit in der der Benutzer die bei Akteur angegebene Rolle innehat.

Beispiele für Berechtigungen sind:

- Die Rolle Sys darf alle Funktionen ausführen (Recht *ausführen* auf alle Objekte der Klasse Funktion).

- Die Rolle Leiter der Org.einheit Senat darf alle Akten ansehen.

- Sachbearbeiter haben das Recht Akten einzusehen, die in ihrer Abteilung (= Abteilung, in der sie Sachbearbeiter sind) angelegt wurden.

Negative Berechtigungen: Um Ausnahmen zu definieren, gibt es negative Berechtigungen, d.h. eine Berechtigung wie oben definiert mit dem zusätzlichen

Kennzeichen „nicht erlaubt“. Damit kann man Ausnahmen definieren, z.B.: Hr. Maier darf alle Akten der Abt. X einsehen, außer den Akt 4711.

Regeln: Um nicht bei jedem einzelnen Objekt angeben zu müssen, wer welches Recht darauf hat, definieren wir objektklassenspezifische Regeln. Beispiel: Die Rollenzuordnungen eines Benutzers darf derjenige bearbeiten, der auch den Benutzer (zu dem die Rollenzuordnung gehört) bearbeiten darf. Das heisst, in diesem Fall wird aus der ursprünglichen Frage „Darf der Benutzer X die Rollenzuordnung Y bearbeiten “ die Frage „Darf der Benutzer X den Benutzer Z bearbeiten“. Diese Regeln werden in @enterprise in Java definiert.

Eine häufige Regel ist die sogenannte „Owner-Regel“: Bei einem Objekt, z.B. einem Dokument, ist der Ersteller (Owner) vermerkt. Die Regel besagt, dass der Owner automatisch die Rechte *Bearbeiten*, *Ansehen* und *Löschen* für dieses Objekt erhält.

Berechtigungslisten: Für die komfortablere Wartung von Berechtigungen dienen die zwei folgenden Erweiterungen:

1. Einzelberechtigungen können zu *Berechtigungslisten* zusammengefasst werden. Wenn z.B. eine Menge von Objekten das gleiche Zugriffsprofil hat, muss dies nur einmal definiert werden. Die Berechtigungsliste kann dann für jedes dieser Objekte verwendet werden.

2. Für jede Objektklasse kann eine Berechtigungsliste als Standard-Berechtigungsliste definiert werden, die für alle neuen Objekte dieser Klasse automatisch verwendet wird.

Zur Überprüfung, ob ein Benutzer ein bestimmtes Recht hat, gibt es in @enterprise die Administrationsfunktion „Berechtigungen überprüfen“, siehe Abb. 3.17. Diese gibt Auskunft, ob und warum ein Benutzer ein bestimmtes Recht auf ein Objekt hat.

3.6.2 Prozessbezogene Berechtigungen

Betrachten wir zunächst die verschiedenen Aktionen, die mit Prozessinstanzen und ihren Daten möglich sind:

1. Prozessinstanz finden, Historie ansehen
2. die aktuellen Formulardaten ansehen
3. die Dokumente und deren Inhalte ansehen

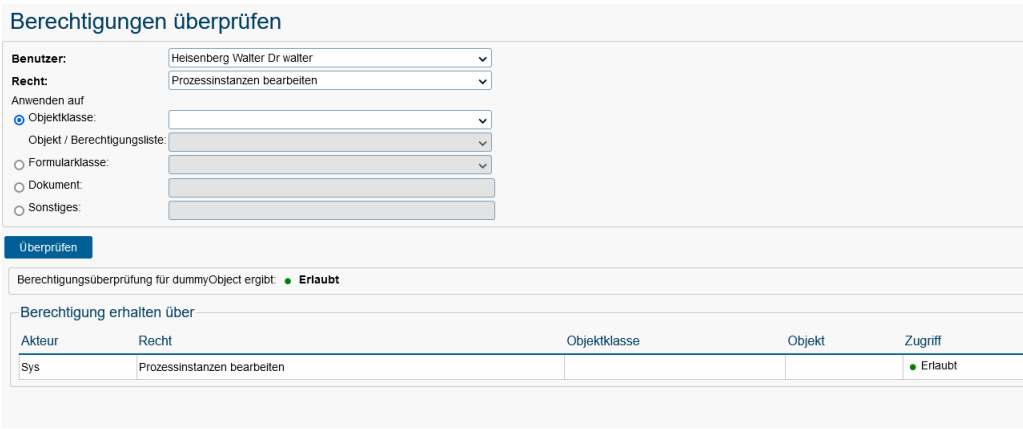


Abbildung 3.17: Berechtigungsüberprüfung

4. Formulare bearbeiten
5. Dokumente bearbeiten, hinzufügen und löschen
6. Standardfunktionen auf den Arbeitskorbeintrag: Zurücklegen, in die Wiedervorlage, Weiterleiten, Zurückgehen, Arbeitskorbeintrag in Benutzerordner organisieren
7. den Akteur eines Prozessschritts ändern
8. applikationsspezifische Zusatzfunktionen zu Arbeitskorbeinträgen ausführen
9. den Prozess abbrechen und reaktivieren
10. den Prozess archivieren (aus der Datenbank entfernen).

Manche dieser Aktionen sind für den aktuellen Akteur nötig, um den Prozess zu bearbeiten. Diese Aktionen sollen, ohne zusätzliche Berechtigungen vergeben zu müssen, möglich sein. Das Berechtigungssystem enthält also eine Regel, die dem Akteur die Aktionen 1 bis 7 aus obiger Liste erlaubt.

Für Funktionen der Applikation (Punkt 8) müssen die Benutzer eigens berechtigt werden. Damit kann man Funktionen an bestimmte Akteure freigeben, an andere nicht.

Eine weitere Regel des Berechtigungssystems existiert für Prozessbeteiligte, die nicht aktueller Akteur sind (also den Prozess früher bearbeitet haben). Diese Benutzer erhalten das Recht, den Prozess über die Suche zu finden und die Prozesshistorie anzusehen (Punkt 1).

Weitere Aktionen sind für nicht am Prozess beteiligte Benutzer sinnvoll: Prozesse finden (1), die Details ansehen (2,3), den Akteur ändern (7) und den Prozess abbrechen, reaktivieren und archivieren (8,9,10). Dafür müssen eigene Berechtigungen vergeben werden. @enterprise hat deshalb die folgenden Rechte für Prozessinstanzen definiert:

- ❑ Prozessinstanzen ansehen
- ❑ Akteur ändern
- ❑ Prozessinstanzen bearbeiten: abbrechen, reaktivieren und Akteur ändern
- ❑ Konfiguration: dieses Administratorrecht erlaubt es, Prozesse zu archivieren

Es ist nur in Ausnahmefällen sinnvoll, Berechtigungen direkt für bestimmte Prozessinstanzen zu vergeben. Es bieten sich zwei Möglichkeiten der Gruppierung an: Die Berechtigung wird auf alle Instanzen einer bestimmten Prozessdefinition vergeben oder auf alle Instanzen, die einer bestimmten Organisationeinheit zugeordnet sind. Die obigen vier Rechte können deshalb für Organisationseinheiten und Prozessdefinitionen vergeben werden.

Eine entsprechende Regel im Berechtigungssystem übersetzt daher die Anfrage „Hat der Benutzer X das Recht Akteur ändern auf die Prozessinstanz Y“ auf die Anfrage „Hat der Benutzer das Recht Akteur ändern oder Prozessinstanzen bearbeiten für die Organisationseinheit oder die Prozessdefinition der Prozessinstanz Y“.

3.7 Vertretungen

Die Definition von Vertretungen ist nötig, um Abwesenheiten von Mitarbeitern zu modellieren. Für die Zeit der Abwesenheit einer Person A führt eine Person B die Arbeiten von A durch. B ist somit Vertreter von A. Um die Aktionen im BPMS ausführen zu können, muss B während der Vertretungsperiode alle oder einen Teil der Rechte und Rollen von A erhalten.

Weiters ist zu berücksichtigen, dass eine Person unterschiedliche Aufgaben zu erfüllen hat, die eventuell unterschiedlichen Vertretern übergeben werden sollen. Zum Beispiel kann ein Abteilungsleiter in manchen Agenden von seinem Assistenten vertreten werden, in anderen nur von seinem eigenen Vorgesetzten oder einer anderen Person derselben Hierarchiestufe. Die Vertretungen müssen daher pro Rolle definierbar sein. Wir bezeichnen dies als *Rollenvertretung*.

Ein Vertreter muss außerdem die direkt adressierten Aufgaben bearbeiten können, die nicht einer Rolle zuordenbar sind. Wir bezeichnen dies als *persönliche Vertretung*. Ein persönlicher Vertreter kann optional alle Rollen mitvertreten, damit ist es nicht nötig, den oder die Vertreter pro Rolle einzutragen.

Für unterschiedliche Zeiträume muss es möglich sein, unterschiedliche Vertreter einzutragen, um z.B. für eine zweiwöchige Abwesenheit in der ersten Woche Vertreter B1 und in der zweiten Woche Vertreter B2 eintragen zu können.

Zusammengefasst ergibt dies folgende Möglichkeiten:

- mehrere persönliche Vertreter, jeweils für einen Zeitraum mit und ohne Übernahme der Rollen;
- mehrere Vertreter pro Rollenzuordnung, jeweils mit Angabe des Zeitraums.

3.8 Schnittstellen und Applikationsintegration

Die Bearbeitung von Geschäftsprozessen ist immer mit der Manipulation von Daten verbunden. Diese Daten sind mit anderen Daten des Unternehmens (oder der Organisation) verknüpft. Da das BPMS nicht die einzige datenverarbeitende Anwendung ist, ist ein Austausch dieser Daten mit anderen Anwendungen nötig.

Nehmen wir als Beispiel den Prozess einer Urlaubsgenehmigung. Der Prozess ist meist recht einfach und manipuliert ein einziges, ebenfalls nicht umfangreiches Formular.

Ist bei solchen simplen Prozessen eine Integration überhaupt nötig? Typischerweise ja, und zwar in nicht geringem Ausmaß:

- Der Prozess wird durch ein Web-Formular gestartet, integriert in eine interne Homepage oder ein Portal, um dem Benutzer zu ermöglichen, di-

rekt ohne explizites Einloggen bzw. mehrstufige Navigation den Prozess zu starten.

- ❑ Wenn der Urlaub genehmigt wurde, sollten die Urlaubstage im Personalsystem automatisch abgebucht werden.
- ❑ Direkt Betroffene, wie z.B. der Vorgesetzte oder der Vertreter, sollten eine Informations-Mail erhalten.
- ❑ Aus den einzelnen Urlaubseinträgen kann man eine Abwesenheitsliste für die Abteilung generieren (eventuell unter Hinzunahme von anderen Abwesenheiten, wie z.B. Krankheit oder Dienstreise).
- ❑ Für die einzelnen Mitarbeiter sollte eine Urlaubstage-Übersicht verfügbar sein, um die Anzahl der noch offenen Urlaubstage sehen zu können.
- ❑ Der Urlaub sollte automatisch im Kalender eingetragen werden.
- ❑ Als Eingabehilfe sollten die Urlaubstage (Werktage) automatisch berechnet werden.

An diesem einfachen Beispiel sieht man bereits den Umfang von Integrationen, die nötig sind, um einen Prozess komfortabel handhaben zu können. Jedes BPM-Projekt ist auch ein Softwareentwicklungs- und Integrationsprojekt.

Um die verschiedenen Integrationen zu klassifizieren, betrachte man Abb. 3.18. Dort steht im Zentrum der im BPMS zu bearbeitende Geschäftsprozess, umgeben von den Integrationsfunktionen und -komponenten.

1. Altdatenübernahme: Manchmal ist es nötig, die Instanzdaten aus einem Vorgängersystemen zu übernehmen.
2. Das Starten eines Prozesses erfolgt oft aus einem Fremdsystem: Zum Beispiel kann eine per E-Mail eingehende Supportanfrage oder eine Störungsmeldung aus einem Alarmsystem einen automatischen Prozessstart bewirken.
3. Autorisierung: Das Login ins Workflowsystem erfolgt im einfachsten Fall mit im System verwalteten User-Ids und Passwörtern. Komfortabler ist es, wenn dass Passwort einer zentralen Passwortverwaltung entnommen wird. Und noch einfacher, wenn der Browser den Benutzer schon kennt und weder Benutzername noch Passwort erforderlich ist (= Single Sign On, SSO).

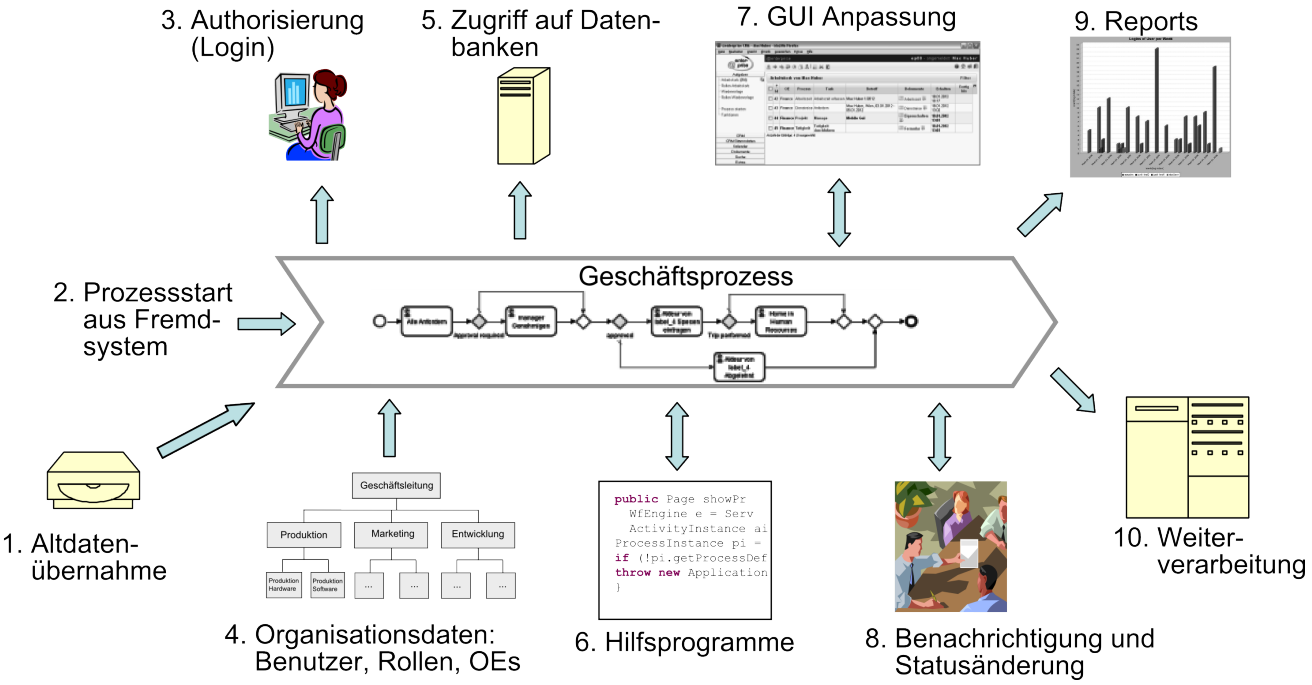


Abbildung 3.18: Applikationsintegration

4. Organisationsdaten: Die Verwaltung von Benutzern, Rollen, Organisationseinheiten und Rollenzuordnungen kann ebenfalls außerhalb des BPMS erfolgen.

5. Während der Prozessbearbeitung muss oft auf Daten aus anderen Systemen zugegriffen werden (z.B. Inventardaten) und kaum etwas konterkariert das Bemühen um Prozessoptimierung mehr, als wenn es nötig ist, Daten aus einem System manuell in ein anderes zu übertragen.

6. Aufruf von Hilfsprogrammen: Bei der Bearbeitung von Daten werden einerseits Formulare direkt im System (Browser) bearbeitet. In diesem Fall sind Eingabehilfen, wie das Ausführen von Berechnungen oder kontextsensitive Auswahl, nötig. Andererseits werden Daten in externen Applikationen bearbeitet. Eine Integration muss das direkte Aufrufen dieser Applikationen aus dem BPMS ebenso gewährleisten, wie die Abspeicherung der Ergebnisdaten im System.

7. Anpassung der Benutzerschnittstelle: Zurverfügungstellung genau der Informationen und Funktionen, die der Benutzer für die Prozessbearbeitung braucht.

8. Benachrichtigung: Benutzer, die nicht ständig mit dem BPMS arbeiten, wollen über neue Einträge unterrichtet werden. Im Normalfall erfolgt diese Benachrichtigung über E-Mail, aber SMS und andere Kommunikationsformen sind ebenfalls gängig.

9. Erstellen von Reports: Für Reports kann die im BPMS integrierte Reportingkomponente verwendet werden, es kann aber auch nötig sein, die Instanzdaten in ein anderes System zur Reporterstellung zu exportieren.

10. Weiterverarbeiten der Ergebnisse: Die im Prozesslauf entstehenden Instanzdaten werden an andere Systeme weiterübermittelt, z.B. Bescheiderstellung am Ende eines Aktenlaufs in einer Behörde.

Im folgenden werden wir die einzelnen Integrationsaspekte behandeln. Der obigen Gliederung wird dabei nicht gefolgt, sondern eine Gliederung nach der Art der technischen Umsetzung gewählt.

3.8.1 Organisationsdaten

Der maßgebliche Standard für die Verwaltung von Organisationsdaten ist LDAP (Lightweight Directory Access Protocol) [**?**]. Von Microsoft gibt es darauf

aufbauend das Active Directory Protokoll.

