# 10 TEUERSTEN FEHLER

## bei der Wartung von Java-Legacy-Systemen, die die Modernisierung bremsen!

Ein Leitfaden für mehr Freiraum in den Entwicklungsteams.

# EINLEITUNG

Die Wartung von Java-Legacy-Systemen und die parallel laufende Modernisierung der Anwendungslandschaft konfrontiert IT-Teams mit signifikanten Herausforderungen. Die Bedeutung der Legacy-Systeme für den Geschäftsbetrieb steht im Widerspruch zu ihrer überholten Technologie, einer meist starren Softwarearchitektur und der damit einhergehenden Komplexität. Dieser Zustand führt zu einem erhöhten Wartungsaufwand, vermindert die Anpassungsfähigkeit an Marktbe-dingungen und zehrt an den Ressourcen, die für Innovationsprojekte vorgesehen sind. IT-Verantwortliche stehen somit vor der schwierigen Aufgabe, die Balance zwischen Aufrechterhaltung und zukunftsorientierter Entwicklung zu finden. Entscheidend ist die Erkenntnis, dass ineffiziente Wartung Ressourcen verschwendet und interne Experten demotiviert, wenn unklare Strukturen und mangelnde Automatisierung vorherrschen. Als Ausweg aus diesem Dilemma bietet sich die Einrichtung eines dedizierten Wartungsteams an, unterstützt durch gezielte Automatisierung und klar definierte Prozesse, die nicht nur die laufenden Systeme effizient pflegen, sondern auch Freiräume für die Entwicklungsarbeit schaffen. Dieses Vorgehen gewährleistet, dass Legacy-Systeme nicht als Bürde, sondern als wertvolles Erbe behandelt werden, das mit Bedacht in die digitale Zukunft überführt werden kann.

Die Wartung von Java-Legacy-Systemen neben der fortlaufenden Modernisierung der IT-Landschaft zwingt IT-Abteilungen oft in ein Dilemma, das sowohl strategische als auch operative Herausforderungen mit sich bringt.

*„Mit dem Wissen können viele Ressourcen in Modernisierungsprozesse gesteckt werden."*

**Ingo Düppe,** Geschäftsführender Gesellschafter

## Insbesondere treffen wir wiederholt auf folgende Schwierigkeiten:

**Fokussierung auf die Vergangenheit**
Das Know-how der internen Experten wird für die Wartung veralteter Systeme eingesetzt und nicht um an den Modernisierungsprozessen mitzuwirken. Das sorgt für einen Know-how Verlust und eine Abnahme der Innovationskraft.

**Parallelarbeit führt zu Konflikten**
Die simultane Bearbeitung von Wartung und Neuentwicklung durch dasselbe Team sorgt für Ressourcenkonflikte, welche sehr schnell in Frustration aller Beteiligten münden kann.

**Fehlende Systematisierung von Fehlern**
Ohne klare Fehlereinordnung und Priorisierung werden wichtige Ressourcen in die Korrektur unbedeutender Fehler investiert.

**Geringe Automatisierung**
Ein unzureichender Automatisierungsgrad, vor allem in der Entwicklungspipeline, führt zu ineffizienten Abläufen und verhindert schnelle Anpassungen an neue Anforderungen.

## Um diese Herausforderungen zu meistern, empfehlen wir folgende Maßnahmen:

**Dedizierte Teams für Wartung und Support**
Befreien Sie Ihre wertvollsten Entwickler von den Fesseln der Legacy-Wartung, indem Sie spezialisierte Maintenance-Teams für diese Aufgaben einsetzen.

**Automatisierung der End-2-End-Tests**
Vereinfachen und beschleunigen Sie den Freigabe-prozess durch vollautomatische Testverfahren.

**Kompatibilitätsprüfung von Updates automatisieren**
Implementieren Sie Tools und Mechanismen, die automatisch die Vereinbarkeit von Updates und Sicherheitspatches prüfen.

**Systematisierung der Fehlerbearbeitung**
Führen Sie eine klare Fehlerklassifizierung ein, die eine wirtschaftlich sinnvolle Bewertung und Priorisierung der Behebungsmaßnahmen ermöglicht. Nutzen Sie automatisierte Lösungen zur Bewältigung wiederkehrender Probleme. Ein guter Nebeneffekt ist auch, dass ihre Entwickler weniger frustriert sind und motivierter arbeiten.

*„Der erste Schritt in die richtige Richtung ist es, dieses Paper zu lesen und die Tipps in die Realität umzusetzen!"*

**Marcus Nörder-Tuitje,** Geschäftsführender Gesellschafter

# Das Ziel ist es,

durch diese Maßnahmen nicht nur die Effizienz der IT-Teams zu steigern, sondern vor allem den Freiraum für Innovation und Wachstum zu schaffen. IT-Verantwortliche müssen die Möglichkeit erhalten, sich voll und ganz auf die Gestaltung der digitalen Zukunft ihrer Unternehmen zu konzentrieren, ohne von Altlasten gebremst zu werden. Durch die konsequente Verfolgung dieses Ansatzes erreichen wir gemeinsam eine höhere Agilität, verbesserte Wettbewerbsfähigkeit und eine zukunftssichere Anwendungslandschaft.

Bei CROWDCODE setzen wir auf Vielfalt. Daher betonen wir ausdrücklich, dass unsere Texte gleichermaßen alle Menschen ansprechen, unabhängig von ihrem Geschlecht. Um die Lesbarkeit zu erleichtern, verwenden wir eine geschlechtsneutrale Schreibweise und vermeiden Mehrfachbezeichnungen. In unseren Texten verwenden wir das generische Maskulinum, um alle Menschen einzubeziehen.

# WAS SIE IN UNSEREM WHITEPAPER ERWARTET:

**Führungskräfte und IT-Verantwortliche mittelständischer B2B-Unternehmen stehen häufig vor der Herausforderung, ihre Anwendungslandschaft zu modernisieren, während sie gleichzeitig Legacy-Systeme warten müssen. Unser Whitepaper richtet sich speziell an diese Gruppe, insbesondere an solche, die Java-Legacy-Systeme im Einsatz haben und ihre Ressourcen zwischen Modernisierungsprojekten und laufender Wartung aufteilen müssen. Unser Ziel ist es, Ihnen Strategien an die Hand zu geben, mit denen Sie diese Herausforderungen meistern und Ihre IT-Infrastruktur zukunftsfähig gestalten können.**

## In diesem Leitfaden erhalten Sie:

**Konkrete Strategien**
Entdecken Sie praxisnahe, direkt umsetzbare Tipps zur Optimierung der Wartung und Pflege Ihrer individuell entwickelten Bestandssoftware. Durch effektive Systematisierung und intelligente Automatisierung schaffen Sie Räume für Innovation.

