AK WI

Herausgeber
Thomas Barton
Burkhard Erdlenbruch
Michael Guckert
Frank Herrmann
Christian Müller
Harald Ritz

Herausforderungen an die Wirtschaftsinformatik:

Integration und Konnexion

unterstützt durch:
THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Beiträge der Fachtagung »**Integration und Konnexion**« im Rahmen der 26. Jahrestagung
des Arbeitskreises Wirtschaftsinformatik an Fachhochschulen (AKWI)
vom 15.09. bis 18.09.2013 an der Technischen Hochschule Mittelhessen

Autoren:
Janis Albersmeier, Wolfgang Alm, Daniel Brunner, Carsten Dorrhauer, Thomas Farrenkopf, Fabian Geist, René Gerlach, Michael Guckert, Andreas Heberle, Timon Held, Benjamin Hoffmann, Georg Rainer Hofmann, Peter Hohmann, Sascha Höhn, Benjamin Horst, Christian Jablonski, Christian Kaiser, Norbert Ketterer, Ute Klotz, Torsten Kluin, Jens Kohler, Oliver Kuchler, Elvira Kuhn, Nikolai Kunz, Martin Kütz, Konrad Marfurt, Frank Morelli, Christian Müller, Gordon Müller, Rainer Neumann, Ertan Özdil, Timo Péus, Martin Przewloka, Jörg Puchan, Harald Ritz, Haio Röckle, Andreas P. Schmidt, Michael Schneppensiefer, Bernhard Schweizer, Meike Schumacher, Christian Seel, Carlo Simon, Thomas Specht, Heiko Thimm, Matthias Willems, Jürgen Zimmermann

Verlag News & Media, Berlin
ISBN 978-3-936527-36-0

Arbeitskreis Wirtschaftsinformatik an Fachhochschulen (AKWI)
www.akwi.de

Herausforderungen an die Wirtschaftsinformatik:

Integration und Konnexion

Tagungsband zur 26. AKWI-Jahrestagung
vom 15. bis 18.09.2013 an der Technischen Hochschule Mittelhessen

herausgegeben von
Thomas Barton, Burkhard Erdlenbruch, Michael Guckert,
Frank Herrmann, Christian Müller, Harald Ritz

Unterstützt durch das Präsidium, die Fachbereiche MND und MNI
und den Bachlor- und Master-Studiengang Wirtschaftsinformatik an der

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Verlag News & Media, Berlin

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

Herausforderungen an die Wirtschaftsinformatik:
Integration und Konnexion
Tagungsband zur wissenschaftlichen Fachtagung am 16.09.2013 anlässlich der 26. Jahrestagung des Arbeitskreises Wirtschaftsinformatik an Fachhochschulen (AKWI) vom 15. bis 18. September 2013 an der Technischen Hochschule Mittelhessen

Herausgeber:
Prof. Dr. Thomas Barton, Fachhochschule Worms
barton@fh-worms.de
Prof. Dr. Burkhard Erdlenbruch, Hochschule Augsburg
burkhard.erdlenbruch@hs-augsburg.de
Prof. Dr. Michael Guckert, Technische Hochschule Mittelhessen
michael.guckert@mnd.thm.de
Prof. Dr. Frank Herrmann, Hochschule Regensburg
frank.herrmann@hs-regensburg.de
Prof. Dr. Christian Müller, Technische Hochschule Wildau (FH)
christian.mueller@th-wildau.de
Prof. Dr. Harald Ritz, Technische Hochschule Mittelhessen
harald.ritz@mni.thm.de

Redaktion:
Teamarbeit der Herausgeber

Redaktionsschluss: 01.08.2013
Erscheinungstermin: 16.09.2013

Die Herstellung dieses Tagungsbandes erfolgte mit freundlicher Unterstützung durch das Präsidium, die Fachbereiche MND und MNI sowie den Bachlor- und Master-Studiengang Wirtschaftsinformatik an der Technischen Hochschule Mittelhessen.

Verlag News & Media, von Amsberg, Berlin

ISBN 978-3-936527-36-0

Konnexion im E-Commerce: Problemfelder und Lösungsansätze anhand eines internetgestützten B2B-Bestellsystems

Christian Jablonski, Daniel Brunner

1 Einleitung

Zur Abwicklung von Geschäftsbeziehungen zwischen Unternehmen haben Plattformen bspw. auf der Basis von EDI eine gewisse Verbreitung gefunden. Für eine derart weitgehende Prozessintegration müssen allen Standardisierungen zum Trotz hohe Kosten aufgewendet werden. Alternativ bieten sich Shop- oder Bestellsysteme an, bei denen die Kunden über ein Portal des Anbieters im Internet die Produkte zusammenstellen und anschließend bestellen können. Dies bietet sich insbesondere an, um schnell eine große Zahl von Nachfragern ansprechen zu können.

