ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Analyse zur Eignung verschiedener Prozessmodellierungsnotationen zur Realisierung von iterativen Softwareprozessmodellen

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
David Rothmaier
david.rothmaier@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Gregor Grambow, M.SC.

2014

Fassung 26. August 2014

Satz: PDF-LaTeX 2ε

Kurzfassung

Die wirtschaftliche Bedeutung des Bereiches Software Engineering, ein Teilgebiet der Informatik ist bereits sehr groß und wird in den kommenden Jahren noch weiter zunehmen. Dadurch steigen auch die Anforderungen, die in Bezug auf Funktionalität, Komplexität, und Qualität an Softwaresysteme gestellt werden.

Um die Entwicklung solcher Softwaresysteme beherrschbar zu machen und diese zu organisieren, werden seit der sogenannten 'Softwarekrise' immer mehr verschiedene Modelle entwickelt, nach denen Entwickler mit einem ingenieursmäßigen Vorgehen handeln können.

Unter anderem gibt es folgende Modelle: Der Open Unified Process (kurz: Open UP), dieser enthält die Kernbestandteile des Rational Unified Process (RUP) und ist für kleine, eher agile Projekte mit kleineren Teams geeignet. Des weiteren gibt es die Agile Softwareentwicklung. Dabei sollen Entwickler kreativ arbeiten können und nicht von Verwaltungsaspekten eingeschränkt werden. Beispiele hierfür sind der Scrum Prozess und Extreme Programming (XP). Die Einarbeitung und der Vergleich solcher Softwareentwicklungsmodelle ist allerdings schwierig, da sie sich stark unterscheiden und in unterschiedlichsten Notationen dargestellt werden.

Eine Überführung solcher Modelle in eine standardisierte Notation ist somit wünschenswert. Geeignet hierfür erscheinen die standardisierte BPMN 2.0 Notation, UML-Aktivitätendiagramme sowie EPKs. Da diese Notationen jeweils in einem Gebiet relativ stark verbreitet sind. So ist BPMN der Standart in der Businesswelt, EPK's waren eine der ersten Versuche Prozesse darzustellen und Die UML genießt im Softwareengeneering eine starke Position.

Diese Bachelorarbeit beschäftigt sich deshalb mit der Analyse und Überführung von Softwareentwicklungsprozessen in die standardisierten Notationen der BPMN, UML und EPKs am Beispiel von drei unterschiedlichen Modellen: XP, dem Open UP und dem Scrum Prozess. Anschließend folgt eine Diskussion über die Verwendbarkeit der Notationstypen zur Darstellung der Prozesse.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Erstellung dieser Bachelor-Arbeit unterstützt und motiviert haben.

Ein besonderer Dank gilt Herrn Gregor Gambow, der meine Arbeit und somit auch mich betreut hat. Er stand mir stets für Fragen zur Verfügung und gab mir einige wertvolle Hinweise. Vielen Dank für die Geduld und Mühen.

Daneben gilt mein Dank Herrn Ulrich Riegger und meinem Vater Reinhold Rothmaier welche in zahlreichen Stunden Korrektur gelesen haben. Sie wiesen auf Schwächen hin und konnten als Fachfremde immer wieder zeigen, wo noch Erklärungsbedarf bestand.

Nicht zuletzt gebührt meinen Eltern Dank, da Sie mich während des Studiums nicht nur finanziell, sondern vor allem auch emotional immer unterstützt haben!

Inhaltsverzeichnis

1

Einleitung

Diese Arbeit behandelt Teile aus dem Bereich Softwareengineering der Informatik.

Software ist der zentrale Werkstoff des Informationszeitalters, das heißt, dass nicht zuletzt der Erfolg der Wirtschaft entscheidend von einer guten und vor allem fehlerarmen Software abhängt. Da jedoch der Softwareentwicklungsprozess ein sehr komplexer Prozess ist, werden gerade hier leicht Fehler gemacht. Je später die Fehler im Softwarentwicklungsprozss entdeckt werden, umso teurer wird deren Behebung. Siehe auch Abbildung 1.1.

Daher ist es äußerst wichtig, dass bei der Softwareentwicklung Fehler frühzeitig erkannt werden und auch der Kunde möglichst viel im Entwicklungsprozess mit eingebunden wird, um späteren Änderungswünschen vorzubeugen.

Um diesen Problemen zu begegnen, gibt es sehr verschiedenartige Prozessmodelle der Softwareentwicklung.

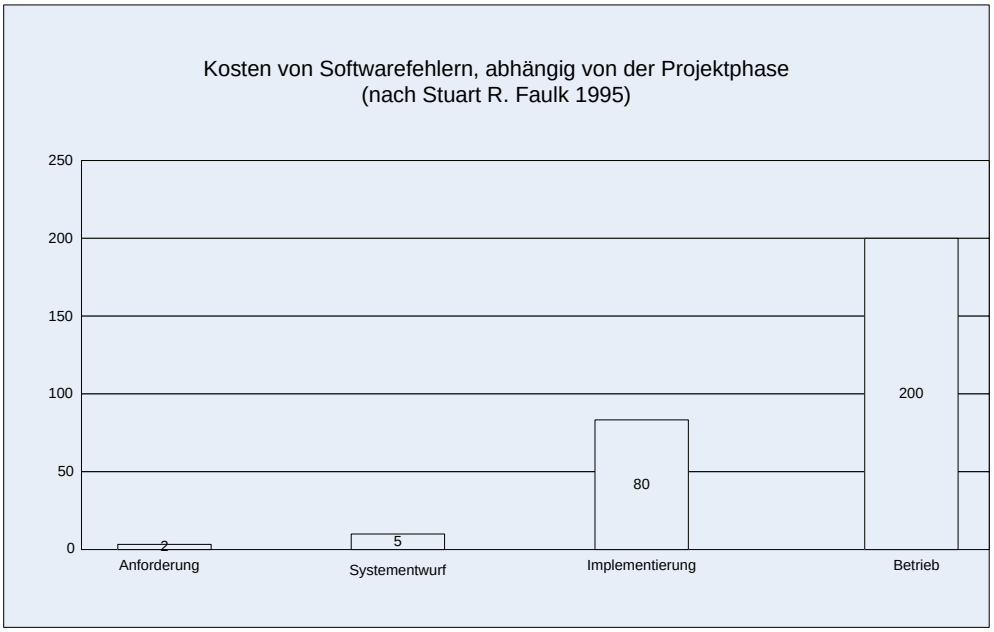

Abbildung 1.1: Kosten von Softwarefehlern in Abhängigkeit vom Entwicklungsstand. In Anlehnung an [blo 5].

1.1 Motivation

Leider sind die existierenden Softwareprozessmodelle schlecht miteinander vergleichbar. Jedes Modell ist in einer eigenen Notation notiert, was für das Verstehen des Modells jeweils das Erlernen einer neuen Notation voraussetzt. Dies macht es einem Unternehmen sehr schwer, sich für ein zum Unternehmen und zum Projekt passendes Prozessmodell zu entscheiden, da sich jedes Prozessmodell für andere Problemstellungen unterschiedlich gut eignet.

Es wäre daher schön, die Softwareprozessmdelle in einer standardisierten Notation vorliegen zu haben. Eine solche Notation sollte ausdrucksmächtig sein und möglichst alle Elemente der Modelle für alle Prozessbeteiligte verständlich abbilden können.

Aufgrund des hohen Verbreitungsgrades scheinen folgende Notationen hierfür als geeignet:

- Business Process Model and Notation (BPMN)

- Erreignisgesteuerte Prozessketten (EPKs)
- UML-Aktivitätendiagramm

Die BPMN 2.0 Notation (Business Process Model and Notation) scheint hierfür ein geeigneter Kandidat zu sein, da sie den Anspruch erhebt, eine gemeinsame Sprache für Business und IT zu sein und somit ein großes Maß an Verbreitung genießt.

EPKs scheinen geeignet, da es sich hierbei um die erste Notationen zur Darstellung von Prozessen handelt, welche weit verbreitet war.

Die UML-Aktivitätendiagramme sind ein Kandidat, da die UML als Gesamtes einen hohen Stellenwert im Softwareengineering einnimmt.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist es, die Notationen BPMN, EPKs und UML Aktivitätendiagramme auf ihre Eignung zur Darstellung von iterativen Softwareprozessmodellen zu untersuchen. Als iterative Prozesse sollen die drei Softwareentwicklungsprozesse Scrum, Xtreeme Programming (XP) und Open Unified Process (Open UP) verwendet werden. Diese sollen mittels erwähnter Notationen miteinander vergleichbar modelliert werden. Es soll bei der Modellierung hierbei strikt darauf geachtet werden, dass die Prozessmodelle so korrekt wie möglich abgebildet werden, um einen möglichst guten Vergleich schaffen zu können.

Ein weiteres Ziel ist es, die Darstellungsarten gegeneinader auf ihre Eignung für diesen Zweck zu diskutieren, um feststellen zu können, mit welcher Modellart sich Softwareentwicklungsprozesse am sinnvollsten vergleichbar darstellen lassen.

1.3 Aufbau der Arbeit

Nach einer kurzen Einführung in die Thematik und einer Einordnung in den Bereich der Informatik in Kapitel 1 beschäftigt sich das nachfolgende Kapitel 2 mit den Grundlagen, welche für das weitere Verstehen der Arbeit benötigt werden.

Kapitel 3 enthält das eigentliche Herzstück der Bachelorarbeit. Hier werden nacheinander die einzelnen Prozessmodelle graphisch dargestellt.

Im Anschluss werden in Kapitel 4 die Ergebnisse aus Kapitel 3 diskutiert. So beschäftigt es sich mit der Eignung der verschiedenen Diagrammarten zu der Darstellung der Softwareentwicklungsprozessmodelle und Vergleicht die Softwareentwicklungsprozesse miteinander und versucht Stärken und Schwächen der einzelnen Prozessmodelle aufzuzeigen.

2

Anforderungen

In diesem Kapitel werden die Anforderungen formuliert, die an diese Arbeit gestellt werden.

Wir unterteilen die Anforderungen in zwei Teilbereiche: Erstens Anforderungen an die Modellierung und zweitens Anforderungen an den Vergleich.

2.1 Anforderungen an die Modellierung

Die Modellierung der Prozesse soll mit Notationen geschehen, welche eine große Verbreitung haben. Dies ist wichtig, um somindest eine theoretische Chance zu erhalten, dass ein Tool mit guter Eignung sich evtl. zum Standart entwickelt. Des Weiteren soll die Modellierung so einfach dargestellt werden, dass auch ein nicht Informatiker, z.B.

Kürzel	Anforderung
AM1	Tools mit großer Verbreitung
AM2	Frei verwendbare Modellierungen
AM3	Verständlichkeit
AM4	Korrekte Abbildung der Modelle
AM5	Vollständigkeit

Tabelle 2.1: Anforderungen an die Modellierung

ein Manager die Prozesse nachvollziehen und aufgrund der Gegenüberstellung Entscheidungen treffen kann. Dies ist wichtig, da im Geschäftumfeld häufig Personen ohne Informatikausbildung Entscheidungen über das Vorgehen im Betrieb bei der Softwareentwicklung entscheiden, oder somindest für die Bereitstellung der Finanzierung der Projekte verantwortlich sind. Auch soll Wert darauf gelegt werden, dass alle Notationen frei verwendbar sind. Um später bei einer Anwendung einer Notation keine auftretenden Lizenzgebühren zu haben. Würden Lizenzgebühren anfallen, wäre die Notation für einen großen Teil des Potentiellen Anwenderkreises bereits wieder uninteressant.

Weiter ist es wichtig, dass sowohl das Modell korrekt abgebildet wird, als auch die Grundsätze der Notation korrekt verwendet wurden. Würde dies nicht vorausgesetzt werden, würden falsche Fakten modelliert werden.

Dies schließt auch ein, dass die Modelle möglichst vollständig umgesetzt werden sollen. Somit sollte gewährleistet sein, dass die Modellierung eine ganzheitliche und geschlossene Dokumentation des Vorgehensmodells ergibt. Eine vollständige Modellierung ist äußerst wichtig, da nur eine solche eine Entscheidungsgrundlage liefern kann.

Die Anforderungen werden in der Tabelle 2.1 nochmals zusammengefasst.

2.2 Anforderungen an den Vergleich

Es soll die Eignung der Notationen zum Darstellen von iterativen Softwareprozessen untersucht werden. Daher ist darauf zu achten, dass der Prozess in jeder Darstellungsart gleich modelliert wird um ein aussagekräftiges Ergebnis über die Eignung der Notation zu erhalten. Weiter soll verglichen werden, mit welcher Notation das Verständnis des Pro-

Kürzel	Anforderung
AV1	Einheitliche Modellierung in jeder Modellierungsart
AV2	Vergleich der Übersichtlichkeit
AV3	Eignung als Standard
AV4	Ausreichende Syntax und Vorteile gegeneinander

Tabelle 2.2: Anforderungen an den Vergleich

zesses am Besten und am Übersichtlichsten darstellbar ist. Weiter soll geprüft werden, welche Notation sich als Standard zur Prozessdarstellung in der Softwareentwicklung eignen würde.

Somit ist auch festzustellen, ob die verwendeten Notationen ausreichend sind, um die Softwareprozesse darzustellen und wo jeweils die Vorteile der einzelnen Notationen liegen, um somit einen Vorschlag für eine Standartnotation abgeben zu können.

Die Anforderungen werden in Tabelle 2.2 nochmals zusammengefasst.

3

Grundlagen

Dieses Kapitel soll eine kurze Einführung in die verwendeten Technologien, sprich Notationen sowie in die später mithilfe der Diagrammarten dargestellten Softwareentwicklungsprozesse geben.

3.1 Die verwendeten Modellierungen

In diesem Abschnitt werden die verwendeten Notationen kurz erklärt. Wir haben uns aus folgenden Gründen für diese Modellierungsarten entschieden: BPMN 2.0 ist Industriestandard für die Darstellung für Geschäftsprozesse und genießt somit ein hohes Maß an Verbreitung. Die Aktivitätendiagramme stammen aus der UML, welche ebenfalls sehr verbreitet ist und unter Anderem auch im Open UP Prozess verwendet wird. EPKs

verwenden wir, da dies eine der ersten verbreiteten Notationen zum Darstellen von Prozessen war.

3.1.1 BPMN

Der Schwerpunkt der BPMN (Business Process Modelling Notation) liegt auf der Notation, d. h. auf der grafischen Darstellung von Geschäftsprozessen [All09]. In unserem Fall wollen wir die Notation für Softwareprozesse verwenden.

Die BPMN kennt drei Diagrammarten. Wir werden jedoch nur das Kollaborationsdiagramm verwenden.

In der BPMN Notation gibt es Grundlegende Notationstypen, diese sind im Einzelnen :

- Flow Objects (die Knoten (Activity, Gateway und Event)) [All09].
- Connecting Objects (die verbindenden Kanten) [All09].
- Pools und Swimlanes (Bereiche, mit denen Aktoren und Systeme dargestellt werden) [All09].
- Artifacts (Elemente wie Data Objects, Groups und Annotations zur weiteren Dokumentation) [All09].

Flow Objects

Aktivität Aktivitäten beschreiben Aufgaben die im Prozess zu erledigen sind. Als erstes gibt es einfache Aktivitäten sogenannte Tasks.

Bei den Tasks können noch Aufgaben Typen mit angegeben werden. Es gibt unter anderem Manuell, Benutzer, Service, Empfangen und Senden [bpm12].

Weiter gibt es Subprozesse, welche einen eigenen Prozess beinhalten [bpm12].

Speziellere Aktivitäten sind Transaktionen, welche eine Gruppe von Aktivitäten beinhalten, welche logisch zusammengehören, Ereignisse, welche in einem anderen Teilprozess

platziert sind und durch ein Ereignis ausgelöst werden sowie Aufrufaktivitäten welche global definierte Prozesse definieren [bpm12].

Die Aktivitätensymbole sind in Graphik 3.1 dargestellt.

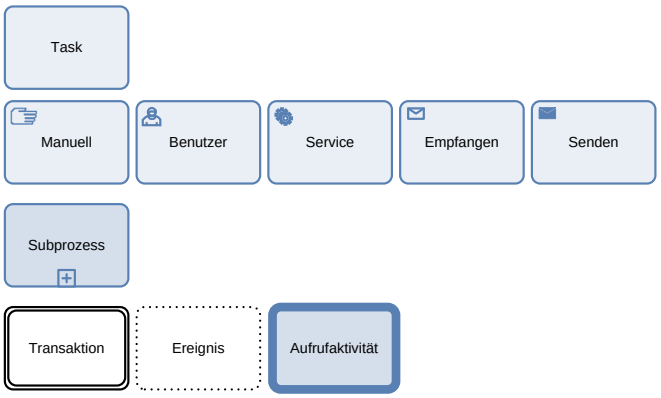

Abbildung 3.1: Aktivitätensymbole in BPMN

Gateways Gateways stellen einen Entscheidungspunkt dar (Split/Fork), oder einen Punkt, an dem verschiedene Kontrollflüsse zusammenlaufen (Join/Merge).

XOR: Bei einer Verzweigung wird der Fluss zu genau einer ausgehenden Kante geleitet. Bei einer Zusammenführung wird auf genau eine eingehende Kante gewartet [bpm12].

AND: Bei einer Verzweigung wird der Fluss zu allen ausgehenden Kanten geleitet. Bei einer Zusammenführung wird auf alle eingehende Kanten gewartet [bpm12].

OR: Je nach Bedingung werden eine oder mehrere Kanten synchronisiert bzw. gestartet [bpm12].

Komplexes Gateway: Alle anderen nicht abgedeckten Anwendungen können mit einem komplexem Gateway modelliert werden, bei dem genaue Verzweigungsbedingungen angegeben werden können. [bpm12].

Die Gatewaysymbole sind in Graphik 3.2 dargestellt.

Ereignisse Ein Ereignis stellt dar, was sich in einem Prozess ereignet.

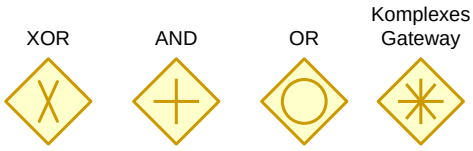

Abbildung 3.2: Gatewaysymbole in BPMN

Ereignisse werden je nach Zeitpunkt im Prozess in Start-, Zwischen- und Endereignisse unterschieden.

Im Inneren des Symboles kann die Art des Ereignisses genauer visualisiert werden. Zum Beispiel gibt es hier Timer und bedingte Ereignisse.

Die Ereignissymbole sind in Graphik 3.3 dargestellt.

Abbildung 3.3: Ereignissymbole in BPMN

Connecting Objects

Flüsse verbinden Activities, Gateways und Events miteinander.

Es gibt drei verschiedene Arten: Der Standardsequenzfluss, der bedingte Fluss, welcher nur ausgeführt wird wenn die Bedingung wahr ist und ein Standardfluss welcher ausgeführt wird wenn alle Anderen nicht zutreffen [bpm12]. Die Flüsse sind in Graphik 3.4 dargestellt.

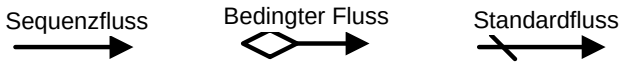

Abbildung 3.4: Flüsse in BPMN

Pools und Swimlanes

Ein Pool repräsentiert einen Benutzer bzw. eine Benutzerrolle oder ein System. Eine Lane unterteilt einen Pool über die komplette Länge [bpm12]. Symbole werden dargestellt in Graphik 3.5.

Abbildung 3.5: Pool und Lane in BPMN

Artifacts

Artefakte sind für sonstige Notationen. Häufig verwendet werden Datenobjekte, welche sowohl elektronische wie auch physische Datenspeicher darstellen können. Dann ist noch der Kommentar zu nennen, mit dem zu einem beliebigen Objekt ein Kommentar angefügt werden kann [bpm12]. Die Symbole werden in Graphik 3.6 dargestellt.

3.1.2 EPKs

EPKs (Erreignisgesteuerte Prozessketten) stellen eine graphische Modellierung für Geschäftsprozesse dar. Wir möchten sie für Softwareentwicklungsprozesse verwenden.

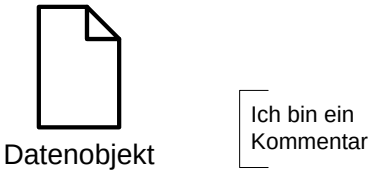

Abbildung 3.6: Artefakte in BPMN

Syntax

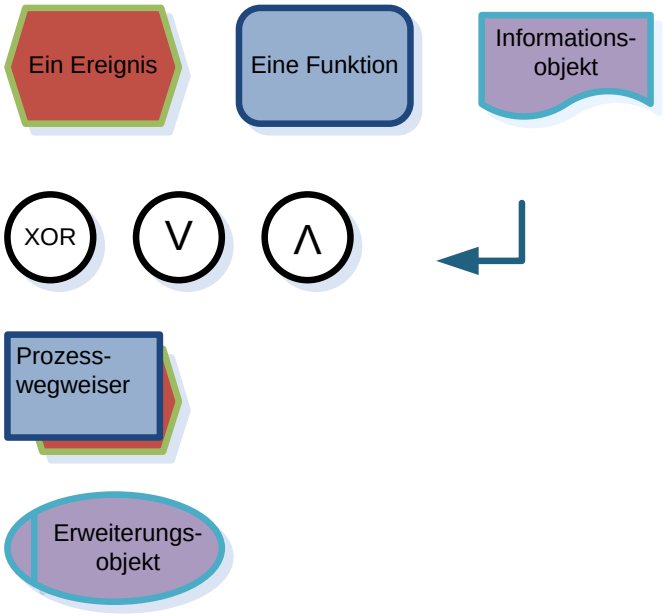

Abbildung 3.7: Syntax von EPKs

Ereignisse Ereignisse steuern den Kontrollfluss im EPK. Das heißt, dass jede Funktion (Funktionen: Erklärung siehe folgender Paragraph) durch ein vorangegangenes Ereignis ausgelöst wird. Ereignisse werden nach der Funktionsausführung erzeugt [Rei13].

Lediglich Endereignisse lösen keine neue Funktion aus, sondern terminieren den Prozess [Rei13].

Alle Ereignisse, außer Start- und Endereignissen, besitzen eine eingehende und eine ausgehende Kante [Rei13].

Die Symbole sind in Abbildung 3.7 zu sehen.

Funktionen Funktionen beschreiben Aufgaben, Tätigkeiten oder Aktivitäten, die in einem Prozess anfallen. Jede Funktion besitzt genau eine eingehende und eine ausgehende Kante. Auf eine Funktion folgt stets ein Ereignis bzw. auf ein Ereignis stets eine Funktion. Es können nicht zwei Funktionen direkt hintereinander stehen [Rei13].

Die Symbole sind in Abbildung 3.7 zu sehen.

Konnektoren Sind EPKs nicht rein sequentiell werden Konnektoren zur Steuerung des Kontrollflusses benötigt. EPKs kennen AND, OR und XOR Verknüpfungen [Rei13].

Alle Eingänge bzw. alle Ausgänge eines Konnektors müssen stets vom selben Typ sein [Rei13]!

Nach einem Ereignis kann kein XOR oder OR Konnektor folgen [Rei13].

Die Syntax ist in Abbildung 3.7 zu sehen.

Prozesswegweiser Mit Prozesswegweisern können Prozesse in Teilprozesse aufgegliedert werden. Der Prozesswegweiser verweist hierbei auf einen anderen Prozess / EPK welcher an dieser Stelle ausgeführt wird [Rei13].

Prozessschnittstellen stehen an Stelle einer Funktion im EPK. Das heißt, in der aufrufenden EPK steht vor der Prozessschnittstelle ein Ereignis. Dieses wird in der aufgerufenen EPK nach dem dortigen Prozessschnittstellensymbol erneut aufgeführt. Rekursionen sind nicht möglich [Rei13].

Die Syntax ist in Abbildung 3.7 zu sehen.

Erweiterungsobjekte Erweiterungsobjekte gehören nicht zu den klassischen EPKs sondern nur zu den erweiterten EPKs. Erweiterte EPKs sind erreignisgesteuerte Prozessketten bei denen es noch weitere Symbole zur Darstellung von speziellem Verhalten

gibt. Durch Erweiterungsobjekte ist es möglich, dem Prozess Rollen / Benutzer hinzuzufügen und werden direkt mit Funktionen verbunden [Rei13].

Die Syntax ist in Abbildung 3.7 zu sehen.

3.1.3 UML-Aktivitätendiagramm

Das Aktivitätendiagramm ist ein Verhaltensdiagramm welches zur UML 2.0 gehört. Der Tokenfluss (Ein Token zeigt an in welcher Stelle der Prozess gerade in der Ausführung steht) bildet die Grundlage für die Interpretation des Diagrammes. Alle Aktionen in einem Aktivitätsdiagramm beschreiben gemeinsam eine Aktivität [hig15].

Ein Aktivitätendiagramm hat zwei Arten von Bausteinen: Knoten und Kanten. Knoten sind die Stellen, an denen Entscheidungen getroffen werden und Aktivitäten ausgeführt werden. Kanten sind die Verbindungslinien zwischen Knoten. Über diese Verbindungslinien fließen sogenannte Token [hig15].

Token zeigen an, an welcher Stelle die Ausführung des Prozesses im Moment steht [hig15].

Syntax

Start- und Endknoten Hier starten und enden Aktivitäten. Das heißt, im Startknoten wird ein Token erzeugt. In einem Endknoten der Token zerstört [hig15].

Es kann in einem Diagramm grundsätzlich mehrere Start- und Endknoten geben. Mehrere Endnoten sind unproblematisch. Mehrere Startknoten sind jedoch problematisch, da sich mehrere Token zeitgleich im Diagramm bewegen [hig15].

Sobald ein Token einen Endknoten erreicht, ist die Aktivität beendet, auch wenn noch weitere Token unterwegs sind [hig15].

Syntax dargestellt in Graphik 3.8.

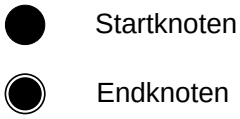

Abbildung 3.8: UML Start und Endknoten

Aktionen und Objektknoten Aktionen stellen die einzelnen Schritte eines Prozesses dar und werden mit einem Rechteck mit abgerundeten Ecken dargestellt. Innerhalb des Rechteckes soll mit einer kurzen Beschreibung die Aktion dargestellt werden [hig15].

Ein Rechteck ohne abgerundete Ecken hingegen stellt einen Objektknoten dar. Objektknoten sind Datenspeicher. Wird ein Objektknoten zwischen zwei Aktionen im Tokenfluss dargestellt, so werden Daten von einer Aktivität in die andere transportiert [hig15].

Syntax dargestellt in Graphik 3.9.

Abbildung 3.9: UML Aktions- und Objektknoten

Verzweigungen Entscheidungsknoten werden durch Rauten dargestellt. An deren ausgehenden Kanten können Bedingungen angegeben werden. Um die verzweigten Flüsse zusammen zu führen wird ebenfalls die Raute als Symbol verwendet [hig15].

Um eine parallele Aufspaltung zu erreichen, wird ein Balkensymbol verwendet. Hier fließt im Eingang ein Token rein und an allen Ausgängen einer raus. Um die Flüsse zusammen zu führen wird ebenfalls der Balken verwendet, nur dass hier an jedem Eingang ein Token angekommen sein muss, damit am Ausgang ein Token ausgegeben werden kann [hig15].

Symbole dargestellt in Graphik 3.10.

Abbildung 3.10: UML Verzweigungen

Verbindungslinien Verbindungslinien werden in Form eines Pfeiles angegeben [hig15].

Darstellung von Rollen

Rollen können über ein Rechteckschema, welches für jede Rolle einen extra Bereich kennzeichnet, dargestellt werden.

Symbole dargestellt in Graphik 3.11.

