Fraunhofer Institut
Experimentelles
Software Engineering

Ein modularisiertes Simulationsmodell zur Entscheidungsunterstützung in Software-Entwicklungsprozessen

Autoren:
Holger Neu
Thomas Hanne

IESE-Report Nr. 098.03/D
Version 1.0
Dezember 2003

Eine Publikation des Fraunhofer IESE

Das Fraunhofer IESE ist ein Institut der Fraunhofer-Gesellschaft.
Das Institut überträgt innovative Software-Entwicklungstechniken, -Methoden und -Werkzeuge in die industrielle Praxis. Es hilft Unternehmen, bedarfsgerechte Software-Kompetenzen aufzubauen und eine wettbewerbsfähige Marktposition zu erlangen.

Das Fraunhofer IESE steht unter der Leitung von
Prof. Dr. Dieter Rombach
Sauerwiesen 6
67661 Kaiserslautern

Abstract

In diesem Dokument wird ein modularisiertes ereignisdiskretes Simulations-Modell vorgestellt welches umfassend die Aktivitäten Design, Kodierung, Inspektion, Test und Rework modelliert. Die einzelnen Aktivitäten und Aufgaben im Modell wurden in Modulen separat modelliert. Zuerst erfolgt eine Beschreibung des Modells auf der Ebene der Modul. Danach wird eine Template zur Beschreibung der Module vorgestellt gefolgt von der Beschreibung der einzelnen Module sowie einigen Generischen Templates welche für Aktivitäten in einem ereignisdiskreten Modell verwendet werden können. Zur Verdeutlichung des Nutzens eines Simulationsmodells werden noch verschiedene Versuche durchgeführt und dokumentiert

Schlagworte: ereignisdiskrete Simulation, Inspektion, Modellierung, Entscheidungsunterstützung

Inhaltsverzeichnis

1 Ein ereignisdiskretes Simulationsmodell für Software-Entwicklungsprozesse.

In diesem Dokument wird ein Simulationsmodell zur Entscheidungsunterstützung in der Domäne Software Engineering beschrieben. Das Modell beschreibt einen Software-Entwicklungsprozess nach dem Wasserfall-Modell, welches folgende Schritte enthält.

- Design
- Kodierung
- Inspektion mit
  - Inspektionsvorbereitung
  - Inspektions-Meeting
  - Korrektur der gefundenen Fehler
- Testen
- Korrektur der beim Testen gefunden Fehler.

1.1 Struktur des Dokumentes

Das Dokument beginnt mit einem kurzen Abschnitt über die Simulation von Software-Entwicklungsprozessen und den damit verbundenen Besonderheiten. Auch die beiden gebräuchlichsten Techniken in der Domäne werden kurz vorgestellt.

Anschließend wird der Aufbau des Modells auf der Grundlage von funktionalen Blöcken beschrieben. Das Modell setzt sich aus verschiedenen kleineren Modulen zusammen, die in einem späteren Abschnitt genauer beschrieben werden.

Nachdem die Teile des Modells und deren Aufgaben beschrieben sind, werden in Kapitel 2.6 die Daten aufgelistet und es wird beschrieben, welchem Teil des Modells sie zugeordnet werden können.

Das Dokument muss nicht linear gelesen werden, abhängig von der Perspektive des Lesers genügt es selektiv bestimmte Teile zu lesen. Ist eine Allgemeine Einführung in das Model gewünscht ist die in Kapitel 2 und besonders 2.5 enthaltenen Information von Interesse. Soll das Modell in einem Kontext eingesetzt werden der keine großen Strukturelle Veränderungen erfordert kann der Leser durch die Beschreibung der Module in Kapitel 4 Teile des Modells auf seine Bedürfnisse anpassen. Die sog. Generic Templates in Kapitel 5 sind für Personen die Teile des Modells in anderen Kontexten für Ähnliche Aufgaben einsetzen möchten.

1.2 Simulation in Software Process Modeling

In den späten 80'ern und Anfang der 90'er wurden erste Modelle entwickelt, mit denen Software-Entwicklungsprozesse simuliert werden konnten. Eines der Modelle mit dem größten Einfluss auf die Forschung [AHM91] wurde mit System Dynamics (SD) entwickelt und modelliert auch die Entscheidungsstrategien eines Projektmanagers. Die Mehrzahl der Modelle, die im folgenden entwickelt wurden, verwendeten System Dynamics. Hierbei wurde aber meist auf die Modellierung des Entscheiders, also des Projektmanagers, verzichtet. SD-Modelle werden graphisch durch wenige Elemente beschrieben und bestehen im wesentlichen aus einem System von Differenzialgleichungen (DGL). Aus diesem Grund nennt man sie auch kontinuierliche Modelle, da es mit DGL nicht möglich ist, z.B. aus einer Menge von Aufgaben einzelne Aufgaben zu extrahieren und von anderen zu unterscheiden. Alle gleichartigen Objekte werden durch eine Zahl repräsentiert, welche als kontinuierliche Größe in den Gleichungen verändert wird. Somit wird pro Zeitschritt immer ein Bruchteil der Aufgabe als beendet gesehen. Weitere Informationen zu SD ist in [Ste00] zu finden.

Ereignisdiskrete Simulationsmodelle wurden erst in den letzten Jahren realisiert. Hier ist der Ansatz das alle gleichartige Elemente durch sog. Moving Units (MU) modelliert werden. Eine solches MU besitzt verschiedene vom Modellierer definierte Attribute. Mittels eines als Attribut gespeicherten Schlüssels lassen sich diese eindeutig unterscheiden. Während der Simulation kann eine MU weitere Attribute bekommen oder bestehende Attribute können gelesen und verändert werden. Die Berechung des nächsten Zustands des Modells erfolgt aufgrund des nächsten Ereignisses im Modell. Es können MUs definiert werden die Verschiedene Objekte oder Resourcen im Prozess beschreiben. Kombination dieser verschiedenen MUs ist möglich sowie ein Routing durch verschiedene Teile des Modells. Durch diese Möglichkeit können andere Fragestellungen mit ereignisdiskreten Modellen untersucht werden als mit System Dynamics Modellen.

Neben den beiden hier erwähnten Simulations-Techniken gibt es noch andere weniger weit verbreitete Techniken für die simulation von Software Entewicklungsprozessen, z.B., Petri-Netze, State-Based Modelle und Regelbasierte Modelle. Eine gute Übersicht über die verschiedenen Anwendungen und Vorraussetzungen ist in [KMR99] zu finden.

2 Architektur des Modells

Die Architektur und Funktionsweise des Modells wird ähnlich beschrieben wie die Module [Kapitel 3.3], mit Name, Aufgabe, Domäne, den Moving Units (MUs), der Funktionsweise und den Daten des Modells. Gegenüber der Beschreibung der Module wird die Beschreibung des Gesamtmodells auf einige Punkte verzichtet.

2.1 Name

Code Inspection Model

2.2 Aufgabe

Simulation eines Software-Entwicklungsprozesses zur Entscheidungsunterstützung bei der Einführung und Planung von Inspektionen.

2.3 Domäne

Das Modell wurde zur Simulation von Software-Entwicklungsprozessen in einer nicht speziell festgelegten Domäne entwickelt.

2.4 Moving Units (Items)

In dem Modell werden zwei unterschiedliche Arten von Moving Units (MU) verwendet .

- Code MU (CMU)
- Developer (DMU)

Die CMUs sind die zu erstellenden Code-Module. Die CMU besitzt zu Beginn Attribute, die ein Code-Modul eindeutig identifizieren (ID, Größe, Komplexität). Im weiteren Verlauf der Simulation werden neue Attribute hinzugefügt. Eine Beschreibung und Auflistung der Attribute finden sich in 2.6.

Die DMUs stellen die Entwickler dar. Die Entwickler sind durch Ihre Fähigkeiten und eine eindeutige Nummer zu identifizieren, welche als Attribute zur DMU gehören. Eine Beschreibung und Auflistung der Attribute finden sich in 2.6.

2.5 Funktionsweise des Modells

Das Modell beschreibt einen Software-Entwicklungsprozess vom Design bis zum Test der entwickelten Software-Module. Damit lässt sich das Modell in drei Komponenten einteilen, Design, Kodierung und Test. Zur Kodierungsphase kann noch eine Inspektion der Software-Module hinzugefügt werden. Jede dieser Phasen besteht aus einem oder mehreren funktionalen Blöcken. Zusätzlich dazu gehören noch Blöcke, welche infrastrukturelle Aufgaben übernehmen oder Management-Entscheidungen modellieren.

