Seminar

Softwareentwicklung in der Wissenschaft

Softwareproduktlinienentwicklung

Christoph Kuberczyk

Abgabedatum: 15.09.2015

Betreuung:
Julian Kunkel

Inhaltsverzeichnis

1 Einleitung

Bei der Softwareproduktlinienentwicklung (SPLE) handelt es sich um eine Art von Softwareentwicklung, die auf die Erschaffung von vielen ähnlichen Softwareprodukten („Softwareproduktlinie“ oder „Softwareproduktfamilie“ genannt) abzielt. Eine Produktlinie wird durch die Verwendung einzelner bei der Softwareentwicklung benötigter Artefakte (im englischsprachigen Raum meist „core assets“ genannt) gekennzeichnet, die in der Regel für die Entwicklung von mehreren Mitgliedern der Produktfamilie gebraucht werden. Der systematische Ansatz, einzelne Softwarebausteine für häufige Wiederverwendung herzustellen und zu benutzen, ist der Kernpunkt von SPLE. Diese Arbeit gibt eine Einführung in das Thema.
Im zweiten Kapitel wird der typische Ablauf bei der SPLE beschrieben. Dieser wird in zwei große Teilbereiche unterteilt. Zum einen in das Domain Engineering, bei dem zunächst lediglich die einzelnen Artefakte, die für die spätere Produktherstellung gebraucht werden, hergestellt und gesammelt werden. Zum anderen das Application Engineering, das die Entwicklung konkreter Softwareprodukte als Ziel hat.
Das dritten Kapitel behandelt das Thema Variabilität, das den Kernpunkt bei der Entwicklung von Softwareproduktlinien bildet. Dieses Kapitel geht auf Arten von Variabilität und die Darstellung dieser im Domaindesign ein.
Im vierten Kapitel werden die Vor- und Nachteile, die typisch für SPLE sind, besprochen. Zunächst wird die Motivation genannt, die viele Unternehmen antreibt, SPLE zu benutzen. Danach werden die Risiken, die die Einführung von SPLE mit sich bringt, besprochen. Das Ende des Kapitels geht dann auf Möglichkeiten ein, wie man die drohenden Risiken minimieren oder ganz vermeiden kann.
Die Arbeit schließt mit einer Zusammenfassung der wichtigsten Punkte und einem Ausblick für die SPLE für die Zukunft.

2 Entwicklung einer Produktlinie

SPLE ist durch die Aufteilung des Softwareentwicklungsprozesses in zwei große Teilbereiche gekennzeichnet. Zum einen in den Aufbau einer Plattform, die verschiedene Artefakte, die für die Entwicklung konkreter Produkte benötigt werden, beinhaltet. Der Begriff „Artefakt“ ist in diesem Zusammenhang ein Sammelbegriff, der unterschiedliche Dinge für den Herstellungsprozess von Software umschließt. Alles was für die Entwicklung von Softwareprodukten nützlich sein kann, kann ein Artefakt auf der Plattform werden. Beispielsweise können Softwarearchitekturen, Softwaremodelle, Domänenmodelle, aber auch Dinge wie Tests oder Use-

Cases ein Teil der Plattform werden. Das Herstellen und Sammeln dieser Artefakte nennt man Domain Engineering(DE). Konkrete Produkte für die Produktlinie werden hier noch nicht hergestellt.[1]

Fertige Softwareprodukte entstehen erst im Application Engineering(AE). Aufbauend auf den Artefakten der Plattform werden die konkreten Produkte abgeleitet. Im Idealfall werden einzelne Artefakte einfach zusammengesetzt, um nutzbare Software zu erstellen.[2]

Im Folgenden wird sowohl der genaue Ablauf beim DE als auch derjenige beim AE beschrieben.

2.1 Domain Engineering

Das DE lässt sich in fünf Schritte unterteilen.

