Portal agil mit Liferay: Best Practices für die Portal-Integration

Dieser Artikel zeigt, wie wir typische Best Practices für die Entwicklung in einem agilen Liferay-Portal-Projekt adaptiert und angewendet haben. Einige davon dürften sinngemäß auch auf andere Portalprodukte übertragbar sein. Wir beschreiben unseren bevorzugten Portal-Entwicklungsansatz, der neben Agilität auch auf Liferay-Besonderheiten eingeht, und erläutern, mit welcher Infrastruktur wir diesen unterstützen. Außerdem schildern wir, wie wir als Portal-Integrator in Portal-Projekten mit mehreren beteiligten Parteien agieren.

Ein neues Banking-Portal

Unser Fallbeispiel ist die Implementierung eines neuen Kunden-Portals mit integriertem Online-Banking und -Brokerage bei einer mittelständischen Bank. Die bisherige Kunden-Website der Bank war eine über viele Jahre gewachsene Eigenentwicklung eines IT-Dienstleisters. Mit den gestiegenen Erwartungen der Endkunden – insbesondere an den Komfort und die Funktionsvielfalt des Online-Bankings – konnte diese Lösung nicht mehr Schritt halten. Die Neuimplementierung auf Basis des Liferay Portal Servers sollte Abhilfe schaffen.

Wenn ein ganzes Finanzportal neu gebaut wird, ist die Versuchung groß, die gesammelten Anforderungen und neuen Ideen vollständig und individuell umzusetzen. Wird die Implementierung mit Standardsoftware – in unserem Fall dem Liferay Portal Server – und zusätzlich von mehreren Implementierungspartnern durchgeführt, führt dieser Wunsch leicht zu einem Ergebnis, das in puncto Wartbarkeit und Betriebsfähigkeit verbesserungswürdig ist. Bindet man die Projektbeteiligten jedoch in einen klar definierten Integrationsrahmen ein und schöpft die Stärken des Liferay Portal Servers bei der agilen Umsetzung der Anforderungen aus, kommt man zu einem auch für die Fachseite sehr zufriedenstellenden Ergebnis, ohne sich zukünftige Probleme aufzubürden. Eine gewisse Flexibilität bezüglich der Anforderungen seitens des Kunden ist dabei Voraussetzung.

Neben der Fachabteilung „Electronic Banking“ und der internen IT waren an dem Projekt mehrere Parteien beteiligt:

- Ein *IT-Dienstleister*, der den Auftrag hatte, die Online-Banking-Komponenten neu zu bauen.
- Eine *Online-Agentur*, die die Vorüberlegungen der Fachabteilung in einem Fachkonzept beschreiben und das Design erstellen sollte.
- Ein *Zulieferer* für die Brokerage-Komponente, die in das Online-Banking integriert werden sollte.
- *Produkthersteller* (extern) und *Betreuer* (intern) weiterer zu integrierender Anwendungen und Services.

Unsere Aufgabe als „Portal-Integrator“ bestand darin, diese Parteien unter einem gemeinsamen Projekt- und Architekturrahmen zu steuern sowie selbst das neue Portal aufzubauen und alle Nicht-Online-Banking-Komponenten zu implementieren.

Best Practice 1: Qualität des Fachkonzepts sichern

Wenn ein Kundenportal neu gebaut wird, gibt es oft zahlreiche Vorüberlegungen und Wünsche auf der Fachseite. Diese fügen sich aber meist nicht zu einem schlüssigen Konzept – das finale Bild muss erst noch entstehen. Insbesondere dann, wenn die alte Lösung bestimmte Schwächen hatte, werden sich manche Anforderungen aufgestaut haben, ohne bewertet und eingeordnet worden zu sein.

Daher ist es sinnvoll, vor oder zu Projektbeginn ein Konzept zu erstellen, das die grundlegende Struktur des Portals als Rahmen für die Einordnung von Ideen und Anforderungen festlegt. Dabei werden sowohl neue als auch viele in der Vergangenheit liegen gebliebene Anforderungen aufkommen, die gezielt hinterfragt werden können. Mock-ups von Portal-Seiten eignen sich dabei gut zur Klärung der grundlegenden Anforderungen und als Grundlage für das visuelle Design.

