Ausgabe 02|2023
Deutschland € 15,90 Österreich € 16,90 Schweiz sfr 24,20
4 194164 115901 02

www.ITSpektrum.de

vormals OBJEKTspektrum

IT Spektrum

Digitaler Wandel & Software-Architektur für Profis

Green IT:

Methoden für zielgerichtete Softwareentwicklung

Green IT: Methoden für zielgerichtete Softwareentwicklung

Docker & Co

Container brauchen Security und Monitoring

Den Erfolg stets im Blick

Methoden für planvolle Softwareentwicklung

Schöne neue API-Welt

API-Gateway – Das unbekannte Wesen

G 6540F

Methoden für zielgerichtete Softwareentwicklung

Den Erfolg stets im Blick!

Unser Verständnis von Softwareentwicklung ist im permanenten Wandel, geprägt durch neue Einflüsse und Erkenntnisse. In den letzten Jahren ist die IT stark durch Agilität beeinflusst worden, doch auch viele andere Themen wie User Experience Design, Produktorientierung oder Domain-Driven Design geben neue Impulse. Doch wie passt alles zusammen? Wie finden sich Teams zurecht in der Vielzahl von Techniken? Und vor allem: Wie behalten Teams den Erfolg während der Softwareentwicklung stets im Blick?

Ein methodischer Rahmen für die Softwareentwicklung sollte vor allem eines leisten: Praktische Orientierung im Dschungel der Praktiken und Techniken bieten, zugleich aber auch genügend Flexibilität einräumen, sodass Teams spezifisch für ihren Kontext individuell Entscheidungen treffen können.

Agile Design Applied Method

Unter diesem Leitgedanken haben wir bei Capgemini mit A·D·A·M (Agile Design Applied Method) einen methodischen Rahmen geschaffen, der auf verschiedenen Ansätzen der modernen Softwareentwicklung basiert und zugleich gesammelte Erfahrungen aus vielen Entwicklungsprojekten und mehreren Unternehmenseinheiten widerspiegelt. Besonders ist dabei, dass wesentliche Phasen des Produktlebenszyklus adressiert werden und somit ein umfassender Ordnungsrahmen entsteht, in dem verschiedenste Techniken einsortiert werden können, um deren Verwendungszweck und auch das Zusammenspiel von Techniken untereinander aufzuzeigen. In diesem Artikel teilen wir die wesentlichen Ideen und Konzepte von A·D·A·M.

Das Grundgerüst von A·D·A·M bilden vier Bausteine, deren Schwerpunkte durch jeweils drei Schlüsselkonzepte geprägt sind (**siehe Abbildung 1**). In den nächsten Abschnitten werden die Bausteine Strategize, Discover, Deliver und Measure im Detail motiviert. Im letzten Abschnitt wird dargestellt wie die vier Bausteine des methodischen Rahmens zu einem praktischen Baukasten mit konkreten Empfehlungen ausgebaut werden können.

Strategize

Jedes Team hat den Anspruch, erfolgreich zu sein. Doch was bedeutet es eigentlich, bei der Softwareentwicklung erfolgreich

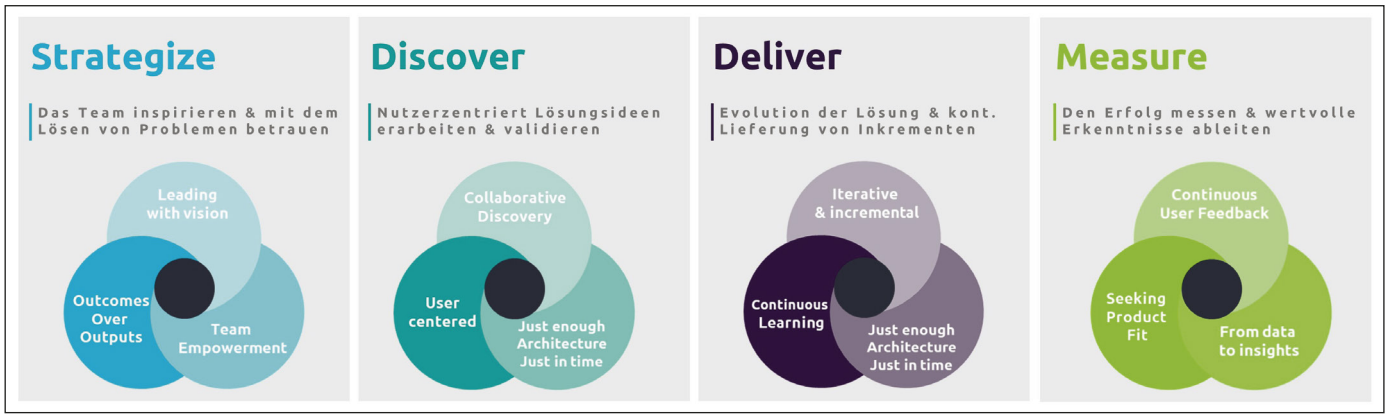

Abb. 1: Die vier Bausteine von A·D·A·M und die jeweils zugehörigen Schlüsselkonzepte (Alle Rechte liegen bei Capgemini. © 2022)

zu sein? Dabei spielt die Veränderung von Projekt- hin zu Produkt-Denken eine entscheidende Rolle. Oftmals befinden sich Teams in engen Projektstrukturen, die sich typischerweise in drei Dimensionen ausdrücken:

- ein festgelegter Umfang in Form von Anforderungen oder Features,
- ein Zeitrahmen zur Umsetzung sowie
- ein zur Verfügung stehendes Budget.