1. Product Management
2. Domain Requirements Engineering
3. Domain Design
4. Domain Realisation
5. Domain Quality Assurance[3]

In der Literatur finden sich auch andere Unterteilungen, wobei die Unterteilungen sich aber stets ähneln.[4]

2.1.1 Product Management

Softwareproduktlinienentwicklung ist eine proaktive Art der Softwareentwicklung. Das bedeutet, dass die meisten wichtigen Entscheidungen für die Entwicklung bereits am Anfang des Entwicklungsprozesses getroffen werden müssen und spätere Änderungen meist nur schwer vorzunehmen sind. Aufgrund der Tatsache, dass bis zur Erstellung der ersten Softwareprodukte bereits eine große Menge an Aufwand betrieben werden muss, wird dieser Ansatz auch der Big-Bang-Ansatz genannt.[5] Aus diesem Grund kommt dem Management eine große Rolle bei der Softwareentwicklung zu. Das Management muss festlegen, welche Produkte im Produktportfolio enthalten sind. Diesen Vorgang nennt man „Scoping“. Diese Aufgabe ist gerade bei SPLE besonders wichtig. Dadurch, dass bei SPLE die Plattform mit viel Aufwand fertiggestellt wird, bevor das Unternehmen erstmals Feedback zu seinen Produkten erhält, können

1 Vgl. He Zhongua, „Einführen einer Software Produktlinie[n], Siegen 2015, S.2
2Vgl. Pohl, Klaus; Metzger, Andreas Software Product Line Engineering and Variability Management: Achievements and Challenges 2014, S.1
3Vgl. ebd, S.2
4 Vgl. Diaz, Jessica; Perez, Jennifer; Alarcon, Pedro; Garbajosa, Juan, Agile Product Line Engineering - A Systematic Literature Review, 2011, S.3
5 ebd

falsche Entscheidungen große Auswirkungen auf den Erfolg der Produktlinie haben.[6] Das Scoping wird zum Teil, um seine Wichtigkeit für den Entwicklungsprozess darzustellen, in einigen Modellen nicht als Unterpunkt des DE dargestellt, sondern als (eigener) dritter großer Punkt neben dem DE und dem AE. [7]

Das Management muss bei der Erstellung des Produktportfolios eine Gratwanderung unternehmen. Zum Einen sollte eine Produktlinie nicht zu starr festgelegt sein, da gerade die Möglichkeit, mit einer Plattform viele verschiedene Produkte herzustellen, den Reiz der SPLE ausmacht. Damit der Aufwand für die Erstellung der Plattform allerdings nicht zu groß wird und nicht zu viel Zeit in Anspruch nimmt, ist es andererseits ratsam, die Vielfalt innerhalb einer Produktfamilie zu begrenzen.[8]

2.1.2 Domain Requirements Engineering

Das Domain Requirements Engineering dient als die Schnittstelle zwischen dem Management und dem Entwicklungsteam.[9] Hierbei muss festgelegt werden, welche Gemeinsamkeiten die verschiedenen geplanten Produkte miteinander teilen, aber auch, in welchen Punkten sich die verschiedenen Produkte unterscheiden sollen. Ähnlich wie beim Scoping wird diskutiert, wie viel Variabilität möglich ist, ohne den technischen Aufwand zu groß werden zu lassen. Das Festlegen bestimmter Eigenschaften muss also im richtigen Maße und an den richtigen Stellen stattfinden.[10]

2.1.3 Domain Design

Beim Domain Design geht es darum, die Referenzarchitektur der Produktlinie zu entwerfen. Die Referenzarchitektur einer Produktlinie unterscheidet sich von der Architektur eines einzelnen Softwaresystems durch die Variationspunkte, die die Referenzarchitektur enthält. Da alle Mitglieder einer Produktfamilie auf dieser Architektur aufbauen, ist hier besondere Sorgfalt geboten.[11]

6Vgl. ebd

7Vgl. Kaindl, Hermann; Mannion, Mike„A Feature-Similarity Model for Product Line Engineering“ in „Software Reuse for Dynamic Systems in Schaefer, Ina; Stamelos, Ioannis(Hgg.) „the Cloud and Beyond: 14th International Conference on Software Reuse“ 2015, S.40

8Vgl. Pohl; Metzger, „Software Product Line Engineering and Variability Management: Achievements and Challenges“ 2014, S.6

9Vgl. ebd S.6-7

10Vgl. ebd S.7

11Vgl. ebd S.8

2.1.4 Domain Realization

Bei der Domain Realization werden die wiederverwendbaren Bestandteile der Produktlinie implementiert. Da die fertigen Artefakte dafür gedacht sind, dass sie mehrmals verwendet werden, muss hierbei besonderer Wert auf Performance und Effizienz gelegt werden, da bei schlechter Umsetzung der Anforderungen gleich mehrere Produkte davon betroffen wären.[12]

2.1.5 Domain Quality Assurance

Die Qualitätssicherung für die Artefakte auf einer Plattform ist eine der größten Herausforderungen für die Entwickler einer Softwareproduktlinie. Wie schon bei den anderen Schritten im DE erwähnt, hat jede begangene Nachlässigkeit negative Auswirkungen auf gleich mehrere Produkte der Produktlinie. Von daher sollte die Qualitätssicherung nicht erst bei den fertigen Softwareprodukten beginnen, sondern bereits bei den einzelnen Komponenten der künftigen Systeme.[13]

