RED HAT®
CONSULTING

redhat.

WHITEPAPER

EIN STRUKTURIERTER DESIGNANSATZ FÜR DIE BUSINESS-AUTOMATISIERUNG

Fortschritte bei der Infrastruktur und den Methodologien der Softwareentwicklung haben Tools und Wege zur Modernisierung der Business-Automatisierung hervorgebracht. Ein Designansatz, mit dem Best Practices der modernen Softwareentwicklung auf die Business-Automatisierung angewendet werden, ermöglicht Organisationen die Ausschöpfung dieser Möglichkeiten, und zwar durch die Integration strikter technischer Verfahren bei gleichzeitiger Eliminierung zusätzlicher Risiken.

facebook.com/redhatinc
@redhat
linkedin.com/company/red-hat

de.redhat.com

ZUSAMMENFASSUNG

In vielen führenden Unternehmen ist es die Aufgabe der IT-Abteilung, Tools bereitzustellen, mit denen die Wettbewerbsfähigkeit aufrechterhalten, Industrierichtlinien erfüllt und eine langfristige Kundenbindung gewährleistet werden können. Die Lösungen sollen hochwertig sein, eine ausreichende Flexibilität für rasche und häufige Veränderungen bieten und zu vorhersehbaren und vorzugsweise niedrigen Kosten erhältlich sein.

Maßgeschneiderte Softwareentwicklung und Business-Automatisierung sind zwei Möglichkeiten, mit denen die von Business-Stakeholdern gewünschten Lösungen zur gewünschten Zeit bereitgestellt werden können. Allerdings bringen diese beiden Ansätze auch unterschiedliche Vor- bzw. Nachteile mit sich. Die Softwareentwicklung ist eine technische Disziplin mit rigorosen, aber traditionell langsamen Prozessen. Die Business-Automatisierung sorgt für eine schnellere Markteinführung, allerdings resultiert die Kodifizierung der Geschäftslogik durch nicht technische Stakeholder wiederum in einem erhöhten Risiko.

Fortschritte bei der Infrastruktur und den Methodologien der Softwareentwicklung haben Tools und Wege zur Modernisierung der Business-Automatisierung hervorgebracht. Ein Designansatz, mit dem Best Practices der modernen Softwareentwicklung auf die Business-Automatisierung angewendet werden, ermöglicht Organisationen die Ausschöpfung dieser Möglichkeiten, und zwar durch die Integration strikter technischer Verfahren bei gleichzeitiger Eliminierung zusätzlicher Risiken. Über diesen Ansatz der Business-Automatisierung können Unternehmen die Kodifizierung der Geschäftslogik von Experten durchführen lassen und so eine höhere Qualität, schnellere Markteinführung sowie mit moderner Softwareentwicklung verbundene niedrigere und vorhersehbare Kosten gewährleisten.

DIE EVOLUTION VON BUSINESS-AUTOMATISIERUNG UND SOFTWAREENTWICKLUNG

IT-Manager sind ständig auf der Suche nach neuen Produkten oder Technologien, die sich zur Erfüllung ihrer Geschäftsanforderungen eignen. Mit gewerblicher Software aber können Unternehmen oft keine schnellen Innovationen realisieren oder sich von ihren Wettbewerbern abheben, weshalb IT-Teams zur Entwicklung kundenspezifischer Software gezwungen sind.

Die Softwareentwicklung war schon seit jeher ein extrem zeitaufwändiger und komplexer Prozess, der während der gesamten Projektdauer eine effiziente Kommunikation zwischen Geschäftsexperten und Entwicklern erforderte. Häufig im Einsatz war hierbei das Wasserfallmodell, bei dem die Geschäftsexperten zunächst den Entwicklern ihre Anforderungen mitteilten und letztere dann Design, Entwicklung und Prüfung der Anwendung ausführten. Allerdings konnte die Anwendung erst nach ihrer Fertigstellung bewertet werden, was die Implementierung notwendiger Änderungen schwierig und teuer machte.