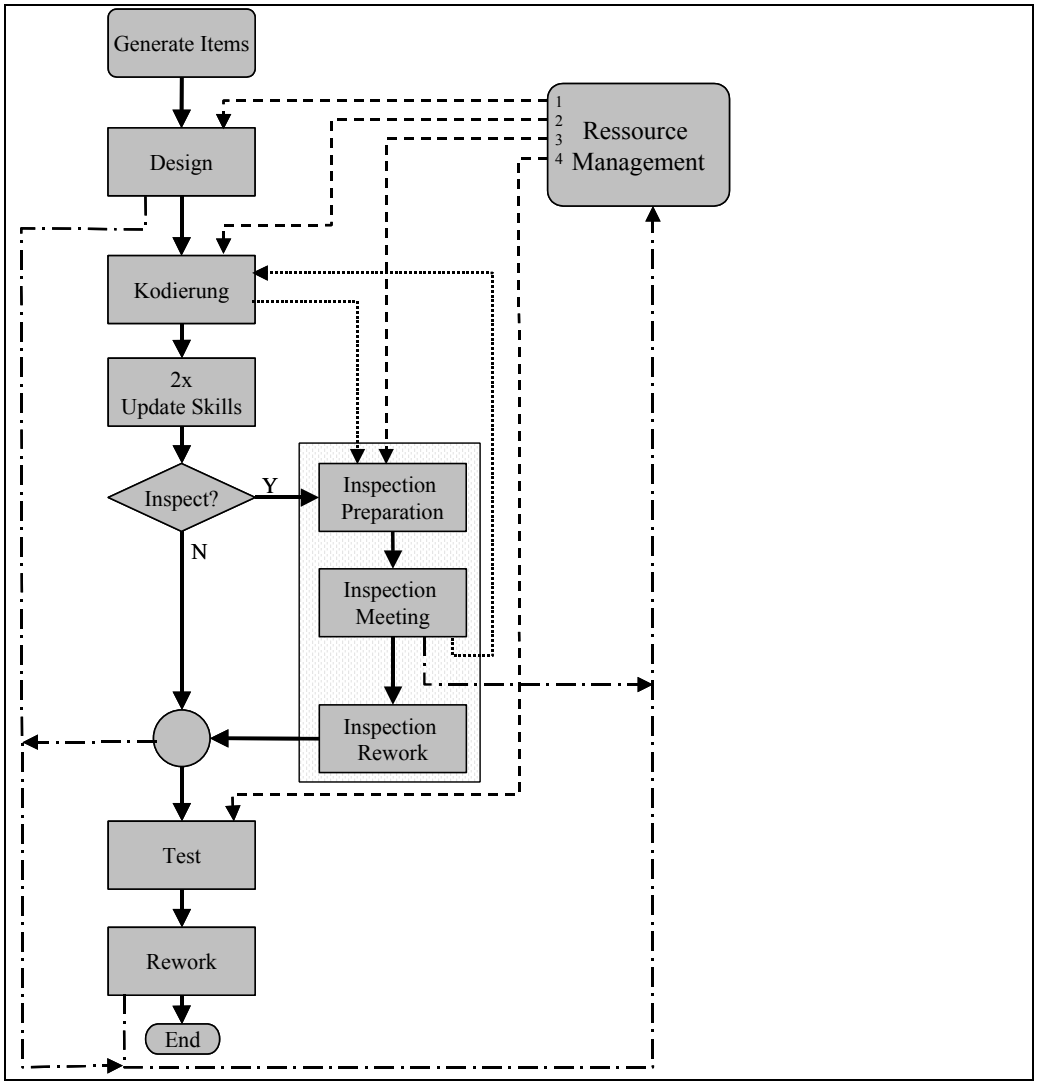


Fig 1: Module des Modells

In Fig 1 sind die Blöcke des Modells sowie die Verbindung zwischen diesen Blöcken dargestellt. Eine genauere Beschreibung der internen Funktionsweise der Elemente finden sich in Kapitel 4 inklusive der mathematischen Beziehungen und Verteilungen. An dieser Stelle soll nur das Zusammenwirken der einzelnen Blöcke und der Fluss der Moving Units (MU) durch das Modell beschrieben werden.

Code Moving Units:

Die Code Moving Units (CMU) folgen dem in Fig 1 dargestellten Verlauf der durchgezogenen Linie. Sie werden nach der Generierung in *Generate Items* im Block *Design* mit einer Entwickler-MU (DMU) zusammengeführt. Abhängig von der im *Design* berechneten Zeit werden die beiden MUs in dem Block gespeichert. Beim Verlassen des Blocks werden beide MUs wieder getrennt. Als Ergebnis-Daten werden die Zeiten sowie die IDs der MUs abgelegt.

Nachdem die CMUs den Block *Design* verlassen haben werden sie in einem Resource Pool im *Resource Management* gesammelt. Durch eine Priorisierung im *Resource Management* Block werden die CMUs erst in dem Kodierungs-Block bearbeitet, sobald Ressourcen (DMU) frei sind. Der *Resource Pool* verwaltet die DMUs nach einen einfachen FCFS (First Come First Serve) Algorithmus.

Im Block *Kodierung* wird wieder eine CMU mit einer DMU verbunden und verlässt nach der intern berechneten Zeit den Block *Kodierung* wieder. Die Zeit für die Kodierung kann durch eine Inspektion unterbrochen werden, in der der Entwickler nicht mehr an dem ihm zugeordneten Code-Modul arbeitet. Nach Beendigung der Inspektion kehrt er zurück und bearbeitet das Code-Modul weiter (fein gepunktete Linie in Fig 1). Bei Beendigung der Kodierung eines Code-Moduls wird entschieden, ob das Code-Modul einer Inspektion unterzogen wird. Dies geschieht in dem Block *Inspect?.* Dort werden aufgrund von Einstellungen von Parametern die kombinierten MUs in den Inspektionsblock geleitet. Wird keine Inspektion durchgeführt erfolgt die Trennung, die CMU wartet einem FIFO (First In First Out) Block bis Ressourcen (DMU) frei sind für den Test der Code-Module. Ergebnisdaten sind die Zeiten (Dauer, Start- und Endzeitpunkt) für die Kodierung sowie die eingeführten Fehler (Defects).

Bei Durchführung einer Inspektion wird die Anzahl der benötigten Inspektoren aus dem Block *Kodierung* abgezogen oder direkt aus dem Resource Pool genommen, sofern verfügbar. In der Inspektion werden alle Inspektoren (mehrere DMUs) und der Autor (die DMU welche mit der CMU zur Kodierung verbunden wurde) durch die Aktivitäten *Inspection Preparation* und *Inspection Meeting* geführt. Nach dem Meeting verlassen die Inspektoren den Inspektionsblock und werden zurück zur Fortführung der Kodierung oder in den Resource Pool geleitet. Der Autor entfernt in der *Inspection Rework* Aktivität noch die gefundenen Fehler und verlässt den *Inspektion* Block. Danach wird die CMU von der DMU getrennt, die DMU kehrt in den Resource Pool zurück.

Sobald Ressourcen frei sind (Priorität liegt auf den Phasen davor) werden die Code-Module (CMU) mit einem Entwickler (DMU) zusammengeführt um den *Test* durchzuführen. Im *Test* wird berechnet, wie lange der Test dauert, und die Anzahl der gefundenen Fehler sowie die verbleibenden Fehler. Für die Länge der berechneten Zeit verbleibt die MU in der Aktivität. Ergebnisdaten sind die Zeiten (Dauer, Start- und Endzeitpunkte) für den Test sowie die gefundenen und verbleibenden Fehler.

Im Anschluss wandern beide MUs in den *Rework* Block in dem die gefundenen Fehler beseitigt werden. Abhängig von der Anzahl der Fehler wird die Dauer der Aktivität berechnet und die MUs verbleiben so lange in der Aktivität *Rework*. Beim Verlassen der Aktivität werden die MUs CMU und DMU wieder aufgetrennt. DMU wird zurück in den Resource Pool gesendet und CMU verlässt das Modell da seine Aufgabe erfüllt ist. Die in diesem Block entstehenden Ergebnisdaten sind die Zeiten (Dauer, Start- und Endzeitpunkte). Die Anzahl der verbleibenden Fehler wird nicht mehr geändert

Developer Moving Units

Die Developer Moving Units (DMU) werden zum Anfang der Simulation in dem *Resource Management* Block generiert und initialisiert. Danach werden die DMU in einem internen Resource Pool gespeichert, aus dem sie zu den verschiedenen Blöcken abgerufen werden können. Als Scheduling-Algorithmus ist ein First Come First Serve (FCFS)-Algorithmus (implizit realisiert) mit einer Priorisierung der Aktivitäten in der Reihenfolge Design, Kodierung, Inspektion und Test.

Für den *Design* Block wird ein DMU aus dem Resource Pool angefordert und nach dem Beenden des Designs wieder in diesen zurückgeführt.

