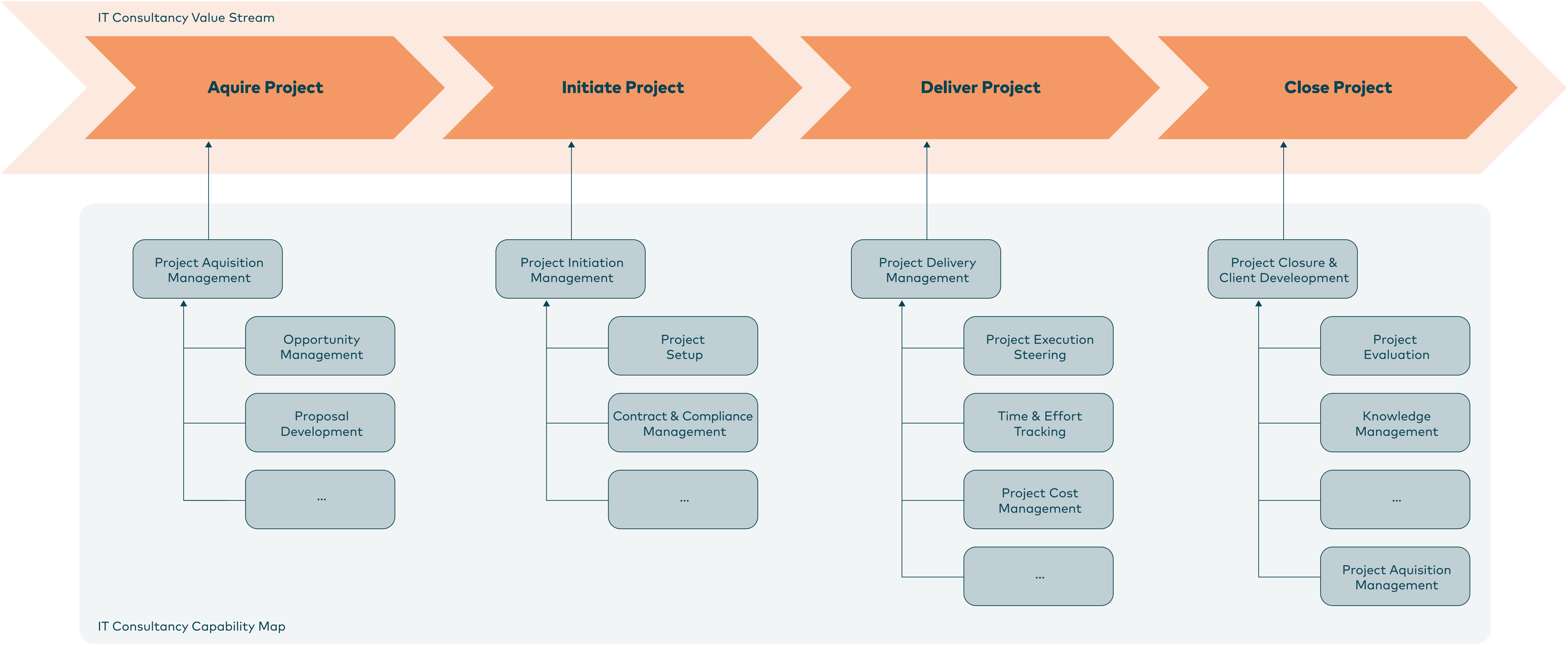


Abbildung „Capability Map" zeigt (gekürzt), wie eine Capability Map konkret für eine IT-Beratung aussehen könnte (auf Basis von ArchiMate). Entlang des definierten Value Streams[17] von „Acquire Project" bis „Close Project" werden die benötigten Capabilities hierarchisch angeordnet. So lässt sich schnell erkennen, welche Fähigkeiten für den gesamten Wertschöpfungspozess relevant sind.

