Software Lifecycle Management als Grenzgänger

Software Lifecycle Management bedeutet zunächst einmal die ganzheitliche Verwaltung von Software über den Lifecycle «Plan – Build – Run». Bei genauerem Hinsehen ist es jedoch eine Grenzüberschreitung zwischen Bereitstellung und Betrieb. Diese Grenze gilt es zu überwinden – dies ist nicht ganz einfach, denn die gängigen Software-Engineering-Praktiken ignorieren die Bedürfnisse des geregelten Betriebs in sträflicher Weise. Daniel Liebhart

SCHWERPUNKT: SOFTWARE LIFECYCLE MANAGEMENT

Bild: Fotolia

Software Lifecycle Management (SLM) umfasst die Bereitstellung von Richtlinien, Abläufen, Methoden, Diensten und Instrumenten für den strukturierten Umgang mit Software über den ganzen Lebenszyklus hinweg. Einfacher formuliert umfasst SLM die Phasen «Plan – Build – Run» einer Software und stellt Instrumente für die ganzheitliche Verwaltung zur Verfügung. Während sich für andere IT-Assets wie Hardware oder Daten gute Lifecycle-Modelle durchgesetzt haben oder zumindest immer mehr Verbreitung finden, sind solche Modelle für Software alles andere als einheitlich. Dies hat verschiedene Gründe.

Daniel Liebhart ist Dozent für Informatik an der ZHAW und Solution Manager der Trivadis AG.

Software ist nicht gleich Software. Es gibt einen grossen Unterschied zwischen den betrieblichen Informationssystemen, die zentrale Unterstützungsfunktionen für die tägliche Arbeit wahrnehmen, und anderen Softwaretypen wie beispielsweise Betriebssystemen oder Infrastruktursoftware (Applikationsserver oder Datenbanken). Software (zumindest betriebliche Software) ist wesentlich komplexer aufgebaut als beispielsweise Hardware oder die Datenbestände eines Unternehmens. Ausserdem kommt noch dazu, dass sich die IT-Community ganz und gar nicht einig ist, wo SLM beginnt und aufhört.

Tatsache ist jedoch, dass betriebliche Software im Schnitt zwölf bis fünfzehn Jahre lang im Einsatz ist, ob sie nun als Standardprodukt auf die betrieblichen Anforderungen angepasst oder als Individualsoftware speziell für einen bestimmten Anwendungszweck gebaut wurde. Die Bereitstellung (Plan- und Build-Phasen) erfolgt meist in ein bis drei Jahren. Die restlichen zehn bis zwölf Jahre läuft die Software im Betrieb (Run-Phase). Sieht man sich die gängigen SLM-Modelle an, so ist ein grundlegender Unterschied festzustellen: Es gibt SLM-Modelle, die auf die Bereitstellung, also auf die Entwicklung von Software, fokussiert sind und solche, die vor allem den Betrieb von Software regeln.

SLM als Entwicklungsprozess

Wird Software Lifecycle Management aus Sicht der Entwicklung von Individualsoftware oder auch aus der Sicht des Anpassens und Einführens von Standardsoftware betrachtet, so umfasst SLM Instrumente wie beispielsweise ein Version-Control-System, ein Issue-Tracking-System sowie Mechanismen für die Qualitätssicherung und das Build-Management. Aus dieser Sichtweise beschreiben SLM-Modelle die Entwicklungsphasen in Modellen, wie beispielsweise dem Wasserfall, der Spirale, der agilen Entwicklung oder anderer Vorgehensmethoden, die für das Engineering von Software eingesetzt werden. Der Fokus liegt dabei auf der plan- und kontrollierbaren Entwicklung eines defi-