Ein DMU, welche in der Kodierungsaktivität ist kann durch eine Anforderung zur Inspektion aus dieser entfernt werden, um als Inspektor an der *Inspection Preparation* und dem *Inspection Meeting* teilzunehmen (fein gepunktete Linie in Fig 1). Nach dem Meeting wird die DMU wieder zurück in die Aktivität Kodierung geführt, dort verbleibt die DMU mit der ihr zugeordneten CMU bis die zuvor berechnete Kodierungszeit beendet ist. Eine solche Unterbrechung kann mehrfach während der Kodierung erfolgen. Nach dem Verlassen der Kodierung werden die Skills des Entwicklers, also die Skill Attribute der DMU, in *Update Skills* neu berechnet und gespeichert. Aufgrund der Auswahl im Block *Inspect?* wird die kombinierte MU aus DMU und CMU in den Inspektionsblock geführt oder aufgetrennt. Bei der Auftrennung wird der Entwickler (DMU) wieder zurück in den Resource Pool geleitet und steht für neue Aufgaben zur Verfügung.

Bei einer Überführung in die Inspektion wird die CMU, also das zu inspizierende Artefakt, vervielfältigt und mit jeder DMU, die aus dem Block *Kodierung* oder dem Resource Pool kommt, kombiniert. Diese wandern dann durch die Vorbe-

reitung und das Meeting der Inspektion, bevor die Inspektor-MUs (DMU) wieder zurück zu ihrem Ausgangspunkt geführt werden. Die DMU hat also zwei Rollen in dem Modell, einmal Entwickler und einmal Inspektor. Jeder Entwickler hat auch noch Skills als Inspektor die abhängig von der Zeit in der Vorbereitung der Inspektion verändert werden. Nach dem Meeting werden die identifizierten Fehler vom Autor, der DMU, welche zu Beginn des Kodierungs-Blockes mit der CMU verbunden wurde, entfernt. Hier werden auch nochmals die Skills, welche mit der Kodierung in Zusammenhang stehen, neu berechnet. Danach werden CMU und DMU wieder getrennt und verlassen dann den Inspektions-Block.

Die beide abschließenden Blöcke, Test und Rework sind so verbunden, dass ein Entwickler (DMU) beide Aktivitäten mit dem gleichen Artefakt (CMU) sequenziell durchläuft. Der Block *Modul Test* fordert eine Ressource (DMU) an, welche dann durch die beiden Blöcke *Modul Test* und *Test Rework* zusammen mit der zugeordneten CMU durchläuft.

2.6 Daten

Die MUs, welche einerseits die zu entwickelnden Code-Elemente und andererseits die Personen, welche in dem Prozess involviert sind, modellieren, müssen zu Beginn oder während des Simulationslaufes erzeugt werden. Dazu steht jeweils ein TeilModell zur Verfügung. Diese Teilmodelle (Generate Items und Resource Management) initialisieren die MUs. Dazu werden die Input-Variablen aus Tabelle 1 (Ausgangswerte für DMU) und Tabelle 2 (CMU) verwendet.

2.6.1 Input Variablen (für Person k)

Diese Variablen beschreiben die Ausgangswerte der personenbezogenen Skills zum Start der Simulation. Die Werte liegen zwischen 0 und 1.

Bezeichnung der Tabellenspalte	**Abk.**	**Verwendung**
Person Identifier	P_ID	Identifier für die Person
Coding Productivity Skill Baseline	$CPSB_k$	anfängliche Coding-Produktivität von Person k
Coding Quality Skill Baseline	$CQSB_k$	anfängliche Coding-Qualität von Person k
Preparation Productivity Skill Baseline	$PPSB_k$	anfängliche Inspektionsproduktivität von Person k für die Inspection Preparation
Defect Detection Skill Baseline	$DDSB_k$	anfängliche Inspektionsqualität von Person k
Domain Specific Skill	DSS	Skill in domain d (wird derzeit nicht benutzt)???
Time Pressure	TP_k	Zeitdruck für Person k (evtl. durch Team TP ersetzt)

Tabelle 1 : Skill-Ausgangswerte der Personen zum Initialisieren der DMU

2.6.2 Input Variablen (für Unit i)

Abhängig von den zur Verfügung stehenden Daten kann ein Teil (Größe und Komplexität der Kodemodule) der Input-Variablen generiert werden. Dazu kann im Block *Generate Items* mittels eines „Schalters" ausgewählt werden, ob die Generierung der Daten aktiv ist. Die Daten werden bei der Generierung in die Tabelle *Unit Input Table* geschrieben und stehen danach für weitere Experimente zur Verfügung. Generiert werden TLOC, ILOD, DLOC.

Werden die Daten nicht generiert, liest der Block die Daten aus der Tabelle. Daten aus vorhergehenden Projekten können in der Tabelle manuell eingegeben werden.

Die Verteilungen welche dem Generieren der Daten zugrunde liegen, sind in der Tabelle 3: General Distributions abgelegt.

Bezeichnung der Tabellenspalte	**Abk.**	**Verwendung**
Unit _ID	Unit_ID	Identifier
Total Lines of Code (Size)	$TLOC_i$	Gesamtgröße von Artefakt i
Inspected Lines of Code	$ILOC_i$	zu inspizierende Größe
Delta Lines of Code	$DLOC_i$	Größe des neuen bzw. geeänderten Codes
Complexity	$CPLX_i$	Komplexität von Unit i, um 1 verteilt
Planned Author	PA_i	Für Unit i geplanter Autor, -1 keine Planung
Author	A_i	Autor der Unit i bearbeitet
Inspection Technique	IT_i	
Number of Inspectors	NI_i	Anzahl der Inspektoren für dieses Modul

Tabelle 2: Unit Input Table

Die Spalten Author und Planned Author werden in dem vorliegenden Modell nicht genutzt. Hintergrund der Spalten war die Möglichkeit der Vorgabe von Autor und Codemodul Kombinationen für Scheduling-Versuche. Inspection Technique definert die verwendete Lesetechnik (Ad Hoc, Checklist Based oder Perspective Based Reading). In der Spalte Number of Inspectors kann die Anzahl der Inspektoren für ein Code-Modul vorgegeben. Bei einer Generierung der Anzahl mittels einer Verteilung (Verteilung in Tabelle 4: Allgemeine Input-Daten (General Input Data)) wird der generierte Wert in der Unit Input Tabelle gespeichert.

2.6.3 Allgemeine Verteilungen (General Distributions)

In der Tabelle General Distributions werden die Verteilungen, die im Modell verwendet werden, definiert. Die Verteilungen Nummer 10-15 werden für die Generierung der Input-Daten verwendet, sofern benötigt.

NR.	Bezeichnung	Abk.	Verwendung
1	Random Unit Size Factor	RUSF	Basisgröße der Größe einer Code Unit
2	Random Design Time Factor	RDTF	Verteilung der Aufgewendeten Zeit in der Phase Design
3	Random Coding Time Faktor	RCTF	Verteilung für die Berechnung der Kodierungszeit
4	Random Preparation Time Factor	RPTF	Verteilung für die Berechnung der Inspektionszeit (Preparation)
5	Random Rework Time Factor	RRTF	Verteilung für die Berechnung der Rework Zeit
6	Random Meeting Time Factor	RMTF	Verteilung für die Berechnung der Meeting Zeit
7	Random Defect Density Factor	RDDF	Verteilung für die Anzahl der im Coding eingeführten Fehler.
8	Random Defect Detection distribution	RDDD	Verteilung für die Anzahl der im Inspektionsmeeting gefunden Fehler
9	Random Unit Complexity Factor	RUCF	Verteilung der Komplexität bei der Generierung der Code Moving Units
10	Basic Size	BLOC	Basis Größe der einzelnen Units, hier kann eine Verteilung angegeben werden
11	Basic Delta Size	BDLOC	Basis Größe der Anzahl der veränderten Lines of code, auch hier kann eine Verteilung verwendet werden
12	Basic Inspection Size	BILOC	Basis Größe der Anzahl der inspizierten Lines of code, auch hier kann eine Verteilung verwendet werden
13	Size Distribution	LOCD	Verteilung für die Basisgröße
14	Delta Size Distribution	DLOCD	Verteilung für die Delta Lines of Code
15	Inspection Size Distriburion	ILOCD	Verteilung für die Inspizierten Lines of Code

Tabelle 3: General Distributions

2.6.4 Allgemeine Input Daten: (General Input Data)

Hier sind die globalen Werte definiert, welche für alle MUs gelten und zur Berechnung verschiedener Parameter verwendet werden.