Es ist daher naheliegend, die Benutzer und die Organisationsstruktur von einem solchen Verzeichnis zu übernehmen. In @enterprise gibt es dazu eine erweiterbare Schnittstelle: Ein LDAP Verzeichnis mit Zugangsdaten, Suchpfad etc. kann definiert werden. Das konkrete Mapping zwischen den LDAP Daten und den Stammdaten von @enterprise führt ein API Programm durch, welches leicht angepasst werden kann. Eine Implementierung für den Import von Benutzern aus dem Microsoft Active Directory ist vorhanden.

Man beachte allerdings, dass die Informationen in solchen Verzeichnissen meist nicht ausreichen und man entweder im BPMS die fehlenden Informationen, z.B. Rollenzuordnungen und Berechtigungen, nachträgt oder das Verzeichnis um diese Informationen erweitert.

3.8.2 Autorisierung

Die Autorisierung in @enterprise erfolgt normalerweise mit Benutzername und Passwort. Dabei wird das Passwort unverschlüsselt über das Netzwerk übertragen. Die Methode der Autorisierung kann allerdings angepasst werden. Die Authorisierungsklasse `SSLAuth` verschickt das Passwort verschlüsselt über das Netz, dazu ist die Konfiguration von SSL nötig.

Passwörter sind allerdings - auch wenn sie verschlüsselt übertragen werden - potentiell unsicher. Sie können erraten werden, werden auf Zettel notiert (die dann am Bildschirmrand kleben) oder leichtfertig weitergegeben.

Eine Möglichkeit ohne - eigenes - @enterprise Passwort auszukommen, ist single-sign-on. Dazu muss man in @enterprise das Zusatzpackage SSO for Windows installieren.

Will man komplett auf Passwörter verzichten, sind Chipkarten die Alternative. Die Autorisierung muss dann an das API der Chipkartensoftware angepasst werden. Wenn die Chipkartensoftware ein Client-Zertifikat am Browser installiert, kann die Authorisierungsklasse ohne clientseitigen Code auskommen. Ein Beispiel hierfür ist die Demo Klasse `ClientCertDemoAuth`.

Für jede single-sign-on Lösung ist immer eine Übernahme der Basis-Benutzerdaten aus einem anderen System nötig.

3.8.3 Services

Wie eingangs erwähnt ist eine der wesentlichen Aufgaben eines BPMS die Integration verschiedener Applikationen. Das BPMS dient dabei dazu, die Aufrufe der einzelnen Systeme zu koordinieren („orchestrieren“).

Dieses Konzept des Zusammensetzens eines Geschäftsprozesses aus einzelnen Teilapplikationen, den *Services*, nennt man auch *Serviceorientierte Architektur* (SOA).

Bei der Implementierung von Services ist ein Hauptziel die Wiederverwendbarkeit, der Aufruf soll möglichst plattformunabhängig erfolgen. Web-Services haben sich als Standardschnittstelle für Services durchgesetzt. Dabei werden über ein Internet-Transportprotokoll, meist HTTP, die Daten in Form von XML Dokumenten übertragen.

In einer serviceorientierten Architektur kann das BPM-System selbst als Service angesehen werden. Geschäftsprozesse können somit von anderen Applikationen angestoßen werden, z.B. ein Bestellprozess aus einem Web-Portal. Ebenso können Applikationen Daten aus dem BPM-System abfragen.

Die zur Verfügung stehenden Services werden bei @enterprise bei den Applikationsstammdaten definiert. Web-Services werden in WSDL (Web Service Description Language) beschrieben. Eine solche Beschreibung wird jedem Service unterlegt.

Web-Services im Prozesskontext

In der Prozessdefinition gibt es eigene Knotentypen für die Kommunikation mit Web-Services. In diesen Knoten wird ein registriertes Web-Service referenziert. Die folgenden Knotentypen für Web-Service Interaktionen gibt es:

- **Invoke**: Es wird ein Web-Service aufgerufen, die Verarbeitung des Ergebnisses erfolgt synchron. Graphisch sieht dies in BPMN folgendermaßen aus:

- **Receive**: Es wird auf eine eingehende Nachricht gewartet, nach Eintreffen der Nachricht (Aufruf des Web-Services durch eine andere Applikation) wird der Prozess fortgesetzt, oder, falls der Receive Knoten der erste Prozessknoten ist, eine neue Prozessinstanz gestartet.

- **Reply**: Der Reply Knoten folgt immer auf einen Receive Knoten (ist aber optional), zwischen Receive und Reply können sich Systemschritte befinden, welche die empfangenen Daten bearbeiten.

Der Datenaustausch zwischen dem Web-Service und den @enterprise Formularen kann deklarativ festgelegt werden. Dabei wird ein Mapping zwischen den Formularfeldern und den Elementen in den zu sendenden und empfangenden XML-Dokumenten festgelegt.

Aufruf von Funktionen des BPM-Systems

Eine Menge von Web-Services als Schnittstellen zu einer Workflow-Engine wird im Wf-XML Standard der Workflow Management Coalition definiert [**?**] und von @enterprise unterstützt. Die Web-Services stellen u.a. die folgenden Funktionen zur Verfügung:

- Liste der Prozessdefinitionen
- Prozessstart
- Abfrage von Prozessinstanzstatus und -daten
- Änderung des Status von Aktivitätsinstanzen

Weitere spezifische Web-Services können implementiert und am BPMS Server installiert werden.

3.8.4 Datenimport und Export

Für die Bearbeitung von Prozessen im Workflow ist häufig der Zugriff auf externe Daten nötig, z.B. beim Ausfüllen eines Bestellformulars der Zugriff auf Daten von Kunden, Artikeln oder Preisen. Man nennt diese Daten „workflow-relevante Daten“ (im Gegensatz zu Workflowdaten, die Daten, die direkt vom Prozess bearbeitet werden).

Die Daten werden entweder direkt im BPM-System verwaltet oder es muss ein Zugriff auf eine der folgenden Arten erfolgen:

- regelmäßiges Kopieren der Daten ins BPM-System: Dafür ist in @enterprise die File-Import Schnittstelle verwendbar. Damit kann ein regelmäßiger Abgleich mit aus Fremddatenbanken exportierten Dateien durchgeführt werden.

  Alternativ zu einem File können die Daten auch mit einem Web-Service gelesen werden.

- direkter Zugriff auf die Daten: Zum Beispiel kann eine Auswahl so implementiert werden, dass die Daten aus der Fremddatenbank gelesen werden und das ausgewählte Element in die Datenbank des BPM-Systems übernommen wird.

Um Daten, die im BPM-System erzeugt werden, anderen Systemen zur Verfügung zu stellen, gibt es ebenfalls mehrere Möglichkeiten:

- Zugriff auf die Datenbank: die Fremdapplikation kann direkt auf die entsprechenden Datenbanktabellen lesend zugreifen. Eventuell wird aus mehreren Tabellen eine Ansicht (View) erzeugt und nur diese wird freigegeben.

- Mit der Reporting Komponente kann ein Report erstellt werden, welcher regelmäßig durch einen Timer ausgeführt wird und die Ergebnisse in ein File schreibt.

- Die dritte Variante ist eine Applikationskopplung durch Web-Services: Das BPM-System bietet ein Web-Service zur Ausführung eines Reports bzw. des Datenabrufs an.

Die Entscheidung, welche dieser Varianten in einem gegebenen Fall die Günstigste ist, hängt davon ab, wie zeitnahe die Daten benötigt werden, wie eng die Kopplung sein darf und wie die Architektur des Partnersystems aussieht.

3.8.5 API-Programmierung im BPMS

Bei der Bearbeitung eines Geschäftsprozesses gibt es verschiedene Situationen, wo man mit den Methoden der Prozessmodellierung nicht auskommt und Programmierung nötig ist: Berechnungen, Bedingungsauswertungen, komplexe Akteurszuordnungen etc.

Im folgenden sei ein Überblick über die Eingriffsmöglichkeiten in den Bearbeitungsablauf gegeben:

1. Prozessablauf
   (a) Bedingungen im Prozessablauf: Verzweigungen, Schleifen, Auswahl, allgemeine Parallelität
   (b) Preprocessing: Aktion bevor Schritt in den Arbeitskorb kommt
   (c) Postcondition: Aktion vor dem Weiterleiten
   (d) Systemschritt: Workflow-Schritt mit Programmaufruf
   (e) Akteurauswahl
2. Formularbearbeitung:
   (a) Anpassung der Formulardarstellung (Eingabehilfen, Zusatzinformationen, ...)
   (b) Aktionen vor dem Einfügen, Ändern, Löschen
3. Erweiterung der Benutzerschnittstelle durch Applikations-Funktionen
4. Anpassung der Darstellung von Tabellen (Arbeitskorb, DMS, ...)
5. Zeitgesteuerte Aktionen: einmalig oder wiederkehrend
6. Aktionen beim Hochfahren oder Beenden des Systems

Charakteristisch bei allen diesen Punkten ist, dass diese Programme oft nur einen sehr kleinen Eingriff in den Ablauf vornehmen. Für den Ersteller von API-Programmen (API = Application Programming Interface) bedeutet dies, dass er eine Methode oder Klasse erstellt, die dann vom System aufgerufen wird, das Ergebnis wird wiederum vom BPMS interpretiert. Dieses Vorgehen nennt man *Inversion of Control* („Umkehrung der Steuerung“), siehe [**?**]. Die Steuerung wird an das Framework abgegeben. Der Vorteil für den API-Programmierer ist, dass er sich auf die Funktionalität konzentrieren kann und alle Umgebungsaspekte außer Acht lassen kann.

Das @enterprise API folgt weitgehend diesem Design-Prinzip. Für die obigen Eingriffe ist entweder eine einzelne Methode zu implementieren (z.B. für Bedingungen, Preprocessing, Systemschritt) oder es ist ein Java-Interface zu implementieren, das aus mehreren Methoden besteht (z.B. für Anpassung von Tabellen). Im zweiten Fall ist es jedoch so, dass zu den Interfaces Adapter-Klassen existieren, sodass nur die jeweils nötigen Methoden implementiert werden müssen.

Das folgende Beispiel soll dies verdeutlichen: In einem Systemschritt soll die Deadline einer Aktivität aus einem Formularfeld gelesen und gesetzt werden. Die Java-Methode, die dies durchführt, sieht folgendermaßen aus:

```
public void setDuedate() {
   WfEngine wfe = ServiceLocator.getWfEngine();
   ActivityInstance ai = wfe.getContext();
   ProcessInstance pi = ai.getParent();
   DMSForm f = wfe.getForm(pi, "item");
   Date dt = f.getField("duedate");
   if (dt != null)
      wfe.setDuedate(ai, dt);
}
```

Die Java-Methode `setDuedate` holt sich zuerst den Kontext – die aktuelle Aktivitätsinstanz `ai`, die Prozessinstanz `pi`, das Formular `f` und den Wert des Formularfelds `dt`. Mit der Engine-Funktion `setDuedate` wird die Deadline gesetzt.

Der Kontext (aktueller Prozess, Formulare) wird im Allgemeinen entweder über Funktionsparameter übergeben oder kann mit entsprechenden Methoden geholt werden (`wfEngine.getContext()`). Die API-Programme können beliebige Aktionen ausführen, z.B. Zugriff auf Datenbanken, andere Systeme, Versenden von E-Mails etc. Eine Transaktionsklammer ist durch die Umgebung bereitgestellt, d.h. die Änderungen werden automatisch kommitiert (engl.: commit), bei Auftreten eines Fehlers entstehen keine inkonsistenten Zustände.

Eine ausführliche Beschreibung des @enterprise APIs ist im Application Development Guide [**?**] und in der API-Dokumentation auf *https://www.groiss.com/customers/ep110/alllangs/api/index.html* zu finden.

3.8.6 Applikationsintegration am Client

Applikationsintegration am Client ist nötig und sinnvoll, wenn die zu integrierende Applikation über eine eigene Benutzerschnittstelle verfügt. Dies ist z.B. der Fall, wenn Dokumente des Prozesses mit Office Applikationen bearbeitet werden sollen. Der Web-Client ist hier sehr eingeschränkt, weshalb bei der Notwendigkeit zur besseren Einbindung von lokalen Applikationen und Hardware (z.B. Scanner) oft eine am Client installierte Software verwendet wird. Der @enterprise Java-Client bietet APIs an, um am Client Systemintegrationen durchzuführen.

Im Falle eines mobilen Clients haben wir die Integrationen bereits in Abschnitt 3.3.6 kennengelernt.

3.8.7 Sicherheitsaspekte

In einer Workflowapplikation arbeiten unterschiedlichste Personen, Abteilungen und Systeme zusammen. Dies macht die Architektur der Applikationssicherheit komplexer als bei vielen anderen Anwendungen.

Zugriff von außen

Eine wesentliche Frage ist, von wo aus auf das System zugegriffen werden kann: aus einem abgeschlossenem Netz, dem Intranet (innerhalb einer Organisation) oder dem Internet.

Kann aus dem Internet zugegriffen werden, ist das System jedenfalls besonders abzusichern, da die Wahrscheinlichkeit eines Hacker-Angriffs zumindest gegeben ist. Eine Absicherung des Zugriffs auf Basis Benutzername und Passwort ist wahrscheinlich zu wenig. Außerdem sollten Zugriffe auf die Systemadministration nur von bestimmten Arbeitsplätzen aus möglich sein, in @enterprise kann man dafür einen sogenannten URLChecker konfigurieren, siehe @enterprise Installation Manual [**?**], Abschnitt *Access Control*.

Autorisierung

Die Applikationen, die auf das System zugreifen, müssen genauso wie die Benutzer in der Autorisierung berücksichtigt werden, siehe Kapitel 3.8.2.

Sicherheit von API Zugriffen

Besonderes Augenmerk sollte auch der Integration mit Fremdsystemen gelten.

Zugriffe auf andere Systeme sind der einfachere Fall. Das andere System muss eine Zugriffsmöglichkeit bereitstellen. Kritisch ist nur der Fall, wenn aus dem Fremdsystem sicherheitsrelevante Daten übermittelt werden - zum Beispiel die Benutzerdaten von einem Directory Server. Es ist klar, dass bei Manipulation des Directory Servers die Zugriffssicherheit nicht mehr gewährleistet ist. Zur Prüfung der Authentizität des Gegenübers können gegebenenfalls Zertifikate verwendet werden.

Wenn andere Systeme das BPMS aufrufen, ist ebenfalls darauf zu achten, dass diese Systeme keinen unbeschränkten Zugang erhalten, die Web-Services

also nur eine begrenzte Zahl von hinsichtlich ihres Umfangs beschränkten Operationen erlauben.

Web-Services sollten außerdem ebenso der Authorisierung unterliegen wie interaktive Zugriffe von Benutzern. Auch hier kann die Authentizität mit Zertifikaten geprüft werden.

Systemumgebung

Obwohl das kein Spezifikum einer Workflowapplikation ist, sei hier darauf hingewiesen, dass die Sicherheit der Systemumgebung wesentlich zur Sicherheit des Gesamtsystems beiträgt. Was dies umfasst, sei nur stichwortmäßig angegeben:

- Sicherung des Datenbankzugriffs
- Sicherung des Netzwerks, physikalischer Zugang zu den Netzwerkkomponenten
- Sicherung der Server-Hardware, physischer Zugang, Benutzeraccounts etc.

3.8.8 Weitere Aspekte der Applikationsintegration

Abseits der funktionalen Anforderungen und der Sicherheitsaspekte gibt es bei der Integration von Applikationen weitere Aspekte zu beachten:

Transaktionen: Behandlung von Fehlerfällen. Hierzu ist eventuell ein Eskalationsmechanismus im Prozess vorzusehen, z.B. automatische Verständigung eines Prozessverantwortlichen. Besonderes Augenmerk ist Fällen zu schenken, wo eine Änderung im BPMS automatisch eine Änderung in einem anderen System auslöst und man davon ausgeht, das beide Systeme synchron gehalten werden.

Was passiert, wenn das Service nicht erreichbar ist. Dieser Fall kann ebenfalls durch Eskalationsmechanismen gelöst werden oder es ist möglich, falls die Kommunikation nur in eine Richtung verläuft, die Ereignisse vorübergehend in eine Queue (Warteschlange) zu stellen.

Logging: Nachvollziehbarkeit der Zugriffe. Die Zugriffe einer externen Applikation auf das BPMS sollten geloggt werden, sodass später nachvollziehbar ist, welche Daten ausgelesen wurden bzw. welche Änderungen an den Daten vorgenommen wurden. Je nach Anforderungen reicht eine Protokollierung in

einem Logfile, welches regelmäßig überschrieben wird oder es ist das persistente Speichern in der Datenbank nötig.

3.9 Anpassung der Benutzerschnittstelle in @enterprise

Bei einem BPMS handelt es sich um ein Framework, das für die unterschiedlichsten Arten von Prozessen geeignet sein soll und alle möglichen Funktionen zur Prozessbearbeitung bereitstellt. Um ein effizientes Arbeiten mit den definierten Prozessen zu unterstützen, wird man genau die Funktionen und Ansichten dem Benutzer zur Verfügung stellen, welche für die definierten Prozesse nötig und sinnvoll sind. Diese Anpassung der Benutzerschnittstelle soll nun behandelt werden.

3.9.1 GUI-Konfiguration

Die GUI-Konfiguration in @enterprise ermöglicht es, für unterschiedliche Rollen das jeweils gewünschte Erscheinungsbild herzustellen.