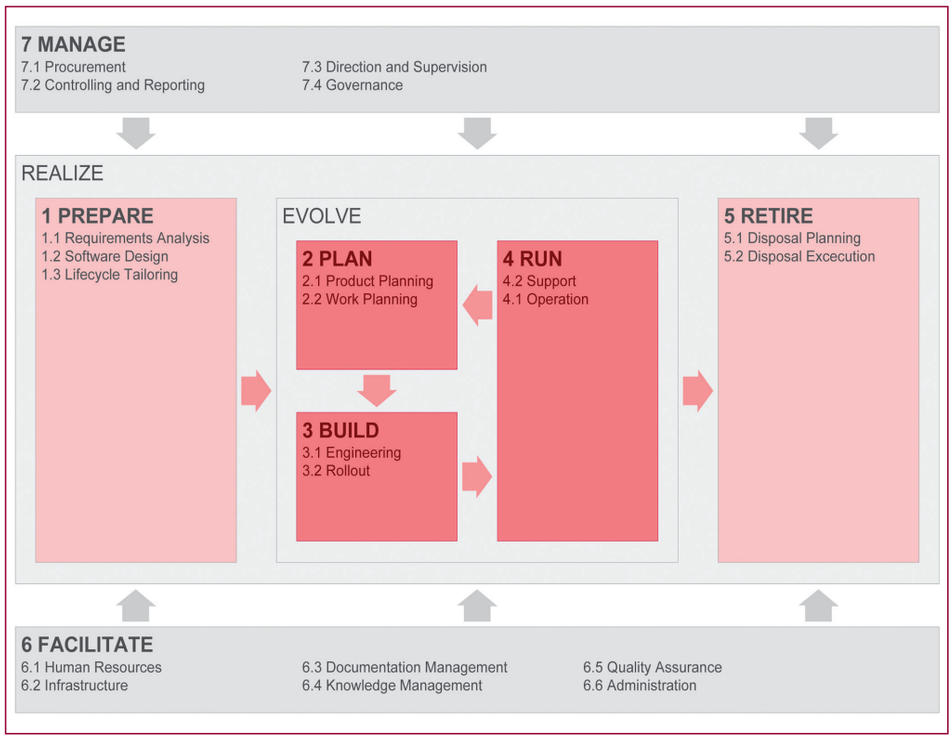
nierten Funktionsumfangs entlang der drei vorgegebenen Kenngrössen Kosten, Zeit und Qualität. Eigenschaften wie die Modularität, die Formbarkeit, die konzeptionelle Integrität und intellektuelle Kontrolle, die Herstellbarkeit oder auch Kopplung und Kohäsion stehen dabei im Vordergrund. Die Instrumente und Toolsets sind in den meisten Fällen aber ausschliesslich auf die Umsetzung betrieblicher Anforderungen im Rahmen von Projekten ausgelegt.

SLM als Betriebsprozesse

Wird Software Lifecycle Management aus Sicht des Betriebs umgesetzt, so ist zumindest das zugrunde liegende Basismodell gut etabliert: es sind die Betriebsprozesse nach ITIL V3/ISO 20000. Der Lebenszyklus von ITIL bezieht sich dabei auf den Service, den eine Betriebsorganisation gegenüber ihrem Kunden (dem Business) erbringen soll. ITIL V3 Lifecycle Service Strategy, Service Design, Service Transition, Service Operation und Continual Service Improvement sind vollständig auf den geregelten Betrieb ausgerichtet. Der Service wird dabei als betriebsunterstützende Funktionalität bestehend aus Hardware, Software, Netzwerk und Betriebsorganisation betrachtet. Im Fokus steht die plan- und kontrollierbare Nutzung von IT-Services entlang der Parameter Funktionalität, Kapazität und Verfügbarkeit, die vertraglich in einem SLA (Service Level Agreement) definiert wurden. Dabei stehen Eigenschaften wie Reaktionszeit, Sicherheit, Stabilität, Verwendbarkeit oder Messgrössen wie RTO (Recovery Time Objective: maximale Wiederherstellungszeit bei Systemausfall), RPO (Recovery Point Objective: maximal erlaubter Datenverlust) oder DPW (Data Protection Window: maximale Offline-Zeit) im Vordergrund.

Ganzheitlich mit ISO/IEC 12207?

