Modellierung und Enactment mit ESSENCE

Michael Striewe, Michael Goedicke
Paluno - The Ruhr Institute for Software Technology, Universität Duisburg-Essen
Gerlingstraße 16, 45127 Essen
{michael.striewe,michael.goedicke}@s3.uni-due.de

Abstract: Komplexe Softwareentwicklungsmethoden entstehen selten in einem Zug und sind danach unveränderlich, sondern werden schrittweise erstellt und kontinuierlich verbessert. Der konkrete Projektfortschritt muss durch sie messbar sein, aber gleichzeitig muss das Vorgehen projektspezifisch adaptierbar bleiben. Vor dem Hintergrund dieser Anforderungen wurde von der SEMAT-Initiative ESSENCE entwickelt, das den agilen Umgang mit Softwareentwicklungsmethoden besser ermöglichen soll als mit den bisher verfügbaren Ansätzen. Dieser Beitrag stellt die Kernkonzepte von ESSENCE vor.

1 Einleitung

Moderne Softwareentwicklung ist ohne komplexe Softwareentwicklungsmethoden kaum denkbar, da bei der Entstehung eines großen Softwareproduktes zahlreiche Schritte ausgeführt und zahlreiche Akteure koordiniert werden müssen. Die Modellierung und Dokumentation der gewählte Methode spielt dabei eine wichtige Rolle: Nur so kann sichergestellt werden, dass allen Beteiligten die notwendigen Schritte überhaupt bekannt sind, der Projektfortschritt kann gegen dokumentierte Zwischenziele gemessen werden und kritische Planabweichungen sowie Verbesserungspotenziale können schneller erkannt werden, wenn ein Vergleich zwischen Soll und Ist möglich ist. Bei komplexen Methoden wird eine solche Dokumentation aber auch selber komplex und damit schwer zu lesen und schwer wartbar; im Extremfall nur durch wenige Mitarbeiter in der Rolle eines Method Engineer. Dies verhindert gegebenenfalls eine zeitnahe und flexible Anpassung der Methoden, um auf individuelle Projektsituationen einzugehen, und schränkt damit den Nutzen von Methoden ein.

Um diesem Problem zu begegnen wurde auf Basis mehrjähriger praktischer Erfahrungen von der SEMAT-Initiative[1] ESSENCE entwickelt. ESSENCE stellt nicht nur Möglichkeiten zur Beschreibung von Softwareentwicklungsmethoden bereit, sondern auch zur Ausführung dieser Methoden, zur Adaption in laufenden Projekten und zur Beurteilung des Projektfortschritts. Philosophie, Bausteine und Sprachdesign von ESSENCE werden im Folgenden aufgezeigt und diskutiert. Dieser Artikel basiert dabei auf der jüngsten Version der ESSENCE-Spezifikation, die unter der Dokument-Nummer *ad/2012-11-01* bei der OMG zur Annahme als Standard eingereicht wurde.

[1] `http://www.semat.org/`

2 Das Konzept von ESSENCE

Um dem Problem der Komplexität aktueller Modellierungsansätze zu begegnen, trennt ESSENCE zunächst zwischen zwei Metamodellen, die bisher üblicherweise gemeinsam betrachtet werden: Während Ansätze wie SPEM [SPE08] und ISO 24744 [ISO07] ein sehr umfangreiches Metamodell anbieten, das sowohl die technische Infrastruktur der Sprache als auch die wichtigsten Modellierungskonzepte abbildet, trennt ESSENCE zwischen einem übersichtlicheren Metamodell der technischen Infrastruktur in Form syntaktischer Element (die „ESSENCE-Sprache“), und einer separat beschriebenen Menge der wichtigsten Konzepte und Begriffe als Metamodell für Methoden (der „ESSENCE-Kernel“) [KMSG+12]. Der Kernel und die daraus gebauten Methoden werden dabei in der Syntax der Sprache ausgedrückt, sind damit also selber bereits Instanz eines Metamodells, auch wenn ihre Elemente zur Ausführung des Vorgehensmodells selber noch einmal instanziiert werden. Der Modellierungsansatz greift damit die Dualität von Klasse und Objekt auf [HSGP05], bleibt aber völlig im Rahmen des Ansatzes von MOF zur Metamodellierung [ESMB12]. Ziel dieses Designs ist es, sowohl die Erstellung als auch die Nutzung von Methoden zu erleichtern. Wie diese Trennung genau vollzogen wird, wird in den beiden folgenden Kapiteln erläutert.

