mehr zum thema:
www.register-factory.de

REGISTER FACTORY: EINE REFERENZARCHITEKTUR IM PRAXISEINSATZ

„Viele Köche verderben den Brei", ist ein bekanntes Sprichwort. Davon, dass dieses auch für Software-Anwendungslandschaften gilt, können viele CIOs aus leidvoller Erfahrung berichten. Jeder neue Dienstleister legt für seine Software eine eigene Architektur fest und bringt eine Vielzahl von Tools und neuen Bibliotheken mit. So gleicht die Anwendungslandschaft oftmals nach wenigen Jahren einem Sammelsurium der Architektur-Kuriositäten. Die Folge: Wartung und Weiterentwicklung sind nur noch von Experten und Spezialisten zu leisten. Abhilfe kann hier ein geeignetes Architekturmanagement schaffen. Ein wesentlicher Baustein für die Vereinheitlichung der Anwendungslandschaft ist die Verwendung einer Referenzarchitektur, wie beispielsweise die in diesem Artikel vorgestellte Register Factory.

die autoren

Simon Spielmann
(simon.spielmann@capgemini.com)
hat große Teile der Register Factory gestaltet. Als Architekt für Organisationen des öffentlichen Bereichs sorgt er bei Capgemini für deren Umsetzung und Weiterentwicklung.

Frank Dörr
(frank.doerr@capgemini.com)
hat als Architekt wesentliche Teile der Register Factory entworfen. Er verantwortet bei Capgemini die technische Umsetzung von Softwaresystemen und hat mehrere auf der Register Factory basierende Projekte erfolgreich umgesetzt.

Felix Senn
(felix.senn@capgemini.com)
hat mehrere IT-Systeme auf Basis der Register Factory entwickelt. Er ist bei Capgemini für die langfristige Weiterentwicklung und Wartbarkeit der neuen Systeme verantwortlich.

Wofür Referenzarchitekturen?

Architekturen strukturieren Softwaresysteme. Damit legen sie die konkrete Ausgestaltung und Umsetzung wichtiger Aspekte in Softwaresystemen fest. Die Architektur soll dabei sicherstellen, dass nicht nur die funktionalen, sondern insbesondere auch die nicht-funktionalen Anforderungen erfüllt werden. Wichtige nicht-funktionale Anforderungen sind zum Beispiel: Leistung, Sicherheit, Wartbarkeit, Betreibbarkeit, Wirtschaftlichkeit und Testbarkeit.

Um Softwaresysteme wirtschaftlich und in hoher Qualität zu entwickeln, sollten IT-Architekten das Rad nicht jedes Mal neu erfinden. Das zu verhindern, ist die Aufgabe von Referenzarchitekturen, indem sie eine grundlegende Struktur für eine Klasse von Softwaresystemen festlegen.

Die *Register Factory* (vgl. [BVA]) beinhaltet solch eine Referenzarchitektur für Softwaresysteme im öffentlichen Bereich. Sie umfasst Vorgaben und Lösungen zu typischen Fragestellungen bei der Entwicklung von Softwaresystemen auf der Basis einer *serviceorientierten Architektur (SOA)*. Dabei wird die Architektur auf der Ebene der technischen Infrastruktur, der Ebene der Softwarearchitektur eines einzelnen Softwaresystems und auf der Ebene der Anwendungslandschaft betrachtet.

Im Folgenden erläutern wir zunächst das Konzept und die Inhalte der *Register Factory*. Anschließend stellen wir vor, welche Lösungsansätze die *Register Factory* für die Umsetzung der oben genannten Aspekte verfolgt.

Was ist die „Register Factory"?

Die *Register Factory* wurde im Auftrag des Bundesverwaltungsamts gemeinsam mit der Firma Capgemini entwickelt. Ziel war es, eine Grundlage zu schaffen, um Softwaresysteme der öffentlichen Verwaltung – insbesondere Register – effizient entwerfen, realisieren und betreiben zu können. Register dienen der Verwaltung von strukturierten Daten, die ein bestimmtes Merkmal verbindet. Beispiele dafür sind das Kraftfahrzentralregister oder das Handelsregister.

Die *Register Factory* ist als Baukasten konzipiert. So können einzelne Bestandteile isoliert verwendet oder auch eine vollständige Anwendungslandschaft mit einer einheitlichen Betriebsplattform für die Softwaresysteme auf Basis der *Register Factory* geschaffen werden. Teile, die für den geplanten Einsatz nicht zweckmäßig sind oder die mit bestehenden Vorgaben kollidieren, können weggelassen oder angepasst werden.