Manchmal ist *eine* solche Konfiguration nicht ausreichend. Wenn die Anforderungen für verschiedene Benutzergruppen relativ unterschiedlich sind, ist es nötig, mehrere GUI-Konfigurationen zu definieren. Bei jeder GUI-Konfiguration kann deshalb angegeben werden, für welche Rollen (oder Benutzer) diese Konfiguration Verwendung findet. Um bei mehreren Möglichkeiten eine Auswahl zu treffen, wird jede Zuordnung mit einer Priorität versehen.

Beispiel:

Akteur	Konfiguration	Priorität
Rolle `all`	`standard`	10
Rolle `manager`	`extended`	20

Jeder, der die Rolle `all` hat, bekommt die `standard` Konfiguration. Wer die Rolle `manager` hat, erhält die Konfiguration `extended`, egal ob er auch zusätzlich die `all` Rolle hat.

Eine Konfiguration definiert den Umfang des GUIs, eine Menge von Navigations-Links, die zu den einzelnen Funktionen führen. Je nach Linktyp können

verschiedene Einstellungen vorgenommen werden:

Text Ein Text in der Navigation.

Link Beliebiger Link oder einer aus den bestehenden @enterprise Funktionen.

Arbeitskorb Anpassung der Spalten, des Arbeitskorbtyps (persönlich, Rollen-AK, Wiedervorlage), der Toolbarfunktionen.

Benutzerordner Wurzel der benutzerdefinierten Ordner, Anpassungen wie oben.

Prozess starten Link zur Liste der startbaren Prozesse oder Startmaske eines einzelnen Prozesses.

Funktionen Link zur Liste der taskunabhängigen Funktionen.

Reports Liste aller Reports oder Aufruf eines bestimmten Reports.

DMS Dokumentenordner, anpassbar sind die Spalten und die Toolbarfunktionen.

Formulartabelle Tabelle eines Formulartyps, angepasst werden können die Tabellenspalten und die Toolbarfunktionen.

Abb. 3.19 zeigt ein Beispiel einer solchen angepassten Navigation. Gegenüber dem Standard wurde z.B. die Rollen-Wiedervorlage entfernt, Icons den Verweistexten vorangestellt und die Funktion „Zeiterfassung“ hinzugefügt.

3.9.2 Anpassung des Styles

Die Vorlieben für Farben und Schriften sind sehr individuell, deshalb wird in @enterprise dem Endbenutzer die Möglichkeit geboten, sich diese selbst einzustellen, siehe unter Extras/Style-Konfigurator.

Darüberhinaus ist es nötig, zentral den Style einstellen zu können, um das Aussehen z.B. an einen Corporate Style anzupassen. Dafür gibt es kein grafisches Interface, die Datei style.prop muss bearbeitet werden.

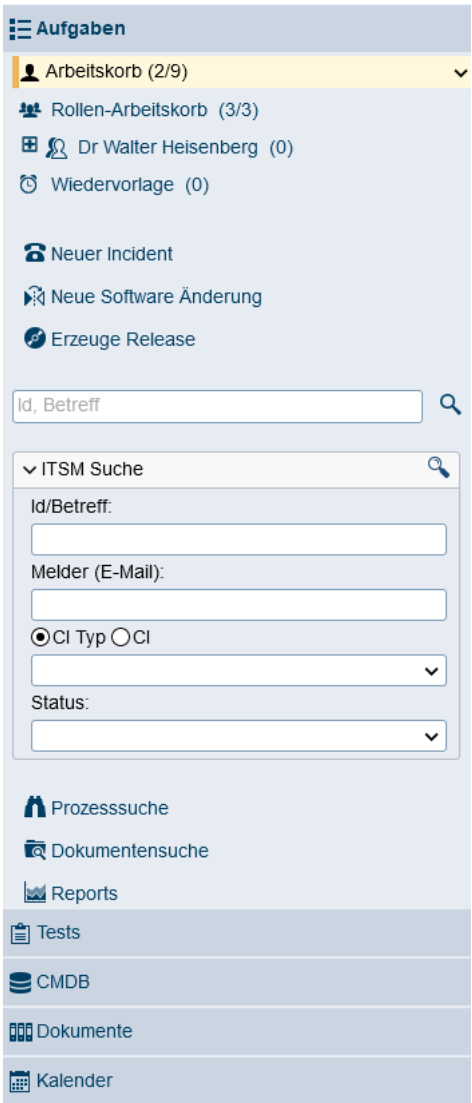


Abbildung 3.19: Angepasste Navigation

3.9.3 Internationalisierung

BPM-Systeme werden vornehmlich in großen Organisationen eingesetzt und in solchen ist es zu erwarten, dass von den Teilnehmern verschiedene Sprachen gesprochen werden.

Eine Art damit umzugehen, ist, die gesamte Benutzerschnittstelle in einer Sprache – der „Konzernsprache“ – zu erstellen. Oft ist dies jedoch z.B. aufgrund mangelnder Sprachkenntnisse der Mitarbeiter nicht möglich, die Applikation muss dann mehrsprachig sein.

Alle im GUI aufscheinenden Namen und Texte - Bezeichnungen, Beschreibungen und Hilfetexte von Prozessen, Formularen und anderen Applikationselementen - müssen also in die verwendeten Sprachen übersetzt werden.

@enterprise bietet dafür Hilfestellung an, sodass bereits während der Modellierung die Namen mehrsprachig erfasst werden können. Für die Namen von Prozessen, Tasks, Formularen etc. wird in der Erfassungsmaske statt dem Namen in das Namensfeld nur ein *Schlüssel* geschrieben, in einem weiteren Feld können dann die Übersetzungen in die verwendeten Sprachen eingegeben werden. Unter dem Navigationslink „Ressourcen“ können diese Schlüsselwerte und deren Übersetzungen verwaltet werden, siehe Abb. 3.20.

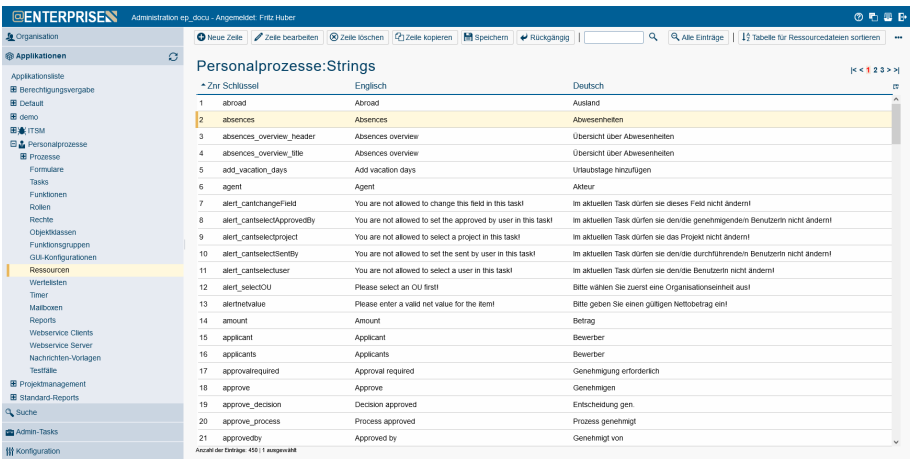


Abbildung 3.20: Editor für die Internationalisierung

3.10 Einen Prozess durchspielen

Im Kapitel Modellierung haben wir gelernt, Prozesse zu definieren. Inhalt dieses Kapitels war die Laufzeitumgebung. Nun bringen wir diese Prozesse zum Laufen, und zwar vorerst auf dem System, auf dem die Modellierung durchgeführt wurde. Mit dem Deployment auf ein Produktionssystem beschäftigen wir uns in Kapitel 5.6.

Um einen Prozess ausführen zu können, muss er ausreichend spezifiziert sein, d.h. für jeden interaktiven Schritt muss ein Task und ein Akteur eingetragen sein, jede Bedingung muss einen gültigen Ausdruck enthalten und jeder Systemschritt ebenfalls. Sind diese Voraussetzungen erfüllt, kann der Prozess aktiv gesetzt und somit gestartet werden.

Für das Bearbeiten des Prozesses ist es noch nötig, alle im Prozess verwendeten Rollen zuzuordnen. Dann kann man sich mit einer der vergebenen Benutzer-Ids einloggen und mit der Prozessbearbeitung beginnen.

Der Link zum Prozess starten befindet sich in der Navigation unter den Arbeitskörben, siehe Abb. 3.7. Nach dem Starten landet die Prozessinstanz im eigenen Arbeitskorb.

3.11 Die Elemente einer BPM Applikation

Zusammenfassend seien noch einmal die einzelnen Elemente einer BPM Applikation dargestellt:

- Prozessdefinitionen: ausführbare Prozesse
- Tasks: interaktive Schritte in einem Prozess mit Spezifikation des Preprocessing, der Postcondition etc.
- Formulartypen: beschreiben eine Entität zur Speicherung von Daten, die Darstellung im GUI, die Typen der Felder
- Funktionen: Funktionen, die bei der Ausführung des Prozesses unterstützen
- Rollen: zur Definition der Akteure von Prozessschritten
- Rechte: zur Definition von Zugriffen auf Daten, Funktionen etc.

- Internationalisierungstabellen/Ressourcen: enthalten die Übersetzungen von Namen und Labels für die Sprachen, in denen die Applikation verwendet wird.
- Web-Service Server und Clients: Definitionen der aufgerufenen und angebotenen Web-Services zur Applikationsintegration
- Reports: Auswertungen über die Daten, die für eine Benutzergruppe freigegeben werden
- Timer: zur wiederkehrenden, automatischen Ausführung von Aktionen

Neben diesen Elementen, die in der Datenbank abgelegt werden, werden bei der Implementierung Programmcode, HTML-Seiten, Icons und Hilfetexte erstellt.

Kapitel 4

Monitoring und Optimierung

Der Zyklus des Prozessmanagements umfasst, wie im Einleitungskapitel erwähnt, das Monitoring der Vorgänge und die Optimierung der Prozesse. Ausgangspunkt für eine Optimierung sind die Prozessdefinitionen und die Laufzeitdaten der Instanzen.

4.1 Auswertung der Laufzeitdaten

Bei der Ausführung von Prozessen fällt eine Menge von Daten an, welche für die Analyse und Optimierung der Prozesse herangezogen werden können.

Aus den Informationen der Prozessinstanzen und Aktivitätsinstanzen können z.B. folgende Aussagen getroffen werden:

- Was ist der Inhalt des Arbeitskorbs von Akteur A?
- Wie lang sind die Arbeitskörbe der Mitarbeiter meiner Abteilung?
- Wieviel Prozesse von Typ X sind derzeit aktiv?

Derartige Datenbankabfragen oder *Reports* können regelmäßig durchgeführt werden, um einen Überblick über die Systemvorgänge zu erhalten. Mit einer Menge von Standard-Reports können die wesentlichen Aussagen über

Performance-Daten getroffen werden. Darüberhinaus kann es Reports, die spezifisch für Prozesstypen sind, geben.

In den folgenden Unterkapiteln werden die Mechanismen der Aufbereitung der Laufzeitdaten beschrieben: die Reporting-Komponente zur Ermittlung der Daten, das Dashboard zur Darstellung der wichtigsten Informationen und das Prozess-Cockpit als Gesamtdarstellung.

4.1.1 Reporting

Eine Reporting-Komponente ist integraler Bestandteil eines BPM-Systems, sie ermöglicht es, Auswertungen über alle Informationen in der Datenbank zu erstellen. Weitere Beispiele von Reports (neben den oben genannten das Laufzeitverhalten betreffenden) sind:

- Liste aller unbezahlten Rechnungen und Summe der ausstehenden Beträge (wenn die Rechnungsbearbeitung in einem Workflow erfolgt).
- Anzahl der offenen Incidents pro Produkt.
- Monatsweise Entwicklung der Fehlermeldungen zu einem Produkt.

Wichtig ist die Möglichkeit, die Daten aus der Prozesshistorie mit den Anwendungsdaten zu verknüpfen.

Die Erstellung von Reports kann recht komplex sein, da Wissen über die Daten und deren Zusammenhang nötig ist. Außerdem liegen dem Ersteller eines Reports alle Daten offen, sodass es nötig ist, den Kreis der Personen, die Reports erstellen dürfen, einzuschränken. Sind solche Reports einmal erstellt, können sie für bestimmte Rollen zur Ausführung freigegeben werden. Die Ausführung erfolgt dann in der Regel in Echtzeit, d.h. wenn der Report angeklickt wird, werden die Daten aus der Datenbank extrahiert und die Ergebnisse dargestellt.

Abb. 4.1 zeigt die Maske zur Erstellung von Reports in @enterprise. Schrittweise kann man dort die einzelnen Elemente eines Reports zusammenstellen. Welche Elemente dies sind, betrachten wir im folgenden:

- Die Anzeigeattribute: Welche Daten werden im Ergebnis dargestellt.
- Bedingungen: Einschränkungen der Ergebnismenge
- Parameter für Bedingungen: Bedingungen können mit Parametern versehen werden, um die Ergebnismenge spezifisch für eine Ausführung

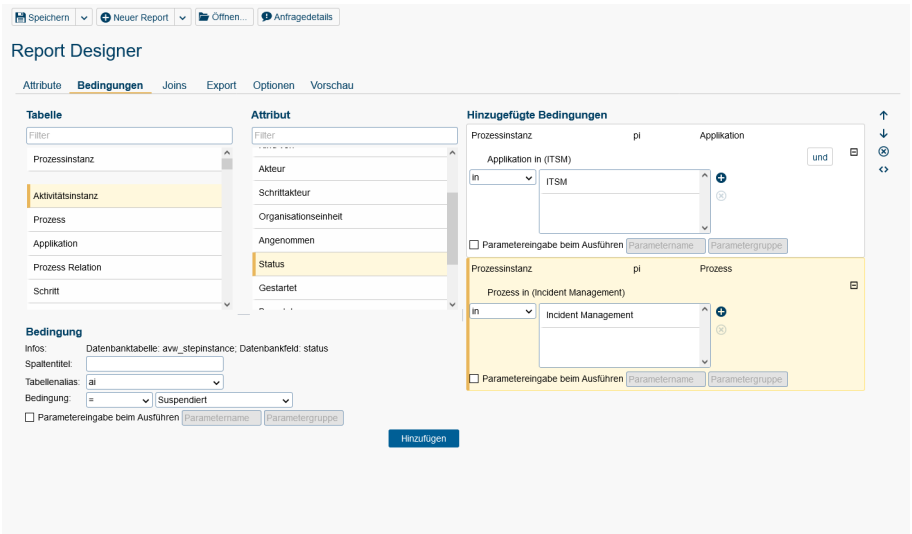


Abbildung 4.1: Reporting Maske

einzuschränken. Zum Beispiel Einschränkung der Prozesse auf ein bestimmtes Jahr, das Jahr wird bei jeder Ausführung des Reports angegeben.

- Implizite Parameter: dienen dazu, die Ergebnismenge abhängig vom Ausführungskontext einzuschränken. Bestandteile des Kontexts sind: der ausführende Benutzer, seine Stamm-Organisationseinheit und der Ausführungszeitpunkt. So kann ein Report formuliert werden, der die Prozesse des aktuellen Monats in der Organisationseinheit des Ausführenden anzeigt.

- Aggregationen: Eine der wesentlichen Aufgaben des Reportings ist das Verdichten der Daten: Zählen von nicht numerischen Daten wie Prozessen, Dokumenten etc.; Bildung von Summen, Mittelwerten, Minima und Maxima von numerischen Werten. Die Aggregation kann mehrstufig verlaufen, zum Beispiel werden im ersten Schritt die Anzahl der Prozesse pro Monat berechnet, im zweiten Schritt werden die Monate mit der kleinsten und der größten Prozessanzahl berechnet.

- Darstellung: Die einfachste Ergebnisdarstellung ist die Tabelle, die nor-

malerweise direkt im Web-Browser angezeigt wird. Um einen Report weiterzugeben oder abzulegen, wird gewöhnlich PDF (Portable Document Format) verwendet. Zur Weiterverarbeitung in einem anderen System kann das CSV-Format (Comma Separated Values) herangezogen werden. Eine direkte Erzeugung einer MS-Excel Tabelle ermöglicht das Weiterbearbeiten mit dieser weit verbreiteten Software.

Für verdichtete Reports sind graphische Darstellungen sehr übersichtlich: Gängig sind Balken-, Torten- und Liniendiagramme.

- Verknüpfungen zwischen Reports: Damit kann der Detaillierungsgrad schrittweise durch Navigation zwischen den Reports erhöht werden.

Ein Beispiel für verknüpfte Reports ist in Abb. 4.2 dargestellt. Ein allgemeiner Report zeigt die Anzahl von vorhandenen Prozessinstanzen pro Prozessdefinition. Mit Klick auf einen der Balken kommt man zu einem zweiten Report, der die Instanzen der ausgewählten Prozessdefinition im aktuellen Monat anzeigt, guppiert nach Produkten und aufgeteilt in offene und abgeschlossene Fälle. Will man z.B. die offenen Fälle (eines Produkts, eines Monats) einzeln sehen, kann man wiederum auf einen der Balken klicken. Damit erhält man eine Auflistung von Prozessinstanzen – von der man über einen Link zur Prozessdetailinformation mit Historie, den Formularen und Dokumenten gelangt.

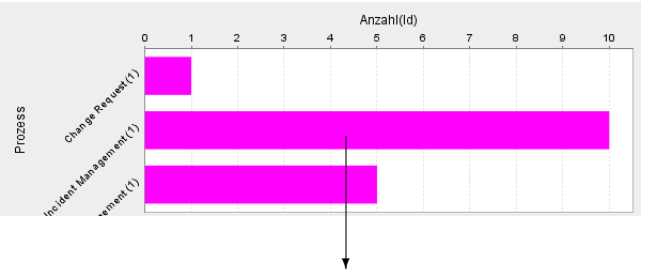
Im GUI des BPM Systems wird man an geeigneter Stelle in der Navigation einen Link auf die Liste der ausführbaren Reports bereitstellen. Dies ist aber nicht der einzige Platz, an dem man Reports platzieren kann.