Derartige Shop-Systeme ließen sich voll integriert anbieten. Dies würde bedeuten, dass die Stammdaten zu Produkten direkt dem Warenwirtschaftssystem des Anbieters entnommen werden könnten. Umgekehrt könnten die Bestellungen direkt als Aufträge erstellt werden. Der starken Integration und ihrer Vorteile stehen mögliche Kosten entgegen: Lizenzkosten der Warenwirtschaft, Technologie-Lock-in, Verfügbarkeit bei schlechter Qualität der Datenleitung zur Warenwirtschaft etc.

Eine andere Lösung bestünde darin, in einem eigenen Bestellsystem die Stammdaten der Warenwirtschaft abzulegen und von Zeit zu Zeit die eingegangenen Bestellungen abzurufen und in der Warenwirtschaft weiter zu verarbeiten. Einer solchen als Konnexion bezeichneten Lösung, deren Verfügbarkeit nicht von der Warenwirtschaft des Kunden abhängt, stehen allerdings auch Kosten und Probleme gegenüber: So muss der Datenbestand zwischen Warenwirtschaft und Bestellsystem synchron gehalten werden. Dies betrifft den Artikelstamm, die Verfügbarkeiten, aber auch die Stati der eingegangenen Bestellungen.

In unserem Beitrag stellen wir anhand eines Bestellsystems und eines Warenwirtschaftsystems die Umsetzung der Konnexions-Lösung dar. Hierzu nehmen wir vorab zur Klassifizierung der Lösungen eine Definition des Begriffs Konnexion vor. Wir zeigen anschließend auf, wie mittels auf der REST-

Architektur basierenden Schnittstellen ein konsistenter Datenbestand und ein durchgängiger Prozess zwischen Anbieter und Kunde abgebildet werden kann. Aus unserem Entwicklungsprojekt zeigen wir anhand aufgetretener Problemfelder einige Lösungsansätze auf.

2 Definition und Realisierungsmöglichkeiten

Der Begriff Integration leitet sich aus dem Lateinischen ab und lässt sich mit „Wiederherstellung eines Ganzen", „Einbeziehung in ein größeres Ganzes" beschreiben.[1] Demgegenüber lässt sich Konnexion als „Zusammenführung, Verbindung" mehr im Sinne einer losen Kopplung verstehen.[2]

Eine solche sprachliche Annäherung an die Begriffe Integration und Konnexion liefert jedoch eine nur sehr oberflächliche Abgrenzung der Begriffe voneinander. Um eine klarere Abgrenzung beider Begriffe und deren Verwendung im Bezug auf betriebliche Anwendungssysteme vornehmen zu können, soll ein spezifischeres Kriterium definiert werden: Das Abgrenzungskriterium soll die „dezentrale Verwaltung von Ressourcen" bzw. im hier vorliegenden speziellen Fall die „dezentrale Datenhaltung" zwischen zwei oder mehreren Systemen sein.

Diese dezentrale Datenhaltung bezieht sich auf die Anwendungsarchitektur der betrachteten Systeme: In betrieblichen Anwendungssystemen, und gerade bei ERP-Systemen, um beim vorliegenden Anwendungsfall zu bleiben, findet man in der Regel eine dreischichtige Anwendungsarchitektur vor, bestehend aus einer Datenhaltungs-, einer Anwendungs- und einer Präsentationsschicht.

Bei zwei oder mehreren in irgendeiner Art und Weise mit einander verbundenen Anwendungssystemen liegt eine dezentrale Datenhaltung dann vor, wenn jedes Anwendungssystem seine eigene Datenhaltungsschicht besitzt und keinen direkten Zugriff auf die Daten eines anderen Anwendungssystems erhält. Ein Austausch von Daten zwischen den Systemen wird dabei nicht ausgeschlossen, jedoch findet dieser ausschließlich über definierte Schnittstellen zwischen den Systemen auf Ebene der Präsentations- oder der Anwendungsschicht statt (siehe Abbildung 1).

1 Vgl. [Bibl13-1].
2 Vgl. [Bibl13-2].

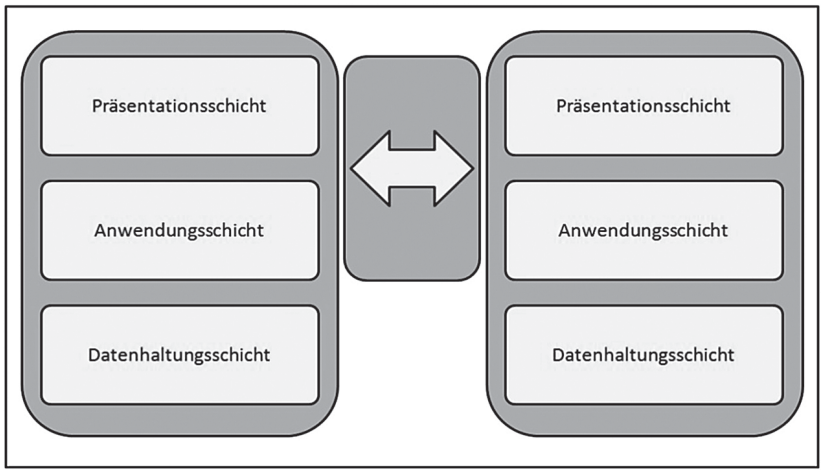

Abbildung 1 – Datenaustausch zwischen konnektierten Anwendungssystemen über eine Schnittstelle