Die Qualitätssicherung bei Produktlinien ist gemeinhin schwieriger als bei Einzelsystemen. Da sich aus den Artefakten einer Produktlinie viele verschiedene Produkte zusammensetzen lassen, ist es unmöglich, alle ableitbaren Produkte auf Fehler zu untersuchen. Sowohl bei Versuchen, formale Verifikationen durchzuführen, als auch bei der Verwendung von Testfällen, ist die Variabilität der Produktlinien und die Möglichkeit die verschiedene Komponenten auf verschiedenste Art zusammenzusetzen ein großer Faktor, der nahezu vollständiges Testen unmöglich macht. Dennoch sollten sowohl einzelne Artefakte isoliert als auch Kompositionen von Komponenten, für die eine spätere Zusammensetzung vorgesehen ist, gründlich untersucht werden, um Fehler schon in diesem Stadium aufzudecken.[14]

2.2 Application Engineering:

Beim Application Engineering werden konkrete Produkte unter Wiederverwendung der Artefakte abgeleitet. Es lässt sich in vier Schritte unterteilen.

1. Application Requirements Engineering
2. Application Design
3. Application Realization
4. Application Quality Assurance[15]

12Vgl. ebd S.8
13 Vgl. ebd
14 Vgl. ebd
15 Vgl. ebd

Auch beim AE werden in der Literatur die Schritte zum Teil anders zusammengefasst, ohne sich inhaltlich stark von dieser Aufteilung zu unterscheiden.[16]

2.2.1 Application Requirements Engineering

Im Application Engineering werden die Ergebnisse aus dem Domain Engineering genutzt, um kundenspezifische Produkte zu definieren. Welche Produkte aus den Artefakten abgeleitet werden, wird entweder vom Management festgelegt oder von Kunden, die Interesse an Produkten der Produktlinie geäußert haben. In beiden Fällen ist es nötig, die Anforderungen an das Softwareprodukt mit den Artefakten der Plattform zu vergleichen, um einen hohen Grad an Wiederverwendung zu erreichen und dabei dennoch Kundenwünsche nicht zu sehr zu missachten. Ein Softwareprodukt komplett aus den Artefakten der Plattform abzuleiten, ohne auch nur kleine Änderungen vorzunehmen, ist in der Praxis aber kaum möglich.[17]

2.2.2 Application Design

Beim Application Design wird die Architektur des zu fertigenden Softwareproduktes erstellt. Hierzu werden die Kundenanforderungen mit der Referenzarchitektur der Produktlinie verglichen. Die Referenzarchitektur muss angepasst werden, sodass eine konkrete Architektur vorliegt. Dazu müssen zumindest die Variationspunkte der Referenzarchitektur so spezifiziert werden, dass sie den Kundenwünschen am ehesten entsprechen. Je nach Kundenwunsch kann die Referenzarchitektur aber auch erweitert oder angepasst werden.[18]

2.2.3 Application Realization

Bei der Application Realization wird mithilfe der erstellten Architektur und den Artefakten der Plattform eine fertige Software erstellt. Je höher die Übereinstimmung von der geplanten Software mit den Artefakten der Plattform ist, desto geringer ist der technische Aufwand in diesem Schritt. Da ein gewünschtes Softwaresystem in der Praxis quasi nie vollständig aus der Plattform ableitbar ist, kann dieser Schritt aber fast immer nicht übersprungen werden.[19]

16 Vgl. Diaz; Perez; Alarcon; Garbajosa, Agile Product Line Engineering - A Systematic Literature Review, 2011, S.3
17 Vgl. Metzger; Pohl 2014, S.9
18 Vgl. ebd S.9-10
19 Vgl. ebd S. 10

2.2.4 Application Quality Assurance

Auch bei den kompletten Softwareprodukten sollte nochmal eine Qualitätssicherung stattfinden. In der Regel wird hier auf Testfälle zurückgegriffen, sowohl auf welche aus der Plattform als auch auf gegebenenfalls neu erstellte. Grund für systematisches Testen der Endprodukte ist zum einen die Tatsache, dass die meisten Produkte nicht komplett aus den Artefakten zusammengesetzt werden und daher meistens zumindest eine kleine Abweichung von den zuvor getesteten Produkten aus dem Domain Engineering haben. Zum anderen kann die Qualitätssicherung im AE weit zuverlässiger stattfinden, da die Probleme Qualitätssicherung aus dem DE nicht mehr vorhanden sind (siehe 2.1.5).[20]