Bezeichnung	Einheit	Abk.	Verwendung
Max Design Productivity	Unit/h	MDP	beschreibt die Maximale Produktivität bezüglich der Design Aktivität
Max Coding Productivity	LOC/h	MCP	Maximum Coding Productivity, Obergrenze fürs Lernen.
Max Preparation Productivity	LOC/h	MPP	Maximale Inspektions-Produktivität (Obergrenze für individuelle Produktivität bei der Inspektionsvorbereitung)
Average Meeting Productivity	LOC/h	AMP	Durchnittliche Produktivität für ein Inspektions-Meeting
Defect Find Rate	Def/h	DFR	Minimale Zeit, um einen Fehler zu finden (wird mit testing skill faktor von Person k verrechnet)
Design Quality	Defects/ Unit	DQ	Beschreibt die Qualität des Design bezüglich der Anzahl der Fehler
Min Defect Density	Def/LOC	MDD	Min Defects per LOC Untergrenze für Coding Quality
Average Defects Size	LOC	ADS	Mittlere Größe eines Defekts (damit und mit coding productivity und Coding Quality je person lässt sich Rework berechnen -> faktor)
Max Test Defect Density	Def/KLOC	MTDD	Maximale Defect Density NACH dem Test. so lange Testen bis MTDD erreicht ist. ***(nicht mehr verwendet)***
Coding Quality Learning Factor	1/h	CQLF	Lernfaktor des Skills Coding Quality
Coding Productivity Learning Factor	1/h	CPLF	Lernfaktor des Skills Coding Productiviy
Preparation Productivity Learning Factor	1/h	PPLF	Lernfaktor des Skills Preparation Productivity
Defects Detection Learning Factor	1/h	DDLF	Lernfaktor des Skills Defect Detection
Rework Defects Factor		RDF	Portion (with respect to the reworked defects) of defects produced during rework
domain skill weight		DSW	Parameter zur Gewichtung der domainspezifischen Skills mit den allgemeinen Skills, in [0, 1]
Person Cost	€/h	PC	Kosten einer Personenstunde für Verwendung im Excel Sheet. Weitere Berechnungen kann Excel übernehmen.
Test_Thoroughness		TTh	bestimmt, wie gründlich getestet wird. Wert liegt zwischen 0 und 1 und beschreibt, wieviel Prozent der Fehler gefunden werden.
Defect_ Density_Threshold	Def/KLOC	DDTsh	Schwellenwert, für die Auswahl der zu inspizierenden Artefakte
Size_Threshold	LOC	STsh	Schwellenwert, für die Auswahl der zu

Bezeichnung	Einheit	Abk.	Verwendung
			inspizierenden Artefakte
Complexity_Threshold		CmplxTsh	Schwellenwert, für die Auswahl der zu inspizierenden Artefakte
Percent_Threshold		Perc_Tsh	Schwellenwert zwischen 0 und 1
Number of Inspectors		No_Insp	Kann als Verteilung oder Konstante eingetragen werden.
Selection_Policy		Insp_Pol	0: keine Inspektionen 1: Percent 2: Complexity 3: Size 4: Defect Density
Number_Of_Units		#Units	Wird vom Modell mit dem im Modell eingestellten Wert beschrieben
Number_of_Developers		#Devlo-pers	Wird vom Modell mit dem im Modell eingestellten Wert beschrieben

Tabelle 4: Allgemeine Input-Daten (General Input Data)

2.6.5 Output-Variablen (für Person k)

Output-Daten für die Personen-MUs sind die Veränderungen der persönlichen Eigenschaften aufgrund von Lerneffekten und Zeitdruck. Da die Personen gleiche Aktivitäten mehrfach durchlaufen, entsteht eine Feedbackschleife.

Die Variablen stellen die aktualisierte Skill-Werte dar. Sie setzen sich zusammen aus den Startwerten und den Veränderungen durch Lerneffekte.

Bezeichnung	Abk.	Verwendung
Actual Coding Productivity Skill	$ACPS_k$	Aktuelle Coding Produktivität von Person k
Actual Coding Quality Skill	$ACQS_k$	Aktuelle Coding Qualität von Person k
Actual Preparation Productivity Skill	$APPS_k$	Aktuelle Inspektions Produktivität von Person k für die Inspection Preparation
Actual Defect Detection Skill	$ADDS_k$	Aktuelle Inspektions Qualität von Person k
Domain-specific Skill	$ADSS_{kd}$	Skill in domain d
Time Pressure	ATP_k	Zeitdruck für Person k (evtl. durch Team TP ersetzt)

Tabelle 5: Output-Werte für die Personen

2.6.6 Output-Variablen für die einzelnen Phasen

Für jede der einzelnen Phasen des Modells ist eine Tabelle vorgesehen. Im einfachsten Fall enthält sie die Identifier der MUs (Code MU und Developer MU) sowie die Start- und Endzeit und die Dauer der Phase.

2.6.6.1 Design

Bezeichung	Einheit	Verwendung
Unit Number		Identifier des Moduls
Complexity		Komplexität des Moduls
Developer		Entwickler, der das Design erstellt
Design Time	h	Zeitdauer in Stunden
Start Time Design	h	Startzeitpunkt in Stunden seit Projektbeginn
End Time Design	h	Endzeitpunkt in Stunden seit Projektbeginn

Tabelle 6 : Output-Werte für die Phase Design

2.6.6.2 Coding

In der Coding-Phase werden noch zusätzlich die errechnete Anzahl der Defects abgelegt.

Bezeichung	Einheit	Verwendung
Unit Number		Identifier des Moduls
Developer		Entwickler, der den Code erstellt
Coding Time	h	Zeitdauer in Stunden
Start Time Coding	h	Startzeitpunkt in Stunden seit Projektbeginn
End Time Coding	h	Endzeitpunkt in Stunden seit Projektbeginn
Defects	Def	Anzahl der Fehler im Modul

Tabelle 7: Output-Werte für die Phase Coding

2.6.6.3 Inspection Preparation

Hier bekommt jeder Eintrag eine fortlaufende Nummer, da die ID der Code MU und der Developer MU hier mehrfach vorkommt. Jede CMU wird von mehreren DMU inspiziert, wobei jede DMU an mehren Inspektionen teilnimmt.
Zusätzlich werden noch die von den einzelnen Entwicklern (DMU) gefundenen Fehler abgelegt.

Bezeichnung	Einheit	Verwendung
ID		Fortlaufende Nummer der Vorbereitungsaktivität
Unit Number		Identifier des Moduls (kommt mehrfach vor)
Developer		Entwickler der das Modul Inspiziert
Preparation Time	h	Zeitdauer in Stunden
Start Time Preparation	h	Startzeitpunkt in Stunden seit Projektbeginn
End Time Preparation	h	Endzeitpunkt in Stunden seit Projektbeginn
Defects	Def	Anzahl der Fehler, welche der Entwickler im Modul gefunden hat.

Tabelle 8: Output-Werte der Preparation der Inspektion

2.6.6.4 Inspection Meeting

Es wird neben den Zeiten des Meetings noch die Nummer des CMU und die Anzahl der Inspektoren abgelegt, sowie zusätzlich die gefunden Fehler und die nicht gefunden Fehler. In einer zusätzlichen Tabelle wird die Zuordnung der IDs des CMU und des DMU abgelegt. Dies ist notwendig, um eine Zuordnung der am Inspektionsmeeting teilnehmenden Inspektoren zu bekommen.

Die ID des CMU, bei dessen Kodierung der Entwickler unterbrochen wird, wird auch gespeichert.

Bezeichung	**Einheit**	**Verwendung**
Unit Number		Identifier des Moduls
No_Of_Inspectors		Anzahl der Entwickler im Meeting
Meeting Time	h	Zeitdauer in Stunden
Start Time Meeting	h	Startzeitpunkt in Stunden seit Projektbeginn
End Time Meeting	h	Endzeitpunkt in Stunden seit Projektbeginn
Found Defects	Def	Anzahl gefundener Fehler nach Meeting
Undetected Defects	Def	Anzahl nichtgefundener Fehler.

Tabelle 9: Output-Werte des Inspection Meetings

Inspection Meeting Zuordnung

Bezeichung	**Einheit**	**Verwendung**
ID		fortlaufende Nummer der Zuordnung im Meeting
Inspected Unit		Identifier des Moduls, welches im Inspektions-Meeting bearbeitet wurde (kommt mehrfach vor)
Developer		Entwickler, der an dem Meeting teilnimmt.
Unit Number (developing)		Id des Moduls, an dem der Entwickler arbeitet und dessen Bearbeitung unterbrochen ist

Tabelle 10: Output Werte der Zuordnung von Entwicklern zu dem zu inspizierenden Code Modul

Um die Zeiten (Start – Dauer – Ende) zu ermitteln, muss in der Tabelle Inspection Meeting die zu Inspected Unit passende Unit Number identifiziert werden.