Werden zwei oder mehrere Anwendungssysteme auf diese Art und Weise verbunden, soll dies als Konnexion bezeichnet werden. Werden jedoch zwei Anwendungssysteme verbunden, ohne dass das Kriterium der dezentralen Datenhaltung erfüllt ist, soll dies nach der vorgeschlagenen Definition nicht als Konnexion betrachtet werden. In diesem Fall wird man die Verbindung der Systeme eher als Integration bezeichnen können. Etwas allgemeiner gesprochen soll Konnexion dann vorliegen, wenn die Verwaltung von Ressourcen dezentral erfolgt. Ressourcen in diesem allgemeinen Fall seien Daten, Datenbanken, Dateien, Ein- und Ausgabegeräte etc.

Vorschlag einer Definition der Konnexion:
Liegt bei der Verbindung von dreischichtigen Anwendungssystemen eine dezentrale Datenhaltung (oder allgemeiner gesprochen, eine dezentrale Verwaltung von Ressourcen wie Daten, Datenbanken, Ein- und Ausgabegeräten etc.) vor und findet der Austausch von Daten (oder allgemeiner der Zugriff auf diese Ressourcen) über Schnittstellen auf der Präsentations- oder der Anwendungsschicht statt, so wird diese Verbindung der Systeme als Konnexion bezeichnet.

Diese Definition erlaubt eine Abgrenzung von Systemen, die lose aneinander gekoppelt sind und bei denen Teilsysteme ausgetauscht werden können.
Es ergeben sich somit zwei Szenarien für die Realisierung der Verbindung der ERP-Software Microsoft Dynamics NAV mit einem B2B-Bestellsystem.

2.1 Integrationsszenario

Die ERP-Software Microsoft Dynamics NAV, im Folgenden NAV genannt, besitzt eine dreischichtige Anwendungsarchitektur. Die Daten, die von NAV genutzt werden, sind in einer Datenbank abgelegt.

Bei einer Integration des B2B-Bestellsystems in die ERP-Lösung NAV, bekäme das Bestellsystem einen direkten Zugriff auf die Datenbank von NAV. Beide Systeme würden zusammengeführt und arbeiteten mit derselben Datenbasis. Damit das Bestellsystem von außerhalb des Unternehmens für dessen Kunden erreichbar ist, müsste es über einen Webserver online bereitgestellt werden, sodass der Endkunde den Webshop über einen Webclient in seinem Browser anzeigen kann. Der Webserver würde dabei direkt auf die Daten in der NAV-Datenbank zugreifen, es läge also eine zentrale Datenhaltung vor.

2.2 Konnexionsszenario

Bei einer Konnexion des B2B-Bestellsystems mit der ERP-Lösung NAV bekäme das Bestellsystem keinen direkten Zugriff auf die Datenbank von NAV. Stattdessen würde es eine eigene Datenbank verwenden. Mittels eines Webservers würde den Kunden außerhalb des Unternehmens das Bestellsystem bereit gestellt. Durch die Verwendung einer eigenen Datenbank benötigt das B2B-Bestellsystem keinen direkten Zugriff mehr auf die NAV-Datenbank. Der Datenaustausch zwischen NAV und B2B-Bestellsystem findet stattdessen über Schnittstellen statt.

3 Kriterien zur Auswahl des umgesetzten Szenarios

Zur Entscheidungsfindung über die Wahl der Architektur des Bestellsystems haben wir folgende Kriterien zugrunde gelegt:

Kriterium 1 – Verfügbarkeit

Im Fall der Vollintegration müssen die Datenbank des ERP-Systems und der Webserver ausreichend performant sein, um alle Zugriffe der Webshop-Bediener ausreichend schnell abzuarbeiten. Darüberhinaus müsste das voll integrierte System insgesamt mit einer entsprechend schnellen und ausfallsicheren Anbindung an das Internet ausgestattet sein. Die Sicherstellung akzeptablen Antwortverhaltens bedeutet, dass die Infrastruktur (Rechenkapazität, Anbindung) ausreichend groß ausgelegt oder skalierbar sein muss.
Im Falle einer Integrationslösung müssten diese Anforderungen an das Gesamtsystem bestehend aus ERP-System und Bestellsystem erfüllt sein. Dies

würde im Fall einer „Vor-Ort“-Lösung im Unternehmen bedeuten, dass dessen Infrastruktur entsprechend ausgelegt sein muss. Würde man sich statt der Integrationslösung für eine Konnexionslösung (vgl. Abschnitt 2.2) entscheiden, so müsste der Webserver nicht direkt im Unternehmen betrieben werden. Stattdessen könnte er beispielsweise in ein Rechenzentrum eines Drittanbieters ausgelagert werden. Viele Rechenzentren bieten skalierbare Lösungen an, was bedeutet, dass die Leistung des Systems sich bei höheren Anforderungen z. B. durch eine hohe Anzahl von gleichzeitigen Nutzern automatisch erhöht. Bei geringen oder gar keinen Zugriffen auf das System, z. B. in der Nacht, würde die Leistung automatisch wieder herunter skaliert. Alternativ könnte auch die ERP-Lösung und das Bestellsystem als Rechenzentrumslösung bereit gestellt werden.
Ein Großteil der NAV-Kunden unseres Unternehmens sind mittelständische Industrie- und Handelsunternehmen. Bei diesen ist in der Regel die Infrastruktur für eine eigene Hochverfügbarkeitslösung nicht vorhanden und die Unternehmen wollen ihre ERP-Lösung im eigenen Haus betreiben. Dies war ein Kriterium für die Verwendung einer Konnexionslösung mit Auslagerung des Webservers zu einem Drittanbieter.

Kriterium 2 – Schnelle Anbindung und Einrichtung des B2B-Bestellsystems
Ein Ziel bei der Planung des B2B-Bestellsystems war es, eine möglichst schnelle und einfache Anbindung an das vorhandene ERP-System zu ermöglichen. Größere Anpassungen an Hardware und Software des Kunden sollten vermieden werden. Stattdessen sollte die Architektur des Bestellsystems so aufgebaut sein, dass eine schnelle und einfache Installation in Form eines Zusatzmoduls für die ERP-Software möglich ist. Das Zusatzmodul sollte in wenigen Schritten installiert und danach sofort einsatzbereit sein. Dies sprach für eine Konnexionslösung und ist außerdem ein wichtiges Verkaufsargument für das B2B-Bestellsystem, mit dem sich das System von anderen am Markt vorhandenen Shop-Systemen abheben soll.

Kriterium 3 – Hohe Flexibilität statt Lock-In-Effekt
Der Lock-In-Effekt beschreibt, dass Kunden, die in die Integration eines Gutes investiert haben, an das zugehörige System gebunden sind. Dies heißt nicht, dass das System nicht gewechselt werden könnte. Je höher jedoch die Kosten für einen Wechsel des Systems sind, desto geringer wird die Neigung des Anwenders, das System zu wechseln.[3]
Sowohl bei einer Integrations- als auch bei einer Konnexionslösung kann ein technologischer Lock-In auftreten. Eine volle Integration in eine ERP-Software

3 Vgl. [ClSc10], S.231.

ist jedoch wesentlich komplexer und kostenintensiver als eine Teilintegration, bzw. Konnexion. Die Gefahr für einen Lock-In wird deshalb an dieser Stelle für eine Konnexionslösung geringer eingeschätzt als für eine Integrationslösung.

Kriterium 4 – Kosten
Je nach Wahl der ERP-Software und der zugrunde liegenden Datenbank fallen unter Umständen für den Kunden weitere Lizenzentgelte an. Die Ausgestaltung und Höhe hängen dabei vom jeweiligen Anbieter der Lösungen ab. Oft berechnet sich die Höhe der Entgelte z. B. an der Anzahl der Benutzer oder der Anzahl zugreifender Geräte. Die Anbindung des Bestellsystems über eine Schnittstelle mit eigener Datenhaltung kann im vorliegenden Fall von Microsoft Dynamics NAV die Kosten begrenzen. Dies sprach für eine Konnexionslösung.
Um die Kosten für die Entwicklung und den Betrieb des B2B-Bestellsystems möglichst gering zu halten, wurden für dessen Komponenten ausschließlich Open-Source-Produkte verwendet. Eine weitere Kostenersparnis kann durch den Betrieb des Webservers in einem Rechenzentrum mit skalierbaren Diensten erreicht werden. Dies sprach ebenfalls für eine Konnexionslösung.