Ein (Java-)Portal-Server ist zunächst einmal eine Ablaufumgebung für Portlets. Portlets sind Java-Programme, die Fragmente von HTML-Code erzeugen und die zusammen mit anderen Portlets – ohne sich gegenseitig zu beeinflussen – in einem Portal-Server flexibel auf Web-Seiten zusammengestellt werden können. Der Portal-Server stellt dabei die grundlegende Infrastruktur zur Verfügung. Darüber hinaus bieten Portal-Server mittlerweile viele weitere, nicht immer standardisierte Dienste und Funktionen an. Das ist auch der Grund, warum Portlets meist nur in der Theorie portabel zwischen verschiedenen Portal-Servern sind. Neben Closed Source Portal-Servern wie „IBM WebSphere Portal Server“ und „Oracle WebCenter“ gibt es Open-Source-Produkte wie „JBoss Portal“ oder den in diesem Artikel betrachteten Liferay Portal Server.

Liferay Portal Server kombiniert Liferay-Eigenentwicklungen mit bewährten Open-Source-Komponenten wie beispielsweise Apache Lucene. Die Funktionalität von Liferay lässt sich wie folgt grob gliedern:

- Basis-Portal mit den Standard-Portlet-APIs und Liferay-APIs.
- Integriertes Content-Management-System und Dokumentenmanagement-System.
- Mitgelieferte Portlets, die viele typische Portal-Anwendungsfälle (z. B. Wiki, Foren, Blogs) abdecken.
- Zusätzlich installierbare Pakete (sogenannte Plug-ins) von Liferay oder Drittanbietern, die Funktionalität nachrüsten. Dabei kann es sich um Portlets oder andere Erweiterungsarten handeln.

Kasten 1: Portal-Server und Liferay in Kürze.

Das Konzept muss mindestens auf einem Niveau sein, das die grundlegende Konfiguration des Liferay-Portals zu Projektbeginn ermöglicht, da sich diese nicht so agil ändern lässt wie beispielsweise Seitenstrukturen und -inhalte. In der durch das Konzept vorgegebenen Struktur lassen sich die Details während der Umsetzung anhand praktischer Anschauung gemeinsam mit dem Kunden ausarbeiten. Das setzt natürlich voraus, dass der Kunde die agile Vorgehensweise mitträgt. Typische Qualitätsmerkmale für ein solches Konzept sind:

- *Vollständigkeit:* Für alle spezifizierten Elemente gibt es eine Beschreibung.
- *Redundanzfreiheit:* Übergreifende Elemente wie beispielsweise Suche und Navigation sind einmal zentral beschrieben.
- *Widerspruchsfreiheit:* Sich an mehreren Stellen wiederholende Funktionalität ist widerspruchsfrei beschrieben.
- *Differenzierung:* Unterschiedliches Verhalten eines mehrfach auftauchenden Elements ist für den jeweiligen Kontext korrekt beschrieben.
- *Umsetzbarkeit:* Die beschriebene Funktionalität widerspricht nicht der Philosophie der eingesetzten Standardsoftware. Natürlich kann man ein Fachkonzept auch technikneutral schreiben – steht das einzusetzende Produkt jedoch bereits fest, ist es viel effizienter, gleich auf die praktische Umsetzbarkeit zu achten.

In unserem Fallbeispiel wurde von der Online-Agentur ein eher visuell orientiertes Konzept erstellt, das noch nicht die notwendige Qualität für den Beginn der agilen Umsetzungsphase hatte. In einer ähnlichen Situation würden wir empfehlen, bei der Erstellung des Konzepts einen Facharchitekten mit Kenntnis der einzusetzenden Portal-Software einzubinden.

Best Practice 2: Das visuelle Design auf das Portal abstimmen

In vielen Portal-Projekten ist eine Online-Agentur für das visuelle Design verantwortlich. Baut die Agentur das Design dabei auf der grünen Wiese, verkompliziert das die Integration mit den bei Liferay mitgelieferten Portlets. Styles und Formatierungen greifen häufig nicht, weil ihre Definition die durch Liferay in den Seitenaufbau eingebrachten Elemente nicht berücksichtigt. Beispielsweise nutzen einige Liferay-Portlets HTML-Tabellen. Falls das gelieferte *Cascading Style Sheet (CSS)* auf einem tabellenlosen Design basiert, ist viel Integrationsaufwand zu leisten. Bei eigenentwickelten Portlets ist das weniger problematisch, aber auch dort kann es durch die Vermischung mit den Liferay-Styles zu Problemen kommen.