2.6.6.5 Inspection Rework

Neben den Standard Werten werden hier noch die verbleibenden Fehler und die Anzahl der neuen, durch die Korrektur neu eingeführten Fehler abgelegt. In den verbleibenden Fehlern sind die neuen Fehler bereits einbezogen.

Bezeichung	**Einheit**	**Verwendung**
Unit Number		Identifier des Moduls
Developer		Autor des Moduls, der die Defects nacharbeitet.
Rework_Time	h	Zeitdauer in Stunden
Start Time Rework	h	Startzeitpunkt in Stunden seit Projektbeginn

End Time Rework	h	Endzeitpunkt in Stunden seit Projektbeginn
Defects_Remaining	Def	Verbleibende Fehler nach Rework (incl. neuer Defects)
New_Defects	Def	Neue Fehler, die während Rework entstanden

Tabelle 11: Output Werte des Rework in der Inspektion

2.6.6.6 Test (Modultest)

Beim Modultest werden zusätzlich noch die Detected Defects und die verbleibenden Defects berechnet und gespeichert.

Bezeichung	**Einheit**	**Verwendung**
Unit Number		Identifier des Moduls
Developer		Tester des Moduls.
Test_Time	h	Zeitdauer in Stunden
Start Time Test	h	Startzeitpunkt in Stunden seit Projektbeginn
End Time Test	h	Endzeitpunkt in Stunden seit Projektbeginn
Detected Defects	Def	Gefundene Fehler nach Test
Defects	Def	Fehler, die während dem Testen nicht gefunden wurden

Tabelle 12: Output-Werte der Phase Modul Test

2.6.6.7 Test (Rework)

Neben den Zeiten und den IDs der MUs werden noch die Fehler in der Tabelle abgelegt, obwohl sie sich gegenüber dem zuvor ausgeführten Test nicht mehr geändert haben.

Bezeichung	**Einheit**	**Verwendung**
Unit Number		Identifier des Moduls
Developer		Entwickler der die gefundenen Fehler Nacharbeitet.
Rework_Time	h	Zeitdauer in Stunden
Start Time Rework	h	Startzeitpunkt in Stunden seit Projektbeginn
End Time Rework	h	Endzeitpunkt in Stunden seit Projektbeginn
Defects	Def	Fehler, die während dem Testen nicht gefunden wurden und im Modul verbleiben

Tabelle 13 Output-Werte des Reworks nach dem Modul Test.

3 Modularisierung des Modells.

In diesem Kapitel werden Teile des Modells als Module (Reusable Assets) dokumentiert. Zu jedem Modul (Asset) sollen auch die interne Struktur, die Gleichungen sowie die benötigten Daten beschrieben werden. Dazu werden zuerst die verschiedenen Aufgaben eines Moduls definiert und die notwendigen Informationen beschrieben Danach wird ein Beschreibungsschema vorgestellt, mit dessen Hilfe die Module beschrieben werden.

3.1 Aufgaben in einem Model:

In einem SimulationsModell sind bestimmte Module zur Darstellung eines Ausschnitts aus der Wirklichkeit (hier: die Abbildung von Software-Entwicklungsprozessen)enthalten. Einige Module haben dabei eine eher technische Funktion.

A. **Aktivitäten**
   Aktivitäten beschreiben die Tätigkeiten die ausgeführt werden, welche immer Aufwand erfordern, also Zeit und/oder Ressourcen verbrauchen. Dazu zählen Aktivitäten wie z.B. Kodieren.
B. **Ressourcen-Management**
   Zu den Ressourcen zählt alles was in irgendeiner Form beschränkt ist, Personen, Räume, Rechenzeit usw. Welche Ressourcen im Modell dargestellt werden, ist von der Fragestellung im Modell abhängig.
   Veränderungen in den Ressourcen, z.B Lerneffekte oder Skills, werden auch hier beschrieben.
C. **Entscheidungspunkte**
   Entscheidungspunkte können zum einen explizite Management-Entscheidungen modellieren oder ein Vorgehen z.B. die Auswahl der zu inspizierenden Module.
   Da diese immer speziell auf ein Problem bezogen sind, ist hier evtl. eine Meta-Beschreibung oder ein Pattern sinnvoll.
D. **Datenzugriff und Speicherung**
   Inwieweit sich Struktur und Aufbau der Daten gleichen, ist von der Umgebung und der Aufgabe des Modells abhängig. Evtl. kann passend zu anderen Modulen ein Datenzugriffs- oder Speicherungs-Modul angeboten werden
E. **Modell Initialisieren**:
   Um ein Modell mit Daten zu initialisieren, stehen im Prinzip zwei Möglichkeiten zur Verfügung, Generierung oder Initialisierung mit vordefinierten Daten.

Liste aller möglichen Module und Ihrer Aufgaben

	Bezeichnung	**Aufgabe**
E	Generate Items	Die Anzahl der Items wird generiert und mit ersten Daten Initialisiert
A	Design	Aktivität für das Design der Codemodule
A	Kodierung	Kodieren der Module mit der Möglichkeit der Unterbrechung für andere Tätigkeiten
A	Inspektion Preparation	Vorbereitung für Inspektions-Meeting
A	Inspektion Meeting	Meeting
A	Inspektion Rework	Rework der Defekte
A	Test	Modultest der kodierten Module
A	Rework	Rework der im Test gefundenen Fehler
B	Set Skill	Veränderung der verschiedenen Skills
B	Resource Management	Initialisierung der Ressource Developer
C	Inspect?	Entscheidung, welche Module inspiziert werden

Tabelle 14: Klassifizierung der unterschiedlichen Aufgaben eines Moduls

3.2 Notwendige Informationen zu den Modulen

Um ein Modul eines diskreten Simulationsmodells zu beschreiben, müssen Informationen zu den sog. Items oder Moving Units, den Daten, der Struktur oder Architektur sowie der Ablauf oder Durchflusslogik vorhanden sein. Im folgenden werden die Informationen genauer aufgeführt, welche zu den Modulen mitgeliefert werden müssen, um diese korrekt einzusetzen, anzupassen oder zu implementieren.

3.2.1 Moving Units (Items)

Unter Items versteht man in der diskreten Simulation Elemente, welche durch die einzelnen Blöcke wandern. Jedes dieser Items kann beliebig viele Attribute besitzen. Für die Modul-Bildung ist es wichtig zu definieren, welche Informationen die Items besitzen müssen, um in dem Modul korrekt bearbeitet werden zu können.

3.2.1.1 Kombination von Items

Bei bestimmten Modulen des Modells werden zwei oder mehrere Items zu einem Item kombiniert. Es gibt verschiedene Optionen, wie die Attribute dabei behandelt werden.

- Sie werden zunächst zusammengeführt und später aufgetrennt. Danach besitzen beide Items alle Attribute. Neue Attribute bleiben erhalten.

- Wie oben, nur werden die Attribute nach dem Auftrennen wieder dem jeweiligen Item zugeordnet. Neue Attribute gehen verloren.
- Bei Attributen mit gleichem Namen kann gewählt werden, welches Item mit seinen Attributen, die der anderen überlagert.

3.2.2 Daten

Es kann zwischen drei verschiedenen Arten von Daten unterschieden werden.

- Daten, welche zur Initialisierung und Kalibrierung des Modells dienen.
- Daten, welche einem Objekt / Item zugeordnet sind. Hier sind als Objekte Teile/Module der Software oder Personen zu verstehen. Die Daten sind sozusagen die Attribute oder Variablen der Objekte.
- Daten, die zu einer Aktivität oder Entscheidung gehören und dort entstehen. Diese Daten werden meist mit zusätzlichen Informationen abgelegt. Hierzu zählen speziell Zeitpunkte, Zeiträume sowie ID-Nummern

Für jedes Modul müssen die benötigten Daten und deren Wertebereiche beschrieben sein.

3.2.3 Formeln und Verteilungen:

Um z.B. Zeiten oder Qualitätsmaße zu errechnen, werden in den Modulen Formeln und stochastische Verteilungen verwendet.

Aufbau und Parameter der Formeln in den Modulen müssen dokumentiert sein. Ebenso die Domäne oder Umgebung, in der die Formel gültig ist.

Gleiches gilt für die Verteilungen, sofern es Gründe für Parameter oder die Art der Verteilung gibt.

3.3 Template zur Beschreibung eines Moduls:

Dieses Template dient dazu, die Module strukturiert zu beschreiben. Das Template ist nur bedingt geeignet, um als Beschreibung einer wiederverwendbaren Komponente oder eines Patterns zu dienen.

3.3.1 Name:

Bezeichnung des Moduls, es sollte eine möglichst verständliche Bezeichnung gewählt werden.

3.3.2 Typ / Aufgabe

Hier wird der Typ des Moduls nach der Klassifizierung unter 3.1 angegeben sowie die Aufgabe des Moduls kurz beschrieben.