4.1.2 Dashboard

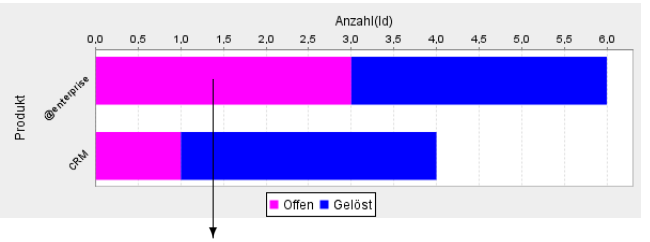
Das Dashboard (Englisch für Armaturenbrett) soll einen personalisierten Überblick über die Systemvorgänge bieten. Jeder Benutzer hat die Möglichkeit, sich die für ihn wichtigen Informationen auf dieser Seite zusammenzustellen, neben Reports können das sein: eine Übersicht über die Arbeitskörbe (Anzahl der Neuen, der Fälligen, aller Einträge), Termine, aktuell eingeloggte Benutzer etc. Abb. 4.3 zeigt ein Beispiel eines Dashboards in @enterprise.

Das Dashboard kann auch als Startseite verwendet werden, sodass man nach dem Einloggen die wichtigsten Informationen gleich vorfindet.

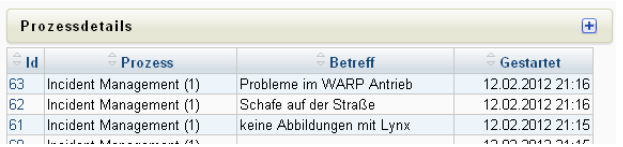
Report 1: Prozessinstanzen ITSM



Report 2: Incidents pro Monat



Report 3: Incidents (offen, Produkt @enterprise)



Prozessdetails

Id	Prozess	Betreff	Gestartet
63	Incident Management (1)	Probleme im WARP Antrieb	12.02.2012 21:16
62	Incident Management (1)	Schafe auf der Straße	12.02.2012 21:16
61	Incident Management (1)	keine Abbildungen mit Lynx	12.02.2012 21:15

Abbildung 4.2: Drill-down Reports

4.1.3 Prozess-Cockpit

Während die Reports verschiedene Detailinformationen liefern können und das Dashboard für den schnellen Überblick geeignet ist, ist für die Gesamtsicht der Prozesse - Definitionen und Instanzen - eine andere Darstellung nötig. In Kapitel 1 haben wird das Prozesshandbuch kennen gelernt, als Sammlung aller dokumentierten Prozesse im Unternehmen. Das sogenannte *Prozess-Cockpit* ist eine Erweiterung davon. Es enthält nicht nur die Prozessdefinitionen, sondern auch aktuelle Laufzeitinformationen der Prozessinstanzen, sofern solche vorhanden sind.

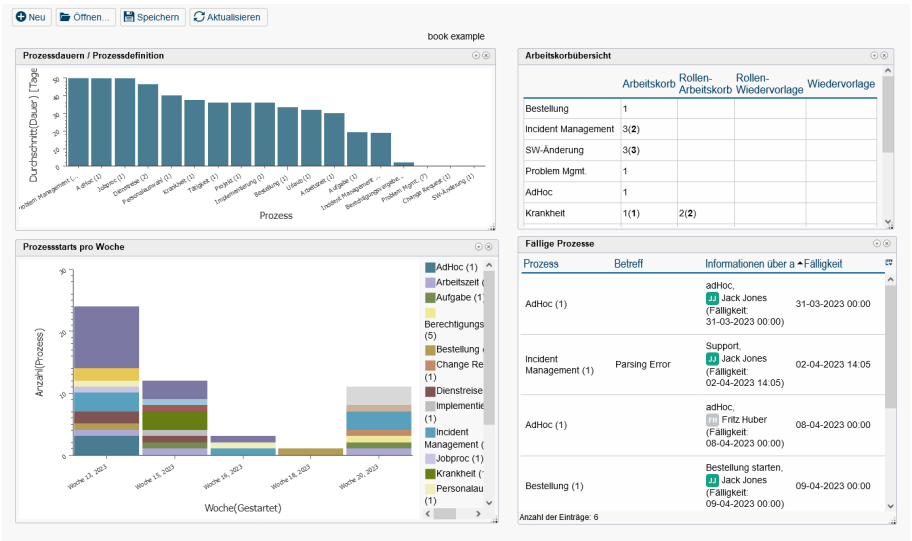


Abbildung 4.3: Dashboard

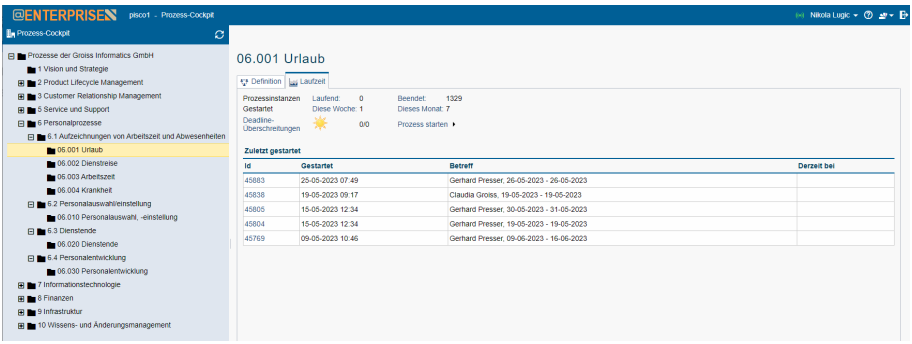


Abbildung 4.4: Prozess-Cockpit

Abb. 4.4 zeigt die Einstiegsseite eines Prozess-Cockpits. Auf dieser werden hierarchisch die Prozesse des Unternehmens dargestellt, die Gliederung folgt hier dem Process Classification Framework der APQC (American Productivity & Quality Center) [?]. Für jeden Prozess sind mehr oder weniger Detailinfor-

mationen vorhanden, abhängig vom Implementierungsgrad des Prozesses. Es gibt

1. Prozesse, die nur textuell beschrieben sind, es existiert dazu kein Workflow-Prozess;
2. Prozesse, zu denen eine Prozessdefinition existiert, die aber nicht im BPM-System ausgeführt werden
3. und Prozesse, die im System auch ausgeführt werden.

Das Prozess-Cockpit enthält dementsprechend die folgenden Informationen:

- ❑ Dokumente, die mit dem Prozess im Zusammenhang stehen: textuelle Prozessbeschreibungen, Durchführungsbestimmungen, Hilfetexte. Diese Information ist für alle Prozesse abrufbar.
- ❑ Darstellung der Prozessdefinition für Prozesse der Gruppen 2 und 3.
- ❑ Für die Prozesse der dritten Gruppe können Informationen über das Laufzeitverhalten abgerufen werden.

Abb. 4.5 zeigt die Ansicht eines Prozesses im @enterprise Prozess-Cockpit. Die Informationen über die Prozessdefinition und die Instanzen sind in zwei Reiter gegliedert. Die Ansicht ist pro Prozess konfigurierbar: Auswählbar sind die Reports, die direkt ausgeführt werden, Reports, die auf der Seite verlinkt sind, Funktionen, sowie beliebige weitere Links.

Die Ansicht unterliegt natürlich dem Berechtigungssystem. Prozessverantwortliche können alle Details zu den einzelnen Instanzen sehen, Prozessbeteiligte die eigenen Prozesse und Unbeteiligte ausgewählte Reports, z.B. mit allgemeinen Performance-Informationen.

4.1.4 Überwachung des Laufzeitverhaltens

Aus den historischen Daten der Prozessinstanzen können wir ermitteln, wie lange die Prozesse und Tasks bisher gedauert haben. Nun ist es aber auch interessant zu wissen, wie lange ein in Bearbeitung befindlicher Prozess noch dauern wird und ob er die geplante Deadline einhalten wird können.

Bei einer Sequenz von Tasks ist das recht einfach zu berechnen: Die verbleibende Zeit ist einfach die Summe der Zeit, die für die verbleibenden Tasks

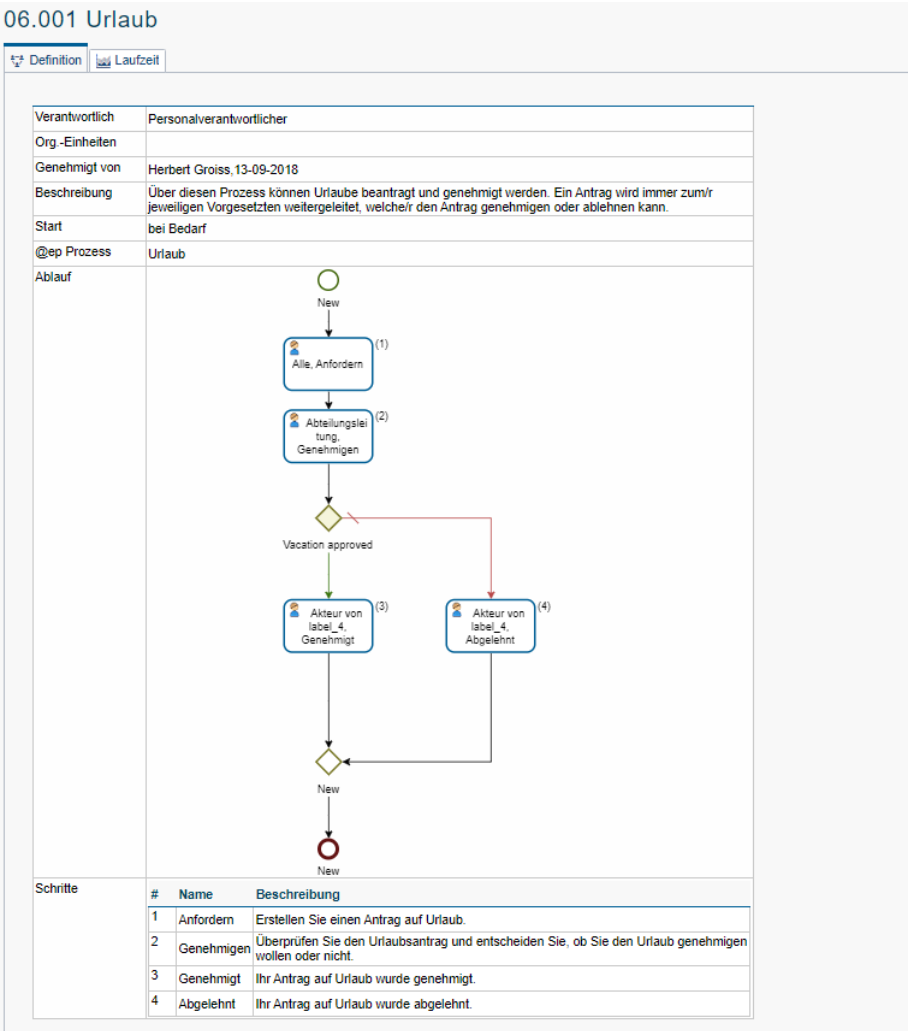


Abbildung 4.5: Prozess-Cockpit, Ansicht eines Prozesses

nötig ist. Enthält der Prozess Verzweigungen, Schleifen oder Parallelitäten, ist die Berechnung komplizierter. Noch komplizierter wird es, wenn für die Bearbeitung eines Schritts keine feste Zeit sondern eine Verteilung angegeben wird (dauert drei bis sieben Tage, meistens vier).

Wir wollen im folgenden den *Probabilistischen Prozessgraphen* vorstellen, mit dem diese Berechnungen möglich sind [**?**].

Dabei werden zwei Arten von Daten benötigt, die aus den Instanzen ermittelt werden oder manuell erfasst werden können:

- Die jeweiligen Dauern der einzelnen Prozessschritte: Für jeden Schritt gibt es pro Prozessinstanz, die ihn durchlaufen hat, einen Meßwert. Um mit diesen Werten rechnen zu können, werden sie in einem Histogramm mit wenigen Kategorien, zum Beispiel zehn, verdichtet. Die Histogramme stellen wir folgendermaßen dar:

p	t
0.7	3
0.3	4

  In der linken Spalte sind die Wahrscheinlichkeiten angegeben (summieren sich auf 1), in der rechten die Dauern. In diesem Fall dauert der Schritt also mit Wahrscheinlichkeit 0.7 3 Zeiteinheiten, mit Wahrscheinlichkeit 0.3 4 Zeiteinheiten. Die Einheiten können z.B. Minuten oder Stunden sein und werden der Einfachheit halber hier generell weggelassen.

- Verzweigungswahrscheinlichkeit: Für jede Kante, die von einem Verzweigungsknoten wegführt, wird die Wahrscheinlichkeit, dass die Prozessausführung dieser Kante folgt, ermittelt.

Diese Daten können automatisch aus den historischen Daten eines Prozesses ermittelt oder vorausschauend manuell erfasst werden.

Abb. 4.6 zeigt einen Prozessgraph mit den Taskdauern und Verzweigungswahrscheinlichkeiten. Als Dauer wurde jeweils nur ein Wert und kein Histogramm genommen, um die anschließenden Berechnungen nicht zu kompliziert werden zu lassen. Die Dauern sind neben den Tasknamen angegeben, d.h. „A|2“ bedeutet Task A dauert 2 Zeiteinheiten. Die Verzweigungswahrscheinlichkeiten bei Verzweigung und Auswahl sind in Prozent angegeben.

Dieser erweiterte Prozessgraph kann nun dazu verwendet werden, die Gesamtdauer des Prozesses sowie die Restdauer des Prozesses von jedem Knoten weg zu berechnen.

Um diese „Zwischenzeiten“ zu berechnen, beginnt man beim Prozessende und summiert die Laufzeiten Knoten für Knoten auf. Zur Summenbildung von Histogrammen werden die Wahrscheinlichkeiten multipliziert, die Laufzeiten addiert.

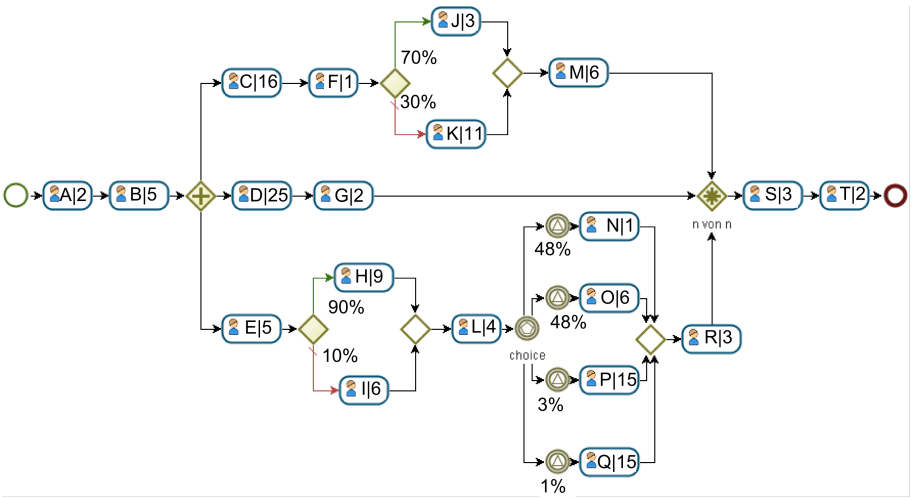


Abbildung 4.6: Probabilistischer Prozessgraph

Beispiel:

p	t
0.7	3
0.3	4

p	t
0.4	5
0.6	6

p	t
0.28	8
0.54	9
0.18	10

Bei einer Verzweigung wird das Ergebnishistogramm folgendermaßen berechnet: Jeder Eintrag jedes der beiden Zweige wird mit der Verzweigungswahrscheinlichkeit gewichtet und zum Ergebnishistogramm hinzugefügt.

Bei einer Und-Parallelität werden die Maximalwerte der Dauern jedes Zweigs genommen, die Wahrscheinlichkeiten werden wieder multipliziert:

p	t
0.7	12
0.3	15

p	t
0.6	10
0.4	13

p	t
0.42	12
0.28	13
0.18	15
0.12	15

Auf diese Weise können die Restlaufzeiten für einen gesamten Prozess berechnet werden, siehe Abb. 4.7 (aus [**?**]). Der Prozessgraph ist der gleiche wie

oben, nun erweitert um die Restlaufzeiten, die unter jedem Task als Tabelle dargestellt sind. Beginnen wir mit dem letzten Task, T, der zwei Zeiteinheiten dauert. Die Restlaufzeit des Prozesses, wenn T gestartet wird, sind also zwei Zeiteinheiten. Beim Task davor, S, sind es bereits fünf (2+3). Entsprechend den weiteren Berechnungen ergibt sich, dass der gesamte Prozess zwischen 39 und 48 Zeiteinheiten dauert, je nachdem, welche Abzweigungen im Prozessablauf genommen werden.