Dann erschien jedoch die Business-Automatisierung, auch Geschäftsprozessautomatisierung genannt, als Alternative zur IT-basierten Softwareentwicklung auf dem Plan. Mit dieser Methode können Geschäftsexperten die benötigte Geschäftslogik schneller und iterativ definieren. Die Geschäftslogik definiert sich als eine Ansammlung von Prozessen, Regeln und Workflows und nicht als Anwendungsfunktionen. Diese Regeln werden von einer Engine als Reaktion auf spezifische Bedingungen oder Ereignisse ausgeführt.

So wird z. B. der Prozess des Onboardings neuer Kunden bei Finanzinstitutionen von den Behörden auf allen Ebenen reguliert, um Geldwäsche oder andere illegale Aktivitäten zu verhindern. Allerdings sind diese Regulierungen häufigen Änderungen unterworfen. Ein Compliance-Spezialist bei einer Finanzinstitution kann mithilfe der Business-Automatisierung Bedingungen und dazugehörige Regeln für neue Regulierungen definieren. Diese neuen Regeln lassen sich dann ganz ohne die Unterstützung von Softwarearchitekten, Entwicklern oder IT-Mitarbeitern in eine Engine implementieren.

Obwohl Business-Automatisierung und kundenorientierte Softwareentwicklung das gleiche Ziel haben, also die Codierung von Geschäftslogik in ein Softwaresystem, wurden sie unabhängig voneinander eingeführt. Die daraus resultierende fragmentierte IT hat aufgrund der Duplizierung von Servern, Betriebssystemen und anderen Softwarelizenzen, Wartung und Support usw. zu höheren Kosten geführt. Dazu hat sich mit der Business-Automatisierung, genauer gesagt der Vermeidung der Beschränkungen der Softwareentwicklung und der daraus resultierenden Nichtanwendung strikter technischer Praktiken ungewollt das Gesamtrisiko erhöht. Will heißen, Softwareentwicklungsprinzipien wie die Änderungshistorie der Versionskontrolle, die Nachverfolgung getesteter und zur Produktionsfreigabe bereiter Komponenten sowie die Konfigurationsverwaltung – inklusive der Lebenszyklen vieler Komponenten – wurden dabei nicht unbedingt eingehalten.

In den letzten Jahrzehnten haben Softwaredesign und -entwicklung durchgreifende Änderungen erfahren. In vielen Fällen war dies eine Reaktion auf die gleiche Kritik, die ursprünglich für den Aufstieg der Business-Automatisierung verantwortlich war. Die Softwarearchitektur hat sich von Designs, die auf riesigen, monolithischen und schwer zu entwickelnden bzw. zu ändernden Anwendungen basieren, zu einer servicebasierten Architektur entwickelt. Diese beinhaltet Anwendungskomponenten, die für einen einzigen Zweck konzipiert und viel einfacher zu erstellen, aktualisieren und warten sind.

Die Methoden der Softwareentwicklung haben sich ebenfalls geändert. Das traditionelle Wasserfallmodell zeichnet sich durch sorgfältige Planung und getrennten Phasen aus, was die Wiederholung von Teilen des Prozesses schwierig und komplex macht. Diese Methode wurde mittlerweile durch iterative Ansätze abgelöst, die eine beschleunigte Implementierung der Anwendung, gemeinsame Prüfung mit den Beteiligten sowie Änderungen mit jeder Iteration vorsehen. Diese Frameworks, wie z. B. Extreme Programming, Scrum und das Scaled Agile Framework, wurden im Laufe der Zeit und als Reaktion auf technologische Fortschritte stetig verbessert. Heute sind es die CI/CD-Praktiken (Continuous Integration/Continuous Delivery), die einen schnellen und iterativen Ansatz im Hinblick auf die Bereitstellung ermöglichen. Diese Praktiken basieren auf der Idee, dass Code mehrere Male pro Tag integriert und im Verlaufe der gesamten Entwicklung getestet werden sollte, um eine flexible Release der Software zu ermöglichen.

