Erfahrungen mit der Modularisierung einer gewachsenen Anwendungslandschaft

Dr. Andreas Reichenbacher, msgGillardon AG, andreas.reichenbacher@msg-gillardon.de
Dieter Ebhart, msgGillardon AG, dieter.ebhart@msg-gillardon.de

Zusammenfassung

Kritische Erfolgsfaktoren für die Modularisierung einer langjährig gewachsenen Anwendungslandschaft, wie sie auch das Produkt THINC in der Banksteuerung darstellt, sind aus unserer Erfahrung folgende vier Punkte:

- Zerlegung des Gesamtvorhabens in machbare kleine Schritte, die eine frühzeitige Überprüfung des Lösungsansatzes erlauben. Dabei ermöglicht ein Vorgehen von „außen nach innen“, d.h. das Schaffen von Schnittstellen für den externen Zugriff als ersten Schritt, schon frühzeitig einen Einsatz der Software in modularen Settings.
- Arbeiten mit Analogien, Mustern und Bildern, z.B. Stecker/Steckdose (vgl. auch Abbildung 3), um ein homogenes Lösungsverständnis in großen und standortübergreifenden Teams zu schaffen.
- Zusammenbringen der technischen, fachlichen, planerischen und budgetären Sichten bei der Projektsteuerung.
- Timeboxing um eine Fokussierung auf die wesentlichen Mehrwerte zu erleichtern.

Im Folgenden beleuchten wir unser Vorgehen sowie die zugrundeliegenden Rahmenbedingungen.

Einleitung

Die deutsche Bankenlandschaft ist geprägt durch rückläufige Erträge, stagnierende Kosten und eine steigende Cost-Income Ratio. Diese stieg 2017 auf 72% (vgl. [SISC18]). Deutsche Banken müssen also im Schnitt 72 Cent ausgeben, um einen Euro zu verdienen. Im Vergleich dazu liegen die effektivsten europäischen Großbanken bei deutlich unter 60 Cent pro Euro (vgl. [MAIS18]).

Zur Verbesserung der Cost-Income Ratio empfiehlt Bain & Company den Banken den Fokus unter anderem auf „Automatisierung und Digitalisierung“ sowie die „Transformation der IT“ zu setzen.

Die optimale Ausrichtung der Anwendungslandschaft, insbesondere bei Geschäftsprozessen die nicht vorrangig der Wertschöpfung dienen, rückt immer stärker in den Fokus. Nicht zuletzt steigt durch regulatorische Anforderungen auch der Bedarf an Datenqualität, Datenanreicherung und definiertem Datenstand zum Verarbeitungszeitpunkt.

Bestehende, gewachsene Anwendungslandschaften haben im Allgemeinen eine sehr hohe Komplexität und verursachen durch interne Abhängigkeiten und redundante Ablage gleicher fachlicher Daten hohe Kosten in Betrieb, Wartung und Weiterentwicklung.

Um diese Herausforderungen zu erfüllen, muss die Granularität der fachlichen Module deutlich reduziert werden, um sowohl die Komplexität bei Änderungen und Weiterentwicklung zu minimieren, als auch die Softwarelösung noch besser auf bestehende Kundenprozesse maßschneidern zu können. Die Anforderungen an Datenqualität, Datenstand und Vermeidung von Redundanzen können durch den Einsatz eines zentralen Datawarehouses erfüllt werden.

Im Folgenden beschreiben wir, am Beispiel der Produktsuite THINC, wie wir mit obigen Herausforderungen umgegangen sind.

Die Produktsuite THINC von msgGillardon AG ist eine Lösung für die Gesamtbanksteuerung, die sowohl die periodische Ergebnisvorschaurechnung sowie ein wertorientiertes Risikomanagement erlaubt.