Die Inhalte der *Register Factory* decken fünf Bereiche ab: Blaupausen, Bausteine, Betriebsplattform, Methodik und Werkzeuge (siehe Abbildung 1). Dabei halten sich Codeanteile und konzeptionelle Anteile in etwa die Waage.

Blaupausen

Blaupausen beschreiben die Architektur und Konzepte von Softwaresystemen sowie deren Einbettung in die Anwendungslandschaft aus drei verschiedenen Sichten: der fachlichen Sicht, der Sicht der softwaretechnischen Umsetzung und der Sicht der technischen Infrastruktur.

Bausteine

Bausteine sind wiederverwendbare Softwarelösungen, zum Beispiel Funktionen zur Protokollierung oder zur Verwaltung von Anwendungskonfigurationen. Diese Bausteine liegen in unterschiedlichen Formen vor: Es gibt fachliche und technische Services im Sinne einer SOA sowie wiederverwendbare Bibliotheken und Programmiervorlagen. Dazu gehören auch am Markt verfügbare Fertigprodukte, für

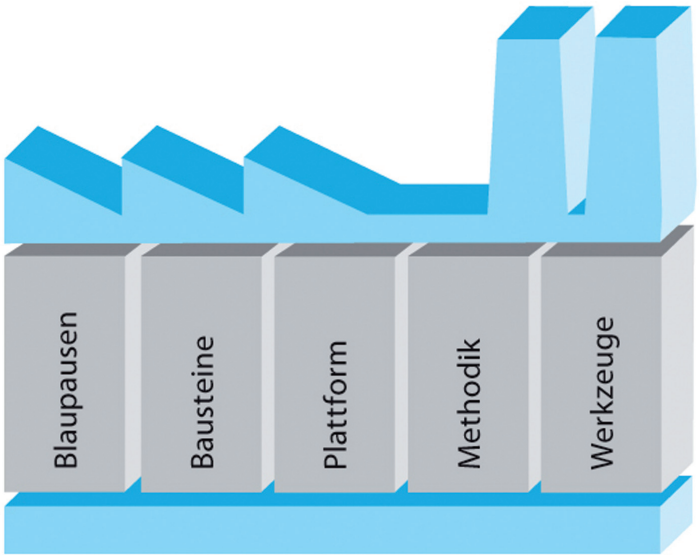

Abb. 1: Inhalte der Register Factory.

die detaillierte Nutzungsvorgaben erstellt werden. Fertigprodukte werden in einem standardisierten Produktauswahl-Verfahren ausgewählt. Dies dient insbesondere dazu, die Qualität der ausgewählten Produkte sicherzustellen. Dabei wird bewusst Open-Source-Produkten der Vorzug gegeben, um herstellerunabhängig zu sein und Lizenzkosten zu sparen.

Betriebsplattform
Es wird eine hochverfügbare Betriebsplattform definiert. Hierzu werden sowohl die Strukturen auf der Netzwerkebene festgelegt als auch die Anforderungen an die benötigte Hardware, die Netzkomponenten und die Middleware mit Anwendungs- oder Datenbankservern. Ziel dieser Festlegungen ist es, eine über alle Softwaresysteme einheitliche Betriebsplattform zu erreichen. So wird ein standardisierter und effizienter Systembetrieb ermöglicht.

Methodik
Die Methodik der *Register Factory* bietet verschiedene Hilfsmittel zu allen Phasen des Software-Entwicklungsprozesses an. Das sind zum Beispiel Vorlagen für die Spezifikation, Modellierungsrichtlinien und Vorlagen für den Systementwurf sowie Programmierkonventionen für Java. Es werden keine Vorgaben zum Projektvorgehen selbst gemacht. Da die *Register Factory* in erster Linie eine Referenzarchitektur definiert, können konkrete Projekte mit allen Vorgehensmodellen zur Softwareentwicklung umgesetzt werden. Die bisher von uns realisierten Projekte verwenden zum Beispiel ein Vorgehensmodell, das konform zum V-Modell-XT ist.

Werkzeuge
Die *Register Factory* setzt auf Automatisierung und Werkzeugunterstützung bei der Erstellung von Softwaresystemen. Dazu bietet sie vorkonfigurierte Werkzeuge – zum Beispiel für die Modellierung, die Programmierung, Tests oder die Fehlerverfolgung – an. Im Folgenden erläutern wir, wie die oben exemplarisch genannten, nicht-funktionalen Anforderungen von der *Register Factory* abgedeckt werden.