Das Projektteam arbeitet entsprechend darauf hin, die gesteckten Ziele zu erfüllen, und das Softwareprojekt wird zunächst als erfolgreich bewertet, wenn es sich im Rahmen der gesteckten Dimensionen bewegt. Je weniger Zeit benötigt wird oder je mehr geliefert werden kann, desto besser! Jetzt mag der ein oder andere sagen, dass dies doch eher ein Problem von Wasserfallprojekten ist. Ja, das ist natürlich ein sehr großes Problem in Wasserfallprojekten, doch auch in agilen Projekten begegnen Teams derselben Problematik, jedoch in kleinerem Umfang: Der Zeitrahmen ist nun zum Beispiel die Sprintdauer, der Umfang das Sprintbacklog und das Budget die Kosten für das Team während des Sprints. Auch hier ist das Team vermeintlich erfolgreich, wenn es im Sprint das Geplante liefert („*done*"). Und je mehr das Team liefert („höhere *Velocity*"), desto besser!
Wenn ein Team Features liefert und Anforderungen erfüllt, hat es zunächst einmal nur ein Ergebnis (*Output*) erzeugt ohne Beleg dafür, dass auch ein tatsächlicher Mehrwert entstanden ist. Wie häufig stellen wir jedoch fest, dass eine Lösung den erhofften Mehrwert nicht erzielt? Und wie häufig ist dem Team gar nicht bewusst, welcher Mehrwert entstehen soll, da es nur Anforderungen und Features übergeben bekommt? Daher ist eine ganzheitliche Betrachtung nötig, die weniger in Projekten organisiert ist, sondern die Lösung als Produkt über seinen gesamten Lebenszyklus betrachtet und dabei entlang der Wertschöpfung ausgerichtet ist:

- Welche Probleme sind zu lösen?
- Welche Vision hat das Team für die Zukunft und was soll sich wie verändern?
- Wie kann das Team die Erreichung der Ziele durch Metriken messbar nachhalten?

Die Beantwortung dieser Fragen ist entscheidend für ein echtes *Empowerment* eines Teams. Denn ein Team, welches beauftragt wird, Probleme zu lösen, wird folglich befähigt, die geeignetste Lösung zu finden, welche die Problemstellung adressiert, und kann die Erreichung von Zielen durch Metriken nachweislich validieren. Metriken zu finden, die auch im direkten Bezug zur Problemstellung stehen und vor allem gut messbar sind, ist jedoch nicht einfach und erfordert eine tiefe Auseinandersetzung mit der Domäne. Geschäftliche KPIs, die zum Beispiel Umsatz oder Kosten beleuchten, helfen hier wenig. Es empfiehlt sich vielmehr, den Fokus darauf zu legen, welche Veränderung die Lösung bei den Nutzern, Kunden oder innerhalb der Organisation herbeiführt.
Josh Seiden hat in diesem Zusammenhang den Begriff *Outcome* mit folgender Definition geprägt: „*an outcome is a change in human behavior that drives business results*" [Sei20]. Teams, die darauf aufbauend Ziele als *Outcomes* definieren, erhalten einen entscheidenden Vorteil: Verhaltungsänderungen sind sehr gut beobachtbar und damit messbar. Allgemein lassen sich Veränderungen gut in drei Bereiche gliedern:

- *Effizienz:* Profitieren Nutzer oder die Organisation durch eine effizientere Abwicklung von zum Beispiel Aufgaben?
- *Effektivität:* Werden Aufgaben oder Prozesse effektiver ausgeführt, indem zum Beispiel die Qualität gesteigert wird oder Fehler gesenkt werden?
- *Zufriedenheit:* Steigt die Zufriedenheit und damit die Nutzung der Lösung?

Es ist kein Zufall, dass die drei Bereiche im Einklang mit der Definition von Ge-

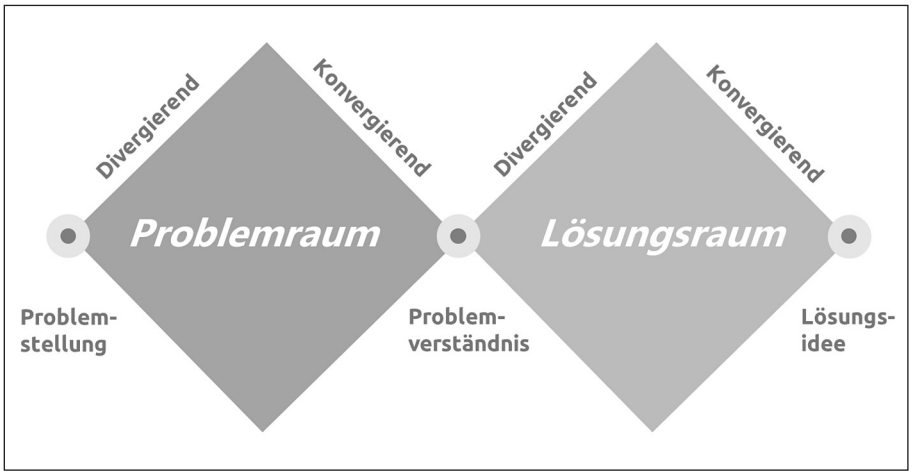

Abb. 2: Schematische Darstellung eines Vorgehens durch den Problem- und Lösungsraum