3 Variabilität

Variabilität ist im SPLE die „Voraussetzung für die gezielt Konstruktion (Domain Engineering) und Wiederverwendung (Application Engineering) von Produktartefakten in Software-Produktlinien. Variabilität definiert innerhalb einer Produktlinie jene Teile, die durch Selektion an unterschiedliche Kundenbedürfnisse angepasst werden können.“[21][1]

Da die Plattform bei SPLE eine Vielzahl an unterschiedlichen Artefakten enthalten soll, tritt die Variabilität einer Softwareproduktlinie auch an unterschiedlichen Stellen zutage. Variabilität kann in Features, d.h. bei der Funktionalität der Software, in Abläufen, im Datenumfang, im Dateiformat, im Systemzugang, in Benutzer- und Systemschnittstellen und in der Qualität auftreten.[22]

Da diese Variabilitäten im Application Engineering für die Produktableitung genutzt werden, ist es nötig, die Variabilität verständlich und vollständig darzustellen. Dazu müssen Variationspunkte und die dazugehörigen Varianten kenntlich gemacht Die Darstellung muss dabei mehr beinhalten können als die reine Unterteilung in variable und verpflichtende Bestandteile. So können sich zwar Varianten untereinander indifferent verhalten, in der Regel bestehen aber Abhängigkeiten zwischen verschiedenen Varianten. So können sich Varianten auch gegenseitig ausschließen oder eine Variante kann nur umgesetzt werden, wenn an einem anderen Variationspunkt eine bestimmte Variante gewählt wird.[23] Wie ein allgemeines Variabilitätsmodell aussieht zeigt Abbildung 1.

---

20 Vgl. ebd. S10-11

21 Bühne, Stan; Halmans Günter; Lauenroth, Kim; Pohl, Klaus „Variabilität in Software-Produktlinien in „Software-Produktlinien, Methoden, Einführung und Praxis, Heidelberg 2004, S.13

22 Vgl. ebd., S.14-16

23 Vgl. ebd., S.17-18

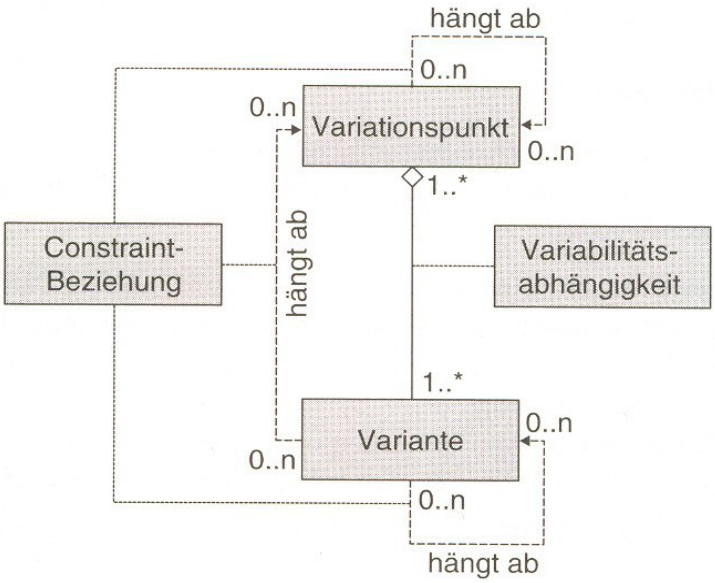

Abbildung 1[24]: Variabilitätsmodell

Um Variabilität entsprechend der Abhängigkeit der Varianten untereinander zu modellieren, hat sich in der Praxis das orthogonale Variabilitätsmodell durchgesetzt (siehe Abbildung 2). Im Vergleich zu dem allgemeinen Variabilitätsmodell bietet eine orthogonale Variabilitätssicht einen Überblick der Variabilität über alle Entwicklungsphasen hinweg. Verschiedene Darstellungen von Variabilität sind hier in einer Ansicht möglich.