IT-Sicherheit als Teil der Architektur

Die IT-Sicherheit ist insbesondere dann für Softwaresysteme eine wichtige Anforderung, wenn diese personenbezogene Daten verarbeiten. Solche Systeme stellen hohe Anforderungen an Vertraulichkeit, Integrität und Verfügbarkeit. Eine Referenzarchitektur muss als grundlegendes Element der IT-Sicherheitsstrategie die Einhaltung solcher Schutzziele unterstützen.

Die *Register Factory* legt die grundlegenden Sicherheitsmechanismen für die Betriebsplattform und die Softwarearchitektur eines Softwaresystems fest. Dabei werden bewährte und allgemein anerkannte Mechanismen eingesetzt. Damit wird bereits der Schutzbedarf für die Verarbeitung normaler personenbezogener Daten erreicht. Zusätzlich wird für jedes zu entwickelnde Softwaresystem ein Sicherheitskonzept angefertigt. Wird dabei erkannt, dass für dieses Softwaresystem ein höherer Schutzbedarf nötig ist, so werden die hierin definierten Maßnahmen im Rahmen der Konstruktion und Realisierung dieses Softwaresystems umgesetzt. Sicherheitskonzepte können zum Beispiel nach BSI-Grundschutz (vgl. [BSI]) erstellt werden.

Auf der Ebene der Betriebsplattform unterscheidet die *Register Factory* zunächst zwischen interner und externer Kommunikation. Eine interne Kommunikation findet nur zwischen den Anwendungen innerhalb der Betriebsplattform statt, eine externe Kommunikation mit Anwendungen außerhalb der Betriebsplattform. Auch legt die *Register Factory* fest, dass die Fachanwendungen innerhalb der Betriebsplattform nicht direkt mit Anwendungen außerhalb der Betriebsplattform kommunizieren dürfen. Der Zugriff auf die Fachanwendungen erfolgt ausschließlich über web-basierte Dialoge und Serviceaufrufe. Dabei werden Dialoge ausschließlich über ein Portal und Services ausschließlich über Service-Gateways angeboten.

Unterstützt wird diese Strukturierung durch die Betriebsplattform auf der Netzwerkebene. Hier legt die *Register Factory* die Unterteilung in Sicherheitszonen gemäß dem SAGA-Standard (vgl. [SAG]) fest (siehe Abbildung 2). Die Web-Server und Service-Gateways stehen dann in der Informations- und Dienstezone, die Fachanwendungen in der Logik- und Verarbeitungszone, Datenbankserver in der Datenzone. Der Zugang zu Anwendungen der Betriebsplattform ist damit auch technisch nur über die Informations- und Dienstezone möglich.

Für die Authentifizierung und Autorisierung werden Vorgaben gemacht. Benutzer und externe Dienste müssen sich per Benutzerkennung, Kennwort und Zertifikat authentifizieren. Die Autorisierung erfolgt über ein rollenbasiertes Berechtigungssystem. Innerhalb der Plattform wird ein zentraler *Access-Manager* eingesetzt, der ein *Single Sign On (SSO)* umsetzt und die Zugriffe auf einzelne Anwendungen autorisiert. Innerhalb der Anwendungen erfolgt dann eine fein-granulare Autorisierung des Zugriffs auf die Funktionalität.

Auch auf der Ebene der Softwarearchitektur gibt es Vorgaben zur Ab-

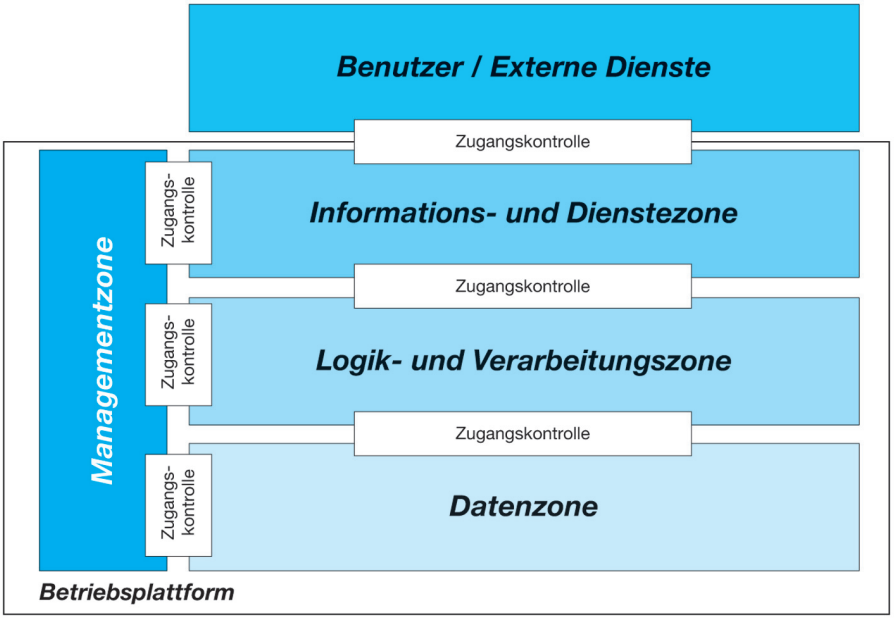

Abb. 2: Netzwerkzonen nach SAGA.

