mehr zum thema:
http://blog.codecentric.de/
http://www.meettheexperts.de/

WER HAT ANGST VOR DEM BETRIEB? EINE KLEINE EINFÜHRUNG FÜR ARCHITEKTEN UND ENTWICKLER

Einer der wichtigsten Stakeholder einer Software ist der Betrieb, denn er stellt sicher, dass die Anwendung den Nutzern zuverlässig zur Verfügung steht. Doch wer ist dieser „Betrieb" eigentlich? Welche Rollen verbergen sich dahinter und welche Anforderungen haben sie an eine Anwendung? Trotz seiner Wichtigkeit ist der Betrieb für viele Architekten und Entwickler wenig transparent. In dem Artikel werden die wichtigsten Rollen und Anforderungen des Betriebs beleuchtet und es wird untersucht, welche Anforderungen sich daraus für die Architektur und das Design einer Anwendung ergeben.

der autor

Uwe Friedrichsen
(E-Mail: uwe.friedrichsen@codecentric.de)
hat langjährige Erfahrungen als Architekt, Projektleiter und Berater. Aktuell verfolgt er als Leiter „Competence Center Architektur" bei der codecentric AG Software- und Unternehmensarchitekturen im Kontext agiler Konzepte.

Betrieb – das unbekannte Wesen

Wenn ich mich mit Softwareentwicklern unterhalte, dann ist der Betrieb (gerne auch *Run the Business* oder einfach nur *Run* genannt) für sie häufig eine nicht weiter differenzierbare Masse aus Personen und Tätigkeiten, deren Haupttätigkeit es ist, alles abzulehnen, was Entwicklern Spaß machen könnte. Auch viele Architekten haben oftmals kaum eine genauere Wahrnehmung vom Betrieb. Das ist in mehrfacher Hinsicht erstaunlich: Zum einen ist der Betrieb – wie die Entwicklung auch – Teil einer Unternehmens-IT und man würde, von außen betrachtet, erwarten, dass sich alle Bereiche einer Unternehmens-IT bestens kennen und eng zusammenarbeiten. Zum anderen ist der Betrieb der Bereich, der sich darum kümmert, dass das, was die Entwickler erschaffen haben, auch über Jahre hinweg zuverlässig und problemlos läuft. Ohne den Betrieb würde das „Baby" des Entwicklers also nie das Licht der Produktion erblicken.

Wahrscheinlich müsste man eine längere, psychologische Untersuchung durchführen, um zu erforschen, warum Entwickler den Betrieb trotz der genannten Gründe häufig so schlecht kennen (da ich kein Psychologe bin, versuche ich es aber erst gar nicht). In einer Vielzahl von Projekten habe ich aber gelernt, wie wichtig es für das Entwicklungsteam (inklusive Architekten) ist, die Anforderungen des Betriebs zu kennen. Umgekehrt habe ich auch mehr als einmal schmerzhaft erfahren müssen, welche Konsequenzen es haben kann, wenn man als Projektteam die Anforderungen nicht berücksichtigt.

Über das Thema kann man sicherlich ein ganzes Buch schreiben, insbesondere wenn man versucht, gute Musterlösungen für die verschiedenen Betriebsanforderungen aufzuzeigen. Im Folgenden beschränke ich mich auf einen Überblick über einige wichtige Rollen, welche Anforderungen diese haben, und die daraus resultierenden Konsequenzen für eine Softwareentwicklung. Diese kleine „Starter-Checkliste" können Sie in Ihrem nächsten Projekt an Ihre konkreten Gegebenheiten anpassen und erweitern und damit prüfen, ob Sie die wichtigsten Punkte in Ihrer Lösung bedacht haben.

Rollen im Betrieb

Wenn man sich den „Betrieb" einmal genauer anschaut, zerfällt dieser in eine ganze Menge unterschiedlicher Rollen, die häufig auch von unterschiedlichen Personen wahrgenommen werden. Es gibt keine genormten Rollenprofile, die in allen IT-Betriebsbereichen Anwendung finden, aber anhand der im Rahmen des Anwendungsbetriebs abzudeckenden Aufgabenstellungen lässt sich eine Reihe prototypischer Rollen festlegen, die man in der einen oder anderen Form immer antrifft (siehe auch Abbildung 1):