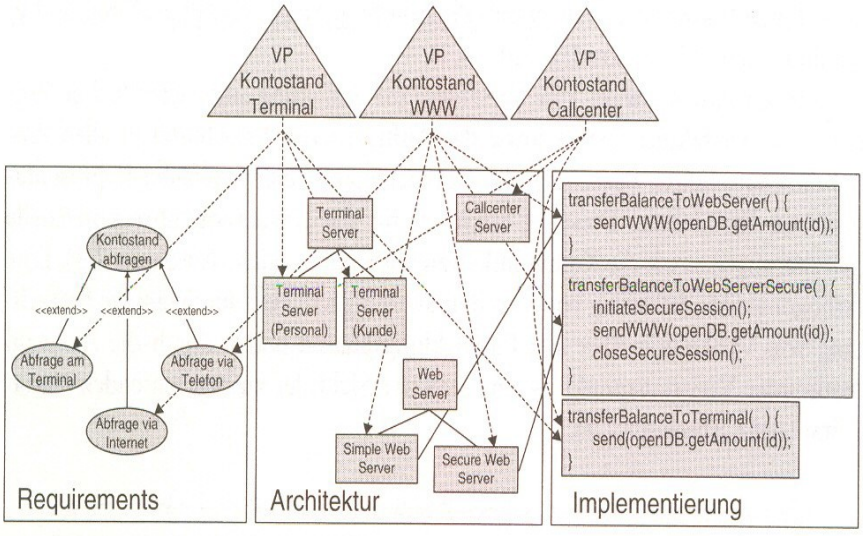

Abbildung 2[25]: Das orthogonale Variabilitätsmodell

Aus Abbildung 2 erkennt man beispielsweise, dass in den Anforderungen („Requirements“) davon ausgegangen wird, dass eine Kontostandabfrage mittels Telefon ermöglicht werden

24 Bühne, Stan; Halmans Günter; Lauenroth, Kim; Pohl, Klaus „Variabilität in Software-Produktlinien in „Software-Produktlinien, Methoden, Einführung und Praxis, Heidelberg 2004, S. 17

25 Ebd. S.23

soll. Die „Abfrage via Telefon“ ist mit der Klasse „Callcenter Server“ verbunden, was in diesem Fall heißt, dass dieses Callcenter aufgrund der geforderten Telefonabfrage benötigt wird. Mit diesem Modell gibt man den Entwicklern in allen Phasen des Entwicklungsprozesses die gleiche Sicht auf die zu entwerfende Produktlinie, sodass alle Beteiligten über das gleiche Verständnis verfügen können.[26]

4 Vor- und Nachteile der SPLE

SPLE hat sich in den vergangenen Jahren als erfolgreiche Art der Softwareentwicklung etabliert und gerade in Unternehmen, die sich wettbewerbsstarken Märkten beweisen müssen, hat sich SPLE bewährt. SPLE bietet diesen Unternehmen einige Vorteile.

4.1 Vorteile

Die Vorteile, die SPLE mit sich bringt, sobald die Plattform mit den Artefakten eingerichtet ist, beruhen auf der Wiederverwendbarkeit der bereits bestehenden Artefakte. Zum einen ist durch das durch die Artefakte gegebene Grundgerüst der Schwierigkeitsgrad für die Herstellung neuer Produkte deutlich geringer, als müsse man neue Software ohne jegliche Vorarbeit herstellen. Zum anderen erleichtert das Vorhandensein von vielen ähnlichen Produkten die Wartbarkeit der Programme, da die zuständigen Mitarbeiter sich nicht ständig in neue Architekturen einarbeiten müssen.[27] Der Hauptgrund für die Verwendung von SPLE ist allerdings nicht die Bequemlichkeit für die Bediensteten. Vielmehr ist es für viele Unternehmen gerade in Zeiten der Globalisierung wichtig, dass sie im Vergleich zu immer stärker werdenden Konkurrenz im Wettbewerb mithalten können. Durch die Wiederverwendung von Softwarebausteinen kann neue Software billiger produziert werden. Forschungen haben ergeben, dass der Break-even-Point der Produktionskosten bei guten Softwareproduktlinien im Schnitt nach Herstellung von 3-4 Produkten erreicht ist.[28] Auch die Time-to-Market sinkt signifikant, Untersuchungen haben Senkungen um bis zu 98% ergeben.[29] Das bedeutet, dass individuelle Kundenwünsche leichter befriedigt werden können. Zudem hat SPLE auch einen Einfluss auf die Produktqualität und damit auf den Kundennutzen. Dadurch, dass zum einen die einzelnen Bestandteile der Software im Domain Engineering getestet und optimiert wurden und zum an-

