Douglas Cunningham, Petra Hofstedt, Klaus Meer, Ingo Schmitt (Hrsg.): INFORMATIK 2015
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015

Integration und Weiterentwicklung bestehender Energiemanagement-Applikationen mit dem OpenResKit-Framework

Johannes Boß[1] und Volker Wohlgemuth[2]

Abstract: Für kleine und mittlere Unternehmen gerät das Thema ressourceneffiziente Produktion vermehrt in den Fokus. Der Einsatz von Informationssystemen ist hier unabdingbar, und viele Unternehmen können bereits auf eine Datengrundlage, jedoch oftmals in unterschiedlicher Qualität, zurückgreifen. Standardprodukte können diese Daten nicht integrieren und sind für KMU zu teuer. Dieser Beitrag beschreibt den Versuch, bestehende Informationsressourcen und Anwendungen im Unternehmen zu nutzen. Am Beispiel des Energiemanagements wird einerseits eine zentrale Energiemanagementanwendung weiterentwickelt und andererseits im Unternehmen bereits vorhandene Daten hierin integriert. Die Umsetzung erfolgte mit dem an der HTW Berlin entwikkelten OpenResKit-Framework. Besonders vorteilhaft erweist sich die modulare Architektur mit einer leicht anpassbaren Datenbank und die nicht proprietären Austauschformate, um die Daten clientunabhängig verfügbar zu machen.

Keywords: BUIS, Energiemanagement, ISO 50001, OpenResKit-Framework

1 Einleitung

Kleine und mittlere Unternehmen (KMU) können mit einer ressourceneffizienten Produktion viele brachliegende Potentiale nutzen [Vd11, S. 4]. Die Motivation für ressourceneffiziente Produktion kann von unterschiedlicher Art sein. Das vordergründige Argument ist die Unterstützung des betrieblichen Umweltschutzes und die damit verbundene Verringerung der Emission von Treibhausgasen. Wirtschaftliche Vorteile sind aus unternehmerischer Sicht jedoch häufig gewichtigere Gründe. Steigende Energiekosten können durch einen sparsameren Energieverbrauch abgefedert werden. Durch die planmäßige Suche nach Verbesserungspotentialen im eigenen Energieverbrauch und die Investition in energieeffiziente Technologien sinken die Energieverbräuche im Unternehmen. Auch der Staat unterstützt Unternehmen mit verschiedenen Fördermaßnahmen. Hier ist die Befreiung von der Umlage nach dem Erneuerbare-Energien-Gesetz (EEG-Umlage) für energieintensive Unternehmen zu nennen (§ 64 Abs. 1 EEG 2014), oder die Steuervergünstigung zum Spitzenausgleich im Rahmen der Energie- und Stromsteuer, zu deren Voraussetzungen ein Energiemanagement im Unternehmen gehört (§ 10 Abs. 3 StromStG). Die Anstrengungen zur Energieeffizienz können nach außen wirksam durch eine Zertifizierung dargestellt werden. Dies kann für die öffentliche Wirkung generell

[1] HTW Berlin, Fachbereich 2, Wilhelminenhofstraße 75A, 12459 Berlin, joha.boss@gmail.com
[2] HTW Berlin, Fachbereich 2, Wilhelminenhofstraße 75A, 12459 Berlin, volker.wohlgemuth@htw-berlin.de

sinnvoll sein, oder durch einen Auftraggeber explizit gefordert werden. Die strukturierte Analyse und Verbesserung der eigenen Unternehmensprozesse, die häufig von externen Fachleuten begleitet wird, ist ein Nebeneffekt der Anstrengungen.