Anforderungen und spezifische Rahmenbedingungen

Die grundlegenden Anforderungen an die Modularisierung von THINC waren:

- Dass die Modularisierung der fachlichen Anwendungslogik folgt, da diese erfahrungsgemäß der langlebigere Anteil in einem Softwaresystem und deshalb für die Strukturierung besser geeignet ist als technische Aspekte (vgl. [SIED04]).
- Lose fachliche Koppelung der Module, d.h. jedes Modul bildet einen klar abgegrenzten fachlichen Kontext ab.

- Alle Eigenschaften eines Moduls, deren Sichtbarkeit nach außen nicht zwingend erforderlich sind, werden in diesem einen Modul verborgen. Dazu werden Funktionalität und zugehörige Daten eindeutig einem Modul zugeordnet. Alle übrigen Module greifen nur über definierte Schnittstellen auf dieses eine Modul zu und machen keine Annahmen über diese Eigenschaften (vgl. Geheimnisprinzip nach Parnas [PARN72]).

Diese Prinzipien harmonieren sehr gut mit dem Domain-driven Design (vgl. [VAUG16]). Domain-driven Design basiert auf folgenden zwei Annahmen:

- Der Schwerpunkt des Softwaredesigns liegt auf der Fachlichkeit und der Fachlogik.
- Der Entwurf komplexer fachlicher Zusammenhänge sollte auf einem Modell der Anwendungsdomäne, dem Domänenmodell basieren (vgl. [Wiki1]).

In unserem konkreten Fall haben wir die Domäne „Banksteuerung“ gewählt und in ihre Sub-Domänen aufgeteilt (vgl. Abbildung 1: *Domäne „Banksteuerung“ und Ihre Sub-Domänen (exemplarisch)*).
Im ersten Schritt wurde die Sub-Domäne Vorschaurechnung als Kerndomäne ausgewählt.

Für diese Sub-Domäne wurden zunächst die fachlichen Prozesse näher betrachtet, in denen unsere Lösung heute von unseren Kunden eingesetzt wird.

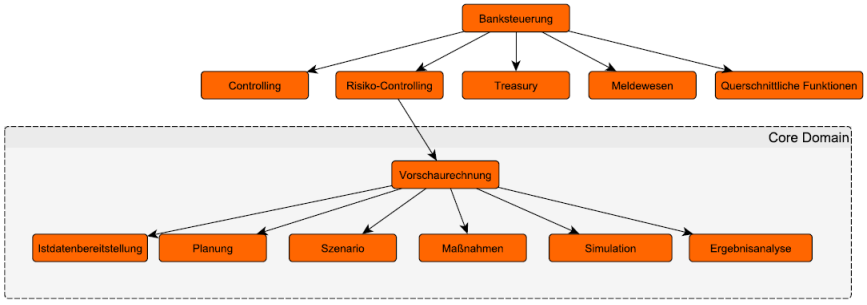

Abbildung 1: Domäne „Banksteuerung“ und Ihre Sub-Domänen (exemplarisch)

Zur Identifikation der Kontextgrenzen haben wir uns an den folgenden Fragestellungen orientiert:

1. Welche Prozesse bzw. Workflows bilden eine geschlossene Einheit mit definiertem Ergebnisoutput?
2. Welche Funktionalitäten, Schritte oder Ergebnisse werden in mehreren Anwendungsfällen verwendet?

Dieses Vorgehen kann iterativ angewendet werden, was unserer Erfahrung nach mehrere Vorteile bietet. Zum einen war es uns möglich, verschiedene Granularitäten zu vergleichen, zum anderen vereinfachte dieses Vorgehen es auch, mit einem „Ersten Schritt“ zu starten und mit den in der Umsetzung gewonnen Kenntnissen iterativ den Lösungsansatz zu verbessern. Somit konnten wir die Komplexität der Lösungsfindung deutlich reduzieren und laufend an die gewonnenen Erkenntnisse anpassen.

Initial haben wir die Kontexte „Planung“, „Maßnahmen“, Szenarien“ und „Simulation“ zur Modularisierung ausgewählt.