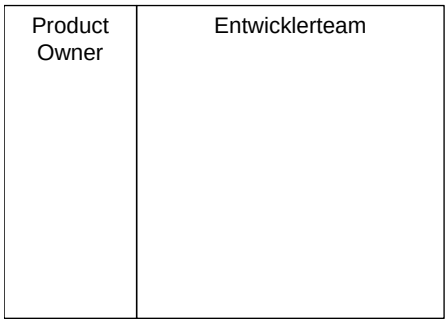

Abbildung 3.11: UML Darstellung von Rollen

3.2 Ausgewählte Softwareentwicklungsprozesse

Prozessmodelle im Softwareengineering lassen sich in verschiedene Kategorien einordnen. Erstens gibt es die linearen Modelle welche noch in Aktivitätengesteuerte- und Phasengesteuertemodelle unterteilt werden. Zweitens gibt es nicht Lineare Modelle und drittens noch sonstige Modelle wie z.B. die Formalen Methoden. Wir beschäftigen uns jedoch im folgenden ausschließlich mit nicht linearen Prozessmodellen. Wir werden Open UP verwenden, welches eine Spezialisierung des RUP darstellt. Weiter betrachten wir extreme Programming und Scrum welche zu den Agilen Methoden gehören. All diese Prozessmodelle werden auch Iterative Prozessmodelle genannt. Eine schematische Einordnung ist in Abbildung 3.12 zu finden.

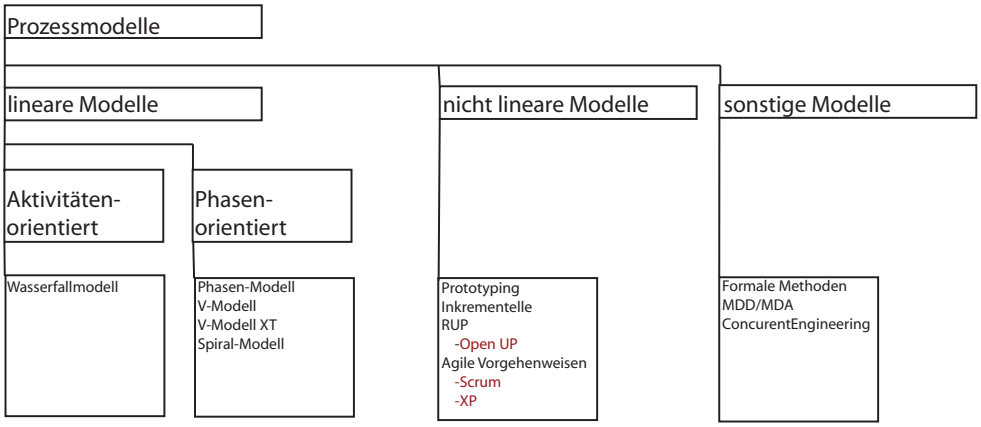

Abbildung 3.12: Überblick über verschiedene Prozessmodelle des Softwareengineering und Einordnung der Verwendeten.

Kapitel 3.1 wird nun Scrum behandeln, 3.2 XP und 3.3 den Open UP Prozess.

3.2.1 Scrum

Scrum ist ein empirisches, inkrementelles und iteratives Prozessmodell der Softwareentwicklung. Das heißt der Prozess wird in einzelne Teilaufgaben, welche sich wiederholen aufgeteilt. In Scrum wird versucht die Komplexität von Softwareprojekten mit folgenden Säulen zu meistern:

- Transparenz: Dies bedeutet, dass alle Ergebnisse für alle am Prozess beteiligten sichtbar sein müssen und alles nach einem gemeinsamen Standard erstellt wird, um Verständnis für die Ergebnisse der anderen am Prozess beteiligten aufbringen zu können. Dies umfasst unter anderem auch eine gemeinsame Prozesssprache [SS13].

- Überprüfung: Scrum Artefakte und deren Fortschritt müssen durch die Scrum Anwender in definierten Intervallen auf ihren Stand im Bezug zum Sprint-ziel überprüft werden [SS13].

- Anpassung: Dies folgt unmittelbar aus der Säule der Überprüfung. Werden Abweichungen zum Ziel erkannt, so muss schnellstmöglich entweder der Prozess, oder die zu bearbeitenden Aufgaben angepasst werden um weitere Abweichungen zu verhindern [SS13].

Für die Überprüfung und Anpassung stellt Scrum im wesentlichen Sprint Planning, Daily Scrum, Sprint Review und Sprint Retrospektive als Instrumente zur Verfügung, welche später noch genauer beschrieben werden [SS13].

Das Ziel von Scrum ist es, dabei möglichst schnell ein qualitativ hochwertiges Produkt entstehen zu lassen welches vorab formulierten Zielen entspricht. Jedoch soll dies hier nicht durch ein Pflichtenheft sondern durch das Ausformulieren von Zielen, welche in sich wiederholenden Sprints realisiert werden, erreicht werden. Am Ende eines solchen Sprints soll als Ergebnis jeweils eine fertige Softwarefunktionalität vorliegen (das sogenannte Produkt-Inkrement) [Glo11].

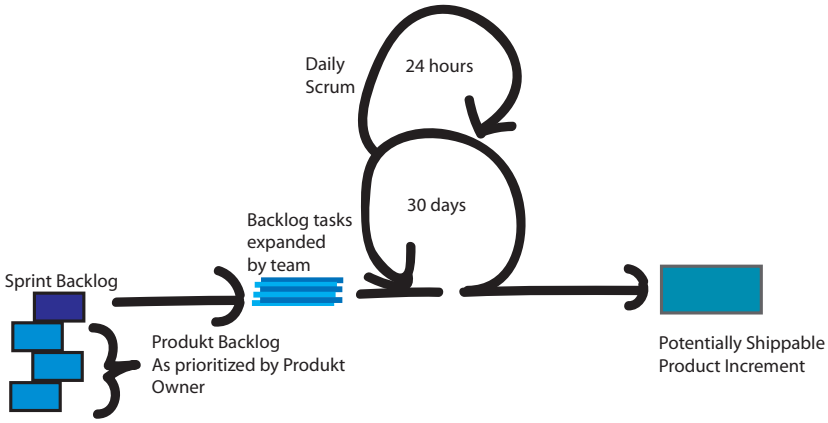

Abbildung 3.13: Der Scrumprozess und dessen Bestandteile. In Anlehnung an [SB08].

Rollen im Scrum

Das Scrum Team, welches selbstorganisierend und interdisziplinär ist, besteht aus Produkt Owner, dem Entwicklungsteam sowie dem Scrum Master [SS13].

Im einzelnen gibt es folgende Rollen:

Product Owner Der Produkt Owner ist für das Management des Product Backlogs (siehe Kapitel Artefakte im Scrum) verantwortlich, das heißt, er muss Produkt Backlog Einträge klar formulieren und dafür sorgen, dass dem Entwicklungsteam klar ist, an welchen Funktionalitäten sie als nächstes arbeiten sollen. Jedoch ist es im Scrum nicht festgelegt ob er diese Aufgaben selbst erledigt oder diese dem Team überlässt. Er ist aber stets rechenschaftspflichtig. Sehr wichtig ist der Fakt, dass es dem Entwicklungsteam nicht gestattet ist, anderen Angaben als denen des Produkt Owner zu folgen. [SS13]

Da der Produkt Owner somit gewissermaßen der oberste Produktentwickler ist, gehört es ebenfalls zu seinen Aufgaben, alle Rücksprachen mit dem Kunden zu treffen. Dies beinhaltet auch, dass er dafür sorgt, dass die Wünsche und Bedürfnisse des Kunden in

die Entwicklung einfließen. Jedoch ist er kein Vertreter des Kunden, denn seine oberste Aufgabe ist die Wirtschaftlichkeit des Produktes.

Die Praxis zeigt leider, dass der Produkt Owner oft nicht die benötigten Kompetenzen für seine Entscheidungen besitzt, bzw. mit anderen Aufgaben überlastet ist [Pic09].

Entwicklungs-Team Die Mitglieder des Entwicklungsteams erstellen das Produkt Inkrement.

Die Entwicklungsteams organisieren ihre Arbeit selbst, das heißt, von außen darf niemand auch nicht der Scrum Master sagen wie Sie denn vorgehen sollen um das Produkt Inkrement zu erreichen [SS13].

Das Team sollte so besetzt sein, dass es als gesamtes Team alle benötigten Fähigkeiten besitzt [SS13].

Wichtig ist, dass im Team alle gleichberechtigt sind und das Team als ganzes rechenschaftspflichtig ist [SS13].

Scrum Master Der Scrum Master sorgt dafür, dass das Team die Theorie, Praktiken und Regeln von Scrum einhält. [SS13]

Der Scrum Master ist gegenüber dem Entwicklungsteam eine Führungskraft, jedoch kein Vorgesetzter. Seine Rolle ist die eines Servant Leaders – der Scrum Master sichert sich Anerkennung und Gefolgschaft, indem er sich den Bedürfnissen der Team-Mitglieder annimmt [Pic09].

Ereignisse / Ablauf von Scrum

Ereignisse im Scrum sollen für Regelmäßigkeit sorgen. Sämtliche Ereignisse des Scrum sind zeitlich begrenzt und dürfen weder verkürzt noch verlängert werden. Außer dem Sprint ist jedes Ereignis ein formaler Punkt zur Überprüfung und Anpassung [SS13].

Sprint Planning Das Sprint Planning Meeting dient dazu, die Ziele des nachfolgenden Sprints festzulegen. Am Sprint Planning nimmt das komplette Scrum Team teil. Für das Meeting wird ein Zeitrahmen von max. 8 Stunden für einen einmonatigen Sprint festgelegt. Der Scrum Master ist für die Einhaltung des Zeitrahmens verantwortlich. Als Ergebnis des Meetings sollen die Sprintziele, sprich was für ein Produkt Inkrement und wie dies erreicht werden soll, festgelegt werden [SS13].

Sprint Ein Sprint sollte maximal 4 Wochen dauern. Innerhalb dieser Zeitbox soll ein neues Produkt Inkrement fertiggestellt werden. Während eines Sprints sollen keine, das Sprintziel gefährdende Änderungen vorgenommen und auch das Qualitätsziel nicht verändert werden. Der Produkt Owner kann einen Sprint abbrechen, mögliche Gründe hierfür sind zum Bsp. eine Änderung der technologischen Rahmenbedingungen, oder auch, dass der Sprint unter den gegebenen Umständen sinnlos ist. Bei einem Abbruch des Sprints muss der Produkt Backlog aktualisiert werden [SS13].

Daily Scrum Der Daily Scrum ist ein tägliches kurzes Treffen (ca 15 min.) welches möglichst immer zur selben Zeit am selben Ort stattfinden sollte. Es dient dazu, Informationen auszutauschen. Wichtig ist, dass hier keine Probleme gelöst werden. Im Regelfall gibt jedes Teammitglied kurz Auskunft, was seit dem letzten Daily Scrum erreicht wurde und was bis zum nächsten erreicht werden soll. Teilnehmen muss das Entwicklungsteam. Der Scrum Master und der Produkt Owner können teilnehmen, verhalten sich aber meist passiv [SS13]. Sollte es zu Fragen kommen die nicht innerhalb von 15 Minuten lösbar sind, werden diese in einem separaten Meeting geklärt für das der Scrum Master zuständig ist [Pic09].

Sprint Review Am Ende eines jeden Sprints organisiert der Scrum Master ein Sprint Review welches ca. eine Stunde je Sprint Woche dauert, sprich bei einem einmonatigen Sprint ca. 4 Stunden. Teilnehmen darf das Scrum Team sowie die Stakeholder. Das Sprint Review ist als informelles Meeting gedacht. Unter anderem wird auch das Produktinkrement vorgeführt was zu einem Feedback führen soll. Als Ergebnis steht

ein überarbeiteter Produktbacklog zur Verfügung, welcher die Backlogeinträge für kommende Sprints enthält. Es ist durchaus möglich, dass der Produkt Backlog umfassend umgearbeitet wird [SS13].

Sprint Retrospektive Die Sprint Retrospektive findet zwischen dem Sprint Review und dem Sprint Planning statt und dauert bei einmonatigen Sprints ca. 3 Stunden. Teilnehmen dürfen das Entwicklungsteam und der Scrum Master, weitere Teilnehmer nur auf Einladung. Es dient zur Reflektion des Verlaufes des Sprints im Bezug auf die beteiligten Personen, Beziehungen, Prozesse und Werkzeuge. Das Ziel ist daraus abgeleitet Verbesserungen zu erzielen [SS13].

Artefakte in Scrum

Die Artefakte im Scrum wurden so entworfen dass sie eine möglichst hohe Transparenz bezüglich der enthaltenen Informationen haben. Sie dienen dazu, Arbeit oder Wert zu dokumentieren.

Produkt Backlog Der Produkt Backlog ist eine priorisierte Liste, welche alle Eigenschaften / Funktionalitäten enthält, die im fertigen Produkt enthalten sein sollen. Der Produkt Backlog wird vom Produkt Owner verwaltet. Wichtig zu wissen ist, dass der Produkt Backlog nie vollständig ist. Er wird im Laufe des iterativen Prozesses immer wieder ergänzt. Im Product Backlog werden alle Features, Funktionalitäten, Verbesserungen und Fehlerbehebungen aufgelistet, die die Änderungen an dem Produkt in zukünftigen Versionen ausmachen. Wenn mehrere Scrum Teams an einem Projekt arbeiten, gibt es trotzdem nur einen Produktbacklog für alle Teams [SS13].

Sprint Backlog Das Sprint Backlog ist die Menge der für den Sprint ausgewählten Produkt Backlog-Einträge, ergänzt um den Plan für die Lieferung des Produkt-Inkrements sowie zur Erfüllung des Sprint-Ziels. Der Sprint Backlog muss ausreichend detailliert sein um im täglichen Daily Scrum den Stand der Arbeit sinnvoll erkennen und beurteilen zu können. Nur das Entwicklungsteam aktualisiert den Sprint Backlog.

Produkt Inkrement Als Inkrement werden die Ergebnisse eines Sprints bestehend aus den realisierten Backlogeinträgen bezeichnet. Diese müssen in einem Auslieferungsbereiten Zustand sein [SS13].

3.2.2 Xtreeme Programming (XP)

Extreme Programming (XP) ist ein agiler Softwareentwicklungsprozess für kleine Teams. XP versucht vage und sich rasch ändernde Anforderungen zu unterstützen und somit dem Kunden ab dem ersten Tag Geschäftswert zu produzieren [Wes 8].

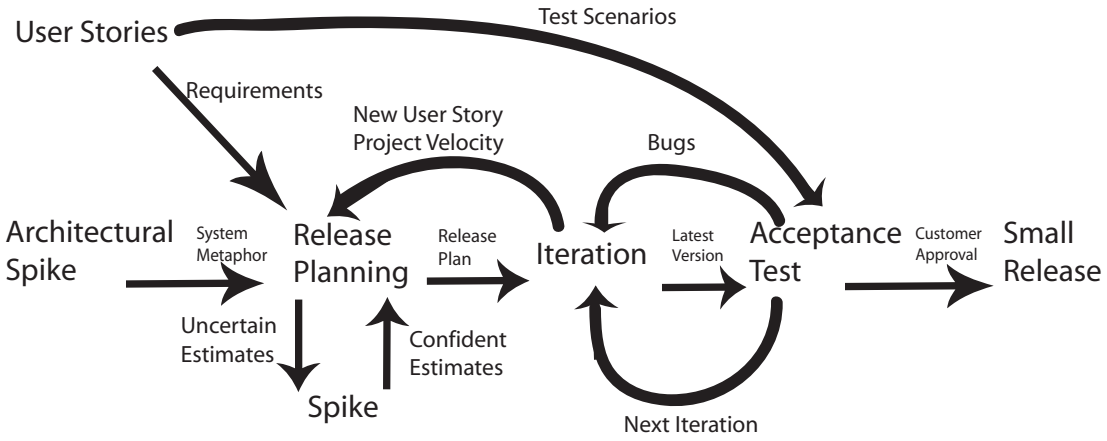

Abbildung 3.14: Xtreeme Programming ein Überblick. In Anlehnung an [Hor23].

Wichtige Werte im XP Prozess sind: Kommunikation, Einfachheit, Feedback, Mut, Lernen, Qualität und Respekt [Wes 8] [Scr10].

Praktiken bei XP Folgende Praktiken stellen die Grundlage von XP dar [Wes 8].

- Pair Programming: Es wird immer zu Zweit an einem Arbeitsplatz gearbeitet.

- Kollektives Eigentum: Aktivitäten werden zunächst nicht an einzelne Personen verteilt, sondern an das ganze Team. Dies soll dafür sorgen, dass spezielles Wissen nicht von einer einzelnen Person abhängig ist.
- Kontinuierliche Integration: Entwickelte Komponenten werden frühestmöglich in das Gesamtsystem integriert.
- Tests: Es wird so früh und so viel wie möglich getestet.
- Kundeneinbeziehung: Der Kunde soll häufig integriert sein um in die richtige Richtung zu entwickeln.

Rollen in XP

Ein XP-Team wird aus zwei bis etwa zwölf Programmierern gebildet, weiter gibt es einen Kunden oder mehrere direkte Ansprechpartner auf Kundenseite und dem Management [Wes 8].

Produkt Owner / Produkt Besitzer Der Produkt Owner setzt die Prioritäten der Entwicklung. Er ist dafür verantwortlich, dass der bestmögliche Gewinn erzielt wird. Product-Owner im Sinne von XP kann beispielsweise ein Vertreter des Produktmanagements, ein Kunde oder ein Nutzer des Produktes sein.

Kunde Der Kunde ist der Auftraggeber des Projektes und entscheidet was gemacht wird. Es ist möglich, dass der Kunde und der Produkt Owner dieselbe Person ist. Weiter sollte der Kunde regelmäßiges Feedback über die Entwicklungen geben.

Entwicklungsteam Das Entwicklungsteam besteht aus 2-12 Personen. Darunter sind Programmierer, Datenbankexperten usw. Häufig wird nach dem Prinzip des Pair Programming gearbeitet. Dies bedeutet, dass sich zwei Entwickler einen Computer teilen und gemeinsam arbeiten [Wes 8].

Projektmanager Der Projektmanager führt das Team, meistens ist der Projektmanager und der Produkt Owner dieselbe Person.

Benutzer Der Benutzer verwendet später das fertige Produkt.

Ereignisse Ablauf von XP

In diesem Abschnitt werden nun die Ereignisse sowie der Ablauf des Prozesses im XP beschrieben.

Planung In der Planung wird ein Releaseplan aufgestellt. Ein Release beinhaltet die Funktionen, die insgesamt und für sich geschlossen die Bereitstellung einer neuen Version des Produktes rechtfertigen. Der Releaseplan legt die Anzahl und Dauer der einzelnen Iterationen fest die benötigt werden um zu einem fertigen Release des Projektes zu kommen (Abbildung: 3.15) [Wes 8].

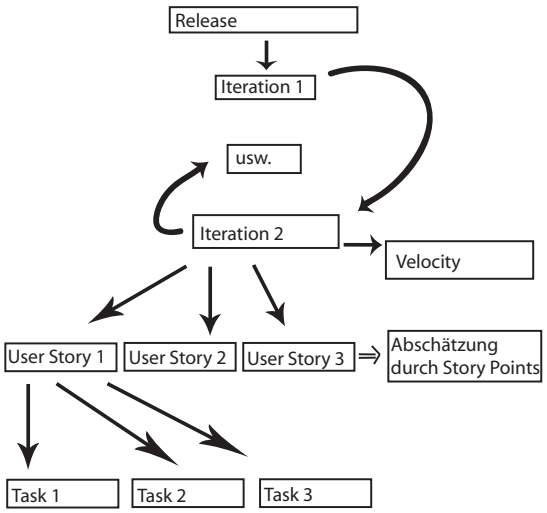

Abbildung 3.15: Xtreeme Programming Umsetzung einer Release. In Anlehnung an [wik15b].

User stories Die innerhalb der Iterationen umzusetzenden Entwicklungen müssen näher definiert werden. Hierfür werden mit dem Kunden sogenannte User Stories erstellt [Scr10]. Bei der Erstellung der User Stories sollte das komplette Entwicklerteam mitarbeiten. Den einzelnen User Stories werden Prioritäten zugewiesen. Ausschlaggebende Bedingungen hierfür sind das Risiko bezüglich Zeitplan, Kosten oder Funktionalität und der gebotene Mehrwert für das Produkt (Abbildung: 3.16). Ausserdem erfolgt eine Abschätzung des Aufwandes der User Stories.

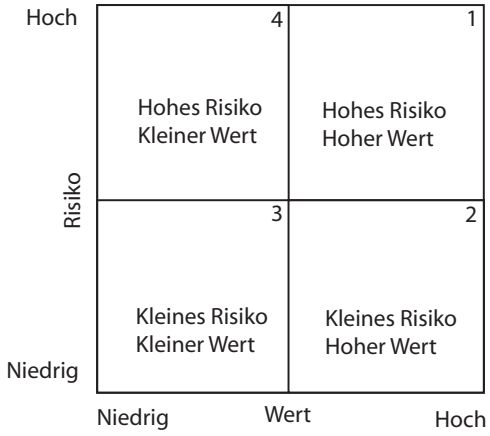

Abbildung 3.16: Xtreeme Programming Schema für Prioritäten. In Anlehnung an [wik15b].

Umsetzung der User Stories Sind die User Stories festgelegt, müssen sie umgesetzt werden. Hierbei wird in der Reihenfolge wie in Abbildung 3.16 dargestellt vorgegangen. Da es am risikoreichstem für das Projekt ist wenn bei User Stories mt hohem Wert Probleme auftreten.

Ein erster Schritt der Umsetzung ist das Zerlegen der User Stories in einzelne Tasks. Jeder Task erhält eine eigene Aufwandsabschätzung. Die Entwickler nehmen sich nach und nach einzelne Tasks vor. Die User Stories sind fertig implementiert, wenn alle Tasks umgesetzt und alle Tests abgeschlossen sind.

Ablauf der Entwicklung Jeder Arbeitstag beginnt mit einem kurzen Stand Up Meeting, in dem jeder Entwickler kurz über die Arbeit des Vortages spricht [Wes 8]. Es werden die Arbeitspaare für das Pair Programming gebildet. Während des Tages finden jederzeit sogenannte Pair-Negotiations statt (Wissensaustausch zwischen einzelnen Entwicklerpaaren) (Siehe Abbildung: 3.17).

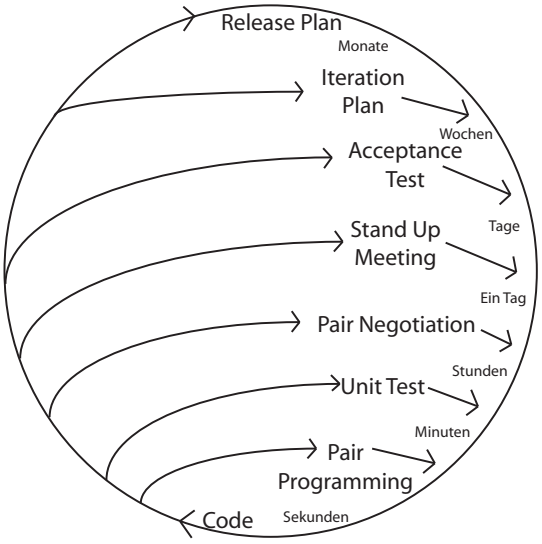

Abbildung 3.17: Xtreeme Programming Iterationen im Prozess. In Anlehnung an [wik15b].

Scheitern einer User Story Wenn eine User-Story in einer Iteration nicht abgeschlossen werden kann, zum Beispiel weil die Tests nicht erfolgreich waren oder sich die Abschätzung als zu knapp beziehungsweise der Umfang als zu groß herausgestellt hat, wird diese entweder komplett in die nächste Iteration verschoben oder in mehrere kleine aufgeteilt.

Akzeptanz Test Am Ende einer jeden User Story steht der sogenannte Akzeptanz Test, welcher von Vertretern des Managements und des Kunden durchgeführt wird. Hierbei können neue Prioritäten oder Ideen eingebracht werden [Wes 8].

Artefakte in XP

In XP wird im Gegensatz zu anderen Vorgehensweisen oft auf Technik und Dokumentation verzichtet. Es wird oft darauf gesetzt, dass sich das Wissen in den Köpfen der Entwickler befindet [Scr10].

User Stories Kurze und formlose Beschreibung der zu entwickelnden Funktionalität. Die Erstellung erfolgt durch das Entwicklungsteam in Zusammenarbeit mit dem Kunden.

Tasks Unterteilung der User Stories in kleine händelbare Arbeitspakete.

Aufwandsabschätzung Eine Abschätzung des Arbeitsaufwandes erfolgt mittels sogenannter Story Points welche im Laufe des Prozesses immer wieder verfeinert werden können.

3.2.3 Open UP

Der Open Unified Process (Open UP) ist ein Open Source-Softwareentwicklungsprozess, welcher an den Rational Unified Process (RUP) angelehnt ist und von der Eclipse Foundation entwickelt wird. Der grobe schematische Ablauf wird in Graphik 3.18 dargestellt [EF 5].

Bei Open UP wird das Projekt in mehrere Iterationen aufgeteilt. Ein Projektplan gibt an was innerhalb einer Iteration erreicht werden soll. Die Entwicklerteams organisieren sich selbst. Sie arbeiten nach und nach kleine Teilgebiete des Iterationsziels ab [EF 5].

Das gesamte Projekt wird hierbei in 4 Phasen gegliedert. Diese sind: Inception (Konzeptionsphase), Elaboration (Entwurfsphase), Construction (Konstruktionsphase), und Transition (Übergabephase).

Im kommenden werden nun die einzelnen Rollen des Open UP beschrieben werden.

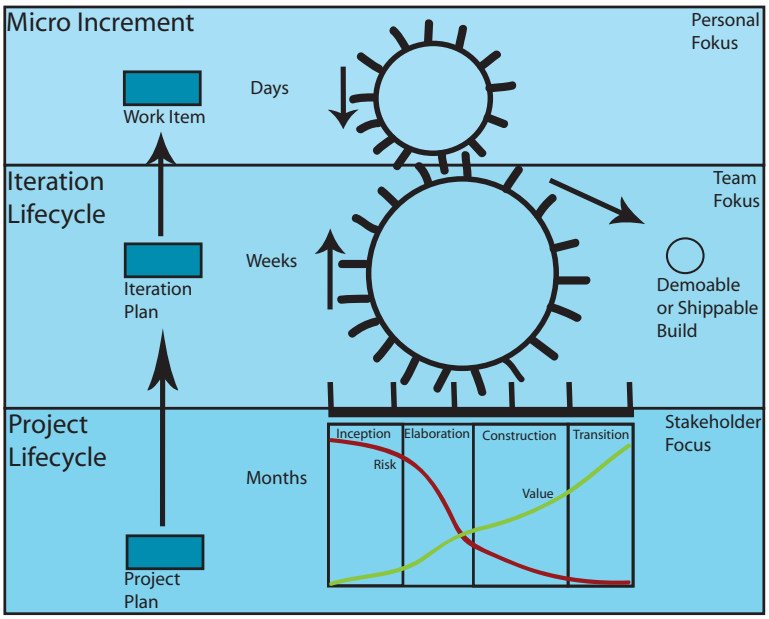

Abbildung 3.18: Der Ablauf von Open UP Bildquelle. In Anlehnung an [EF 5].

Rollen bei Open UP

Rollen, die im Open UP definiert sind, sind dafür zuständig, Aktivitäten durchzuführen, die im Open UP detailliert beschrieben sind. Innerhalb des Teams gibt es Spezialisten für die einzelnen Aufgaben.

Entwicklerteam Das Entwicklerteam führt die einzelnen Aufgaben aus. Hierbei gibt es für jeden Fachbereich der in jeder Iteration durchlaufen wird Spezialisten.

Ereignisse Ablauf bei Open UP

Der Open UP Prozess besteht aus 4 Phasen welche nacheinander durchlaufen werden, wie in Abbildung 3.19 illustriert.

Phase 1: Inception In der ersten Phase soll eine Vision ausformuliert werden, welche ein klares Ziel sowie die Erstellung eines rudimentären Anwendungsfallmodelles

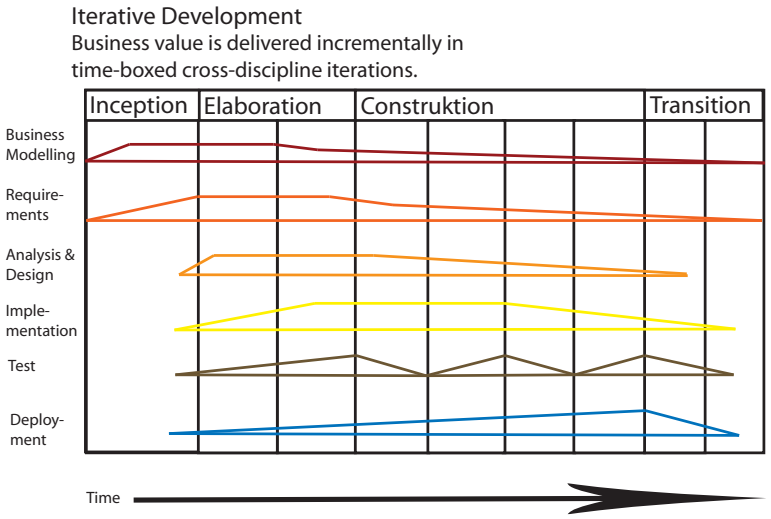

Abbildung 3.19: die Phasen von Open UP. In Anlehnung an [wik15a].

beinhaltet. Dieses soll die wesentliche Funktionalität und eine provisorische Architektur beschreiben. Ebenfalls sollen die wesentlichsten Risiken identifiziert und die Ausarbeitungsphase geplant werden. Als Ergebnis entsteht der Lifecycle Objective Milestone welcher die erstellten Inhalte dieser Phase beinhaltet. [EF 5].