Ein systematischer Ansatz für eine energieeffizientere Produktion ist die Einführung eines Energiemanagementsystems, beispielsweise nach DIN ISO 50001. Die praktische Umsetzung der Dokumentations- und Überwachungspflichten erfolgt mit einem Softwaresystem. Der nächstliegende Ansatz ist die Einführung eines Standardsoftwareproduktes zur Unterstützung des Energiemanagements. In der Praxis der KMU stehen dem einige Vorbehalte entgegen, falls die hohen Anschaffungskosten überhaupt einen Kauf zulassen. Die Funktionalität vieler Produkte ist vom Umfang nicht auf KMU angepasst und überfordert die Fachabteilungen, die geringere Anforderungen an das System haben. Ein zusätzliches Produkt vergrößert die Komplexität der bestehenden IT-Anwendungslandschaft im Unternehmen [Ti13, S. 19]. Bei Standardprodukten müssen die Schnittstellen zu den besonderen Gegebenheiten in einem einzelnen Unternehmen, falls überhaupt möglich, erst konfiguriert werden [Ba09, S. 14]. Dieser Beitrag verfolgt deshalb einen Bottom-Up Ansatz, welcher versucht, einerseits vorhandene Daten im Unternehmen zu nutzen und andererseits eine bereits eingesetzte Anwendung weiterzuentwickeln. Innerhalb der Software sollen alle, das Energiemanagement betreffende Aufgaben verwaltet werden können. Geschäftsprozesse, die bisher mit Excel-Dokumenten oder veralteten Datenbanken durchgeführt wurden, werden in der neuen Anwendung abgebildet und alte Datenbestände, wenn möglich migriert. Diese pragmatische Vorgehensweise wird bei einem Partnerunternehmen validiert.

2 Grundlagen

In der Arbeit werden Elemente zweier Ansätze der Softwaretechnik zur Evolution von Softwaresystemen verwendet. Die erfolgversprechendste Option zur Abdeckung zusätzlicher IT-Bedarfe ist laut Huber, „bestehende Anwendungen durch Erweiterungen und Änderungen an neue Anforderungen anzupassen.“ [Hu14, S. 33] Dies gilt insbesondere dann, wenn es sich bei der Anwendung um eine Eigenentwicklung handelt und im Unternehmen noch ausreichend Wissen verfügbar ist. Dieser Ansatz wird auch als Softwareevolution bezeichnet [Sn12]. Die Prozesse der Informations- und Datenintegration helfen, Komplexität der Anwendungslandschaft zu reduzieren und ermöglichen es, auf die bestehenden Daten zuzugreifen.

2.1 Softwareevolution

Die Begriffe Softwareevolution, Wartung, Pflege oder Veränderungsmanagement werden uneinheitlich verwendet und sind abhängig vom verwendeten Vorgehensmodell und der Betrachtungsweise. Hier wird auf das Staged Model of the Software Lifecycle von Bennett et al. [BRW02] zurückgegriffen, das in Abbildung 1 dargestellt ist. Hier gibt es nach einer initialen Entwicklungsphase eine Evolutionsphase, in der die Software um

einen größeren Funktionsumfang erweitert wird und Anforderungen korrigiert werden. Danach folgt eine Servicephase zur Fehlerbehebung und zur Anpassung minimaler Funktionalitäten, bevor in der Abwicklungs- beziehungsweise der Beendigungsphase das Produkt stillgelegt wird.

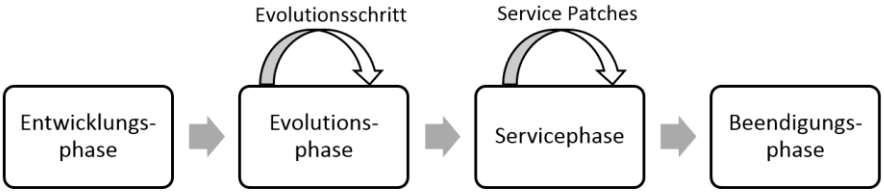

Abb. 1: Staged Model of the Software Lifecycle (vgl. Sommerville [So11, S. 236])

In [BF14] werden einige Techniken, die bei Evolutionsprojekten angewendet werden, beschrieben und nachfolgend umrissen.

- Softwareanalyse: Zunächst muss der Versuch unternommen werden, den Quellcode zu lesen und zu verstehen. Eine Dokumentation der Architektur kann hilfreich sein, ist aber oft nicht vorhanden. Es gibt einige Tools, die die Codeanalyse unterstützen. Viele Entwicklungsumgebungen verfügen mittlerweile über solche Werkzeuge.
- Reengineering: Unter Reengineering versteht man die Überprüfung und Verbesserung kompletter Komponenten. Die Funktionalität bleibt vollständig erhalten. Ziel ist, mittels einer verbesserten Organisation des Codes die Wartbarkeit zu verbessern. Davon abzugrenzen ist das Refactoring, das zwar aus gleichen Gründen durchgeführt wird, aber nur kleinere Teileinheiten des Quellcodes (Methoden, Variablen) umfasst und mit modernen Entwicklungswerkzeugen (teil-)automatisiert durchgeführt werden kann.
- Reverse Engineering: Als Reverse Engineering wird die Analyse eines Programmes mit dem Ziel einer vollständigen Neuentwickelung ganzer Komponenten bezeichnet. Gründe können die Verbesserung der Softwarequalität, die Anpassung an neue Plattformen, oder das Ersetzen von Altsystemen (Legacy Code) sein. Auch hier bleibt die Funktionalität gleich. Von steigender Bedeutung ist das Reverse Engineering von Datenbanken. Hierbei wird auf Basis einer physischen Datenbank das Datenbankschema ermittelt. Eng mit dem Reverse Engineering verbunden ist auch die Nachdokumentationen.
- Migration: Unter Migration wird die Portierung einer Software, oder Teile davon, auf andere Plattformen verstanden. Das kann zum Beispiel die Änderung einer unterliegenden Datenbank oder der Ersatz eines nicht mehr unterstützen Frameworks sein.