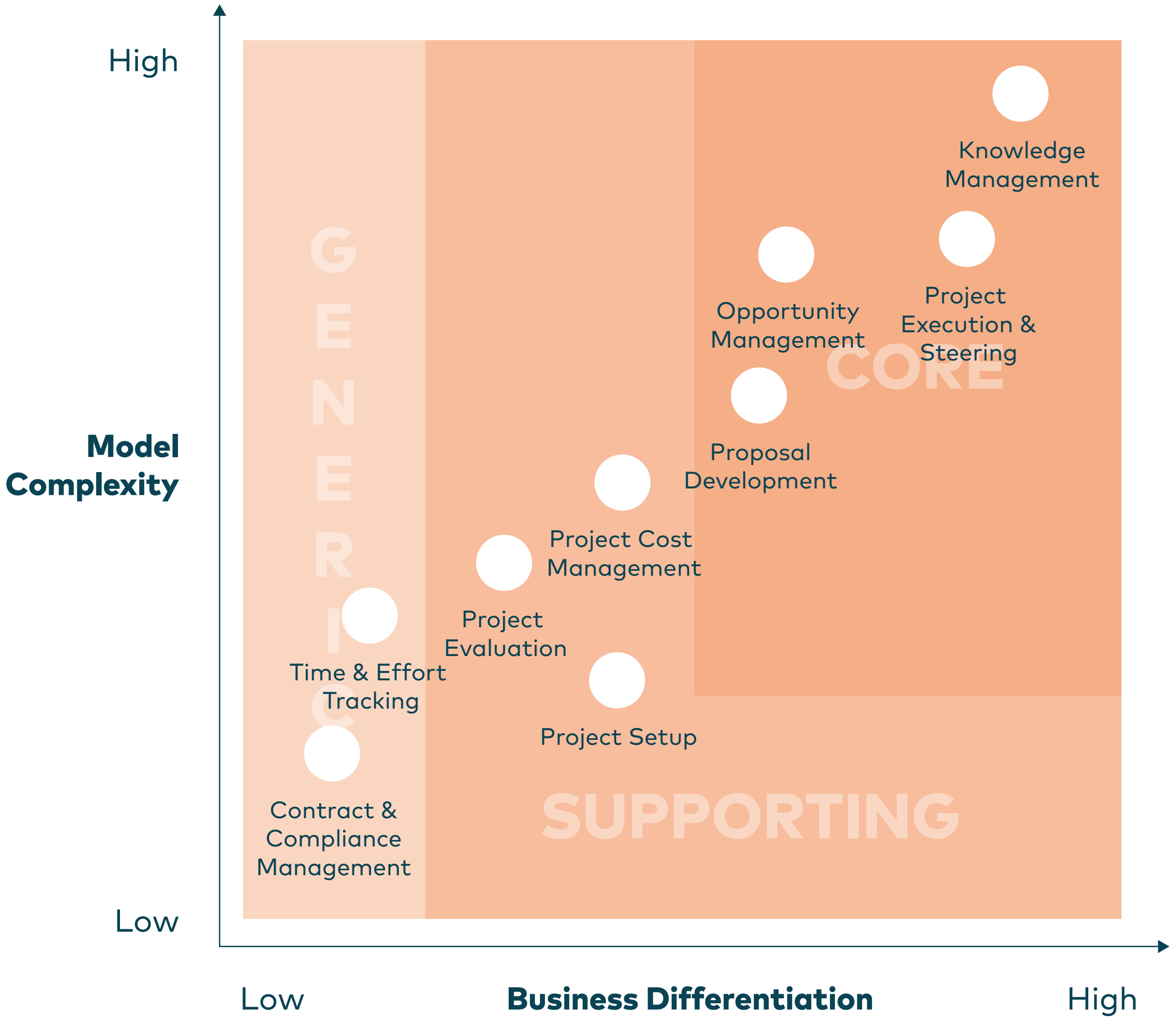
Die Bewertung dieser Capabilities erfolgt mithilfe bewährter Verfahren wie Wardley Maps[23], den DDD „Core-Domain-Charts"[24] oder strategischen Workshops. Wichtig ist dabei, den Reifegrad, die strategische Bedeutung und die gewünschte Marktabgrenzung jeder Capability zu reflektieren.

So lassen sich drei Gruppen unterscheiden:

Core Capabilities: Bereiche, in denen wir besser sein müssen als der Markt – hier lohnt sich Eigenentwicklung.