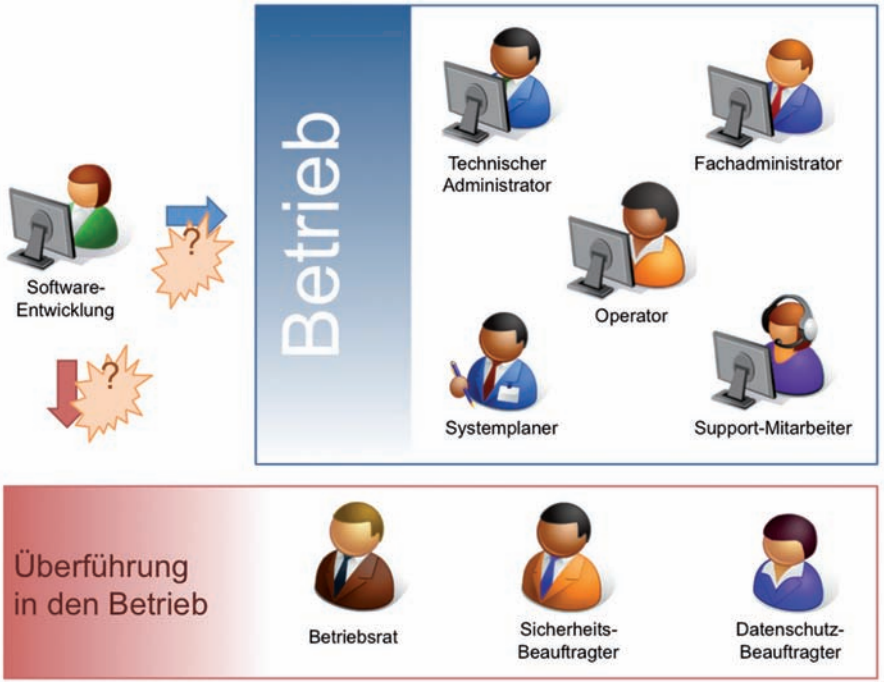

Abb. 1: Typische Rollen im Betrieb und bei der Überführung einer Anwendung in den Betrieb.

- der *Operator*, der den Betrieb der Anwendungslandschaft sicherstellt
- der *technische Administrator*, der die Anwendung auf technischer Ebene betreut
- der *Fachadministrator*, der die Anwendung auf fachlicher Ebene betreut
- der *Systemplaner*, der die benötigten Infrastrukturkomponenten plant und beschafft
- der *Support-Mitarbeiter*, der der Ansprechpartner von Anwendern bei Problemen ist

Ob es diese Rollen genau in diesem Schnitt in Ihrem Unternehmen bzw. dem Unternehmen Ihres Kunden gibt, sei einmal dahingestellt. Ebenso kann es sein, dass einige dieser Rollen bei Ihnen nicht differenziert werden oder anders geschnitten sind. Das ist aber an dieser Stelle nicht wichtig. Wichtig sind hier die Anforderungen, die mit den Rollen verbunden sind. Außerdem ist entscheidend, welche Konsequenzen sich daraus für Architektur und Design einer Anwendung ergeben.

Zusätzlich gibt es auch noch eine Reihe weiterer Rollen, die zwar nicht zum Betrieb im engeren Sinne gehören, an denen man bei der Softwareentwicklung aber dennoch meistens „vorbei" muss, d. h. deren Zustimmung man benötigt, wenn man eine Anwendung in den Betrieb überführen will (**siehe auch Abbildung 1**):

- der *Sicherheitsbeauftragte*, der die Einhaltung der Sicherheitsrichtlinien sicherstellt
- der *Datenschutzbeauftragte*, der die Einhaltung der gesetzlichen Datenschutzbestimmungen sicherstellt
- der *Betriebsrat*, der die Berücksichtigung der Belange der Belegschaft sicherstellt

Diese Auflistung von Rollen erhebt keinen Anspruch auf Vollständigkeit. Bei näherem Nachdenken fallen Ihnen sicherlich noch weitere Rollen ein, mit denen Sie in Ihren Projekten zu tun haben. Hat man aber die Tätigkeiten und die daraus resultierenden Anforderungen der hier genannten Rollen grundsätzlich verstanden, dann hat man aus meiner Erfahrung die wichtigsten Anforderungen aus dem Betrieb abgedeckt und kann sie in seiner Lösung angemessen berücksichtigen.

Im weiteren Verlauf dieses Artikels betrachte ich exemplarisch die ersten drei Rollen nach dem folgenden einheitlichen Schema:

- *Aufgabe:* Welche Aufgaben hat die Rolle?
- *Werkzeuge:* Welche Werkzeuge stehen der Rolle zur Erfüllung ihrer Aufgaben zur Verfügung?
- *Anforderungen:* Welche Anforderungen hat diese Rolle an die Anwendung, um ihre Aufgaben optimal erfüllen zu können?
- *Konsequenzen:* Welche Konsequenzen ergeben sich daraus für Architektur und Design der zu erstellenden Lösung?