- Retirement: Am Ende des Softwarelebenszyklus steht das Abschalten der Software. Die Auswirkungen sind zu überprüfen, wie zum Beispiel die Zugänglichkeit von Archivdaten, und der Ersatz ist sicherzustellen. Das Auslaufen einer Software ist eng mit der Migration verknüpft.

2.2 Informations- und Datenintegration

Im Unternehmensalltag gibt es häufig eine fragmentierte, heterogene Anwendungslandschaft [Ti13, S. 71]. Die Daten liegen in unterschiedlichen Formaten, Medien und Strukturen vor und stehen für einen unternehmensweiten Gebrauch nicht zur Verfügung [Ti13, S. 136]. Konsolidierungsprojekte in den Bereichen Applikationen und Daten können helfen, die Heterogenität der Anwendungen aufzulösen, aber auch um vorhandene Daten unternehmensweit nutzbar zu machen. Das IT-Management wird so vereinfacht, die Qualität der Anwendungslandschaft erhöht und Kosten reduziert[Ti13, S. 129]. Mit der Überführung von einzelnen Datensilos zu einer möglichst unternehmensweiten Informationssicht, lassen sich Mehrfacherfassungen und Datenredundanzen vermeiden. Die Daten- und Informationsintegration lässt sich auf mehreren Ebenen unterscheiden. Jung gibt eine Übersicht über verschiedenen Definitionen [Ju06]. Die Integration lässt sich einerseits durch Verbinden oder durch Vereinigen der Datenelemente erreichen. Beim Verbinden bleiben die möglicherweise redundanten Ausgangsbestände erhalten, aber es werden logische Beziehungen zwischen den Beständen auf Anwendungsebene hergestellt. Manchmal wird diese Art auch als Interoperabilität bezeichnet [PB06]. Bei der Integration durch Vereinigen werden diese Datenbestände verschmolzen und Redundanzen eliminiert.

Weitere Begriffe zielen auf die Abstraktionsebene ab. Ein Data Warehouse wird von verschiedenen unternehmensinternen und -externen Datenquellen gespeist und bereitet die Daten auf. Geschieht die Koppelung der einzelnen Informationsquellen nicht auf Datenebene, sondern auf Anwendungsebene, kann man eher von Enterprise Application Integration (EAI) sprechen [Me12, S. 71]. Hierbei wird jedes eingebundene System mit einem Adapter an einen zentralen Bus angeschlossen. Mit dem Begriff Service-oriented Architecture wird ein ähnliches Konzept beschrieben, dass sich an Geschäftsprozessen orientiert [Me12, S. 71].

Ebene	Konzept	Techniken
Daten	Data Warehouse	Datenintegration
Anwendungen	Enterprise Application Integration	Anwendungsintegration
Prozesse	Service-oriented Architecture	Geschäftsprozessintegration