26 Vgl. ebd. S.22-23
27Vgl. Berger, Christian; Krahn, Holger; Rendel, Holger; Rumpe, Bernhard, Feature-basierte Modellierung und Verarbeitung von Produktlinien am Beispiel eingebetteter Software, S.1
28 Vgl. He, 2015, S.4
29Vgl. Pohl, Klaus; Metzger, Andreas, Software Product Line Engineering and Variability Management: Achievements and Challenges, S.1

deren häufig bereits ähnliche Produkte getestet wurden, verringert sich die Anzahl der Fehler, die vor allem im Zeitraum unmittelbar nach der Markteinführung auftreten, deutlich.[30]

4.2 Nachteile

SPLE kann einem Unternehmen also viele Vorteile bringen. Nichtsdestotrotz ist diese Art der Softwareentwicklung nicht völlig risikolos und für alle Arten von Unternehmen geeignet. Denn die Verwendung von SPLE birgt auch Gefahren.

Zum einen sind die Initialkosten, die für den Aufbau der Plattform nötig sind, deutlich höher als bei der Entwicklung eines einzelnen Softwareproduktes. Für kleine oder neu gegründete Unternehmen ist es daher riskant, mit dem Aufbau einer Produktlinie zu beginnen. Ohne die nötigen finanziellen Mittel in der Hinterhand droht der Bankrott im Zeitraum nach der Investition, wenn der Break-even-Point nicht rechtzeitig erreicht wird.[31]

Zudem kann die Einführung von SPLE auch scheitern, wenn das Scoping falsche Prämissen gesetzt hat (siehe 2.1.1). Gerade bei neuen Märkten ist es schwierig abzusehen, in welche Richtung sie sich entwickeln und welche Funktionalitäten von den Kunden erwartet werden.[32] Fehlendes Kundenfeedback während des Domain Engineering erschwert das Planen der Produkte zusätzlich.[33]

Zudem können selbst bei bereits bestehender Plattform unerwartete Probleme auftauchen und das Konzept von SPLE ad absurdum führen. Beispielsweise kommt es mitunter vor, dass Produktfamilien innerhalb kürzester Zeit veralten. Denn Produktfamilien sind zwar variabel und schnell anpassbar, allerdings gilt dies nur für Änderungen, die beim Scoping miteinbezogen wurden oder die zumindest keine systemimmanente Änderung verlangen. Als Beispiel für dieses allgemeine Problem bei Produktlinienentwicklung kann der „Smartphone-Boom“, der 2007 begann, angesehen werden, der zur plötzlichen Veraltung „herkömmlicher Mobiltelefone“ und dem Absturz von Nokia als Marktführer führte. Dieses Beispiel zeigt zwei Gefahren: Die Inflexibilität, sich nicht von seiner Produktlinie zu trennen und die Nachteile, die sich ergeben, wenn man aufgrund einer funktionierenden Produktlinie keine innovative Forschung mehr betreibt.[34]

30 Vgl He, 2015, S.4
31 Vgl. Diaz; Perez; Alarcon; Garbajosa, „Agile Product Line Engineering - A Systematic Literature Review“ 2011, S.7-8
32 Vgl. ebd
33 Vgl. ebd, S.10
34 Vgl. Gamillscheg, Hannes, „Handy-Marktführer streicht 4000 Arbeitsplätze- Nokia hat den Anschluss an Smartphone-Geschäft verpasst“, Berliner Zeitung, 28.04.2011

4.3 Modifikation des SPLE-Modells

In seiner statischen Form ist die SPLE für die Entwicklung von Produktfamilien aufgrund der genannten Nachteile für viele Unternehmen also wenig empfehlenswert. Um auf die Vorteile, die Softwareproduktlinien mit sich bringen, nicht komplett verzichten zu müssen, werden in der Praxis verschiedene Abwandlungen des unflexiblen SPLE-Modells vorgenommen, um die Risiken, die bei der Einführung von Produktlinien drohen, zu verringern. Einige der wichtigsten Ansätze werden an dieser Stelle vorgestellt.

Ein in der Praxis regelmäßig genutzter Ansatz ist eine Änderung der Sichtweise, die Erstellung von Softwareproduktlinien als einen Big-Bang-Prozess zu betrachten. Tatsächlich werden in der Wirtschaft die meisten Produktlinien nicht auf rein proaktive Weise hergestellt.[35] Anstatt DE und AE zeitlich voneinander zu trennen, verschwimmt bei der Produktlinienherstellung häufig die Grenze zwischen den beiden Entwicklungsarten. Anstatt das AE erst nach dem Ende des DE zu beginnen, findet die Produktableitung bereits statt, nachdem nur die wichtigsten Artefakte für die Plattform erstellt wurden. Die übrigen Artefakte werden in der Folge parallel zum AE hergestellt, wobei auf diese Weise aus der Produktableitung gewonnene Erkenntnisse für die Erstellung der übrigen Artefakte genutzt werden können (reaktiver Ansatz).[36] Die Vorteile liegen im Vergleich zum proaktiven Ansatz in einer kürzeren Time-to-Market der ersten Produkte der Produktlinie und der Möglichkeit, schneller Feedback von den Konsumenten zu erhalten.