Bei der Erstellung des Kernels ist es nicht das Ziel, eine möglichst umfangreiche Terminologie oder Ontologie der Softwareentwicklung zu erstellen, zumal es dafür bereits ältere Ansätze gibt [SWE]. Vielmehr folgt ESSENCE der Philosophie, dass Perfektion dann erreicht ist, wenn es nichts mehr wegzunehmen gibt. Je kleiner und universeller der Kernel ist, umso eher ist er tatsächlich die Essenz dessen, was allen Entwicklungsmethoden zugrunde liegt, während die Ausgestaltung dieser Grundlagen in der Hand der einzelnen Methoden liegt. Analog dazu wird in der Beschreibung einzelner Methoden davon ausgegangen, dass auch hier zunächst einmal nur die essentiellen Bestandteile dieser Methode festgelegt werden sollen, während detailliertere Ausgestaltungen als projekt- oder unternehmensspezifische Erweiterungen geschehen.

3 Die Bausteine von ESSENCE

In der natürlichen Sprache kann ein Text in Sätze zerlegt werden und diese wiederum in Worte. Jedes Wort gehört dabei einer bestimmten Wortart an und die korrekte Bildung von Sätzen unterliegt grammatikalischen Regeln. Wortarten und Regeln sind typischerweise weitgehend starr und verändern sich nur in sehr langen Zeitabstände, während eine lebendige Sprache ständig neue Worte bilden und aus bestehenden Worten immer neue Sätze formen wird. Gleichzeitig gibt es in jeder Domäne wichtige Worte, die immer wieder verwendet werden. In Analogie zu dieser (zweifellos vereinfachten) Darstellung natürlicher Sprache führt die ESSENCE-Spezifikation mehrere Bausteine ein (siehe Abbildung 1): Das Metamodell der Sprache als Sammlung der Wortarten und grammatikalischen Regeln, den „Kernel“ als Sammlung wichtiger Konzepte, „Practices“ als abgeschlossene Vorgehensbeschreibungen für eine konkrete Problemstellung, und letztendlich Methoden als Komposition mehrerer Practices auf Basis eines gemeinsamen Kernels. In der MOF-

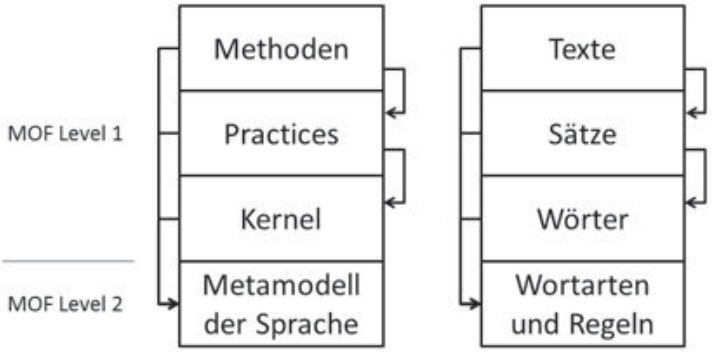

Abbildung 1: Die Bausteine von ESSENCE (links) in Analogie zu den Bausteinen natürlicher Sprache (rechts).

Hierarchie ist die Sprache auf Level 2 anzusiedeln, während sich Kernel, Practices und Methoden auf Level 1 befinden.

3.1 Das Metamodell

Abbildung 2 zeigt eine konzeptionelle Sicht auf das Metamodell der ESSENCE-Sprache und die wichtigsten darin deklarierten Typen. Bei den Elementen in der Mitte der Abbildung (*Alphas* und *Alpha States*, *Activity Spaces* und *Competencies*) handelt es sich um Elemente für die Beschreibung abstrakter, methodenunabhängiger Konzepte. Durch Verbindung mit den auf der rechten Seite aufgeführten konkreten Elementen (*Work Products* und *Activities*) können diese verschiedene Ausprägungen erhalten. Die außen dargestellten so genannten *Patterns* wiederum dienen als generisches Mittel zur Herstellung von beliebigen Bezügen zwischen abstrakten und konkreten Elementen.