3.3.3 Domäne

Die Domäne beschreibt den Kontext, in dem das Modul entwickelt wurde und eingesetzt werden kann.

3.3.4 Moving Units (Items)

Hier werden die Moving Units (MUs) beschrieben. Oft werden sie auch als Items bezeichnet, sie stellen die Informationseinheiten oder Objekte dar, welche durch das Modul wandern. Ihnen zugeordnet sind Informationen in Form von Attributen. Die Attribute, welche das Modul benötigt, und deren Werteberei-che müssen hier beschrieben werden.

3.3.5 Funktionsweise des Moduls

Beschreibung, wie die MUs durch das Modul geführt werden, wo sie sich eine definierte Zeit aufhalten, kombiniert oder getrennt werden. Dies lässt sich oft durch eine Grafik oder Blockschaltbild darstellen. Auch ist darzustellen, welche der MUs bei Kombination dominieren (Daten) und ob die Eigenständigkeit der MUs gewahrt bleiben soll. Dies impliziert auch, dass die Einstellungen einiger Optionen in den Blöcken des verwendeten Simulationstools, in unserem Fall Ex-tend, erwähnt werden, soweit sie sich auf die Funktionsweise auswirken.

3.3.5.1 Eingänge:

Hier werden alle Eingänge aufgezählt, sowohl diskrete als auch kontinuierliche.

3.3.5.2 Ausgänge:

Hier werden alle Ausgänge aufgezählt, sowohl diskrete als auch kontinuierliche.

3.3.6 Benötigte und produzierte Daten

Es müssen alle Daten angegeben und dahingehend klassifiziert werden, ob sie zur Initialisierung/Kalibrierung dienen, einer MU (als Attribut) zugeordnet sind oder zum Modul gehören, also dort entstehen.

Auflistung wie die Daten abgelegt sein müssen, um vom Modul genutzt zu werden und wie die entstehenden Daten gespeichert werden.

3.3.7 Formeln und Verteilungen

Ein Modul sollte nicht mehr als 2-3 Formeln beinhalten. Diese Formeln sind aufzuführen und zu erklären. Dazu gehört eine Darstellung der zugehörigen Parameter-Werte sowie eine graphische Darstellung, sofern notwendig, z.B. bezüglich der Auswirkungen von Änderungen der Parameter-Werte

Werden Verteilungen verwendet, sind diese zu benennen. Auch die die Gründe für ihre Wahl und verwendete Parameter-Werte sind anzugeben.

3.3.8 Varianten

In diesem Punkt können Varianten beschrieben werden, also wie durch kleine Änderungen an der Struktur das Modul auch unter etwas anderen Rahmenbedingungen eingesetzt werden kann. Dieser Punkt ist nicht zwingend notwendig in der Beschreibung eines Moduls.

4 Module des Modells

Die hier aufgeführten Beispiele beschreiben die Module des Codeinspektion-Modells.

4.1 Generate Items

4.1.1 Name:

Generate Items

4.1.2 Typ / Aufgabe

Typ:

Initialisierung des Model, Speziell der Code-Items

Aufgabe:

Generate Code Units generiert MUs und initialisiert diese mit Informationen für den weiteren Durchlauf durch das Modell. Es können wahlweise Daten über Verteilungen generiert, als auch aus einer Datenbank gelesen werden. Die Umschaltung wird durch einen „Schalter" **im** Modul realisiert.

4.1.3 Domäne

Dieses Modul wurde nicht für eine spezielle Domäne entwickelt. Ein Einsatz in verschiedenen Domänen ist denkbar.

4.1.4 Moving Units (Items)

Es wird eine vorher definierte Anzahl von Moving Units (MUs) generiert. Bei Bedarf kann eine Verteilung über die Zeit vorgegeben werden.

Die MUs werden mit folgenden Daten initialisiert, welche als Attribute an jede MU hinzugefügt werden.

- Nummer : laufende Nummer zur Identifikation (Integer).
- Größe(Integer z.B. als Lines of Code oder Function Points)
- Gesamtgröße

- Geänderte Größe
- zu überprüftende Größe
- Komplexität (Real, als Log Normal Verteilung)

Die Angabe der Größe lässt sich in drei Kategorien aufteilen, um die Weiterentwicklung von bestehenden Elementen zu ermöglichen sowie deren Überprüfung z.B. mittels Inspektion.

4.1.5 Funktionsweise des Moduls

Die MUs werden vom Generator erzeugt und beim Durchlauf bis zu einem Speicher (FIFO Stack) mit den Daten initialisiert.

4.1.5.1 Eingänge:

Keine Eingänge vorhanden

4.1.5.2 Ausgänge:

- MU_OUT: Ausgang der initialisierten MUs
- SumDSize: kumulierte Größe der zu bewältigenden Arbeit (Summe Delta Size).

4.1.6 Benötigte und produzierte Daten

Es werden zur Initialisierung folgende Daten benötigt. Diese Daten sollen aus einer Datenbank entnommen werden oder über Verteilungen generiert werden können.

Aus diesem Grund wird die Tabelle, welche die Daten enthält, entweder ausgelesen (Initialisierung) oder neu beschrieben (Generierung).

Die Daten werden in der mit der Extend Suite mitgelieferten Datenbank abgelegt oder von dort gelesen. Die Datenbank ist tabellarisch aufgebaut Es besteht eine Möglichkeit die Daten von und nach Excel zu exportieren.

Daten aus DB lesen

Anzahl der Items (im Modell eingeben, wird in DB geschrieben
[techn. Problem])
Gesamtgröße jeder einzelnen MU
Geänderte Größe jeder einzelnen MU
zu überprüfende Größe jeder einzelnen MU
Verteilungsfunktionen, wenn die Daten generiert werden.
Verteilungsfunktion für die Komplexität.

TABELLE : MU Input Data

Diese Tabelle enthält die Werte für die generierte Anzahl von MUs.

MU ID	SIZE	Delta SIZE	Inspected SIZE	Complexity

TABELLE: General Distributions

Diese Tabelle enthält neben anderen Verteilungen auch die Verteilung für die Komplexität und die Generierung der Daten der MUs, falls dies gewünscht ist. Die Komplexität wird immer, auch beim Einlesen der Daten aus der Tabelle, durch die Verteilung bestimmt.

Bezeichnung	**Abk.**	**Verteilungsdaten**
Random Unit Complexity Factor	RUCF	Log Normal Verteilung
Basic Size	BLOC	Log Normal Verteilung
Basic Delta Size	BDLOC	Log Normal Verteilung
Basic Inspection Size	BILOC	Dreiecks Verteilung
Size Distribution	LOCD	Normal Verteilung
Delta Size Distribution	DLOCD	Normal Verteilung
Inspection Size Distribution	ILOCD	Log Normal Verteilung

Daten, welche in die Datenbank geschrieben werden.

ID Nummer der MU
Gesamtgröße jeder einzelnen MU (sofern generiert)
Geänderte Größe jeder einzelnen MU (sofern generiert)
zu überprüfende Größe jeder einzelnen MU (sofern generiert)
Komplexität der MU

TABELLE : MU Input Data
Diese Tabelle enthält die Werte für die generierte Anzahl von MUs.

MU ID	SIZE	Delta SIZE	Inspected SIZE	Complexity

Initialisierte Attribute:

Jede MU **hat** beim Verlassen des Blocks folgende Attribute

Attribut	**Im Modell**
ID Nummer der MU	Unit_Number
Größe	Size
Geänderte Größe	Delta_Size
zu überprüfende Größe	Inspected Size
Komplexität	Complexity

4.1.7 Formeln und Verteilungen

Es werden nur Verteilungen zur Initialisierung und Generierung der MUs verwendet.

Verteilungen:

	Typ	Param. 1	Param. 2	Param. 3
Generierung der MUs:				
Generate Basic Size	Log Normal	Mean: 1	Std. Dev.: 0,5	Location : 0
Generate Basic Delta Size	Log Normal	Mean: 0,8	Std. Dev.: 0,5	Location: 1
Generate Basic Inspection Size	Triangular	Min: 0,9	Max: 1,5	Most Likley: 1
Generate Size Distribution	Normal	Mean: 300	Std. Dev.: 50	---------
Generate Delta Size Distribution	Normal	Mean: 50	Std. Dev.: 10	----------
Generate Inspection Size Distribution	Log Normal	Mean: 1	Std. Dev.: 2	Location: 1
Komplexität der MUs:				
Random Unit Complexity Factor	Log Normal	Mean: 1	Std. Dev.: 1	Location: 0,5

4.2 Resource Management

4.2.1 Name

Resource Management

4.2.2 Typ / Aufgabe

Typ:

Ressourcen-Verwaltung und Initialisierung des Modells

Aufgabe:

Initialisierung und Verwaltung der Ressourcen, in diesem Fall der Ressource Developer. Diese sind als Moving Units realisiert, welche den Aktivitäten zur Verfügung gestellt werden.

