SINN UND UNSINN VON FRAMEWORKS – TEIL I

Wird für die Realisierung eines Softwarevorhabens ein Framework benötigt? Und wenn ja, welches? Das sind oft die ersten Fragen, die sich Entwickler und Auftraggeber vor einem Softwareprojekt stellen. Es sind ganz zentrale Fragen, denn die Antworten haben maßgeblichen Einfluss auf den gesamten Entwicklungsprozess. Aber wie können die Beteiligten beurteilen, ob der Einsatz eines Frameworks sinnvoll ist?

| von JOHN LOUTZENHISER

FRAMEWORKS – JA ODER NEIN?

Es gibt keine allgemeingültige Faustregel für die Wahl des richtigen Frameworks. Deswegen sind gerade Entscheidungsträger und Manager, die nicht über tiefergehendes technisches Fachwissen verfügen, mit der Auswahl eines geeigneten Frameworks oft überfordert. Dazu trägt auch die in den letzten 20 Jahren entstandene Vielzahl an neuen Technologien und Frameworks bei.

Da liegt es scheinbar nahe, diese Entscheidung gleich der Entwicklungsmannschaft zu überlassen. Oder sich von Presales-Consultants beraten zu lassen. Oder einfach ein Framework auszuwählen, das häufig verwendet wird und daher als „Standard“ gilt. Da jedoch ein Framework den Entwicklungsprozess maßgeblich beeinflussen kann, können sich Entscheidungsträger und Manager nicht (nur) auf die Meinung anderer verlassen. Sie müssen in der Lage sein, sich eine eigene, fundierte Meinung zu bilden.

Vier Argumente, die für eine eigene Meinung sprechen:

- Entwicklungsteams neigen dazu, alles selbst machen zu wollen und meiden daher fertige Frameworks.
- Oder, im Gegenteil: Entwicklungsteams wollen aus Neugier und Experimentierfreudigkeit das neueste, „coolste" Framework einsetzen, auch wenn es nicht das passendste ist.
- Verkaufsberater sind keine wirkliche Hilfe, denn sie müssen „ihr" Framework und „ihre" Beratungsleistungen verkaufen,
- Auch Frameworks, die als „Standard" gelten, können schwierig einzusetzen oder ineffizient sein.[1]

Mit unserer Artikelserie fokussieren wir darauf, wie Frameworks Softwareprojekte beeinflussen können, und helfen Ihnen, die Vorschläge zur Auswahl von Frameworks kritisch zu hinterfragen und zu beurteilen.

DER EINSATZ EINES FRAMEWORKS – EINE ENTSCHEIDUNG MIT FOLGEN

Da ein Framework in der Regel einen grundlegenden und umfassenden Rahmen für die Applikationsentwicklung vorgibt, kann die Entscheidung für den Einsatz eines Frameworks nicht mehr so einfach rückgängig gemacht werden. Diese Entscheidung impliziert eine viel größere Verpflichtung als die Entscheidung für eine Software-Bibliothek oder ein kleineres Toolkit.

Außerdem hat ein Framework potenziell eine sehr große Auswirkung auf die Arbeits- und Denkweisen der Entwickler im Team, denn die Entwickler müssen sich in ihrem Softwaredesign und ihrem Programmierstil erheblich an das Framework anpassen. Letzteres ist auch der Grund, weshalb Frameworks so viel Leidenschaft – bis hin zu Grabenkämpfen und Glaubenskriegen – unter Entwicklern wecken können. Als Entscheidungsträger muss man wissen, dass ein Framework der Entwicklungsmannschaft selten egal ist. Es ruft entweder helle Begeisterung und Freude oder aber Frustration bis hin zur Verachtung hervor – und ist damit ein nicht unerheblicher Faktor für die Produktivität und die Motivation im Team.

DER SINN VON FRAMEWORKS

Framework-Hersteller versprechen vor allem eines: dass ihr Framework die „Entwickler-Produktivität erhöht", indem es „Komplexität reduziert" und den „Entwicklungsprozess vereinfacht". Sie versprechen hingegen nicht, dass eine Software durch den Framework-Einsatz performanter wird oder eine bessere Akzeptanz beim Kunden findet, sondern lediglich, dass der Entwicklungsprozess schneller und einfacher wird.[2] Ob das jedoch zutrifft, ist eine offene Frage und hängt von vielen Faktoren ab. Ein Framework kann auch negative Auswirkungen haben. Unstrittig ist aber, dass das richtige Framework einen positiven Effekt auf folgende nichtfunktionelle Qualitätsmerkmale haben kann:

- Wiederverwendung von Code und Funktionalitäten
- Erweiterbarkeit/Änderbarkeit von Modulen und Subsystemen
- Wartbarkeit/Verständlichkeit – die (gefühlte) Komplexität des Systems wird reduziert

Dadurch kann auch der Entwicklungsprozess effizienter werden.