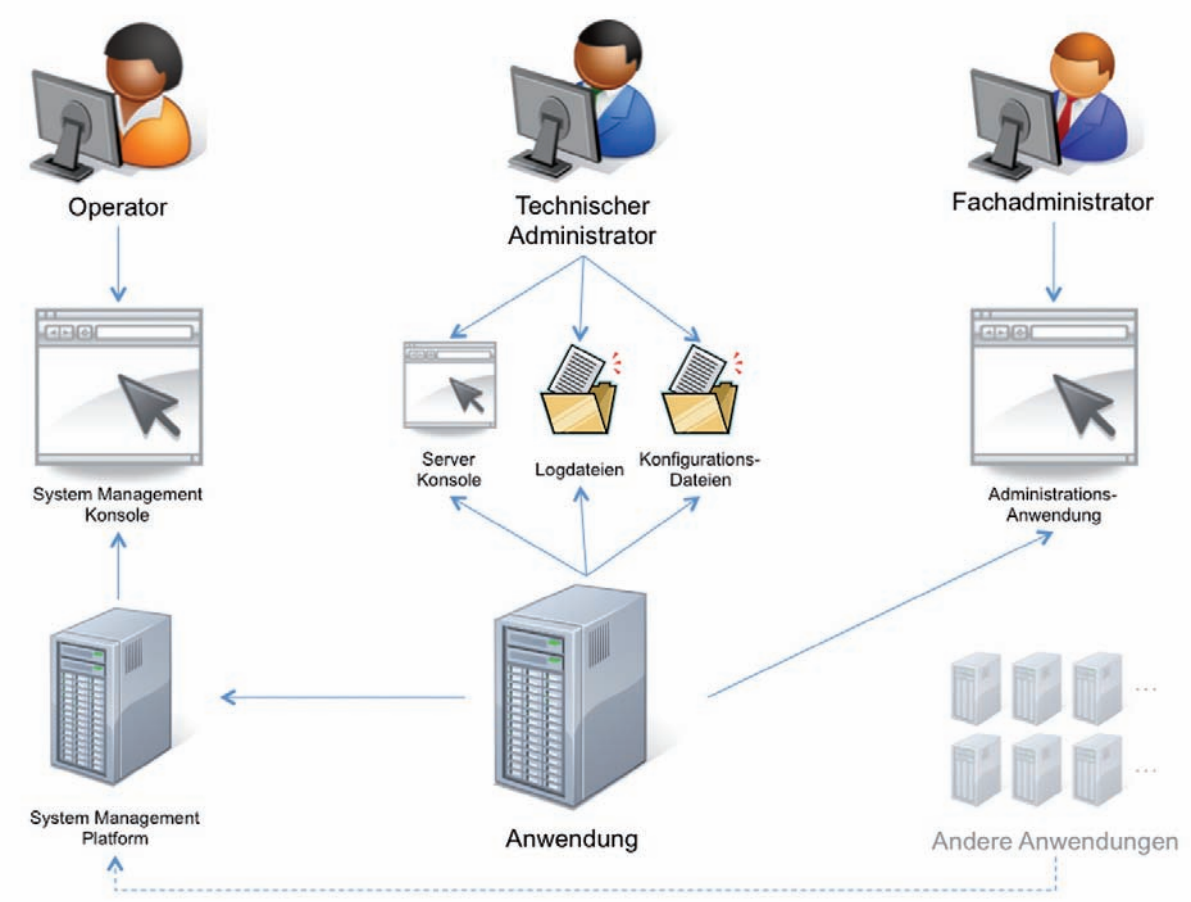

Abb. 2: *Werkzeuge des Operators und der Administratoren beim Zugriff auf eine Anwendung.*

Dieses Schema können Sie jederzeit nutzen, um Ihre eigenen Rollen und die daraus resultierenden Konsequenzen für Architektur und Design auf einfache Weise nachzupflegen.

Der Operator

Aufgabe

Der Operator ist dafür verantwortlich, den reibungslosen Betrieb einer Anwendungslandschaft zu gewährleisten, eventuell auftretende Probleme schnell zu erkennen und schnell darauf zu reagieren. Außerdem ist er typischerweise auch noch für die Datensicherung zuständig. Der Operator betrachtet die Systemlandschaft meistens eher aus einer Infrastruktursicht – für ihn ist eine Anwendung nur ein „Kästchen" in seiner Landschaft aus Servern, Routern, Netzwerkverbindung, Datenbanken, Anwendungen usw., das entweder funktioniert oder aber ein Problem hat. In der Regel kennt er die einzelnen Anwendungen nicht inhaltlich. Insbesondere bei einem Outsourcing des Betriebs haben wir häufig reine Operatoren, die den Betrieb der ausgelagerten Systemlandschaft überwachen.