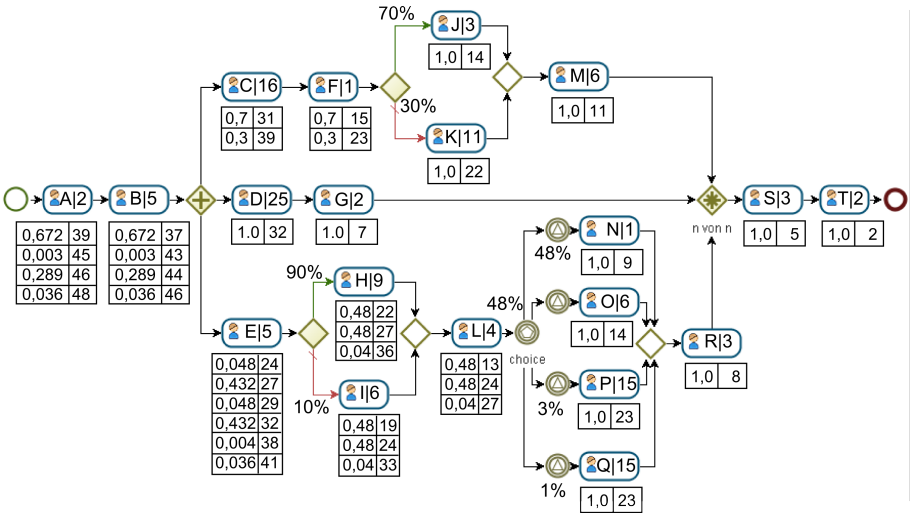


Abbildung 4.7: Probabilistische Berechnung der Prozessdauer

Nutzung der Zeitinformationen bei der Prozessausführung

Wurde der probabilistische Prozessgraph für eine Prozessdefinition berechnet, kann er zur Laufzeit für die folgenden Funktionen verwendet werden:

- Vorschlag einer Deadline beim Prozessstart: Bei einer fixen Vorgabe für eine Prozesslaufzeit wäre die vorgeschlagene Deadline der aktuelle Zeitpunkt plus die Prozessdauer. Beim probabilistischen Modell gibt es keine Prozessdauer, sondern eine Verteilungskurve. Hier ist die vorgeschlagene Deadline der Zeitpunkt, an dem die Wahrscheinlichkeit, dass der Prozess fertiggestellt ist, größer gleich n ist. n ist dabei einstellbar und wird im Normalfall recht hoch, etwa bei 0,9 liegen.

- Anzeigen von realistischen Deadlines: Für jede Aktivitätsinstanz kann ein Zeitplan angezeigt werden. Es wird dargestellt, wieviel Zeit bereits verbraucht wurde, wieviel noch zur Verfügung steht und wieviel Restzeit für den gesamten Prozess noch übrig ist.
- Auslösen von Eskalationen: Da die Deadline für jeden einzelnen Schritt aus dem probabilistischen Graphen errechnet wird, können Eskalationen bereits frühzeitig ausgelöst werden: wenn die Deadline einer Aktivität verletzt wird und somit die Wahrscheinlichkeit der rechtzeitigen Fertigstellung des Prozesses unter einen Schwellwert fällt.

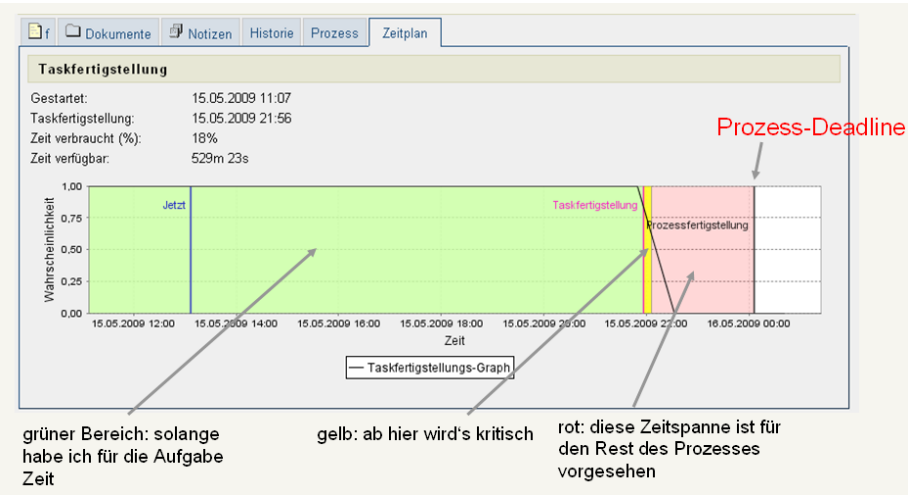


Abbildung 4.8: Zeitinformationen zu einer Aktivitätsinstanz

Abb. 4.8 zeigt die Zeitinformationen aus dem probabilistischen Prozessgraphen in der Detaildarstellung einer Prozessinstanz. In der Graphik ist der Startzeitpunkt der aktuellen Aktivität sichtbar, der aktuelle Zeitpunkt (Balken „Jetzt“), der späteste Zeitpunkt für die Taskfertigstellung und die Prozess-Deadline.

Der Nutzen der Zeitinformationen ist umso höher je ähnlicher sich die einzelnen Prozessinstanzen sind und je strukturierter der Prozess ist. Dies ist meist bei den sogenannten Produktionsprozessen, wo es eine hohe Zahl gleichartiger Instanzen gibt, der Fall.

Mithilfe der Zeitinformationen können weitere Berechnungen durchgeführt

werden:

- Es kann ermittelt werden, wann welche Tasks voraussichtlich gestartet werden. Zusammen mit Informationen über vorhandene Ressourcen kann somit die Ressourcenauslastung in der Zukunft abgeschätzt werden.
- Die einzelnen Mitarbeiter können eine Vorausschau über ihre zukünftige Belastung abrufen [**?**].
- Die einzelnen Aktivitäten in den Arbeitskörben können so gereiht werden, dass die Gesamt-Performance des Systems optimiert wird, z.B. nach der Anzahl der Deadlineverletzungen.

Diese oben genannten Aspekte sind Gegenstand laufender Forschung und Entwicklung, zeigen aber das Potential der Nutzung der Zeitinformationen auf.

4.2 Optimierung von Prozessen

Wir kommen nun zu einem Hauptziel des Geschäftsprozessmanagements und des Einsatzes von BPMS: Steigerung der Effizienz von Geschäftsprozessen. Es ist weder ein Zufall noch schlechtes Design, dass dieses Kapitel, welches viele Leser interessieren wird, fast am Ende des Buchs kommt. Prozessdefinition, Bearbeitung in einem BPMS und Analyse der Laufzeitdaten sind Voraussetzungen für die Prozessoptimierung.

Mit all den Detailinformationen aus Reports, Dashboard und Cockpit können wir daran gehen, unsere Prozesse zu optimieren. Die im BPMS gesammelten quantitativen Daten sind natürlich nur ein Teilaspekt und für eine Optimierung im Allgemeinen nicht ausreichend, da sie keine Aussagen z.B. über Kundenzufriedenheit oder Produktgüte geben. Diese qualitativen Informationen müssen ebenfalls erhoben werden.

Für eine Prozessoptimierung kann es unterschiedliche Stoßrichtungen geben:

- Geschwindigkeit: Ziel ist es, möglichst kurze Prozessdurchlaufzeiten zu erreichen, wobei der Fokus entweder auf der Durchschnittszeit liegen kann oder darin, Ausreißer möglichst zu vermeiden.

- Qualität: Es soll ein optimales Ergebnis erreicht werden, sodass möglichst selten weitere Iterationen zur Verbesserung nötig sind - z.B. bei der Herstellung eines Produkts.
- Ressourcenauslastung: Es soll eine möglichst optimale Ressourcenauslastung gefunden werden, um die Prozesskosten zu minimieren.

Die Ziele widersprechen einander teilweise, sodass es nicht möglich ist, einen Prozess gleichzeitig in alle Richtungen zu verbessern, mehr Qualität bedeutet möglicherweise längere Bearbeitungszeiten. Niedrigere Wahrscheinlichkeit von Deadlineverletzungen bedeutet schlechtere Ressourcenauslastung. Es ist daher Aufgabe des Managements, die Stoßrichtung auszuwählen.

Entsprechend dem Fokus dieses Buches beschäftigen wir uns nicht mit der Optimierung von Prozessen im Allgemeinen sondern mit der Optimierung, die im Zusammenhang mit dem BPM-System steht. Dabei sind drei Aspekte zu untersuchen:

- verbesserter Ressourceneinsatz,
- optimierter Ablauf,
- Optimierung der Aktivitäten.

Weitere Aspekte, auf die wir nicht eingehen, sind zum Beispiel organisatorische Aspekte, Aufgabenbearbeitung außerhalb des BPM-Systems, Definition und Gewichtung der Ziele.

Die Optimierungen können in verschiedenen Stufen durchgeführt werden (vgl. [**?**]), siehe Abb. 4.9:

- Innerhalb des BPMS: Das System nimmt ohne manuellen Eingriff Reihungen vor.
- Operatives Management: Die Prozesse werden beobachtet und in den Ablauf wird eingegriffen.
- Strategisches Management: Es wird nicht in einzelne Prozessinstanzen eingegriffen, sondern es werden die Prozessdefinitionen selbst, die Teilaufgaben oder die Ressourcenzuordnung angepasst.

Die Stufen unterscheiden sich prinzipiell durch ihren Zeithorizont, den Aufwand und ihre nachhaltige Wirksamkeit.

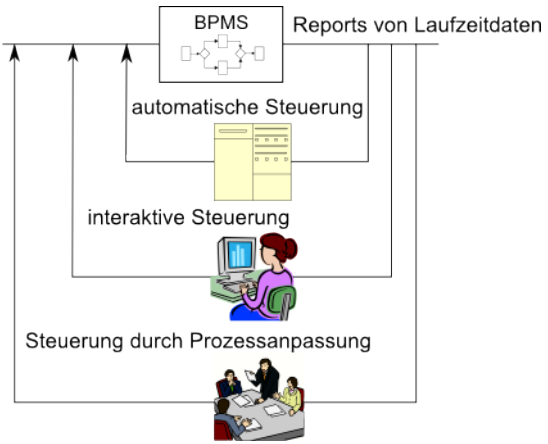


Abbildung 4.9: Zyklen der Optimierung

4.2.1 Optimierung innerhalb des BPM-Systems

Treten bei der Bearbeitung von Geschäftsprozessen Engpässe auf, können Mechanismen im BPM-System dazu beitragen, diese zu beheben.

Da die zur Verfügung stehenden Ressourcen vom BPM-System nicht vermehrt werden können, ist nur eine Neuverteilung der Arbeit möglich. Dabei werden von jedem laufenden Geschäftsprozess zwei Kriterien als Eingangsgrößen genommen:

- Priorität des Prozesses
- Wahrscheinlichkeit der Deadlineverletzung

Aus beiden kann eine Dringlichkeit des Prozesses berechnet werden, z.B. durch Multiplikation beider Größen miteinander.

Das zweite Kriterium für ein Re-Scheduling von Aktivitäten ist die Belastung der Ressourcen. Diese kann aus den Einträgen im Arbeitskorb, jeweils gewichtet mit dem Aufwand, ermittelt werden.

Eine Scheduling-Komponente kann nun die folgenden Funktionen implementieren:

- neue Aktivitätsinstanzen: werden automatisch den Bearbeitern mit der

geringsten Belastung zugeordnet. D.h. die Annehmen Funktion wird automatisch ausgeführt.

- bestehende Aktivitätsinstanzen: können in regelmäßigen Intervallen neu zugeordnet werden, falls mit der Arbeit noch nicht begonnen wurde.
- Die Berechnung von Deadlines neuer Prozesse kann auf Basis der aktuellen Ressourcenbelastung erfolgen.

Werden die Aktivitäten im Arbeitskorb nach Dringlichkeit geordnet, bleibt die Frage, ob man den Benutzern erlaubt, diese auch in einer anderen Reihenfolge abzuarbeiten. Das würde natürlich den Bemühungen des Schedulers zuwiderlaufen, ist aber die gängige Methode: Ein System, das den Benutzer dazu zwingt, von mehreren zu erledigenden Tätigkeiten genau eine bestimmte als Nächstes zu erledigen, wird wahrscheinlich auf keine große Akzeptanz stoßen.

4.2.2 Optimierung durch das operative Management

Dem operativen Management stehen die Informationen aus dem Prozess-Cockpit und den Performance-Reports zur Verfügung. Mögliche Aktionen auf dieser Ebene sind:

- Manuelle Zuordnung zu bzw. Verschiebung von Aktivitätsinstanzen zwischen Benutzern.
- Kapazitätserhöhung: Engpässe können auch dadurch behoben werden, indem Ressourcen umgeschichtet werden, z.B. nicht ausgelasteten Benutzern aushilfsweise eine zusätzliche Rolle zuzuteilen. Ebenso könnten Dienstzeiten ausgeweitet werden.
- Veränderung von Deadlines: Die bestehenden Prozessinstanzen werden analysiert: Eventuell können Deadlines verschoben oder Prioritäten herabgesetzt werden, ohne dass Verträge oder Zusagen an Kunden verletzt werden.
- Anpassung der Deadlines für neue Prozesse. Damit kann man den Kunden frühzeitig über eine zu erwartende längere Prozesslaufzeit informieren.

Die Verantwortung für solche manuellen Eingriffe liegt üblicherweise beim Prozessverantwortlichen (Process-Owner).

4.2.3 Optimierung durch das strategische Management

In der äußersten Feedbackschleife gemäß Abb. 4.9 gibt es zwei Möglichkeiten, die Performance zu verbessern:

- Weitere Ressourcen herbeischaffen: Wenn alle Möglichkeiten der Umschichtung ausgeschöpft sind und die Prozesse in der einmal definierten Art und Weise durchgeführt werden müssen, ist wohl eine Lösung, weitere Ressourcen herbeizuschaffen. Dies kann gegebenenfalls saisonal erfolgen.
- Prozesse optimieren: Zuletzt nun die wesentlichste und meist ertragreichste Optimierung: die Verbesserung der Prozesse. Es kann einerseits die Prozessstruktur verbessert werden, andererseits können die einzelnen Schritte optimiert werden.

Während die Ressourcenkontrolle nicht weiter betrachtet wird, wird auf die Prozessoptimierung - eine der Hauptaufgaben des Managements - im folgenden Abschnitt näher eingegangen.

4.2.4 Veränderung der Prozessstruktur

Betrachten wir zuerst die Möglichkeiten bei der Veränderung der Prozessstruktur. Abb. 4.10 zeigt eine Übersicht.

a) Parallelisierung

Wenn zwei Schritte B und C hintereinander ausgeführt werden, die Ergebnisse von B allerdings C nicht beeinflussen, ist es möglich, B und C parallel auszuführen.

b) Schritte zusammenfassen

Wenn es so ist, dass ein Bearbeiter meist oder immer zwei aufeinanderfolgende Schritte ausführt, dann sollte man diese zu einem zusammenfassen: Werden die bei beiden Schritten notwendigen Aktionen auf einmal ausgeführt, bedeutet dies meist eine Zeitersparnis, da die Start und Endaktivitäten bei der Bearbeitung der Aktivität nur einmal durchgeführt werden (Finden der Zeile in der Worklist; Öffnen; Lesen und Herausfinden, worum es geht; Schließen des Falls; Weiterleiten; Auswahl des Nachfolgepfads oder Akteurs).

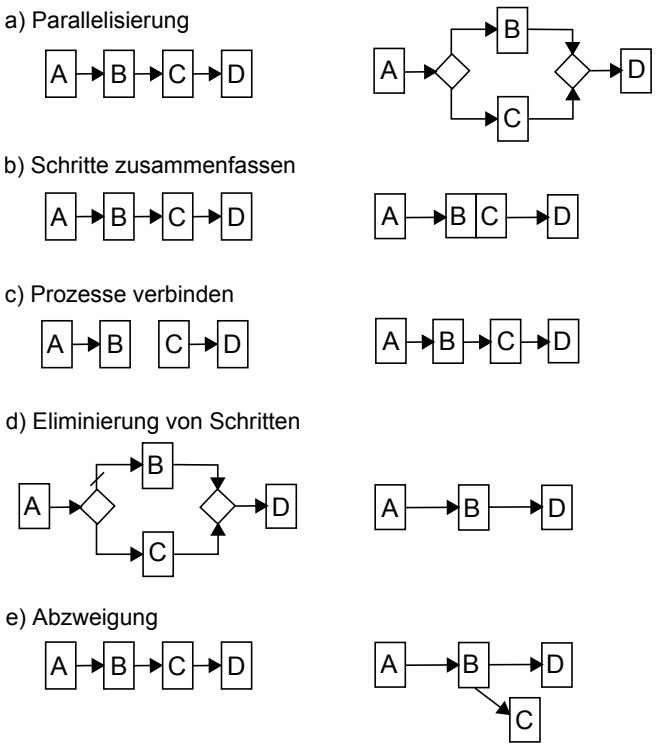


Abbildung 4.10: Optimierung durch Änderung der Prozessstruktur

c) Verbinden von Prozessen

Oft sind die Ergebnisse eines Prozesses die Eingaben für den Nachfolgeprozess.

Ein einfaches Beispiel ist die Beantragung einer Dienstreise. Wird die Dienstreise genehmigt, erfolgt später eine Spesenabrechung. Dies sind Teile eines Gesamtprozesses, der als ein Workflow modelliert und ausgeführt werden sollte. In der Zeit zwischen Genehmigung – normalerweise vor Antritt der Reise – und Spesenabrechnung nach der Reise kann der Prozess in der Wiedervorlage liegen. Damit werden Doppeleingaben vermieden und es kann der zweite Teil des Prozesses automatisch mit dem Enddatum der Reise gestartet werden. D.h. das System nimmt zu diesem Zeitpunkt den Prozess automatisch aus der Wiedervorlage.