Empfehlenswerter ist es, der Agentur in einer Testumgebung frühzeitig Zugriff auf Beispiel-Liferay-Portlets und -Seiten bzw. auf das von diesen generierte HTML zu geben und sie dafür passende CSS bauen zu lassen.

Stufe	Umsetzungsebene	Liferay-Funktionalität
1	Übergreifende visuelle Anpassung und Seitenraster	Themes und Layouts
2	Angepasste Vorlagen für einzelne Inhalte und deren visuelle Darstellung	Strukturen und Templates
3	Visuelle Darstellung von Inhalten plus einfache Funktionalität	Portlet, das den Web-Content in Daten konvertiert und darstellt
4	Komplexere Funktionalität mit Ähnlichkeit zu bestehenden Liferay-Portlets oder -Funktionen	Hooking eines Liferay-Portlets oder einer Liferay-Funktion
5	Komplexere Funktionalität ohne Ähnlichkeit zu bestehenden Liferay-Portlets	Eigenentwickeltes Portlet
6	Funktionalität, die anders nicht umsetzbar ist	Extension (nicht empfohlen)

Tabelle 1: Umsetzungsebenen.

Best Practice 3: Die Umsetzung agil machen

In unseren Liferay-Projekten hat es sich bewährt, die Anforderungen des Kunden so umzusetzen, dass dieser rasch Ergebnisse sieht und sie verstehen und bewerten kann. Nur wenn der Kunde frühzeitig sieht, was er bekommt, kann er auch rechtzeitig eingreifen, wenn etwas nicht seinen Erwartungen entspricht. Daher setzen wir auf eine agile Vorgehensweise. Diese muss nicht formal und komplex sein – aus der Struktur eines Portals ergeben sich natürlicherweise Sprint-Inhalte, die man gemeinsam mit dem Kunden priorisieren kann.

Eine der Stärken von Liferay ist, dass es „out of the box“ viel Funktionalität mitbringt, die bewusst auf mehreren Ebenen änder- und erweiterbar ist. Damit lassen sich Anforderungen so umsetzen, dass das Ergebnis für den Kunden einerseits frühzeitig sichtbar wird, andererseits aber so anpassbar bleibt, dass Feedback des Kunden noch einfließen kann.

Zusammen mit einer entsprechenden agilen Vorgehensweise ergibt sich mehrfacher Nutzen:

- *Transparenz:* Der Kunde sieht frühzeitig, was er bekommt – er kann sich ein Bild machen.
- *Zusammenarbeit:* Basierend auf den Zwischenergebnissen kann der Kunde seine Anforderungen konkretisieren. Auf diese Weise steigt nach unserer Erfahrung seine Bereitschaft, die Anforderungen im Detail flexibel anzupassen.
- *Wartbarkeit:* Das Endergebnis ist besser zu warten, da wir den Kunden gegebenenfalls zum Verzicht auf oder die Änderung von Anforderungen bewegen können, die nicht gut zum Portalstandard passen.

Best Practice 4: Anpassen statt neu implementieren

Entwicklung in Liferay ist auf mehreren Ebenen möglich, die von der rein visuellen Anpassung bis zur Änderung des Portal-Quellcodes reicht. Wir streben an, für die Umsetzung von Anforderungen immer die höchstmögliche, also einfachste Ebene zu nutzen und möglichst nahe am Portalstandard zu bleiben, um die Wartbarkeit und zukünftige Updates nicht zu erschweren. Wenn sich eine Anforderung in leicht veränderter Form auf einer einfacheren Ebene oder näher am Portalstandard umsetzen lässt, diskutieren wir das mit dem Kunden und demonstrieren die veränderte Umsetzung bei Bedarf prototypisch.

Zunächst kümmern wir uns um das übergreifende visuelle Design des Portals. Dafür

bauen wir ein gegebenenfalls über Parameter konfigurierbares Theme, das den Seitenrahmen bereitstellt und die Anforderungen an Navigation und Design erfüllt sowie je nach Bedarf Layouts für verschiedene Seitentypen enthält.