4 Resultierende Architektur des B2B-Bestellsystems

Auf Basis der Entscheidung zu einer Konnexionslösung ergeben sich drei Hauptkomponenten:

(1) In der ERP-Software Microsoft Dynamics NAV werden die Daten erfasst, die über das B2B-Bestellsystem den Kunden außerhalb des Unternehmens zur Verfügung gestellt werden sollen. Dabei geht es vor allem um Artikelstammdaten des Unternehmens. Die Daten sind in einer Datenbank des ERP-Systems gespeichert. Die Mitarbeiter im Unternehmen greifen über Clients auf den NAV-Server zu und erhalten über dessen Anwendungsfunktionalität Zugriff auf diese Daten.
Der NAV-Server wird für die Konnexion des B2B-Bestellsystems erweitert. Dies geschieht in Form eines Zusatzmoduls, ohne dessen Standardfunktionalität zu verändern. Das Zusatzmodul, welches als Webshop-Modul bezeichnet wird, stellt außerdem die Schnittstelle zum B2B-Bestellsystem dar. Es enthält die Funktionalitäten zur Kommunikation mit dem B2B-Bestellsystem. Dies sind z. B. Funktionen zum Hochladen von Artikeldaten und Preisen.

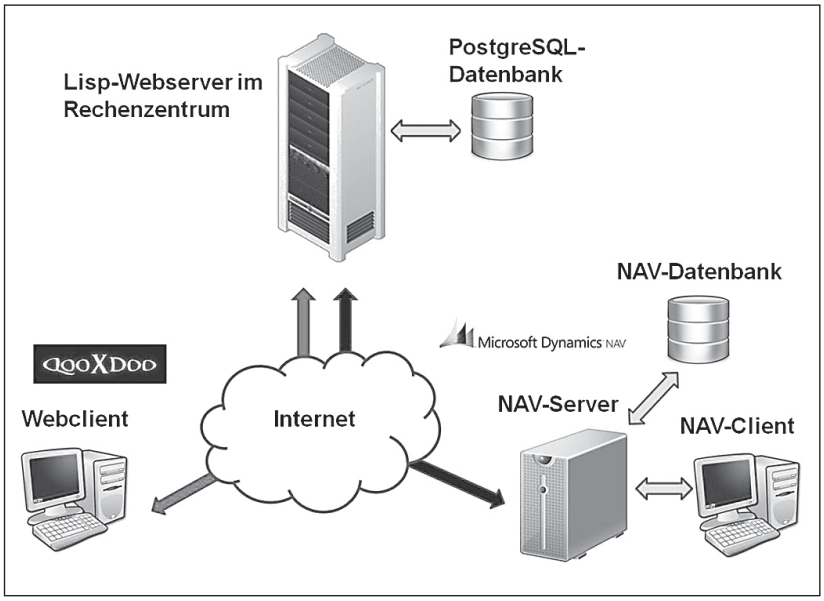

Abbildung 2 – Überblick über die Systemarchitektur des B2B-Bestellsystems mit Anbindung an NAV

(2) Den Kern des B2B-Bestellsystems bildet ein Webserver, der mit der Programmiersprache Lisp programmiert wurde. Dieser stellt die Funktionalität des B2B-Bestellsystems über einen Dienst nach außen bereit. Dieser Dienst ist eine Eigenentwicklung unseres Unternehmens. Dessen Datenhaltung erfolgt in einer PostgreSQL-Datenbank.[4] Webserver und Datenbank werden in einem Rechenzentrum betrieben. Somit ist eine hohe Verfügbarkeit des B2B-Bestellsystems gewährleistet. Alle im Webshop benötigten Daten können aus den vorhandenen Stammdaten in NAV mit dem Webshop-Modul an das Bestellsystem übertragen werden.
Die Übertragung der Daten erfolgt über eine REST-Schnittstelle.[5] Die Abkürzung REST steht für „Representational State Transfer“. Diese Architektur erlaubt eine zustandslose Datenübertragung zwischen NAV und dem Webserver. Die Daten werden hierbei im JSON-Format (JavaScript Object Notation) über eine http-Nachricht übermittelt. Die http-Methoden (GET, PUT, POST, DELETE) werden verwendet, um zu bestimmen, ob Daten abgerufen, angelegt, geändert oder gelöscht werden sollen.

4 Vgl. [Thep13].
5 Vgl. [Fiel00], Kapitel 5.

(3) Die dritte Komponente des B2B-Bestellsystems ist der Webclient, der den eigentlichen Webshop darstellt. Der Webshop kann über einen Browser von den Geschäftskunden aufgerufen werden. Die Kommunikation zwischen Webclient und Webserver läuft analog der Kommunikation zwischen NAV und Webserver unter Verwendung einer REST-Schnittstelle ab. Der Webclient wurde mit dem JavaScript-Framework qooxdoo programmiert.[6] Neben dem klassischen Webclient für Desktop-PCs bzw. Notebooks wurde noch ein weiterer Webclient für mobile Endgeräte entwickelt, der ebenfalls auf dem qooxdoo-Framework basiert. Dieser ist für die Anzeige auf mobilen Geräten wie Smartphones und Tablets optimiert.