Phase 2: Elaboration Es wird ein Architekturprototyp und ein Großteil der Anwendungsfälle erstellt. Diese Phase endet mit dem Lifecycle Architecture Milestone [EF 5]. Ebenfalls wird die kommende Phase geplant.

Phase 3: Construction Danach wird die Software erstellt und getestet bis eine lauffähige Version entstanden ist. Das Ergebnis ist der Operational Capability Milestone [EF 5].

Phase 4: Transition Jetzt wird das Produkt dem Kunden übergeben. Abgeschlossen wird diese Phase mit dem Product Release Milestone [EF 5].

Jede der Vier Phasen läuft in Iterationen ab, welche die Grundtechniken Business Modelling, Requirements, Analysis und Design, Implementation, Test und Deployment beinhaltet, welche je nach Phase unterschiedlich ausgeprägt sind (Siehe Abbildung 3.19).

Artefakte bei Open UP

Im folgenden werden nun Artefakte des Open UP beschrieben.

Meilensteine Die einzelnen Phasen werden immer durch Meilensteine abgeschlossen.

Projektplan Der Projektplan regelt, wie das Projekt aufgeteilt wird und wann was erledigt werden soll.

Iteration Plan Beschreibt, was in einer einzelnen Iteration umgesetzt werden soll.

Work Item Ein einzelnes Arbeitspaket welches ein Entwickler herausgreift und innerhalb der Iteration in kurzer Zeit umsetzt.

4

Darstellung der Softwarentwicklungsprozesse

In diesem Kapitel steckt der Kern dieser Arbeit. Hier werden die einzelnen Softwareentwicklungsprozesse mittels UML-Aktivitätendiagrammen, EPKs sowie der BPMN Notation dargestellt.

4.1 Scrum

In diesem Abschnitt werden wir Scrum mittels den verschiedenen Notationen darstellen.

4.1.1 Scrum mit UML-Aktivitätendiagramm modelliert

Im Folgenden wird der Scrum Prozess mittels UML Aktivitätendiagramm modelliert. Hierbei wird der Prozess in einen Mainprozess (Abbildung 4.1) und einen Subprozess aufgeteilt. Der Subprozess beschreibt den Sprint genauer. Dieser wird in Abbildung 4.2 dargestellt. Diese Aufteilung wurde so vorgenommen, da der Sprint sehr gut als eigenständiger Teil des Ganzen betrachtet werden kann, da dieser ein in sich geschlossener Teilprozess ist.

Um Rollen darzustellen wird das Konzept der Lanes verwendet. Sind mehrere Rollen an einem Teilprozess beteiligt so wird dies mittels Parallelisierung dargestellt. Das heißt, wir verwenden eine UND Verknüpfung und stellen dann in der Lane von jeder Rolle die Aktivität parallel dar. Anschließend werden die einzelnen Pfade wieder mit einer Vereinigung zusammengefasst. Diese Möglichkeit wurde gewählt da dies Die einzige Möglichkeit der Aktivitätendiagramme ist. Eine Alternative hierzu wäre gewesen eine Erweiterung einzuführen mit dieser das modelliert werden kann.

Ein Augenmerk wurde auch darauf gelegt wie denn die freiwillige Teilnahme des Scrum Masters und des Produkt Owners am Daily Scrum modelliert werden kann. Wir haben uns hierbei ebenfalls für eine Parallelisierung entschieden. In den parallelen Zweigen wird dann durch eine ODER Verknüpfung die Entscheidung dargestellt ob es einen Teilnahmewunsch der einzelnen Personen gibt oder nicht. Gibt es keinen Teilnahmewunsch so passiert weiter nichts. Gibt es einen so nehmen die Personen am Daily Scrum teil. Eine Alternative Hierzu wäre gewesen dies über einen Kommentar zu verdeutlichen. Wir haben uns jedoch explizit für die erste Variante entschieden da die Entscheidung im Scrum-Prozess deutlich dargestellt ist und wir diesen Entscheidungsvorgang direkt abbilden wollten.

Der Scrum Master ist für die Überwachung des Zeitrahmens des Sprint Planning Meetings verantwortlich. Dies wurde nur durch eine Aktivität modelliert. Man hätte überlegen können, dies durch ein Zeitereignis oder ähnliches zu modellieren. Da aber die Angaben, wie genau dies in der Praxis geschieht recht schwammig sind, habe ich mich gegen diese Lösung entschieden und dies als einfache Aktivität notiert. Was den Vorteil hat

dass kein genauer Zeitraum in der Notation festgelegt werden muss und das ganze nach dem Empfinden des Scrum Masters gestaltet werden kann.

Es wurde versucht, die wichtigsten Datenelemente zu modellieren. Leider waren wir hierbei an die Notation der UML, welche Daten im Kontrollfluss modelliert, gebunden.

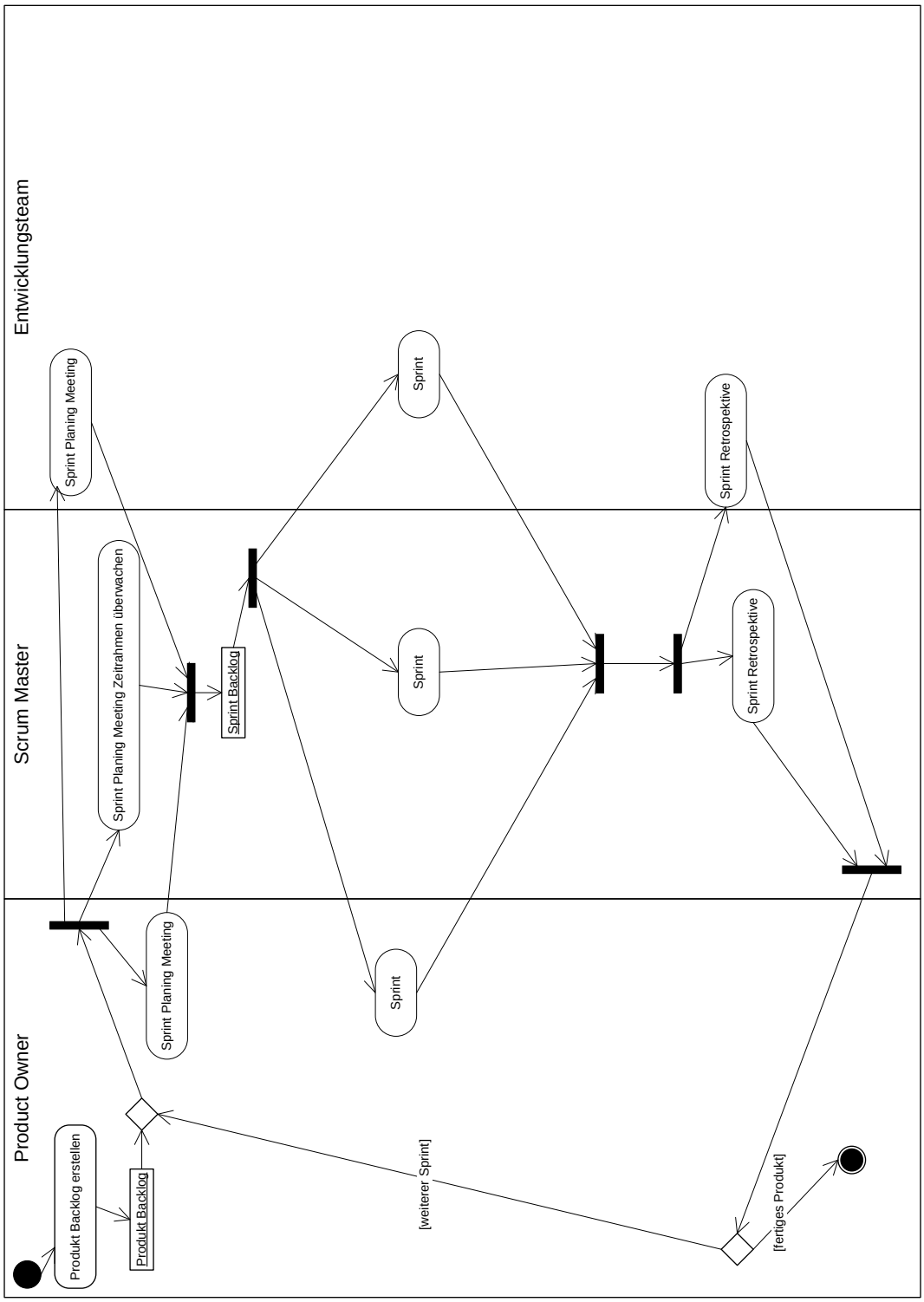

Abbildung 4.1: Scrum mit UML Aktivitätendiagramm modelliert - Main Prozess

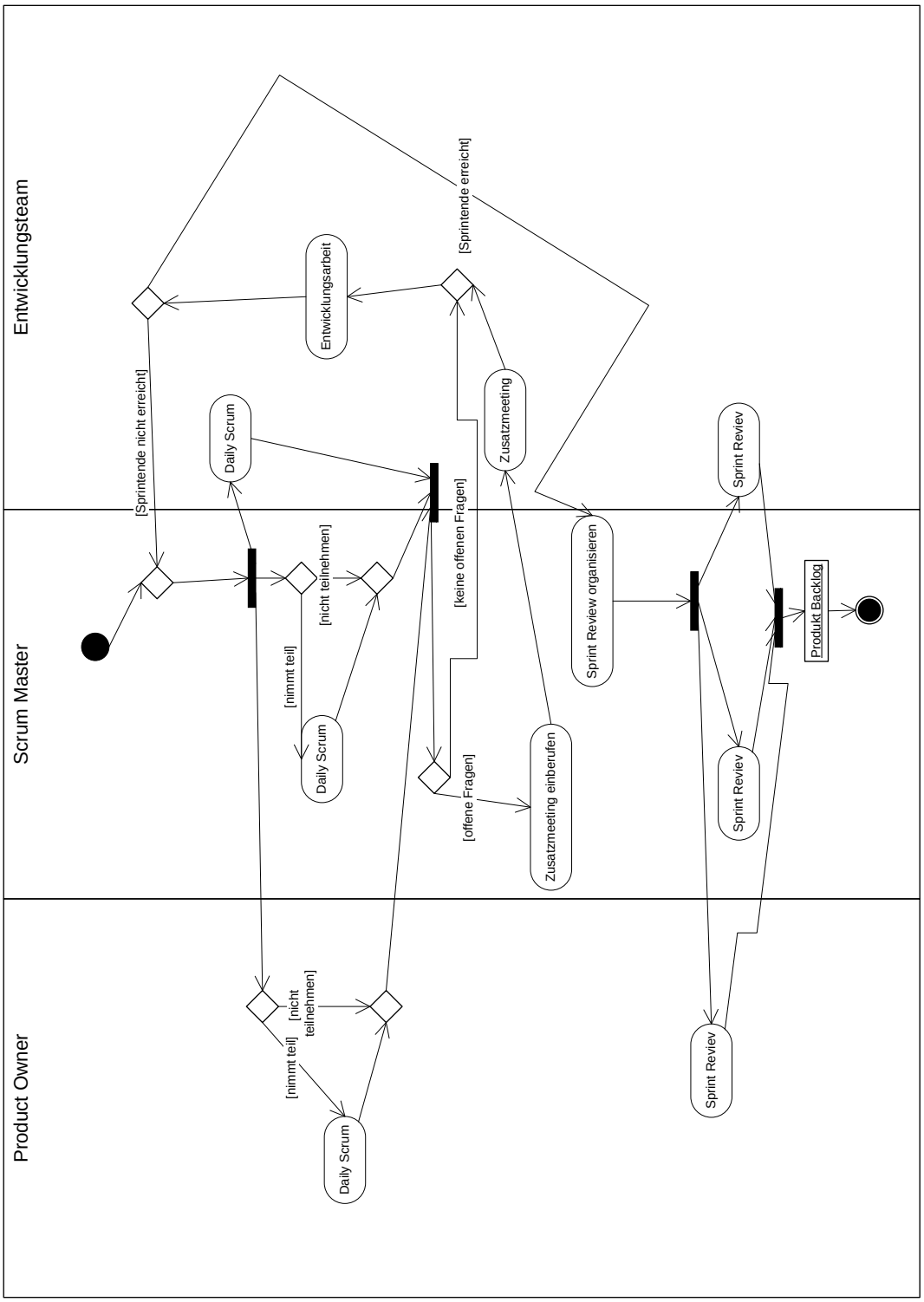

Abbildung 4.2: Scrum mit UML Aktivitätendiagramm modelliert - Sprint Prozess

4.1.2 Scrum mit EPKs modelliert

Um Scrum mit EPKs darzustellen teilen wir den Prozess ebenfalls auf. Zuerst wird ein Mainprozess siehe Abbildung 4.3 modelliert. Dieser beschreibt jedoch noch nicht die im Sprint stattfindenden Aktivitäten.

Um den Sprint genauer zu beschreiben, wird ein separater Subprozess modelliert, welcher in Abbildung 4.4 zu sehen ist. Diese Aufteilung bietet sich an, da der Sprint als eigenständiger Prozessteil gesehen werden kann, welcher sich sehr gut heraus trennen lässt.

Für die Darstellung der Benutzerrollen verwenden wir Elemente aus den erweiterten EPKs. Erweiterte EPKs umfassen alle Elemente der EPKs mit noch zusätzlichen Erweiterungen, beispielsweise zur Darstellung von Benutzerrollen. Somit kann bei den EPKs auf eine Parallelisierung verzichtet werden, wenn eine Aktivität von mehreren Benutzern ausgeführt werden soll, da bei jeder Aktivität notiert werden kann welche Rollen an dieser beteiligt sind.

Um darzustellen dass eine Rolle teilnehmen kann, dies aber nicht muss, kann das einfach in der Rollenzuweisung notiert werden. Was wiederum die Parallelisierung und semperate Entscheidungsknoten spart und den Prozess deutlich übersichtlicher macht als durch eine parallele Aufspaltung.

Traditionell werden erreignisgesteuerte Prozessketten normalerweise mit einem Fluss von oben nach unten dargestellt. Dies haben wir grundsätzlich auch so übernommen. Jedoch haben wir uns überlegt welche Teilaktivitäten auf irgendeine Weise zusammenhängend sind. Diese haben wir mit einem waagrechten Sequenzfluss dargestellt. Dies soll der Übersichtlichkeit der Diagramme dienen.

Geschriebene oder gelesene Daten werden durch die modellierten Ereignisse deutlich.

Die Überwachung des Zeitrahmens des Sprint Planning Meeting wird durch eine Funktion des Scrum Masters dargestellt.

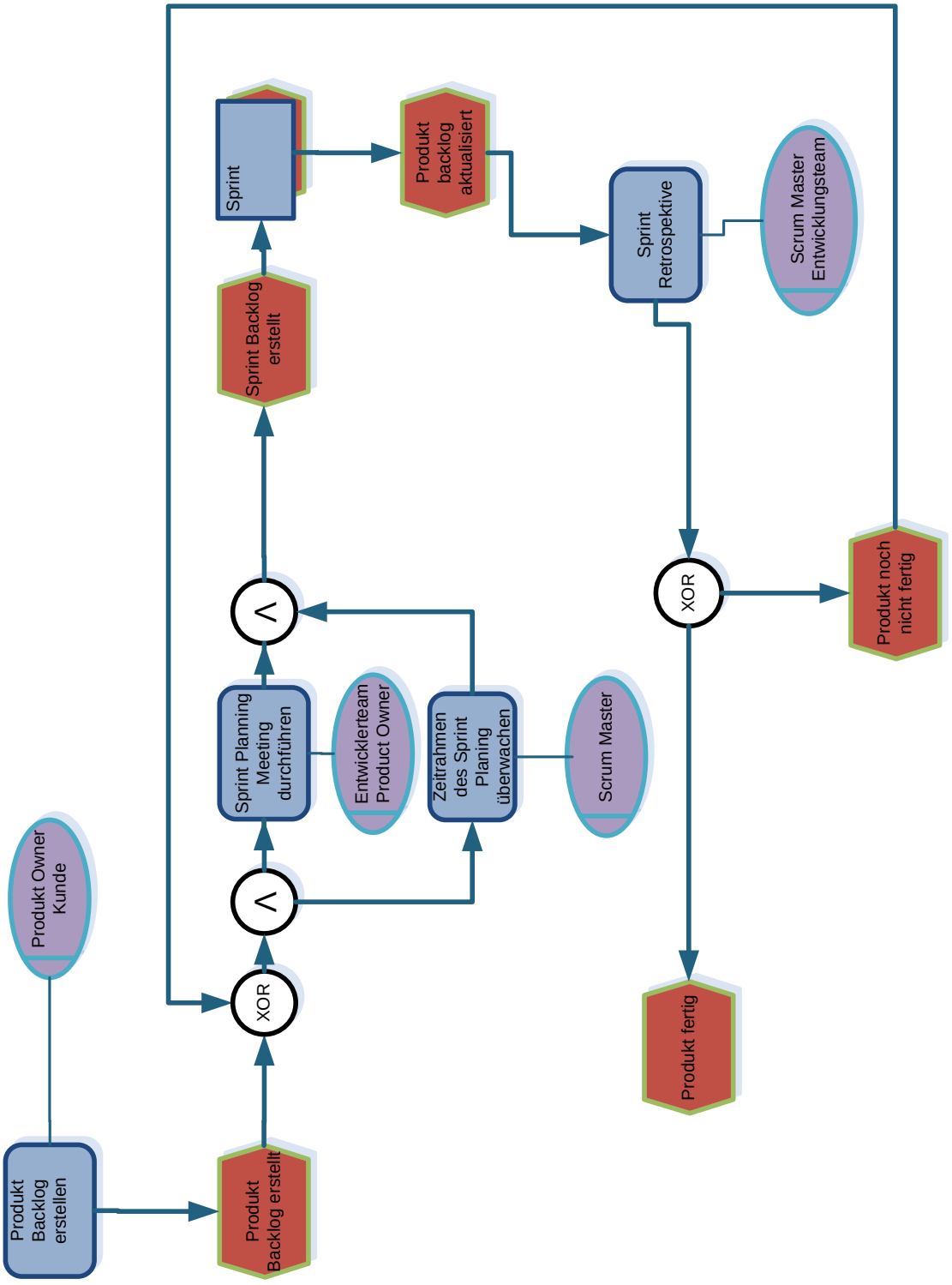

Abbildung 4.3: Scrum mit EPK modelliert - Main Prozes

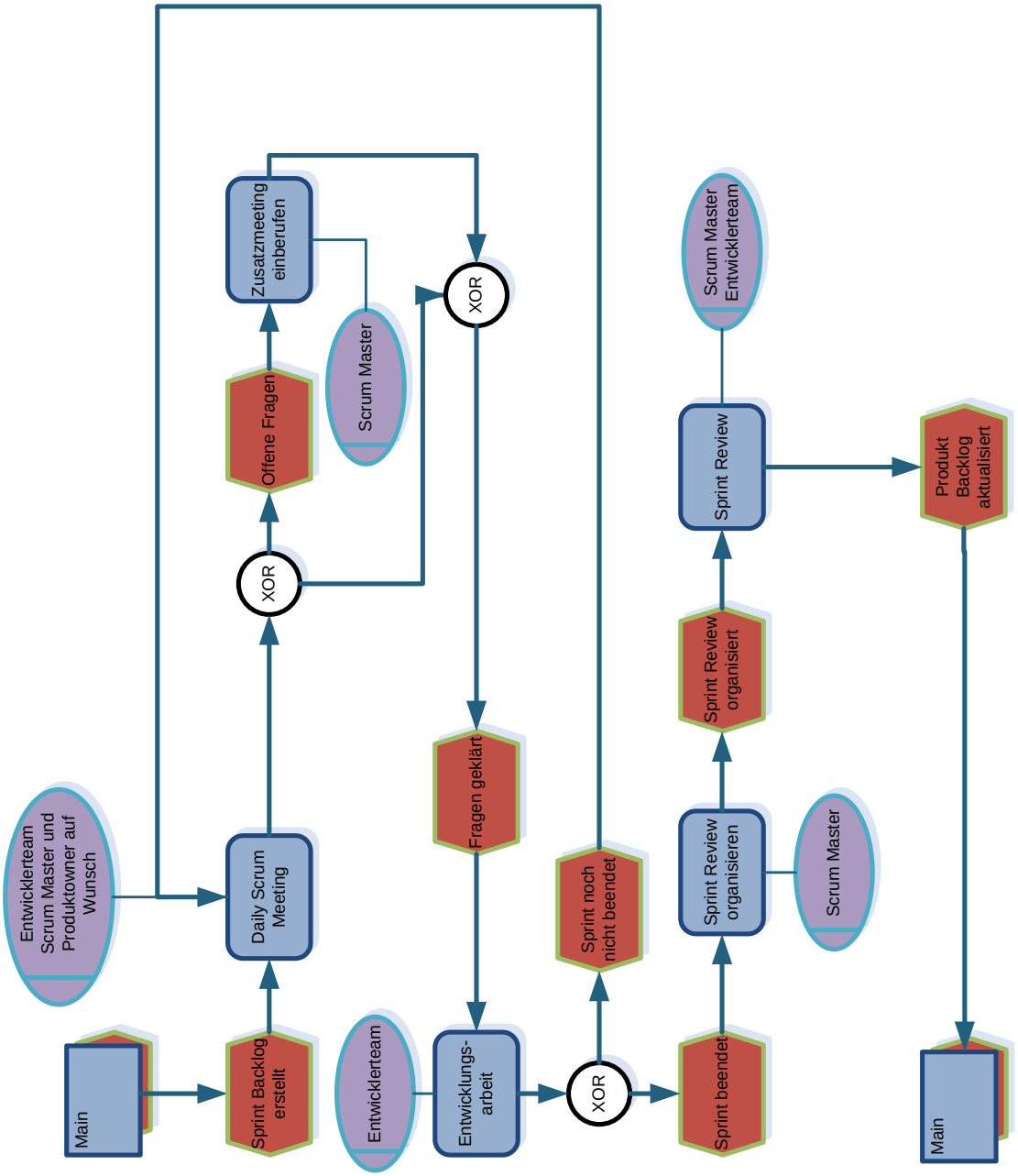

Abbildung 4.4: Scrum mit EPK modelliert - Sprint Prozess

4.1.3 Scrum mit BPMN modelliert

Auch bei der Modellierung mit BPMN verwenden wir eine Aufgliederung in einen Mainprozess und einen Subprozess für den Sprint. Der Mainprozess wird in Abbildung 4.5 dargestellt. Der Subprozess wird in Abbildung 4.6 modelliert.

Um Benutzerrollen darzustellen, verwenden wir das Konzept der Pools und der Lanes. In dem speziellen Fall von Scrum gibt es genau einen Pool, welcher das komplette Scrum Team beinhaltet. In diesem Pool werden drei Lanes modelliert. Eine Lane jeweils für den Scrum Master, den Produkt Owner sowie für das Entwicklerteam.

Um darzustellen, dass eine Aktivität durch mehrere Rollen gemeinsam bearbeitet wird, verwenden wir eine Parallelisierung, so dass die Aktivität dann parallel in jeder betroffenen Lane dargestellt wird. Anschließend wird dann der Kontrollfluss wieder zusammengeführt.

Man hätte sich auch Alternativen zu dem Konzept der Pools und Lanes überlegen können. Jedoch hätten diese zu einer Erweiterung der BPMN Notation geführt, was wir jedoch nicht wollten.

Wir haben wichtige Daten die gelesen oder geschrieben werden sollen, mittels Datenelementen dargestellt.

Um darzustellen ob der Scrum Master und Produkt Owner freiwillig am Daily Scrum teilnehmen wollen, habe ich ebenfalls Parallelisierung mit einer darauffolgenden Entscheidung in jeder einzelnen Rolle verwendet. Alternativ hätte man hier über einen Kommentar arbeiten können, was jedoch die Entscheidung der Personen nicht so schön dargestellt hätte.

Um das zeitliche Ende des Sprints darzustellen bietet sich eine benutzerdefinierte Entscheidung in Verbindung mit einem Timer Signal an. Liegt das Timer Signal am Entscheidungsknoten an, so ist der Sprint zu Ende. Liegt es nicht an geht es in einen weiteren Entwicklungstag. Auch hier sind Alternativen denkbar, beispielsweise dass eine Entscheidung SSprint-Ziel erreicht"modelliert wird. Wir haben uns jedoch für den Timer entschieden da dieser besser deutlich macht dass die vorgegebene Zeit für einen Sprint eingehalten werden soll.