brauchstauglichkeit (*Usability*) stehen und damit die Wichtigkeit von Nutzerzentrierung bereits beginnend mit strategischen Überlegungen unterstreichen.

Discover

Der Baustein Discover verfolgt in erster Linie das Ziel, passende Lösungen zur Aufgabenstellung zu finden und Risiken zu minimieren, bevor eine Zeile Code geschrieben wird (vgl. [Cag18]). Dabei werden insbesondere Antworten auf folgende Fragen gesucht:

- Welche Lösung werden Nutzer lieben?
- Welche Lösung wird das Geschäft bestmöglich unterstützen?
- Welche Lösung steht in einem adäquaten Kosten/Nutzen-Verhältnis?

Aufbauend auf den Konzepten des Bausteins Strategize, die zu einem *Empowerment* des Teams führen, ist ein Team befähigt, im Lösungsraum nach Lösungsideen zu suchen. Doch bevor ein Team sinnvollerweise in den Lösungsraum eintauchen kann, ist es erforderlich, das Problemverständnis zu vertiefen. Ansätze wie das Double Diamond Framework [Brit] oder Design Thinking [Hpi] können hierbei sehr gut eingesetzt werden, um unter anderem den Fokus auf den Endnutzer zu legen und auch Innovation durch kreatives Denken anzuregen. Ein grundlegendes Prinzip dabei ist ein sowohl divergierendes als auch konvergierendes Vorgehen im Problem- und Lösungsraum (**siehe Abbildung 2**). Divergierend bedeutet, dass das Team in die Breite denkt und verschiedene Perspektiven einnimmt. Konvergierend bedeutet, dass das Team Erkenntnisse wieder zusammenführt und konkretisiert.

Eine konkrete Möglichkeit zur Anwendung wäre ein Google Design Sprint (vgl. [Kna16]), ein 5-tägiges Workshop-Format, welches auf Design Thinking basiert. Unter dem Begriff Problemraum ist die Sicht auf die zu betrachtende Domäne des Vorhabens zu verstehen mit den unter anderem zugehörigen Herausforderungen, Nutzerbedürfnissen oder geschäftlichen Anforderungen. Da in der modernen Softwareentwicklung die Nutzerzentrierung eine wichtige Säule ist, muss ein besonderer Schwerpunkt bei der Betrachtung des Problemraums auf dem Aufbau von Empathie für die betroffenen Nutzergruppen liegen. Daher finden hier Techniken aus dem User Experience Design Anwendung mit dem Schwerpunkt User Research. Ein zunächst divergierendes Vorgehen im Problemraum ist wichtig, um ein umfassendes Verständnis zu erlangen. Ein darauffolgendes konvergierendes Vorgehen hilft, die Problemstellung genauer zu konkretisieren und zu definieren.
Ein divergierendes Vorgehen im Lösungsraum öffnet den Blick für verschiedene Lösungsideen und erleichtert vor allem die Einbringung von Innovation. Vielversprechende Lösungsideen sollten unter Einbeziehung von Nutzern validiert werden. Hier empfiehlt es sich, leichtgewichtige Prototypen zu erstellen und diese im Rahmen von Nutzertests zu prüfen. Gerade wenn die Erfolgsmetriken im Sinne der Definition von *Outcomes* als Verhaltensänderungen definiert wurden, können diese hier schon validiert werden. Im Rahmen des konvergierenden Schritts wird die Entscheidung herbeigeführt, mit welcher Lösungsidee(n) weitergearbeitet wird.
Insgesamt ist es nicht empfehlenswert, sich linear durch den Problem- und Lösungsraum zu bewegen, sondern durchaus mit Iterationen zu planen und auch mal bei Bedarf zurückzuspringen. Oftmals entsteht bei der Lösungssuche ein besseres Verständnis des Problemraums oder bei der Validierung einer Lösung kommen ganz neue Lösungsideen zum Vorschein. Es ist ein Lernprozess!
Am Ende der bisherigen Betrachtung entsteht als Ergebnis eine validierte Lösungsidee repräsentiert durch einen Prototypen. In unserem methodischen Rahmen geht die Idee einer Product Discovery jedoch noch einen entscheidenden Schritt weiter, denn die Discovery soll die Umsetzung der Lösung angemessen vorbereiten.
Zunächst wird die Lösungsidee in einem Backlog abgebildet. Hierbei werden weitere funktionale, aber auch nichtfunktionale Aspekte der Lösung ausgearbeitet und in Form von Backlog Items dargestellt. Allerdings sollte dies nach Maß erfolgen, um einerseits Flexibilität zu wahren und andererseits so wenig Annahmen wie möglich zu treffen. Daneben sind erste Architekturarbeiten mit einem fachlichen Schwerpunkt durchzuführen. Hilfreich sind an dieser Stelle Konzepte aus dem Strategic Domain-Driven Design, um zum Beispiel eine funktionale Dekomposition mittels *Bounded Contexts* zu erarbeiten und Beziehungen zwischen Kontexten und deren Teams zu definieren.
Gerade bei größeren Vorhaben kann der Aufbau einer Teamorganisation im Einklang mit der Architektur ein Erfolgsfaktor sein. Weitere Ideen liefern die Konzepte von Team Topologies (vgl. [Ske19]). Da die funktionale Ausdetaillierung der Lösung nach Maß erfolgt, sind ebenso auch die Architekturarbeiten angemessen zu erfolgen, um insgesamt ein *Big Design Up Front* und verfrühte Entscheidungen zu vermeiden (*Just in Time – Just enough*