EIN MODERNER DESIGNANSATZ FÜR DIE BUSINESS-AUTOMATISIERUNG

Viele Lösungen für die Business-Automatisierung lassen die Fortschritte in Sachen Softwaredesign und -entwicklung vermissen. Mit einem effizienten Designansatz für die Integration der Vorteile der modernen Softwareentwicklung in die Business-Automatisierung werden Designprobleme in einem Satz Architekturkomponenten organisiert, von denen jede mindestens eine wichtige Bereitstellungsentscheidung erfordert. Mit einer auf diesem Ansatz basierenden Umgebung können Organisationen die aktuelle Softwareentwicklungsinfrastruktur auf die Business-Automatisierung anwenden und so beide effektiv unterstützen.

Diese Architekturkomponenten entsprechen den primären Aktivitäten einer effektiven Business-Automatisierungsabfolge:

1. **Modellieren** der Geschäftslogik und der sie definierenden Testszenarien durch Entwicklung des Anwendungscodes oder Definition der grundlegenden Elemente der Business-Automatisierung: Regeln, Events, Prozesse und Workflows.
2. **Versionieren** der Modellartefakte, darunter eine prüfbare Änderungshistorie.
3. **Erstellen** eines einsatzfähigen Pakets aus einer Ansammlung an Artefakten sowie Ausführung von Tests und Protokollierung der Pakethistorie.
4. **Speichern** des Pakets in einem Master Repository.
5. **Bereitstellen** und Prüfung des Pakets in einer Ausführungsumgebung, wie z. B. einem Anwendungsserver für Code oder einer Engine für die Elemente der Business-Automatisierung.
6. **Ausführen** der Geschäftslogik.

In den folgenden Abschnitten werden die Architekturkomponenten und wichtigen Implementierungsentscheidungen beschrieben. Komplexität, Funktionen und Kosten einer Umgebung hängen von den für jede Komponente getroffenen Entscheidungen ab.

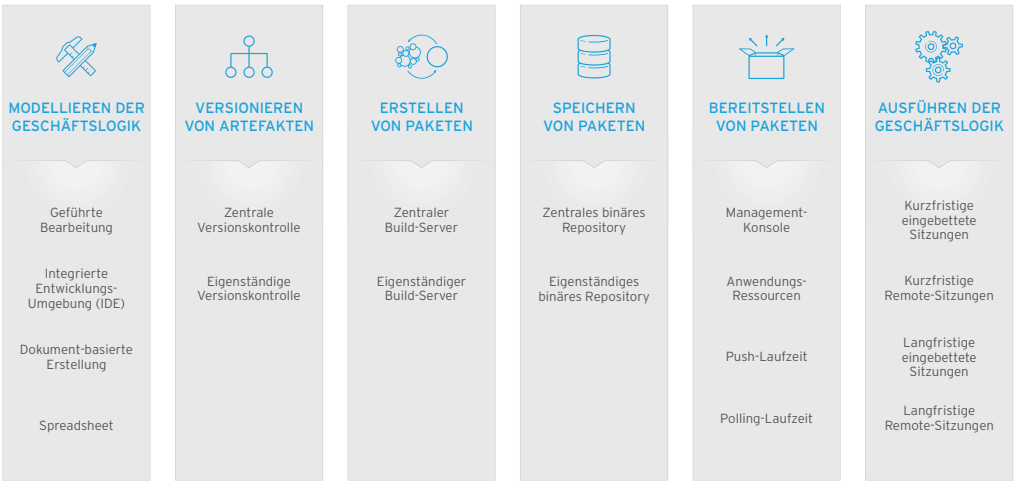

Abb. 1. Architekturkomponenten und dazugehörige Implementierungsentscheidungen für die Business-Automatisierung

1. MODELLIEREN

Unter Modellierung versteht man die Erfassung von Geschäftslogik zwecks Ausführung. Dabei modellieren Nutzer eine solche Geschäftslogik typischerweise basierend auf ihren Kenntnissen des erwarteten Systemverhaltens. Die Geschäftslogik kann mithilfe von Anwendungscode, Elementen der Business-Automatisierung oder einer Kombination aus beiden modelliert werden.