Die Elemente *Alpha*, *Alpha State* und *Work Product* befassen sich dabei mit den Dingen, die Gegenstand eines Projektes sind. Beispielsweise kann das Alpha „Requirements“ die Anforderungen abstrakt repräsentieren und vom Zustand „Conceived“ bis zum Zustand „Fulfilled“ geführt werden, während die konkrete Repräsentation der Anforderungen in einem Projekt durch User Stories erfolgt, die demnach ein konkretes Work Product darstellen. Andere Projekte können sich für andere Work Products entscheiden, das Alpha bleibt jedoch dasselbe. Die Zustände eines Alphas sind dabei eines der entscheidenden Konzepte in ESSENCE: Jeder Zustand ist mit einer Anzahl von Checkpoints versehen, die jederzeit überprüft werden können. Auf diese Weise kann unabhängig von der verwendeten Methode ein Ziel formuliert und seine Erreichung überprüft werden. Im Laufe eines Projektes können sich alle Alphas unabhängig voneinander in ihren Zuständen vorwärts oder auch zurück bewegen. Die Namen der Alpha States sind ausnahmslos positiv formuliert, da es sich um gewünschte Zustände handelt, deren Erreichung für das Projekt wertvoll ist. In der Realität werden zweifellos auch ungünstige Zustände existieren, deren

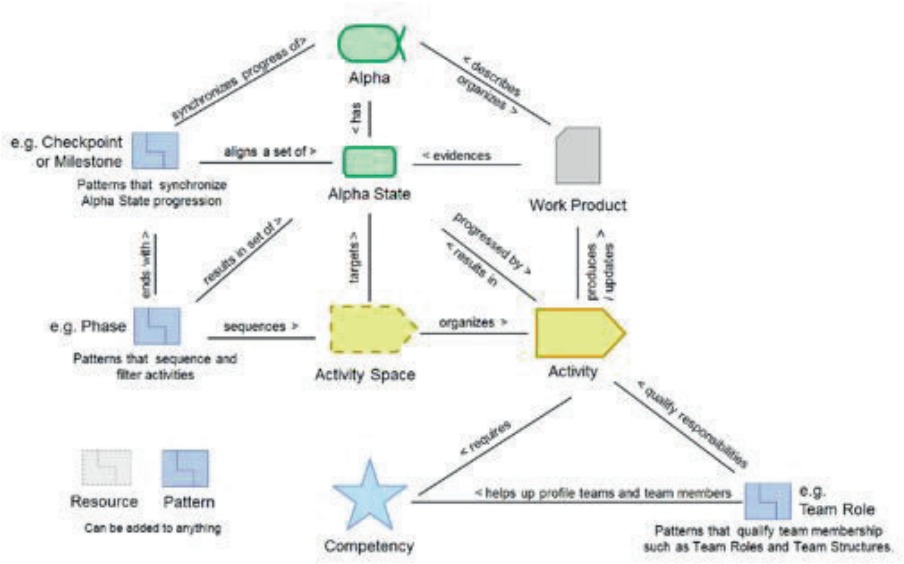

Abbildung 2: Konzeptionelle Sicht auf die zentralen Elemente des Metamodells der ESSENCE-Sprache (aus *ad/2012-11-01*). Nicht abgebildet sind Elemente für Kernel, Practices und Methoden, die im Wesentlichen Container für die abgebildeten Elemente sind.

explizite Benennung für die Erstellung von Methoden aber ohne Belang ist und die daher in ESSENCE nicht weiter betrachtet werden.

Die Elemente *Activity Space* und *Activity* befassen sich mit den Tätigkeiten, die in einem Projekt auszuführen sind. Auch hier gilt die Trennung zwischen abstrakter und konkreter Beschreibung, so dass z.B. der Activity Space „Implement the System“ abstrakt das gesamte Tätigkeitsfeld der Implementierung umfasst und mit konkreten Aktivitäten wie Codegenerierung, manueller Implementierung usw. gefüllt werden kann. Wissen, Fähigkeiten und Einstellungen, die zur erfolgreichen Durchführung der Tätigkeiten notwendig sind, werden wiederum abstrakt als *Competency* beschrieben.