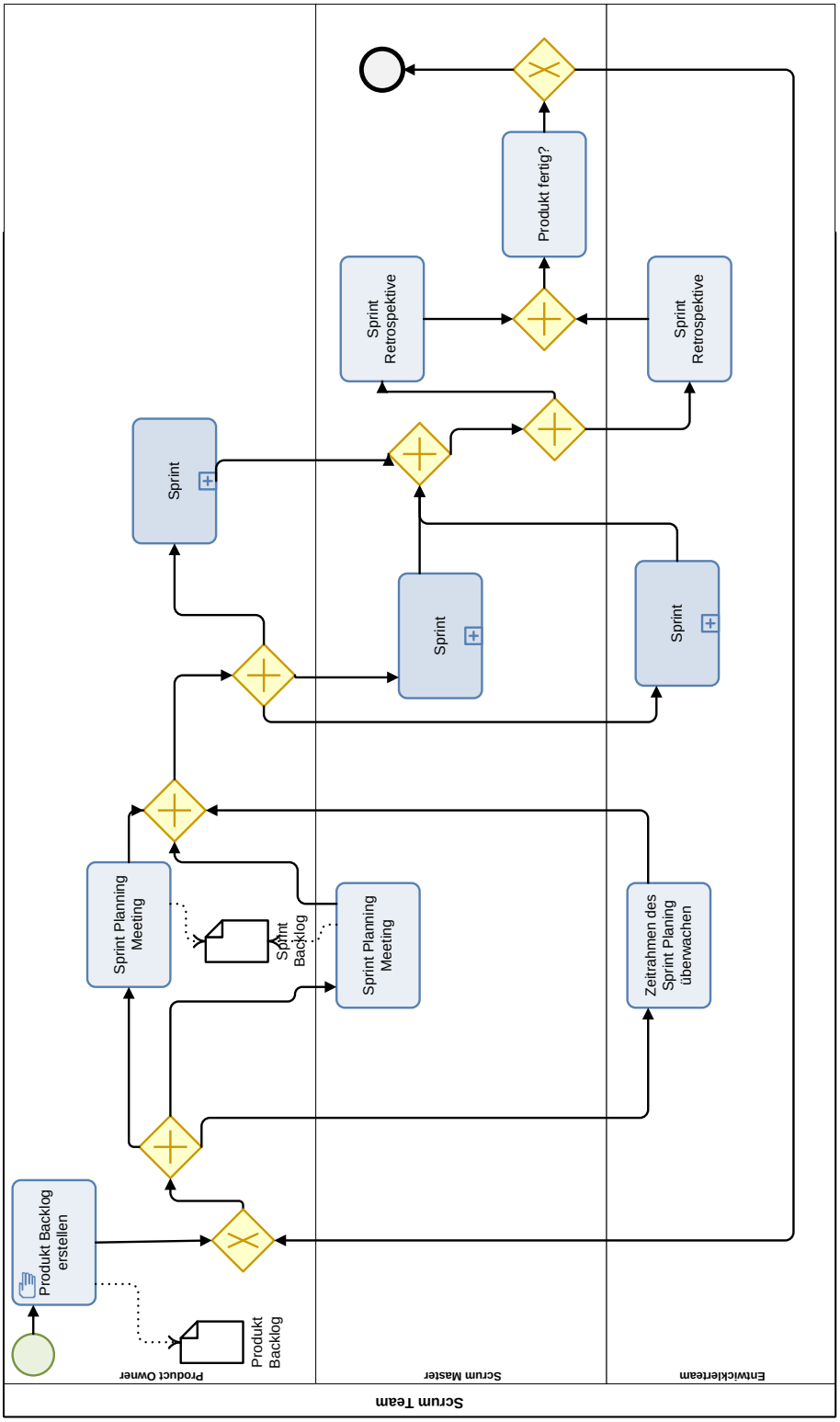

Abbildung 4.5: Scrum mit BPMN modelliert - Main-Prozess

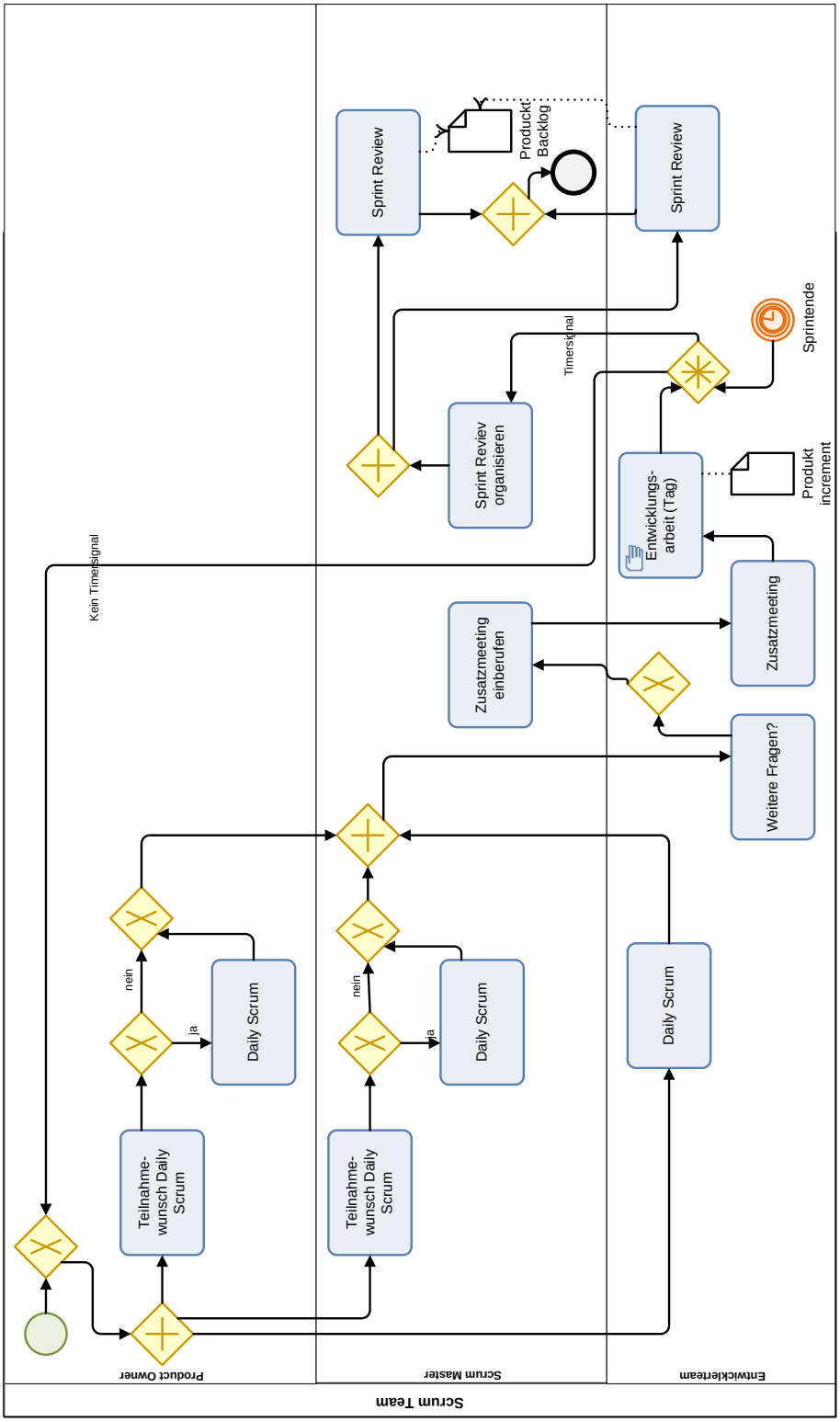

Abbildung 4.6: Scrum mit BPMN modelliert - Sprint-Prozess

4.2 Xtreeme Programming

In diesem Abschnitt werden wir XP mit drei verschiedene Notationen darstellen.

4.2.1 XP mit UML-Aktivitätendiagrammen modelliert

Wir haben den XP Prozess bei der Modellierung mittels UML-Aktivitätendiagramm, um die Modellierung übersichtlich zu gestalten, in einen Mainprozess und zwei Verfeinerungsstufen aufgeteilt. Der Mainprozess ist dargestellt in Abbildung 4.7.

Der Mainprozess behandelt lediglich wie die User Stories entstehen und dass diese implementiert werden müssen. Er geht jedoch noch nicht auf die Implementierung dieser ein. Dies bietet sich an da die Implementierung der Stories einen Teilprozess mit sehr wenigen Schnittellen zum restlichen Prozess darstellen und somit gut herausgegriffen werden kann.

Die erste Verfeinerungsstufe stellt einen Teilprozess dar welchen wir mal den Stories Prozess nennen. Dieser zeigt wie der Ablauf bei der Implementierung der User Stories ist. Er wird in Abbildung 4.8 dargestellt. Das heißt, es wird modelliert nach welchen Prinzipien die User Storys nacheinander abgearbeitet werden und was zwischen der Bearbeitung der einzelnen User Stories geschieht. Es wird aber noch nicht dargestellt wie dann eine konkrete User Story implementiert wird.

Da der Stories Prozess noch nicht darauf eingeht wie die Implementierung einer User Story tatsächlich abläuft, ist eine dritte Verfeinerungsstufe nötig. Der Story Prozess, welcher in Abbildung 4.9 dargestellt ist, stellt das Abhandeln einer einzelnen User Story dar.

Der gesamte Prozess beginnt damit, dass der Kunde den Auftrag gibt, die neue Release zu produzieren. Dies wird mit der Aktivität, eine neue Release fordern dargestellt.

Als Ergebnis der parallel ausgeführten Releaseplanung entsteht ein Releseplan welcher als Datenobjekt modelliert wurde.

Darauffolgend werden die User Stories ausgearbeitet. Das Ergebnis wird wiederum als Datenobjekt dargestellt. Zuerst nur als User Stories, später dann als priorisierte Liste von User Stories.

An dieser Stelle wird der Subprozess User Stories gerufen.

Sind alle Subprozesse ausgeführt, wird mit einer Entscheidung noch dargestellt, ob es eine weiterer Release geben soll oder ob der Kunde keine weiteren Neuerungen wünscht.

Im Subprozess Abbildung 4.8 wird in einer ersten Aktivität die höchst priorisierte User story gewählt. Mit der gewählten User Story wird ein Subprozess 4.9 gerufen welcher die Aktivitäten zum Implementieren dieser Story aufzeigt. Ist die Story implementiert, entsteht als Ergebnis ein Produkt Increment welches wir mittels Datenobjekt dargestellt haben.

Wird der anschließende Akzeptancetest nicht bestanden, so wird eine neue User Story für diese Funktion formuliert, was wir mittels einer Aktivität und dem Datenobjekt User Storys darstellen.

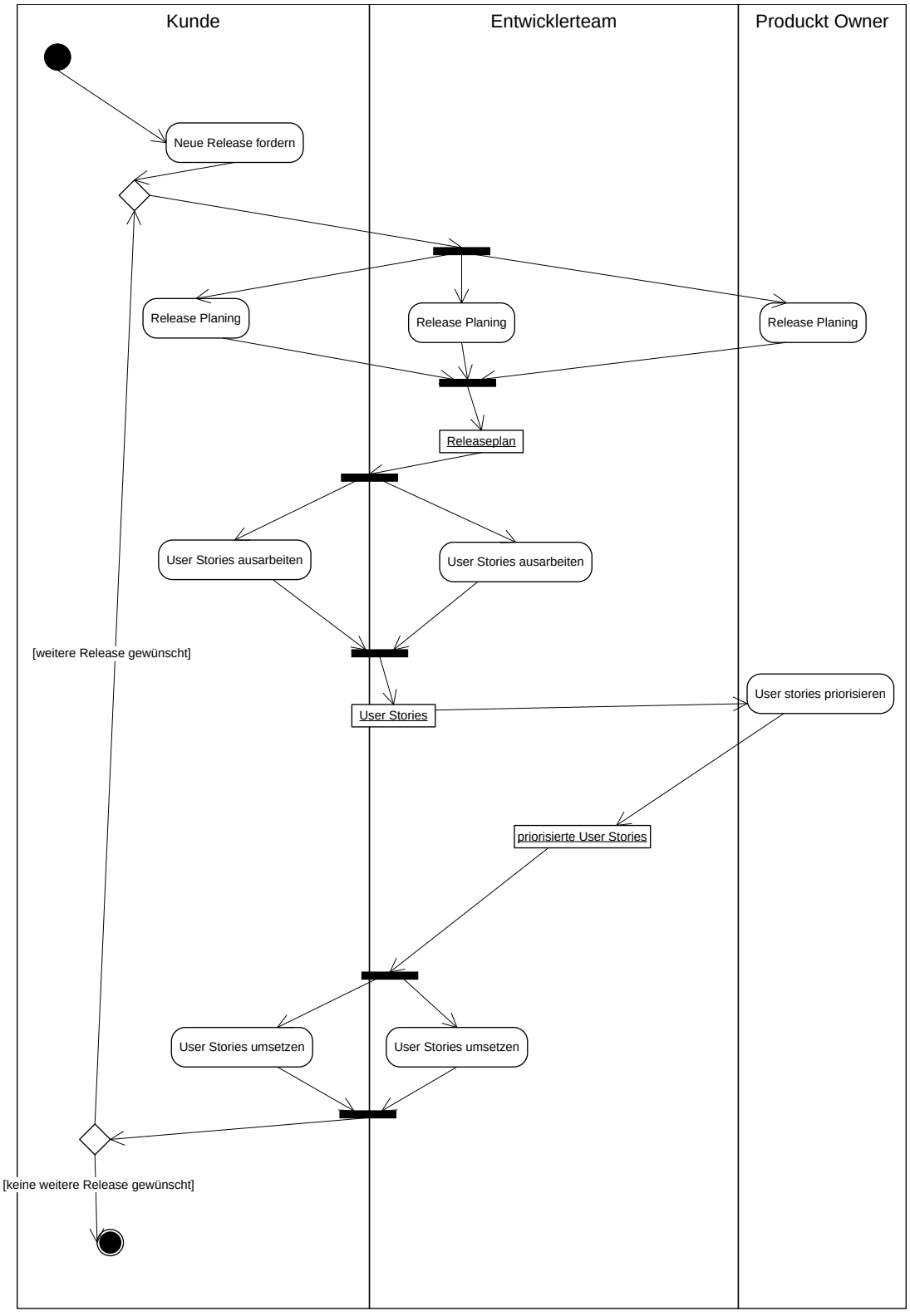

Abbildung 4.7: XP mit Aktivitätendiagrammen modelliert - Main-Prozess

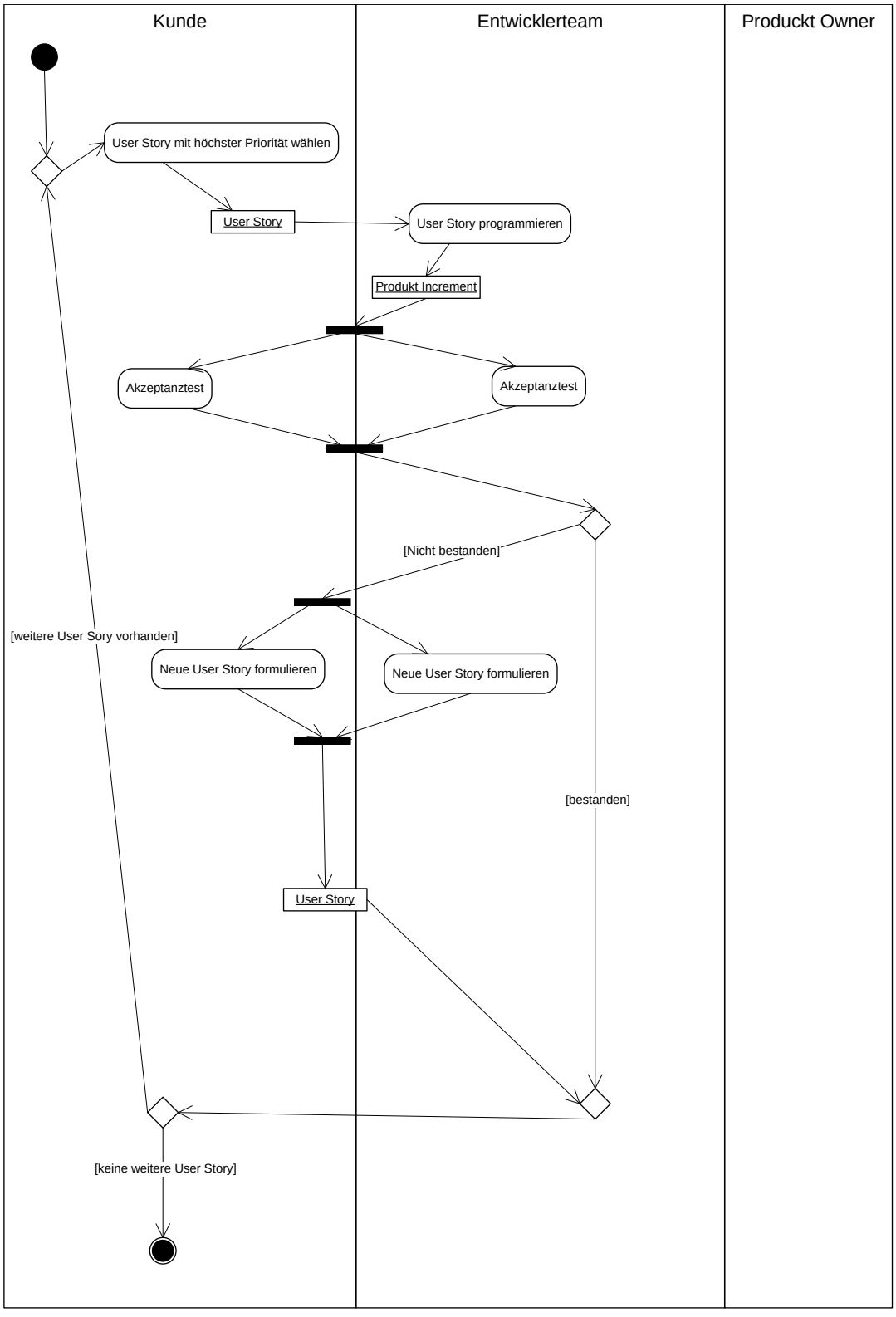

Abbildung 4.8: XP mit Aktivitätendiagrammen modelliert - Storys-Prozess

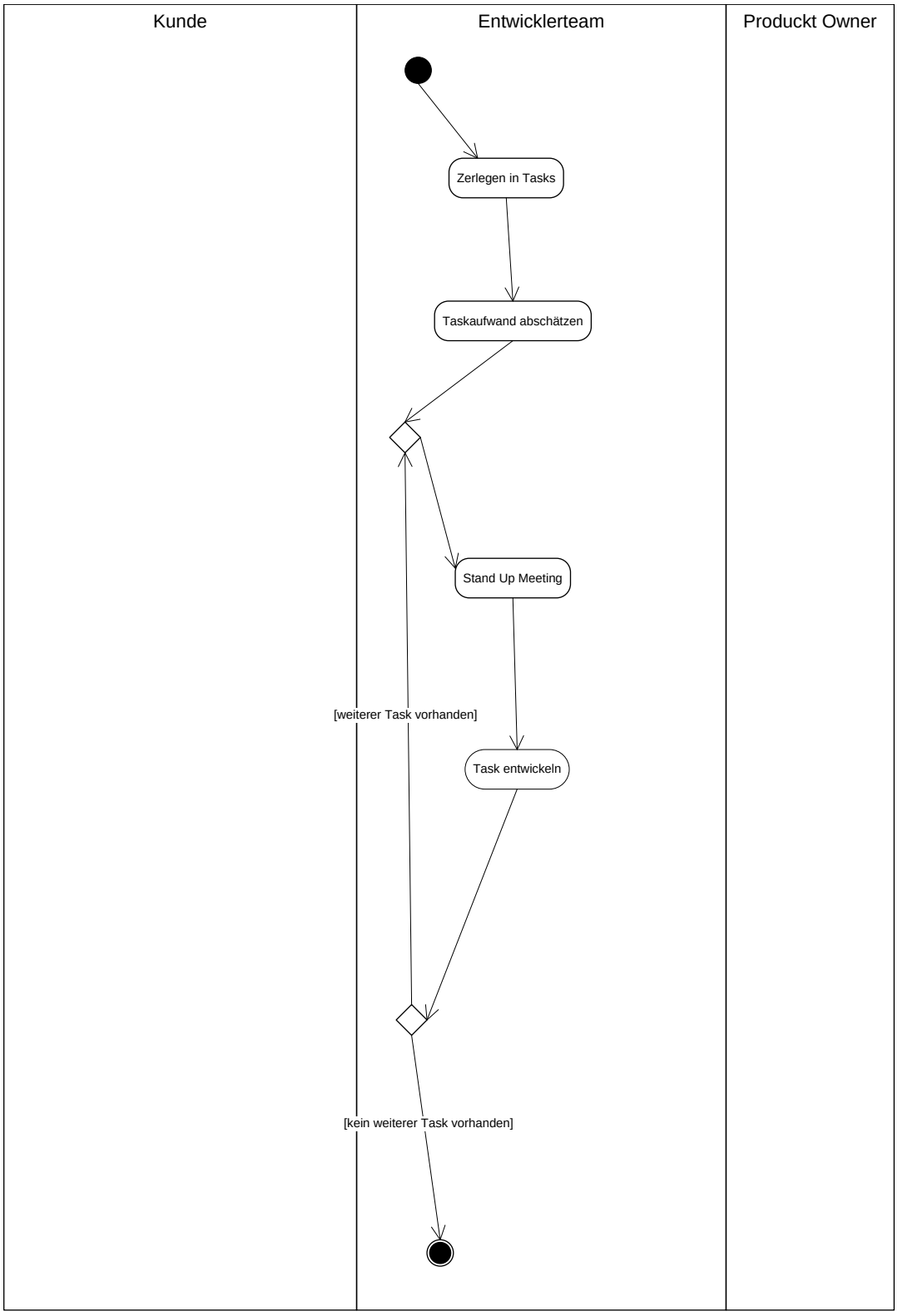

Abbildung 4.9: XP mit Aktivitätendiagrammen modelliert - Story-Prozess

4.2.2 XP mit BPMN-Notation modelliert

Die Darstellung mittels BPMN ist der mit den UML Diagrammen recht ähnlich allerdings ist sie deutlich übersichtlicher. Daher nehmen wir dieselbe Aufteilung in einen Mainprozess und in zwei Verfeinerungen, wie auch schon bei den UML Diagrammen vor. Dies bietet sich aus den selben Gründen wie schon bei den UML Diagrammen an.

Der Main Prozess wird in Abbildung 4.10 dargestellt, die erste Verfeinerungsstufe, der Stories Prozess in Abbildung 4.11 und die letzte Verfeinerung, der Story Prozess in Abbildung 4.12.

Der größte Unterschied im Vergleich zu den vorangegangenem UML Diagrammen ist die Darstellung der Datenelemente. Hier werden Datenelemente direkt den Aktivitäten zugeordnet und nicht in den Kontrollfluss integriert. Dies sorgt für eine deutlich bessere Lesbarkeit des Prozesses.

Im Wesentlichen haben wir bei der BPMN dieselben Aktivitäten und Entscheidungen modelliert wie schon mit der UML, nur mit etwas anderer Syntax.

Um das Stand Up Meeting genauer zu beschreiben verwenden wir im Diagramm 4.12 einen Kommentar welcher angibt, dass in diesem Meeting ebenfalls die Paare für das Pair Programming eingeteilt werden. Alternativ hätte man sich überlegen können eine eigene Aktivität Paare einteilen zu modellieren. Dies ist jedoch nicht geschehen da dies eigentlich zu der Aktivität Stand Up Meeting automatisch dazugehört. Für den Leser des Diagrammes dient es jedoch zum besseren Verständnis wenn er mittels Kommentar kurz darauf hingewiesen wird.

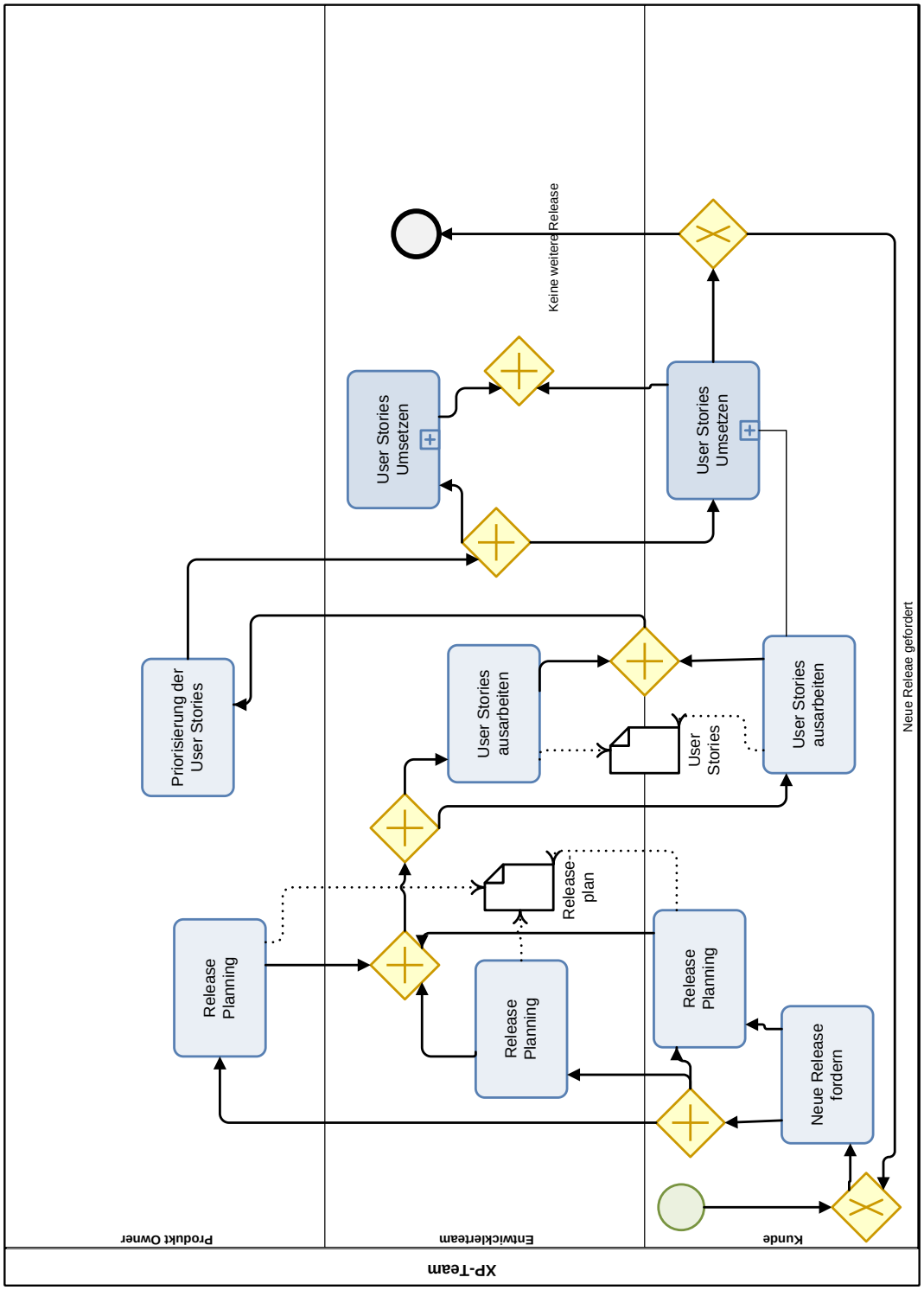

Abbildung 4.10: XP mit BPMN modelliert - Main-Prozess

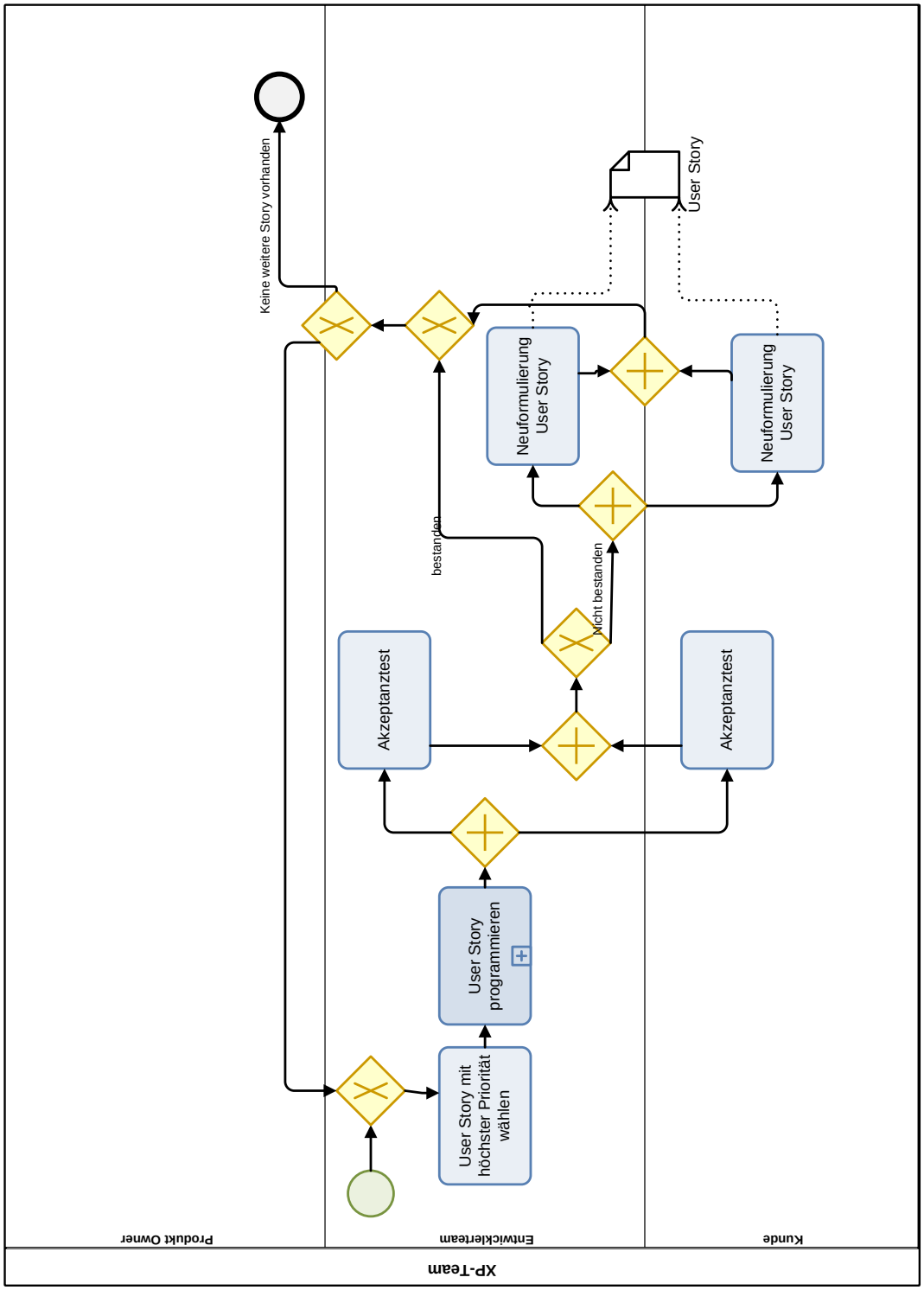

Abbildung 4.11: XP mit BPMN modelliert - Stories implementieren-Prozess

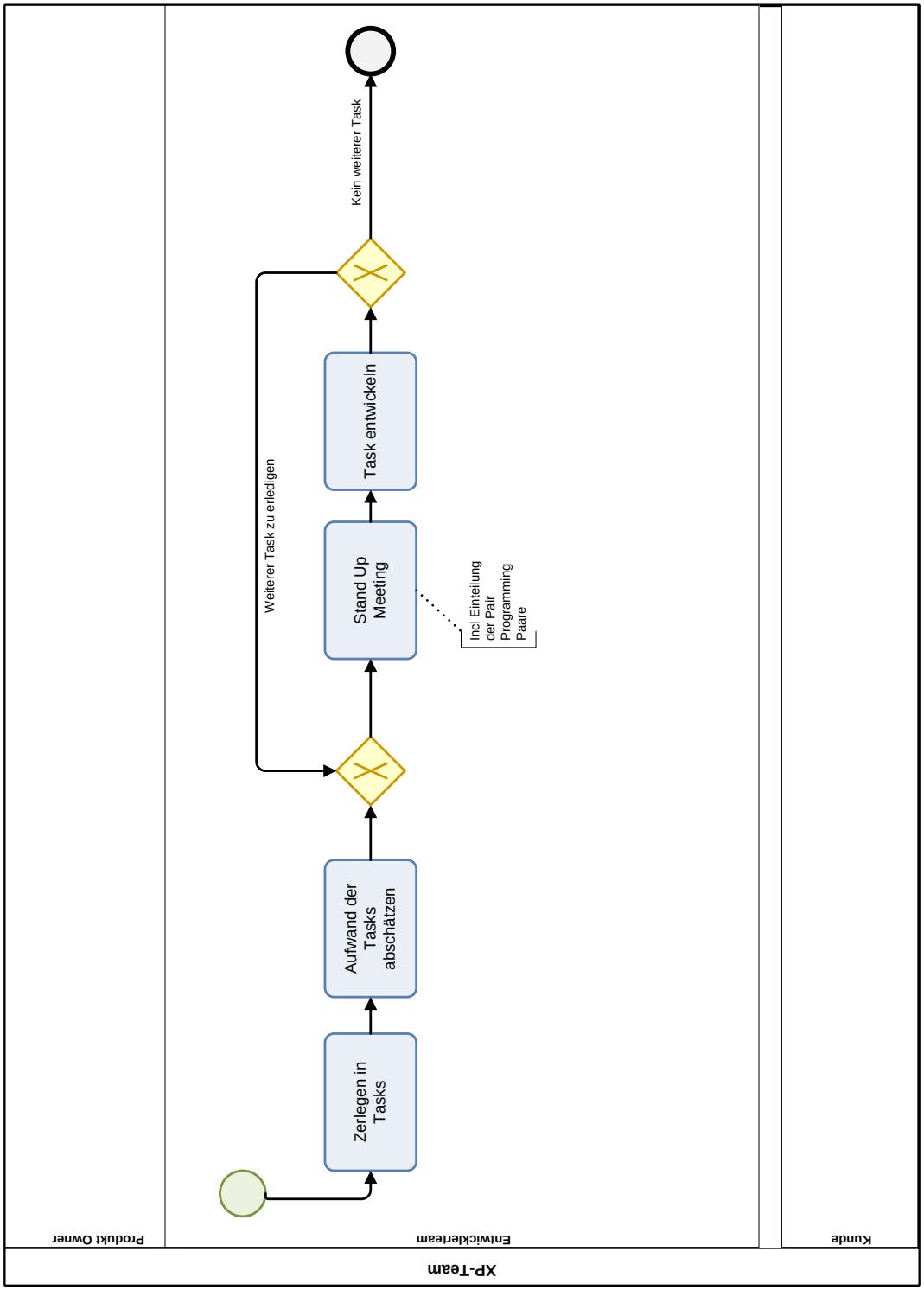

Abbildung 4.12: XP mit BPMN modelliert - Story implementieren-Prozess

4.2.3 XP mit EPKs modelliert

Auch um XP mittels EPKs darzustellen bedienen wir uns eines Main Prozesses mit zwei Verfeinerungsstufen. Welche nach den selben Kriterien wie in den bereits dargestellten Modellierungen gebildet werden.