4.2.3 Domäne

Dieses Modul wurde nicht für eine bestimmte Domäne entwickelt.

4.2.4 Moving Units (Items)

Es wird eine vorher definierte Anzahl von Moving Units (MUs) generiert. Bei Bedarf kann eine Verteilung über die Zeit vorgegeben werden.

Die MUs werden mit folgenden Daten initialisiert, welche als Attribute an jede MU hinzugefügt wird.

Attribut	**Im Model**
ID Nummer der Person	Person_ID
Code Productivity Skill	CPS
Preparation Productivity Skill	PPS
Code Quality Skill	CQS
Preparation Quality Skill	PQS
Test Quality Skill	TQS
Domain Specific Skill	(nicht genutzt)
Time Pressue (im Moment nicht genutzt)	Time_Pressure

Um zu verhindern, dass Attribute, welche während der Bearbeitung der MU hinzugefügt wurden, beim nächsten Einsatz der Ressource sich störend bemerkbar machen, werden alle außer den oben genannten Attributen gelöscht.

4.2.5 Funktionsweise des Moduls

Bei der Erzeugung der MUs werden diese mit den oben genannten Attributen initialisiert und in den Resource Pool überführt. Dort werden sie dann von den verschiedenen Blöcken angefordert. Die Reihenfolge, in der diese Anforderungen bearbeitet werden, legt die Priorisierung der Ausgänge fest. Erst wenn keine Anforderung an dem Ausgang mit der Priorität eins anliegt werden die Anforderungen an den Ausgang mit der Priorität zwei erfüllt, usw.

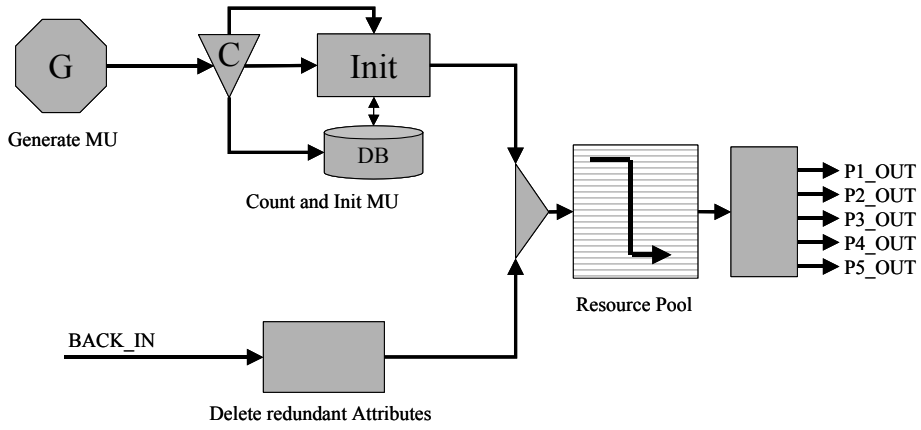


Fig 2: Erzeugen von Moving Units für einen Resource Pool

Module, die in den Resource Pool zurückkehren, werden von allen Attributen befreit, welche nicht direkt zu der MU der Person gehören. Damit werden Probleme vermieden, die sich ergeben, wenn Attribute von vorherigen Durchläufen und anderen MUs noch mit der MU der Person verbunden sind. Diese Verbindung entsteht durch das Batch und Unbatch von verschiedenartigen MUs

4.2.5.1 Eingänge

- BACK_IN: Eingang für die Resource Items welche wieder frei geworden sind.

4.2.5.2 Ausgänge

- FIRST_OUT: Ausgang mit der Priorität eins
- SECOND_OUT: Ausgang mit der Priorität zwei
- THIRD_OUT: Ausgang mit der Priorität drei
- FOURTH_OUT: Ausgang mit der Priorität vier
- FIFTH_OUT: Ausgang mit der Priorität fünf

4.2.6 Benötigte und produzierte Daten

Es werden die Ausgangswerte für die Initialisierung der MUs benötigt. Diese werden aus einer Tabelle in der Datenbank gelesen. Gleichzeitig werden diese Anfangswerte in eine Tabelle geschrieben, welche die aktuellen Werte während der Simulation enthält.

Die Initialisierungstabelle mit den Baselines für die Skills hat folgende Struktur

Person ID	CPS Baseline	CQS Baseline	PPS Baseline	PQS Baseline	TQS Baseline	Domain Specific Skill Baseline	Time Pressure

Die Tabelle mit den aktuellen Skill-Werten der Entwickler ist identisch aufgebaut. Grund für diese Konstruktion ist, dass die Unterschiede zwischen den Ausgangswerden der Skills und den Endwerten der Skills nach der Simulation erkennbar sind. Die Werte in dieser Tabelle werden von dem Block Set Skill (Kapitel 4.5) während des Durchlaufs ständig aktualisiert.

Person ID	Actual CPS	Actual CQS	Actual PPS	Actual PQS	Actual TQS	Actual Domain Specific Skill	Time Pressure

4.2.7 Formeln und Verteilungen

Es werden weder Formeln noch Verteilungen verwendet.

4.3 Design

4.3.1 Name:

Design
Das zugrundeliegende Template ist Simple Activity.

4.3.2 Typ / Aufgabe

Typ:

Aktivität

Aufgabe:

Verbindet ein MU mit einer zweiten MU (Ressource) z.B. einer Person um eine Tätigkeit auszuführen. Nach der Ausführung werden beide MUs wieder getrennt. Es werden verschiedene Daten bezüglich der Zeit (Dauer, Anfangs- und Endzeitpunkt) gespeichert.

4.3.3 Domäne

Dieses Modul wurde für die Simulation einer Aktivität in der Software-Entwicklung entwickelt, dort allerdings nicht für eine spezielle Domäne. Ein Einsatz in verschiedenen Domänen aus dem Software-Entwicklungsbereich ist denkbar.

4.3.4 Moving Units (items)

Es sind zwei Arten von MUs notwendig.

Eine MU stellt das Objekt dar, welches in der Aktivität verändert, bearbeitet oder erstellt wird, das zweite stellt die benötigte Ressource dar, z.B. eine Person oder ein Werkzeug. Es werden beide MUs benötigt, damit das Modul funktioniert

4.3.5 Funktionsweise des Moduls

Das Modul hat zwei Eingänge für MUs und ebenso zwei Ausgänge für MUs. Ein Eingang sowie Ausgang ist für die MUs bestimmt, die das Objekt, welches be-

arbeitet, verändert oder erstellt wird, bestimmt, z.B. Design erstellen aus Anforderungen.

Der zweite Ein- sowie Ausgang ist für die Ressource bestimmt, welche benötigt wird, um die Aktivität auszuführen. Da die Ressource als MU mit Attributen ausgeführt ist, bietet diese Art der Benutzung einer Ressource die Möglichkeit Eigenschaften der Ressource zu modellieren und in die Berechnungen mit einzubeziehen.

Beide MUs werden am Anfang zu einem Paar zusammengefügt und am Ende wieder getrennt. Dadurch ist es möglich, auf die Attribute beider MUs zuzugreifen.

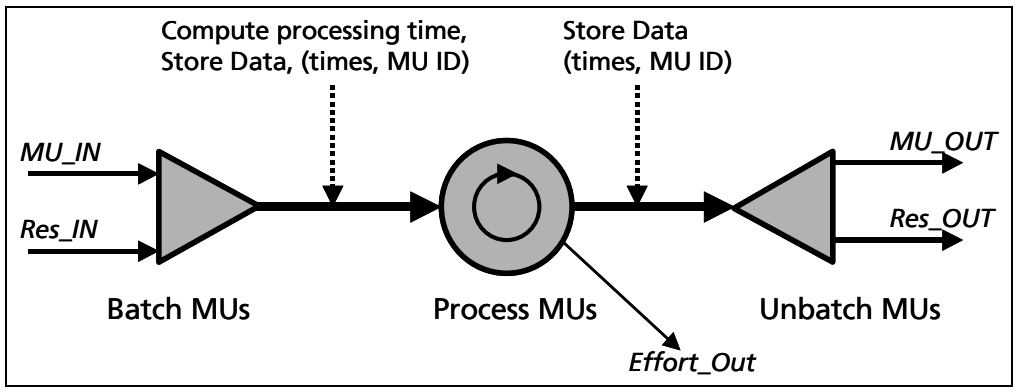


Fig 3: Schematische Darstellung des flusses der MUs durch Simple Activity

Die MU für die Ressource wird erst in das Modul „gezogen", wenn eine MU am Eingang für das zu bearbeitende Objekt vorhanden ist. Dadurch wird die Ressource erst aus dem Resource Pool entfernt, wenn sie auch wirklich benötigt wird und steht bis dahin für andere Aufgaben zur Verfügung.