Neben der direkten Zuweisung an Aktivitäten und Activity Spaces können diese insbesondere auch in *Patterns* verwendet werden, die zum Beispiel komplexe Rollen und Verantwortlichkeiten abbilden können. So kann beispielsweise modelliert werden, dass der Inhaber einer bestimmten Rolle über bestimmte Kompetenzen verfügen muss, für bestimmte Work Products verantwortlich ist, einen bestimmten Activity Space leitet und für die Erreichung eines bestimmten Alpha States verantwortlich zeichnet. Das Metamodell definiert dabei insbesondere keine grundsätzlichen Beschränkungen, wie detailliert und umfangreich derartige Beziehungen über Patterns gestaltet werden. Definiert ein Unternehmen seine Rollen also gänzlich anders, ist dies trotzdem mit demselben Mechanismus abbildbar. Genauso können beliebige andere Zusammenhänge über Patterns modelliert werden.

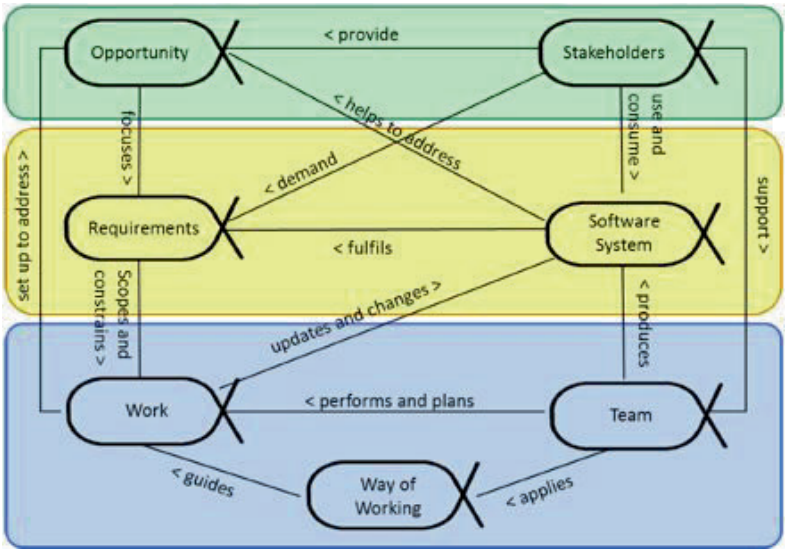

Abbildung 3: Die Alphas des Kernels und ihre Zusammenhänge (aus *ad/2012-11-01*).

3.2 Der Kernel

Der von der SEMAT-Initiative erarbeitete und zur Standardisierung in ESSENCE vorgeschlagene Kernel des Software Engineering enthält sieben Alphas, die die zentralen Aspekte des Software Engineering und ihre Zustände methodenunabhängig beschreiben: *Requirements*, *Software System*, *Team*, *Work*, *Way-of-Working*, *Stakeholder* und *Opportunity*. Diese Alphas und ihre Zusammenhänge sind in Abbildung 3 dargestellt. Die Alphas *Stakeholder* und *Opportunity* bilden dabei den unternehmerischen Bereich des Software Engineering ab, d.h. insbesondere die Schnittstelle zu Kunden, Nutzern und dem wirtschaftlichen oder sozialen Wert, der durch die Entwicklung eines Softwareproduktes geschaffen werden soll. Die Alphas *Requirements* und *Software System* bilden den technischen Aspekt der Softwareentwicklung ab, während sich *Team*, *Work* und *Way-of-Working* mit dem Management von Softwareentwicklung befassen und die Steuerung der Methode fokussieren. Dabei ist insbesondere zu berücksichtigen, dass über den Way-of-Working das modellierte Vorgehen selber mit in die Betrachtung eingeschlossen wird. Das Vorgehen, das die Anpassung des eigenen Vorgehens beschreibt, kann somit selber auch in ESSENCE ausgedrückt werden. Neben den Alphas enthält der von der SEMAT-Initiative vorgeschlagene Kernel 15 Activity Spaces zur Beschreibung zentraler Tätigkeitsbereiche, die sich ebenfalls auf die genannten drei Teilbereiche verteilen, sowie sechs Competencies.