Der Main Prozess wird in Abbildung 4.13 modelliert.

Wie die einzelnen User Stories abgearbeitet werden wird in 4.14 gezeigt. Hierbei wird jedoch noch nicht darauf eingegangen wie genau die einzelnen Stories implementiert werden, sondern nur darauf nach welcher Reihenfolge das Implementieren der Stories geschieht.

Und schließlich modelliert 4.9 dann das Umsetzen einer einzelnen User Story im Entwicklungsteam. Hierbei wird genau angegeben wie das Ausarbeiten der konkreten Story geschieht.

Die Diagramme mit den EPKs enthalten wesentlich mehr Elemente wie die vorangegangenen Darstellungsarten. Dies liegt nicht daran, dass hier wesentlich mehr Inhalte modelliert wurden, sondern daran, dass sich in einer ereignisprozessgesteuerten Prozesskette stets Funkionen und Ereignisse abwechseln müssen. So dass Elemente eingefügt wurden, welche zum Verständnis unnötig aber zum Einhalten der korrekten Notation nötig sind.

Die entstandenen Daten werden mittels der entstehenden Ereignisse dargestellt.

Im klassischen Sinne werden EPKs von oben nach unten dargestellt. Wir haben dies zwar grundsätzlich beibehalten aber immer wieder einen Sequenzfluss von links nach rechts modelliert. Der Gedanke hierbei war, dass ein waagrecht modellierter Sequenzfluss Aktivitäten verbindet welche irgendwie zusammenhängen. Dies soll dem besseren Verständnis des Prozesses dienen.

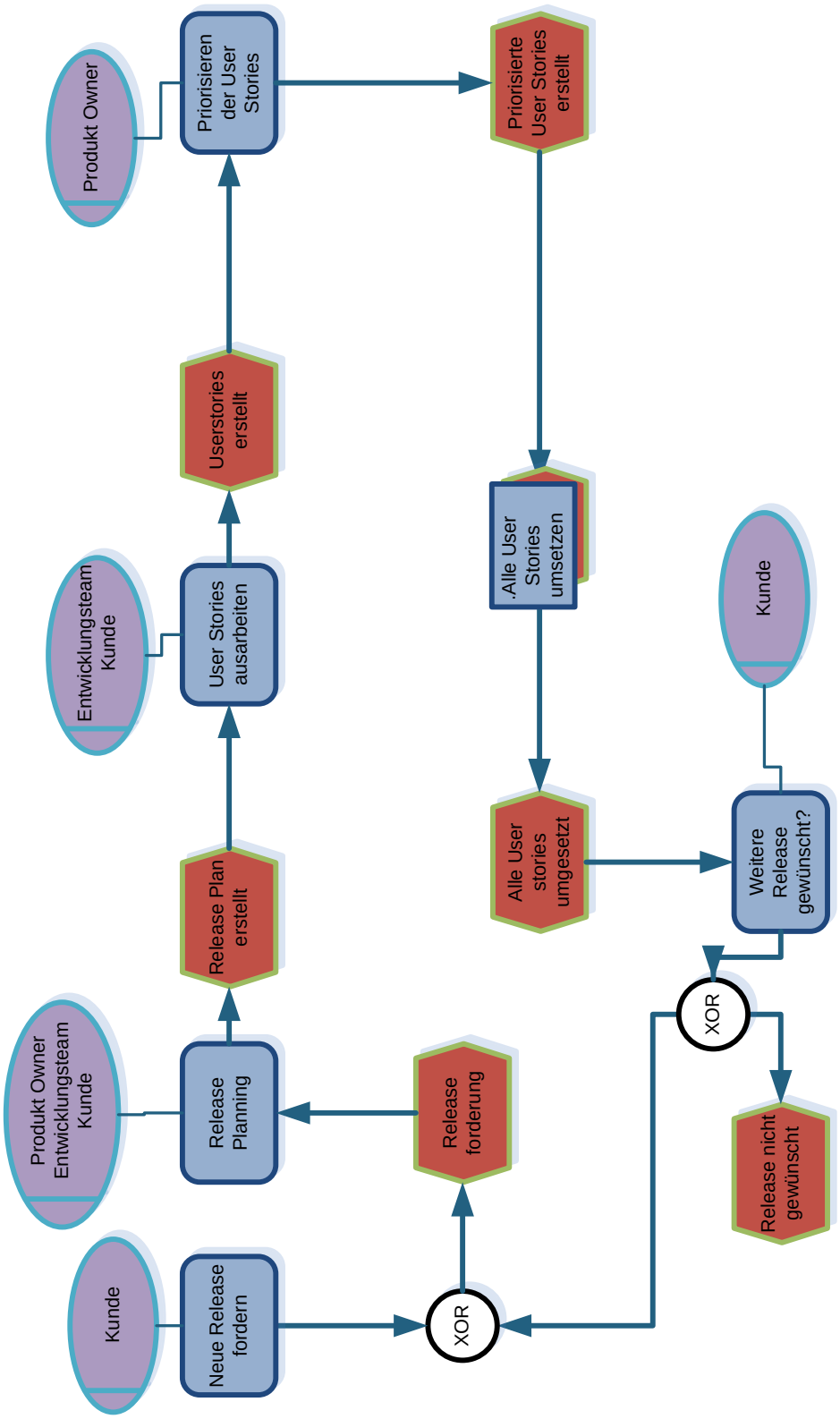

Abbildung 4.13: XP mit EPK modelliert - Main Prozess

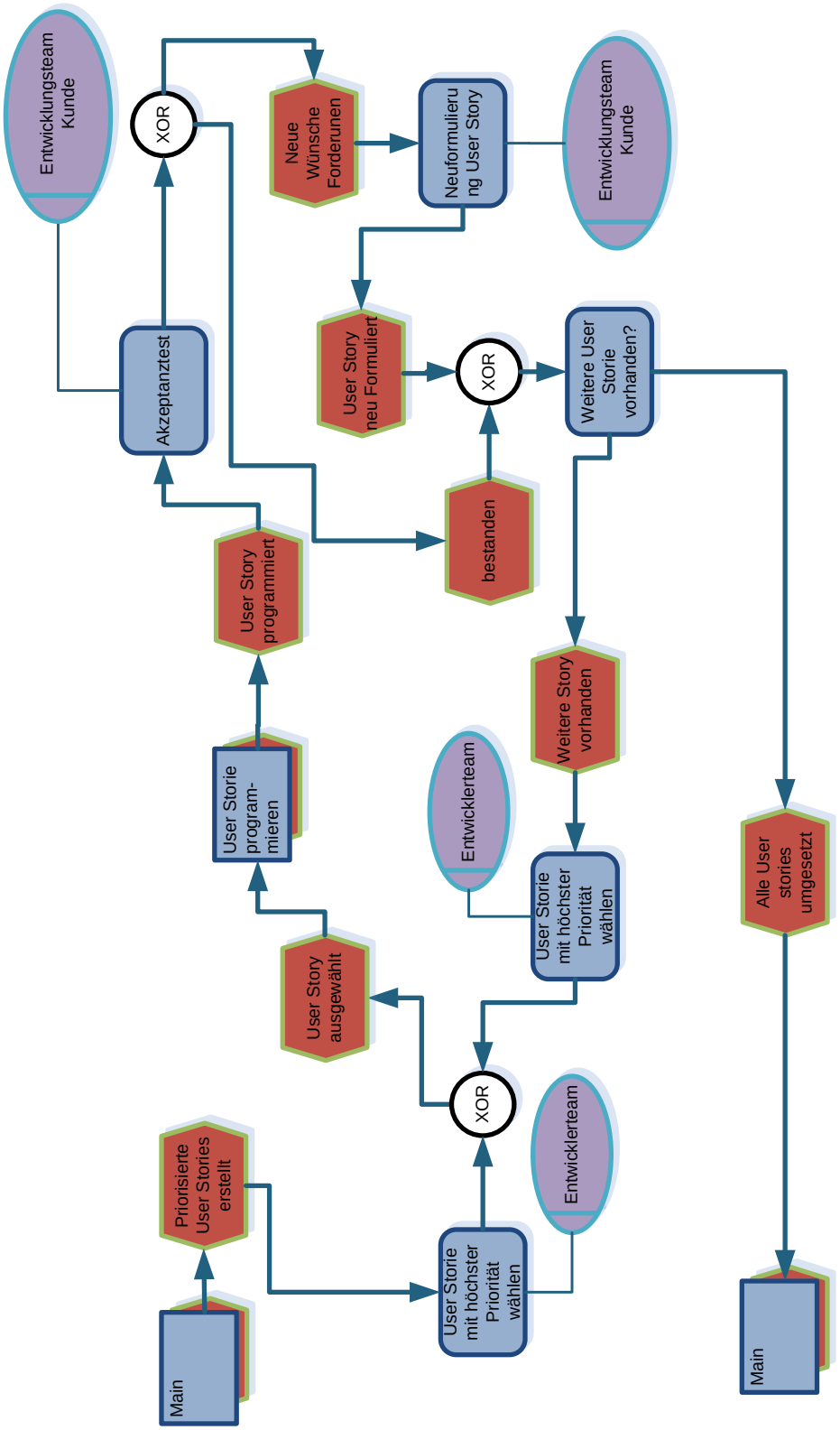

Abbildung 4.14: XP mit EPK modelliert - Stories Prozess

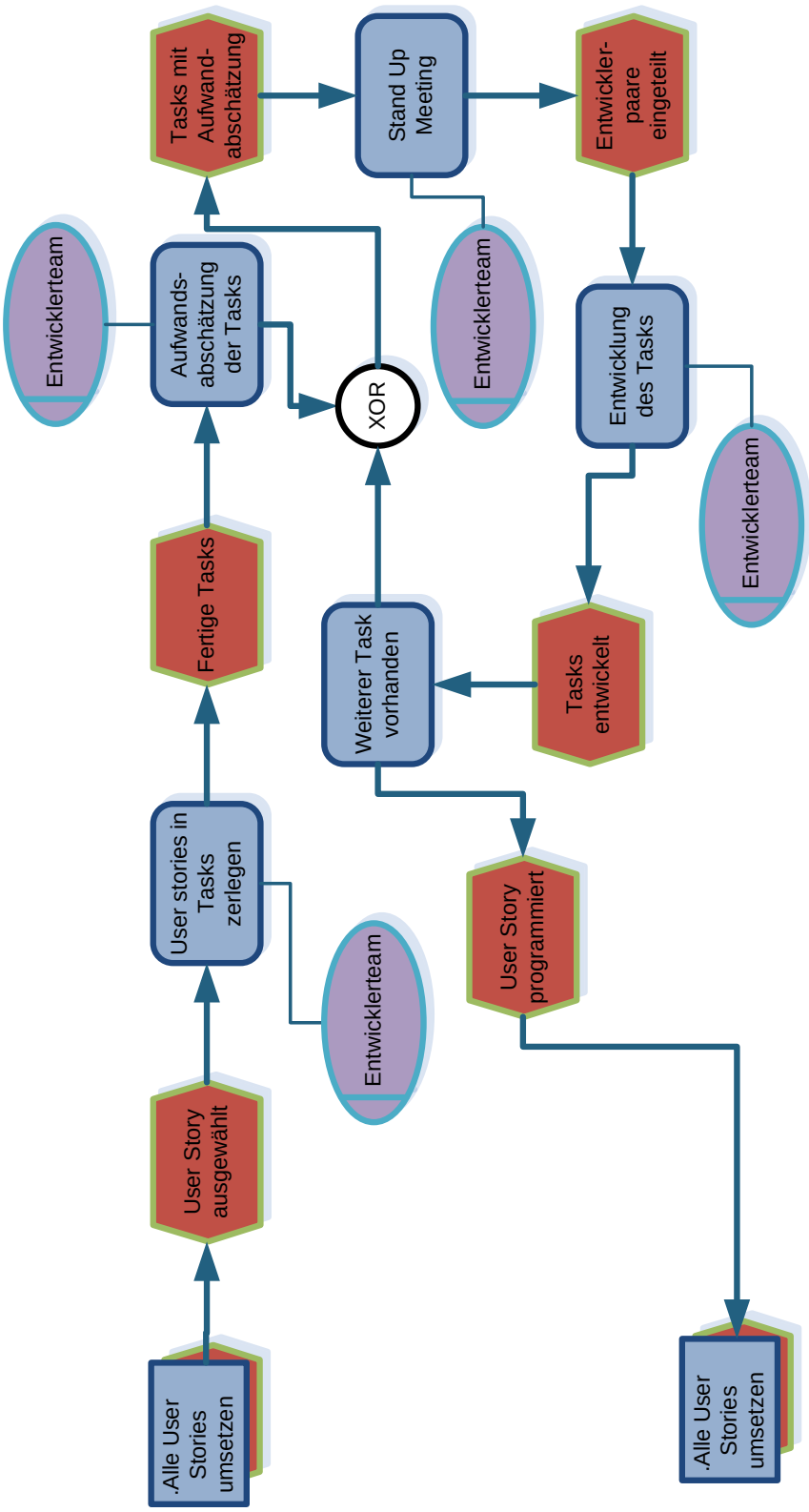

Abbildung 4.15: XP mit EPK modelliert - Story Prozess

4.3 Open UP

In diesem Abschnitt modellieren wir noch den vom RUP Prozess abgeleiteten Open UP Prozess.

Da das Vorgehen im Open UP sehr detailliert beschrieben werden kann, haben wir nicht jedes Detail implementiert. Dies bedeutet es gibt noch weitere Workflows, welche auf eine ähnliche Weise als Diagramm dargestellt werden könnten.

In allen drei Notationen haben wir zunächst einen Main Prozess modelliert welcher den Ablauf der vier Phasen verdeutlicht. Dargestellt in den Abbildungen 4.22, 4.28 und 4.16.

In einer ersten Verfeinerungsstufe haben wir jede der vier Phasen modelliert.

Alle Aufgaben, welche in den einzelnen Phasen enthalten sind, sind nochmals näher spezifiziert. Da jedoch eine Modellierung all dieser weiteren Verfeinerungen den Umfang dieser Arbeit sprengen würde, haben wir nur noch exemplarisch den Develop Solution Increment Workflow weiter verfeinert. Für alle anderen könnte ebenfalls anhand der bereitgestellten Workflows der Eclipse Foundation [EF 5] eine Notation in den einzelnen Diagrammarten erstellt werden.

4.3.1 Open UP mit UML-Aktivitätendiagrammen modelliert

Wir modellieren Open Up zunächst mittels UML Aktivitätendiagrammen. Der Main Prozess welcher den Ablauf der Phasen darstellt wird in Abbildung 4.16 abgebildet.

Dieser stellt einen rein sequentiellen Ablauf dar. Iterationen kommen erst in den Verfeinerungen dazu.

Die einzelnen Phasen werden dargestellt in den Abbildungen:

1. Inception in Abildung: 4.17,
2. Elaboration in Abbildung: 4.18,
3. Construction in Abildung: 4.19,
4. Transition in Abildung: 4.20.

Die Verfeinerung des Develop Solution Increment wird in Abbildung 4.21 illustriert.

Innerhalb des Entwicklungsteams gibt es Spezialisten für de einzelnen Arbeiten. Auf eine Darstellung dieser Rollenzuordnung verzichten wir, um das Diagramm lesbar zu halten. Daher finden alle Vorgänge im Entwicklungsteam statt. Das heißt dass auf eine parallele Aufspaltung um parallele Aktivitäten mehrerer Rollen verzichtet werden kann.

Der Kunde und der Projektmanager spielen nur im Mainprozess eine Rolle.