sicherung. So wird zum Beispiel festgelegt, dass Datenbankabfragen ausschließlich über *Named-Queries* erfolgen dürfen, um *SQL-Injection* zu verhindern. Des Weiteren existieren strikte Vorgaben zur Prüfung von Eingabewerten.

Softwaresysteme im Betrieb

Der Aspekt der Betreibbarkeit wird beim Entwurf eines Softwaresystems oft nicht genug oder zu spät berücksichtigt. Damit moderne Rechenzentren wirtschaftlich arbeiten können, ist ein hoher Automatisierungsgrad für den Betrieb von Softwaresystemen unerlässlich. Das bedeutet, dass Softwaresysteme idealerweise darauf ausgerichtet sind und dass dies bereits in der Architektur des Softwaresystems verankert sein sollte. Die *Register Factory* legt hierfür Standards fest. Sie definiert für alle Anwendungen eine einheitliche Überwachungsschnittstelle, wobei die Überwachung auf dem JMX-Standard basiert.

Weiterhin enthält die *Register Factory* ein standardisiertes Konzept zum Umgang mit Ausnahmesituationen (Störungen). Hier wird festgelegt, wie Anwendungen mit Fehlern umgehen und wie Fehlermeldungen und Fehlercodes aufgebaut sind. Für das Logging wird „Log4J" eingesetzt und ein standardisiertes, für den Betrieb leicht auswertbares Format für Log-Meldungen verwendet.

Für Batches wurde ein Batch-Rahmen in Form eines Bausteins erstellt. Damit wird sichergestellt, dass alle Batches standardisierte Return-Codes zurückgeben und die Batches auf eine einheitliche Art und Weise aufgerufen werden können.

Alle Anwendungen nutzen als Laufzeitumgebung „Apache Tomcat". Es wurden Vorgaben für die Konfiguration festgelegt und ein Default-Tomcat in Form eines RPM-Pakets erstellt. Alle Softwaresysteme werden ebenfalls als *RPM*-Paket (*RPM Package Management*) dem Betrieb übergeben und einheitlich konfiguriert. So ist es einfach, ein Softwaresystem neu zu installieren und in Betrieb zu nehmen.

Eine leistungsfähige Architektur

Mit Hilfe der *Register Factory* wurden bereits über 20 Softwaresysteme umgesetzt, wobei viele dieser Systeme ein großes Datenvolumen und eine große Anzahl von Zugriffen haben. Ein solches Softwaresystem verwaltet z. B. einen mehrere 100 Gigabyte großen Datenbestand mit circa 100 Millionen Datensätzen, die große Anzahl von Nutzern führt rund 100.000 Zugriffe täglich durch. Bei solchen Systemen ist auf eine hohe Leistungsfähigkeit und Skalierbarkeit zu achten. Entscheidend hierbei ist, dass Register-Factory-Services grundsätzlich zustandslos sind und dadurch insbesondere die Skalierbarkeit unterstützen.

Performante Service-Kommunikation

Die Betriebsplattform verwendet für interne und externe Service-Kommunikation unterschiedliche Kommunikationsmechanismen. Extern werden typischerweise SOAP-Web-Services per HTTP oder SMTP angeboten. Während diese Technologie für die interoperable organisationsübergreifende Kommunikation gut geeignet ist, ist sie hinsichtlich der Performance nicht optimal. Web-Services sind vergleichsweise aufwändig und wenig performant.

Innerhalb der Anwendungslandschaft erfolgt die Kommunikation daher über das schnelle HTTP-Invoker-Protokoll von Spring (vgl. [Spr]). Diese Technologie ist vergleichbar mit RMI, aber auf Grund der Verwendung von HTTP aus Sicht des Infrastrukturbetriebs viel besser handhabbar. Das Load-Balancing erfolgt über herkömmliche HTTP-Load-Balancer oder über Apache-Web-Server mit dem Zusatzmodul „mod_jk".