Tab. 1: Integrationsformen nach Abstraktionsebene

3 OpenResKit-Framework

Die technische Umsetzung erfolgt mit dem OpenResKit-Framework. Dies ist ein Forschungsprojekt, dass an der HTW Berlin im Studiengang Betriebliche Umweltinformatik durchgeführt worden ist. Hierbei wurden Open-Source-basierende Softwarewerkzeuge zur innerbetrieblichen Ressourcen- und Effizienzfragestellungen erstellt [Wo14]. Der Nutzerfokus liegt auf kleinen und mittleren Unternehmen, die mit einfachen und problemspezifischen Applikationen die Ressourceneffizienz in ihrem Betrieb erhöhen können. Ziel war, einen zentralen Datenpool mit einer standardisierten Schnittstelle zu schaffen. Dieser wurde in Form des OpenResKit-Hubs, hier nachfolgend auch ORK-Server genannt, realisiert. Die Plattform ist leicht erweiterbar und modular verwendbar. Innerhalb des Forschungsprojektes und weiteren Projekten sind bereits einige Softwarebausteine entstanden. Darunter ist auch eine Anwendung zum Maßnahmenmanagement, die erweitert und an die Bedürfnisse des Partnerunternehmens angepasst wurde. Bei der OpenResKit-Architektur handelt sich um eine typische zweischichtige Client-Server Architektur. Innerhalb des Forschungsprojektes sind hauptsächlich Desktop-Clients und mobile Anwendungen entstanden. Für die Desktopanwendung sind einige wiederverwendbare Komponenten vorhanden, auf denen die zu erstellende Software aufbaut. Die Integrationsfragestellung dieser Arbeit rückt allerdings den Server in den Mittelpunkt.

Der Server erfüllt zwei zentrale Aufgaben. Erstens stellt er die Domänenmodelle bereit und persistiert sie in der eigenen Datenbank, üblicherweise eine Microsoft SQL Server Datenbank (Express Edition). Dabei wird das Microsoft Entity Framework (EF) als objektrelationaler Mapper (ORM) verwendet. Dieser wandelt die Klassenstrukturen in das Datenbankmodell um und bietet eine Abstraktionsschicht, die zwischen Datenbank und Anwendung vermittelt. Zweitens wird eine Webschnittstelle zum Datenaustausch, nach dem OData-Protokoll[3], angeboten, um die Clients anzubinden. Besonders erwähnenswert ist das dynamisch anpassbare Domänenmodell, dass für die große Flexibilität und Modularität verantwortlich ist. So können zur Laufzeit neue Module hinzugefügt werden. Die Datenbank reagiert mit einer entsprechenden Schemaänderung und einer automatischen Datenbankmigration. Bei der Umsetzung wurde dabei auf das Managed Extensibility Framework (MEF) zurückgegriffen. Diese Bibliothek ist Teil des Microsoft .NET-Frameworks und ermöglicht die Entwicklung erweiterbarer Anwendungen, ohne eine Konfiguration vornehmen zu müssen oder fest codierte Abhängigkeiten zu berücksichtigen. Somit lassen sich unbekannte, kompilierte Erweiterungen einbinden. Der Anwender hat damit eine einfache Möglichkeit das Datenmodell anzupassen und zu erweitern, was den Prozess der Datenintegration beschleunigt.

[3] http://www.odata.org/

4 Konzept

Basis der Softwarelösung ist der bestehende OpenResKit-basierte Client zum Maßnahmenmanagement und das dazugehörige Servermodul. Innerhalb des Partnerunternehmens, einem KMU aus der Kunststoffverarbeitung, werden im Bereich Umwelt- und Energiemanagement und der damit eng verknüpften Betriebstechnik im Wesentlichen die nachfolgend genannten Anwendungen eingesetzt. Die Anlagen und damit Energieverbraucher werden in einer MySQL-Datenbank mit verschiedenen Zugriffsclients verwaltet. Ein Raumbuch wird in einer lokalen Microsoft Access Datei geführt. Zusätzlich werden einige Auswertungen mithilfe von e!Sankey und einer damit verknüpften Microsoft Excel Tabelle gemacht. Die Anforderungen des Partnerunternehmens sehen vor, sämtliche Aufgaben im Rahmen des Energiemanagements nach DIN ISO 50001 innerhalb einer Anwendung zu bearbeiten. Deshalb sollen die vorher genannten Anwendungen in abgelöst und die Funktionalität in die OpenResKit-Anwendung Maßnahmenmanagement übertragen werden. Dies betrifft insbesondere die Informationen der Energieverbraucher und eine Raumdatenbank aus der Betriebstechnik. Zusätzlich sind zu den Energieverbrauchern Messwerte und Ablesungen zu speichern. Die zusätzlichen Informationen sollen auch für eine zeitbezogene Energieverbrauchsauswertung innerhalb der Anwendung genutzt werden, die die geschätzten Energieeinsparungen mit den tatsächlichen Betriebswerten vergleicht. Die lokale MySQL-Datenbank wird für diesen Bereich nicht mehr weitergeführt. Allerdings muss die Möglichkeit bestehen, weiterhin auch außerhalb des Clients auf die Datensätze zuzugreifen, und auch dort Veränderungen vorzunehmen. Die Daten sollen auch für weitere Anwendungen, zum Beispiel für die Sankey-Auswertungen, zur Verfügung stehen.