Bei der Methode mit dem Anwendungscode wird eine Programmiersprache eingesetzt. Viele Programmierer verwenden zum Schreiben von Code eine IDE (Integrated Development Environment), allerdings taugt dazu auch ein Texteditor, solange er die Erstellung von standardkonformem Code ermöglicht.

Bei der Business-Automatisierung modellieren Geschäftsexperten die Geschäftslogik durch die Erstellung von Regeln, Prozessen, Events und Workflows, und zwar unter Einsatz unterschiedlicher Methoden:

- **Tool zur geführten Bearbeitung.** Ein solches Tool bietet eine intuitive Schnittstelle, die technische Details ausblendet und keine technischen Kenntnisse erfordert. Mit seiner Hilfe kann der Nutzer Testfälle zwecks Validierung ihrer Akkuratheit definieren. Die geführte Bearbeitung erfüllt den ursprünglichen Zweck der Business-Automatisierung, nämlich die Bereitstellung eines Tools, mit dem Geschäftsexperten ohne Unterstützung von Programmierern nach Belieben modellieren können.
- **Spreadsheet.** Alternativ können Geschäftsexperten auch Spreadsheets für die Definition von Regeln als Entscheidungstabellen nutzen, die üblicherweise mit einem Lesemuster für die Zeilen konfiguriert werden. Geschäftsexperten kennen sich traditionell mit Spreadsheets aus und haben kein Problem mit der Bearbeitung der Zellenwerte zwecks Definition der gewünschten Logik. Spreadsheets können auch für die Prozessmodellierung eingesetzt werden.
- **Dokument-basierte Erstellung.** Bei diesem Ansatz wird die Struktur einer Regel oder eines Prozesses mit einer Vorlage definiert. Dabei werden Geschäftsdaten, die in einem Dokument wie z. B. einem Vertrag oder Werbeangebot gespeichert sind, auf die Vorlage angewendet und Regeln sowie Prozesse generiert.
- **Integrated Development Environments.** IDEs werden von Programmierern zum Schreiben von Code genutzt, können aber auch zur Erstellung von Business-Automatisierungselementen verwendet werden. IDEs erfordern ein Plug-in, mit dem Nutzer Business Rules und Prozessdiagramme erstellen können. Sie bieten aber auch zusätzliche Funktionen, wie z. B. die Syntaxhervorhebung, und stellen eine vertraute Arbeitsumgebung für den eher technisch orientierten Nutzer bereit.

Bei der Modellierung von Geschäftslogik können Geschäftsexperten BDD (Behavior Driven Development – verhaltensgetriebene Softwareentwicklung) unterstützend für die Entwicklung und letztendliche Prüfung der Anwendung einsetzen. Mit BDD lassen sich kurzfristige Szenarien erstellen, die das gewünschte Anwendungsverhalten in einer verständlichen Sprache beschreiben und den Anforderungen der Anwendung entsprechen. Diese Szenarien wiederum werden von Entwicklern oder auch Geschäftsexperten für die Modellierung der Geschäftslogik mithilfe der Business-Automatisierung verwendet. In den späteren Phasen des Entwicklungszyklus werden die gleichen Szenarien als Basis für die automatischen Anwendungstests genutzt.