Alles, was reinen Inhalt darstellt, bilden wir als Web-Content ab. Dafür erstellen wir eine Struktur und ein oder mehrere Templates, über die sich mit der Struktur angelegte Web-Contents im Web-Content-Display-Portlet (Liferay-Standard) darstellen lassen. Ist Funktionalität erforderlich, die nicht oder nur schwer als Template abbildbar ist, erzeugen wir ein einfaches Portlet, das den Web-Content in ein Datenobjekt konvertiert und darstellt.

Alles, was jetzt noch übrigbleibt, ist vermutlich komplizierter. Aber vielleicht kann Liferay das bereits? Also prüfen wir, ob es ein mitgeliefertes Portlet gibt, das die passende Funktionalität anbietet, und implementieren dafür einen sogenannten *Hook*. Hooking ist an vielen Stellen in Liferay vorgesehen und ermöglicht die Anpassung und Erweiterung des Portals ohne Änderung des Quellcodes. So sind beispielsweise Änderungen der Portal-*JSPs (Java Server Pages)* und -Services machbar. Im Fallbeispiel nutzen wir diese Variante unter anderem für die Suche und Formulare. Erst wenn all das nicht reicht, entwickeln wir selbst ein komplett neues Portlet.

Die tiefgreifendste Erweiterungsmöglichkeit in Liferay ist die *Extension*, mit der sich der Liferay-Quellcode direkt verändern lässt. Das ist eine sehr mächtige Option, auf die wir aufgrund ihrer Nachteile jedoch grundsätzlich verzichten. Extensions erschweren durch den direkten Eingriff in den Quellcode künftige Updates und jedes Deployment einer Extension erfordert einen Neustart des Servers. Stoßen wir auf eine Anforderung, die nur als Extension umsetzbar wäre, loten wir auf jeden Fall mit dem Kunden eine Änderung der Anforderung aus.

Best Practice 5: Build often, deploy often

Um bei der agilen Umsetzung früh und oft Ergebnisse zeigen zu können, ist es notwendig, den Build-Prozess möglichst vollständig zu automatisieren. So lassen sich mit geringem Aufwand neue Versionen des Portals bereitstellen und ausrollen.

Der automatisierte Prozess bedient mehrere Umgebungen, die kurz eingeführt werden, beginnend mit den Vorintegrationsumgebungen. Für jeden im Projekt beteiligten Software-Lieferanten existiert eine solche Umgebung, in der dieser seine Entwicklungen testen und gegebenenfalls auch das Zusammenspiel mit anderen Komponenten ausprobieren kann. Die Verantwortung für diese Umgebungen liegt bei den Lieferanten. Die Ergebnisse der Vorintegration – deploybare Portlets, Hooks oder Themes – werden für die Integration bereitgestellt. Wir nennen diese Ergebnisse im Folgenden *Artefakte*.

Nachdem die Ergebnisse aus der Vorintegration bereitstehen, werden sie auf der Integrationsumgebung gesammelt. Hier werden alle Artefakte das erste Mal zusammen auf einer Umgebung installiert. Danach prüfen wir in Integrationstests das Zusammenspiel der einzelnen Artefakte und stellen fest, ob Wechselwirkungen auftreten. Auch wenn hier vornehmlich technische Aspekte getestet werden, empfiehlt sich ein Funktionstest der fachlichen Komponenten. Dadurch lässt sich sicherstellen, dass sich die Funktionalität der Anwendung wie spezifiziert verhält.

Nach unserer Projekterfahrung ist mit bekannten Continuous-Integration-Produkten der Aufbau eines effektiven Build-Prozesses möglich. Wir verwenden dafür das Tool „Jenkins“ zusammen mit einem Maven-Build-Prozess (**siehe Abbildung 1**). Wichtig ist die Verfügbarkeit eines Repositorys, in das sich alle Artefakte einstellen lassen. Über dieses findet die Übergabe der Artefakte von den Vorintegrationsumgebungen zur Integrationsumgebung statt. Im Folgenden betrachten wir den Prozess und die Unterschiede und Umsetzung der einzelnen Umgebungen näher.