Häufig findet in der Praxis sogar komplett darauf verzichtet, das DE der Produktableitung voranzustellen. Das bedeutet, dass entweder DE und AE von Beginn an parallel laufen oder die Produktherstellung bereits zeitlich vor dem Entwickeln der Plattform stattfindet. Der zweite Fall tritt in der Regel dann auf, wenn ein als Einzelprodukt gedachtes Objekt häufig nachgefragt wird und das Management des Unternehmens aufgrund dessen weitere ähnliche Produkte herstellen will. In diesem Fall sind die Risiken des Scoping deutlich geringer und da Teile des bestehenden Produktes bereits als Artefakte für die Plattform genutzt werden können, sind die Initialkosten, die für die Erstellung der Plattform anfallen, geringer.[37]

Eine weitere Möglichkeit, Gefahren des SPLE zu umgehen, besteht in der Einbindung agiler Methoden in den Entwicklungsprozess. Agile Softwareentwicklung und SPLE teilen eine Reihe von Gemeinsamkeiten und Vorteilen, die sie ihren Anwendern versprechen. Beispielsweise zeichnen sich beide Arten der Softwareentwicklung durch eine geringe Time-to-Market aus. Beide Arten legen einen hohen Wert darauf, Kundenwünsche individuell zu erfüllen und va-

35Vgl. Pohl; Metzger, „Software Product Line Engineering and Variability Management: Achievements and Challenges“ 2014, S.1
36Vgl. ebd S.3
37 Vgl. ebd

riabel anpassbar zu sein. Durch Komposition beider Ansätze sollen diese Vorteile erhalten bleiben und die Nachteile der einzelnen Entwicklungsmethoden reduziert werden.[38] Aus Sicht des SPLE spricht für die Aufnahme agiler Methoden in den Entwicklungsprozess beispielsweise die geringere Zeit, die gebraucht wird, und der geringere Aufwand, der erbracht werden muss, bis die ersten fertigen Produkte entstanden sind.[39] Da agile Softwareentwicklung und SPLE allerdings von ihrer Herangehensweise komplett unterschiedlich sind - SPLE plant den Aufbau der Produkte vorab, während agile Programmierung gerade durch die Möglichkeit, alle Facetten der Produkte ändern zu können, ausgezeichnet ist - , ist eine Zusammenarbeit beider Ansätze in der Praxis mitunter schwierig.[40]

Ein Versuch der Integration stellt beispielsweise der A-Pro-PD-Ansatz dar. A-Pro-PD ist die Abkürzung für „Agile Process Model for Product Derivation“ und zielt darauf ab, bei der Produktableitung eine Verbesserung des reinen SPLE-Modells zu sein. Bei A-Pro-PD werden agile Methoden also beim Application Engineering eingesetzt. Bei den meisten Beispielen, bei denen SPLE um agile Methoden erweitert wird, werden diese Methoden beim AE und nicht beim Domain Engineering eingesetzt. Grund hierfür ist das hohe Maß an Planung, das beim DE benötigt wird und schwierig mit agilen Methoden kombinierbar ist. Die konkrete Produktentwicklung hat im Gegensatz dazu mehr mit agiler Entwicklung gemein.[41]

A-Pro-PD entstand bei der Entwicklung des vom Irischen Softwareentwicklungs-Forschungszentrum Lero entwickelten Pro-PD (Process Model for Product Derivation) , das auf die Entwicklung eines generischen Prozessreferenzmodells für die Ableitung von Softwareprodukten abzielte.