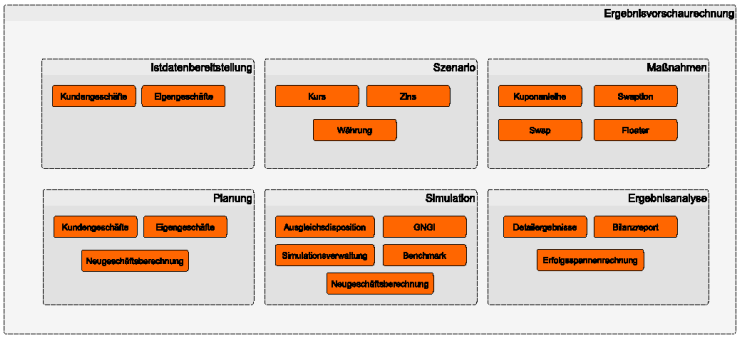

Abbildung 2: Details zu den gewählten Kontexten

Da die Standardsoftware THINC parallel zum Umbau auch weiterhin in halbjährlichen Releases an neue Anforderungen angepasst werden musste, konnte dieser Umbau nur schrittweise erfolgen. Neben dem halbjährlichen Releasezyklus galt es auch die folgenden internen und externen Rahmenbedingungen zu berücksichtigen:

- Diese Umstrukturierung muss zeitlich begrenzt sein, um am Markt für die Kunden nach einer definierten Zeit wieder fachliche Weiterentwicklungen in gewohnter Geschwindigkeit anbieten zu können.
- Auf Kundenseite muss frühzeitig, möglichst schon vor endgültiger Fertigstellung, mit der Anbindung der Module begonnen werden um einen reibungslosen Übergang zu den Komponenten zu gewährleisten.

- Parallel zur „technischen Umstellung“ mussten auch zwingend notwendige fachliche Weiterentwicklungen in den Komponenten erfolgen.

Unser Vorgehen

Um diese Rahmenbedingungen zu erfüllen, haben wir das Vorgehen wie folgt strukturiert:

1) Definition eines Projektes mit festem Zeithorizont und klar abgegrenzten Inhalten.
2) Detaillierung der Vorgehensweise in die folgenden 4 Schritte:

Schritt 1: Definition der Schnittstellen
Dieser Schritt dient dazu, die Datenströme aus fachlicher Sicht abzustimmen. Die so entstehenden Beschreibungen der Schnittstellensignaturen müssen am Ende dieses Schrittes detailliert und weitestgehend stabil sein.

Schritt 2: Prototypische Umsetzung der Schnittstellen
Auf Basis der Schnittstellenbeschreibungen können die Schnittstellen-Signaturen implementiert werden. Diese werden noch nicht an die Fachlogik angebunden (Steckdose ohne Verkabelung).

Schritt 3: **Anschließen der Schnittstellen an die bestehenden Workflows und Anwendungsfälle**
Nach vollständiger Implementierung der Schnittstellen inkl. Ausnahmebehandlung und Anschluss an die Fachlogik können die Schnittstellen vollständig an die bestehenden Anwendungsfälle angebunden werden (Steckdose inklusive Verkabelung).

Schritt 4: Umsetzung der fachlichen Erweiterungen
Die in den Schnittstellensignaturen schon berücksichtigten fachlichen Erweiterungen konnten in unserem Fall nachgelagert implementiert werden. Erst nach Implementierung der Erweiterungen waren alle über die Schnittstelle angebotenen Berechnungen fachlich korrekt.

Unsere Erfahrungen bei der Umsetzung

In der Detaillierung und Umsetzung des Vorhabens galt es für uns in den einzelnen Schritten weitere Herausforderungen zu bewältigen, welche wir im Weiteren gemeinsam mit unseren Lösungsansätzen und Erfolgsfaktoren näher beleuchten wollen.