Der Grundgedanke des Kernels bringt es mit sich, dass die genannten Elemente einigen Anwendern nicht ausreichend detailliert erscheinen oder ein für eine bestimmte Domäne wichtiges Element fehlt. Dies ist in der ESSENCE-Spezifikation berücksichtigt, indem der Kernel erweitert werden kann. Zusätzliche Alphas können entweder als eigenständi-

ge Elemente hinzugefügt werden, oder als untergeordnete Alphas ein vorhandenes Alpha weiter detaillieren. Die ESSENCE-Spezifikation schlägt mehrere solcher Erweiterungen für jeden der drei Teilbereiche vor. Beispiele dafür sind *Task* als weitere Detaillierung für *Work* oder *Component* als weitere Detaillierung für *Software System*. Dabei ist entscheidend, dass es sich bei diesen Erweiterungen nicht um eine Erweiterung des Metamodells der Sprache und damit einen Eingriff in die Sprachkonstruktion handelt, sondern dass die Kernel-Elemente und die Erweiterungen selber bereits Instanzen dieses Metamodells sind. Damit ist es grundsätzlich sogar möglich, mit dem Metamodell und der Ausführungssemantik von ESSENCE auch das Vorgehen in gänzlich anderen Domänen als der Softwareentwicklung zu modellieren, indem statt des oben beschriebenen Kernels andere Alphas identifiziert werden.

3.3 Die Practices

Als Zwischenschritt zwischen dem Kernel als Sammlung abstrakter Elemente und einer kompletten Softwareentwicklungsmethode verwendet ESSENCE das Konzept von Practices, die konkrete Handlungsempfehlungen für begrenzte Probleme des Software Engineering geben, z.B. die Verwendung von Use Cases zur Erhebung der Anforderungen. Sie ordnen den abstrakten Elementen des Kernels die jeweils angemessenen konkreten Ausprägungen zu und können bei Bedarf auch weitere Alphas einführen.

Wie detailliert eine einzelne Practice ausgestaltet wird, ist dem Autor der jeweiligen Beschreibung überlassen. Die Zuordnung eines einzelnen Work Products zu einem Alpha ist genauso eine valide minimale Practice wie die Zuweisung einer Aktivität zu einem Activity Space. Auch die Definition einiger Patterns, die Alpha States zu Meilensteinen gruppieren, ohne überhaupt Bezug auf konkrete Work Products und Aktivitäten zu nehmen, kann eine minimale Practice bilden. Lässt eine Practice Fragen unbeantwortet bzw. gibt sie für bestimmte Situationen keine Handlungsempfehlungen, kann diese Lücke entweder durch die Komposition mit einer anderen Practice oder durch spontane Adaption während der Durchführung des Projektes geschlossen werden.

3.4 Die Methoden

Auf Basis der gemeinsamen Elemente des Kernels können Practices miteinander kombiniert (oder als inkompatibel identifiziert) werden, um letztlich in der flexiblen und agilen Komposition und Konfiguration von Methoden zu münden. Dabei müssen gleichartige Elemente, die in verschiedenen Practices auftreten, miteinander verschmolzen werden, so dass eine Methode nicht einfach nur eine Sammlung von nebeneinander stehenden Practices ist.

Durch den Bezug zum Kernel kann schnell geprüft werden, ob eine passende Auswahl an Practices getroffen wurde, die in die Methode integriert werden. Zum Beispiel kann geprüft werden, ob allen Alphas mindestens ein Work Product zugewiesen wurde oder