Werkzeuge

Üblicherweise überwacht der Operator seine Systemlandschaft mit Hilfe von Systemmanagement-Lösungen (siehe Abbildung 2), wie zum Beispiel „IBM Tivoli", „HP OpenView" oder „Nagios" (um auch eine Open-Source-Alternative zu nennen). In den Überwachungslösungen hat der Operator voraggregierte Sichten auf die Landschaft. Tritt an einer Stelle ein Problem auf, kann er sich mit Hilfe des Werkzeugs in das gemeldete Problem „hineinklicken".

Anforderungen

Vereinfacht ausgedrückt, hat der Operator genau zwei Fragen an eine Anwendung:

- Läuft die Anwendung ohne Probleme?
- Wenn nicht, was muss ich tun?

Zur Beantwortung der ersten Frage benötigt er einen oder mehrere Messpunkte, die er – typischerweise via *Simple Network Management Protocol (SNMP)* – in seine Systemmanagement-Lösung integriert. Sollte an der Stelle ein Problem gemeldet werden, will er eine möglichst klare Probleminformation über sein Werkzeug angezeigt bekommen. Diese Meldung nutzt er dann, um in einem Leitfaden (gedruckt oder online) nachschlagen zu können, was er tun soll bzw. wen er im Rahmen einer Problem-Eskalation darüber informieren soll.

Bezogen auf das Thema Datensicherung möchte er wissen, welche Daten er nach welchem Verfahren sichern muss, wie oft er das tun muss, ob er dafür die Anwendung stoppen muss usw.

- Versuchen Sie, die Anzahl der Datenablagestellen zu minimieren.
- Prüfen Sie kritisch, was alles gesichert werden muss (Dateien wie z. B. Konfigurationsdateien nicht vergessen).
- Prüfen Sie, ob Sicherungen ohne ein Herunterfahren der Anwendung möglich sind. Nutzen Sie bei Hochverfügbarkeits-Lösungen gegebenenfalls alternative Sicherungsverfahren (z. B. Replikation).
- Bieten Sie bei großen Datenmengen Verfahren für Vollsicherung und Deltasicherung an.
- Nutzen Sie – wenn möglich – die Sicherungskonzepte der eingesetzten Produkte (z. B. Datenbanken). Das spart meistens eine Menge Kopfzerbrechen.

Kasten 1: *Tipps für die Konzeption der Datensicherung.*

Konsequenzen

Für die Systemüberwachung muss eine technische Anwendungsüberwachung zur Verfügung stehen, die per SNMP in die Überwachungslösung des Operators eingebunden werden kann. Die meisten heutigen Anwendungs-Infrastrukturserver – beispielsweise Anwendungsserver, Datenbankserver, Webserver – haben solche Funktionen bereits an Bord. Da es typischerweise um eine rein technische Überwachung von Ressourcen, wie z. B. den verfügbaren Hauptspeicher, geht, muss dafür nichts extra realisiert werden. Gegebenenfalls muss man ein vorhandenes SNMP-Plug-In integrieren oder aktivieren und eventuell noch konfigurieren, welche Ressourcen überwacht werden sollen. Diese Informationen werden dann in die zugehörige Dokumentation (meistens ein Betriebshandbuch) gepackt – und fertig ist man an der Stelle. Häufig überwacht der Operator auch noch die System-Log-Dateien, um Probleme zu erkennen und zu analysieren. Aber auch an der Stelle schreiben die heutigen Lösungen in der Regel die benötigten Informationen ohne weiteres Zutun bzw. mit geringfügiger Nachhilfe, meistens in Form geeigneter Konfigurationseinstellungen.

Da die erforderlichen Maßnahmen bei Problemen häufig anwendungsabhängig sind, kommt man nicht umhin, sich Gedanken über die möglichen Probleme zu machen, die einem Operator von der Anwendung gemeldet werden können, und wie er darauf reagieren soll. Das Ganze sollte man im Kapitel „Problembehebung" des Betriebshandbuchs dokumentieren, wenn man keine Lust hat, womöglich nachts um drei Uhr bei einem Problem vom Operator aus dem Bett geklingelt zu werden.