Schritt 1: Definition der Schnittstellen

Da auch die Schnittstellenauswirkungen der fachlichen Erweiterungen frühzeitig für Abstimmungen zur Verfügung stehen mussten, gab es früh im Projekt einen größeren Block fachlicher Konzeption abzuarbeiten. Dies hatte auch einen erhöhten Bedarf an fachlicher Koordination und Abstimmung zur Folge.

Der Wechsel von einer Datenversorgung durch eine eigene Datenbank mit direkten Zugriffen, hin zu einer Versorgung durch ein zentrales Data-Warehouse mithilfe von Schnittstellen, bedeutete in einigen Fällen auch umfangreichere Änderungen des Datenflusses und der Workflows.

Ein Beispiel dafür sind Kopfdaten und Hierarchien, die bisher in Systemen außerhalb der gewählten fachlichen Kontexte (siehe Abbildung 2) administriert wurden. Hier galt es Funktionalitäten, die diese Daten benötigen und deshalb direkt auf die Datentöpfe anderer Systeme zugegriffen haben, mit neuen oder alternativen Datenquellen über die Schnittstellen zu versorgen.

Ein weiteres Beispiel ist die Abbildung und Bewertung von Produkten des Eigengeschäftes. Wurde bisher für jede Simulation der Ergebnisvorschau von einem Nachbarsystem die zukünftige Entwicklung unter den verwendeten Marktannahmen explizit angefordert und bereitgestellt, so ist in der neuen Architektur eine fremdgesteuerte Belieferung von Quellen außerhalb zu ermöglichen. Hierfür mussten zunächst entsprechende Datentöpfe konzipiert und implementiert werden. Zudem war auch die Konfiguration der Simulationen so zu erweitern, dass der Anwender das zu seinen weiteren Simulationsparametern passende Datenset erkennen und auswählen konnte.

Die Externalisierung bisher interner Schnittstellen bedingt eine umfangreiche Dokumentation, welche in der gewachsenen Systemlandschaft nicht in ausreichender Qualität zur Verfügung stand.

Die vorbereitende Analyse der über mehrere Jahre gewachsenen Datenstrukturen zeigt oft Optimierungsbedarfe der Modellierung und notwendige Flexibilisierungen auf.
Deshalb erarbeiteten wir uns zunächst ein grobes fachliches und technisches Zielbild, um uns

anschließend direkt auf die Schnittstellen zu fokussieren.

Schritt 2: Prototypische Umsetzung der Schnittstellen

Aufbauend auf die im ersten Schritt erarbeiteten Schnittstellenbeschreibungen galt es einen „Fertigstellungsgrad“ zu definieren, der einerseits die Abstimmung mit dem Kunden und die prototypische Anbindung ermöglicht, andererseits aber auch eine möglichst schnelle Lieferung der Schnittstellensignaturen erlaubt, um möglichst frühzeitig Rückmeldung zum Komponentenschnitt und den Schnittstellen zu erhalten.

Bei der Parallelisierung der Umsetzungspakete gab es für uns verschiedene Abhängigkeiten zwischen den Schnittstellen zu berücksichtigen.

Bei „Roundtrip-Schnittstellenpaaren“ sollte sowohl die Input- wie auch die Output-Schnittstelle vom selben Team umgesetzt werden, um den Abstimmungsbedarf zu minimieren. Roundtrip bedeutet in diesem Fall, dass die Formate der Input- und Output-Schnittstelle übereinstimmen und so ermöglichen, dass ein Anwender eine Datei exportiert, Änderungen im Sinne einer Datenpflege vornimmt und diese direkt wieder importieren kann.

Die Output-Formate prozessual vorgelagerter Komponenten müssen mit den Input-Formaten nachgelagerter Komponenten kompatibel sein.