**Lösungen für gängige Fallstricke**
Erfahren Sie, wie Sie typische und vermeidbare Stolpersteine im Management von Legacy-Systemen identifizieren und umgehen. Unser Leitfaden hilft Ihnen, sich auf das Wesentliche zu konzentrieren und unnötige Aufwände zu reduzieren.

**Optimale Ressourcennutzung**
Wir zeigen Ihnen nicht nur Wege auf, wie Sie den Aufwand für Wartung und Pflege nicht nur effizienter gestalten, sondern auch wie Sie die freigewordenen Ressourcen für zukunftsorientierte Entwicklungsprojekte einsetzen können.

## Sofortmaßnahmen für greifbare Ergebnisse:

**Klarheit schaffen**
Bewertung und Festlegung der Restlaufzeit Ihrer Systeme für strategische Entscheidungen!

**Prozesse systematisieren**
Für effizientere Abläufe ohne Ballast!

**Vollautomatisierung einführen**
Um manuelle Fehlerquellen zu eliminieren und Effizienz zu steigern!

**Aufbau eines dedizierten Wartungsteams**
Um mehr Raum für Innovation zu schaffen und Ihre Entwickler von Routineaufgaben zu entlasten.

**Mit diesem Whitepaper** erhalten Sie Tipps, um den Spagat zwischen der Wartung von Java-Legacy-Systemen und der dringend notwendigen Modernisierung Ihrer Anwendungslandschaft erfolgreich zu meistern. Profitieren Sie von unserem Expertenwissen und setzen Sie Ihre Ressourcen dort ein, wo sie den größten Mehrwert schaffen – in der Zukunft Ihres Unternehmens.

# INHALTSVERZEICHNIS

# DAS DILEMMA DER MODERNISIERUNG

**In der heutigen digitalisierten Welt stehen Unternehmen vor der Herausforderung, ihre technische Infrastruktur zu modernisieren, um wettbewerbsfähig zu bleiben und neue Geschäftsfelder erfolgreich zu erschließen. Eine moderne und agile technische Infrastruktur ist unerlässlich, um im globalen Wettbewerb zu bestehen. Jedoch erweisen sich die essenziellen Legacy-Systeme häufig als Innovationsbremse, die schnelle Anpassungen an technologische Entwicklungen und Marktanforderungen behindert.**

Trotz des erkennbaren Bedarfs zur Modernisierung ihrer Anwendungssysteme bewegt sich die Mehrheit der Unternehmen langsamer vorwärts als notwendig. Obwohl mehr als die Hälfte der Unternehmen plant, ihre Bestandssysteme in naher Zukunft zu modernisieren, hat nur ein kleiner Bruchteil tatsächlich die Kernprojekte in diesem Bereich abgeschlossen. Sowohl große als auch mittelständische Firmen ringen mit dem Druck, ihre Systeme auf den aktuellen technischen Stand zu bringen. Dies verdeutlicht den allgemeinen Nachholbedarf und die Dringlichkeit dieser Bestrebungen unterstreicht.

Legacy-Systeme stehen im Zentrum dieses Dilemmas. Aufgrund ihrer Verlässlichkeit und bewährten Funktionalität sind sie zwar unverzichtbar für den täglichen Betrieb, gleichzeitig aber auch zentrale Hemmnisse für Innovation und Flexibilität. Um diese Herausforderung zu meistern, ist ein Umdenken erforderlich: Es gilt, die Balance zwischen der Bewahrung bewährter Strukturen und dem unumgänglichen Bedarf an Modernisierung und Flexibilität zu finden.

Zur Überwindung des Modernisierungsdilemmas bieten sich strategische Ansätze und Lösungen, die sowohl das wertvolle Erbe der Legacy-Systeme berücksichtigen als auch Raum für Innovationen schaffen. Die Etablierung dedizierter (managed) Maintenance-Teams, systematische Fehlerbewertungen und die Automatisierung von Entwicklungsprozessen sind dabei entscheidende Maßnahmen, um den Konflikt zwischen der Wartung älterer Systeme und der Entwicklung neuer Anwendungen zu lösen.

Unternehmen stehen vor der Entscheidung, ob eine Migration älterer Systeme auf moderne Plattformen oder eine vollständige Substitution der Legacy-Systeme die sinnvollere Option darstellt. Beide Wege haben ihre Vor- und Nachteile, wobei eine gut durchdachte Roadmap entscheidend für den Erfolg der Modernisierung ist. Eine sorgfältige Analyse der vorhandenen Systeme ermöglicht es, klare Prioritäten zu setzen und zu entscheiden, welche Technologien erneuert oder ersetzt werden sollten.

Nicht zuletzt erfordert die erfolgreiche Modernisierung der IT-Landschaft auch eine Neuorientierung der internen Experten. Diese sollten nicht mit der Wartung veralteter Systeme belastet werden, sondern ihre Fachexpertise in die Gestaltung der digitalen Zukunft einbringen. Durch strategische Planung und effizienten Ressourceneinsatz müssen Unternehmen einen Weg finden, die Herausforderungen alter Systeme zu bewältigen und eine solide Basis für zukünftiges Wachstum und Innovation zu legen.

## Lösungsansätze

Die effektive Modernisierung von IT-Landschaften, insbesondere bei mittelständischen Unternehmen mit starken Legacy-Systemabhängigkeiten, erfordert innovative Lösungsansätze. Durch die Implementierung von exklusiven (managed) Maintenance-Teams, der Systematisierung von Prozessen und der Automatisierung der Softwareentwicklungsprozesse können Unternehmen das Dilemma der Modernisierung effektiv angehen.

## Dedizierte Maintenance-Teams

Die Einrichtung von speziellen Teams, die ausschließlich für die Wartung und Pflege der