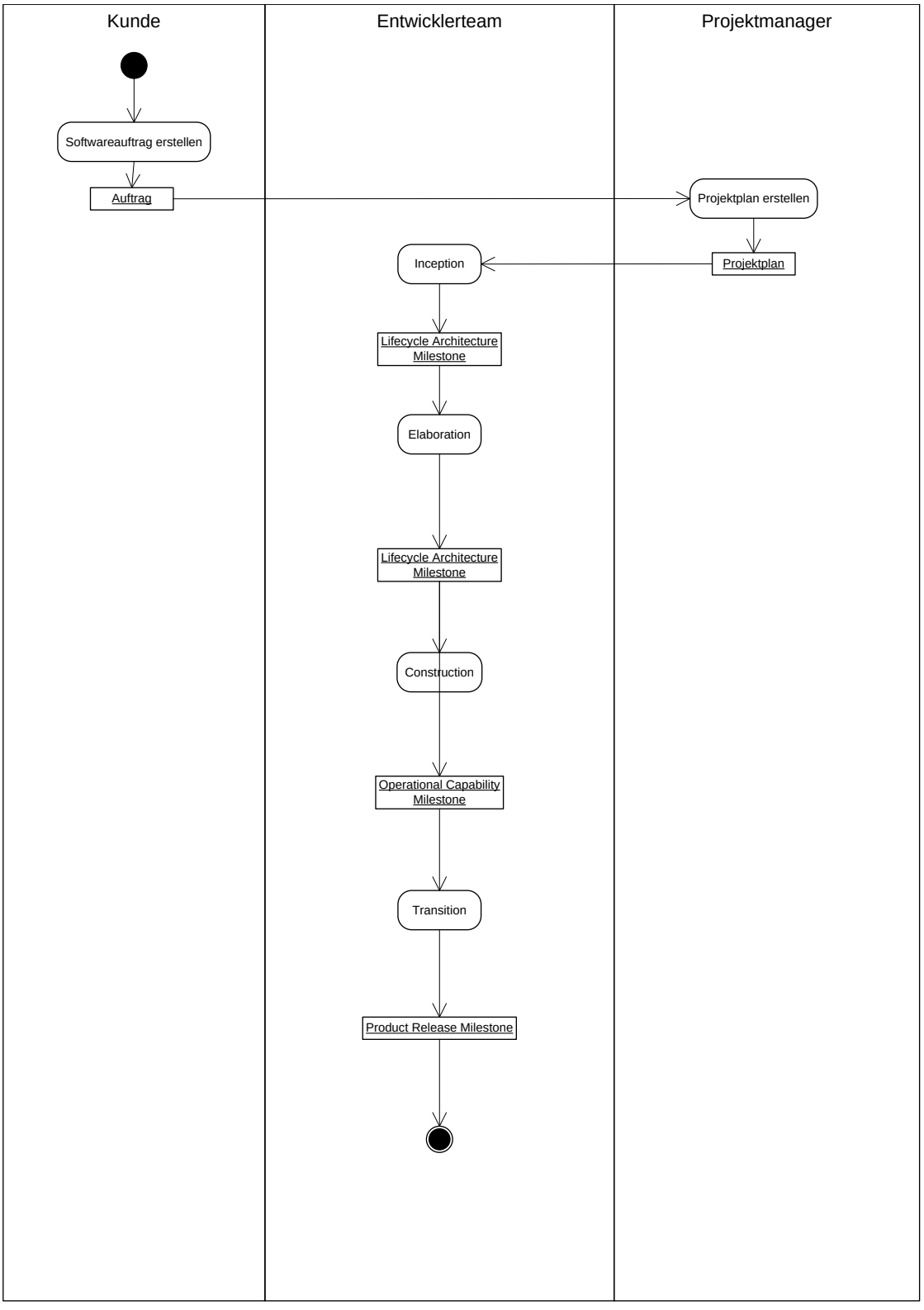

Abbildung 4.16: Open UP mit UML modelliert - Main Prozess

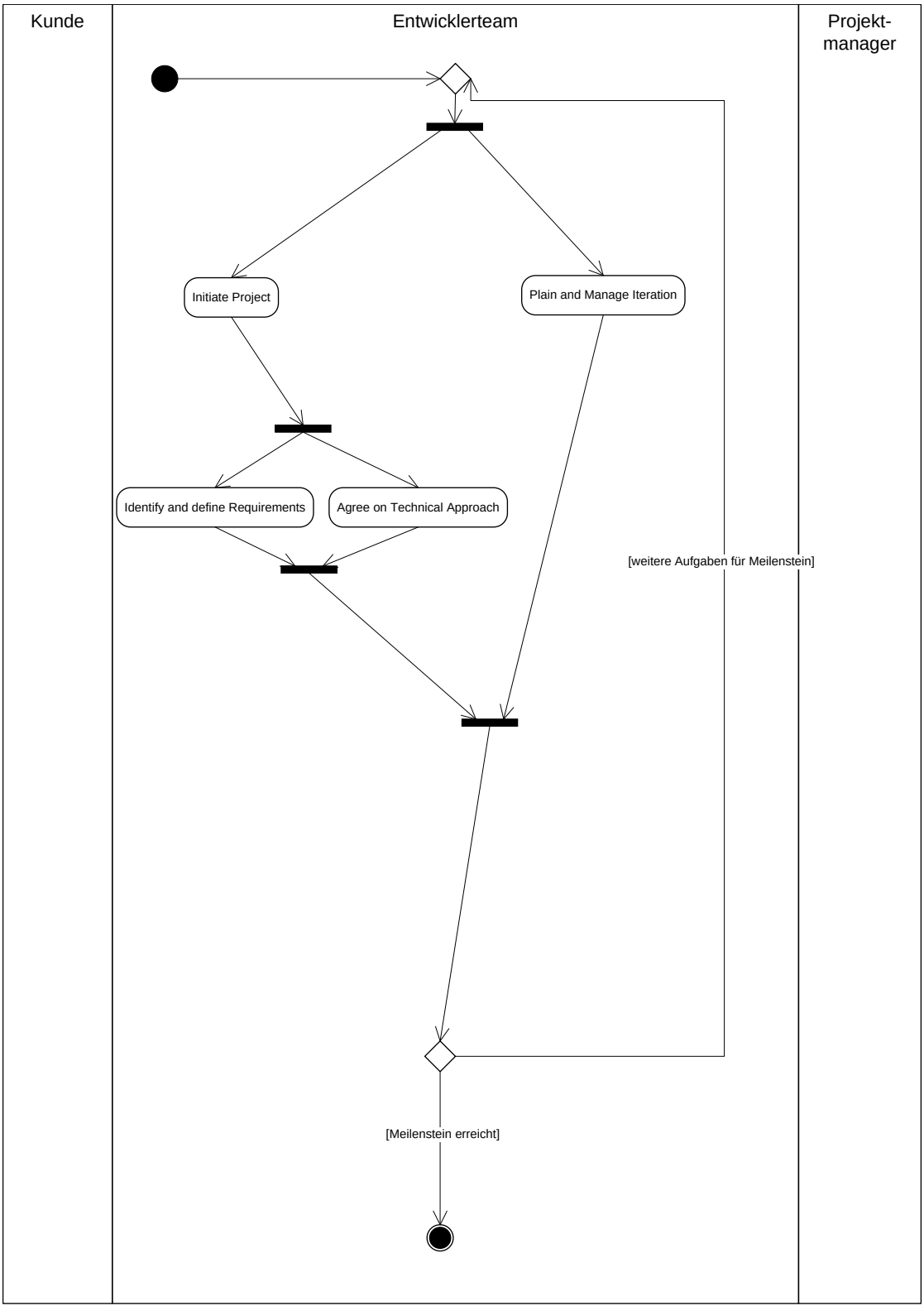

Abbildung 4.17: Open UP mit UML modelliert - Inception Prozess

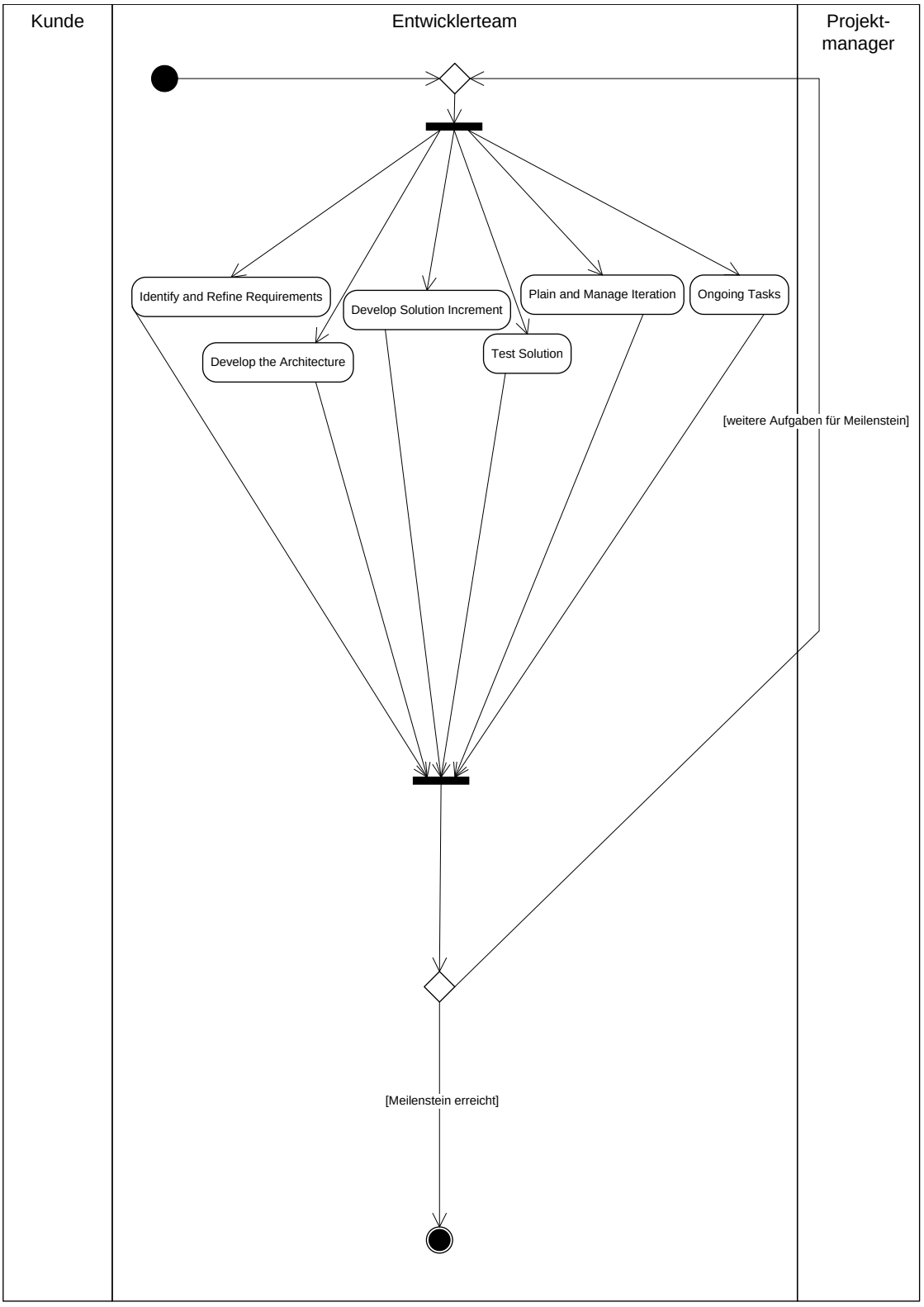

Abbildung 4.18: Open UP mit UML modelliert - Elaboration Prozess

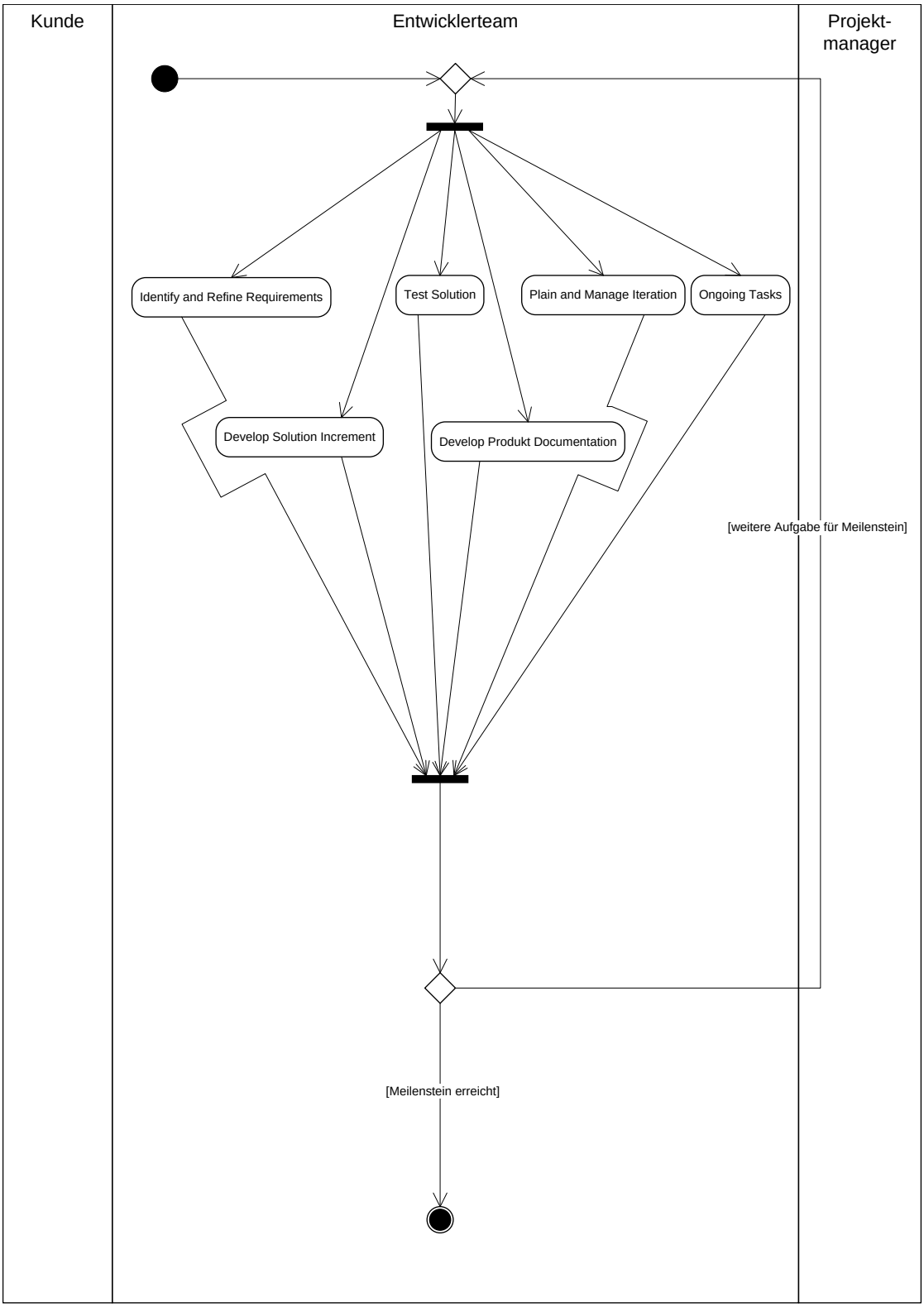

Abbildung 4.19: Open UP mit UML modelliert - Construction Prozess

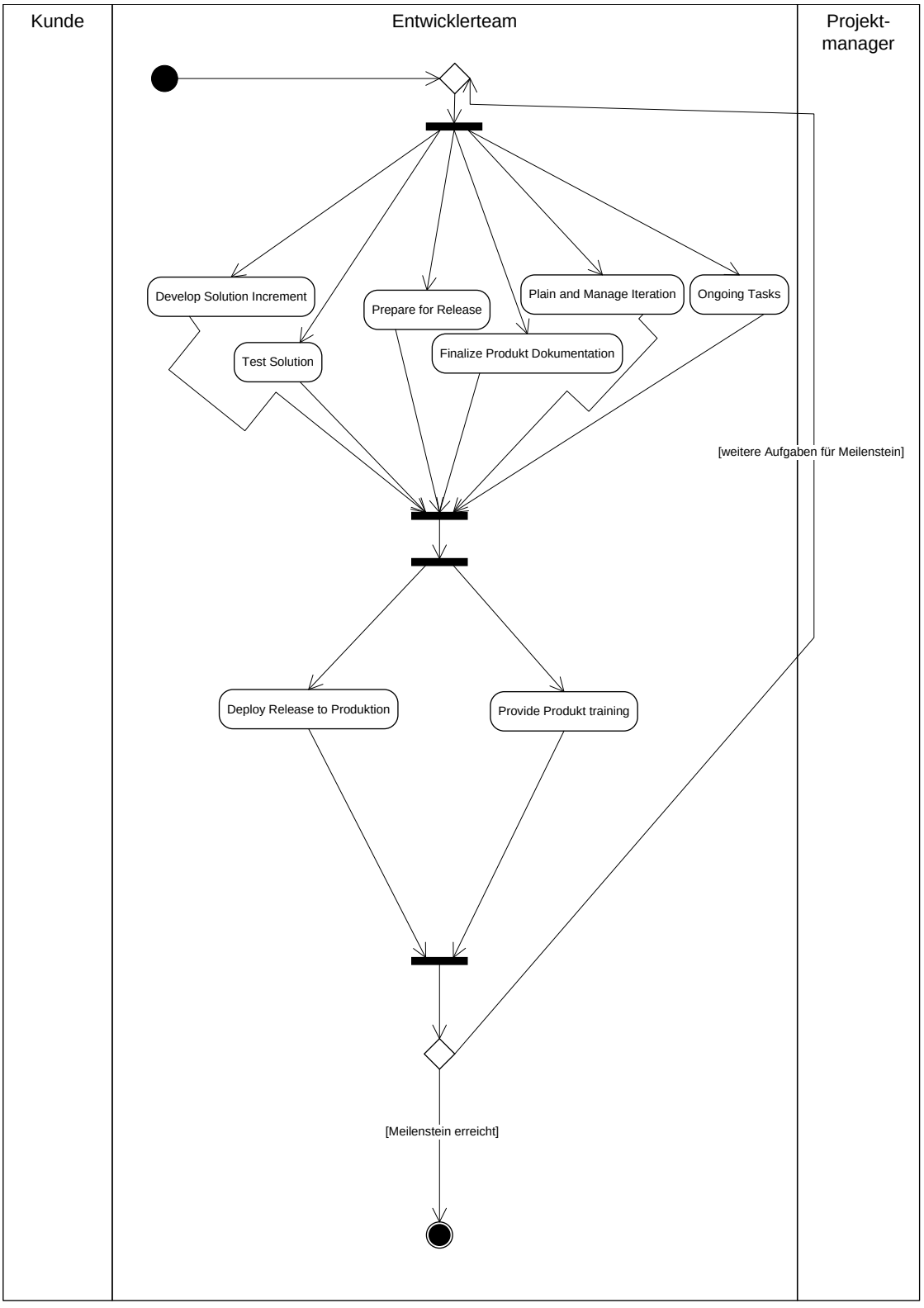

Abbildung 4.20: Open UP mit UML modelliert - Transition Prozess

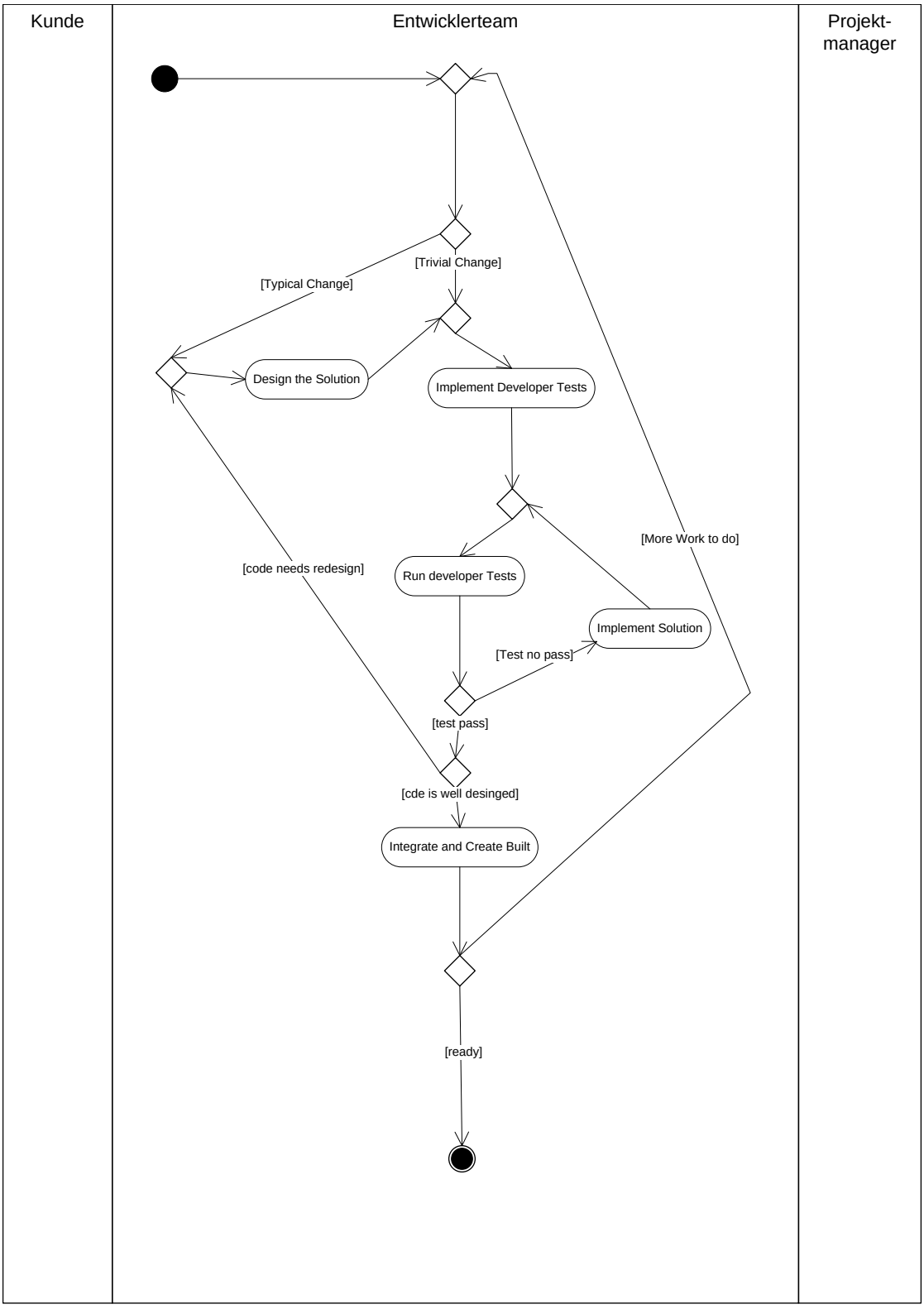

Abbildung 4.21: Open UP mit UML modelliert - Develop Solution Increment Prozess

4.3.2 Open UP mit BPMN-Notation modelliert

Wir stellen nun die Modellierung von Open Up mittels BPMN dar. Der Main Prozess, welcher den Ablauf der Phasen zeigt, wird in Abbildung 4.22 dargestellt. Dieser stellt einen rein sequentiellen Ablauf dar.

Die einzelnen Phasen werden in den Abbildungen (Inception 4.23, Elaboration 4.24, Construction 4.25, Transition 4.26) dargestellt.

Um diese Phasen zu modellieren, konnten die Workflows der Eclipse Foundation [EF 5] nahezu eins zu eins umgesetzt werden.

Auf eine Abbildung der Rollen der einzelnen Personen im Entwicklerteam wurde verzichtet um den Prozess übersichtlicher gestalten zu können. Hätten wir diese Rollen modelliert, wäre es zu sehr vielen Parallelisierungen gekommen um die zeitgleiche Bearbeitung der einzelnen Aktivitäten darstellen zu können.

Wir haben alle einzelnen Aktivitäten als zugeklappten Unterprozess dargestellt, da jeder dieser Prozesse noch genauer modelliert werden könnte. Auch wenn wir hier nur noch für den Develop Solution Increment eine Notation angeben haben, da das Darstellen der restlichen Worflows auf die selbe Weise möglich wäre.

Die Verfeinerung des Develop Solution Increment wird in Abbildung 4.27 illustriert. Dieser hier steht exemplarisch für alle anderen Modellierungen dieser Verfeinerungsstufe, zu denen es von der Eclipse Foundation ebenfalls ausgearbeitete Workflows geben würde.

Die abgeschlossenen Meilensteine stellen wir mittels Datenobjekten dar.

Die Lanes für den Kunden und Projekt Manager werden nur im Mainprozess benötigt. Mit der eigentlichen Entwicklungsarbeit in den Verfeinerungsstufen haben diese jedoch nichts mehr direkt zu tun.