Vorintegration

Die Vorintegrationsumgebungen bei den einzelnen Software-Lieferanten dienen in erster Linie dazu, auf kontrolliertem und reproduzierbarem Weg den Quellcode in die Artefakte zu übersetzen und diese zu testen. Die geprüften Artefakte werden anschließend mit einem eindeutigen Bezeichner in das zentrale Repository eingestellt.

Nach unserer Erfahrung bewährt es sich, zu jedem Artefakt auch Javadoc- und Quellcode-Bundles bereitzustellen. Der Hintergrund ist, dass die (Wieder-)Verwendung des Artefakts möglichst einfach sein soll. Javadoc und Quellcode helfen dem Benutzer des Artefakts, die Schnittstellen und Funktionen besser zu verstehen. Zusätzlich kann man sich mit Hilfe des Quellcodes bei

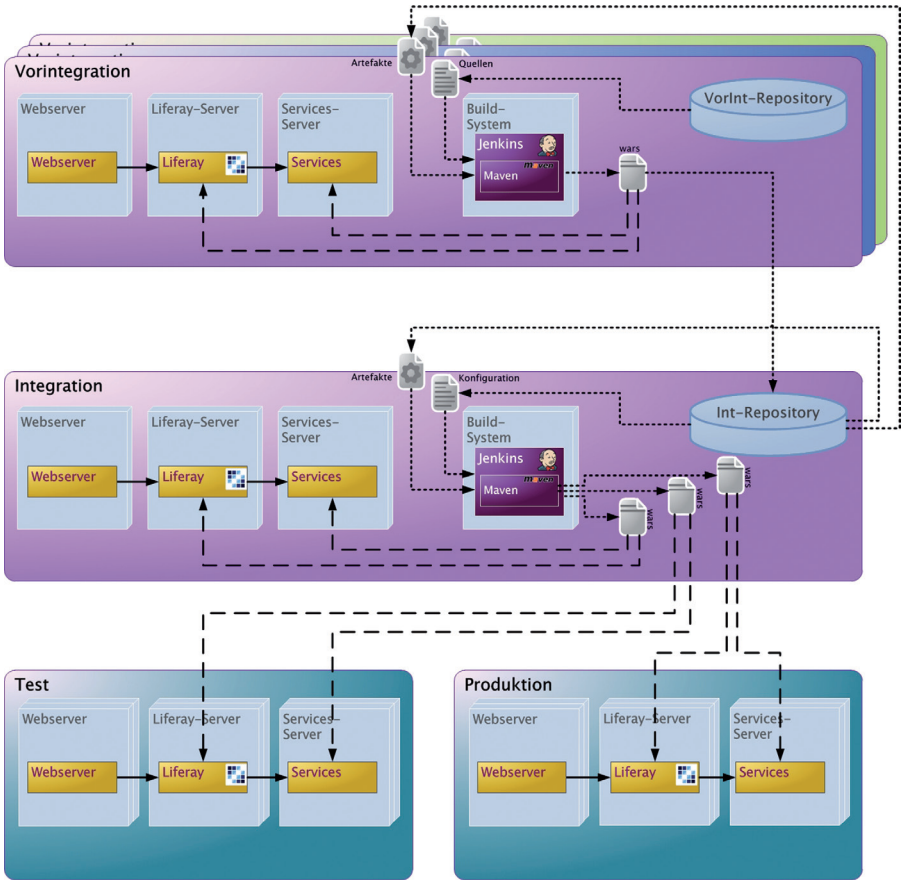

Abb. 1: Überblick über die Umgebungen.

Integrationsproblemen schnell einen Überblick verschaffen und erkennen, woher der Fehler kommen könnte.
Der letzte Schritt während der Vorintegration ist die Übergabe der neuen bzw. aktualisierten Artefakte an den Portal-Integrator. Es muss sich dabei nicht zwangsläufig um die aktuellste Version handeln, falls diese noch fehlerhaft oder unvollständig ist. Die Übergabemeldung geschieht über ein XML-Dokument, das die Artefakte und Versionen beschreibt, die zur Integration vorgesehen sind. Technisch handelt es sich hierbei um eine Maven-POM-Datei, die für jedes auszuliefernde Artefakt eine Abhängigkeit definiert hat.