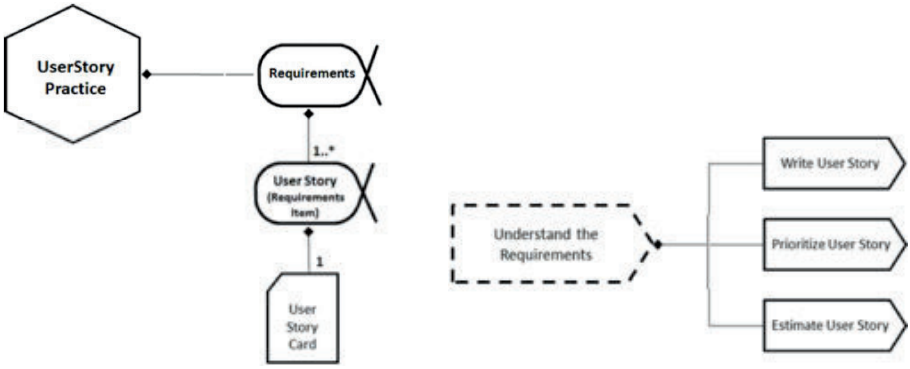

Abbildung 4: Beispiele für die Modellierung von User Stories.

jeder Activity Space mit mindestens einer Aktivität belegt wurde. Auch eine Überprüfung, ob eine Methode alle Alphas des Kernels abdeckt ist möglich, um so Hinweise darauf zu erhalten, ob eine Methode als vollständig erachtet werden kann oder weitere Practices hinzugefügt werden müssen, wenn alle Aspekte abgedeckt werden sollen.

4 Der Einsatz von ESSENCE

Eine der Zielsetzungen von ESSENCE ist es, alle „Lebensphasen“ von Softwareentwicklungsmethoden abzudecken, d.h. ihre initiale Beschreibung, ihre Komposition und Adaption und ihre Ausführung im Projekt. Für einige ausgewählte Beispiele wird dies im Folgenden diskutiert.

4.1 Modellierungsbeispiele

Im modellierungstechnischen Sinne bilden Kernel, Practices und Methoden Instanzen des Metamodells der ESSENCE-Sprache. Bereits die Beschreibung eines einzelnen Alphas, Work Products oder einer einzelnen Aktivität stellt damit ein gültiges, minimales Modell in ESSENCE dar. Auch der in Abbildung 3 bereits gezeigte Überblick über den Kernel ist (abgesehen von der farbigen Hinterlegung der drei Teilbereiche) ein gültiges Modell. Abbildung 4 zeigt einige Modellierungsbeispiele, die aus der ESSENCE-Spezifikation entnommen sind und die gemeinsam das Vorgehen bei der Anforderungserhebung mit User Stories modellieren. Es ist leicht zu erkennen, dass diese Modellierung in der hier gezeigten Form nicht vollständig ist, da den einzelnen Elemente keine detaillierteren Beschreibungen zugewiesen sind. Für diese führt ESSENCE die im Folgenden erläuterte Karten-Metapher ein.

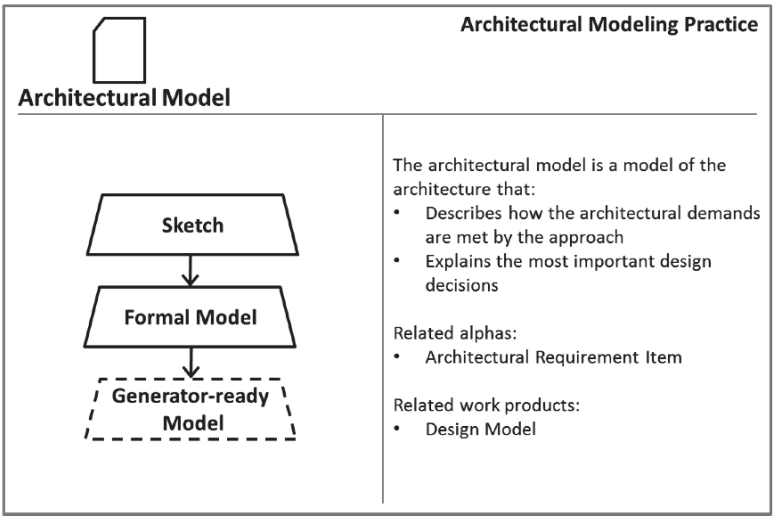

Abbildung 5: Kartenlayout für die Beschreibung eines Work Products (aus *ad/2012-11-01*).