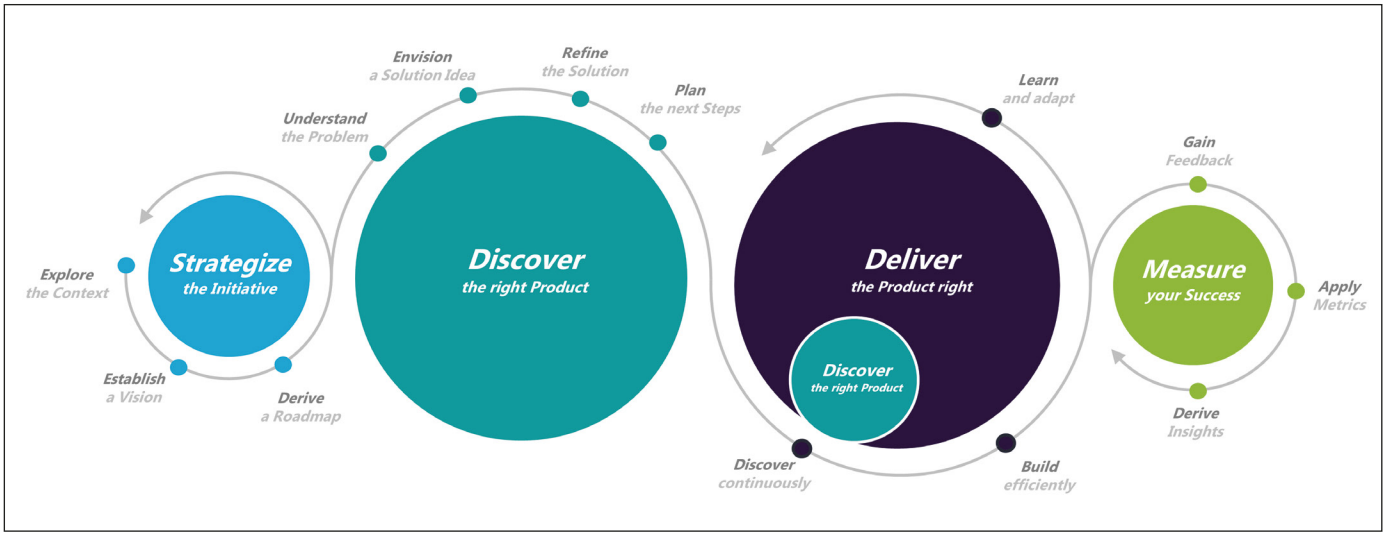

Abb. 3: Darstellung der Aktivitäten im logischen Zusammenhang (Alle Rechte liegen bei Capgemini. © 2022)