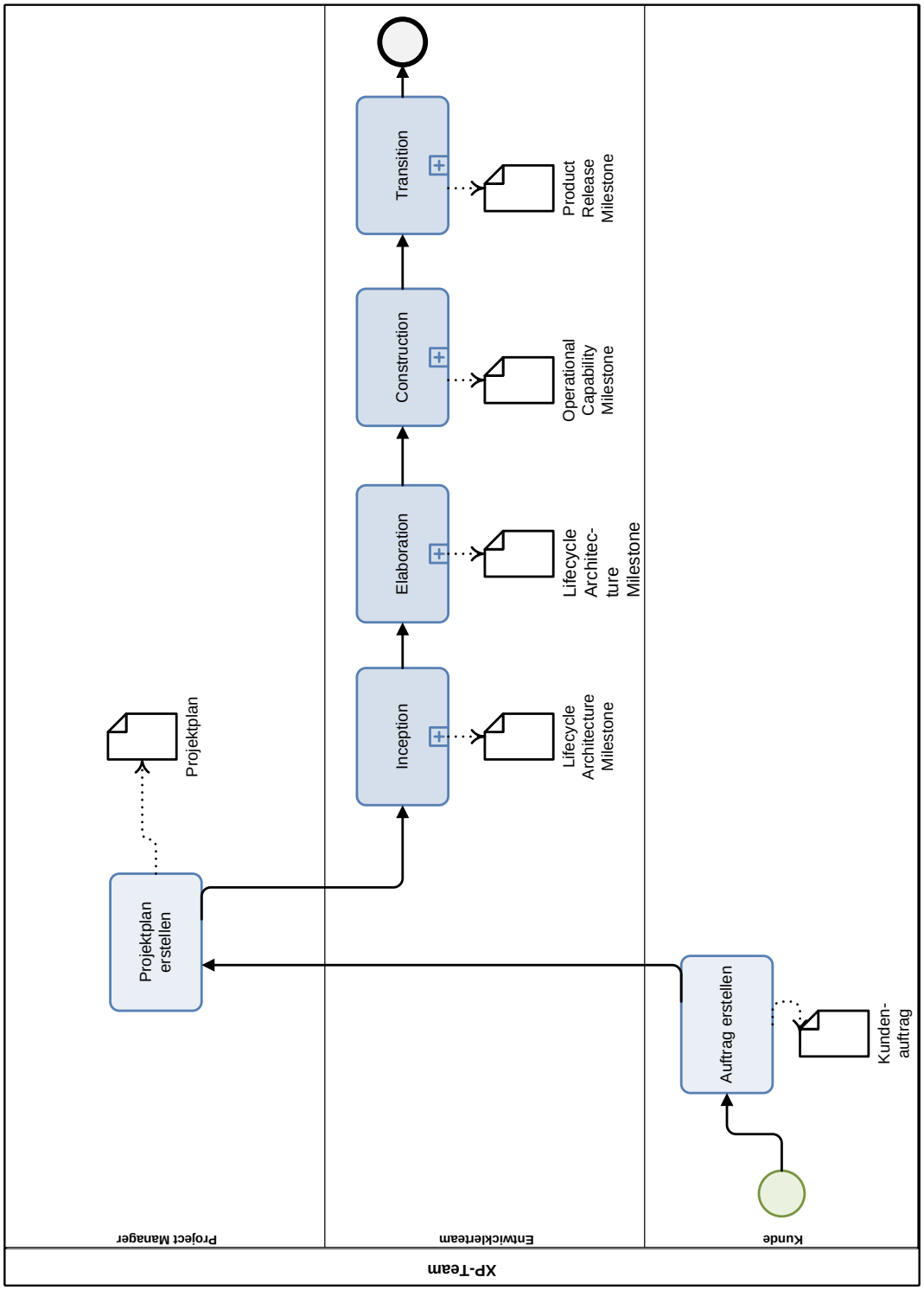

Abbildung 4.22: Open UP mit BPMN modelliert - Main Prozess

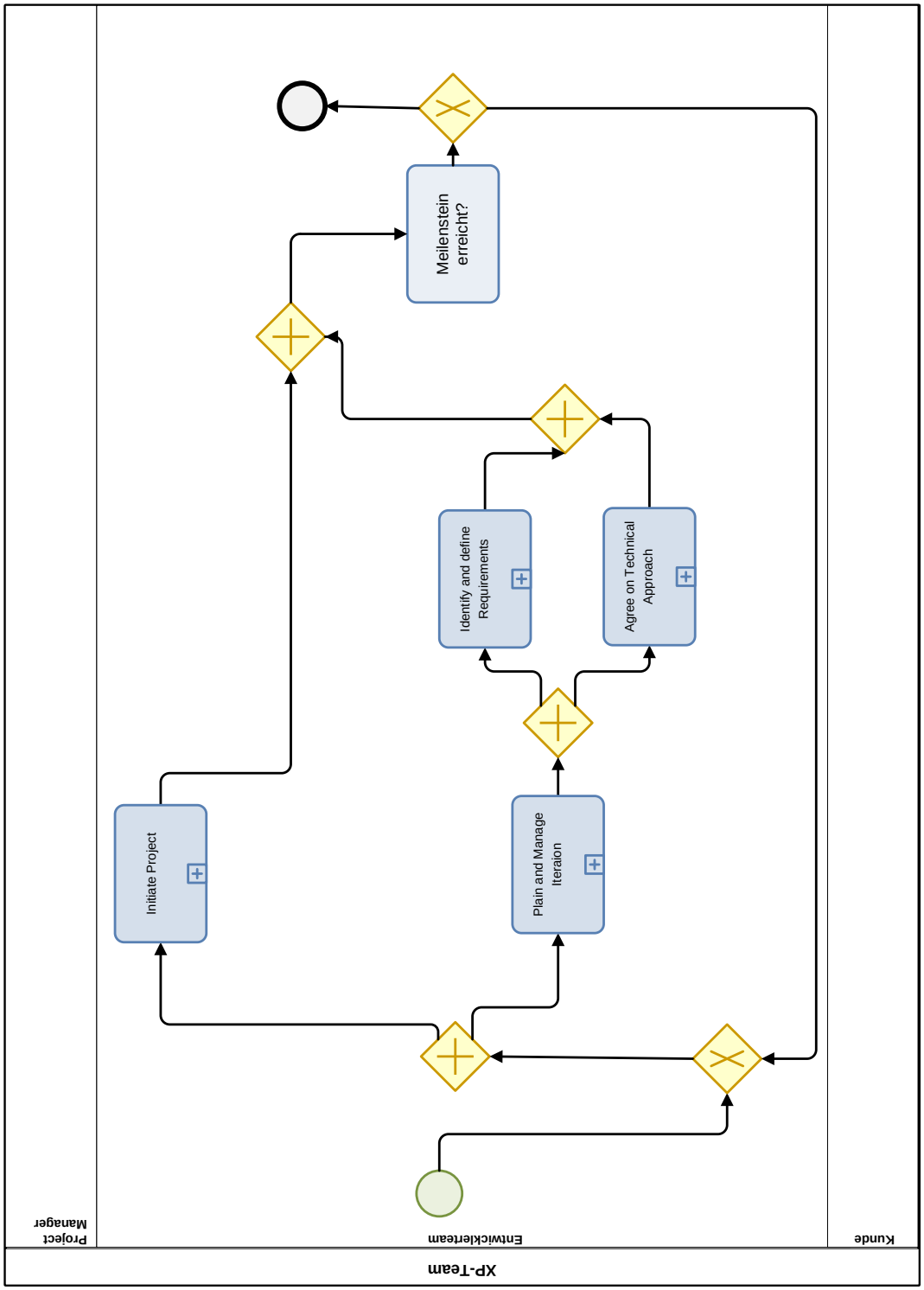

Abbildung 4.23: Open UP mit BPMN modelliert - Inception Prozess

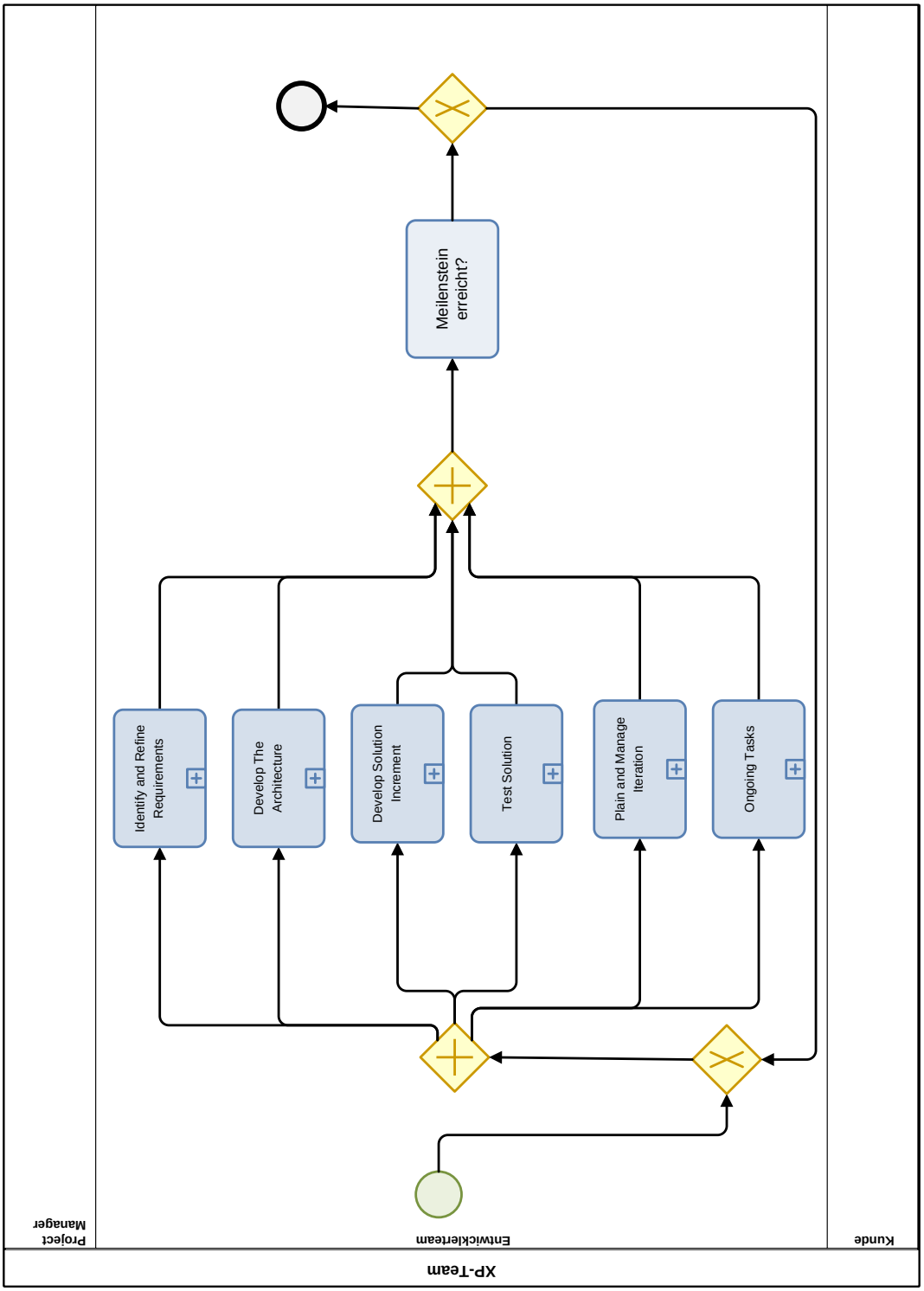

Abbildung 4.24: Open UP mit BPMN modelliert - Elaboration Prozess

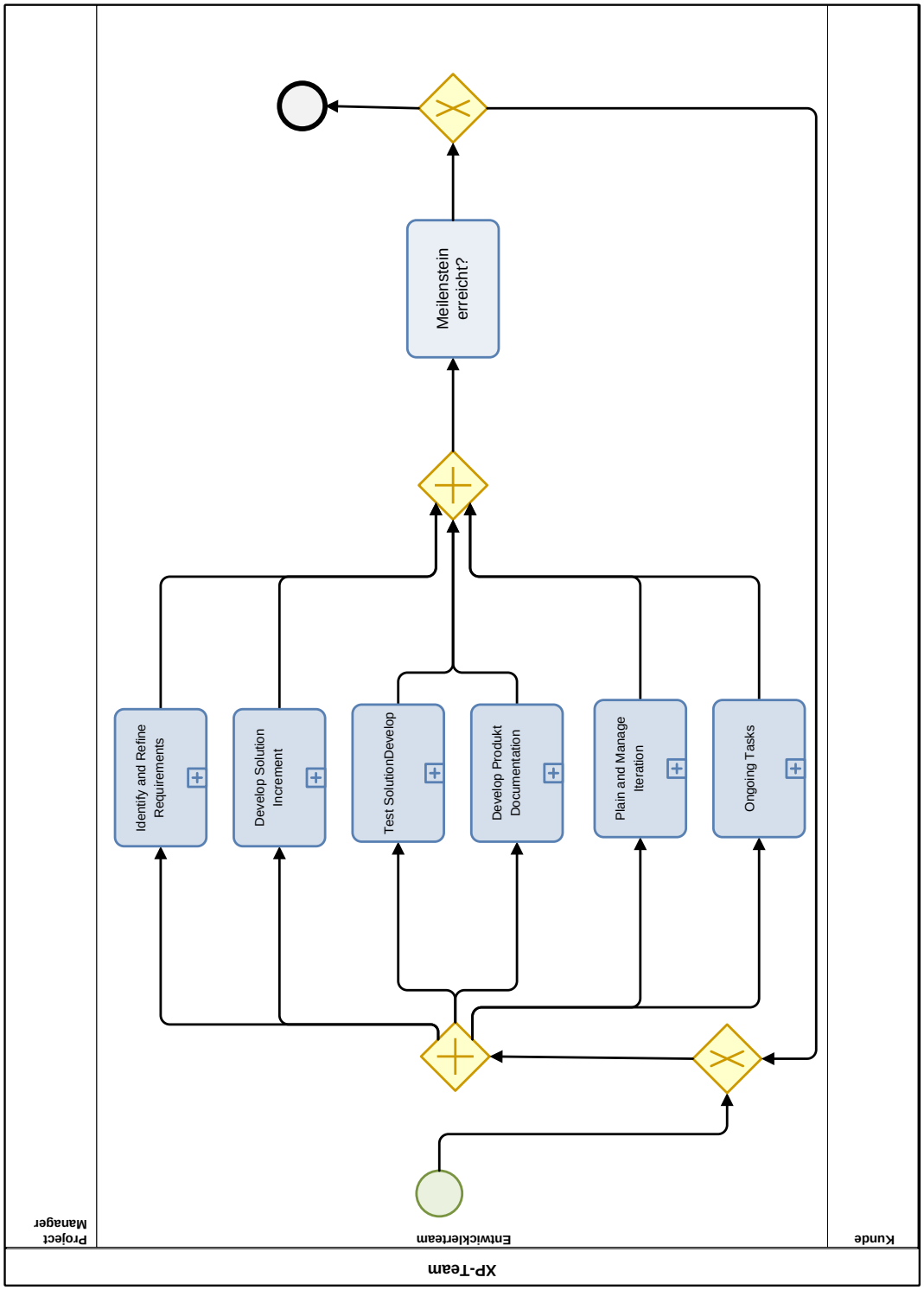

Abbildung 4.25: Open UP mit BPMN modelliert - Construction Prozess

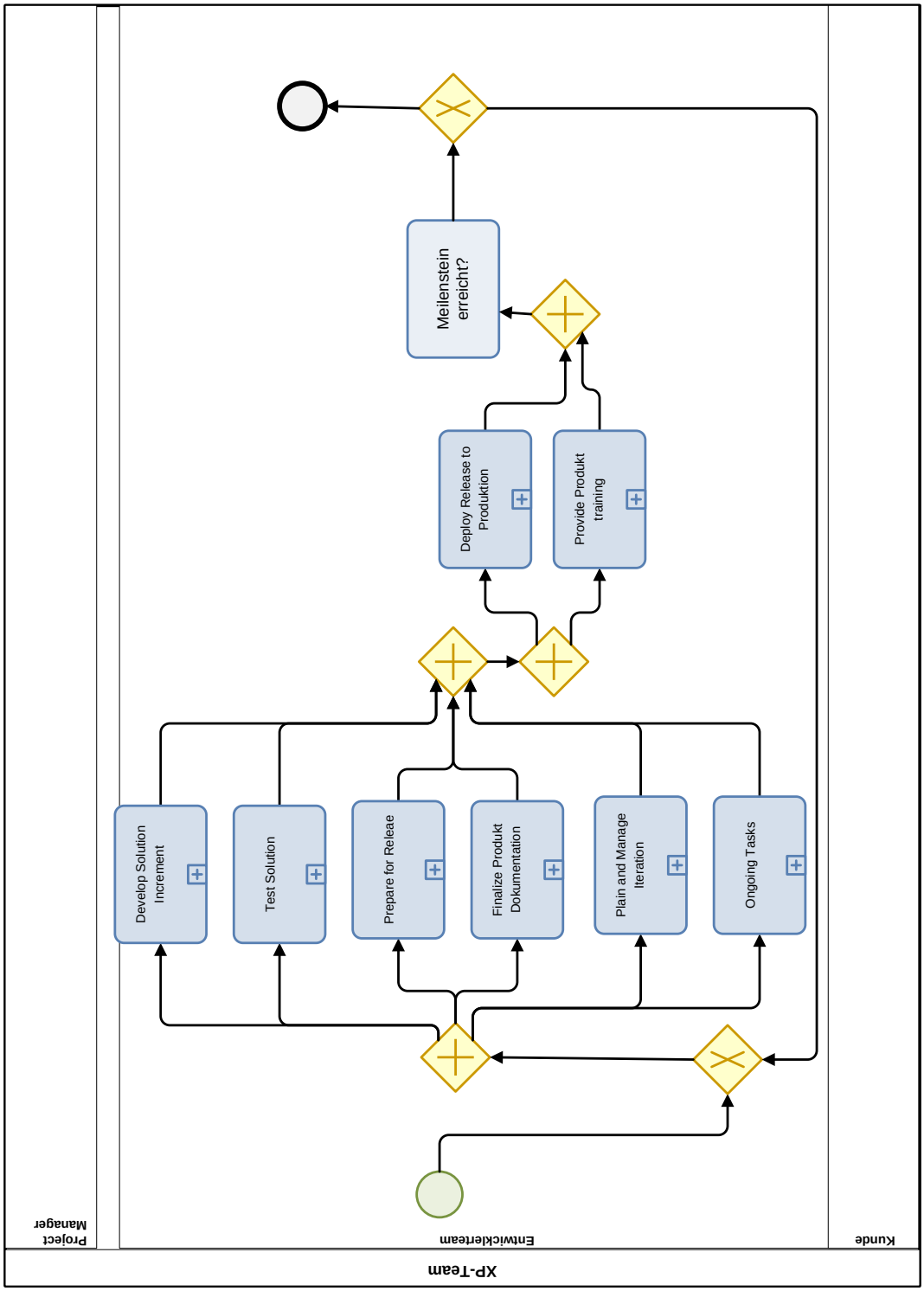

Abbildung 4.26: Open UP mit BPMN modelliert - Transition Prozess

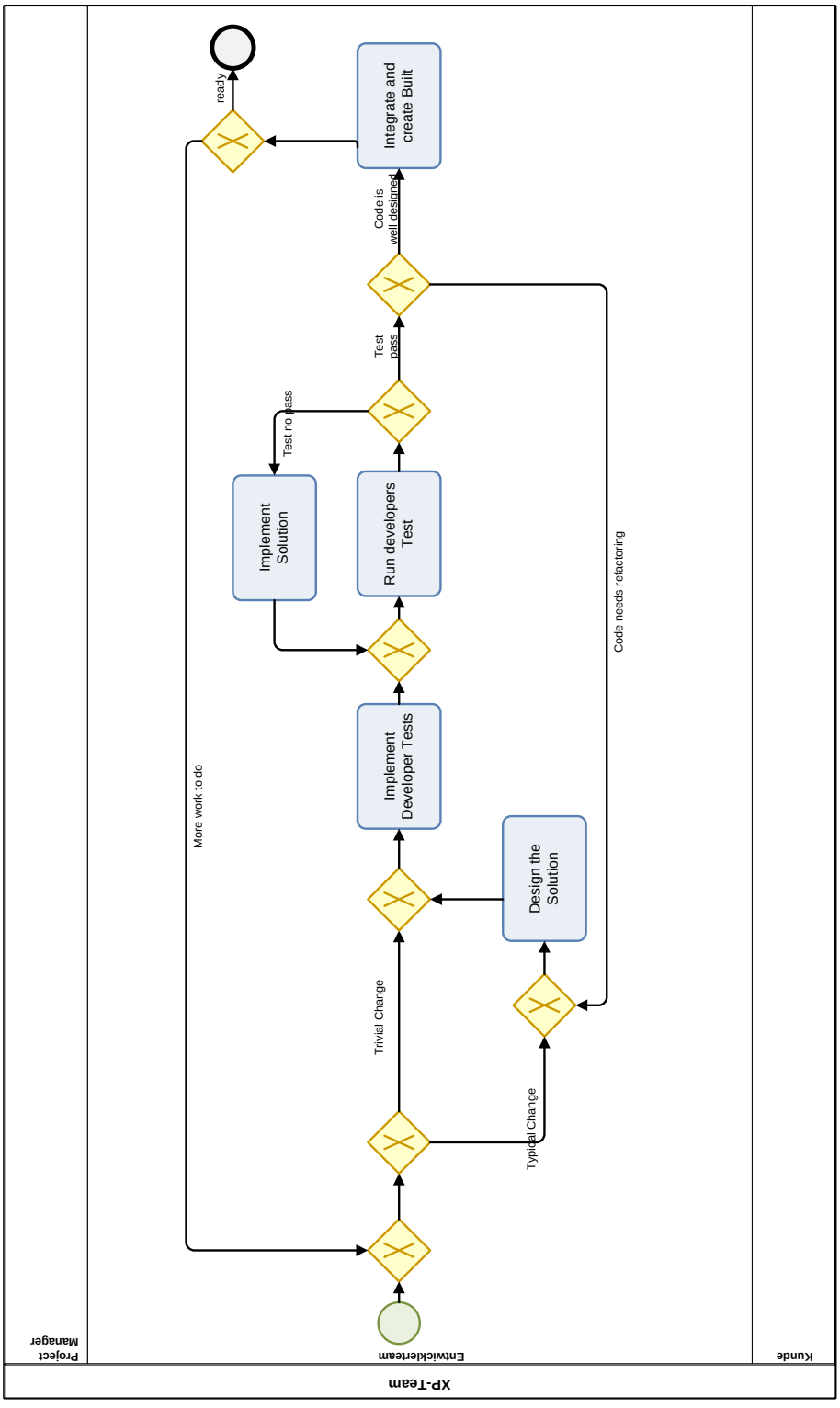

Abbildung 4.27: Open UP mit BPMN modelliert - Develop Solution Increment Prozess

4.3.3 Open UP mit EPKs modelliert

Zuletzt modellieren wir den Open UP Prozess mittels EPKs. Der Main Prozess ist in Abbildung 4.28 zu sehen. Dieser stellt einen rein sequentiellen Ablauf dar. Die Iterationen sind erst in den Verfeinerungen enthalten.

Die einzelnen Phasen werden in den Abbildungen (Inception 4.29, Elaboration 4.30, Construction 4.31, Transition 4.32) dargestellt.

Zu bemerken ist noch, dass bei einem Aufruf einer Verfeinerung bei einer EPK in der Verfeinerung stets das zuvorkommende und nachfolgende Ereignis in der Verfeinerung ein weiteres Mal dargestellt wird, Dadurch werden diese Prozesse in der Darstellung noch weiter aufgeblasen.

Auf eine Darstellung der Rollen der einzelnen Personen im Entwicklerteam wurde verzichtet auch wenn dies hier ohne eine Parallelisierung darstellbar gewesen wäre, um die Vergleichbarkeit zu den anderen Notationen besser zu erhalten.

Die Verfeinerung des Develop Solution Increment wird in Abbildung 4.33 illustriert. Diese steht auch in dieser Modellierung stellvertretend für alle anderen Teilprozesse auf dieser Verfeinerungsstufe.