4.2 Karten-Metapher

Ergänzend zur Modellierung von Methoden in Diagrammen führt ESSENCE zur geeigneten Präsentation von Details der Modellelemente die Metapher von Karten im Sinne von Kartei- oder Spielkarten ein. Ob diese Karten physisch vorliegen oder in einem Softwarewerkzeug virutell repräsentiert werden, ist für den Ansatz unerheblich. Die ESSENCE-Spezifikation definiert für jedes der in Abbildung 2 gezeigten Elemente des Metamodells ein Layout für eine Karte, mit der dieses Element dokumentiert werden kann. Abbildung 5 zeigt ein Beispiel für eine solche Karte, die in diesem Fall ein Work Product beschreibt. Jede Practice kann so durch einen Kartensatz repräsentiert werden. Die Komposition von Practices zu Methoden entspricht dann dem Zusammenstellen eines großen Kartensatzes aus mehreren kleinen Sätzen, wobei auch der Kernel durch einen Kartensatz repräsentiert werden kann. Die notwendige Zuordnung von Elementen zueinander kann dabei physisch und kollaborativ in einer größeren Gruppe von Beteiligten geschehen und ist damit nicht auf die Nutzung eines spezialisierten Softwarewerkzeugs beschränkt. Analog kann die Beobachtung eines laufenden Projektes ebenfalls durch die Verwendung von Karten (insbesondere für die Alpha States) geschehen, die auf einem Tableau (z.B. einem Kanban-Board [Epp11]) verschoben werden, um den aktuellen Zustand anzuzeigen. Details zum Enactment folgen weiter unten in diesem Artikel.

Auch außerhalb des eigentlichen Einsatzes im Projekt und in der Erstellung und Dokumentation von Methoden können die Karten eingesetzt werden: Sie bilden zum Beispiel in sich abgeschlossene Einheiten, die als Lerneinheiten sowohl in unternehmensspezifischen Trainee-Programmen als auch im allgemeinen universitären Curriculum Anwendung finden können. Genauso, wie aus einzelnen Karten Kernel, Practices und Methoden zusam-

mengestellt werden können, können sie auch für Lehrveranstaltungen und Kurse thematisch gruppiert werden und einen roten Faden bilden, der den Stoff in handliche Stücke zerlegt.

4.3 Enactment

Die Semantik von ESSENCE zur Ausführungszeit einer Methode basiert auf drei Aspekten: Der Feststellung des aktuellen Projektzustandes in Form von Alpha States, der Erzeugung von Handlungsempfehlungen in Form von Activities sowie der Möglichkeit der Adaption durch Komposition und Erweiterung.

Die Feststellung des aktuellen Projektzustandes erfolgt anhand der Checkpoints, die mit jedem Alpha State verknüpft sind. Unabhängig von der Verwendung in Methoden können diese Checkpoints zu jedem beliebigen Zeitpunkt ausgewertet werden, um eindeutig festzustellen, ob sich eine Instanz eines Alphas in einem bestimmten Alpha State befindet oder nicht. Dabei wird davon ausgegangen, dass jede Instanz eines Alphas seinen Zustand zu jedem beliebigen Zeitpunkt wechseln kann und an keine Reihenfolge der Zustände gebunden ist. Ebenso können zu jedem Zeitpunkt neue Instanzen eines Alphas in das Projekt eintreten oder es verlassen.

Ist der aktuelle Zustand eines Projektes bekannt, kann dieser mit dem gewünschten Zustand verglichen werden. Da die Alpha States geordnet sind, d.h. einen eindeutigen Vorgänger und Nachfolger haben, kann aus diesen Informationen abgeleitet werden, welche Zustände gegebenenfalls zusätzlich durchlaufen werden sollten, um vom aktuellen Zustand in den gewünschten zu gelangen. Aus dieser Zustandsmenge heraus kann dann wiederum eine Menge von Aktivitäten abgeleitet werden, die in der Erreichung dieser Zustände resultieren sollten. Diese bilden eine konkrete Handlungsempfehlung. Dabei ist insbesondere zu berücksichtigen, dass nicht automatisch Instanzen von Aktivitäten erzeugt werden, d.h. nicht davon ausgegangen wird, dass alle empfohlenen Aktivitäten auch tatsächlich sofort durchgeführt werden. Ebenso wenig wird aus der Beendigung einer Aktivität automatisch geschlossen, dass nun der angestrebte Alpha State auch erreicht ist. Die Rückkopplung geschieht ausschließlich über die Checkpoints der Alpha States. Dadurch deckt das Enactment in ESSENCE auch die Fälle ab, in denen eine Aktivität abgebrochen werden kann oder muss, weil der angestrebte Zustand aufgrund anderer Umstände bereits erreicht wurde oder ein Rückschritt eingetreten ist, der andere Aktivitäten wichtiger macht.