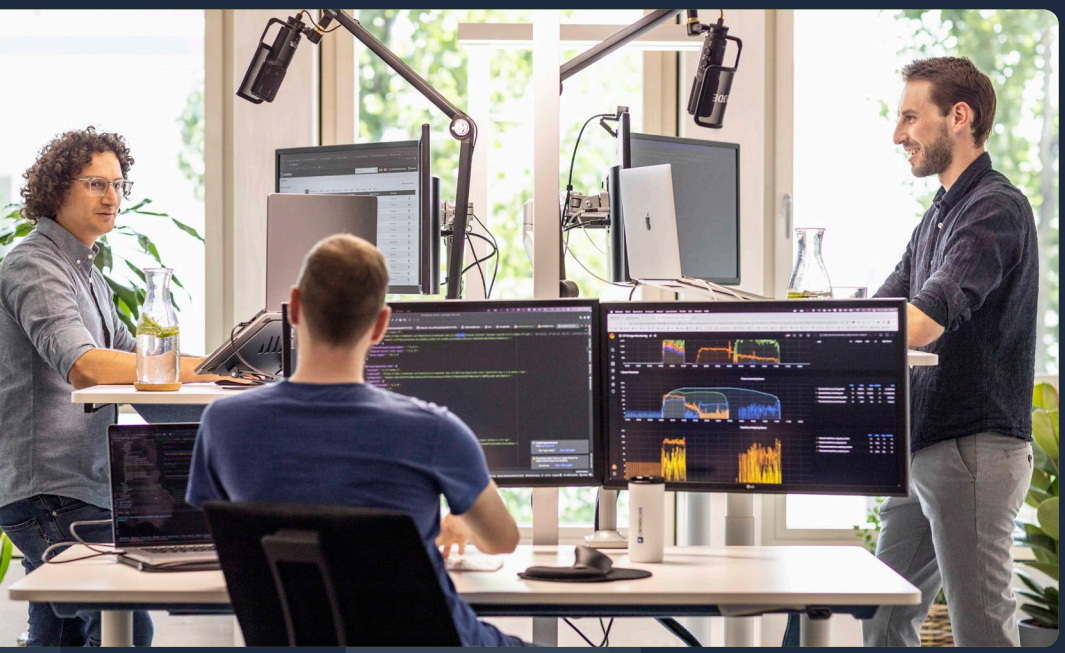
Java-Legacy-Systeme verantwortlich sind, ermöglicht es, die internen Entwicklungsressourcen zu entlasten. Dies eröffnet wertvollen Freiraum für die Modernisierung und Innovation innerhalb des Unternehmens. Daraus resultiert eine erhöhte Produktivität und Fokussierung auf Projekte, die eine strategische Neuausrichtung und technologische Innovation vorantreiben. Zusätzlich können Sie das Risiko minimieren, dass Ihre Experten das Unternehmen verlassen, weil sie sich in ihren Fähigkeiten nicht voll entfalten können.

## Systematisierung

Eine klare und effiziente Strukturierung des Umgangs mit Legacy-Systemen ist entscheidend, um die Wartungskosten kalkulierbar und übersichtlich zu gestalten. Durch die Systematisierung der Fehlerbehandlung und eine konsequente wirtschaftliche Bewertung von Problemen lassen sich Prioritäten setzen. Dies beinhaltet eine klare Definition der Restlebensdauer der Legacy-Systeme und Module, wodurch entschieden werden kann, welche Bestandteile tatsächlich einer sofortigen Wartung bedürfen und welche eventuell bis zur vollständigen Ablösung oder Migration aufrechterhalten werden können. Eine solche strukturierte Herangehensweise unterstützt eine kosteneffiziente Allokation von Ressourcen und minimiert Ineffizienzen im Wartungsprozess.

## Automatisierung

Die Automatisierung von Entwicklungsprozessen spielt eine zentrale Rolle in der Strategie zur Bewältigung der Herausforderungen von Legacy-Systemen. Durch die Implementierung von Tools und Methoden zur Automatisierung — beispielsweise Continuous Integration und Continuous Deployment (CI/CD) — lassen sich manuelle Schritte in der Softwareentwicklung und -wartung reduzieren. Dies führt zu einer signifikanten Beschleunigung der Entwicklungszyklen und stellt die Konsistenz sowie Qualität der Software sicher. Automatische Tests, etwa Regressionstests, erhöhen die Softwarequalität und minimieren das Risiko von Ausfällen und Fehlfunktionen. In Kombination mit der automatischen Überprüfung von Abhängigkeiten und Kompatibilitäten auf Sicherheitsupdates.



# 10 TEUERSTEN FEHLER VERMEIDEN

**Die Wartung von Java-Legacy-Systemen erweist sich in der Praxis oft als komplexes Unterfangen, das nicht selten von Fehlern durchzogen ist. Diese Fehler haben nicht nur technologische, sondern auch organisatorische und kommunikative Dimensionen, deren Auswirkungen tiefgreifende Konsequenzen für Unternehmen und die beteiligten Mitarbeiter mit sich bringen können. In der nachfolgenden Betrachtung werden die zehn teuersten Fehler bei der Wartung von Java-Legacy-Systemen aufgezeigt sowie deren potenzielle Folgen und Maßnahmen zu ihrer Vermeidung diskutiert.**

## 1. Fehlende Klarheit in der Modernisierungsstrategie

Ein entscheidender Fehler, der die Effizienz von Legacy-Systemwartungen untergräbt, ist das Fehlen einer klar definierten Modernisierungsstrategie. Mangelnde Transparenz und Unklarheiten bezüglich der zukünftigen Roadmap – insbesondere bezüglich des Zeitpunkts der Substitution einzelner Systeme oder Module – erschweren eine adäquate wirtschaftliche Bewertung von Bugfixing- und Wartungsmaßnahmen. Ohne dieses Verständnis wird es riskant, in kostspielige Fehlerkorrekturen zu investieren, deren Nutzen möglicherweise aufgrund bevorstehender Systemablösungen marginal oder sogar irrelevant sein könnten.

## 2. Suboptimale Ressourcennutzung

Eine suboptimale Ressourcennutzung zeigt sich daran, dass erfahrene interne Entwickler vorrangig mit der Wartung von Java-Legacy-Systemen betraut werden, um dringende Produktionsprobleme zu lösen. Dies kann deren Motivation mindern und verhindert den Einsatz dieser Fachexperten für die Entwicklung innovativer Projekte. *Entwickler geraten in ein Dilemma zwischen der Priorisierung akuter Produktionsprobleme und der Erfüllung von Zusagen zur Implementierung neuer Funktionen.* Häufige Verschiebungen oder Kürzungen von Abstimmungsmeetings stören die Kommunikation und lassen Entwickler, die in der Wartung eingesetzt werden, fühlen, als wären sie Entwickler zweiter Klasse innerhalb des Modernisierungsteams. Dies verdeutlicht die Notwendigkeit einer effektiven Ressourcenallokation, um die Motivation aufrechtzuerhalten und das Innovationspotenzial des Unternehmens voll auszuschöpfen.

## 3. Automatisierte Tests

Mangelhafte Testverfahren bei der Wartung von Java-Legacy-Systemen können dazu führen, dass kritische Fehler übersehen werden und erst im Produktivbetrieb zu ernsthaften Störungen führen. *Diese Lücken in der Testabdeckung bergen das Risiko, dass Systeme instabil werden und die Geschäftsprozesse beeinträchtigen.* Um solchen Problemen entgegenzuwirken, ist die Implementierung automatisierter und umfassender Testverfahren unumgänglich. Diese ermöglichen es, ein breites Spektrum von Fehlern frühzeitig zu identifizieren und zu beheben, womit die Zuverlässigkeit des Systems deutlich erhöht wird. Effektive Testprozesse sind somit ein zentraler Bestandteil, um die Integrität von Legacy-Systemen zu wahren und deren reibungslosen Betrieb sicherzustellen. Hier gilt insbesondere der Aufbau ausreichender End-2-End-Tests, welche den Freigabeprozess für neue Versionen beschleunigt und den Fachbereich mit dem Test auf Regressionsfehlern entlastet. Gleichfalls können auch schneller und sicherer über die Kompatibilität von Code-Änderungen oder Abhängigkeiten getroffen werden. Im Idealfall voll automatisch.