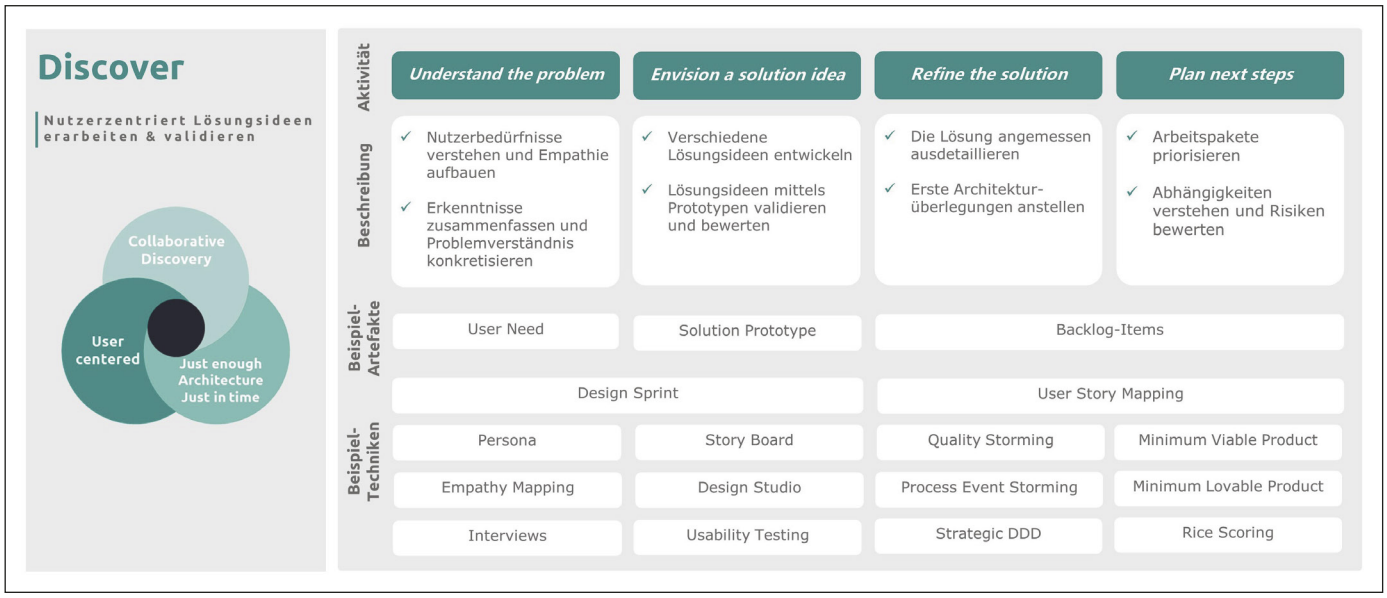

Abb. 4: Exemplarische Darstellung des Baukastenprinzips für den Baustein Discover

Architecture). Die Architektur sollte stets an der Erfüllung der definierten Qualitätsmerkmale gemessen werden. Dies kann zum Beispiel durch den Einsatz der Architecture Tradeoff Analysis Method (vgl. [Carn]) während der Product Discovery oder auch zu späteren Zeitpunkten erreicht werden.

Als Letztes ist es ratsam, eine Priorisierung durchzuführen, um die nächsten Schritte zu planen. Ist ein schneller Time-to-Market wichtig? Werden bestimmte Benutzergruppen oder Prozesse priorisiert? Konzepte wie das *Minimum Viable Product* oder *Minimum Loveable Product* helfen dabei, einen Schwerpunkt zu finden.

Abschließend ist noch zu erwähnen, dass Product Discovery eine Teamaufgabe ist. Das Team sollte die Tätigkeiten innerhalb einer Product Discovery gemeinsam und kollaborativ durchführen. So werden verschiedene Perspektiven und Kompetenzen gewinnbringend eingebracht und Silos innerhalb der Teams durchbrochen. Typische Übergabeprozesse, wie zum Beispiel die Übergabe von Anforderungen, werden verschlankt und die Notwendigkeit, umfangreiche Spezifikationen zu erstellen, wird auf ein notwendiges Minimum reduziert.

Deliver

Der Baustein Deliver bietet Orientierungshilfe im Wesentlichen für die Aktivitäten zur Umsetzung von Lösungsideen. Sofern in einer vorangegangen Product Discovery Lösungsideen bereits validiert worden sind, besteht nun schon ein gewisser Grad an Zuversicht, auf dem richtigen Weg zu sein, der jedoch weiter verfestigt werden muss. Daher empfiehlt sich ein iteratives und inkrementelles Vorgehen mit dem Ziel, Produktinkremente schnellstmöglich produktiv zu setzen. Dies setzt unter anderem einen hohen Grad an Automatisierung und die durchgängige Sicherstellung der Qualität voraus. Alternativ zum iterativen Vorgehen kann aber auch ein kontinuierlicher Produktentwicklungsprozess mittels zum Beispiel Kanban angewendet werden. Unabhängig vom konkreten Vorgehen gilt folgende einfache Regel: Je schneller ein Produktinkrement in der Nutzung von Endnutzern ist, desto schneller kann das Team den tatsächlichen Mehrwert der Lösung bewerten und Verbesserungspotenzial aufgreifen.

Neben der Fokussierung auf die Lieferung von produktionsreifen Produktinkrementen ist es wichtig, einen kontinuierlichen Lernprozess zu etablieren. Dabei geht es nicht nur darum, gewonnene Erkenntnisse bei der Weiterentwicklung zu berücksichtigen, sondern auch ein angemessenes Gleichgewicht zwischen Delivery- und kontinuierlichen Discovery-Aktivitäten herzustellen. Denn eine kontinuierliche Product Discovery hilft unter anderem dabei, die Nutzer weiterhin ins Zentrum zu stellen, Probleme und Nutzerbedürfnisse zu verstehen oder auch Ideen leichtgewichtig durch Prototypen zu validieren. Damit ergänzt eine kontinuierliche Product Discovery ein typisches Backlog Refinement um wertvolle Aspekte. Der Pool von Techniken gerade aus dem Bereich User Experience Design ist groß, sodass unabhängig von der Größe der Themen immer passende Techniken oder Workshop-Formate zur Erarbeitung gefunden werden können.