Für die externe Kommunikation bieten so genannte Service-Gateways SOAP-Web-Services an. Service-Gateways sind in der *Register Factory* eigene schlanke Softwaresysteme ohne Fachlichkeit, welche die Umsetzung des externen SOAP-Protokolls auf HTTP-Invoker vornehmen und die Authentifizierung der Nutzer über den zentralen *Access-Manager* übernehmen.

Drei Arten zur Nutzung von Anwendungen

Neben der Betriebsplattform macht die *Register Factory* konkrete Architekturvorgaben für die Umsetzung der einzelnen Softwaresysteme. Zur Steigerung der Leistungsfähigkeit der Softwaresysteme wird eine modifizierte Drei-Schichten-Architektur (siehe Abbildung 3) vorgegeben. Zu der bekannten Datenzugriffsschicht und dem eigentlichen Anwendungskern tritt eine differenzierte Nutzungsschicht. Hier wird zwischen GUI, Batch und Service unterschieden. Diese drei Elemente bieten die Dienste des Anwendungskerns, optimiert für die jeweilige Nutzungsart, an.

Zur Umsetzung von Dialogabläufen ist häufig eine komplexere mehrschrittige Transaktionssteuerung erforderlich, die auf den Dialogfluss der Anwendung zugeschnitten ist. Diese ist als Teil der GUI-Schicht zusammen mit den Dialogen implementiert. Die Service-Schicht implementiert die HTTP-Invoker-Services der Anwendung. Hier umfasst jede fachliche Transaktion – d. h. jeder Service-Aufruf –

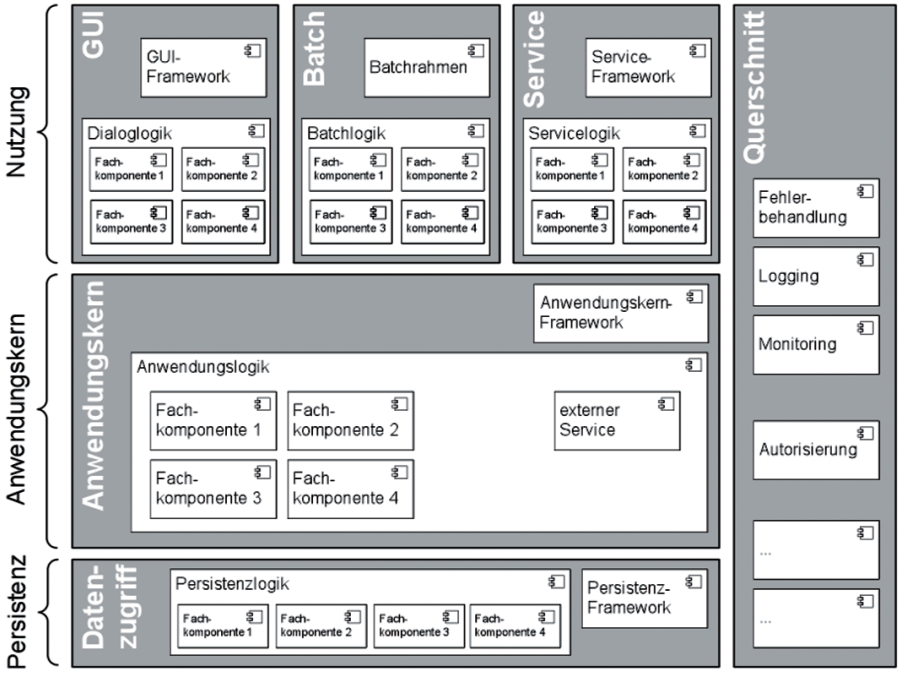

Abb. 3: Technische Architektur für Softwaresysteme der Register Factory.

genau eine technische Transaktion. Für Batches ist dies häufig nicht performant genug und daher nicht geeignet. Dort werden immer Blöcke von zum Beispiel 100 Geschäftsvorfällen innerhalb einer technischen Transaktion abgebildet. Jede Zugriffsart bietet so eine optimierte Nutzung der Anwendung.

Batches haben besondere Anforderungen

Softwaresysteme der *Register Factory* werden zweimal installiert: einmal für die Services beziehungsweise die GUI in einem Apache Tomcat und separat davon zur Massenverarbeitung im Batch-Betrieb. Während des Builds werden aus den Sourcen zwei Deployment-Pakete in Form von RPM-Dateien erzeugt. Beide Pakete enthalten identischen Code. Im Batch-Paket sind jedoch GUI- beziehungsweise Service-Schicht deaktiviert, dafür sind Skripte für den Start der Batches enthalten.