Eine Möglichkeit der Prozessverbindung, die größtes Optimierungspotential hat, ist die Verknüpfung von Prozessen über Organisationsgrenzen hinaus. Hierzu ist es erforderlich, mit der Partnerorganisation die nötigen organisatorischen und technischen Abstimmungen zu treffen. Der Bereich der B2B Kommunikation ist allerdings eine umfangreiche Materie und geht über den Fokus dieses Buchs hinaus.

d) Eliminierung von Schritten

Aus den Analysedaten ist bekannt, welche Schritte kaum oder nie durchgeführt werden. Es sollte geprüft werden, ob es aus fachlicher Sicht möglich ist, die Schritte zu eliminieren oder mit anderen zusammenzufassen. Dadurch könnte der Prozess vereinfacht werden.

e) Eliminierung von Genehmigungsschritten

Viele administrative Workflows enthalten Genehmigungsschritte. Manchmal kann man diese durch parallele Informationsschritte ersetzen, d.h. der Antrag wird nicht genehmigt und kann danach weitergeführt werden, sondern der Vorgesetzte wird nur informiert (in einem parallelen Schritt). Der Prozess läuft ohne Verzögerung weiter. Gerade dort, wo die Genehmigungsrate sehr hoch ist, ist dies möglicherweise ausreichend, wenn die Auswirkungen von zu Unrecht genehmigten Prozessen nicht gravierend sind.

Hierzu ein konkretes Beispiel [**?**]: Die Auslandsdienstreisen von Landesbediensteten im Bundesland Steiermark (Ö) müssen von der Landesregierung (!) genehmigt werden. Der Landesrechnungshof ermittelte, dass ein bis zwei Anträge pro Jahr abgelehnt werden und schlägt die genannte Änderung vor - Information statt Genehmigung. Die Änderung wurde nicht umgesetzt.

Kommen wir nun zu den Optimierungen innerhalb einzelner Tasks.

4.2.5 Optimierung von Tasks

Die Optimierung der Bearbeitung eines elementaren Tasks kann vor allem bei der Formularbearbeitung und Applikationsintegration stattfinden:

a) Formularbearbeitung

Bei der Bearbeitung von Formularen ergeben sich folgende Optimierungsmöglichkeiten:

1. Unnötige Daten vermeiden: Welche Felder sind änderbar, aber in keinem Schritt Mussfelder? Welche Felder werden in nachfolgenden Systemen bzw. im Endprodukt des Prozesses nicht verwendet? Beide Arten von Feldern sind Kandidaten für eine ersatzlose Streichung.

2. Automatische Übernahme von Daten aus anderen Systemen bzw. anderer Prozessinstanzen.

3. Vorausfüllen mit sinnvollen Defaults: Defaultwerte sind z.B. der eigene Name in einem Antragsformular oder die Anzahl Eins in einem Bestellformular.

4. Auswahl statt Eingabe: Einen Wert auswählen ist meist schneller und einfacher als einen Wert eingeben. Die Anzahl der Auswahlmöglichkeiten sollten auf das Minimum beschränkt werden (unter Ausnutzung der Kontextinformation, z.B. Befüllung anderer Felder). Werte nur auswählen statt eingeben zu lassen erhöht außerdem die Datenqualität bzw. vereinfacht die Prüfung der Daten.

In Abb. 4.11 ist ein Formular dargestellt, dass von 46 Feldern nur 4 Freitext-Felder hat, die restlichen Felder werden entweder aus anderen Systemen übernommen oder die Werte können aus Wertelisten ausgewählt werden.

b) Unstrukturierte Daten strukturieren

Das Bearbeiten von unstrukturierten Daten dauert erheblich länger als das Bearbeiten von strukturierten, d.h. Formularen. Außerdem ist die Qualität besser, da auf Muss- und Kann-Felder hingewiesen wird, Ausführungshinweise direkt bei Feldern gegeben werden etc. Deshalb gibt es bei unseren Behörden so viele Formulare! Es sollte daher tunlichst vermieden werden, dass Dokumente mit unstrukturierten Daten erstellt werden. Zeit und Mühe kosten:

- Der Aufruf der Werkzeuge (MS-Word, MS-Excel etc.)
- Das Zusammensammeln der Daten - oft aus den Prozessformularen
- Die Formatierung: Jahre an Arbeitszeit werden mit der Formatierung von Word-Dokumenten verschwendet.
- Je nach BPM-System: Die Einbindung des Dokuments in den Workflow

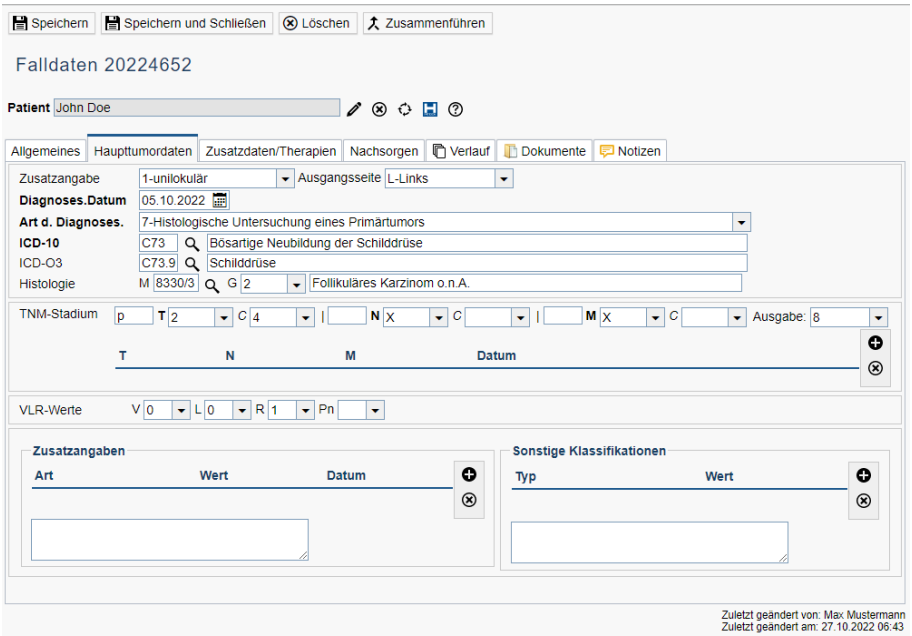


Abbildung 4.11: Optimiertes Formular

Besonders häufig trifft man auf die manuelle Bearbeitung eines Dokuments, wenn am Ende des Prozesses ein Bescheid oder Brief erstellt werden soll. Eine automatische Generierung des Briefes aus den Prozessdaten ist sicher effizienter.

Diskussionswürdig ist die Frage, ob dem Bearbeiter die Möglichkeit gegeben werden sollte, einen solchen automatisch erstellten Brief nachzubearbeiten. Sofern es sich um Standardschriftstücke handelt, ist davon abzuraten: Es wird zu oft an der Formatierung herumgespielt - siehe oben.

c) Teilaufgaben oder Schritte automatisieren

Hier liegt ein weiteres großes Optimierungspotential durch den Einsatz von BPMS. Im Kapitel 3.8, Applikationsintegration, wurde darauf bereits eingegangen. Kandidaten für die Automatisierung sind vor allem die folgenden Aufgaben:

- Übertragung von Daten zwischen IT-Systemen
- Weitergabe von Informationen an Personen außerhalb des BPMS, z.B. durch Versenden einer Email
- Prüfung von Bedingungen, Beispiele:
  - Beantragung einer Internet Leitung: Ist das technisch bei dieser Adresse möglich?
  - Beantragung eines Gewerbescheins: Hat der Antragsteller die nötigen Voraussetzungen?
  - Eröffnung eines Kontos: Wie ist die Bonität des Kunden?

  Viele dieser Prüfungen können entweder vollständig automatisiert werden oder es können alle relevanten Informationen in einem Formular gesammelt werden, sodass die Entscheidung durch einen Bearbeiter effizient erfolgen kann.

Übrig bleiben im Prozess die produktiven Schritte, wo *neue* Daten erfasst werden, Entscheidungen getroffen und extern (nicht am Computer) durchgeführte Tätigkeiten dokumentiert werden.

4.2.6 Umsetzung der Optimierung

Die Änderungen werden in neue Versionen von Prozessen, Tasks, Formularen und Integrationsprogrammen eingearbeitet und freigeschaltet.

Bei der Prüfung der Ergebnisse sind wir wieder bei der Auswertung der Laufzeitdaten - dem Beginn dieses Kapitels - angelangt. Es lassen sich nun die Durchlaufzeiten der Prozesse und die Bearbeitungszeiten der einzelnen Tasks miteinander vergleichen. Da meist bei einem Versionswechsel eine Vielzahl von Einzeländerungen durchgeführt wird, ist eine direkte Messung der Ersparnis, die eine einzelne Maßnahme gebracht hat, nicht möglich.

Genauso wichtig ist somit das Feedback, das die Benutzer und die Kunden des Prozesses geben. Welche Änderungen wurden positiv aufgenommen, welche nicht. Hier ist es nötig, auf die bewährten Analysetechniken der Software Ergonomie zurückzugreifen: Fragebogen, Beobachtung, Videoanalyse. Denn letztlich hängt die Performance maßgeblich von der Zufriedenheit der Endbenutzer und Kunden ab. Diese haben bei der Frage, ob sich eine Änderung positiv oder negativ auswirkt, berechtigterweise das letzte Wort.

Kapitel 5

Durchführung von Workflowprojekten

Die Durchführung von Workflowprojekten unterscheidet sich nicht grundsätzlich von der anderer Softwareprojekte. Mathias Weske und andere stellen in [?] ein Referenzmodell zur Durchführung von Workflowprojekten vor, das aber besonders auf die Spezifika solcher Projekte eingeht. Wir folgen, mit einigen kleinen Abweichungen, im Wesentlichen diesem Modell. Der Prozess in Abb. 5.1 zeigt die einzelnen Phasen der Projektdurchführung.

Nicht explizit modelliert sind die Pfade des Zurückgehens von einem Schritt zum vorigen, z.B. von Testphase zu Implementierung. Diese „Iterationen“ sind in jedem Abschnitt des Prozesses möglich.

5.1 Bestandsaufnahme

Der erste Schritt ist die Bestandsaufnahme, an dessen Beginn die Identifikation des Problems steht, das mit einer Workflowapplikation gelöst werden soll.

Zur Problemidentifikation soll die folgende Checkliste dienen:

1. Name der Applikation
2. Beschreibung der Prozesse
3. Beteiligte (Abteilungen, Personen)

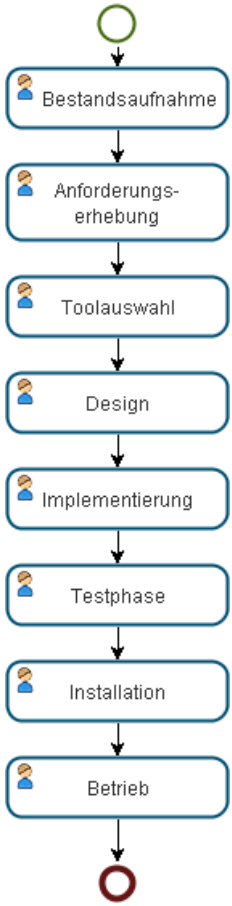


Abbildung 5.1: Entwicklungsprozess

4. Schnittstellen zu anderen Applikationen
5. Schnittstellen zu Personen oder Applikationen außerhalb der Organisation
6. Welchen Nutzen soll das Projekt haben
   - ❑ geringere Durchlaufzeit
   - ❑ Automatisierung von Teilschritten
   - ❑ bessere Nachvollziehbarkeit und Auswertbarkeit
   - ❑ Verbindung von Legacy Applikationen

- ❑ Erfüllung gesetzlicher Bestimmungen
- ❑ Erhöhung der Qualität (weniger Fehler, vollständiger, ...)
- ❑ Anderes

7. Geplanter Einsatztermin
8. Kostenschätzung
9. Nutzenschätzung

In einem Kick-off Meeting kann man auf der Basis eines solchen Identifikationsformulars die prinzipielle Entscheidung über die weitere Verfolgung des Projekts fällen.

Kosten und Nutzen können natürlich nur grob geschätzt werden, aber als Ergebnis des Kick-off Meetings sollten die drei folgenden Fragen geklärt sein.

1. Ist die Workflow-Technologie für die Problemlösung adäquat?
2. Rechtfertigt der Nutzen die Durchführung des Projekts?
3. Können die notwendigen Ressourcen zur Verfügung gestellt werden?

Auf die Beantwortung der Frage 1 wurde bereits in Kapitel 1 (Seite 12) eingegangen. Der Sinn der beiden anderen Fragen dürfte klar sein. Können alle drei Fragen positiv beantwortet werden, kann ein Projektteam bestimmt und mit der Erhebung der Anforderungen begonnen werden.

5.2 Anforderungserhebung

Ziel dieser Phase ist die Erstellung eines Lastenhefts. Dies beschreibt im Kern, wie die Prozesse in Zukunft ablaufen sollen, umfasst aber auch alle anderen Aspekte des zu erstellenden IT-Systems.

Die folgende Gliederung enthält die wesentlichen Punkte, die im Lastenheft enthalten sein sollten:

1. Allgemeines
   1.1. Zweck
   1.2. Benutzerkreis
2. Prozesse
   2.1. Prozess 1

2.1.1. Definition des Ablaufs
2.1.2. Prozessstart: wodurch ausgelöst
2.1.3. Eskalationen / Ausnahmebehandlungen
3. Tasks
3.1. Task 1
3.1.1. Akteure
3.1.2. Pre- und Postconditions
3.1.3. Kompensationen
3.1.4. Eskalationen
4. Systemschritte
4.1. Systemschritt 1
5. Formulare
5.1. Gesamtschema der Applikation
5.2. Formular 1
5.2.1. Beschreibung der Felder mit Datentypen und Wertebereichen
5.2.2. grafische Darstellung
5.2.3. Bearbeitungshilfen
5.2.4. Feldsichtbarkeiten in Prozessschritten
5.3. Wertelisten
5.3.1. Werteliste 1
5.3.1.1. Verwendung
5.3.1.2. Wertebereiche
6. Funktionen
6.1. Funktion 1
6.1.1. Funktionalität
6.1.2. Berechtigungen
6.1.3. Platzierung im GUI
7. Timer
7.1. Timer 1
7.1.1. Funktionalität
7.1.2. Intervall
8. GUI
8.1. generelle Navigation
8.2. Spezifika pro Benutzergruppe
8.3. Berechtigungen
8.4. Anpassungen der Worklist
8.5. Anpassungen im DMS

8.6. Desktop Integration / Werkzeuge
9. Reports
9.1. Report 1
9.1.1. Berechtigungen
9.1.2. Art der Darstellung (HTML, Grafik, pdf etc.)
10. Systemintegration
10.1. System 1
10.1.1. Art des Systems
10.1.2. Kommunikationsprotokoll
10.1.3. Beschreibung der Interaktion
10.1.4. Fehlerbehandlung
11. Termine

Diese Gliederung lehnt sich an den diesbezüglichen Vorschlag der IEEE [**?**] an. Auf einige dieser Punkte möchten wir nun näher eingehen. Im Kapitel „Allgemeines“ der obigen Gliederung wird das Wesentliche aus der Problemidentifikation wiederholt, eine kurze Beschreibung des Projekts und sein Nutzen sowie die Beteiligten.

5.2.1 Prozessbeschreibungen

Der Kern der Anforderungen sind die Prozessbeschreibungen. Je nach Organisationsgrad des Unternehmens werden mehr oder weniger detaillierte und vollständige Informationen über den Ablauf existieren. Auf jeden Fall ist es notwendig, die vorgefundenen Daten zu überprüfen, d.h. die am Prozess beteiligten Personen zu interviewen.

Um den Ablauf und die einzelnen Tätigkeiten zu verstehen, empfiehlt es sich mit den Daten „mitzulaufen“, d.h. eine konkrete Prozessinstanz von Bearbeiter zu Bearbeiter zu verfolgen und dabei für jeden Schritt zu ermitteln:

- wer bekommt das, gibt es Alternativen zu diesem Schritt,
- was tut der damit,
- warum tut er es,
- welche Daten werden benötigt, welche erhoben und dem Prozess hinzugefügt,
- wie werden die Ergebnisse im weiteren Ablauf berücksichtigt.

Das Ergebnis sollte eine Prozessbeschreibung sein, die mit den Prozessbeteiligten akkordiert ist.

Als weiterer Input für Prozessbeschreibungen dienen gesetzliche Bestimmungen oder allgemeine Richtlinien, die den Prozessablauf beeinträchtigen können. Beispiele sind Aufbewahrungspflicht, Dokumentationspflicht, Vier-Augen-Prinzip.

Zu beachten sind weiters Ausnahmefälle: Was ist, wenn nicht alle Daten rechtzeitig vorliegen oder wenn es besonders schnell gehen muss. Eventuell müssen vorhersehbare Ausnahmefälle als Prozessvarianten eingebaut werden.

Der Prozess kann auch in dieser Phase bereits mit @enterprise modelliert werden - dies bringt in späteren Phasen die Ersparnis, dass der Prozess nicht von einem System in ein anderes übertragen werden muss.

Neben der Definition der Abläufe sind weitere Aspekte zu beachten: Prozessstart: Wodurch wird der Prozess gestartet? Manuell, oder durch ein externes Ereignis? Oder regelmäßig zu einem bestimmten Zeitpunkt?

Tasks: Teil der Prozessbeschreibung ist die Beschreibung der einzelnen Schritte. Für die manuellen Schritte ist folgendes festzulegen:

- ❑ Akteur: Wer führt den Schritt aus.
- ❑ Pre- und Postconditions: Aktionen, die beim Start des Schrittes und bei seiner Beendigung durchgeführt werden.
- ❑ Kompensationen: Aktionen, die durchgeführt werden, um einen Schritt rückgängig zu machen.
- ❑ Eskalationen: Behandlung von Ausnahmen und Zeitüberschreitungen.

Systemschritte: Für die automatisch ablaufenden Schritte sind die aufzurufenden Programme zu benennen und welche Daten ausgetauscht werden.