Mit der agilen Entwicklung einer Produktlösung wächst ebenso die Architektur der Lösung mit. Während die Architekturarbeit innerhalb der Product Discovery nur eher wenige übergreifende Fragestellungen adressiert, geht es nun darum, die Architektur passend zum Rhythmus der Feature-Umsetzung mitwachsen zu lassen (*Just in Time – Just enough Architecture*). Dabei sollten auch hier verfrühte Entscheidungen vermieden werden. Einige Architekturmuster, wie Clean Architecture oder Hexagonale Architektur, unterstützen dabei, Architekturentscheidungen hinauszuzögern oder Änderungen in der Architektur mit geringerem Aufwand durchzuführen. Grundsätzlich kann sich die Architektur einer Lösung auch sehr grundlegend verändern. Der bekannteste Ansatz ist vielleicht die *Monolith First*-Strategie, bei der zunächst eine monolithische Architektur gewählt wird und später mit mehr Wissen eine Zerlegung in eine verteilte Systemarchitektur erfolgt.

Während die Muster des Strategic Domain-Driven Design vor allem in der Product Discovery zu empfehlen sind, bieten sich die Tactical Domain-Driven Design-Muster für die Detaillierung der Architektur und des Softwaredesigns während der Product Delivery an.

Measure

Mit dem Baustein Measure wird die Brücke zurück zum Baustein Strategize geschlagen, indem der Fokus nun auf der Erfolgskontrolle für die gesteckten Ziele liegt. Während die Validierung von Lösungsideen auf Basis von Prototypen innerhalb der Product Discovery erste

Erkenntnisse geliefert hat, geht es nun innerhalb des Bausteins Measure darum, eine Erfolgskontrolle auf Basis eines fertigen Produktinkrements durchzuführen.
Um weiterhin die Nutzerzentrierung als Kernprinzip zu verankern, ist im Baustein Measure das kontinuierliche Einholen von Nutzerfeedback über den ganzen Produktlebenszyklus hinweg zu etablieren. Für ein umfassendes Bild aus Nutzersicht empfiehlt es sich, sowohl qualitative als auch quantitative Erhebungstechniken zu berücksichtigen. Auch hier dienen Techniken aus dem User Experience Design als wichtige Basis.
Neben dem Nutzerfeedback ist natürlich die entscheidende Frage, ob die geschaffene Produktlösung auch den erhofften Mehrwert generiert. Dazu sind die definierten Erfolgsmetriken auszuwerten. Dabei ist durchaus zu unterscheiden, ob die Auswertung auf einer Testumgebung erfolgt oder in einer Produktivumgebung. Es ist naheliegend, dass in den meisten Fällen die Aussagekraft am stärksten ist, wenn eine Auswertung in der Produktivumgebung erfolgt. Unter Umständen sind auch andere Metriken oder Indikatoren nötig, je nachdem, auf welcher Umgebung geprüft wird.
Beispiel: Eine Metrik, welche die Nutzung von Features misst, lässt sich kaum aussagekräftig auf einer Testumgebung bewerten. Somit müsste das Team auf andere Metriken ausweichen, die bei Anwendung auf einer Testumgebung oder auch bei Nutzung von Prototypen eine Indikation abgeben, ob die gewünschte Feature-Nutzung erreicht wird. Für das Beispiel könnte die Messung der Nutzerzufriedenheit bei der Nutzung von Features darauf hindeuten, dass ein Feature auch tatsächlich genutzt werden wird. Wenn das Team auf andere Metriken während des Entwicklungsprozesses ausweicht, muss darauf geachtet werden, dass auch eine Korrelation der Zusammenhänge besteht.
Kontinuierliches Nutzerfeedback und die Evaluierung von Metriken liefern potenziell eine Fülle von Informationen. Daher ist es als weiterer wichtiger Schritt nötig, aus all den Informationen die richtigen Schlüsse zu ziehen, um wertvolle Einsichten zu gewinnen, die in den kontinuierlichen Lernprozess einfließen und zur Verbesserung der Lösung führen. Bei der Analyse hilft es, unterschiedliche Perspektiven einzunehmen und verschiedene Daten miteinander zu verknüpfen. Zum Beispiel könnten Erkenntnisse aus qualitativen Nutzerfeedbacks im Zusammenhang stehen mit der Auswertung der Erfolgsmetriken. Auch andere Informationsquellen wie Tickets aus dem Support können interessante Einblicke geben.
Die gewonnenen Erkenntnisse können von unterschiedlicher Tragweite sein. Kleinere Erkenntnisse können möglicherweise für nächste Iterationen direkt eingeplant werden. Größere Erkenntnisse könnten sogar von strategischerer Bedeutung sein und müssten in Rahmen einer Product Discovery-Phase betrachtet werden.