Die Batches umfassen also die Fachlichkeit des Anwendungskerns und müssen dafür keine Service-Aufrufe oder ähnliches durchführen. Sie kommunizieren direkt mit der zentralen Datenbankinstanz, die auch von der Service-Anwendung genutzt wird (siehe Abbildung 4). Die Batches werden auf eigenen Servern betrieben, damit sie die Performance der Service-Anwendungen nicht negativ beeinflussen.

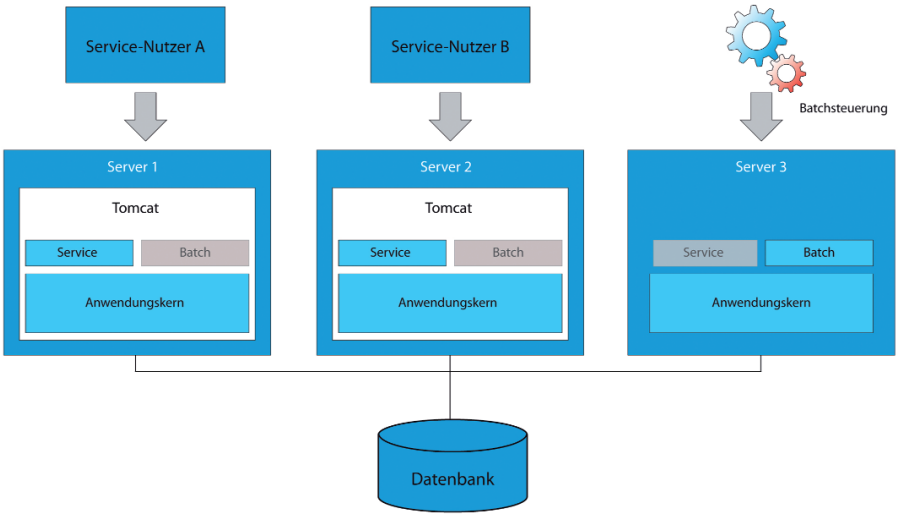

Abb. 4: Deployment von Batches.

Gute Architekturen müssen wirtschaftlich sein

Bei der Gestaltung von Standardarchitekturen stehen häufig technische Aspekte im Vordergrund. Um wirklich erfolgreich zu sein, muss eine Architektur aber auch kostengünstig umzusetzen sein.

Grundlage dafür ist eine einfache Architektur. Diese sorgt für gute Wartbarkeit, weniger Fehler und geringe Einarbeitungszeiten. Dieser Gedanke ist eines der Leitmotive der *Register Factory*: Komplexere Technologien wie *aspektorientierte Programmierung (AOP)* werden nur sparsam eingesetzt. Um den Entwicklungsprozess und die Entwicklungsumgebung möglichst einfach zu halten und die Verständlichkeit der Softwaresysteme nicht zu verschlechtern, ist die Architektur so gestaltet, dass sie grundsätzlich auch ohne Generatoren sehr effizient implementierbar ist.

Natürlich umfasst aber auch die *Register Factory* Teile, die sehr einfach strukturiert und daher gut generierbar sind. Entsprechend werden Generatoren für die Implementierung der Service- und Datenzugriffsschicht oder auch zur Erstellung der Steuerdateien für den RPM-Paketbau angeboten. Die Service-Gateways können fast vollständig automatisch generiert werden.

Der hohe Standardisierungsgrad der *Register Factory* ermöglicht es, Softwaresysteme sehr effizient umzusetzen. Die Homogenität der damit erstellten Softwaresysteme spart Aufwände für die Einarbeitung neuer Mitarbeiter, zum Beispiel für Wartungstätigkeiten, aber auch für die Fehlersuche. Der Technologie-Stack besteht aus weit verbreiteten, häufig auch standardisierten Technologien wie *JPA (Java Persistence API)*, sodass nur wenig Spezialwissen erforderlich ist.

Softwaresysteme müssen wartbar sein

Die Wartbarkeit von Software zeigt sich in dem Aufwand, der für Erweiterungen und Änderungen betrieben werden muss. Der Bedarf an Wartbarkeit steigt mit der

Nutzungsdauer der betroffenen Software. Die Qualität der Software bezüglich ihrer Wartbarkeit wird oft erst Monate oder gar Jahre nach dem Abschluss ihrer Entwicklung deutlich: dann nämlich, wenn sich fachliche und technische Anforderungen ändern, zentrale Entwickler und damit Wissensträger nicht mehr zur Verfügung stehen, die Kosten für kleine Änderungen nicht vertretbare Höhen erreichen oder sich bei jedem Release-Wechsel immer wieder viele neue Fehler einschleichen.