Beim Thema Datensicherung (siehe Kasten 1) muss man sich überlegen, welche Daten es zu sichern gilt, wie häufig diese zu sichern sind und wie man diese sichern kann. Liegen alle relevanten Daten in einer einzigen Datenbank-Instanz, ist man damit meistens schnell fertig, aber häufig ist es nicht so einfach. Wichtig ist an der Stelle, dass man es nicht dem einzelnen Entwickler überlässt, wie und wo er seine Daten auf welche Weise ablegen will, sondern dass es dafür gemeinsame klare Richtlinien gibt. Meistens bestehen diese Richtlinien nur aus ein paar Zeilen, aber wenn man sie hinschreibt, können sie einem im Betrieb einer Anwendung viel Ärger ersparen. Für den Operator wird dann wieder eine Beschreibung, wie und wie häufig er die Daten sichern soll, in das Betriebshandbuch aufgenommen – das war's.

Der technische Administrator

Aufgabe

Der technische Administrator ist für die technische Konfiguration der Anwendung zuständig. Häufig führt er auch die initiale Installation der Anwendung durch oder er unterstützt sie zumindest. Normalerweise bereitet er auch die *Deployments* neuerer

- Unterscheiden Sie zwischen technischer und fachlicher Überwachung.
- Unterscheiden Sie Protokollierung (*Logging*) und Überwachung (z. B. aktuell angemeldete Benutzer, aktuell genutzte Threads). Nutzen Sie das Logging nicht für die Überwachung.
- Verwenden Sie für das technische Logging vorhandene Logging-Lösungen. Verwenden Sie drei bis fünf Log-Ebenen: *Debug* (optional), *Info, Warning, Error, Severe/Fatal* (optional). Mehr Ebenen benötigen Sie normalerweise nicht (weniger aber auch nicht).
- Legen Sie explizit fest – am besten zusammen mit den Administratoren –, welche Informationen (was und welche Details) oberhalb der Log-Ebene *Debug* protokolliert werden. Alle anderen Meldungen sind in dieser Ebene zu protokollieren. Klären Sie das, auch wenn es häufig nicht einfach ist, denn es spart Ihnen jede Menge genervter Anrufe bei Problemen.
- Achten Sie bei der fachlichen Protokollierung darauf, dass der Fachadministrator einen geeigneten „Viewer" zur Verfügung hat, den er auch problemlos bedienen kann. Verwenden Sie nur drei Log-Ebenen: *Info, Warnung* und *Error*. Mehr brauchen Sie nicht.
- Stimmen Sie die fachliche Protokollierung und Überwachung explizit mit dem Fachadministrator ab, wenn möglich. Fachliche Protokollierung und Überwachung „auf gut Glück" zu programmieren, bringt in der Regel nichts, weil der Administrator meistens andere Informationen benötigt, als Sie denken.
- Kapseln Sie die Details der Protokollierung und Überwachung gegenüber dem Rest der Anwendung.

Kasten 2: *Tipps für die Konzeption der Überwachung.*

Anwendungsversionen vor und führt sie durch (bzw. unterstützt die Durchführung). Letztlich überwacht er die Anwendung auf technischer Ebene. Im Gegensatz zum Operator liegt sein Fokus auf der einzelnen Anwendung und nicht auf der Gesamtlandschaft. Dafür kennt er die Anwendung deutlich detaillierter als der Operator und ist häufig auch die erste Eskalationsinstanz des Operators bei Problemen. Der technische Administrator arbeitet normalerweise zu den üblichen Bürozeiten und überwacht die Anwendung nicht permanent, sondern eher „immer wieder".

Werkzeuge

Der technische Administrator hat typischerweise Zugang zu den Überwachungskonsolen der Infrastruktur-Server. Im Fall einer JavaEE-Anwendung wäre das beispielsweise die Konsole des eingesetzten Applikationsservers. Zusätzlich hat er Zugang zu den verschiedenen Konfigurationsdateien und Deskriptoren einer Anwendung und für die Überwachung zu den Log-Dateien, die die Anwendung schreibt (siehe Abbildung 2). Manchmal setzt er zur Überwachung zusätzlich spezielle Überwachungswerkzeuge ein, beispielsweise „CA Wily Introscope" oder „dyna Trace". Das passiert meistens dann, wenn die von der Anwendung angebotenen Überwachungsmöglichkeiten nicht ausreichend sind.