5 Problemfelder und Lösungsansätze

In diesem Kapitel werden am Beispiel der Konnexion von Microsoft Dynamics NAV und dem von uns entwickelten B2B-Bestellsytems die Problemfelder beschrieben, die sich bei der Konnexion von Anwendungssystemen ergeben können und welche Lösungsansätze bei der Umsetzung der Konnexion im konkreten Fall entwickelt wurden.

5.1 Datenhaltung und Synchronisierung

Durch die Konnexion von Anwendungssystemen kommt es nach unserer Definition zu einer getrennten Datenhaltung. Trotzdem muss ein Austausch von Daten zwischen den Systemen über definierte Schnittstellen möglich sein. Datenbestände wie Artikeldaten müssen in diesem Fall regelmäßig synchronisiert werden.

In unserer Architektur werden u.a. Artikeldaten aus NAV auf den Webserver geladen, um diese online bereit zu stellen. Werden die Artikeldaten geändert, weil z. B. ein Artikel nicht mehr angeboten werden soll oder sich Preise ändern, dann müssen alle in NAV geänderten Daten erneut auf den Webserver geladen werden. Würde diese Synchronisierung nicht stattfinden, könnte dies zu Fehlern führen. So könnten z. B. Artikel bestellt werden, die eigentlich nicht mehr verfügbar sind oder Artikel würden zu einem falschen Preis angeboten.

Für jeden geänderten Datensatz wird dabei eine Nachricht erzeugt, die in einer Nachrichten-Warteschlange abgelegt wird. Wurden alle gewünschten Änderungen am Datenbestand vorgenommen, wird diese Nachrichten-Warteschlange abgearbeitet und die vorliegenden Nachrichten im JSON-Format

6 Vgl. [Qoox13].

an den Webserver gesendet. Dieser ändert anhand der empfangenen Nachrichten seinen Datenbestand, sodass er mit den Daten im Webshop-Modul in NAV synchron ist. Mit jeder Aktualisierung werden Rückgabewerte vom Webserver über Erfolg oder Misserfolg der Änderung empfangen und gesendete Nachrichten mit einem „gesendet“-Hinweis quittiert. Dadurch ist sichergestellt, der Datenbestand beider Systeme synchron ist und dass die Nachrichten nicht erneut gesendet werden. Daten vom Webserver können entweder manuell oder automatisch abgerufen werden. Auf diese Weise kann z. B. eine Bestellung, die ein Kunde im Webshop getätigt hat, vom Server abgerufen und in NAV eingelesen und weiterverarbeitet werden. Die REST-Architektur wurde derart gewählt, dass der Webserver jederzeit verfügbar sein muss, das ERP-System jedoch nicht. Somit muss NAV Änderungen immer von sich aus abrufen, ein direktes Senden des Webservers an NAV ist nicht vorgesehen.

5.2 Schnittstellen

Im Rahmen der Konnektierung von Microsoft Dynamics NAV und dem B2B-Bestellsystem musste eine Schnittstelle für die Kommunikation zwischen NAV und dem Webserver sowie eine Schnittstelle für die Kommunikation zwischen dem Webserver und dem Webclient definiert werden.

Über die Schnittstelle zwischen NAV und Webserver werden alle Daten aus NAV zum Webserver gesendet, die dieser für die Bereitstellung von Daten und Funktionen für den Webshop benötigt. Innerhalb des Webshop-Moduls sind Funktionen zum Erstellen und Senden sowie zum Abrufen und Verarbeiten von Nachrichten implementiert.

Alle Daten, die an den Webserver gesendet werden, müssen vom Webshop-Modul zuvor in das JSON-Format, der Objektnotation der Programmiersprache JavaScript, umgewandelt werden. Für alle Daten, die aus NAV gesendet werden können, legt die Schnittstelle eine Struktur im JSON-Format fest. Bedingt durch die REST-Architektur[7] der Schnittstelle können http-Nachrichten an unterschiedliche URI („Unified Resource Identifier“) des Webservers gesendet werden. Wie URIs und Nachrichtentypen an welchen URI des Webservers gesendet werden dürfen, ist in der Schnittstellenbeschreibung vereinbart; dadurch wäre es auch möglich, ein beliebiges anderes ERP- oder Warenwirtschaftssystem in gleicher Weise an das B2B-Bestellsystem anzubinden. Beim Abrufen von Daten vom Webserver durch NAV steht neben einer JSON-Datenstruktur auch der Abruf im XML-Format zur Verfügung.