Der Grundstein für eine gute Wartbarkeit wird durch die Architektur der Software gelegt. Folgende Aspekte sind für eine langfristige Wartbarkeit wichtig.

Dokumentation
Die Dokumentation der Software ist über Spezifikation, Konstruktion und Implementierung durchgängig und eindeutig aufeinander abbildbar. Fachliche Anwendungsfälle aus der Spezifikation haben ihre Entsprechung in der Konstruktion als fachliche Komponenten und werden im Quelltext als eigenständige Klasse implementiert. Gibt es beispielsweise einen Anwendungsfall „Statistik erstellen", dann gibt es auch eine gleichnamige fachliche Komponente und im Quellcode wird man den Einstieg in den Anwendungsfall in der Klasse AwfStatistikErstellen finden. Bei der Kommunikation zwischen Fachbereich und Entwicklern ist der Transfer zwischen Fachlichkeit und Technik wesentlich einfacher – Missverständnisse werden so vermieden.

Modularer Aufbau
Die Software wird nach Vorgaben der *Register Factory* aus fachlichen und technischen Komponenten mit Hilfe von *Spring Dependency Injection* zusammengesetzt. Vorgaben regeln die Kommunikation zwischen den Schichten und den Komponenten. Ziel dieser Regelungen ist es, die Komponenten so zu entkoppeln, dass Änderungen im Rahmen der Wartung möglichst lokal und Seiteneffekte ausgeschlossen sind. Dadurch wird vermieden, dass Änderungen an der einen Komponente zu Fehlern in einer anderen fachlichen Komponente führen (vgl. [Sie04]).

Klarer Programmfluss
Die Strukturierung durch Schichten und Komponenten gibt bereits einen klaren Programmfluss vor, da höhere Schichten nur auf niedrigere zugreifen dürfen und auch für Komponenten eine eindeutige Zugriffsrichtung festgelegt ist.

Trennung von Fachlichkeit und Technik
Nahezu alle fachlichen Regeln werden durch Regelwerke der Regelmaschine „JBoss Drools" implementiert. Das aktuell größte Softwaresystem, das auf Basis der *Register Factory* entwickelt wurde, enthält knapp 4.000 Regeln, die hoch-performant ausgeführt werden. Die Regeln sind fachlich gut lesbar, damit leichter verständlich und somit weniger anfällig für Fehler. Damit wird ein klassisches Problem der Wartung gelöst: Expertenwissen in der Fachlichkeit, die durch das Regelwerk abgebildet wird, ist von dem Entwickler für die Weiterentwicklung nur noch in beschränktem Umfang notwendig.

Automatisierte Tests
Für die *Register Factory* wurde ein eigenes Test-Framework auf der Basis von JUnit erstellt. Die Testautomatisierung ermöglicht kürzere Release-Zyklen und sorgt dauerhaft für eine gleichbleibende Qualität.

Einsatz von Standards
Die *Register Factory* legt großen Wert auf die Standardisierung oder die Nutzung existierender Standards wie JPA und Standardwerkzeuge wie „Maven" oder „Eclipse". Der *Register Factory* liegt ein definierter Produktkatalog mit ausgewählten Bibliotheken und Produkten zu Grunde, der für jedes Produkt die Produktentscheidung nachvollziehbar macht. Dadurch wird ein technischer Wildwuchs verhindert. Gleichzeitig beschreibt die *Register Factory* im Rahmen der Methodik das Vorgehen bei der Produktauswahl und der Aktualisierung von neuen Versionen.

Softwaresysteme müssen testbar sein

Testbarkeit ist eine wichtige Anforderung an Softwaresysteme, die über einen langen Zeitraum gepflegt und weiterentwickelt werden oder die einem besonderen Maß an Qualität genügen müssen. Sie beschreibt den Aufwand, den Tester und Entwickler betreiben müssen, um Änderungen an der Software zu testen. Dabei stehen zwei Kernfragen im Mittelpunkt:

- Kann man die Funktionalität der Software testen?
- Wie hoch ist die Wahrscheinlichkeit, dass ein Test einen vorhandenen Fehler findet?

Diese zwei Kernprobleme werden durch eine hohe und gleichzeitig risikobasierte Testabdeckung gelöst. Dazu werden Softwaretests auf unterschiedlichen Abstraktionsebenen durchgeführt. Die *Register Factory* bringt eine Methodik und Werkzeuge für die Erstellung und Durchführung von Tests mit.