## 4. Kein adäquates Abhängigkeitsmanagement

Das Vernachlässigen des Abhängigkeits-managements bei Java-Legacy-Systemen

kann schwerwiegende Probleme verursachen, indem es zu Konflikten zwischen Bibliotheken führt und Updates erschwert. Eine schlechte Verwaltung dieser Abhängigkeiten erhöht Risiken durch Inkompatibilitäten und Sicherheitslücken, da veraltete Komponenten oft Angriffspunkte bieten. Ein proaktives Abhängigkeitsmanagement, das den Einsatz von Tools für die automatische Überwachung und Aktualisierung von Bibliotheken beinhaltet, ist entscheidend. Es fordert eine enge Zusammenarbeit im Team und eine sorgfältige Planung, um sicherzustellen, dass alle verwendeten Bibliotheken aktuell und miteinander kompatibel sind, womit die Systemstabilität und Sicherheit gewährleistet wird.

## 5. Fehlende Automatisierung

Um die Wartung von Java-Legacy-Systemen effizient zu gestalten und das hohe Fehlerpotenzial manueller Tätigkeiten zu reduzieren, ist ein hoher Grad der Automatisierung in der CI/CD-Pipeline entscheidend. Viele ältere Java-Systeme sind oft nicht vollständig oder nur durch manuelle Prozesse in moderne CI/CD-Pipelines eingebunden, was die Entwicklungs- und Deployment-Geschwindigkeit bremst und die Fehleranfälligkeit erhöht. Die nahtlose Integration und Automation dieser Systeme ermöglichen schnellere Reaktionszeiten auf Sicherheitslücken und Marktveränderungen, verbessern die Softwarequalität durch konsistente Tests und minimieren das Risiko von Deployment-Fehlern. Eine strategische Anpassung und Optimierung der CI/CD-Prozesse für Java-Legacy-Systeme ist somit unerlässlich, um zukunftssichere und effiziente Wartungsabläufe zu gewährleisten.

## 6. Fehlende Systematisierung im Fehlermanagement

Ein weiteres bedeutendes Hindernis in der Wartung von Java-Legacy-Systemen ist die fehlende Systematisierung im Umgang mit Fehlern in der Produktion. Ohne klare Prozesse und Handlungsoptionen neben den gängigen Praktiken wie erstmal Ignorieren, Workaround-Suche oder direkter Ursachenlösung bleiben viele Möglichkeiten zur effektiven Fehlerbehebung unausgeschöpft. Ein systematischer Ansatz ermöglicht es, Fehler nach ihrer Dringlichkeit und ihrem Einfluss auf das Geschäft zu klassifizieren und effiziente Maßnahmen zur Behebung einzuleiten – seien es temporäre Korrekturen oder tiefgreifende systemische Lösungen. Darüber hinaus birgt das Fehlen einer strukturierten Fehlerbehandlung das Risiko, dass dieselben oder ähnliche Probleme wiederholt auftreten und unnötig Ressourcen binden, die anderweitig eingesetzt werden könnten. Die Etablierung eines klaren Fehlermanagementprozesses, der über die simplen Optionen von Ignorieren oder Behelfslösungen hinausgeht und alle möglichen Handlungsoptionen einbezieht, ist entscheidend, um die Anpassungsfähigkeit, Sicherheit und Effizienz von Legacy-Systemen langfristig zu gewährleisten.

## 7. Mangel an Dokumentation

Der Mangel an umfassender und aktueller Dokumentation in den Bereichen Architektur-, Prozess- und Design-Entscheidungen ist eine der größten Hürden bei der Wartung von Java-Legacy-Systemen. Die Wartung und Weiterentwicklung von Systemen ohne klare Dokumentation und damit ohne klares Verständnis ihrer Funktionsweise und Architektur erhöht die Fehleranfälligkeit signifikant und verlangsamt die Arbeitsprozesse. Es ist daher essenziell, die Dokumentation nicht als nachrangige Aufgabe zu betrachten, sondern sie kontinuierlich zu pflegen, zu aktualisieren und für alle Beteiligten zugänglich zu machen. Eine gut gepflegte Dokumentation erleichtert nicht nur die Einarbeitung neuer Teammitglieder, sondern dient auch als wertvolle Ressource bei der Fehlerbehebung und der Implementierung neuer Features.

## 8. Ignorieren von technischen Schulden

Technische Schulden anzuhäufen, indem notwendige Refaktorisierungen und Optimierungen aufgeschoben

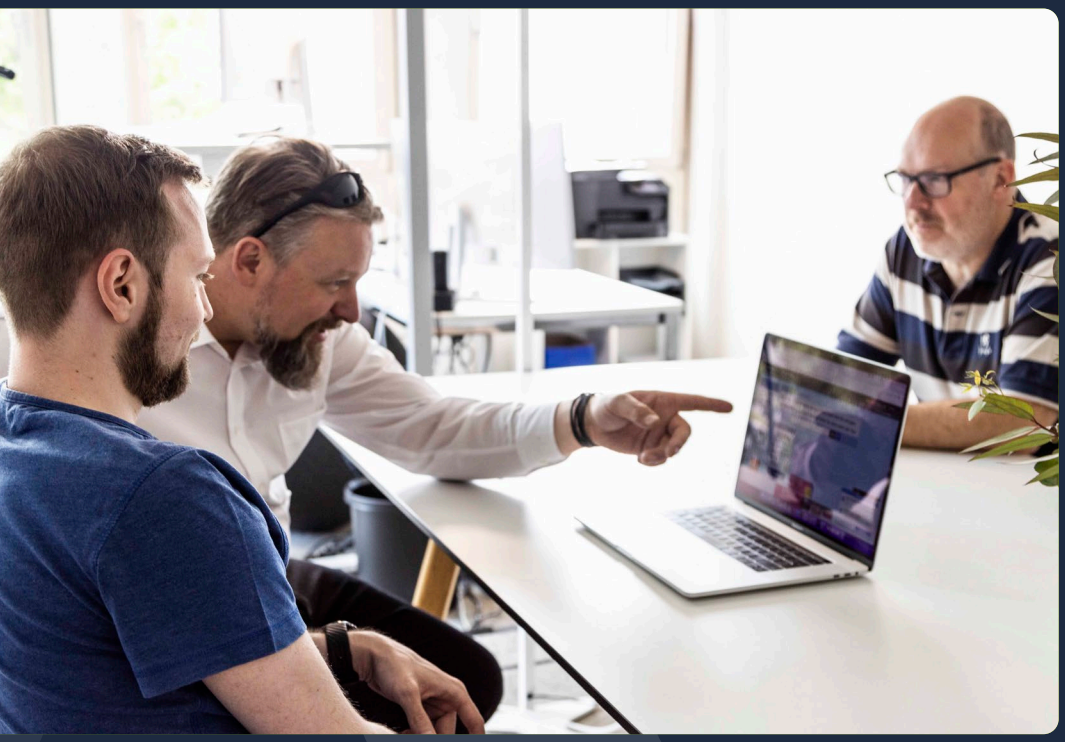
werden, ist eine weit verbreitete Praxis, die langfristig erhebliche Nachteile mit sich bringt. Zunehmende technische Schulden erschweren die Wartbarkeit und Skalierbarkeit von Systemen und können Innovationen blockieren. Durch die proaktive Adressierung und Reduzierung technischer Schulden – etwa durch regelmäßige Code-Reviews, das Anwenden von Best Practices in der Softwareentwicklung und die frühzeitige Behebung von Architekturproblemen – lassen sich zukünftige Wartungsarbeiten vereinfachen und die Grundlage für nachhaltige Innovation schaffen.