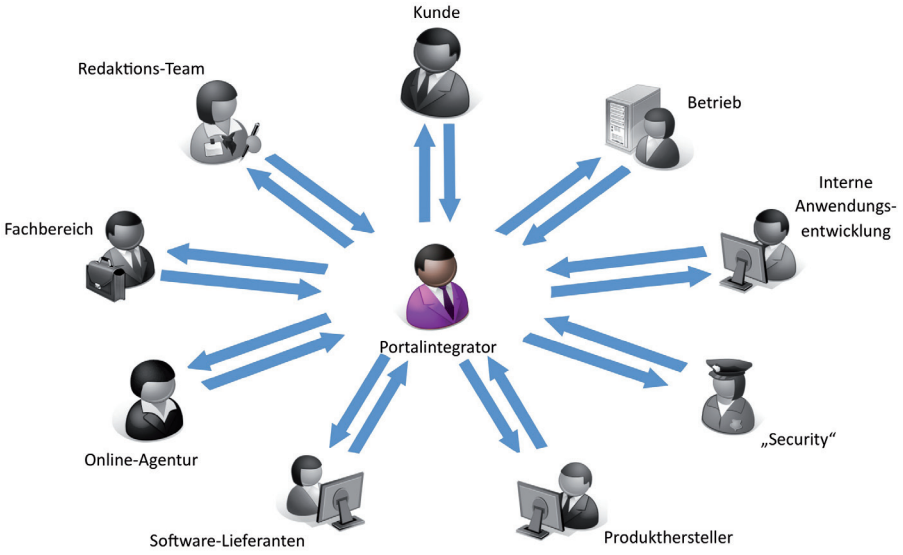

Abb. 2: Die Rolle des Portal-Integrators.

Integration
Der Portal-Integrator betreut die zentrale Umgebung für die Integration aller Zulieferungen. Diese Integrationsumgebung sollte der Test- und Produktionsumgebung möglichst ähnlich sein, wobei gewisse Einschränkungen bei Serverleistung oder Clustering akzeptabel sein können.
Bei der Integration werden die gesamten Abhängigkeiten, die von den Software-Lieferanten definiert wurden, zu einer kompletten Auslieferung aggregiert und verpackt. Neben der Auslieferung für die Integrationsumgebung lassen sich hier mit den gleichen Abhängigkeiten auch Test- und Produktionsauslieferungen erstellen. Diese Auslieferungen unterscheiden sich z. B. bezüglich der konfigurierten Datenbank-Anbindung. Auch die Art der Auslieferungen unterscheidet sich, denn während sich auf die Integrationsumgebung vollautomatisch deployen lässt, ist dies auf Test und Produktion nicht gewünscht, da für diese Umgebungen in der Regel andere Vorgaben für Zugriffsberechtigung und Verantwortung gelten. Für Test und Produktion werden daher die Auslieferungen in einem Ordner bereitgestellt.
Die weitgehende Automatisierung ist in der Integration aus unserer Erfahrung Pflicht. Die komplette Auslieferung sollte sich durch das Starten eines einzelnen Jobs erstellen lassen. Das nachträgliche Bearbeiten von Artefakten ist auf jeden Fall zu vermeiden, weil sonst der Prozess durch die manuellen Schritte unterbrochen wird und dies eine große Fehlerquelle sein kann.

Integrationstests
Bis zu diesem Punkt wurden die Zulieferungen alle einzeln oder in kleinem Umfeld getestet. Während der Integration finden nun auch die ersten Tests des Gesamtsystems statt. Es kann immer Situationen geben, in denen es Wechselwirkungen zwischen Artefakten gibt oder bei denen z. B. neue Style-Definitionen nicht überall passen. Auch die Anbindungen an die Backend-Systeme sind erst in der Integrationsumgebung gewährleistet.
In unserem Fallbeispiel-Projekt hat es sich etabliert, bei jeder Integration einmal quer über das Portal zu testen. Wir prüfen die gesamte Funktionalität, rufen jedes Artefakt einmal auf und stellen fest, ob es funktioniert. Die Kommunikation mit Backend-Systemen wird ebenso getestet wie die Schnittstellen zu externen Diensten. Gibt es hierbei technische Probleme oder finden wir Fehler, beheben wir diese, bevor der Fachbereich das erste Mal testet. Auf diese Weise vermeiden wir Frust bei den Testern, weil nichts funktioniert. Auch überprüfen wir, ob die Vorgaben bezüglich Konfiguration oder Logging von den Software-Lieferanten eingehalten wurden.