Es gibt Tests auf den Ebenen von Komponenten, Softwaresystemen und der gesamten Anwendungslandschaft. Die Methodik sieht vor, auf Komponentenebene mit vielen, aber sehr einfachen Tests eine hohe Testüberdeckung zu erreichen. Auf den höheren Ebenen werden Testfälle naturgemäß komplexer, da beispielsweise mehrere Anwendungen integriert werden müssen. Auf dieser Ebene sind dann nur noch wenige komplexe Tests notwendig. Ein vollkommener Test des Softwaresystems ist aufgrund der Komplexität mit vertretbarem Aufwand in der Regel nicht möglich. Daher wird zusätzlich eine Priorisierung der Tests vorgenommen. Die Funktionalität mit der größten Bedeutung für den Geschäftsprozess des jeweiligen Unternehmens wird am intensivsten getestet. Zur Messung der Testüberdeckung wird „Cobertura", eine Software zur Messung der Code-Überdeckung, eingesetzt.

Die *Register Factory* enthält ein Framework zur einfachen Erstellung von System- oder Komponententests auf der Basis von JUnit. Für Tests müssen nur der fachliche Ablauf und die Testdaten festgelegt werden. Die technischen Einzelheiten des Systemzugriffs und der Auswertung der Rückgabewerte werden durch das Framework bereitgestellt. Gleichzeitig sorgt das Test-Framework durch die Trennung von Testdaten und -ablauf für eine sehr gute Lesbarkeit und Wartbarkeit der Tests.

Teams sind flexibel einsetzbar

Bereits in der Entwurfsphase eines Softwaresystems unterstützen die im Rahmen der Methodik angebotenen Hilfsmittel den Softwarearchitekten. Sie vereinfachen die Erstellung gleichartig aufgebauter Entwurfsdokumente und Modelle mit einem gleichartigen Detaillierungsgrad.

Teams, die erstmalig ein auf der *Register Factory* basierendes Softwaresystem umsetzen, können sich schnell einarbeiten, da die *Register Factory* auf erprobten, allgemein anerkannten und verbreiteten Technologien basiert.

Die Praxis hat gezeigt, dass Teams, die bereits Erfahrung im Einsatz der *Register Factory* haben, auch in anderen Projekten in einer auf der *Register Factory* basierenden Anwendungslandschaft flexibel eingesetzt werden können. Hier bewährt es sich, dass sie sich schnell und vollständig auf die umzusetzenden fachlichen Anforderungen konzentrieren können. Der einheitliche Aufbau von Entwurfsdokumenten und Modellen ermöglicht dabei ein rasches und gutes Verständnis des neu zu realisierenden Softwaresystems.

Fazit

Mit Hilfe der *Register Factory* wurde bereits eine große Anzahl von Softwaresystemen realisiert – sie hat sich damit in der Praxis bewährt. Der wichtigste Erfolgsfaktor dabei ist, dass die *Register Factory* als Referenzarchitektur für die wesentlichen und immer wiederkehrenden Fragestellungen beim Entwurf solcher Softwaresysteme zuverlässige und bewährte Lösungen anbietet. Unsere praktische Erfahrung hat gezeigt, dass gerade auch kleine oder auf den ersten Blick triviale Lösungen großen Nutzen stiften können. Beispiele hierfür sind das Fehlerkonzept und die Vorgabe zur Vergabe von Fehlercodes, die dem Betrieb und dem Service-Desk die tägliche Arbeit erleichtern.

Die *Register Factory* basiert auf erprobten, allgemein anerkannten und verbreiteten Technologien. Dadurch wird die Einarbeitung neuer Mitarbeiter vereinfacht, das technologische und wirtschaftliche Risiko bei der Erstellung eines neuen Softwaresystems minimiert und eine gute Wartbarkeit erreicht. Eine Referenzarchitektur lebt und verändert sich kontinuierlich weiter – gemäß den sich wandelnden Anforderungen an Softwaresysteme und dem technologischen Fortschritt. Damit hier kein Wildwuchs entsteht und die Vorteile einer Standardarchitektur erhalten bleiben, ist ein gutes Architekturmanagement unerlässlich. ■

Links

[BSI] Bundesamt für Sicherheit in der Informationstechnik (BSI), BSI-Grundschutz: IT-Grundschutz, siehe: bsi.bund.de/DE/Themen/ITGrundschutz/itgrundschutz_node.html

[BVA] Bundesverwaltungsamt (BVA), Register Factory, siehe: register-factory.de

[SAG] Die Beauftragte der Bundesregierung für Informationstechnik, SAGA, siehe: cio.bund.de/DE/Architekturen-und-Standards/SAGA/saga_node.html

[Sie04] J. Siedersleben: Moderne Softwarearchitektur, dpunkt.verlag 2004

[Spr] SpringSource, What is Spring?, siehe: springframework.org