Ein SLM mit Fokus auf die Entwicklung unterscheidet sich erheblich von einem SLM aus Betriebssicht. Dass eine Kombination der beiden grundlegend verschiedenen Perspektiven nicht einfach ist, zeigt der Umfang und Aufbau des SLM-Standards ISO/IEC 12207 (Software Lifecycle Process), der bereits 1995 in seiner ersten Version publiziert worden ist. Der Standard umfasst eine Vielzahl vorgegebener Prozesse und teilt diese in primäre, unterstützende und organisatorische Prozesse ein. Die Primärprozesse umfassen die Beschaffung, die Lieferung, die Entwicklung, den Betrieb und die Wartung von Software. Allerdings schliesst der Standard den Bereich der Standardsoftware aus und versteht sich lediglich als Prozessrahmen. Aus den ursprünglichen



Ein integratives SLM-Modell erweitert den normalen Plan-Build-Run-Zyklus, der auf die Entwicklung ausgerichtet ist, um Mechanismen, die für den Betrieb wichtig sind. So kann ein umfassendes Lifecycle-Modell etabliert werden. Bild: Trivadis

Standards haben sich ganze Standardfamilien entwickelt, wie beispielsweise CMM/CMMI (Capability Maturity Model) oder auch SPICE (Software Process Improvement and Capability Determination). Die konkrete Umsetzung und Abbildung des Standards in ein Software-Lifecycle-Modell mit den entsprechenden Instrumenten und Regeln hat sich bis heute nicht etabliert. Der Grund dafür ist neben der Komplexität der umfassenden Standards auch die unterschiedliche Fokussierung. Der Bau von Software unterscheidet sich grundlegend von deren Betrieb. Leider ist die systematische und gut strukturierte Erfassung der Anforderungen aus Betriebssicht nicht in den gängigen Requirements-Engineering-Praktiken vorgesehen. Es braucht ein SLM, das die Grenzen zwischen Bau und Betrieb überwindet.

Grenzen überwinden?

Die Grenze zwischen dem Bau und dem Betrieb von Software ist durch die verschiedenen Blickwinkel gegeben. Während sich also die Entwicklung auf Eigenschaften wie Modularität, Formbarkeit, Herstellbarkeit und Kopplung/Kohäsion fokussiert, sind für den Betrieb Reaktionszeit, Sicherheit, Stabilität, Verwendbarkeit und andere Messgrössen wichtig. Aus betrieblicher Sicht sind Instrumente und Schnittstellen für die Überwachung und das Tracing zur Laufzeit essenziell, die selten in einer Individualsoftware gut umgesetzt werden. Für die Entwicklung sind Abstraktionen wichtig, wie beispielsweise die universelle Verfügbarkeit von Speicher oder der Einsatz verschiedener Libraries zur Steigerung der Produktivität, die für den Betrieb höchstens eine zusätzliche Komplexität darstellen. Software sollte möglichst aus einfach verständlichen Modulen bestehen, die auch als Legacy-System noch gut verstanden werden kann.

Entwicklungsteams dokumentieren den Aufbau der Software, während der Betrieb sich lediglich für ein einfaches Installations- und Betriebshandbuch interessiert. Diese Grenze kann nur vom Engineering überwunden werden. Dies würde bedeuten, dass die heute gängigen Entwicklungsprozesse vermehrt die Anforderungen des mittel- und langfristigen IT-Betriebs mit einbeziehen müssen. Ein gutes und in der Praxis umsetzbares Software Lifecycle Management erweitert also den Plan-Build-Run-Zyklus um betriebliche Elemente, beispielsweise mit vorbereitenden Prozessen, die nicht nur das Entwicklungsprojekt, sondern den gesamten zwölf- bis fünfzehnjährigen Lebenszyklus planen oder mit Prozessen einen geplanten Rückbau einer Software vorsehen (siehe Grafik). Die Suche nach pragmatischeren Lösungen ist momentan im Gange und wird sicherlich in den nächsten Jahren interessante Ansätze zutage bringen. <