Ausbau des methodischen Rahmens hin zu einem praktischen Baukasten

Die vorgestellten Bausteine und die zugehörigen Schlüsselkonzepte bilden die Grundlage des Baukastens. Für einen praktischen Baukasten sind aber noch weitere Elemente zu ergänzen. Zunächst

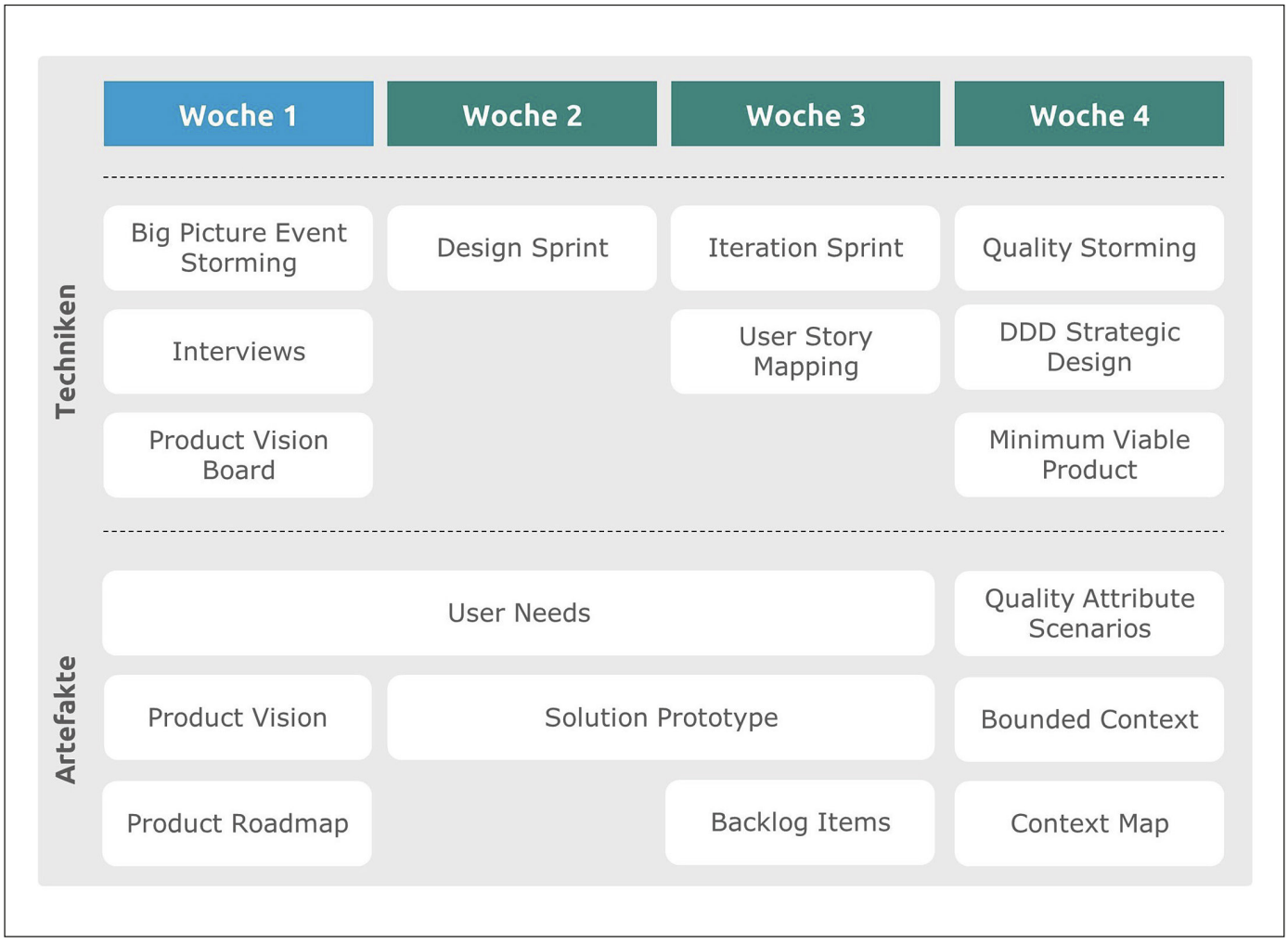

Abb. 5: Vorlage für einen Sprint 0

Literatur & Links

[Brit] British Design Council, Framework for Innovation: Design Council's evolved Double Diamond, siehe: **https://www.designcouncil.org.uk/our-work/skills-learning/tools-frameworks/framework-for-innovation-design-councils-evolved-double-diamond/**

[Cag18] M. Cagan, INSPIRED: How to Create Tech Products Customers Love, Wiley/Wiley & Sons, 2017

[Carn] Carnegie Mellon Software Engineering Institute, Architecture Tradeoff Analysis Method Collection, siehe: **https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=513908**

[Hpi] HPI Academy, Was ist Design Thinking?, siehe: **https://hpi-academy.de/design-thinking/was-ist-design-thinking/**

[Kna16] J. Knapp, J. Zeratsky, B. Kowitz, Sprint, Bantam Press, 2016

[Sei20] J. Seiden, Outcomes Over output: Why customer behavior is the key metric for business Success, Sense & Respond Press, 2020

[Ske19] M. Skelton, M. Pais, Team Topologies: Organizing Business and Technology Teams for Fast Flow, IT Revolution Press, 2019

werden die Bausteine um Aktivitäten erweitert, die konkret beschreiben, welche Aufgaben ein Team innerhalb eines Bausteins zur Umsetzung der Schlüsselkonzepte erledigen muss. In **Abbildung 3** sind die Aktivitäten zu den Bausteinen aufgeführt und in einer logischen Reihenfolge angeordnet. Die Abbildung sollte nicht als linearer Prozess verstanden werden, sondern stellt in erster Linie Zusammenhänge dar.