Zur Kommunikation an die beteiligten Teams und für ein besseres Verständnis der Abgrenzung von Schritt 2 zu Schritt 3 hat sich aus unserer Sicht das Bild „Stecker – Steckdose“ bewährt: Die Steckdose muss fertig gestellt sein, um mit der

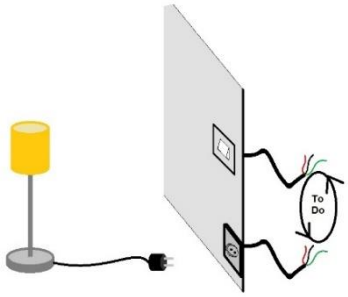

Abbildung 3: Stecker, Steckdose und Verkabelung

Produktion und Anpassung der Stecker beginnen zu können, aber die Verkabelung in der Wand wird erst im nächsten Schritt verlegt und angeschlossen.

Bei einigen Anwendungsfällen war es hilfreich, die Testdaten einzelner Schnittstellen zwischen den verschiedenen Teams zu teilen oder direkt in die Datenbank einzuspielen und so Abhängigkeiten zwischen verschiedenen Input- und Output-Schnittstellen zu reduzieren. Zudem konnte auf diese Weise schon vor Abschluss von Schritt 3 mit dem Test weiterer Anwendungsfälle begonnen werden.

Schritt 3: Anschließen der Schnittstellen an die bestehenden Workflows und Anwendungsfälle

Nach Abschluss der Implementierung der Schnittstellensignaturen wurden die bestehenden Funktionalitäten an die neue Datenversorgung angebunden. Insbesondere galt es eine Vielzahl von Abhängigkeiten einzelner Funktionalitäten zu anderen Komponenten zu identifizieren und aufzulösen.

Hierbei war ein wesentlicher Erfolgsfaktor, die Umsetzung entlang der Kundenprozesse einzuplanen. So konnten frühzeitig in sich geschlossene fachliche Pakete umgesetzt werden. Dies wiederum ermöglichte technische und fachliche Zwischenabnahmen. Diese wiederum halfen, zunächst unbekannte Abhängigkeiten frühzeitig aufzudecken und die nachfolgenden Schritte auf einem bereits qualitätsgesicherten Stand aufzusetzen.

Die Orientierung an den Kundenprozessen ermöglicht eine Aufteilung notwendiger Anpassungen am Workflow und am Datenfluss in natürliche und intuitive Teilpakete. Diese Zerlegung erleichtert die Beherrschung der Gesamtkomplexität. Allerdings konnten nicht alle Arbeiten in unabhängige Teilpakete unterteilt werden, da sowohl für den Test von im Prozess nachgelagerten Funktionalitäten als auch für integrative Regressionstests eine abgestimmte Testdatenbasis benötigt wurde. Auch an dieser Stelle erwies sich die wöchentliche Abstimmung der Beteiligten als ungemein hilfreich.

Schritt 4: Umsetzung der fachlichen Erweiterungen

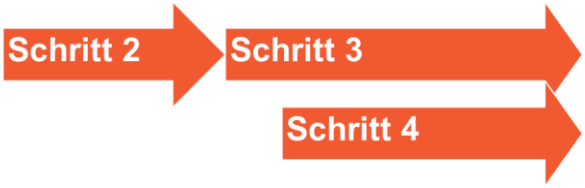

Abbildung 4: Parallelisierung

Entgegen der durch die Nummerierung

implizierten zeitlichen Reihenfolge fand die Umsetzung der fachlichen Erweiterungen nicht ausschließlich nach Schritt 3, sondern weitestgehend parallelisiert statt.