Wichtig ist zu verstehen, dass ein Framework

- **den Anspruch hat, positiv auf den Entwicklungsprozess und die Entwicklerproduktivität zu wirken,**
- **eine längerfristige technologische Verpflichtung darstellt**
- **eine signifikante Auswirkung auf Arbeitsweise, Denkweise, und Motivation der Entwicklungsmannschaft hat.**

WIEDERVERWENDUNG

Frameworks stellen große Mengen an technischer Infrastruktur und Funktionalitäten zur Verfügung, die sowohl innerhalb eines Projektes als auch projektübergreifend verwendet und wiederverwendet werden können. Diese scheinbar banale Tatsache ist in der Tat das stärkste Argument für den Einsatz eines Frameworks. Denn das, was das Framework liefert, muss nicht mehr selbst programmiert werden – und das kann sehr viel Zeit sparen. Hier ist es aber sehr wichtig, zu evaluieren, ob das Framework tatsächlich genau die Funktionalitäten liefert – entweder direkt „out of the box" oder durch Anpassungen/Erweiterungen, die man jetzt im Projekt und auch projektübergreifend braucht. Problematisch wird es nämlich, wenn die Entscheidung für ein Framework gefallen ist und man später feststellt, dass die gelieferten Funktionalitäten doch nicht flexibel genug sind, um Kunden oder internen Anforderungen zu genügen. Denn Änderungen am Framework selbst sind schlimmstenfalls unmöglich, bestenfalls jedoch schwierig.

Das Framework selbst sollte auch projektübergreifend Wiederverwendung finden. Ein Framework bringt Komplexität und eine Lernkurve mit, was zunächst auch Aufwand für das Entwicklungsteam bedeutet. Diesen initialen Aufwand kann man rechtfertigen, wenn das Team später Erfahrung mit dem Framework hat und folglich auch Projekte schneller mit dem Framework realisieren kann.

1 Die IT hält genug Beispiele für Technologien bereit, die einst „Standard" waren und sich im Nachhinein als problematisch erwiesen haben (wie zum Beispiel EJB 1.0).

2 Frameworks können durchaus Funktionalitäten mitbringen, die performant und attraktiv sind. Das heißt nicht, dass sie zwangsläufig besser sind als das, was ein Entwicklungsteam mit ausreichend Zeit selbst programmieren könnte. Das

Entscheidungsträger und Manager können sehr viel über die Tauglichkeit eines Frameworks für ihre Projekte herausfinden, wenn sie folgende Fragen stellen:

- Welche wiederverwendbaren Funktionalitäten bringt das Framework mit?
- Genügen diese Funktionalitäten unseren Anforderungen jetzt und auch in Zukunft?
- Wie steil ist die Lernkurve für unser Team, um mit dem Framework zu arbeiten?
- Werden wir das Framework auch für zukünftige Projekte verwenden?

ERWEITERBARKEIT UND VERÄNDERBARKEIT

Frameworks bieten Abstraktionen und Schnittstellen an, hinter denen viele Implementierungsdetails versteckt werden. Diese „Kapselung" – oder Trennung von Schnittstelle und Implementierung – ermöglicht es, Systemgrenzen zu ziehen. Hinter diesen Grenzen können Subsysteme „minimalinvasiv" ausgetauscht werden. Da, wo das Framework Schnittstellen anbietet und Implementierung nicht festlegt, bleibt das System flexibel und veränderbar.

Bei der Evaluierung von Frameworks sind daher folgende Fragen angebracht:

- Wo erwarten wir später größere Systemänderungen und welche Subsysteme müssen austauschbar bleiben?
- Ermöglicht dieses spezielle Framework Austauschbarkeit an genau diesen Stellen?
- Wo muss unser System flexibel und konfigurierbar bleiben, zum Beispiel weil Anforderungen häufig wechseln oder Änderungen auftreten?
- Ermöglicht dieses Framework an genau dieser Stelle Flexibilität?

WARTBARKEIT/VERSTÄNDLICHKEIT UND KOMPLEXITÄT

Ein Framework kann helfen, die Komplexität eines Systems langfristig zu verringern und dadurch seine Wartbarkeit zu verbessern. Damit ist aber nicht gemeint, dass ein Framework die interne oder inhärente Komplexität des Systems reduziert. Interne Komplexität wird durch die Anzahl von Komponenten und die Art der Abhängigkeiten zwischen diesen Komponenten bestimmt. Diese Komplexität wird durch ein Framework eher erhöht als reduziert, da Frameworks sehr viele Komponenten und interne Abhängigkeiten mitbringen.

Argument für Frameworks ist, dass man diese Funktionalitäten „out of the box" bekommt und sie wiederverwendbar sind.

Die Komplexität, die tatsächlich reduziert werden kann, ist die psychologische Komplexität, die Programmierer wahrnehmen, wenn sie das System weiterentwickeln und pflegen.