Supporting Capabilities: Unterstützende Funktionen, die wichtig, aber nicht differenzierend sind – hier kann Standardsoftware sinnvoll sein.

Commodity Capabilities: Austauschbare Bereiche, bei denen Effizienz im Vordergrund steht – hier ist Standardsoftware der Standardweg.



In unserem Beispiel der IT-Beratung ist die Capability „Time & Effort Tracking" im Bereich *Generic* eingeordnet, während „Project Execution & Steering" als *Core* gilt, da sie direkt die Wertschöpfung im Kundenprojekt beeinflusst. „Time & Effort Tracking" hingegen folgt weitgehend standardisierten Prozessen und bietet wenig Differenzierungspotenzial, sodass Standardsoftware hier sinnvoll ist.

Diese Analyse mündet in einer Make-or-Buy-Strategie, die für jede Capability festlegt, ob sie intern entwickelt oder durch Standardlösungen abgedeckt wird.

Eine eingefärbte Capability Map (z. B. Blau für Eigenentwicklung, Grün für Standardsoftware) schafft Transparenz und dient als zentrales Steuerungsinstrument für die weitere Planung.

Damit entsteht eine IT-Landschaft, in der Standardsoftware gezielt dort eingesetzt wird, wo sie Freiräume schafft – und Eigenentwicklung dort stattfindet, wo sie den größten Mehrwert liefert. Dies ist der erste Schritt, um Standardsoftware nicht zufällig, sondern bewusst und souverän einzusetzen.

Architektonische Leitplanken für mehr Freiheit

Die Entscheidung, wo Standardsoftware eingesetzt wird, ist nur der erste Teil. Genauso wichtig ist, wie diese Standardsoftware eingebettet wird. Ohne klare Architekturvorgaben führt selbst die klügste Make-or-Buy-Strategie schnell zu einem unübersichtlichen, schwer zu ändernden Flickenteppich.

Deshalb braucht es eine Makroarchitektur[15]: übergreifende Leitlinien, die für alle Systeme gelten – unabhängig davon, ob es sich um Eigenentwicklungen oder Standardprodukte handelt. Diese Architektur beschränkt sich auf eine überschaubare Anzahl zentraler Themengebiete, sorgt aber dafür, dass es an den entscheidenden Schnittstellen gemeinsame Standards gibt. So entsteht Homogenität innerhalb eines heterogenen System-of-Systems: Einzelne Systeme können unabhängig voneinander entwickelt oder ersetzt werden, ohne die Gesamtlandschaft zu destabilisieren.

Besonders beim Einsatz von Standardsoftware ist dies essenziell. Integrationsrichtlinien müssen gewährleisten, dass die Kopplung zwischen Systemen möglichst lose bleibt, um Anbieter bei Bedarf ohne großen Aufwand wechseln zu können. In der Praxis haben sich asynchrone, eventgetriebene Architekturen[25] als wirksames Muster bewährt, da sie Systeme entkoppeln und eine flexible Reaktion auf Veränderungen ermöglichen.

Entsprechend sollten Unternehmen bei der Auswahl von Standardsoftware darauf achten, dass diese über geeignete APIs verfügt und im besten Fall bereits eigenständig Events veröffentlicht – etwa über Webhooks.

Ein häufiger Fehler besteht darin, proprietäre Datenstrukturen direkt in andere Systeme weiterzuleiten. Dadurch entstehen schwer auflösbare Abhängigkeiten, die spätere Anbieterwechsel extrem teuer und riskant machen. Stattdessen sollten eingehende Daten über Wrapper in unternehmenseigene Formate transformiert werden. Zwar erhöht dies zunächst den Integrationsaufwand, langfristig reduziert es jedoch die Komplexität und erleichtert die Wiederverwendbarkeit.

Schließlich ist auch bei der Auswahl der zentralen Integrationskomponenten – beispielsweise des Event Brokers – darauf zu achten, dass sie auf offenen und standardisierten Formaten basieren. Proprietäre Protokolle an dieser kritischen Stelle würden das gesamte System erneut in eine Abhängigkeit bringen, die nur schwer auflösbar ist. Ein offener, idealerweise Open-Source-basierter Ansatz schützt die langfristige Unabhängigkeit.