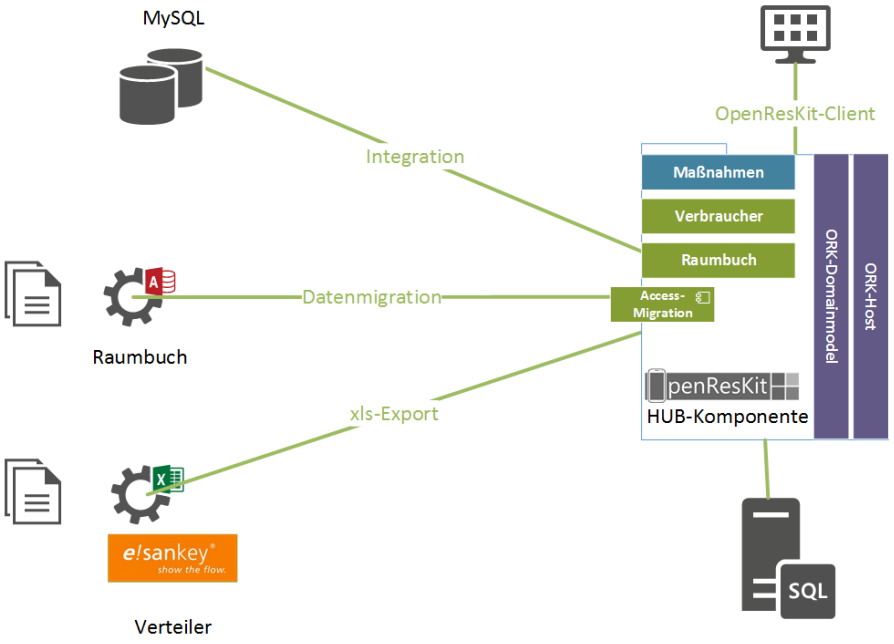

Abb. 2: Konzept für die Integration und Erweiterung der bestehenden Anwendung

5 Implementierung

Als Erstes war die Software der bestehenden Anwendung zu analysieren. Obwohl diese nach Benutzerangaben fast fehlerfrei läuft, wurde ein Reengineering durchgeführt, um die Weiterentwicklung zu ermöglichen und die Software zu verbessern. Dies beinhaltete auch das Entfernen sogenannter „Code Smells“, also schlecht strukturierter Programmcode, der die Verständlichkeit und damit die Wartbarkeit erschwert [Fo00]. Davon war besonders der Client betroffen. Für diesen entstanden in der Folge neue Module, die die zusätzlichen Kundenanforderungen abdecken. Besonderes Augenmerk sollte jedoch auf die technischen Besonderheiten gelegt werden, die mit der Erweiterung und Integration zusammenhängen.

5.1 Datenbankmigrationen

Für das Erstellen oder Ändern des Domänenmodells bietet EF zwei Ansätze an. Beim Model-First-Ansatz wird zunächst mit einem Designer ein grafisches Datenmodell erstellt. Daneben gibt es den Code-First-Ansatz, der bei OpenResKit verfolgt wurde. Es werden Klassen erstellt, die das Domänenmodell repräsentieren. Mit einer Konfigurationsdatei können Abhängigkeiten und Schlüsselbeziehungen modelliert werden. Seit

Version 4.3 bietet EF zwei Migrationsstrategien an, um in der Datenbank die Modelländerungen widerzuspiegeln. OpenResKit setzt auf die automatische Migration. Hierbei wird das bestehende Datenmodell mit der Datenbank abgeglichen und gegebenenfalls ein Update durchgeführt. Leider erkennt der automatisierte Updatemechanismus keine Umbenennungen von Tabellen oder Spalten, so dass alte Spalten gelöscht und neue erstellt werden.