## 9. Vernachlässigte Sicherheitsupdates

Die fortwährende Vernachlässigung von Sicherheitsupdates ist insbesondere bei Java-Legacy-Systemen ein kritisches Problem. Veraltete Softwarekomponenten, die nicht regelmäßig aktualisiert werden, öffnen Tür und Tor für Sicherheitslücken, die das gesamte System kompromittieren können. Um die Sicherheit und Stabilität von Legacy-Systemen zu gewährleisten, ist es unerlässlich, Sicherheitspatches zeitnah zu installieren und bekannt gewordene Sicherheitslücken umgehend zu adressieren. Die Einführung eines systematischen Patch-Management-Prozesses hilft dabei, Sicherheitsrisiken zu minimieren und die Zuverlässigkeit der IT-Infrastruktur zu stärken.

## 10. Keine Framework-Spezialisten

Die Pflege und Weiterentwicklung von Java-Legacy-Systemen, die auf großen Frameworks basieren, erfordert tiefgehende Kenntnisse über deren interne Architektur und Funktionsweisen. Insbesondere bei inkompatiblen Updates von Frameworks ist es wichtig, Änderungen präzise zu verstehen und in Einklang mit der Architektur und den Konzepten der vorhandenen Java-Anwendungen zu bringen. Das Fehlen von Spezialisten für diese Frameworks im Team kann zu ineffizienten Lösungen, vermehrten Fehlern und erhöhtem Wartungsaufwand führen. Investitionen in die Aus- und Weiterbildung von Mitarbeitern oder die Einbeziehung externer Experten sind daher essenzielle Schritte, um die Kompetenzlücken zu schließen und die langfristige Wartbarkeit der Anwendungen zu sichern.



# MAßNAHMENPLAN ZUR LÖSUNG DER HERAUSFORDERUNGEN

**Effektive Wartung und Modernisierung von Java-Legacy-Systemen benötigen eine klar definierte Strategie und ein strukturiertes Vorgehen. Nachfolgend ein angepasster Fahrplan für Sie als Geschäftsführung und/oder IT-Verantwortlicher, um mit den Herausforderungen von Legacy-Systemen umzugehen:**

### Modernisierungsstrategie entwickeln

Beginnen Sie mit einer genauen Bewertung des aktuellen Zustands und der zukünftigen Anforderungen des Java-Legacy-Systems. Entscheiden Sie, ob eine Beibehaltung, Migration oder Substitution das Ziel sein soll. Wichtig ist eine klare Definition der Restlaufzeit oder Mindestlaufzeit des gesamten Systems sowie einzelner Module.

### Roadmap erstellen

Basierend auf der Modernisierungsstrategie sollten Sie eine detaillierte Roadmap ausarbeiten. Diese beinhaltet sowohl die geplante Migration oder Substitution von Modulen als auch Meilensteine für das Gesamtsystem. Eine Roadmap ermöglicht es, Aufgaben zu priorisieren und die Ressourcen effizient zu verteilen.

### Prozess der Fehlerbehandlung systematisieren

Eine strukturierte Vorgehensweise beim Umgang mit Fehlern ist essenziell. Durch Klassifizierung der Fehlerbilder kann die Fehlerbehebung zielgerichtet und effizient durchgeführt werden. Priorisieren Sie Fehler nach ihrer kritischen Bedeutung für das Geschäft, um den Wartungsaufwand zu verringern.

### Automatisierung innerhalb der CI/CD-Pipeline evaluieren und anpassen

Überprüfen Sie regelmäßig den Grad der Automatisierung in Ihrer CI/CD-Pipeline. Passen Sie diesen bei Bedarf an aktuelle Erfordernisse an, um den Wartungsprozess zu optimieren und eine schnelle Implementierung von Änderungen zu gewährleisten.

### Verantwortung für die Wartung definieren

Für Systeme, die aus der aktiven Entwicklung herausfallen, empfiehlt es sich, ein spezialisiertes und dediziertes Wartungsteam zu etablieren. Dies vermeidet Ressourcenkonflikte und stellt sicher, dass die Wartung effizient und fokussiert durchgeführt wird.

**Dieser Maßnahmenplan soll** Ihnen helfen, die Effizienz und Stabilität ihrer Java-Legacy-Systeme zu verbessern und gleichzeitig die Basis für eine flexible und zukunftsorientierte IT-Infrastruktur zu schaffen. Die Umsetzung verlangt Engagement und Flexibilität, um sich an neue Herausforderungen und Veränderungen anpassen zu können.

# SYSTEMATISIERUNG

**Um eine nachhaltige Systematisierung von Fehlern in Java-Legacy-Systemen zu gewährleisten, empfiehlt es sich, einen mehrdimensionalen Ansatz zu verfolgen, der sowohl die Effizienz als auch die Kosteneffektivität der Fehlerbehandlung berücksichtigt. Dieser Ansatz schlägt eine Klassifizierung von Fehlern vor, um die jeweils geeignete Handlungsoption zu bestimmen, basierend auf Faktoren wie der erwarteten Häufigkeit des Auftretens des Fehlers, dem Aufwand der Fehlerbehandlung und der verbleibenden Restlaufzeit einzelner Module oder des gesamten Systems. Die folgenden Handlungsoptionen stehen zur Verfügung und erlauben eine effiziente Ressourcenallokation sowie Maßnahmen, die im Einklang mit der Serverlebensdauer und Wichtigkeit stehen.**

### Organisatorische Lösungen im Fachbereich

Manchmal können Fehler durch Umstrukturierungen oder Anpassungen in den Fachprozessen behoben oder umgangen werden, ohne dass das IT-System selbst verändert werden muss. Dies setzt oft schnelle und kosteneffiziente Maßnahmen um, insbesondere wenn es um die Behandlung von Fehlern geht, die durch missverständliche oder veraltete Geschäftsprozesse verursacht werden.