---

7 Vgl. [Fiel00], Kapitel 5.

Zusätzlich zur Schnittstelle zwischen NAV und Webserver gibt es auch eine Schnittstelle zwischen Webserver und Webclient. Auch diese Schnittstelle basiert auf der REST-Architektur. Die Daten werden hier jedoch ausschließlich im JSON-Format ausgetauscht. Dies bietet sich an, da der Webclient mit JavaScript programmiert wurde. Die empfangenen Daten können auf diese Weise gleich als JavaScript-Objekte weiterverarbeitet werden.

5.3 Optimierung der Datenübertragung

Im E-Commerce findet der Datenaustausch zwischen den beteiligten Anwendungssystemen über das Internet statt. Eine Herausforderung hierbei ist es, trotz möglicherweise geringer Bandbreiten eine schnelle und verlässliche Kommunikation zwischen den Anwendungssystemen zu gewährleisten. Am Beispiel unseres B2B-Bestellsystems betrifft dies vor allem den Webclient. Diesen muss der Kunde schnell bedienen können und es darf zu keinen langen Ladezeiten kommen, während durch den Webshop navigiert wird. Bei der Programmierung des Webclients wurden daher verschiedene Techniken angewendet, um eine schnell reagierende Benutzeroberfläche mit kurzen Ladezeiten zu realisieren:

Eine Maßnahme zur Verkürzung von Ladezeiten ist beispielsweise das Caching (Zwischenspeichern) des Warenkorbes im Webclient. Ein Warenkorb wird nicht nur auf dem Webserver gespeichert, sondern auch lokal im Webclient. Dies hat den Vorteil, dass nach dem Hinzufügen oder Löschen eines Artikels im Warenkorb nicht noch einmal der komplette Warenkorb vom Server angefordert werden muss, um die aktuelle Menge des Artikels im Warenkorb anzuzeigen. Stattdessen wird der lokale Warenkorb aktualisiert und nur eine einzige Nachricht an den Server gesendet, damit die Änderung dort auch übernommen wird und beide Warenkörbe synchron sind. Es werden darüber hinaus Zeitstempel der Änderungen mitgeführt, um vom Webclient aus beurteilen zu können, ob sich der Warenkorb aufgrund anderer Umstände verändert hat (beispielsweise, weil der Besteller das Bestellsystem in zwei unterschiedlichen Browsern gleichzeitig geöffnet hat). In dem Fall ruft er erneut den vollständigen Warenkorb vom Server ab.

Eine weitere Maßnahme zur Verkürzung der Ladezeiten ist die Verwendung des JSON-Formats beim Nachrichtenaustausch zwischen Webserver und Webclient. Die JSON-Nachrichten können direkt von JavaScript weiter verarbeitet werden, da sie in der entsprechenden Objektnotation vorliegen. In den Nachrichten werden zur Optimierung der übertragenen Datenmenge nur die Daten übermittelt, die für die jeweilige Operation benötigt werden. Das bedeutet beispielsweise, dass für die Anzeige einer Artikelübersicht nur die Daten

der Artikel vom Server abgerufen werden, die auch in der Artikelübersicht angezeigt werden. Wenn es für einen Artikel eine Detailansicht gibt, in der mehr Daten angezeigt werden als in der Artikelübersicht, so werden diese Daten erst beim Aufruf der Detailansicht vom Webserver abgerufen. Ein weiterentwickelter Ansatz ist in diesem Zusammenhang die Verwendung von „Prefetching“ (vorausschauendem Laden) und „Caching“ (Zwischenspeichern).[8] Dabei werden z. B. während dem Betrachten einer Seite mit Artikeln im Hintergrund bereits die Artikeldaten für die Anzeige der nächsten Seiten geladen und zwischengespeichert. Beim Aufruf der nächsten Seite durch den Benutzer wurden die Daten dann bereits empfangen und müssen nur noch angezeigt werden. Verzögerungen durch lange Ladezeiten beim Navigieren durch Webanwendungen werden dadurch verringert oder treten im besten Fall gar nicht erst auf. Gerade bei der Entwicklung von Webanwendungen für mobile Endgeräte spielt dies eine große Rolle. Die Bandbreite, die diesen Geräten zur Verfügung steht, ist oft geringer als die Bandbreite an einem Desktop-PC und unterliegt größeren Schwankungen.

6 Fazit

Die Konnexion von Anwendungssystemen ist eine Möglichkeit, wie Anwendungssysteme miteinander verbunden werden können. Eine Konnexion von Anwendungssystemen bedeutet, dass zwischen den Systemen eine „lose“ Kopplung hergestellt wird. Dies ermöglicht einen Austausch von Daten und Funktionalitäten zwischen den Systemen, ohne diese zu einer schwer wieder zu trennenden großen Einheit zusammen zu führen. Dadurch unterscheidet sich die Konnexion von der Integration von Anwendungssystemen.