5.2.2 Prozessdaten

Neben dem Ablauf sind die Prozessdaten zu analysieren. Oft existieren Papierformulare, die die Daten enthalten. Eine andere Quelle sind angebundene IT-Systeme, deren Datenschemata man erheben kann.

Bei Papierformularen ist es wichtig, auf ausgefüllte Formulare zurückzugreifen, da man nur darin sieht, welcher Art und welchen Umfangs die Daten

sind. Ausnahmebehandlung auf Papierformularen funktioniert mit Randbemerkungen und Zusatzzetteln ja recht einfach. Im BPMS kann man zwar mit Prozessnotizen Ähnliches erreichen, doch sollte versucht werden, zum Formular gehörende Daten auch in diesem abzubilden.

Neben den Prozessformularen sind die weiteren mit dem Prozess zusammenhängenden Daten wesentlich:

- ❑ prozessrelevante Daten
- ❑ Wertelisten

Für all diese Daten sind festzulegen:

- ❑ die einzelnen Felder, ihre Datentypen und Wertebereiche,
- ❑ die Berechtigungen zum Ansehen und Ändern, und zwar bis auf Feldebene im jeweiligen Kontext,
- ❑ die Art der Verwaltung: Werden die Daten aus einem anderen System übernommen, referenziert oder manuell im BPMS verwaltet.

5.2.3 Funktionen

Neben den Standardfunktionen sind für gewöhnlich weitere Funktionen zu implementieren. Deren Aufgabe muss beschrieben werden, ebenso die Art der Einbindung in das GUI und welche Benutzer die Berechtigung haben, diese Funktionen auszuführen.

5.2.4 Timer

Bei den zeitgesteuerten Ereignissen ist anzugeben, wann diese ausgelöst werden (Intervall) und welche Aktionen auf welche Daten oder Prozesse durchgeführt werden.

5.2.5 GUI

Ein wesentlicher Bestandteil der Anforderungen an ein IT-System ist die Benutzeroberfläche.

Bei BPM-Projekten nimmt die Definition der Benutzeroberfläche einen nicht so zentralen Punkt ein, da bereits durch das BPM-System vieles vorgegeben ist.

Wesentliche Aspekte des GUIs sind außerdem bereits bei den Funktionen und Formularen behandelt.

Es sind also im Wesentlichen die Anpassungen am Standard-GUI zu definieren. Dabei ist zu entscheiden, ob für die unterschiedlichen Benutzergruppen verschiedene GUI Konfigurationen (wie in Abschnitt 3.9.1 dargestellt) zum Einsatz kommen oder ob eine Grundkonfiguration reicht - diese kann dann mit dem Rechtesystem angepasst werden (Ein- und Ausblenden von Links und Funktionen je nach Rollenzugehörigkeit).

Weitere GUI-Aspekte sind die Anpassung der Arbeitskörbe und der Tabellen und Funktionen des Dokumentenmanagementsystems.

5.2.6 Reports

Die Beschreibung der Reports umfasst: Welche Parameter, was ist das Ergebnis, was ist das Ausgabeformat, wer darf den Report ausführen, gibt es Verknüpfungen zu anderen Reports.

5.2.7 Integration

Der letzte Bereich der Anforderungserhebung sind die Integrationen zu anderen IT-Systemen. In Kapitel 3.18 wurden die unterschiedlichen Integrationen vorgestellt. In den Anforderungen sind die Systeme zu nennen und die Art der Integration: Einmal-Datenübernahme, regelmäßige Übergabe, Echtzeitzugriff, Fehlerbehandlungen etc.

5.2.8 Termine

In dieser Phase kann noch kein Zeitplan erstellt werden, wohl aber zeitliche Constraints definiert werden, z.B. „muss mit 1.1.2012 in Betrieb gehen“ oder „Testbetrieb in den Sommermonaten“.

5.3 Produktauswahl

In Weskes Arbeit [**?**] steht die Produktauswahl nach der Design-Phase. Damit ist sichergestellt, dass man die Produktauswahl erst trifft, wenn man genau weiss, was man braucht. Wir möchten die Auswahl trotzdem voranstellen,

damit die Designphase und das Prototyping bereits mit dem ausgewählten Tool vorgenommen werden kann.

Die Kriterien der Produktauswahl hängen vom Profil der Anforderungen ab, trotzdem versuchen wir in der Folge einen kleinen Kriterienkatalog aufzustellen, der als Ausgangspunkt für einen eigenen Katalog herangezogen werden kann [**?**].

Die Gliederung auf erster Ebene entspricht weitgehend der Kapitelstruktur dieses Buches: Modellierung, Ausführung, Integration, Reporting und Analyse, Allgemeines. Nur der letzte Punkt weicht also vom Bisherigen ab, da er allgemeine Punkte zur Beschaffung von IT-Systemen enthält.

1 Modellierung

+20pt

1.1. Prozessdefinition

+20pt

- Prozessdefinitionssprache: In welcher Sprache werden Prozesse definiert?
- Falls BPMN, welche Sprachelemente werden unterstützt?
- Möglichkeiten und Art der Prozessversionierung
- Prozessdokumentation: Gibt es eine - ausdruckbare - Prozessdokumentation?
- Enthält diese alle im System definierten Details? Wie sieht der Endbenutzer eine Prozessdefinition?
- Grafischer Prozessdesigner: Ist die Prozessdefinition grafisch möglich?
- Wie werden Bedingungen, Postconditions, Constraints definiert?
- Prozessdefinition übers Web: Ist die Prozessdefinition übers Web mit einem Browser möglich?
- Welche Zusatzkomponenten sind dafür ggfls. erforderlich?
- Ist die Prozessdefinitionskomponente mit der Laufzeitumgebung integriert oder getrennt davon?
- Wie sieht die API Integration zur Prozesssteuerung aus?

+20pt

1.2. Datendefinition

+20pt

- Enthält das System einen Formulardesigner?
- Welche Datentypen werden unterstützt?
- Welche Eingabeprüfungen sind vorhanden, wie können diese definiert und erweitert werden?
- Wie können Abhängigkeiten zwischen Eingabefeldern definiert werden?
- Wie lassen sich 1:n und n:m Relationen abbilden?
- Welche Layouts werden unterstützt (beliebig, Tabelle, zwei- oder mehrspaltig)?
- Können Formulare außerhalb des Designers nachbearbeitet werden?

+20pt

+20pt

1.3. Organisationsdefinition

+20pt

- Welche Entities enhält das Organisationsmodell? Gibt es ein Schema?
- Welche Vertretungsregelungen gibt es: Rollenvertretungen, Mehrfachvertretungen, zeitlich eingeschränkte Vertretungen?
- Gibt es einen Organisations-Browser?
- Gibt es hierarchische Strukturen: für Organisationseinheiten? für Rollen (eine Rolle schließt eine andere ein)? für Rollenzuordnungen (wirksam in OE und Unter-OEs)?

+20pt

2 Ausführung

+20pt

2.1 Welche Laufzeitflexibilität wird angeboten

+20pt

- Zurückgehen
- Delegieren
- Kopie an?
- Ad-hoc Erweiterungen, Umfang der einfügbaren Kontrollstrukturen?

+20pt

2.2 Eskalations- und Ausnahmebehandlung

+20pt

- Welche Zustände / Ereignisse können eine Eskalation auslösen?
- Was sind die möglichen Eskalationsaktionen?
- Wie erfolgt die Fehlerbehandlung bei Aufruf externer Applikationen?

+20pt

2.3 Benutzerschnittstelle

+20pt

- Ist der Client im Browser verfügbar? Welche Browser werden unterstützt?
- Sind Plugins erforderlich?

- Wie ist die Benutzerschnittstelle anpassbar? Konfigurierbar, per API?
- Können die Spalten im Arbeitskorb frei definiert werden?
- Kann der Arbeitskorb frei strukturiert werden (Tagging, Unterordner)?
- Können für ein Formular unterschiedliche Sichten definiert werden?

Wie geschieht dies?
- Können für unterschiedliche Benutzergruppen (Rollen) unterschiedliche Layouts konfiguriert werden?
- Wie erfolgt die Anpassung an Corporate Layouts?
- In welchen Sprachen ist die Benutzerschnittstelle verfügbar?
- Wie können Prozesse, Formulare und andere Prozessbestandteile internationalisiert werden?
- Wie sieht der Benutzer den bisherigen Ablauf?
- Ist es möglich einen Prozess abzubrechen und später wieder fortzusetzen?
- Wie werden Änderungen protokolliert? Ist es möglich Änderungen auf Feldebene nachzuvollziehen?
- Ist die Administration über Web-Oberfläche möglich?

- Gibt es ein Interface für mobile Clients?
- Ist dies eine Web-Applikation oder eine mobile App (für welche Betriebssysteme geeignet)?
- Ist das mobile GUI anpassbar, wie?
- Welche Einschänkungen hat das mobile GUI?

- Gibt es eine kontextsensitive Online-Hilfe?
- Welche Handbücher liegen vor?

+20pt

2.4 Sicherheit

+20pt

- Welche Authorisierungsmechanismen gibt es? Username/Passwort, Zertifikat, Chipkarte?
- Ist Single-Sign-On möglich und in welchen Umgebungen?
- Ist das Berechtigungssystem rollenbasiert?
- Können Berechtigungen OE-bezogen vergeben werden?
- Können Berechtigungen auf einzelne Prozesse, Dokumente oder Ordner vergeben werden?
- Können mit dem Berechtigungssystem eingeschränkte Systemrechte vergeben werden, z.B. für Operator, Benutzeradministrator, Prozessverantwortliche?

+20pt

+20pt

2.5 Dokumentenmanagement

+20pt

- Können Dokumente als Attachments an Prozesse angehängt werden?
- Existiert eine prozessunabhängige Dokumentenablage?
- Unterliegt der Zugriff auf die Dokumente dem Berechtigungssystem?
- Können die Dokumente mit beliebigen Metadaten versehen werden?
- Wie sieht die Anbindung von Scannern aus?
- Welche Versionierungsmöglichkeiten gibt es?
- Wie werden Dokumente archiviert?
- Können Dokumente verschlüsselt gespeichert werden?
- Ist eine elektronische Unterschrift auf Dokumenten möglich?

+20pt

2.6 Reporting und Analyse

+20pt

- Existiert eine integrierte Reporting-Komponente?
- Können damit Reports erstellt werden, die Laufzeitdaten verdichten?
- Gibt es ein adäquates User Interface zur Erstellung der Reports?
- Können die Reports Anwendungsdaten (Formulardaten) enthalten?
- Welche Ausgangsformate gibt es (HTML, PDF, Excel, CSV, Chart)?
- Gibt es die Möglichkeit Reports zu verknüpfen (drill-down)?

+20pt

3 Integration

+20pt

3.1 Enterprise Application Integration

+20pt

- Auf welche Weise können Fremdapplikationen integriert werden?
- Welche Formen der E-Mail und Social Software Integration gibt es?
- Gibt es einen Notification Mechanismus (Benachrichtigung der Benutzer über Ereignisse)?
- Wie kann die Synchronisation der Organisationsdaten erfolgen?

+20pt

3.2 Architektur

+20pt

- Skalierbarkeit: Wie wird bei großer Last ausreichende Performanz

gewährleistet (Cluster, Lastverteilung etc.?)
- Gibt es Richtlinien für die Ausstattung der Hardware?
- Welche Datenbankmanagementsysteme werden unterstützt?
- Welche Softwarevoraussetzungen müssen vorhanden sein (Plattformen)?
- Welche Standards werden unterstützt?

+20pt

3.3 API

+20pt

- Gibt es ein API?
- Wie ist dieses dokumentiert?
- In welchen Sprachen kann programmiert werden?
- Ist die Verwendung der Engine embedded möglich?

+20pt

4 Allgemeines

+20pt

4.1 Lizenzen und Kosten

+20pt

- Welche Lizenzmodelle werden angeboten? Named User, Concurrent User, pay per use, Unternehmenslizenz?
- Ist ein Test-Environment in den Lizenzkosten abgedeckt?
- Wie sehen Umfang und Kosten der Wartungsleistungen aus?
- Welche Service Level Agreements sind möglich?
- Finanzielle Stabilität des Anbieters?
- Seit wann ist das Produkt am Markt?
- Anzahl Installationen und Benutzer?
- Referenzinstallationen?
- Können Ressourcen zu Consulting und Implementierung zur Verfügung gestellt bzw. am Markt besorgt werden? Zu welchen Kosten?

+20pt

4.2 Betrieb

+20pt

- Wie werden Upgrades der Software durchgeführt?
- Wie werden Upgrades von Applikationen durchgeführt?
- Import/Export Schnittstellen für Prozesse und weitere Artefakte?
- Gibt es dazu automatisierte Verfahren?

- Art, Kosten und Dauer von Schulungen für Endbenutzer, Prozess-Designer, Administrator, Entwickler?

Neben diesen allgemeinen Kriterien gibt es in der Regel auch projekt- und kontextspezifische zusätzliche Kriterien, die in einer Auswahl zu berücksichtigen sind. Weiters enthält die obige Aufstellung keinerlei Gewichtung, d.h. Bewertung, welche Kriterien als wichtig oder weniger wichtig einzustufen sind. Eine Aufteilung in Muss-Kriterien und Soll-Kriterien ist ebenfalls sinnvoll, um nicht passende Produkte von einer Auswahl auszuschließen.

5.4 Design und Implementierung

Die nächsten beiden Phasen - Design und Implementierung - wollen wir gemeinsam behandeln, da sie ineinander übergehen können und nicht strikt hintereinander abgearbeitet werden müssen. Diese Art des agilen bzw. prototypischen Vorgehens kann deutliche Vorteile hinsichtlich Zeitbedarf, Qualität und Benutzerbeteiligung bringen.

In der Designphase werden die einzelnen Komponenten des Systems spezifiziert, d.h. deren Verhalten wird möglichst genau bzw. formal beschrieben. In der Implementierungsphase wird der Programmcode erstellt.

Die Komponenten der Workflowapplikationen sind zum Teil Definitionen von Prozessen und deren Bestandteilen, zum Teil Programme, die in einer Programmiersprache abgefasst werden.

Ersteres ist mit der Definition fertig, nur für den Programmcode ist eine gesonderte Implementierungsphase erforderlich. Der Übergang zwischen beiden ist aber fließend, betrachten wir die Beispiele:

- ❑ Prozessbeschreibung in BPMN
- ❑ Definition einer Bedingung in XPath
- ❑ Definition einer Postcondition in Java

Der Unterschied zwischen den drei Beispielen liegt in der Allgemeinheit der verwendeten Sprache. BPMN ist für die Definition von Geschäftsprozessen,

XPath für die Navigation in XML Dokumenten und die darauf basierende Formulierung von mathematischen Ausdrücken gedacht und Java ist eine allgemeine Programmiersprache. Die Erstellung von Software aus Modellen in domain-spezifischen Sprachen bezeichnet man als *Modellgetriebene Softwareentwicklung* [**?**]. In diesem Ansatz entfällt der Implementierungsschritt weitgehend.

Bei der Erstellung der Applikation werden also für jede Komponente die Spezifikation durchgeführt und danach - falls erforderlich - die Implementierung dieser Komponente.

Die Ergebnisse dieser Phase sind zweierlei: Einerseits ein Designdokument, das in der Gliederung dem Lastenheft folgt, nun aber alle Details für die Implementierung enthält. Andererseits ist das Ergebnis eine Implementierung des Prozesses oder der Prozesse, die im nächsten Schritt bereits getestet werden kann.

5.5 Testphase

In der Testphase wird ermittelt, ob das Programm reif für den Einsatz ist, eventuell auftretende Fehler oder Abweichungen von der Spezifikation werden behoben. Für gewöhnlich besteht die Testphase aus zwei Teilen, dem sogenannten *Labortest* und dem *Feldtest*.

5.5.1 Labortest

Der Labortest (engl.: Lab Test) kann beginnen, wenn wesentliche Teile der Implementierung abgeschlossen sind.

Die Tests werden anhand eines *Testplans* ausgeführt, wobei beim Testen einer Workflowapplikation die zu testenden Prozesse bereits eine gewisse Struktur vorgeben.

Weiters muss eine Test-Organisationsstruktur aufgebaut werden, die Benutzer mit den in den Prozessen benötigten Rollen enthält.

Der Tester arbeitet den Prozess ab, indem er sich abwechselnd als eine Person einloggt, welche die für den aktuellen Schritt nötige Rolle hat. In mehreren Durchläufen pro Prozess werden die Varianten des Prozesses durchgespielt, bis alle Pfade begangen wurden.

In jedem Prozessschritt ist die Übereinstimmung mit der Spezifikation zu prüfen:

- Funktionen: Welche Funktionen stehen zur Verfügung - und machen sie das Richtige.
- Formulare: Stimmen die Sichtbarkeiten, sind alle nötigen Informationen vorhanden und sichtbar, die richtigen Felder änderbar und die Mussfelder richtig gesetzt.
- Bei allen Eingabefeldern sind Fehleingaben zu prüfen: Nicht zulässige Eingaben sollen mit sinnvollen Fehlermeldungen beantwortet werden.
- Übereinstimmung mit der Dokumentation.

Treten Fehler auf, werden diese an das Entwicklungsteam gemeldet und von diesem behoben. Eine neue Version der Applikation wird am Testsystem zur Verfügung gestellt.

Am Abschluss der Phase steht ein *Testprotokoll*, das Aufschluss darüber gibt, welche Testfälle getestet wurden, welche Fehler erkannt und behoben wurden und welche Fehler offen bleiben. Ist der Labortest erfolgreich, kann der Feldtest beginnen.