### Manuelle Lösungen durch den Entwickler

Für Fehler, die eine einmalige oder selten auftretende Störung darstellen, kann eine manuelle Lösung durch den Entwickler, wie die Reparatur von Datenständen in der Datenbank, die sinnvollste Option sein. Diese Handlungsoption erfordert zwar direktes Eingreifen, vermeidet aber den Aufwand, den eine automatisierte Lösung oder tiefergehende Fehlerbehebung im Code mit sich bringen würde.

### Automatische Lösung der Symptome

Für wiederkehrende Fehler, deren umfassende Behebung aufgrund der Restlaufzeit des Systems nicht gerechtfertigt ist, bietet sich die Entwicklung eines Skripts oder Automatismus an. Dieser Ansatz ermöglicht es, die Auswirkungen des Fehlers zu minimieren oder zu neutralisieren, ohne den Fehler selbst dauerhaft zu beheben. Solche automatischen Lösungen sind besonders wertvoll, wenn ein Fehler zwar störend, aber nicht kritisch genug ist, um eine vollständige Fehlerbereinigung zu rechtfertigen.

### Ursache innerhalb des Java-Legacy-Systems beheben

In Fällen, wo ein Fehler eine signifikante Beeinträchtigung des Betriebs darstellt und wenn die verbleibende Restlaufzeit des Systems die Investition rechtfertigt, ist eine tiefergehende Korrektur innerhalb des Java-Legacy-Systems angebracht. Dies könnte von einfachen Code-Anpassungen bis hin zu umfassenderen Refactoring reichen, je nach der Natur des Fehlers und der Architektur des Systems.

Die Entscheidung, welche der genannten Handlungsoptionen zu wählen ist, sollte immer im Kontext einer Strategie getroffen werden, die sowohl die kurzfristigen Ziele der Fehlerbehebung als auch die langfristige Vision der Systemmodernisierung berücksichtigt. Durch diesen ausgewogenen Ansatz kann sichergestellt werden, dass Ressourcen dort eingesetzt werden, wo sie den größten Nutzen stiften, und dass Java-Legacy-Systeme so effizient und fehlerarm wie möglich betrieben werden können, bis zu ihrer geplanten Ablösung oder Migration.

# KERNTECHNOLOGIEN FÜR DIE AUTOMATISIERUNG EINER CI/CD-PIPELINE

**Die Automatisierung der Continuous Integration-/Continuous Deployment-Pipeline (CI/CD-Pipeline) ist ein entscheidender Schritt, um die Effizienz, Zuverlässigkeit und Sicherheit der Softwareentwicklung und -wartung zu steigern. Insbesondere für Java-Legacy-Systeme ist es wichtig, dass diese Automatisierung sinnvoll durchgeführt wird, um die besonderen Herausforderungen dieser Systeme zu adressieren. Im Folgenden werden Kerntechnologien und -praktiken beschrieben, die für eine ausgereifte CI/CD-Pipeline unerlässlich sind.**

### Containerisierung & Test-Container

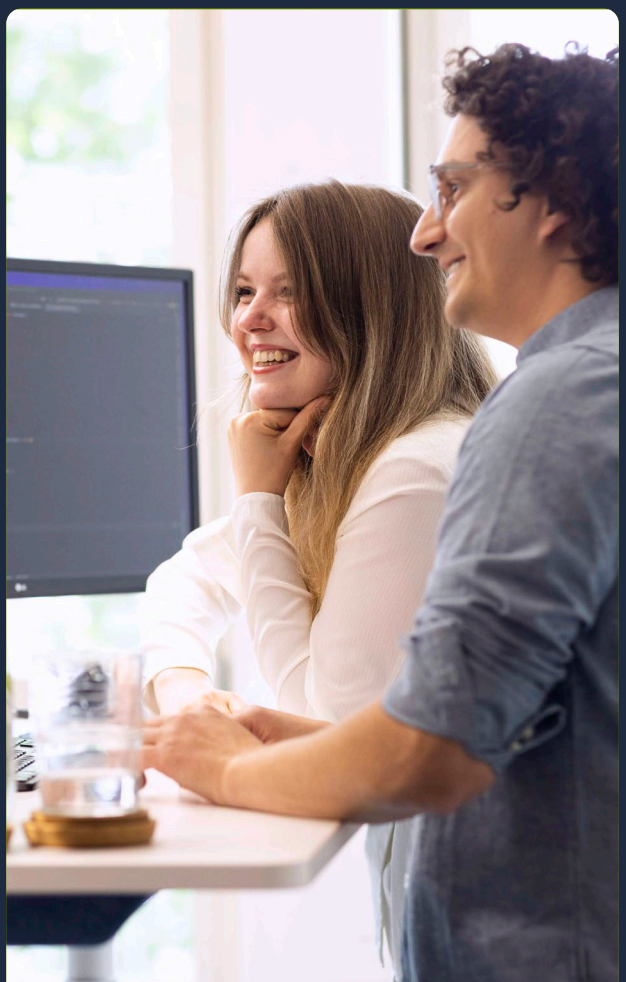
Container-Technologien wie Docker haben die Art und Weise revolutioniert, wie Anwendungen entwickelt, getestet und betrieben werden. Sie ermöglichen die Erstellung von isolierten, selfcontained Einheiten (Containern), die alle notwendigen Abhängigkeiten enthalten. Dies vereinfacht das Abhängigkeitsmanagement erheblich und gewährleistet eine konsistente Umgebung über alle Stufen des Entwicklungsprozesses hinweg. Im Kontext der Testautomatisierung bieten Test-Container eine hervorragende Lösung, um Datenbanken, Message Broker oder andere externe Abhängigkeiten in Tests zu verwenden, ohne dass dafür komplexe Mocks oder Testumgebungen aufgebaut werden müssen. So lässt sich eine hohe Isolation der Tests erreichen, und die Umgebung wird durch die Containerisierung genau definiert und kontrolliert.

### End-2-End-Tests

Für die Sicherstellung der Anwendungsqualität und die Erleichterung der Freigabe durch den Fachbereich sind End-2-End-Tests entscheidend. Werkzeuge wie Cypress ermöglichen es, komplette User Stories durchzutesten, um sicherzustellen, dass die Anwendung wie erwartet funktioniert. End-2-End-Tests spielen eine zentrale Rolle in der CI/CD-Pipeline, indem sie ermöglichen, dass automatische Builds und Deployments nur dann durchgeführt werden, wenn alle Tests erfolgreich waren. Dies fördert das Vertrauen in die Stabilität und Funktionalität der Software.

### Renovate für Kompatibilitätschecks

Das Management von Abhängigkeiten und die Sicherstellung der Kompatibilität bei Updates sind wesentliche Herausforderungen in der Softwareentwicklung. Tools wie Renovate automatisieren die Aktualisierung von Abhängigkeiten und führen Kompatibilitätschecks durch. Dadurch werden Entwicklerteams entlastet und können sich sicher sein, dass ihre Anwendung stets auf dem neuesten Stand ist, ohne dass Kompatibilitätsprobleme auftreten.