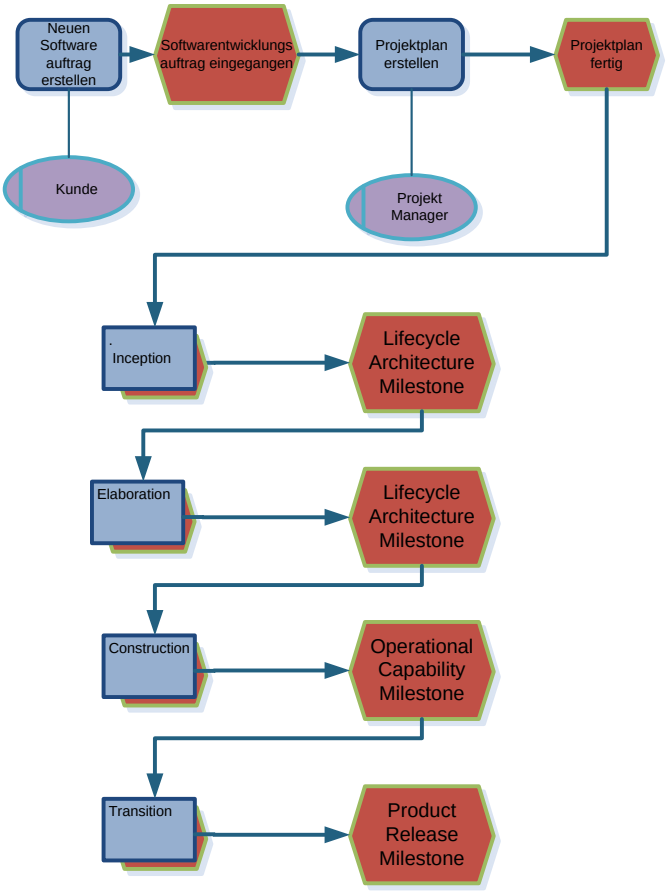

Abbildung 4.28: Open UP mit EPK modelliert - Main Prozess

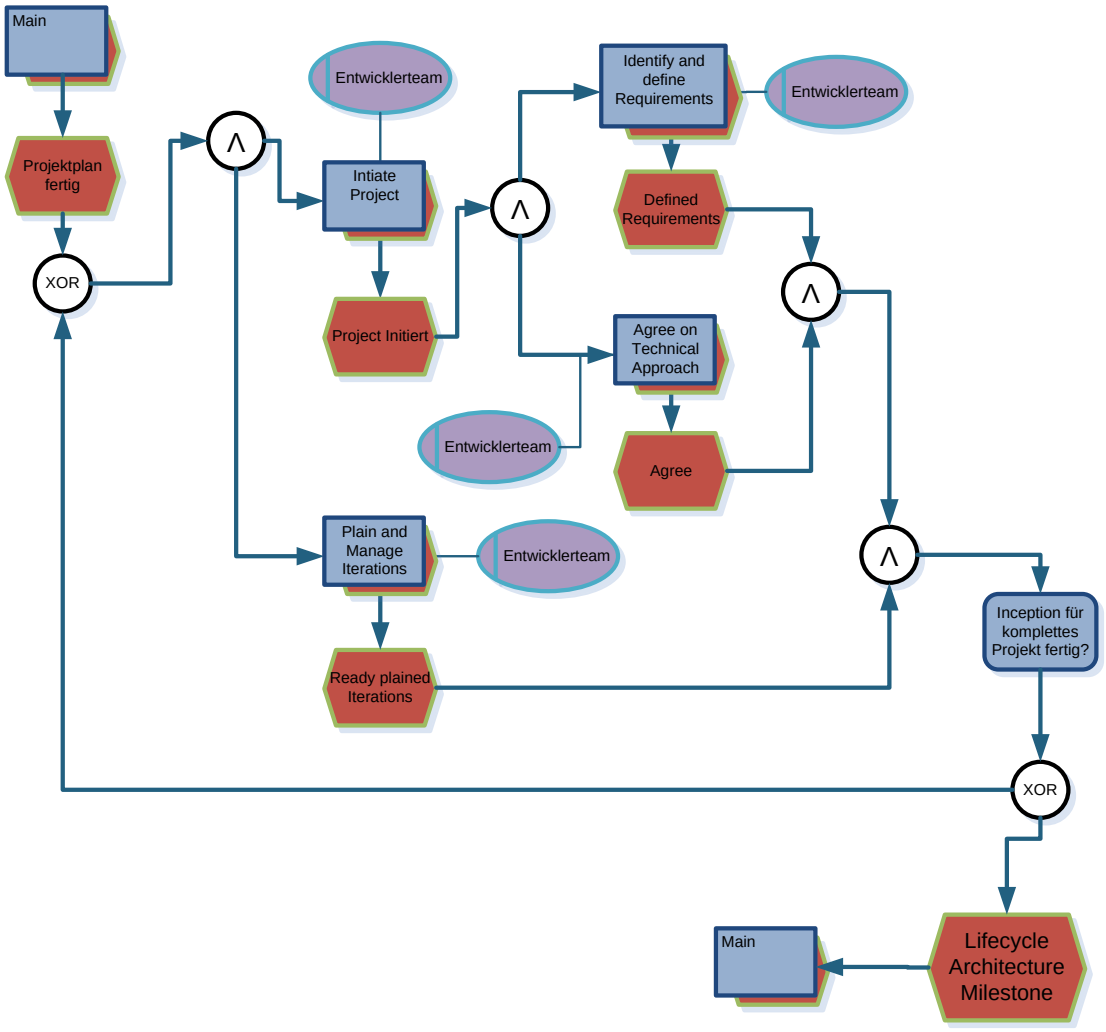

Abbildung 4.29: Open UP mit EPK modelliert - Inception Prozess

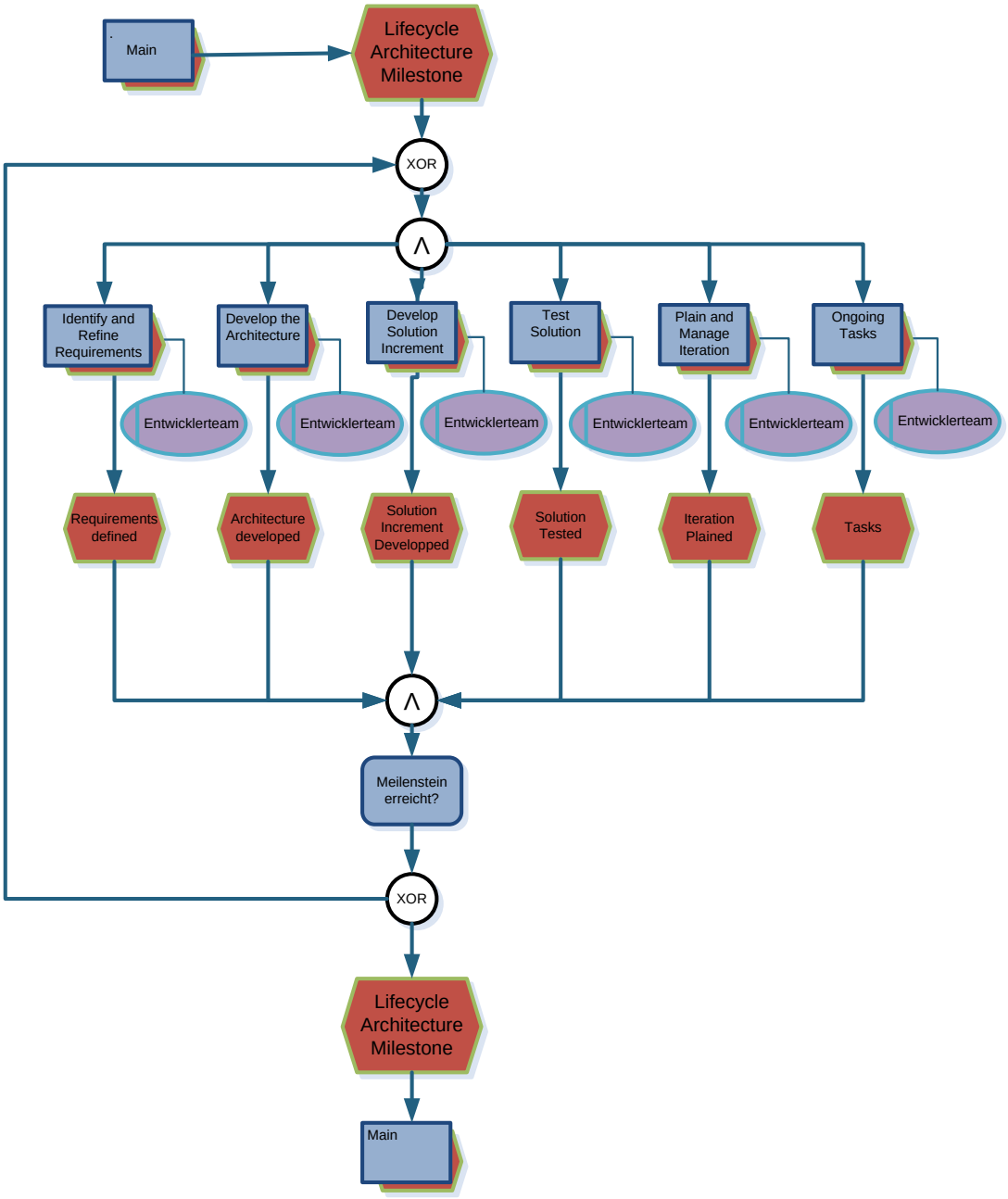

Abbildung 4.30: Open UP mit EPK modelliert - Elaboration Prozess

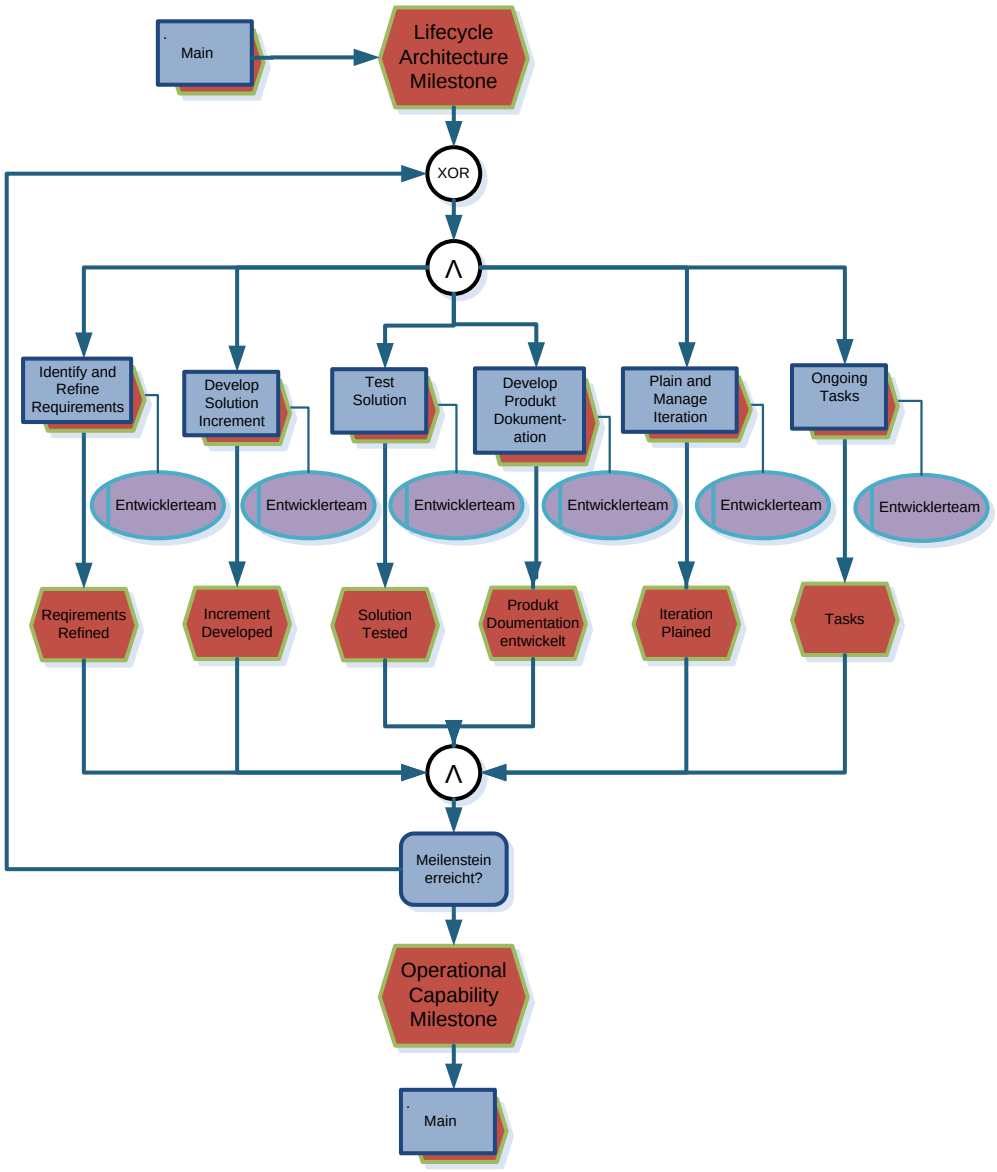

Abbildung 4.31: Open UP mit EPK modelliert - Construction Prozess

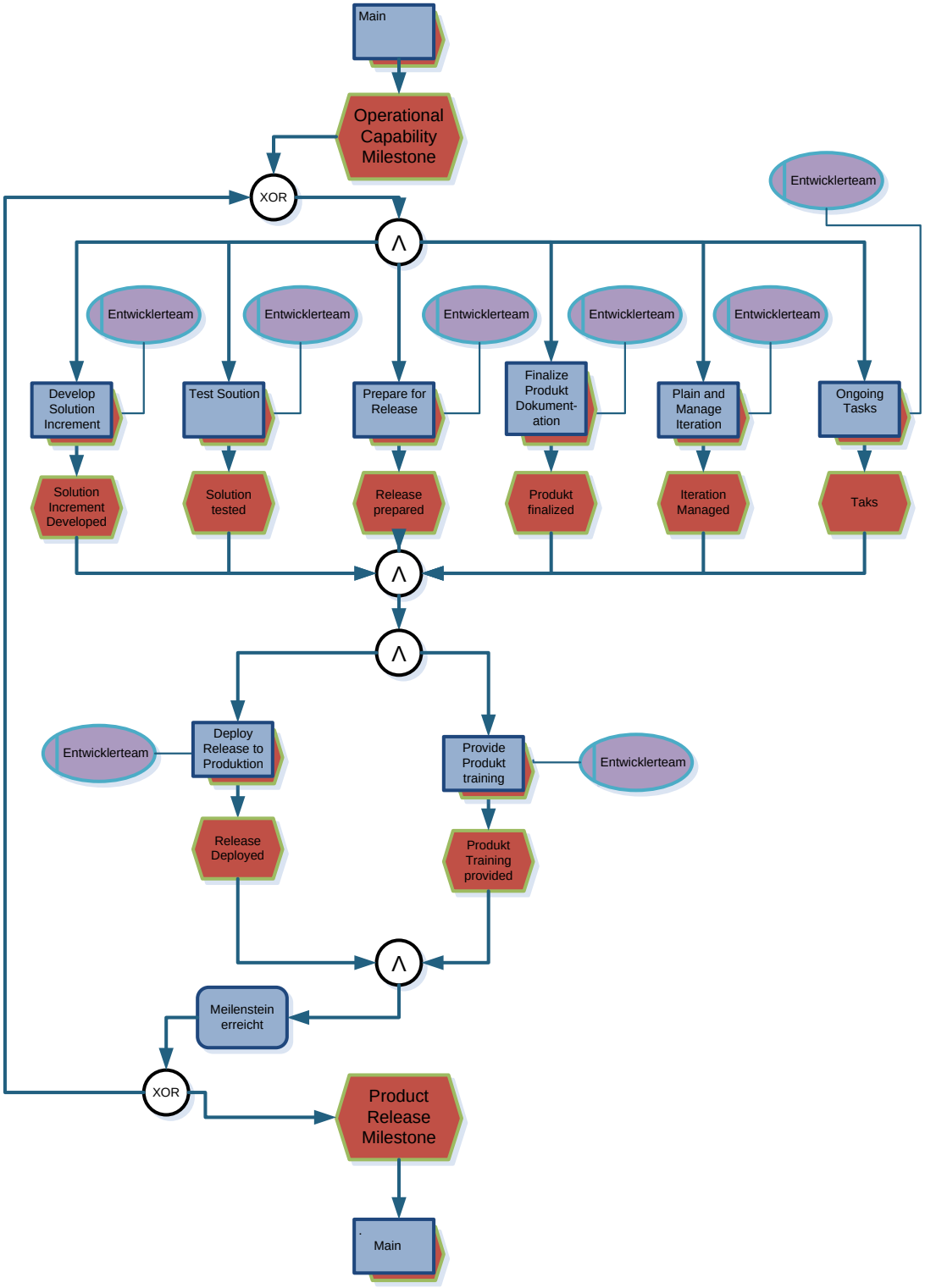

Abbildung 4.32: Open UP mit EPK modelliert - Transition Prozess

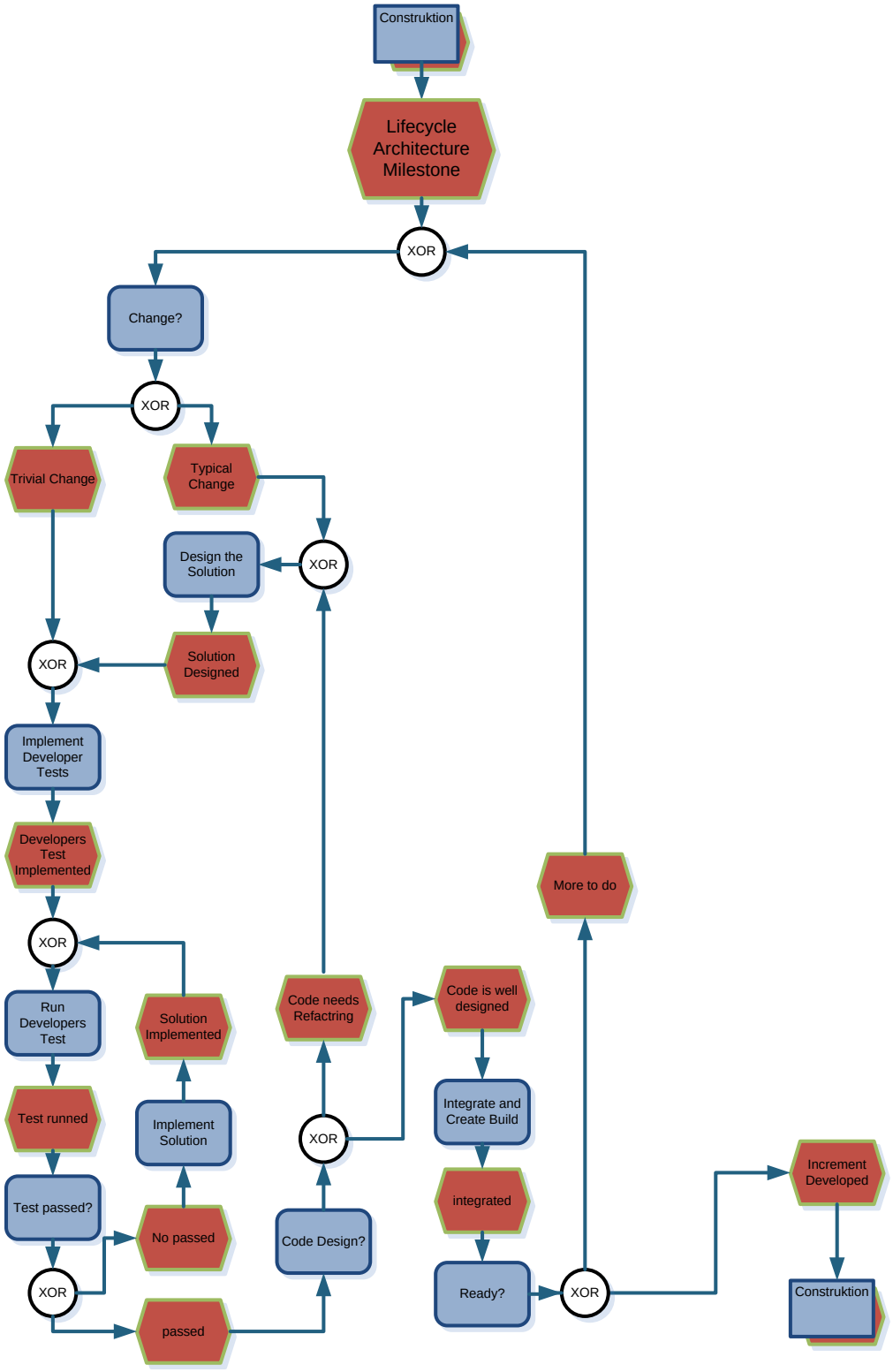

Abbildung 4.33: Open UP mit EPK modelliert - Develop Solution Increment Prozess

5

Diskussion der Ergebnisse

In diesem Kapitel werden die Ergebnisse aus Kapitel 4. diskutiert. Es wird unter anderem auf die Eignung der einzelnen Notationen sowie auf die Unterschiede der einzelnen Prozesse eingegangen.

5.1 Vor- und Nachteile der Notationen

In diesem Abschnitt werden die einzelnen Modellierungsnotationen auf ihre Eignung hin zur Modellierung von iterativen Softwareentwicklungsprozessen untersucht.

Eignung allgemein Allgemein kann ausgesagt werden, dass jede der Modellierungnotationen unterschiedliche Stärken und Schwächen hat. Wir finden, dass keine der Notationen so richtig zu iterativen Softwareentwicklungsprozessen passt. Näheres jedoch in

den einzelnen Abschnitten zu den Notationen. Wir finden finden, es wäre eine Notation wünschenswert, welche aus allen drei Notationen positive Elemente aufgreift und durch Erweiterungen für iterative Softwareprozesse angepasst wird.

Allgemein kann ausgesagt werden:

Unterprozesse sind eines der wichtigsten Hilfsmittel einer übersichtlichen Modellierung. Stellt man sich vor, einen Prozess wie den Open UP in einem Diagramm zu modellieren, weiß man, dass sobald die Prozesse größer werden, man ohne Unterprozesse nicht mehr sinnvoll arbeiten kann.

Datenobjekte sinnvoll zu verwenden stellt in allen drei Modellierungsnotationen ein Problem dar. Hierbei stellt sich die Frage, wie detailliert werden Dokumente und Daten abgebildet? Wie behandelt man optionale Daten. Wie Unterscheidet man Daten die zum Softwareprodukt gehören von Daten die zur Organisation des Prozesses dienen? Diese Probleme lassen sich zurückführen auf die vorgeschlagenen Notationen sowie den ungenauen Angaben der Prozesse wie mit Daten verfahren werden soll.

5.1.1 UML Aktivitätendiagramme

Bei der Betrachtung der UML Aktivitätendiagramme fällt zuerst auf, dass diese wirr wirken. Es ist definitiv nachteilig, dass es nicht möglich ist gebogene Verbindungslinien darzustellen.

Des weiteren ist das Konzept der Aktivitätenbereiche um verschiedene Nutzerrollen darzustellen nicht besonders geschickt. Soll ein und dieselbe Aktivität durch mehrere Benutzerrollen gemeinsam bearbeitet werden, so ist dies nur darstellbar, indem der Prozess zuerst parallelisiert und die Aktivität mehrfach für jede Rolle dargestellt wird. Dies führt jedoch Erstens zu unübersichtlichen Diagrammen und Zweitens noch zu folgendem Gedankenspiel: Was passiert, wenn die Darstellung verwendet werden soll um den Prozess ausführbar zu machen? Es hätte zur Folge, dass die Subaktivität mehrfach gestartet wird und zwar aus jeder Rolle heraus in der sie notiert ist. Dies ist aber nicht das Ziel der Darstellung, da die Aktivität lediglich einmal gestartet werden soll, allerdings daran mehrere Rollen beteiligt werden sollen.

Auch die Darstellung von Datenelementen ist nicht positiv gelöst, da sie in den Aktivitätsfluss integriert ist. Besser wäre es, wenn einfach durch eine Notation angegeben werden könnte, woraus Daten gelesen und wohin Daten geschrieben werden ohne diese direkt in den Fluss zu integrieren.

Ebenfalls schlecht gelöst finden wir die Entscheidungsknoten. Sie ermöglichen kein schnelles Verstehen der Diagramme wenn Entscheidungsbedingungen mittig an den ausgehenden Pfaden des Entscheidungsknoten notiert werden. In unserem Falle kommt noch die schlechte Darstellungsmöglichkeit mit Visio hinzu. Visio ist das Tool das zum Erstellen der Diagramme verwendet wurde. Evtl. können dies andere Tools besser darstellen. Eine gute Lösung wäre es, wenn es Entscheidungsknoten geben würde in denen eine Frage und an den Ausgängen die entsprechende Bedingung notiert werden könnte.

Parallele Bearbeitung lässt sich sinnvoll darstellen, jedoch halten wir die Symbolwahl nicht sonderlich glücklich.

Des Weiteren fehlt eine Notation um optimale, nicht verpflichtende Vorgänge darzustellen wie z.B. die Teilnahme des Scrum Masters an der Daily Scrum.

Daher ist aus unserer Sicht die UML der Schlechteste der drei Ansätze um iterative Softwareentwicklungsprozesse darzustellen.

Desweiteren gibt es keine Syntax für Subprozesse, dies erschwert die Lesbarkeit enorm!

5.1.2 BPMN

Nach unserem Dafürhalten ist BPMN schon deutlich besser als die UML zur Darstellung geeignet, weil die Diagramme deutlich übersichtlicher zu lesen sind.

Jedoch auch in der BPMN haben wir durch das Konzept der Pools und Lanes dasselbe Problem wie mit der UML. Auch hier ist es nur durch Parallelisierung möglich der selben Teilaktivität mehrere Benutzerrollen zuzuweisen, was genau zu den selben Problemen wie schon bei den Aktivitätendiagrammen beschrieben führt.

Jedoch bleibt das Diagramm hier noch deutlich besser lesbar, weil die Notation der Symbole und Verbindungslinien deutlich besser gewählt ist, da sich unter anderem Konnektoren gebogen darstellen lassen.

Auch die dezent verwendeten Farben der Symbole sorgen für einen deutlich besseren Überblick,da so Funktionen, und Entscheidungen usw. beim Betrachten sofort getrennt werden können.

Die Verzweigungen sind nicht nicht optimal gelöst. Es gilt dasselbe wie auch bei der UML. Es wäre schön, wenn am Verzweigungsknoten eine Frage und an den Ausgängen die entsprechende Lösung notiert werden könnte. Dann wäre das verstehen eines Entscheidungsvorganges deutlich einfacher.

Die Verwendung von Daten ist recht gut gelöst. Optimieren könnte man dies noch indem man verschiedene Symbole für Datenelemente einsetzen könnte, welche darstellen ob es sich um Daten des entstehenden Produkts oder um Daten, welche für den Prozess benötigt werden handelt.

Das Konzept der Angabe von Aufgabentypen bei Aktivitäten könnte recht nützlich sein, wenn es passend zu dem Softwareprozess, passende Typen geben würde. So wären beispielsweise Typen wünschenswert, welche angeben ob es sich um Entwicklung am Softwareprodukt, Aktivitäten zur Prozessverbesserung oder der Planungs- und Verwaltungsaktivitäten handelt.

5.1.3 EPKs

EPKs werden leider aufgrund der Regelung, dass es zwangsweise immer eine Funktion und ein Ereignis im Wechsel geben muss sehr unübersichtlich. Da sie durch viele zusätzliche Elemente den ganzen Prozess umfangreicher erscheinen lassen als er in Wirklichkeit ist.

Ebenfalls sind Entscheidungen noch schlechter gelöst, da aus der vorangehenden Funktion und dem nachfolgendem Ereignis geschlossen werden muss, welcher Pfad verfolgt wird. Hierbei gilt selbiger Wunsch wie auch schon bei der UML und BPMN nach einem Entscheidungsknoten mit Bedingungen.

Sehr praktisch sind jedoch Benutzerrollen durch die Erweiterung der EPKs gelöst. Dies führt, da unnötige Parallelisierungen wegfallen können, zu einer sehr guten Lesbarkeit der Notation.

Sehr umständlich hingegen ist wiederum der Aufruf von Subprozessen, da leider bei jedem Aufruf eines Subprozess das zuvorkommende und das darauffolgende Ereignis erneut notiert werden muss.

Leider sind in den EPKs keine Notationen für Datenelemente vorgesehen.

5.2 Die Prozesse im Vergleich

Die Prozesse bieten alle recht ähnliche Anforderungen an eine Modellierung. In jedem der Prozesse müssen Schleifen und somit Entscheidungen modelliert werden. Jeder Prozess benötigt eine sinnvolle Darstellung von Rollen.

Es fällt jedoch auf, dass der Open UP sehr genau und detailliert in den einzellnen Workflows beschreibt was hier wie gemacht werden soll. Die anderen beiden Prozesse sind hier deutlich freier. Das heißt, der Open UP gibt eigentlich für jedes einzelne Szenario einen eigenen Workflow an, welcher mittels Unterprozessen abgebildet werden kann.

Der Scrum Prozess basiert auf der Annahme, dass zu Beginn der Entwicklung nicht klar ist, wie die fertige Lösung aussehen soll. Deswegen werden als Aktivitäten auch nur wenige Meetings angegeben, welche dann zur Planung oder Nachbetrachtung der Arbeit gedacht sind, was auch dazu führt, dass die einzelnen Entwickler ein hohes Maß an Eigenverantwortung tragen und im Sprint ohne Unterbrechung von außen arbeiten sollen. Ein Problem bei der Modellierung von Scrum ist des Weiteren, wie mit einem möglichen Sprintabbruch umgegangen werden soll. Wir haben uns entschieden, die Option eines Sprintabruches nicht zu modellieren, da dies die Diagramme unübersichtlicher gemacht hätte und es verschiedene Vorgehensweisen und Gründe hierzu gibt. Dies hätte dann verschiedene Versionen oder zumindest mehrere Entscheidungen im Diagramm benötigt.

Bei der Modellierung von Xtreme Programming wird deutlich, dass hiermit eher kleinere Softwareprojekte zu verwirklichen sind, da hier die Organisatorischen Rahmenbedingungen für große Teams fehlen.

5.3 Umsetzung der Anforderungen

Durch die Wahl der UML Aktivitätendiagramme, der BPMN und der EPKs wurden Notationen verwendet, deren Verwendung jedermann frei steht und durch zahlreiche kostenlose Tools modelliert werden können. Wie in Anforderung AM2 gefordert.

Ebenfalls genießen diese Tools eine hohe Verbreitung. BPMN ist Standard im Business Bereich. Die UML ein wichtiger Standard im Softwareengeneering und die EPKs bestechen durch ihr Alter und ihre Einfachheit. Wie in Anforderung AM1 gefordert.

Da die verwendeten Prozessmodellierungen einfach zu lesen sind, ist es nun auch Nicht- Informatikern möglich, den Ablauf sinnvoll nach zu vollziehen und die einzelnen Prozesse miteinander zu vergleichen. Wie in Anforderung AM3 gefordert.

Bezüglich der Vergleichbarkeit der Vorgehensmodelle kann man sagen, dass diese durch die Modellierung mittels BPMN, EPKs und UML Aktivitätendiagrammen auf jeden Fall verbessert wird, da es nicht mehr notwendig ist, sich in die speziellen Notation einzuarbeiten, in der die verschiedenen Modelle präsentiert werden. Nach den einzelnen Modellierungen sieht man schnell, wo sich die Modelle ähnlich sind und wo die drei Vorgehensmodelle komplett unterschiedlichen Ansätzen folgen.

Jedoch konnte keines der Modelle zu 100 Prozent korrekt abgebildet werden, da dies die Notationen nicht ermöglichen. Es konnte aber eine weitestgehend korrekte Modellierung geschaffen werden. Somit ist Anforderung AM4 teilweise Erfüllt.

Ebenfalls war es nicht möglich eine 100 prozentige Vollständigkeit der Modelle beim Modellieren zu erreichen. Jedoch sind alle wichtigen Aspekte modelliert. Daher ist die Anforderung AM5 nicht zu 100 Prozent erfüllt.

Die Prozesse wurden in allen Notationen nach denselben Richtlinien modelliert und sind somit direkt vergleichbar. Wie in Anforderung AV1 gefordert.

Jedoch konnte sich keine Notation als die Beste profilieren. Viel mehr bietet jede einzelne unterschiedliche Vorteile und es wäre eine Notation wünschenswert, welche die einzelnen positiven Elemente aufgreift. Somit wurden die Modelle auf eine Eignung als Standard wie in der Anforderung AV3 gefordert untersucht wenn sich auch keine Notation dafür profilieren konnte.

Die Übersichtlichkeit wurde verglichen, es wurde festgestellt, dass die UML Diagramme sehr unübersichtlich sind, die EPKs zu aufgebläht und die BPMN relativ übersichtlich sind. Somit wurde auch Anforderung AV2 erfüllt.

Ebenfalls wurde untersucht ob die Syntax der Notationen ausreichend ist und wo die Vorteile der Notationen gegeneinander liegen. Dies wurde im Abschnitt Diskussion gemäß der Anforderung AV4 dargestellt.

6

Ähnliche Arbeiten

In diesem Kapitel erfolgt ein Vergleich dieser Bachelorarbeit mit anderen Arbeiten, die thematische Ähnlichkeiten besitzen.

6.1 Ähnliche Arbeiten

Arbeiten die genau dasselbe Thema behandeln sind uns nicht bekannt. Jedoch gibt es ähnliche Arbeiten vor allem in Bezug auf die Verwendbarkeit der BPMN als Notation.

So beschäftigt sich zum Beispiel Nicole Menhorn mit der Analyse und Überführung von Softwareentwicklungsprozessen in die standardisierte BPMN Notation [Men14]. als Bachelorarbeit an der Universität Ulm. Sie untersucht die Eignung der BPMN-Notation und gibt sinnvolle Erweiterungen zur Darstellung von Scrum, Open UP und dem V-Modell XT an.

Ein weiterer Artikel der sich mit der Frage beschäftigt, ob die BPMN 2.0 Notation dazu geeignet ist, Softwareentwicklungsprozesse zu modellieren, ist der Artikel Software Processes with BPMN: An Empirical Analysis ([CO13]). Jedoch wurde hier nur der Open UP Process untersucht. Es wurden hierfür drei Fallstudien durchgeführt. Hierbei wurde deutlich, dass sich die BPMN zwar eignet aber dringend Erweiterungen benötigt. Der Unterschied zu dieser Bachelorarbeit liegt darin, dass nur der OpenUP und BPMN Gegenstand der Analyse sind. Es werden nur der Aufbau und die Ergebnisse der Fallstudien kurz vorgestellt und Schlüsse, basierend auf den Ergebnissen der Studien gezogen.

Der Blogbeitrag A BPMN view on Scrum von Sebastian Stein [Ste 2], befasst sich mit der Überführung des Scrum Prozesses in die BPMN 2.0 Notation. Sebastian Stein benutzt bei der Modellierung der Zuständigkeiten Pools und Lanes. Hierbei macht er auch auf das Problem aufmerksam, dass viele Aufgaben in Teamarbeit (z.B. vom Scrum Team und dem Product Owner) bearbeitet werden und dies in BPMN 2.0 nicht gut modellierbar ist. Er löst das Problem dadurch, dass er die Aufgabe jeweils der Lane der verantwortlichen Rolle zuordnet. Allerdings ist dann nicht mehr zu sehen, welche Rollen bei der Bearbeitung zusätzlich beteiligt sind.

Herr Mario Beyer und Herr Wolfgang Hesse untersuchen in ihrem Paper Einsatz von UML zur Software-Prozessmodellierung ([BH02]), die Eignung der UML zur Darstellung für Softwareprozesse, Sie verwenden im Gegensatz zu dieser Arbeit mehrere Diagrammtypen der UML und untersuchen einen anderen Prozess.

Weitere Arbeiten welche sich im weiteren Sinne mit dieser Thematik beschäftigt stammen von Gregor Gambow welcher sich viel mit dem Ausführbar machen von Prozessmodellen beschäftigt.

So beschäftigt er sich in dem Paper Ënabling Automatic Process-aware Collaboration Support in Software Engineering Projects"([GOR12a]) mit dem Modellieren und Ausführen von Prozessen mit dem Schwerpunkt von Kollaborationsaspekten. In dem Paper "Contextual Injection of Quality Measures into Software Engineering Processes"([GOR11b]) geht es mehr um die Verbindung mit Softwarequalitätsaspekten. In einem weiteren Paper ([GOR11a]) geht er dann verstärkt auf dynamische operationale Worklfows ein. In dem

Paper "Knowledge Provisioning: A Context-sensitive Process-oriented Approach Applied to Software Engineering Environments"([GOR12b]) geht es dann wiederum um das Modellierung und Ausführung von Prozessmodellen jedoch nun mit dem Schwerpunkt Wissensmanagement. Ebenfalls behandelt er in dem Paper Äutomated Software Engineering Process Assessment: Supporting Diverse Models using an Ontology"([GOR13]) die Bewertung von aufgeführten Prozessen.

7

Fazit und Ausblick

Dieses Kapitel zieht ein abschließendes Fazit und gibt einen Ausblick was in der Zukunft zu erwarten ist.

7.1 Fazit

Unser Fazit aus der Arbeit ist, dass keine der vorhandenen untersuchten Notationen sich wirklich zur Darstellung von iterativen Softwareentwicklungsprozessen eignet.

Jede der Notationen bietet Vorteile, aber auch z.T. große Nachteile. Die BPMN ist unserer Meinung nach am besten hierfür geeignet, die UML am Unbrauchbarsten. EPKs sind zwar an sich ein ganz guter Ansatz, aber aufgrund des enormen Aufblähens der Prozesse durch den vorgeschriebenen Wechsel von Funktionen und Ereignissen auch nicht der optimale Weg.

Es wäre wünschenswert, wenn sich eine Notation entwickeln ließe, welche Konzepte aus allen drei Notationen aufgreift und diese zusätzlich erweitert.

So wäre es beispielsweise gut, die grundlegende Notation der BPMN zu verwenden, allerdings statt der Pools und Lanes die Angabe von Rollen wie in den EPKs zu übernehmen. Neue Symbole für Verzweigungen mit klar beschriebenen Bedingungen sollten hinzugefügt werden. Die Darstellung von Aktivitäten so zu ermöglichen, dass daraus ersichtlich wird ob es sich um Entwicklungsarbeit, Management oder Prozessverbesserung handelt.

7.2 Ausblick

Da Softwareentwicklung ein teurer aber sehr wichtiger Zweig der Wirtschaft ist und es für Unternehmen sehr kostenintensiv ist, muss daher auch fundiert entschieden werden können welcher Softwareentwicklungsprozess der Richtige für die Aufgabenstellungen ist. Es ist zu vermuten dass sich hierfür eine standardisierte Notation zur Darstellung bilden wird. Prinzipiell stellt sich nur die Frage, wann und von wem die Initiative dazu ausgeht, denn der Bedarf wäre bereits jetzt vorhanden.

Sollte sich jedoch eine Notation entwickeln, so wäre es sehr wichtig dass diese sich schnell verbreitet und zum Standard wird, denn geschiet dies nicht gibt es nur eine weitere Notation im derzeit verwendeten Dschungel und es würde keine bessere Übersicht sondern eher noch mehr Chaos entstehen.

Der guter Weg wäre wohl evtl. wenn die UML einen Standard für die Notation definieren und diesen dann auch verwalten würde, oder sich ein ähnliches Gremium finden würde welche die Notation standardisiert.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[All09] ALLWEYER, T.: *BPMN 2.0 Business Process Model and Notation. Einführung in den Standard für die Geschäftsprozessmodellierung.* Books on Demand, 2009

[BH02] BEYER, M. ; HESSE, W.: *Einsatz von UML zur Software-Prozssmodellierung*. GI-Softwaretechnik-Trends Bd. 22, Heft 1, 2002

[blo 5] BLOG.MILSYSTEMS.DE: *Softwarefehlerkosten*. `http://blog.milsystems.de`. Version: 2014, Mai 5

[bpm12] BPMB.DE: *BPMN 2.0 - Business Process Model und Notation*. `http://www.bpmb.de/images/BPMN2_0_Poster_DE.pdf`. Version: 2014, Mai 12

[CO13] CAMPOS, A. ; OLIVEIRA, T.: *Software Processes with BPMN: An Empirical Analysis.* Springer, 2013

[EF 5] ECLIPSE-FOUNDATION: *Open UP*. `http://epf.eclipse.org/wikis/openup/`. Version: 2014, Mai 5

[Glo11] GLOGER, B.: *Scrum: Produkte zuverlässig und schnell entwickeln.* Hanser Verlag, 2011

[GOR11a] GRAMBOW, G. ; OBERHAUSER, R. ; REICHERT, M.: Contextual Generation of Declarative Workflows and their Application to Software Engineering Processes. In: *Int'l Journal on Advances in Intelligent Systems* 4 (2011), Nr. 3&4, S. 158–179

[GOR11b] GRAMBOW, G. ; OBERHAUSER, R. ; REICHERT, M.: Contextual Injection of Quality Measures into Software Engineering Processes. In: *Int'l Journal on Advances in Software* 4 (2011), Nr. 1&2, S. 76–99

[GOR12a] GRAMBOW, G. ; OBERHAUSER, R. ; REICHERT, M.: Enabling Automatic Process-aware Collaboration Support in Software Engineering Projects. In: *Selected Papers of the ICSOFT'11 Conference*. Springer, 2012 (Communications in Computer and Information Science (CCIS) 303), S. 73–89

[GOR12b] GRAMBOW, G. ; OBERHAUSER, R. ; REICHERT, M.: Knowledge Provisioning: A Context-sensitive Process-oriented Approach Applied to Software Engineering Environments. In: *7th Int'l Conf. on Software Paradigm Trends (ICSOFT'12)*, 2012, S. 506–515

[GOR13] GRAMBOW, G. ; OBERHAUSER, R. ; REICHERT, M.: Automated Software Engineering Process Assessment: Supporting Diverse Models using an Ontology. In: *Int'l Journal on Advances in Software* 6 (2013), July, Nr. 1 & 2, S. 213–224

[hig15] HIGHSCORE.DE: *Der moderne Softwareentwicklungsprozess mit UML*. `http://www.highscore.de/uml/`. Version: 2014, Mai 15

[Hor23] HORN, T.: *Vorgehensmodelle zum Softwareentwicklungsprozess*. `http://www.torsten-horn.de/techdocs/sw-dev-process.htm#XP`. Version: 2014, Mai 23

[Men14] MENHORN, N.: *Analyse und Überführung von Softwareentwicklungsprozessen in die standardisierte BPMN Notation. Bachelor Thesis*. Universität Ulm - Fakultät für Ingenieurwissenschaften und Informatik, Institut für Datenbanken und Informationssysteme, 2014

[Pic09] PICHLER, R.: *Scrum – Agiles Projektmanagement erfolgreich einsetzen.* D.Punkt Verlag, 2009

[Rei13] REICHERT, M.: *Course on Business Process Management (WS 2013/14).* Universität Ulm - Fakultät für Ingenieurwissenschaften und Informatik, Institut für Datenbanken und Informationssysteme, 2013

[SB08] SCHWABER, K. ; BEEDLE, M.: *Agile Software Development with Scrum*. PEARSON STUDIUM, 2008

[Scr10] SCRUM.KOMPAKT.DE: *Grundlagen des Projektmanagements » Extreme Programming (XP)*. http://www.scrum-kompakt.de/grundlagen-des-projektmanagements/extreme-programming-xp/. Version: 2014, Mai 10

[SS13] SCHWABER, K. ; SUTHERLAND, J.: *Der Scrum Guide™*. Scrum.org, 2013

[Ste 2] STEIN, S.: *A BPMN view on Scrum*. http://www.ariscommunity.com/users/sstein/2010-08-09-bpm-view-scrum. Version: 2014, Mai 2

[Wes 8] WESTPHAL, F.: *Extreme Programming*. http://www.frankwestphal.de/ExtremeProgramming.html. Version: 2014, Mai 8

[wik15a] WIKIMEDIA.ORG: *Ablauf Open UP*. http://upload.wikimedia.org. Version: 2014, Mai 15

[wik15b] WIKIPEDIA.ORG: *Wikipedia Extreme programming*. https://en.wikipedia.org/wiki/Extreme_programming. Version: 2014, Mai 15

Name: David Rothmaier Matrikelnummer: 747355

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

David Rothmaier