2. VERSIONIEREN

Das Versionskontrollsystem ist das Master Repository der Geschäftslogik-Artefakte. Unabhängig davon, ob die Geschäftslogik als Anwendungscode oder Geschäftsregel erfasst oder welches Tool verwendet wird, die erstellten Artefakte müssen in jedem Fall gespeichert werden. Bei der Versionierung aber geht es um mehr als nur die Speicherung des Artefakts und die Protokollierung aller über die Zeit auftretenden Änderungen und ihrer Gründe sowie die Möglichkeit der Wiederherstellung einer spezifischen älteren Version. Mit ihr lassen sich auch mehrere Verzweigungen eines Artefakts zwecks Nachverfolgung unterschiedlicher Änderungssequenzen unterstützen.

Traditionell werden bei Produkten der Business-Automatisierung Datenbanken zur Speicherung von Artefaktversionen verwendet, allerdings bieten sie nicht alle Vorteile eines Versionierungstools. Standardmäßige dateibasierte Versionskontrollsysteme wie Git, Mercurial oder Apache Subversion werden üblicherweise für Anwendungscode genutzt. Viele Organisationen verwenden sie bereits zur Erstellung einer zentralen gemeinsamen Versionskontrolle. Der Einsatz eines bestehenden Versionskontrollsystems befreit Programmierer und Geschäftsexperten von Sorgen in Bezug auf Sicherheit oder Datenspeicherung und -sicherung. Wenn eine zentrale Versionierungsinfrastruktur allerdings nicht verfügbar, oder im Hinblick auf die Integration zu komplex oder zu teuer ist, können Geschäftslogik-Artefakte auch mit einem eigenständigen dedizierten Versionskontrollsystem gespeichert werden.

3. ERSTELLEN

Das Build-System ist der primäre Prozess zur Vorbereitung und Kombinierung von Artefakten in Pakete für den Release-Schritt. Der effizienten und einheitlichen Integration der Artefakte in das Paket, idealerweise ganz ohne oder nur mit geringem Zutun des Nutzers, kommt eine übergeordnete Bedeutung zu.

Diese Pakete werden üblicherweise mit einem Build-Automation-Tool wie Apache Maven erstellt. Dieses Tool ist nicht nur auf die Ausführung aller erforderlichen Befehle zum Abruf von Artefakten aus dem Versionskontrollsystem, sondern auch auf ihre Vorbereitung und Kombinierung für den Release-Schritt ausgelegt. Builds können so konfiguriert werden, dass bei jeder Speicherung neuer Artefakte im Versionskontrollsystem automatisch neue Pakete erstellt werden.

In dieser Phase wird im Rahmen der Continuous Integration ein automatischer Test der neuen Pakete durchgeführt. Hierbei werden in kürzester Zeit und noch vor der Versionierung zusätzlichen Codes und der Erstellung neuer Pakete zu behebende Integrationsfehler festgestellt. Aus diesem Grund sollte diese Phase des Release-Prozesses keine Pakete mit Integrationsfehlern hervorbringen.

Viele Organisationen verfügen neben einem Versionskontrollsystem auch über ein Build-Automation-System. Die Nutzung des aktuellen Systems entbindet die Nutzer von der Notwendigkeit und Verantwortung, ein solches installieren, konfigurieren und pflegen zu müssen. Dazu müssen Softwarepakete meist Firmenrichtlinien und externe Vorschriften erfüllen, die im zentralen Build-System bereits abgearbeitet wurden.

Wenn letzteres keine Unterstützung für Maven bietet oder andere Hindernisse für die Nutzung des bestehenden Systems vorliegen, kann ein separates Build-System verwendet werden. Dessen Erstellung resultiert allerdings in mehreren Build-Systemen, z. B. eines pro Anwendungsteam oder gar für jeden Programmierer und Geschäftsexperten.

4. SPEICHERN

Das Paket-Repository ist die Master-Quelle für einsatzbereite Softwarepakete. Letztere werden nicht selten mehrmals und für unterschiedliche Server in verschiedenen Rechenzentren bereitgestellt. Die Installation gespeicherter Pakete aus dem Repository gestaltet sich sehr viel effizienter und einheitlicher als ihre Neuerstellung mit jeder Bereitstellung und garantiert dazu die jederzeitige Auswahl des gleichen Pakets.