Beim Einsatz eines Frameworks wird oft und gerne in Kauf genommen, dass die Software um mehrere Hunderttausend Zeilen Source-Code und um Dutzende von Modulen größer und komplexer wird (interne Komplexität), wenn es nur verspricht, die Entwicklung zu erleichtern und verständlicher zu machen (psychologische Komplexität).

ESSENZIELLE UND AKZIDENTELLE KOMPLEXITÄT

Frederick P. Brooks unterscheidet in seinem Aufsatz „No Silver Bullet"[3] zwischen „essenzieller" und „akzidenteller" Komplexität. Diese Unterscheidung kann helfen, zu verstehen, wie Frameworks die psychologische Komplexität eines Systems reduzieren können.

„Essenzielle Komplexität" ist die Komplexität, die inhärent im Problem enthalten ist, das mit der Software gelöst werden soll. Zur Erläuterung: Die Verkörperung eines Problems in Softwarecode kann nicht einfacher sein als die einfachste konzeptuelle Lösung des Problems. Der vermutlich einfachste (Pseudo-)Code für das Problem „Teile 73 durch 42 und gib die Antwort aus" sieht so aus: „print 73/42".

Dahingegen sind mehrere Dutzend Zeilen Code nötig, um dieses Problem im Assembler-Code auszudrücken. Hier muss man sich mit der Rechnerarchitektur, den Registern und vielen weiteren komplizierten Details auseinandersetzen. Diese vielen Details, die nicht zum eigentlichen Problem selbst gehören, nennt Brooks „akzidentelle Komplexität".

Laut Brooks kann essenzielle Komplexität nicht durch Tools, Technologie oder Frameworks reduziert werden. Essenzielle Komplexität bleibt immer, weil sie zum Problem selbst gehört und nicht zur Technologie. Bestenfalls kann man sie beseitigen, indem man es so einfach wie möglich macht, diese Komplexität zu verstehen, zu erfassen und in Code auszudrücken.

Im Gegensatz dazu lässt sich akzidentelle Komplexität sehr wohl reduzieren. Hier können Frameworks einen wichtigen Beitrag leisten. Ersetzen wir in einer bekannten Feststellung von Brooks „Hochsprache" durch „Frameworks", dann erhalten wir eine sehr gute Beschreibung, wie ein Framework die psychologische und akzidentelle Komplexität des Systems auf einem Minimum halten kann.

3 Brooks, Frederic P.: No Silver Bullet – Essence and Accidents of Software Engineering, IEEE Computer 20 (4):10-19

„Die wahre Bedeutung von „Software-Komplexität“ ist, wie schwierig oder einfach es ist, die Software zu warten, zu ändern und zu verstehen. Es handelt sich um die psychologische Komplexität von Software-Programmen.“

Horst Zuse[4]

„[Ein Framework] befreit das Programm von seiner akzidentellen Komplexität. Dadurch, dass [das Framework] alle konzeptionellen Konstrukte liefern kann, die in dem abstrakten Programm erwünscht sind, und alle anderen vermeidet, beseitigt es eine Menge an Komplexität, die nie zum eigentlichen Problem gehört hat. [...] [Ein Framework] kann nicht mehr machen, als die Konstrukte liefern, die der Programmierer braucht, um die Essenz des Problems zu lösen.“[5]

FAZIT

Um also die Fragen beantworten zu können, ob für die Realisierung eines Softwarevorhabens ein Framework benötigt wird, und wenn ja, welches, sollte man bei der Framework-Evaluierung kritisch hinterfragen:

- Was ist die Essenz des Problems, das es zu lösen gilt?
- Wie stellt man sich die abstrakte Lösung vor, abgesehen und unabhängig von einer spezifischen Technologie?
- Welche konzeptionellen Konstrukte braucht das Entwicklungsteam, um das Problem zu lösen?
- Inwiefern liefert das Framework genau diese Konstrukte und vermeidet andere, die nicht gebraucht werden?
- Hilft das Framework dabei, die Essenz eines Problems klar darzulegen und lösbar zu machen?
- Inwiefern kapselt und versteckt das Framework die Komplexität, die nicht zum eigentlichen Problem gehört, und hält sie vom Entwicklungsteam fern?
- Bringt das Framework seine eigene Komplexität mit? Wenn ja, steht diese in Relation zu der Komplexität, die es beseitigt?

Die Artikelserie wird fortgesetzt. Im **zweiten Teil**[6] beschäftigen wir uns mit potenziellen Tücken beim Einsatz von Frameworks. ●

ANSPRECHPARTNER – JOHN LOUTZENHISER

Senior IT Consultant

Public Sector

- +49 173 859 4235
- john.loutzenhiser@msg-systems.com

4 Zuse, Horst: Software Complexity: Measures and Methods, New York: De Gruyter 1990, S. 5

5 Brooks, Frederic P.: No Silver Bullet –Essence and Accidents of Software Engineering, IEEE Computer 20 (4):10-19

6 Erscheint in .public 02-2015