Alternativ gibt es die eine codebasierte Migration. Dies widerspricht aber dem modularen Aufbau der OpenResKit-Architektur. Erst zur Laufzeit kennt der OpenResKit-Server die eingebundenen Module. Folglich können sich eventuelle codebasierte Migrationsdateien auch nicht auf die letzte Datenbankversion beziehen, da die Module das aktuelle Schema nicht kennen. Ein Update der Datenbank wäre nicht möglich. Ob sich dieses Problem mit automatisierten Merge-Migrations-Befehlen, analog zur Vorgehensweise in Teamumgebungen, lösen lässt, bedarf einer tiefergehenden Betrachtung.

5.2 Integration mittels ETL-Prozess

Eine typische Vorgehensweise zur Datenkonsolidierung ist der ETL-Prozess. ETL steht für Extract-Transform-Load, also Extrahieren-Transformieren-Laden. Zunächst werden die relevanten Daten aus den Quellsystemen entnommen. Der Prozess muss nicht einmalig ablaufen, sondern kann ereignisgesteuert, periodisch oder permanent sein. In der Transformationsphase werden die Datensätze so geändert, dass sie einheitlich in das Zielsystem passen. Dieser Teilschritt ist die aufwändigste Aufgabe, da die strukturelle, syntaktische und semantische Heterogenität der Quellsysteme überwunden werden muss. Anschließend werden die Daten in das Zielsystem geladen [Ro13, S 37 - 40]. Dieser Prozess wurde mithilfe von SQL Server Integration Services durchgeführt. Die Konfiguration mit der Entwickleroberfläche ist denkbar einfach. Es können verschiedene Datenflussaufgaben definiert werden. Sogenannte Quellen- und Zielassistenten werden angeboten, um die Verbindung zu den Ausgangsdaten beziehungsweise der Zieldatenbank herzustellen.

In dem hier beschriebenen Projekt wurden einmalig eine Access-Datei und einige Datensätze aus einer MySQL-Datenbank in die SQL Express Zieldatenbank migriert. Die Zielverbindung zur SQL-Datenbank kann mit dem Assistenten hergestellt werden. Das gewünschte Datenbankschema muss bereits vorhanden sein und wird nach dem Code-First-Ansatz erstellt. Die Quellverbindung kann mit einer OLE DB oder ODBC Schnittstelle realisiert werden. Beides sind von Microsoft entwickelte Schnittstellen, die eine einheitliche Datenbankzugriffsschicht bilden. Gegebenenfalls muss für das verwendete Quellsystem ein passender Treiber oder Connector dem Entwicklungssystem hinzugefügt werden. In diesem Projekt wurde eine MySQL Datenbank in der Version 4.1 verwendet. Ein standardmäßig nicht mehr verfügbarer Treiber konnte über die Website heruntergeladen werden.

5.3 OData Schnittstelle

Die Daten werden vom OpenResKit-Server mit einer OData-Schnittstelle angeboten. OData ermöglicht die Erstellung und Verarbeitung von REST-basierten Datendiensten. Ressourcen werden dabei über eine HTTP-Nachricht angefragt und bearbeitet. REST steht für Representational State Transfer und beschreibt Prinzipien zur Entwicklung verteilter Systeme. Antworten werden im JSON- oder ATOM-Format zurückgegeben. Da OData auf diesen weitverbreiteten Internettechnologien aufbaut, ist OData mit allen gängigen Programmiersprachen ansprechbar und es können plattformunabhängig Clients entwickelt werden. Die Datenbank kann natürlich auch weiterhin direkt mittels SQL selbst angesprochen werden.

6 Fazit

Durch das Projekt, das konsequent zusammen mit den Nutzern durchgeführt wurde, konnte die bisherige Komplexität der IT-Anwendungslandschaft verringert werden. Arbeitsschritte, die bisher in vier verschiedenen Anwendungen ausgeführt wurden, lassen sich nun in einer Anwendung bearbeiten. Eine Datenbank mit einer mehr als zehn Jahre alten Softwareversion wurde abgelöst. Die verschiedenen Datenstrukturen wurden in einer Datenbank unter Nutzung der beschriebenen Hub-Komponente vereinheitlich. Diese wird anders als die Ausgangsdatenbestände, auf einem zentralen Server redundant gesichert ausgeführt. Damit wurden aus Sicht des IT-Managements erhebliche Qualitätsverbesserungen erreicht. Die größten Vorteile ergaben sich aber aus Sicht der Nutzer. Alle relevanten Daten des Energiemanagements sind nun in einer Anwendung einsehbar und bearbeitbar. Der bisherige manuelle Datenabgleich zwischen verschiedenen Anwendungen entfällt. Weiterhin konnten aufgrund der zusätzlichen Möglichkeiten weitere Anforderungen, wie eine Auswertefunktion, umgesetzt werden. Die Daten sind über die OData-Schnittstelle auch in anderen Anwendungen, wie einer Excel-Tabelle für die e!Sankey-Auswertung, verfügbar.