### Statische Codeanalyse

Die Bewertung technischer Schulden wird durch Werkzeuge für statische Codeanalyse unterstützt. Diese Tools analysieren den Code auf Muster, die zu Fehlern führen können, oder auf Best Practices, die nicht eingehalten werden, und unterstützen so die Identifikation und Behebung technischer Schulden. Eine regelmäßige statische Codeanalyse innerhalb der CI/CD-Pipeline hilft dabei, die Codequalität kontinuierlich zu verbessern und potenzielle Probleme frühzeitig zu erkennen.

### OWASP und Abhängigkeiten-Checks

Die Überprüfung von Sicherheitsaspekten und die Risikobewertung sind unverzichtbar für die Entwicklung und Wartung sicherer Anwendungen. Tools, die auf den Richtlinien von OWASP basieren, ermöglichen die automatisierte Überprüfung von Abhängigkeiten auf bekannte Sicherheitslücken. Diese Prüfung kann entweder manuell oder als Teil der CI/CD-Pipeline automatisiert erfolgen, um sicherzustellen, dass Risiken erkannt und angegangen werden, bevor die Software ausgeliefert wird.

**Indem diese Kerntechnologien** und -praktiken in einer CI/CD-Pipeline implementiert werden, können Entwicklungs- und Wartungsteams eine hohe Softwarequalität, schnelle Reaktionszeiten auf Änderungen und ein hohes Sicherheitsniveau erreichen. Diese Investition in die Automatisierung beleuchtet den Weg zu effizienteren, zuverlässigeren und sichereren Softwareentwicklungs- und Wartungsprozessen.

# DEDIZIERTES WARTUNGSTEAM

**Ein dediziertes Wartungsteam für Java-Legacy-Systeme nimmt eine zentrale Rolle in der fortlaufenden Pflege und Optimierung der vorhandenen IT-Infrastruktur ein. Die Teamstruktur und die Aufgabenfelder sind darauf ausgerichtet, technische Herausforderungen effizient zu bewältigen und die Lebensdauer der Systeme bei gleichzeitig hoher Funktionalität und Sicherheit zu maximieren. Ein solches Team setzt sich aus verschiedenen Spezialisten zusammen, die gemeinsam ein breites Spektrum an Fachwissen abdecken.**

### Wartungsprobleme und Kapazitäten

Wartungsarbeiten bei Java-Legacy-Systemen sind in der Regel technischer Natur, da sie oft mit Komplexitätsfallen im Source Code verknüpft sind. Die Bestimmung der Auslastung zeigt, dass für eine effektive Wartung ein Pool von 7 - 8 verschiedene Experten benötigt werden. Aufgrund der variablen Natur der Anforderungen sind aber die benötigten Rollen nicht immer gleich voll ausgelastet. Um dies zu managen, kann die Einbindung externer Dienstleister sinnvoll sein, die mit Teamanteilen arbeiten und somit Flexibilität in der Ressourcenallokation ermöglichen. Experten in der »Brownfield«-Entwicklung, also Spezialisten für die Wartung und Optimierung bestehender Java-Legacy-Systeme, spielen hier eine entscheidende Rolle.

**Ein dediziertes Wartungsteam** für Java-Legacy-Systeme umfasst eine diverse Gruppe von Experten, die zusammenarbeiten, um die Systeme zu warten, zu optimieren und gegebenenfalls zu modernisieren. Jede Rolle innerhalb dieses Teams bringt spezialisiertes Wissen und Fähigkeiten ein, die für die effektive Wartung und Weiterentwicklung des Systems notwendig sind.

### Key-User

Key-User aus dem Fachbereich spielen eine essenzielle Rolle bei der Bewertung von Fehlern und der Freigabe von Updates und Hotfixes. Sie dienen als Bindeglied zwischen den technischen Aspekten des Systems und den funktionalen Anforderungen des Geschäfts. Die Zusammenarbeit mit diesen Key-Usern stellt sicher, dass technische Lösungen den Bedürfnissen des Fachbereichs entsprechen und von diesem akzeptiert werden.

### Solution Architekt

Ein Solution Architekt setzt die langfristige technische Vision und Strategie für das Java-Legacy-System um. Er oder sie greift ein, wenn größere technologische Wechsel oder Updates erforderlich sind, indem er die besten technologischen Alternativen und Umsetzungsstrategien identifiziert. Diese Rolle ist entscheidend, um das System auf dem neuesten Stand zu halten und zukunftsfähig zu gestalten.

### Framework-Spezialisten

Framework-Spezialisten haben eine tiefgehende Kenntnis über die internen Strukturen und Funktionen der verwendeten Frameworks. Sie sind unverzichtbar, wenn es darum geht, das System an die neuesten Versionen von Frameworks anzupassen und dabei die Kompatibilität und Leistungsfähigkeit zu gewährleisten.

### Performance-Spezialisten

Performance-Spezialisten fokussieren sich auf die Identifikation und Behebung von Engpässen innerhalb des Systems. Durch gezielte Maßnahmen tragen sie zur Verbesserung der Systemleistung bei, was sich direkt auf die Nutzererfahrung auswirkt.

### Test-Spezialisten

Test-Spezialisten sind verantwortlich für die

Erweiterung der automatisierten Testabdeckung, insbesondere durch die Entwicklung und Implementierung von End-2-End-Tests.
Eine robuste Testabdeckung sichert die Qualität der Software und minimiert die Risiken bei der Implementierung von Updates und Hotfixes.

**Entwickler**
im Wartungsteam, insbesondere Experten in der »Brownfield«-Entwicklung, analysieren und beheben Fehler, entwickeln Werkzeuge zur automatisierten Behebung sich wiederholender Fehlerszenarien und implementieren Hotfixes. Sie sind auch für das Management des Release-Prozesses verantwortlich und gewährleisten, dass Änderungen effizient und sicher in die Produktionsumgebung überführt werden. Da sie tagtäglich mit dieser Aufgabe beschäftigt sind, sind sie Helden in ihrem Fachgebiet und in der Umsetzung schneller, als andere!

**DevOps-Engineer**
Ein DevOps-Engineer optimiert die CI/CD-Pipeline und den gesamten Release-Prozess. Diese Rolle ist zentral, um die Effizienz der Softwareentwicklung und -bereitstellung zu verbessern und gleichzeitig die Automatisierung zu erhöhen, was für die effektive Wartung von Legacy-Systemen essenziell ist.

**Durch die Kombination** dieser Rollen innerhalb eines dedizierten Teams können Java-Legacy-Systeme effektiv gewartet und optimiert werden. Dies stellt sicher, dass diese Systeme auch weiterhin ihren Beitrag zum Geschäftserfolg leisten können, während sie gleichzeitig an neue Technologien und Anforderungen angepasst werden.