Der Vorteil der Konnexion gegenüber der Integration von Anwendungssystemen ist eine höhere Flexibilität aufgrund der „losen“ Kopplung. Die Konnexion eines Anwendungssystems kann leichter wieder aufgehoben werden als bei einer vollen Integration. Ein Anwendungssystem kann dadurch auch einfacher durch ein anderes Anwendungssystem ersetzt werden. Insbesondere können die Teilsysteme am Beispiel unseres B2B-Bestellsystems in unterschiedlichen Betriebsumgebungen (ERP-Lösung im Haus, Bestellsystem im Rechenzentrum) betrieben werden. Dies sicherte insbesondere die Verfügbarkeit des Bestellsystems.

8 Vgl. [Mats13], S.59.

Erkauft werden die Vorteile der Konnexion im Falle des B2B-Bestellsystems durch Herausforderungen und Problemfelder, die bei einer vollen Integration nicht auftreten würden. So müssen aufgrund der dezentralen Datenhaltung Schnittstellen bereit gestellt werden, welche einen schnellen und sicheren Austausch von Daten zwischen den Systemen gewährleisten und eine Synchronisierung der Daten möglich machen. Dies bedeutet für den NAV-Anwender einen erhöhten Aufwand, da er nach jeder Änderung des Datenbestandes in NAV sicherstellen muss, dass die geänderten Daten auch an den Webserver gesendet werden.

Abschließend kann die entwickelte Konnexionslösung als durchaus praktikable, kostengünstige und flexible Alternative zu einer Integrationslösung betrachtet werden, sofern die genannten Schwachstellen kein Ausschlusskriterium darstellen.

Literatur

[Bibl13-1] *Bibliographisches Institut GmbH:* Suchwort Integration. http://www.duden.de/rechtschreibung/Integration. Abruf am 2013-04-24

[Bibl13-2] *Bibliographisches Institut GmbH:* Suchwort Konnexion. http://www.duden.de/rechtschreibung/Konnexion. Abruf am 2013-04-24

[ClSc10] *Clement, Reiner; Schreiber, Dirk:* Internet-Ökonomie – Grundlagen und Fallbeispiele der vernetzten Wirtschaft. Physica-Verlag 2010; Online-Ausgabe bei Springer-Verlag, Berlin Heidelberg 2010. DOI: 10.1007/978-3-7908-2596-1

[Fiel00] *Fielding, Roy Thomas:* Architectural Styles and the Design of Network-based Software Architectures; Diss., University of California, Irvine; 2000; http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf. Abruf am 2013-04-18

[Kaib04] *Kaib, Michael:* Enterprise Application Integration: Grundlagen, Integrationsprodukte, Anwendungsbeispiele. 1. Auflage, Nachdruck; Deutscher Universitäts-Verlag, Wiesbaden, 2004

[Mats13] *Matsudaira, Kate:* Making the Mobile Web Faster. In: Communications of the ACM, Band 56, Heft 3/2013, S. 56 ff.. DOI: 10.1145/2428556.2428572

[Me++12] *Mertens, Peter; Bodendorf, Freimut; König, Wolfgang; Picot, Arnold; Schumann, Matthias; Hess, Thomas:* Grundzüge der Wirtschaftsinformatik. 11. Auflage 2012; Online-Ausgabe bei Springer-Verlag; Berlin Heidelberg 2012. DOI: 10.1007/978-3-642-30515-3

[Wa++12] *Wagner, Klaus-P.; Hüttl, Thomas; Backin, Dieter; Vieweg, Iris; Werner, Christian:* Einführung in die Wirtschaftsinformatik, IT-Grundwissen für Studium und Praxis. Gabler Verlag; Wiesbaden 2012; Online-Ausgabe bei Springer-Verlag, Berlin 2012, DOI: 10.1007/978-3-8349-6856-2

[Qoox13] *qooxdoo developers:* Offizielle Webseite des qooxdoo-Projektes; http://www.qooxdoo.org. Abruf am 2013-05-09

[Thep13] *The PostgreSQL Global Development Group:* Offizielle Webseite des PostgreSQL-Projektes; http://www.postgresql.org/about/licence. Abruf am 2013-05-10

Kontakt

Christian Jablonski, B. A.
Brunner GmbH & Co. KG Informationsverarbeitung
Schulstraße 8, 35216 Biedenkopf
und
Technische Hochschule Mittelhessen, Studium Plus, Wetzlar
christian.jablonski@systemhaus-brunner.de

Dipl.-Vw. Daniel Brunner
Brunner GmbH & Co. KG Informationsverarbeitung
Schulstraße 8, 35216 Biedenkopf
und
Philipps-Universität Marburg, Marburg
daniel.brunner@systemhaus-brunner.de

Dieser Beitrag stellt eine von beiden Autoren überarbeitete Fassung der Bachelor-Thesis von Christian Jablonski dar, die während seines Dualen Studiums von Daniel Brunner betreut wurde.