Anforderungen

Die Anforderungen lassen sich in zwei Bereiche aufteilen:

- Konfiguration
- Überwachung

Für die Konfiguration möchte sich der technische Administrator nicht 10 Lokationen und 12 verschiedene Vorgehensweisen zur Konfiguration der Anwendung merken müssen. Vielmehr möchte er möglichst alle Konfigurationsparameter eines Typs auf einheitliche Weise bearbeiten können. Dabei ist es ihm wichtig, dass die umgebungsbezogenen und die anwendungsbezogenen Parameter sauber voneinander getrennt sind. Während sich umgebungsabhängige Parameter für jede Umgebung (z. B. Entwicklung, Test, Integration, Schulung, Produktion) unterscheiden, bleiben die anwendungsbezogenen Parameter über die Umgebungen hinweg gleich. Die umgebungsbezogenen Parameter sollen durch ein neues *Deployment* der Anwendung in eine Umgebung nicht überschrieben werden. Die anwendungsabhängigen Parameter sollen beim Übertragen der Anwendung in eine andere Umgebung hingegen mitge-

- Unterscheiden Sie technische und fachliche Konfiguration.
- Unterscheiden Sie mindestens, welche Konfigurationsparameter zur Laufzeit geändert werden dürfen und welche einen Neustart der Anwendung erfordern. Stimmen Sie ein eventuell verwendetes internes Caching darauf ab.
- Trennen Sie umgebungs- und anwendungsabhängige Konfigurationsparameter. Entwickeln Sie ein Konzept, wie Sie neue umgebungsabhängige Parameter einführen (z. B. durch eine getrennte Verwaltung von Default-Werten und kundenspezifisch angepassten Parametern). Beschreiben Sie, wie man bei einem *Staging* (Übertragen eines Anwendungsstands, z. B. aus der Test- in die Produktionsumgebung) die anwendungsabhängigen Parameter mit überträgt und die umgebungsabhängigen Parameter nicht.
- Bieten Sie eine einheitliche Verwaltung der Konfigurationsparameter je nach Typ, z. B. XML-Dateien in einem bestimmten Verzeichnis für technische Parameter mit Neustart, zur Laufzeit änderbare technische Parameter via Konsole und fachliche Parameter in der Datenbank mit eigener Pflegeanwendung.
- Kapseln Sie die Details der Konfigurationsparameter, wie Unterschiede der Arten und Ablageorte, Zugriffe und eventuell Caching gegenüber dem Rest der Anwendung.

Kasten 3: *Tipps für die Konzeption der Konfiguration.*

nommen und eventuell vorhandene Parameter überschrieben werden.

Für die Überwachung (siehe auch Kasten 2) möchte der technische Administrator alle relevanten Informationen möglichst übersichtlich an einer Stelle, z. B. der Konsole, einsehen können. Für das Nachverfolgen von Problemen möchte er möglichst klare Fehlermeldungen in den Log-Dateien haben. Außerdem möchte er die Fehlermeldungen nicht zwischen Unmengen anderer, für ihn irrelevanter Informationen suchen müssen.

Konsequenzen

Es muss ein klares Konfigurationskonzept her (siehe auch Kasten 3). Wo kommen die umgebungsabhängigen Parameter hin und wo die anwendungsabhängigen? Wie werden sie bearbeitet – mit einem Texteditor, einer Konsole oder einem anderem Werkzeug? Welche Parameter bedingen einen Neustart der Anwendung, welche können zur Laufzeit geändert werden? Wie *cache* ich zur Laufzeit änderbare Parameter? Das sind einige der Fragen, die man *vor* dem Beginn der Entwicklung einmal durchdenken sollte. Aus den Antworten sollte man die entsprechenden Richtlinien ableiten, am besten aber eine kleine Komponente, die den Zugriff auf alle Konfigurationsparameter kapselt. Das ist kein großer Aufwand, bewahrt einen aber vor einem chronisch schlecht gelaunten technischen Administrator beim Kunden und erspart einem häufig auch viel Ärger bei *Deployments*.

Ein ähnliches Konzept muss für die Überwachung und die Log-Dateien her. Welche Werte sollen dem Administrator angeboten werden? Reichen die Standardinformationen, die die Server in ihren Konsolen anbieten, oder müssen noch zusätzliche Werte – meistens via JMX – angeboten werden? Welche Log-Dateien schreibe ich? Was kommt in die Log-Dateien? Welche Log-Level unterscheide ich und was gehört in welches Level? Häufig trifft man auf das Problem, dass ein Entwickler ohne entsprechende unterstützende Richtlinien gar nicht vernünftig entscheiden kann, wann er was protokollieren muss. Als Ergebnis hat man dann Log-Dateien, die ein wildes Durcheinander im Log-Level „Debug" auswerfen und jenseits des Levels plötzlich komplett verstummen. Da müssen entsprechende Richtlinien her, was auf welche Weise in den höheren Log-Leveln zu protokollieren ist. Das ist auch kein Hexenwerk, hilft aber erfahrungsgemäß ungemein – und reduziert die genervten Anrufe des Kunden bei Problemen drastisch.