5.5.2 Feldtest

In dieser Phase werden die Prozesse von einer ausgewählten Teilmenge der Endbenutzer getestet.

Bevor der Feldtest (engl.: Field Test) beginnen kann, muss mit den Benutzern eine Schulung durchgeführt werden. Dabei werden die Funktionen der Applikation vorgestellt und die Benutzer haben die Möglichkeit, Testfälle durchzuspielen.

Der Feldtest ist nicht für das Aufdecken von Programmfehlern gedacht sondern zur Findung von Unzulänglichkeiten in der Applikation, dem Design, und dem Lastenheft:

- Nicht vorhandene Funktionalität: Das Durchspielen von echten Fällen führt zum Aufspüren von Funktionen, die für den Einsatz der Applikation notwendig oder hilfreich wären.

- Falsche Funktionalität: Die Umsetzung einer Funktion entspricht nicht den Anforderungen bzw. dem Bedarf. Auch hier kann der Fehler bereits im Lastenheft, im Design oder in der Implementierung liegen.
- Schlechte Bedienbarkeit: Das Bearbeiten eines einzelnen Geschäftsfalles durch einen Tester ist etwas anderes als das Abarbeiten von Dutzenden Fällen an einem Vormittag. Langwierige Klickerei für häufig wiederkehrende Funktionen ist ein Usability Problem, welches oft erst im Feldtest aufgedeckt wird.

Auch der Feldtest erfolgt in einem Zyklus, in dem nach Melden der Probleme die nötigen Entwicklungsschritte durchgeführt werden, bis eine neue Version der Applikation am Testsystem installiert wird.

Nach erfolgreichem Abschluss der Testphase (wiederum in einem Protokoll dokumentiert) kann die Applikation auf das Produktionssystem übertragen werden.

5.6 Installation

Die Implementierung eines Workflow-Projekts erfolgt normalerweise auf einem oder mehreren Entwicklungssystem(en). Von dort wird der Projektcode auf ein Testsystem übertragen. Auf diesem können in der Testphase die Tests durchgeführt werden. Sind die Tests abgeschlossen, wird das Projekt auf das Produktionssystem übertragen. Der Testserver ist in weiterer Folge nicht nutzlos sondern kann für das Nachvollziehen von in der Produktion aufgetauchten Problemsituationen genutzt werden.

Ein weiteres System kann als Schulungssystem verwendet werden. Das Testsystem sollte dafür nicht herangezogen werden, da dort möglicherweise schon die nächste Version zum Test bereitsteht.

Es ist zu beachten, dass die Installation normalerweise nicht von den Entwicklern durchgeführt wird, sondern die Software zusammen mit einer *Installationsanleitung* an den Betriebsverantwortlichen übergeben wird. Diese ist daher so abzufassen, dass keine Workflowspezifika vorkommen und weitgehend auf manuelle Eingriffe über eine graphische Benutzerschnittstelle verzichtet wird.

Die Installation erfolgt in zwei Teilen: Erstens die Installation des BPM-Systems, danach die Installation der Applikation. Bei ersterem kann oder muss man in der Regel auf die vom Hersteller vorgegebenen Prozeduren zurückgreifen.

Für die Installation der Applikation gibt es zwei Möglichkeiten, je nach Charakteristik des Zielsystems: Entweder es werden die einzelnen Schritte im Zielsystem manuell unter Verwendung des Administrations-Interfaces durchgeführt oder die Installation erfolgt über ein Script.

Wir stellen beide Varianten anhand von @enterprise vor.

Applikation installieren über GUI

Es gibt eine GUI-Funktion *Applikation installieren*, mit der eine Applikation am Server installiert werden kann. Die Applikation wird dabei in eine Archiv-Datei (zip) gepackt, in dem sowohl der Programmcode, als auch die initiale Konfiguration, sowie der XML-Export der Datenbankobjekte enthalten ist. Die Funktion macht dann folgendes: Die Dateien werden ins Zielverzeichnis ausgepackt, der XML-Import wird durchgeführt, der Pfad im Applikationsobjekt angepasst und der Klassenpfad entsprechend erweitert.

Manuell sind danach eventuell Konfigurationsparameter der Applikation anzupassen, dies ist ebenfalls aus dem Admin-GUI möglich.

Falls es aus Gründen der Policy des Betreibers nicht möglich ist, die Installation über das GUI durchzuführen, ist alternativ der folgende Weg zu beschreiten:

Administrations-Shell

@enterprise verfügt über eine Script-Console, mit der Administrationsaktionen ohne Browser über die Kommandozeile durchführbar sind. Für die Applikationsinstallation kann ein solches Script erstellt werden. Der Administrator muss dann folgende Schritte ausführen:

- ❑ Dateien ins Zielverzeichnis auspacken
- ❑ Parameter im Script anpassen
- ❑ Script mit Installationsaktionen ausführen

Das Script wird in Groovy (*http://groovy.codehaus.org/*) geschrieben, eine Script-Sprache, die auf Java basiert.

In Kapitel 12 des @enterprise 8.0 Administrationshandbuchs ist die Administrations-Shell ausführlich beschrieben [**?**]. Der Vorteil dieser Methode ist, dass der Administrator keine manuellen Eingriffe (außer den oben beschriebenen) durchführen muss, alle Schritte laufen automatisch ab, der Vorgang ist dokumentiert und nachvollziehbar.

5.7 Applikationsupgrade

Neben der Erstinstallation wird es immer wieder nötig sein, Updates zu installieren. Man kann davon ausgehen, dass sich sowohl Datenbankobjekte (Prozesse, Tasks etc.) wie auch Code geändert haben.

Das Vorgehen sieht nicht viel anders aus als bei der Erstinstallation: Für die Datenbankobjekte wird wieder der XML-Import verwendet, die geänderten Dateien werden ins Applikationsverzeichnis kopiert.

Die XML-Import Funktion erkennt bereits vorhandene Objekte und nimmt an diesen nur die nötigen Modifikationen vor. So ist es zum Beispiel möglich, in einem bestehenden Prozess einen neuen Schritt einzufügen.

Manchmal ist es im Zuge eines Upgrades nötig, weitere Aktionen, z.B. Datenbankoperationen, durchzuführen. Ein Beispiel wäre das Verändern von Rechtezuordnungen zu Rollen.

Diese Aktionen können in einer Java-Methode implementiert werden, welche bei der Applikation als Upgrade Methode eingetragen wird. Dann können diese Aktionen aus dem Administrations-GUI durchgeführt werden. Eine andere Möglichkeit ist die Erstellung eines Scripts für die Administrations-Shell.

Die Schritte, die der Administrator durchführen muss, sind dann:

- ❑ Benutzer über die Systemunterbrechung informieren,
- ❑ Login deaktivieren,
- ❑ Server stoppen,
- ❑ Dateien ins Zielverzeichnis kopieren,
- ❑ Server starten,
- ❑ Script mit Upgradeaktionen ausführen,
- ❑ Login aktivieren.

Die Verwendung eines Admin-Scripts macht die Upgrade-Aktionen wiederholbar und dokumentiert. Mit dem Aufrufparameter `-log` kann ein Protokoll der Ausführung auf eine Datei geschrieben werden.

Mit der Inbetriebnahme des Systems ist der Entwicklungsprozess abgeschlossen. Abb. 5.7 gibt einen Überblick über die verschiedenen im Entwicklungsprozess entstehenden Dokumente.

Dokument	Projektschritt
Problemidentifikation	Bestandsaufnahme
Lastenheft	Anforderungserhebung
Systemspezifikation	Design
Benutzerdokumentation	Implementierung
Administratordokumentation	Implementierung
Installationsanleitung	Implementierung
Testplan	Implementierung
Testprotokoll	Testphase

Abbildung 5.2: Dokumente, die im Entwicklungsprozess erstellt werden

Wir betrachten in diesem Kapitel abschließend noch die Aufgaben, die während des Betriebs einer Workflowapplikation anfallen.

5.8 Betrieb

Die Aufgaben während des Betriebs gliedern sich in folgende Bereiche:

- technische Administration: Betreuung der Infrastruktur
- organisatorische Administration: Wartung der Organisationsstruktur
- fachliche Administration: Kontrolle der Abläufe, Monitoring
- Fehler- und Change-Management: Organisation der laufenden Änderungen

Die Aufteilung der Aufgaben können unterschiedlich zusammengefasst werden, je nach Größe der Organisation oder wo der Betrieb erfolgt – in-House oder bei einem externen Dienstleister.

5.8.1 Technische Administration

Die technische Administration dient der Aufrechterhaltung der technischen Infrastruktur rund um das BPMS: Datenbankkapazität, Festplattenkapazität, Rechenleistung.

Der technische Administrator kümmert sich um die Server-Infrastruktur. Von der Workflowsoftware muss er nicht viel wissen, außer die Aspekte:

1. Einspielen von Updates
2. Bereitstellen von Log-Informationen für Fehlersuche
3. Kontrolle des Ressourcenverbrauchs und ob das System ordnungsgemäß läuft.
4. Durchführung und Überwachung von Backups

Das Einspielen von Updates erfolgt meist nach genauer Vorgabe der Entwickler, siehe oben. Die Punkte 2 und 3 werden in der @enterprise Administration durch die Bereiche Server-Monitor, Serversteuerung und Konfiguration abgedeckt. Im Server-Monitor lassen sich zum Beispiel der Speicherverbrauch und der Verbrauch von Datenbankverbindungen kontrollieren. Der technische Administrator loggt sich bei @enterprise normalerweise als *sysadm* ein, da dieser Benutzer der Einzige ist, der auch ohne Datenbankverbindung funktioniert.

Aufgabe 3 und 4 sind regelmäßig durchzuführen. Beim Ressourcenverbrauch ist zu beachten, dass die Auslastung von Datenbank, CPU, Filesystem, Netzwerk und Lizenzen nicht an die Systemgrenzen stößt.

5.8.2 Organisatorische Administration

Falls kein automatischer Abgleich mit einer externen Benutzerverwaltung durchgeführt wird, erfolgt die Pflege der Organisationsdaten manuell. Das heißt, neue Benutzer müssen eingetragen werden, ausscheidende Mitarbeiter müssen deaktiviert werden. Ebenso müssen Rollenzuordnungen angepasst werden. Meist sind auch bei einem automatischen Abgleich der Benutzer mit einem unternehmensweiten Verzeichnis einige dieser Aktionen manuell auszuführen, z.B. Rollenzuordnungen verwalten.

Ein weiterer Punkt sind Reorganisationen: Es kann zum Wegfall kompletter Organisationstrukturen kommen. Da Prozesse Organisationeinheiten zugeordnet sind, muss gewährleistet werden, dass bestehende Prozesse geordnet

weitergeführt werden und neue Prozessinstanzen automatisch den neuen Organisationsstrukturen folgen.

Bei (unvorhergeseher) Abwesenheit eines Mitarbeiters kann es nötig sein, Aufgaben von einem Arbeitskorb in einen anderen zu verschieben (Funktion Akteur ändern) oder Vertretungen einzutragen - ebenfalls Aufgaben der organisatorischen Administration.

5.8.3 Fachliche Administration

Neben der Kontrolle der Systemperformance, wofür der technische Administrator verantwortlich ist, ist es nötig, auch die Performance der Prozesse zu prüfen. Die Fragen, die es dabei zu beantworten gibt, sind:

- Prozess-Performance: Wie lange dauern die Prozesse? Wie ist der Trend? Soll/Ist Vergleich.
- Engpässe: Gibt es Engpässe bei einzelnen Benutzern oder Rollen?
- Ausnahmefälle: Überschrittene Deadlines, lang laufende abgeschlossene oder nicht abgeschlossene Tasks, lang laufende Batch Schritte, Prozesse ohne aktiven Akteur.

Die Fragen können mit Hilfe der Administrationsfunktionen des Systems und durch Reports beantwortet werden. Aufgrund der erhaltenen Informationen können Steuerungs-Aktionen durchgeführt werden.

Die Rolle des fachlichen Administrators kann für das gesamte BPMS oder für eine Applikation oder für einzelne Prozessdefinitionen vergeben werden. Man bezeichnet diese Rolle auch als Prozessverantwortlichen oder Process Owner.

In den Aufgabenbereich eines fachlichen Administrators fällt auch die Erstellung von Reports, einerseits um sich selbst ein Bild über das Geschehen zu machen, andererseits um dem Management verdichtete Informationen bereitstellen zu können. In der Regel werden bestehende Reports während des laufenden Betriebs weiterentwickelt und neue entworfen.

5.8.4 Fehler- und Change-Management

Während des Arbeitens mit der Applikation werden Fehler und Unschönheiten auftreten, die zu beheben sind. Auch die Prozesse unterliegen in einer lebenden

Organisation einer ständigen Veränderung: Die Vorgaben ändern sich, eventuell die gesetzlichen Grundlagen eines Prozesses. Weitere Daten müssen erfasst werden, andere sind nicht mehr nötig. Es müssen andere Systeme angebunden werden oder das System für eine neue Softwareumgebung angepasst werden. All dies sind Gründe für eine Weiterentwicklung der Workflowapplikation. Es empfiehlt sich daher, die laufend eintreffenden Änderungsnotwendigkeiten und Wünsche zu sammeln.

In @enterprise kann die Applikation IT-Service Management (ITSM) installiert werden und parallel zur Anwendung betrieben werden. Damit können Incidents (Anregungen, Wünsche, Fehler) gesammelt und Bearbeitern zugeteilt werden. Aus den Incidents werden entweder Problem-Prozesse, die zur Bearbeitung von Fehlern dienen, oder Change-Prozesse, die neue Features behandeln.

Aufgabe des Release-Managements ist es, die Probleme und Änderungswünsche zu gruppieren, zu bearbeiten und eine neue Release zu erstellen.

Der konkrete Prozess des Change Managements ist je nach Organisation sehr unterschiedlich, enthält zumindest die folgenden Phasen: Zusammenstellung der gewünschten oder notwendigen Änderungen, Implementierungsfreigabe, Implementierung, Tests, Installation.

5.8.5 BPM in der Cloud

Immer populärer ist das Auslagern des Betriebs von Anwendungen an einen Dienstleister. Die Anwendung läuft dann auf einem Server in einem Rechenzentrum und ist über das Internet zugänglich. Da man sich nicht darum kümmern muss, wo genau die Anwendung installiert ist, spricht man von Cloud Computing (ein deutscher Begriff hat sich nicht etabliert, ev. Rechnerwolke). Mehrere Varianten sind vorstellbar (vgl. [**?**]):

1. Infrastruktur (Infrastructure as a service): Der Anbieter stellt die Plattform zur Verfügung, also Betriebssystem, Datenbank, Netzwerkzugang und übernimmt die technische Administration, d.h. gewährleistet die Verfügbarkeit dieser Komponenten. Das BPMS liegt in der Hand des Kunden.
2. Plattform (Platform as a Service): Das BPMS wird ebenfalls vom Anbieter betreut, der Kunde definiert und wartet die Prozesse und Interfaces (Web-Services) zu anderen Applikationen. Die oben beschriebene technische Administration obliegt weitgehend dem Anbieter.

3. Software (Software as a Service): Eine BPM Anwendung wird vom Anbieter zur Verfügung gestellt, z.B. CRM, Personalprozesse, IT-Service Management. Eigene Daten, z.B. Organisationsstrukturen, können über vordefinierte Schnittstellen eingebunden werden.

4. Process as a Service: Die weitgehendste Form des Outsourcings, in der ganze Prozesse oder Teilprozesse von einem externen Dienstleister übernommen werden.

Betrachten wir zwei Beispiele für das Outsourcing von Prozessen: Die Prüfung von Eingangsrechnungen kann vom Einscannen der eingehenden Rechnungen bis zu Bezahlung von einem Dienstleister übernommen werden, nur der Schritt der Rechnungsfreigabe, der einfach über das Web-Interface des BPMS vorgenommen werden kann, erfolgt in-House.

Service-Management: Das Annehmen von Reklamationen, Hilfestellung bei der Bedienung oder Austausch von fehlerhaften Geräten kann ebenfalls von einem Dienstleiter übernommen werden.

Die Vorteile, die man sich durch BPM in der Cloud erhofft, ergeben sich durch das Überlassen von Aufgaben, die nicht zu den Kernkompetenzen der Organisation gehören, an Spezialisten. Der Betrieb eines Rechenzentrums, einer Datenbank, das Aktualisieren der verschiedenen Softwarekomponenten ist bei einem IT-Spezialisten besser aufgehoben als z.B. in einem Produktionsbetrieb. Demgegenüber stehen Risiken bezüglich Sicherheit und Abhängigkeit vom Cloud-Anbieter – die Problematik der Gestaltung von Outsourcing-Verträgen füllt ganze Bücher.

Geschäftsprozessmanagement mit @enterprise • Die Beschäftigung mit Geschäftsprozessen wird zunehmend zu einem zentralen Thema der Unternehmensführung. Die Anforderungen lauten: Ausrichtung des Unternehmens auf die Kernprozesse und schnelles Reagieren auf Veränderungen durch flexible Prozesse.

Mit sogenannten Geschäftsprozessmanagementsystemen (BPMS) kann die Modellierung der Prozesse, die Ausführung und die Optimierung mit IT-Unterstützung durchgeführt werden.

Dieses Buch bietet eine kompakte Einführung in dieses Thema. Die praktische Umsetzung wird anhand des BPMS @enterprise dargestellt.

Der Autor • Herbert Groiss beschäftigt sich seit mehr als 10 Jahren mit den theoretischen und praktischen Aspekten des Geschäftsprozessmanagements. Er ist Geschäftsführer der Groiss Informatics GmbH.

ISBN 978-3-85312-085-9

9 783853 120859

NOREA VERLAG