Dieser flexible Umgang mit Handlungsempfehlungen ist auch Teil des Adaptionskonzeptes von ESSENCE. Das Modell der Entwicklungsmethode darf zur Ausführungszeit jederzeit geändert werden, so dass in derselben Situation möglicherweise unterschiedliche Handlungsempfehlungen gegeben werden, wenn zwischenzeitlich eine Adaption des Modells erfolgt ist. Diese Adaption kann mit denselben Techniken der Komposition und Erweiterung von Practices durchgeführt werden, wie zum Zeitpunkt der Erstellung von Methoden. Dadurch ist es insbesondere möglich, dass die Beschreibungen von Alphas und Alpha States geändert und erweitert werden können, ohne dass deren Instanzen ungültig werden.

5 Erfahrungen und Ausblick

Die Entwicklung von ESSENCE basiert auf mehrjährigen Erfahrungen in industriellen Projekten, aus denen die ESSENCE-Spezifikation mehrere Beispiele enthält, in denen vorhandene Vorgehensmodelle erfolgreich in ESSENCE modelliert wurden. Ferner wird dort an Beispielen gezeigt, wie aus einer Menge von Practices auf Basis des Kernels Entwicklungsmethoden für verschiedene Projektkategorien (z.B. Prototypprojekte oder Maintenance-Projekte) zusammengestellt werden, wobei in den verschiedenen Methoden unterschiedliche Aspekte fokussiert werden.

Die SEMAT-Initiative betreibt derzeit (Stand Januar 2013) die Standardisierung von ESSENCE im Rahmen der OMG. Die reine technische Dokumentation wird außerhalb des Standards durch Publikationen [JNM+13] und Workshops [SWE12] ergänzt.

Literatur

[Epp11] Thomas Epping. *Kanban für die Softwareentwicklung*. Informatik im Fokus. Springer Berlin Heidelberg, 2011.

[ESMB12] Brian Elvesaeter, Michael Striewe, Ashley McNeile und Arne-Jorgen Berre. Towards an Agile Foundation for the Creation and Enactment of Software Engineering Methods: The SEMAT Approach. In *Second Workshop on Process-based approaches for Model-Driven Engineering (PMDE 2012)*, 2012.

[HSGP05] Brian Henderson-Sellers und Cesar Gonzalez-Perez. The rationale of powertype-based metamodelling to underpin software development methodologies. In *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling - Volume 43*, APCCM '05, Seiten 7–16, 2005.

[ISO07] Software Engineering – Metamodel for Development Methodologies, ISO/IEC 24744. International Organization for Standardisation (ISO), February 2007.

[JNM+13] Ivar Jacobson, Pan-Wei Ng, Paul E. McMahon, Ian Spence und Svante Lidman. *The Essence of Software Engineering: Applying the SEMAT Kernel*. Addison-Wesley Professional, 2013.

[KMSG+12] Mira Kajko-Mattsson, Michael Striewe, Michael Goedicke, Ivar Jacobson, Ian Spence, Shihong Huang, Paul McMahon, Bruce MacIsaac, Brian Elvesater, Arne J. Berre und Ed Seymour. Refounding software engineering: The Semat initiative (Invited presentation). In Martin Glinz, Gail C. Murphy und Mauro Pezzè, Hrsg., *ICSE*, Seiten 1649–1650. IEEE, 2012.

[SPE08] Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0, Document formal/2008-04-01. Object Management Group (OMG), April 2008. `http://www.omg.org/spec/SPEM/2.0/`.

[SWE] IEEE Software Engineering Body of Knowledge, Version 3. `http://www.computer.org/portal/web/swebok`.

[SWE12] SEMAT Workshop on Evaluation of Essence, 2012. `http://semat.org/?page_id=678`.