Der Fachadministrator

Aufgabe

Der Fachadministrator ist für die fachliche Betreuung der Anwendung zuständig. Er kümmert sich um die fachliche Konfiguration der Anwendung und ist der erste Ansprechpartner, wenn die Anwendung fachlich nicht korrekt funktioniert. Entsprechend ist er nicht unbedingt ein technischer Experte, kennt aber die fachlichen Inhalte der Anwendung sehr genau. Manchmal kommt er sogar aus dem Fachbereich und nicht aus der IT. Der Fachadministrator arbeitet normalerweise zu den üblichen Bürozeiten und überwacht die Anwendung nicht permanent, sondern agiert eher bei Bedarf, obwohl er sich manchmal auch gerne ohne konkrete Probleme einen Eindruck vom Zustand der Anwendung verschafft.

Werkzeuge

Der Fachadministrator hat keine typischen Werkzeuge. Da er nicht unbedingt ein technischer Spezialist ist, mag er einfach und intuitiv zu bedienende Oberflächen (siehe Abbildung 2) am liebsten. Für Konfigurationsparameter lässt er sich auch einmal auf eine Textdatei ein, wenn sie nicht zu kryptisch ist. XML ist in der Regel nicht seine Sache, er mag es besser lesbar. Typischerweise hat er keinen Zugriff auf die Konsole der Infrastruktur-Server (Applikationsserver oder Ähnliches).

Anforderungen

Die Anforderungen an den Fachadministrator ähneln auf den ersten Blick denen des technischen Administrators, unterscheiden sich in der Detailbetrachtung dann aber doch deutlich. Der Fachadministrator möchte möglichst alle Konfigurationsparameter sowie eventuell weitere von ihm zu pflegende Daten (z. B. Benutzerdaten und Berechtigungen) auf einheitliche Weise bearbeiten können, am liebsten in einer eigens für ihn gestalteten Administrationsanwendung. Da alle von ihm betreuten Konfigurationsparameter anwendungsabhängig sind, müssen hier anwendungs- und umgebungsabhängige Parameter nicht unterschieden werden

Normalerweise erwartet der Fachadministrator, dass alle Änderungen sofort wirksam werden, d. h. dass kein Neustart der Anwendung notwendig ist, damit die Änderung wirksam wird. Sollte doch ein Neustart erforderlich sein, sollten Sie den Fachadministrator explizit darauf aufmerksam machen, z. B. durch einen entsprechenden deutlich sichtbaren Hinweis in der Administrationsanwendung, wenn Sie regelmäßige Anrufe wegen vermeintlicher Anwendungsfehler vermeiden möchten.

Für die Überwachung möchte der Fachadministrator alle fachlich relevanten Informationen möglichst übersichtlich an einer Stelle angezeigt bekommen, am liebsten in einer Art fachlichem Dashboard. Falls er Log-Dateien ansehen soll, dann möchte er dies am liebsten aus seiner Administrationsoberfläche heraus tun und nicht im Dateisystem suchen müssen – insbesondere deshalb, weil er oft gar keinen Zugriff auf das Dateisystem des Anwendungsservers hat. Fehlerhinweise möchte er in Geschäftssprache haben und nicht in technischer Sprache.

Konsequenzen

Die Konsequenzen sind ähnlich wie beim technischen Administrator. Allerdings sind jetzt zusätzliche Arten von Konfigurationsparametern und Überwachungsmöglichkeiten einzuplanen. Ergänzend kommt hinzu, dass neue Oberflächen für den Fachadministrator geschaffen werden müssen, da er keinen Zugriff auf die Oberflächen des technischen Administrators hat und ihm die Oberflächen außerdem in der Regel zu technisch und unverständlich sind. Für die Oberflächen kann man in der Regel auch keine Bordmittel der eingesetzten Server-Komponenten nutzen; man muss sie selbst erstellen, quasi eine Art Administrationsanwendung, die neben der eigentlichen Anwendung steht. Diese muss genauso geplant, entworfen und umgesetzt werden wie die eigentliche Anwendung. Projekte, die das nicht im Blick haben, kommen bei den geplanten Aufwänden und Terminen in der Regel ziemlich ins Schwitzen.