OpenResKit hat sich für die Umsetzung als geeignet erwiesen. Insbesondere der modulare Aufbau des Hub-Servers bietet sich für pragmatische Umsetzung der Erweiterungen und Integrationsaufgaben an. Ein Update des Entity Frameworks auf Version 6.1 ist zu empfehlen, da hier umfangreiche Möglichkeiten zum Reengineering angeboten werden. So kann aus bestehenden Datenbanken der Quellcode generiert werden, um den Code-First Ansatz zu verfolgen. Für Integrations- und Erweiterungsaufgaben erleichtert dies die Erstellung des Domänenmodells erheblich.

Literaturverzeichnis

[Ba09] Balzert, H.: Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering. Spektrum Akademischer Verlag, Heidelberg, 2009.

[BF14] Bourque, Pierre; Fairley, R. E.: Swebok: Guide to the software engineering body of knowledge. Version 3.0. Los Alamitos, CA, IEEE Computer Society, 2014.

[BRW02] Bennett, Keith H.; Rajlich, Vaclav; Wilde, Norman: Software evolution and the staged model of the software lifecycle. In: Adcvances in computers (2002), Nr, 56, S. 1-54.

[Fo00] Fowler, M.: Refactoring. Wie Sie das Design vorhandener Software verbessern. Addison-Wesley, München u.a., 2000.

[Hu14] Huber, Sebastian: Informationsintegration in dynamischen Unternehmensnetzwerken: Architektur, Methode und Anwendung. Wiesbaden, Gabler, 2014.

[Ju06] Jung, R.: Architekturen zur Datenintegration. Gestaltungsempfehlungen auf der Basis fachkonzeptueller Anforderungen, Deutscher Universitäts-Verlag/GWV-Fachverlage GmbH, Wiesbaden, 2006.

[Me12] Mertens, P. et al.: Grundzüge der Wirtschaftsinformatik. Springer Berlin Heidelberg; Imprint: Springer, Berlin, Heidelberg, 2012.

[PB06] Pellegrini, Tassilo ; Blumauer, Andreas ; Semantic Web: Wege zur vernetzten Wissensgesellschaft, 1. Auflage, Berlin, Springer, 2006.

[Ro13] Rossak, I. (Hrsg.): Datenintegration. Integrationsansätze, Beispielszenarien, Problemlösungen, Talend Open Studio; mit 15 Tabellen sowie Kontrollfragen und Aufgaben. Hanser, München, 2013.

[Sn12] Sneed, H. M.: Nachhaltigkeit durch gesteuerte Software-Evolution. In (Brandt-Pook, H. Hrsg.): Nachhaltiges Software Management. Fachtagung des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung im Fachbereich Wirtschaftsinformtik (WI-MAW); 14. - 16. November 2012 in Bielefeld. Ges. für Informatik, Bonn, 2012, S. 50–68.

[So11] Sommerville, Ian: Software Engineering. 9th ed., Pearson, Boston, 2011.

[Ti13] Tiemeyer, Ernst (Hrsg.): Handbuch IT-Management: Konzepte, Methoden, Lösungen und Arbeitshilfen für die Praxis. 5.überarb. und erw. Aufl., Hanser, München, 2013.

[Vd11] VDI Zentrum für Ressourceneffizienz GmbH: Umsetzung von Ressourceneffizienz-Maßnahmen in KMU und Ihre Treiber. Berlin, 2011.

[Wo14] Wohlgemuth, Volker; Krehahn, Peter; Ziep, Tobias; Schiemann, Lars: Entwicklung eines Open-Source basierten Baukastens zur Unterstützung und Etablierung der Ressourceneffizienz in produzierenden KMU. In: Wohlgemuth, Volker (Hrsg.) ; Lang, Corinna V. (Hrsg.) ; Marx Gómez, Jorge (Hrsg.): Konzepte, Anwendungen und Entwicklungstendenzen von betrieblichen Umweltinformationssystemen (BUIS). Shaker, Aachen, S. 41 – 57, 2014.