# FAZIT UND EMPFEHLUNG

## Einführung des Managed Maintenance-Teams: Ein bewährter Prozess

**Die Wartung und Modernisierung von Java-Legacy-Software ist ein entscheidender Schritt auf dem Weg zur digitalen Transformation und zur Sicherstellung einer effizienten und stabilen Anwendungslandschaft. Mit unserem strukturierten Ansatz wird eine schnelle Übernahme der Wartung & Pflege Ihrer Java-Legacy-Software, was nicht nur eine bedeutende Entlastung Ihres Entwicklungsteams bewirkt, sondern auch den Prozess der Modernisierung beschleunigt.**

**Der von unserem Team entwickelte Prozess unterteilt sich in zwei Hauptphasen:**

1. **Die Ermittlung des Zielbildes und der Roadmap, inklusive Beratungsgespräch, Anamnese und Roadmap-Session.**
2. **Die Implementierungsphase, bestehend aus Kick-Off, Ramp-Up, Transition, Operation und Fade-Out.**

## Phase 1: Ermittlung des Zielbildes und der Roadmap

**Beratungsgespräch**

Am Anfang steht ein initiales Beratungsgespräch, in dem wir Ihre spezifischen Herausforderungen und Ziele bezüglich Ihrer Java-Legacy-Software und Modernisierungsanforderungen erörtern. Dies dient als Fundament für die Entwicklung einer effektiven Unterstützungsstrategie.

**Zielbild-Workshop und Anamnese**

In einem tiefgehenden Workshop, der sowohl die Anamnese als auch die Entwicklung eines Zielbildes umfasst, arbeiten Sie mit unseren Experten - einem Softwarearchitekten und einem Prozessexperten - zusammen, um die Hauptengpässe Ihrer Java-Legacy-Software zu identifizieren. Dies ermöglicht eine umfassende Analyse Ihrer derzeitigen Situation und legt den Grundstein für klar definierte Wartungsprozesse.

**Roadmap Session**

Im Anschluss formulieren wir in einer dedizierten Session eine maßgeschneiderte Roadmap. Diese beinhaltet konkrete Aktionen, die auf die Erreichung Ihres Zielbildes ausgerichtet sind. Wir definieren primäre Handlungsfelder und verdeutlichen, wie unser Managed Maintenance-Team gezielt Räume für Ihre Modernisierungsbemühungen freimachen kann.

## Phase 2: Implementierung

**Kick-Off für das Managed Maintenance-Team**

Das Kick-Off-Meeting bringt alle Stakeholder zusammen, um ein gemeinsames Verständnis über die Ziele und die vorgestellte Roadmap zu schaffen. Die Hauptakteure dieses Prozesses, einschließlich KeyUser, IT-Operations und das neue Maintenance-Team, tauschen sich über potenzielle Risiken und Chancen aus und stellen wichtige Kommunikationswege her.

**Ramp-Up**

Das Managed Maintenance-Team vertieft sich in dieser Phase eingehend in die Software. Durch die Einrichtung von notwendigen Zugängen und Kommunikationskanälen beginnt das Team, sich mit der Anwendungsarchitektur und den grundlegenden Software-Design-Prinzipien vertraut zu machen.

**Transition**

In der Transition-Phase nimmt das Maintenance-Team die Führung des Wartungsprozesses über. Mit systematischer Fehlerbehandlung und Maßnahmen zur Effizienzsteigerung durch Automatisierung gewährleistet das Team einen nahtlosen Übergang, unterstützt durch die ursprünglichen Entwickler im Hintergrund.

**Operation**

In der Operationsphase übernimmt das Managed Maintenance-Team vollständig die Wartungs- und Supportaufgaben, was zu einer deutlichen Entlastung der internen Entwicklungsressourcen führt. Aus Erfahrung liegt der Aufwand für Ihr Entwicklungsteam bei 5 % des ursprünglichen Wartungsaufwands. Diese Entlastung erlaubt es den internen Teams, sich auf neue Projekte und Innovationen zu konzentrieren. Die nahtlose Integration des Managed Maintenance-Teams in den täglichen Betrieb gewährleistet einen kontinuierlichen stabilen Betrieb ihrer Java-Legacy-Software.

**Fade-Out**

Die Fade-Out-Phase markiert die finale Etappe des Wartungszyklus und beginnt mit dem erfolgreichen Abschluss der Transition sowie der Implementierung neuer Systeme und Prozesse. In dieser Phase wird die betreute Legacy-Softwareversion systematisch abgeschaltet. Das Managed Maintenance-Team führt eine sorgfältige Übergabe der Verantwortlichkeiten und des operativen Know-hows an die internen Teams durch. Eine ausführliche Dokumentation der geleisteten Arbeit und der neu implementierten Systeme unterstützt die internen Teams dabei, die Verantwortung für die zukünftige Wartung und Entwicklung der Software zu übernehmen. Gleichzeitig zieht sich das Managed Maintenance-Team schrittweise zurück, bleibt jedoch für eine vordefinierte Zeit als Berater verfügbar, um die Stabilität der Systeme zu gewährleisten und Unterstützung bei eventuell auftretenden Fragen oder Herausforderungen zu bieten.

**Indem wir diesem** strukturierten Prozess folgen, stellen wir sicher, dass der Übergang so reibungslos und effektiv wie möglich verläuft.

# JETZT BERATUNGSTERMIN SICHERN!

**Wenn Sie bereit sind, sich aus dem Dilemma zwischen Modernisierung und Altlastbekämpfung zu befreien, dann vereinbaren Sie ein kostenloses Erstberatungsgespräch mit Ingo Düppe.**

## Das kostenlose Beratungsgespräch:

In einem kurzen Video-Call lernen wir uns kennen und sprechen darüber, wo Ihre akuten Herausforderungen mit Ihrer individuell entwickelten Bestandssoftware liegen und welche Hürden dabei auftreten. Auf Wunsch teilen wir gerne offen unsere Gedanken und Erfahrungen mit Ihnen und vereinbaren dann – wenn sinnvoll – gemeinsame nächste Schritte.

## Was Sie aus dem Erstgespräch direkt mitnehmen:

- Schnelles Feedback zu Ihrer Situation von einem Berater mit mehr als 25 Jahren Erfahrung in der Softwareentwicklung in mittelständischen Unternehmen und Konzernen.
- Basierend auf Ihren Wünschen und Vorstellungen bekommen Sie eine ehrliche Einschätzung, ob wir für Sie ein passender Partner sind.
- Wenn es »matcht« besprechen wir gemeinsam, welche nächsten Schritte sinnvoll sind.
- Falls wir für Ihren Fall nicht die passenden Ansprechpartner sind, dann empfehlen wir Ihnen eine Alternative.

**CROWDCODE GMBH & CO. KG**

Am Mittelhafen 16
48155 Münster

Robert-Bosch-Str. 10/3
56410 Montabaur

Tel.: +49 2602 83 900 30
kontakt@crowdcode.io