Als Nächstes wird die Frage beantwortet, wie ein Team die genannten Aktivitäten ausüben kann. Dies ist nun der Schritt, bei dem verschiedenste Techniken der modernen Softwareentwicklung verortet werden. Jeder Aktivität werden mehrere passende Techniken zugeordnet, die konkret zur Ausführung der Aktivität herangezogen werden können. Außerdem werden zu den Aktivitäten noch Artefakte beschrieben, die das Ergebnis einer Aktivität dokumentieren. Zur Erläuterung soll als einfaches Beispiel die Aktivität „Establish a Vision" dienen. Es gibt verschiedene Techniken, die dabei helfen, eine Produktvision zu erarbeiten, wie zum Beispiel das Product Vision Board oder Postcard from the future. Je nach konkretem Kontext ist die ein oder andere Technik besser geeignet. Für den Baukasten werden alle passenden Techniken aufgenommen. Die Produktvision selbst ist ein Artefakt, welches bei der Ausübung der Aktivität erarbeitet wird.

Der Baukasten entsteht nun also dadurch, dass zu allen Aktivitäten sowohl Techniken als auch Artefakte zugeordnet werden. **Abbildung 4** stellt exemplarisch einen Überblick für den Baustein Discover dar. Einige Techniken lassen sich in mehreren Aktivitäten anwenden, genauso können Artefakte über mehrere Aktivitäten hinweg erarbeitet werden.

Durch die Struktur Baustein/Aktivität/Technik/Artefakt entsteht ein modularer Baukasten, aus dem sich beliebig bedient werden kann. Ein Team kann nun individuell entscheiden, welche Aktivitäten oder sogar Bausteine von Relevanz sind. Bei dieser Entscheidung reflektieren Teams, was für sie wichtig ist, sodass der methodische Rahmen als Gedankenstütze dient und zu wichtigen Diskussionen anregt. Im Weiteren können Teams entscheiden, welche Techniken sie im konkreten Fall einsetzen wollen und welche Artefakte erzeugt werden sollen. Für eine bessere Entscheidungsfindung empfiehlt es sich, zu den genannten Techniken und Artefakten weitere Informationen im Baukasten zu ergänzen. Dies sind unter anderem:

- Zweck & Nutzen,
- Hilfestellungen zur Anwendung/Erstellung,
- Best Practices,
- Beispiele & Vorlagen sowie
- Referenzen zu weiteren Materialien.

Der Baukasten hilft auch bei der Fragestellung, welche Techniken sich gut ergänzen und in welcher logischen Reihenfolge sie ausgeführt werden sollten. Auf dieser Basis lassen sich weitere Rezepte entwickeln, wie zum Beispiel eine Vorlage für einen „Sprint 0", in dem für die ersten Wochen einer Initiative eine konkrete Auswahl von geeigneten Techniken und Artefakten aus dem Baukasten festgelegt wird. **Abbildung 5** zeigt eine Vorlage, die für eine Produktinitiative von mittlerer Komplexität geeignet wäre. Teams können diese Blaupause als Ausgangsbasis nutzen und passend für die konkrete Aufgabenstellung feinjustieren.

Die Zuordnung von Techniken und Artefakte über alle vier Bausteine hinweg komplettiert A·D·A·M als methodischen Rahmen. Für A·D·A·M erfolgte die Auswahl von Techniken und Artefakten durch Capgemini-Experten insbesondere aus den Bereichen User Experience Design, Business Analysis und Architektur. Die Adaption und Nutzung des Baukastens empfiehlt sich vor allem für Unternehmen, Unternehmensbereiche oder größere Programme mit mehreren Produktteams, um einerseits ein durchgängiges Mindset zu etablieren und andererseits die Effizienz und Effektivität der Teams durch eine umfangreiche praktische Unterstützung zu erhöhen. Zuletzt ist noch zu erwähnen, dass der Baukasten kein starres Konstrukt ist, sondern basierend auf gesammelten Erfahrungen und Erkenntnissen kontinuierlich adaptiert wird.

Fazit

Die steigende Vielfalt von Techniken und Praktiken in der Softwareentwicklung macht es Teams nicht einfach, einen Überblick zu bewahren und passende Techniken für konkrete Aufgabenstellungen in Entwicklungsvorhaben auszuwählen. Der vorgestellte methodische Rahmen liefert durch seinen modularen Aufbau konkrete Empfehlungen und Hilfestellungen über den gesamten Lebenszyklus eines Produkts hinweg. Insbesondere vermitteln die Schlüsselkonzepte der Bausteine das nötige Mindset für eine moderne und vor allem zielgerichtete Softwareentwicklung! ||

Der Autor

Sebastian Schnelker
(sebastian.schnelker@capgemini.com)
arbeitet seit über zehn Jahren als Architekt bei Capgemini mit dem Schwerpunkt auf geschäftskritische Anwendungen. Daneben bewegen ihn vor allem jene Themen, die unser Denken und Handeln in der Softwareentwicklung verändern.