Best Practice 6: Reden, reden, reden

In Projekten mit vielen Stakeholdern ist es sehr wichtig, dass die Sorgen und Wünsche aller beachtet werden. Es ist aber nicht immer ganz einfach, die Anforderungen aus dem Fachbereich oder dem Betrieb mit den Produkt- oder Software-Lieferanten abzustimmen.
Dies gilt insbesondere in Portalprojekten wie in unserem Fallbeispiel. Die Etablierung der Rolle des Portal-Integrators, die für das gesamte Portal an sich Verantwortung trägt, hat sich sehr bewährt.
Als Portal-Integrator haben wir bereits aufgrund unserer Aufgaben direkt mit allen am Projekt beteiligten Stakeholdern beim Kunden sowie allen weiteren Dienstleistern zu tun (**siehe Abbildung 2**). Wir erfahren also aus erster Hand die Probleme und Anforderungen aller Beteiligten und helfen bei ihrer Bewertung.
In der Regel gibt es in einem Portal mehrere Wege, wie man etwas umsetzen kann, und viele, wie man es nicht umsetzen sollte, auch wenn es technisch möglich ist. An dieser Stelle muss der Portal-Integrator auch Architekt sein. Wir helfen dem Kunden, seine Anforderungen auf sein Portal abzustimmen, und machen Vorgaben an die Dienstleister, wie die Komponenten ins Portal zu integrieren sind und welche betrieblichen Anforderungen erfüllt werden müssen.
In der Rolle des Portal-Integrators nehmen wir einen zentralen Platz in der Kommunikation und in der Auslieferung der Software ein – und zwar immer mit dem Ziel, alle Projektteilnehmer zu entlasten, damit diese sich auf ihre Kernaufgaben konzentrieren können.

Fachbereich
Als Portal-Integrator sind wir erster Ansprechpartner des Fachbereichs, wenn beim Test etwas nicht funktionieren sollte. Um-

gekehrt brauchen wir vom Fachbereich Informationen, was zu testen ist und wie gegebenenfalls bestimmte Testkonstellationen herbeigeführt werden können. Wir kümmern uns zusammen mit dem Fachbereich auch um die Release-Planung (Umfang und Termine), wenn es dafür keine eigene Rolle gibt. Im Hinblick auf den Release-Umfang stimmen wir ab, welche Teile aus fachlicher, technischer oder Integrationssicht wann bereitgestellt werden müssen. Es kann zum Beispiel fachlich oder technisch sehr komplexe Artefakte geben, die zur Risikominimierung möglichst früh getestet werden müssen, obwohl es aus Integrationssicht erst später notwendig wäre. Ebenso gibt es Artefakte, bei denen aus Integrationssicht ein größeres Risiko besteht, dass sie nicht funktionieren. Hier ist eine enge Abstimmung mit allen Beteiligten notwendig.
Sollten bei den fachlichen Tests Fehler oder Probleme auftreten, können wir den Fehler analysieren und einzelnen Artefakten zuordnen. Dann können wir prüfen, ob die Konfiguration korrekt ist und gegebenenfalls den Lieferanten des Artefakts kontaktieren oder dem Fachbereich bestätigen, dass es sich um einen fachlichen Fehler handelt, und entsprechende Logauszüge zur weiteren Analyse bereitstellen.

Dienstleister

Wenn mehrere Dienstleister (interne Anwendungsentwicklung, Software-Lieferanten, Online-Agentur) am Projekt beteiligt sind, deren Entwicklungsaufgaben Abhängigkeiten haben – was in der Regel der Fall ist –, muss der Portal-Integrator für klare Abgrenzungen sorgen. Das betrifft weniger die Schnittstellen zwischen den Dienstleistern als vielmehr die übergreifenden Portal-Vorgaben. Beim Portal-Design gilt es, Entscheidungen zum Seitenaufbau und zur Seitenstruktur, zur Verwaltung von Inhalten, zu Rollen und Rechten, zum Schnitt der Artefakte usw. zu treffen und durchzusetzen. Insbesondere wenn ein Dienstleister weniger Portal-Erfahrung hat, ist immer darauf zu achten, dass die Lösungen zu der Portal-Architektur passen. Es ist besser, vorab die Randbedingungen festzulegen und den Dienstleistern beim Aufbau der Vorintegrationsumgebungen zu helfen, sodass diese auch dem Zielsystem entsprechen, als erst bei der Integration zu prüfen, ob die Vorgaben eingehalten wurden.