Wie bereits erwähnt, werden zur Erfüllung von Anforderungen in Bezug auf Sicherheit, Governance, regulatorische Compliance, Datenspeicherung und Prüfung oft zentrale Repositorys verwendet. Auch hier kann ein eigenständiges, mit Maven kompatibles Repository eingesetzt werden, allerdings führt dies zur Verwendung mehrere Repositorys im gesamten Unternehmen.

5. BEREITSTELLEN

Softwarepakete werden aus dem Paket-Repository in einer für den jeweiligen Pakettyp vorgesehenen Ausführungsumgebung bereitgestellt. So wird beispielsweise eine eigenständige binäre Anwendung zwecks Ausführung auf einen Server kopiert, während eine Enterprise Java-Applikation auf einen Anwendungsserver geladen wird. Business-Automatisierungspakete werden in der Business-Automation-Engine, also die Ausführungsumgebung für Regeln und Prozesse bereitgestellt.

Diese Pakete können ganz ohne oder nur mit wenig Zutun des Nutzers bereitgestellt werden. Auch wenn von einem Nutzer ausgelöst, so sind doch alle Bereitstellungsschritte vordefiniert und vorab validiert, um das Fehlerpotenzial gering zu halten. Dazu können Bereitstellungsregeln wie Prioritäten und Zeitpläne festgelegt werden.

Diese Regeln werden auch zur Definition der notwendigen Pre-Relase-Tests verwendet. Bevor Pakete für die Produktionsumgebung freigegeben werden, werden sie üblicherweise zwecks Prüfung in einer QS- und Vorproduktionsschicht bereitgestellt. Dieses Tests sind sehr viel umfassender als die automatischen Prüfungen der Build-Phase. So agiert z. B. der Prüfer in der QS-Phase als Nutzer, der die Funktionalität testet, während in der Vorproduktionsphase Lade- und Leistungstests durchgeführt werden. Die Pakete werden erst dann zur Produktion freigegeben, wenn alle Tests in diesen Schichten bestanden wurden.

Pakete werden vorzugsweise im Push-Verfahren implementiert, d. h. mithilfe eines automatisierten Prozesses eines Systems in die Ausführungsumgebung geladen. Bei einem CD-Ansatz (Continuous Delivery) erfolgt die Bereitstellung, sobald das Paket im Repository aktualisiert wurde oder auf die Einhaltung eines Zeitplans konfiguriert werden kann. Jede Ausführungsumgebung verfügt über die aktuelle Paketversion gemäß Bereitstellungsregeln, wodurch ein jeglicher Benutzereingriff überflüssig wird.

In einem ähnlichen Ansatz wird für die Bereitstellung der Pakete eine Verwaltungskonsole – üblicherweise eine Webanwendung mit Benutzerschnittstelle – eingesetzt. Der Nutzer konfiguriert die Parameter für die Bereitstellung, darunter Name, Version und Ausführungsumgebung, und löst den Vorgang manuell aus. Mit der Verwaltungskonsole wird das Paket dann per Push-Verfahren in die Ausführungsumgebung geladen. Es handelt sich hier also um einen vom Benutzer initiierten Push und nicht um einen automatischen Vorgang.

Der Einsatz eines Pull- oder Polling-Ansatzes ist hier ebenfalls denkbar. Dabei scannt die Ausführungsumgebung das Repository auf aktualisierte Pakete und ruft diese eigenständig ab. Dieser Ansatz sorgt für zusätzliche betriebliche Komplexität, weil dadurch eine engere Verbindung zwischen der Build- und Bereitstellungsphase hergestellt und das aktualisierte Paket unmittelbar nach seiner Erstellung im Pull-Verfahren abgerufen wird. Wenn z. B. mehrere Ausführungsumgebungen (wie in den Bereichen Entwicklung, Test und Produktion) das gleiche

Repository verwenden, wird von allen Umgebungen der aktuelle Build im Pull-Verfahren abgerufen. Zu diesem Zweck sind verschiedene Methoden verfügbar, allerdings werden hier Push-Mechanismen bevorzugt.