Zusammengefasst unterteilt A-Pro-PD den Entwicklungsprozess in zwei Ebenen(Layers): die Increments Layer und die Iteration Lifecycle Layer.[42]

In der Increments Layer werden Aufgaben in kleine Einheiten unterteilt, wie beispielsweise die Anpassung von Artefakten der Plattform. Bei der Iteration Lifecycle Layer werden aufbauend auf den Leistungen der Increment Layer die Produkt entwickelt, wobei sich dieser Ablauf stets komplett von der Anforderungsanalyse über die Produktherstellung bis zur Qualitätssicherung und Bewertung erstreckt. Increment Layer und Lifecycle Layer bauen gegenseitig auf ihren Erkenntnissen und Zwischenprodukten auf, um am Ende sowohl eine optimierte Plattform als auch optimierte Endprodukte entwickelt zu haben.[43]

---

38 Vgl. Diaz; Perez; Alarcon; Garbajosa, „Agile Product Line Engineering - A Systematic Literature Review“, 2011, S.8-9

39 Vgl. ebd

40 Vgl. ebd, S.10-11

41Vgl. O´Leary, Padraig; McCaffery, Fergal; Thiel, Steffen; Richardson Ita An Agile Process Model for Product Derivation in Software Product Line Engineering 2010 S.2

42 Vgl. ebd S.6-8

43 Vgl. ebd

5 Fazit

Die Softwareproduktlinienentwicklung ist eine Art der Softwareentwicklung, die ihren Anwendern bei der Entwicklung mehrerer ähnlicher Softwareprodukte eine Menge an Vorteilen bietet. Die geringe Time-to-Market und die geringen Produktionskosten für die Erschaffung von weiteren Mitgliedern einer Softwarefamilie, machen diese Art der Softwareentwicklung für viele Unternehmen attraktiv. Vor allem für Unternehmen, die einem starken Wettbewerb ausgesetzt sind, ist SPLE diejenige Softwareentwicklungsart, die den größten Nutzen erbringt. Um eine Softwareproduktlinie aufzubauen, muss gerade am Anfang viel Geld und Aufwand investiert werden. Dieser Nachteil macht es für kleine und umsatzschwache Unternehmen schwierig, mit dem Aufbau einer Produktlinie zu beginnen. Dennoch besitzen auch solche Organisationen die Möglichkeit dazu, wenn sie die Artefakte ihrer Plattform aus bereits bestehenden Produkten entnehmen.

Die Variabilität spielt bei der Softwareproduktlinienentwicklung eine große Rolle. Das richtige Maß an Variabilität festzulegen ist am Anfang der Entwicklung eine schwierige Aufgabe. Bei richtiger Umsetzung wird aber eine große Vielfalt an Produkten ermöglicht, die mit der orthogonalen Modellierung auf übersichtliche Art darstellbar ist und Kundenwünsche individuell erfüllen kann. Durch Adaptionen des rein proaktiven Ansatzes - etwa der Einbindung agiler Methoden in den Ablauf der Produktableitung - ist es möglich, die Herstellung von Produkten individuell zu optimieren. Aufgrund all dieser Punkte wird Softwareproduktlinienentwicklung wahrscheinlich auch in der Zukunft eine der am häufigsten verwendeten Entwicklungsarten von Software bleiben.

6 Quellenverzeichnis

- Böckler, Günter; Knauber, Peter; Pohl, Klaus; Schmid, Klaus: „Software-Produktlinien - Methoden, Einführung und Praxis“ Heidelberg 1. Auflage 2004

- Schaefer, Ina ;Stamelos, Ioannis „Software Reuse for Dynamic Systems in the Cloud and Beyond: 14th International Conference on Software“ Miami 2015

- Oleary,´Pádraig; McCaffery, Fergal; Thiel, Steffen, Richardson, Ita „An Agile Process Model for Product Derivation in Software Product Line Engineering“ Limerick, Dundalk, Furtwangen 2010

- He, Zhongua, „Einführen einer Software Produktlinie[n], Siegen 2015

- Pohl, Klaus; Metzger, Andreas „Software Product Line Engineering and Variability Management: Achievements and Challenges“ 2014

- Diaz, Jessica; Perez, Jennifer; Alarcon, Pedro; Garbajosa, Juan, „Agile Product Line Engineering - A Systematic Literature Review“, 2011

- Berger, Christian; Krahn, Holger; Rendel, Holger; Rumpe, Bernhard „Feature-basierte Modellierung und Verarbeitung von Produktlinien am Beispiel eingebetteter Software“ Aachen/Braunschweig 2009

- Voelter, Markus; Visser, Eelco, „Product Line Engineering using Domain-Specific Languages“ Stuttgart/Delft 2010

- Gamillscheg, Hannes, „Handy-Marktführer streicht 4000 Arbeitsplätze- Nokia hat den Anschluss an Smartphone-Geschäft verpasst“, Berliner Zeitung, 28.04.2011