IT-Betrieb

Ein weiterer, sehr wichtiger Partner des Portal-Integrators ist der Betrieb. Nach der Abnahme muss das Gesamtsystem vom Betrieb übernommen werden. Darum muss man frühzeitig über die Anforderungen des Betriebs Bescheid wissen. Gibt es beispielsweise Vorgaben für die Installation, Konfiguration oder Auslieferung? Sollen alle Artefakte in einer Auslieferung enthalten sein oder sollen nur Deltas ausgeliefert werden? Welche Dokumentation wird benötigt?
Der Portal-Integrator muss die Anforderungen sammeln und an alle betroffenen Lieferanten kommunizieren. Auch ist es sehr hilfreich, in den Integrationstests zu prüfen, ob die Betriebsanforderungen eingehalten wurden. Das vermeidet den typisch hektischen Aktionismus kurz vor Projektende, wenn auffällt, dass bis dato niemand diese Anforderungen geprüft hat.
In diesen Bereich gehört zudem die Abstimmung mit den Infrastruktur-Teams. Welche Firewall-Regeln sind notwendig? Wie müssen die Load-Balancer konfiguriert werden? Was ist mit Sicherheitsvorgaben? Wo findet die SSL-Terminierung statt? Was passiert mit den Log-Dateien?

Querschnitt

Beim Kunden gibt es noch weitere Querschnitts-Aufgabenbereiche, die vom Portal-Integrator eingebunden werden können oder müssen. Dazu gehören z. B. Qualitätssicherung, Training, Release-Management, Partnersysteme und Compliance. Diese sind alle bei der Integration oder spätestens beim Test mit einzubeziehen.

Fazit

Nach unserer Erfahrung wurde im Fallbeispiel-Projekt die Umsetzung des neuen Portals durch den Funktionsumfang und die Anpassbarkeit des Liferay Portal Servers deutlich erleichtert. Voraussetzung war allerdings die Bereitschaft des Kunden, die Stärken der Software optimal auszunutzen (Nutzung der Bordmittel) und dafür gegebenenfalls die Anforderungen zu überdenken. Eine Individualentwicklung wäre aufwändiger gewesen, ohne ein besseres Ergebnis zu garantieren.
Die Flexibilität in den Anforderungen und die Konzentration auf Bordmittel brachten signifikante Vorteile bei der Wartung und Weiterentwicklung. Ebenso nützlich war und ist die aufgebaute Entwicklungs- und Deployment-Infrastruktur, die die Zulieferungen möglichst automatisiert zusammenführt.
Die Rolle des Portal-Integrators bestand einerseits darin, als „Zuchtmeister“ die Anforderungen, die Architektur und die Implementierung zu steuern und auf die Qualität zu achten. Andererseits fungierte der Portal-Integrator als zentrale Anlaufstelle bei Problemen sowie als Helfer und Kümmerer.
Die Strukturierung und Steuerung sowie die Unterstützung haben entscheidend zu einer Entlastung aller Beteiligten im Projekt beigetragen. ||

Die Autoren

|| Florian Schäfer
(florian.schaefer@iteratec.de)
ist IT-Architekt bei iteratec. Er unterstützt Kunden bei Konzeption, Design und Implementierung von IT-Anwendungen. Seine Schwerpunkte sind Java-Entwicklung und Liferay Portal Server in allen Projektphasen, von der Konzeption bis zum Betrieb.

|| Tjark Kalow
(tjark.kalow@iteratec.de)
ist Senior IT-Management-Berater bei iteratec. Er unterstützt Kunden bei der Konzeption und Umsetzung von IT-Projekten. Seine Schwerpunkte sind neben Facharchitektur und technischen Assessments das Anforderungsmanagement, für das er iteratec-interner Trainer ist.