Für Artefakte der Business-Automatisierung ist zwecks Bereitstellung auch der Einsatz von Anwendungsressourcen denkbar. Bei diesem Ansatz werden die Artefakte zusammen mit der Ausführungs-Engine als Bundle in die Anwendung integriert. Mit dieser einfachen und einheitlichen Methode wird allerdings eine monolithische Anwendung erstellt. Die Bereitstellung ähnelt eher der für Pakete traditioneller Anwendungen als der für die Business-Automatisierung. Idealerweise sollte dieser Ansatz mit einem Push- oder Verwaltungskonsolen-Verfahren kombiniert werden.

6. AUSFÜHREN

Die Ausführungsumgebung integriert üblicherweise alle zur Ausführung der Geschäftslogik benötigten Server, Storage, Betriebssysteme, Netzwerke und anderen Systeme. Zum Zwecke dieses Dokuments wird hier nur die Business-Automatisierung berücksichtigt.

Die Überlegungen hinsichtlich der Bereitstellung für die Ausführungsumgebung unterscheiden sich von denen für die anderen Architekturkomponenten des in diesem Dokument beschriebenen Ansatzes. Der Modellierungsschritt richtet sich nach den Fähigkeiten und Präferenzen des Benutzers, während Versionierung, Build und Speicherung lediglich eine Entscheidung zum Standort des Repositorys erfordern, die dazu jederzeit geändert werden kann. Die Überlegungen zur Bereitstellung beeinflussen, wie die Pakete in der Ausführungsumgebung platziert werden, aber nicht ihre eigentliche Ausführung.

Die Umgebung beeinflusst die Ausführung der Geschäftslogik in Sachen Leistung, Skalierbarkeit und Zuverlässigkeit durch technisches Design. Die Designentscheidungen für diese Umgebung können für die Kundenzufriedenheit bzw. den Erfolg oder Misserfolg von größter Bedeutung sein.

Ebenfalls zu den wichtigen Designentscheidungen gehört die Frage, ob die Business-Automation-Engine im Rahmen der Anwendung verteilt oder als zentraler Shared Service eingerichtet wird. Bei einer eingebetteten Ausführung ist die Engine Teil einer Anwendung und dient ausschließlich deren Funktionalität. Anwendung und Engine werden zwecks Maximierung der Leistung auf dem gleichen virtuellen Java™-Rechner ausgeführt. Diese Art der Ausführung ist kostenintensiver, weil überall dort, wo die Engine verwendet wird, zusätzliche Rechenleistung erforderlich ist.

Auf der anderen Seite wird durch die Einrichtung eines Shared Service die Anwendungslogik von der Business-Automation-Engine sowie den dort implementierten Regeln und Prozessen getrennt. Für alle Anwendungen steht in der gesamten Organisation eine einzelne einheitliche Engine zur Verfügung. Das Ergebnis: Die Anwendungen benötigen weniger Rechenleistung und viele von ihnen können per Skalierung der Shared-Engine flexibel unterstützt werden. Dieser Ansatz erfordert die Verwendung von verteilten Programmierungstechniken, bietet aber auch Engine-Zugriff auf eine Vielfalt an Protokollen, wodurch die Engine auch für nicht Java-basierte Anwendungen verfügbar wird.

Ebenfalls beantwortet werden muss die Frage, wie Anwendungen mit der Engine interagieren. Manche von ihnen führen Regeln aus und geben die Ergebnisse im Rahmen einer einzelnen Interaktion zurück. Bei diesen kurzfristigen Sitzungen speichert die Engine nach der Regelausführung keine Anwendungsdaten. Die Anwendung verwaltet Daten üblicherweise in einem unabhängigen persistenten Speicher und gibt sie bei Bedarf an die Engine weiter.