Auch hier orientierte sich die Zerlegung in Teilpakete entlang der Kundenprozesse und ermöglichte so:

- Eine parallele Weiterentwicklung, wenn Workflows oder Datenflüsse fast vollständig neu konzipiert werden mussten oder notwendige Anpassungen frühzeitig hinreichend transparent waren.
- Ein iteratives Vorgehen mit der Möglichkeit Zwischenschritte fachlich und technisch abzusichern. Dies kam insbesondere in komplexen oder schwierig zu analysierenden Fällen zum Tragen. Auch in Fällen, in denen im Rahmen der Konzeption der fachlichen Erweiterungen ein Anpassungsbedarf an den Schnittstellen identifiziert wurde, konnte dieser so mit minimalen Reibungsverlusten eingearbeitet werden.

Regelmäßiges Timeboxing der Teilpakete und ein striktes Priorisieren der Arbeitspakete erwies sich als sehr hilfreich, um einerseits unsere eigene Planungssicherheit zu gewährleisten und andererseits die Grundlage für eine verlässliche Kundenkommunikation und Verzahnung mit Integrationsarbeiten zu schaffen. Zusätzlich konnten durch dieses Vorgehen Optimierungspotentiale übergreifend identifiziert, bewertet und im späteren Projektverlauf umgesetzt werden.

Fazit

Mit dem im vorigen Kapitel beschriebenen Vorgehen konnten wir die Modularisierung unserer Produktsuite einen entscheidenden Schritt voranbringen. Insbesondere die Definition und Implementierung der Schnittstellen ermöglicht es uns, einen ersten Teil unserer Produktsuite unabhängig und in Zusammenspiel mit einem zentralen Datawarehouse einzusetzen. Der erarbeitete fachliche Schnitt hat sich als tragfähig und praxistauglich erwiesen.

Ein wesentlicher Erfolgsfaktor des Umsetzungsprojektes war ein Projektleitungsteam aus Projektleiter, fachlichem und technischem Architekt sowie einem Vertreter der internen Auftraggeber mit intensiven Abstimmungen alle zwei Tage. Weiter setzten wir auf wöchentliche Abstimmung aller konzeptionell Arbeitenden und des technischen Architekten um Reibungsverluste beim Abgleich aktueller Entwicklungen mit dem Zielbild zu minimieren und die Transparenz des Zielbildes zu erhöhen. Die zum Reengineering interner Schnittstellen notwendigen Mitarbeitenden benachbarter Abteilungen wurden für die Dauer ihrer Tätigkeiten ebenfalls in diese Struktur integriert.

Literatur:

[SISC18] Walter Sinn & Dr. Wilhelm Schmundt (2018) Deutschlands Banken 2018: Schneller, stärker … und rentabler? S. 13, Hrsg.: Bain & Company München

[MAIS18] Michael Maisch: Diese 5 Charts zeigen die Probleme von Deutscher Bank und Commerzbank in Handelsblatt vom 30.6.2018. https://www.handelsblatt.com/finanzen/banken-versicherungen/bankenmarkt-diese-5-charts-zeigen-die-probleme-von-deutscher-bank-und-commerzbank/22751422.html

[SBMS19] Dr. Sebastian Bindick, Michael Stoye: Von Monolithen zu Microservices: Domain-Driven Design als Schlüssel zum Erfolg https://entwickler.de/online/development/microservices-domain-driven-design-579861186.html

[MKAG16] Martin Kernland, Adrian Gygax: DDD: Amazing! Einführung in die Konzepte von Domain-driven Design. https://jaxenter.de/domain-driven-design-2475

[VAUG16] Vaughn Vernon: Domain-Driven Design Distilled, Addison-Wesley 2016, ISBN 10-0-13-443442-0.

[SIED04]: Johannes Siedersleben: Moderne Softwarearchitektur; dpunkt.verlag 2004 ISBN 978-3-89864-292-7. Auszüge in: https://beckassets.blob.core.windows.net/product/readingsample/399323/9783898642927_excerpt_001.pdf

[PARN72] – On the Criteria To Be Used in Decomposing Systems into Modules, David L. Parnas, in: Communications of the ACM, 1972

[Wiki1] – Domain-driven Design, https://de.wikipedia.org/wiki/Domain-driven_Design