Weitere Rollen

Aus den drei hier exemplarisch betrachteten Rollen lässt sich eine Menge interessanter Anforderungen an Architektur und Design einer Anwendung ablesen, die typischerweise übersehen werden, wenn man sich darüber nicht vorab einige Gedanken macht. Der resultierende Aufwand ist mei-

ner Erfahrung nach nicht höher, als wenn man die Anforderungen nicht beachtet, nur ist der Kunde mit der Lösung wesentlich zufriedener und die Anzahl der entnervten Support-Anrufe ist deutlich geringer.

Die folgenden Rollen haben wir noch gar nicht betrachtet:

- Der *Systemplaner*, der wissen möchte, welche Infrastrukturanforderungen die Lösung hat und wie viel Hardware er dafür einplanen muss.
- Der *Support-Mitarbeiter*, der wissen möchte, was er seinen anrufenden Kunden zu deren Problemen erzählen soll.
- Der *Sicherheitsbeauftragte*, dessen Aufgabe es ist sicherzustellen, dass die gesetzlichen und firmenspezifischen Sicherheitsbelange erfüllt sind.
- Der *Datenschutzbeauftragte*, der die gesetzlichen Datenschutzbestimmungen erfüllt sehen will.
- Der *Betriebsrat*, der die Interessen der von der Anwendung betroffenen Mitarbeiter vertritt.

Eine nähere Betrachtung dieser Rollen würde noch eine Reihe weiterer Anforderungen zutage bringen, worauf ich an dieser Stelle aus Platzgründen leider verzichten muss.

Ergänzende Hinweise

Es gibt noch ein paar ergänzende Aspekte, zu denen ich kurz etwas anmerken möchte.

Den „Agilisten" unter Ihnen sei gesagt, dass ich hier nicht über ein *Big Design UpFront (BDUF)* rede. Es geht um die notwendigen und sinnvollen Gedanken zu den Themen Architektur und Design, die man sich vor Beginn einer Entwicklung immer machen sollte, also das *Just enough Design UpFront (JDUF)*.

In dem Artikel habe ich an mehreren Stellen gesagt, dass Dokumentation zu erstellen ist. Das heißt aber nicht, dass der Architekt oder Designer auch automatisch all diese Dokumentation erstellen muss. Er muss aber die notwendigen Informationen zur Dokumentation beitragen, denn wer außer ihm sollte über dieses Wissen verfügen?

Batches, die es aus meiner Erfahrung in jeder größeren Anwendung gibt, haben andere Anforderungen an Konfiguration und Überwachung als die oben beschriebenen Online-Anwendungen. Auch dafür muss man sich überlegen, welche Rollen und Anforderungen es in diesem Kontext gibt. Das ist umso wichtiger, da Batches aus meiner Erfahrung heute häufig stark unterschätzt werden und außer „altgedienten" Host-Entwicklern kaum noch jemand weiß, wie man einen guten, d. h. zuverlässigen und performanten, Batch konzipiert.

Nicht betrachtet habe ich außerdem das große Thema *Service Level Agreements (SLAs)* und die daraus resultierenden Anforderungen an die Überwachung von Anwendungen. Dieses Thema kann man als eine Art zusätzliche Dimension zu den verschiedenen Rollen bei den Anforderungen betrachten. Es kann extrem schwierig werden, die Überwachung der SLAs nachträglich in eine Anwendung zu integrieren.

Ein ebenfalls spannendes, aber hier nicht betrachtetes Thema sind die Implikationen, die sich aus der vorgegebenen oder gewählten Infrastruktur auf die Softwarearchitektur ergeben. Häufig wird diese Abhängigkeitsrichtung überhaupt nicht betrachtet. Dabei hat die Wahl der Infrastruktur durchaus Auswirkungen darauf, wie man seine Softwarearchitektur gestalten muss.

Fazit

Der Betrieb – als einer der zentralen Stakeholder einer Anwendung – wird heute in der Anwendungsentwicklung immer noch häufig viel zu wenig beachtet. Dabei fördert die nähere Untersuchung des Betriebs eine Reihe von Rollen und Anforderungen zutage, die Konsequenzen für Architektur und Design einer Anwendung haben und sehr wichtig für die Zuverlässigkeit und gefühlte Qualität einer Anwendung sind. Die dadurch entstehenden Aufwände in der Entwicklung sind nicht höher, als wenn man die Anforderungen ignoriert. Häufig sind die Aufwände über längere Sicht betrachtet sogar deutlich niedriger. Der Unterschied ist aber ein deutlich zufriedenerer Betrieb.

Da der Betrieb wesentlich seltener Rückfragen stellen muss, bedeutet das für das Entwicklungsteam deutlich weniger Stress und eine entspanntere Nachtruhe. ■