Andere Anwendungen wiederum führen Geschäftsprozesse aus, die Tage, Wochen oder noch länger dauern können. In diesen Interaktionen erstellt die Anwendung langfristige Sitzungen mit der Engine und letztere ist für die in den nachfolgenden Schritten zu verwendenden persistenten Anwendungsdaten zuständig. In diesem Fall haben auch andere Anwendungen Zugriff auf die Daten.

Sitzungslebensdauer und Statusverwaltung können Leistung und Skalierbarkeit der Ausführungsumgebung erheblich beeinflussen. Kurzfristige Sitzungen können ohne weitere Komplexität horizontal skaliert werden, während langfristige Sitzungen entweder vertikal oder horizontal skalierbar sind, aber eine wohl durchdachte Datenbank erfordern. Aufgrund ihrer Ressourcenanforderungen werden langfristige Sitzungen nur im Extremfall verwendet, wie z. B. für die Verwaltung von Geschäftsprozessen.

FAZIT

Die Business-Automatisierung wurde entwickelt, weil die Softwareentwicklung nicht mehr in der Lage war, moderne Geschäftsanforderungen zu erfüllen. Um diesen Anforderungen gerecht zu werden, hat die Business-Automatisierung jedoch ein erhöhtes Risiko und andere Herausforderungen mit sich gebracht. In den vergangenen 20 Jahren hat sich das Innovationstempo dermaßen gesteigert, dass traditionelle Business-Automatisierungslösungen nicht mehr Schritt halten können. Die Wirtschaft definiert sich mittlerweile durch die digitale Disruption reifer Märkte sowie eine sich schnell weiterentwickelnde und zunehmend komplexere regulatorische Umgebung.

Branchenstudien wie der 2015 State of DevOps Report[1] von Puppet haben gezeigt, dass leistungsstarke Softwareunternehmen für den Wettbewerb in dieser volatilen Umgebung am besten gewappnet sind. Aus diesem Grund müssen Business-Automatisierungslösungen moderne Infrastrukturen sowie Ansätze für die Softwareentwicklung und -bereitstellung integrieren.

Die Business-Automatisierungspraxis von Red Hat® Consulting nutzt den in diesem Dokument vorgestellten einheitlichen Ansatz, um Unternehmen dabei zu unterstützen, den vollen Mehrwert der Business-Automatisierung durch die Nutzung und nicht die Umgehung moderner Techniken der Softwareentwicklung auszuschöpfen.

Weitere Informationen zur Business-Automatisierungspraxis von Red Hat Consulting finden Sie im Datenblatt unter redhat.com/de/resources/red-hat-consulting-discovery-session-business-automation.

1 https://puppet.com/resources/white-paper/2015-state-of-devops-report

ÜBER RED HAT

Red Hat, der weltweit führende Anbieter von Open Source-Lösungen, folgt einem Community-basierten Ansatz, um verlässliche und leistungsstarke Technologien in den Bereichen Cloud, Linux, Middleware, Storage, Mobile und Virtualisierung bereitzustellen. Darüber hinaus bietet Red Hat vielfach ausgezeichnete Support-, Training- und Consulting-Services. Red Hat ist ein S&P 500-Unternehmen mit über 80 Niederlassungen weltweit, das seine Kunden und Partner mithilfe hochwertiger Services und Technologien dabei unterstützt, ihr Geschäft voranzutreiben.

EUROPA, NAHOST UND AFRIKA (EMEA)
00800 7334 2835
de.redhat.com
europe@redhat.com

TÜRKEI
00800-448820640

ISRAEL
1-809 449548

VAE
8000-4449549

facebook.com/redhatinc
@redhat
linkedin.com/company/red-hat

de.redhat.com
#INC0410962_v1_201606_KVM