Bei der Auftrennung der MUs am Ende des Moduls behält jedes Modul seine Attribute. Neue Attribute, die während der Vereinigung hinzugefügt wurden, sind danach in keiner der MUs enthalten.

4.3.5.1 Eingänge:

- MU_IN: MU input, typischerweise das zu bearbeitende Objekt
- Res_IN: MU Ressource input, typischerweise die Ressource, welche zur Bearbeitung benötigt wird

4.3.5.2 Ausgänge:

- MU_OUT: MU output, typischerweise das bearbeitete Objekt

- Res_OUT: MU Ressource output, typischerweise die Ressource, welche zur Bearbeitung benötigt wurde
- Effort_Out: der kumulierte Aufwand.

4.3.6 Benötigte und produzierte Daten

In diesem Modul werden als Input-Daten die Daten, welche in den Attributen der MUs gespeichert sind, sowie die aus der Datenbank benötigten Daten zur Berechnung der Zeit für die Aktivität.
Output-Daten sind die Informationen, welche in die Datenbank geschrieben werden. Dazu zählen alle Informationen über die Zeit (Dauer, Start- und Endzeit) sowie die IDs der MUs.

MU Attribute:

Zu bearbeitendes Objekt:

Attribut	Name im Modell	Kommentar
ID der MU	Unit_Number	Zur Identifikation der MU
Größe der MU	Size	Hier kann wahlweise die Gesamtgröße oder geänderte Größe angegeben sein.
Komplexität	Complexity	

Ressource MU:

Attribut	Name im Modell	Kommentar
ID der MU	Personnumber	Um die Ressource zuordnen zu können.

Input-Daten aus Datenbank:

Zur Berechnung der Dauer der Aktivität wird eine Verteilung aus der Datenbank gelesen. Bei Bedarf können noch weitere Informationen in den Equation Block eingegeben werden.

TABELLE: General Distributions

Diese Tabelle enthält neben anderen Verteilungen auch die Verteilung für die Berechnung der Zeit in Simple Activity. Da Simple Activity für die Aktivität Design eingesetzt wird, ist die Bezeichnung auch Random Design Time Factor. Für andere Aktivitäten muss dies geändert werden.

Bezeichnung	Abk.	Verteilungsdaten
Random Design Time Factor	RDTF	Log Normal Verteilung

Output-Daten in Datenbank

In der Datenbank wird eine Tabelle mit allen Informationen beschrieben, welche in diesem Modul anfallen.

Tabelle DESIGN

Unit Number	Complexity	Developer	Design Time	Start Time	End Time

Für jedes MU, die durch dieses Modul geht, wird eine Zeile angelegt. Die Zeile wird über die Information in „Unit Number" festgelegt. Dadurch ist es nicht möglich, das gleiche MU (mit der gleichen Unit Number) mehrfach durch das Modul zu schicken. Hier müsste dann eine Append-Funktion genutzt werden, die das SDI Interface von Extend nicht bietet.

4.3.7 Formeln und Verteilungen

Es wird nur eine Verteilung zur Berechnung der Zeit in der Aktivität. Dies ist auch die einzige Formel / Berechnung in diesem Modul.

Vereilung:

	Typ	Param. 1	Param. 2	Param. 3
Random (Design) Time Factor	Log Normal	Mean: 1	Std. Dev.: 0,5	Location : 1

Formel:

$$\text{Design Time} = (\text{Size} \cdot \text{Random\_Design\_Time\_Factor} \cdot \text{Complexity})$$

Size und Complexity werden aus den Attributen des MUs gelesen. Der Random Design Time Factor ist ein Wert aus der Datenbank und wird durch eine Verteilung realisiert. Das Ergebnis wird in dem Attribut Design_Time abgelegt und in der Datenbank gespeichert. Das Attribut Design_Time wird bei dem Auftrennen der MUs gelöscht aufgrund der Einstellung Preserve Uniqueness in den Batch Blöcken.

Erweiterung der Formel:

Die Formel kann noch erweitert werden, um die Fähigkeiten der Ressource (Entwickler) indem noch ein Basic Productivity Faktor hinzukommt (aus der Datenbank gelesen) und die Ressource einen Skill-Wert hat, der mit dem Faktor Basic Productivity Skill der Ressource (Entwickler) multipliziert wird. Damit würde die Formel folgendermaßen lauten

$$\text{Design\_Time} = \frac{\text{Size} \cdot \text{Random\_Design\_Time\_Factor} \cdot \text{Complexity}}{\text{max\_Design\_Productivity} \cdot \text{Design\_Produktivity\_Skill}}$$

4.4 Kodierung

4.4.1 Name:

Kodierung
Das zugrunde liegende Template ist Pre-emptive Activity.

4.4.2 Typ / Aufgabe

Typ:

Aktivität

Aufgabe:

Zwei bereits verbundene MUs (zu bearbeitendes Objekt und Ressource). Nach der Ausführung bleiben beide MUs zusammen. Grund ist die Möglichkeit einzelne Ressourcen während der Aktivität zu unterbrechen und später dort fortfahren zu lassen. Es werden verschiedene Daten bezüglich der Zeit (Dauer, Anfangs- und Endzeitpunkt) gespeichert.

4.4.3 Domäne

Dieses Modul wurde für die Simulation einer Aktivität in der Software-Entwicklung entwickelt, dort allerdings nicht für eine spezielle Domäne. Ein Einsatz in verschiedenen Domänen aus dem Software-Entwicklungsbereich ist denkbar.

4.4.4 Moving Units (items)

Es sind zwei Arten von MUs notwendig, die bereits vorher durch einen sog. Batch-Block zu einer MU zusammengefasst wurden.

Ein MU stellt das Objekt dar, welches in der Aktivität verändert, bearbeitet oder erstellt wird. Das zweite stellt die benötigte Ressource dar, z.B. eine Person oder ein Werkzeug. Prinzipiell kann auch eine MU verwendet werden, die alle Informationen besitzt, oder wenn Informationen über die Ressource nicht notwendig sind.

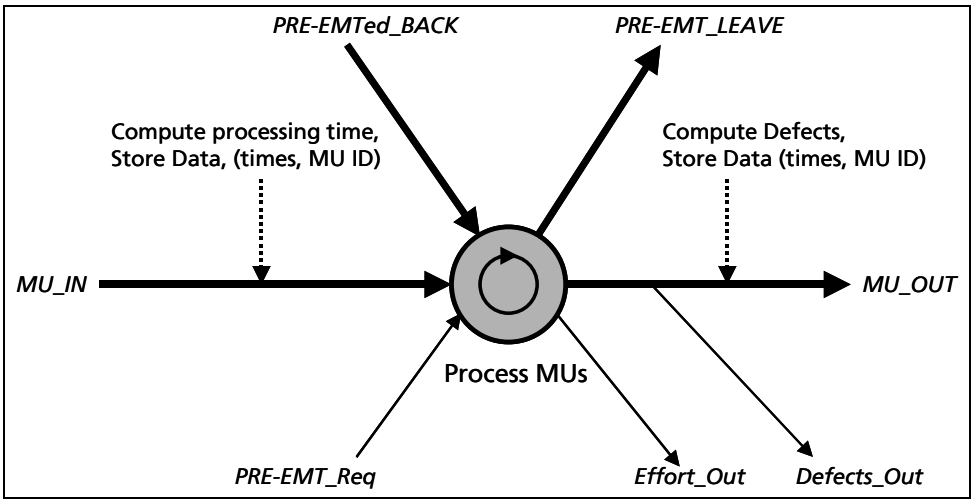


Fig 4: Struktur einer Pre-Emptiven Aktivität

4.4.5 Funktionsweise des Moduls

Die MUs kommen über MU_IN in das Modul und die Bearbeitungszeit, in diesem Fall die Dauer des Kodierens, wird berechnet. Nach Ablauf der Zeit verlassen die MUs das Modul wieder durch MU_OUT. Dabei wird der Gesamtaufwand errechnet und über Effort_Out nach Außen gegeben. Die Anzahl der Fehler im Code wird vor dem Verlassen des Moduls berechnet und sowohl als Attribut in der MU gespeichert als auch über Defects_Out nach Außen geliefert. Über den Eingang PRE-EMT_Req kommen die Anforderungen nach einer Ressource z.B. für eine Inspektion, dann verlässt eine MU die Aktivität und speichert intern die noch aufzuwendende Zeit (Attribut Rem_Coding_Time). Nach Beendigung der Unterbrechung kommt die MU über den Eingang PRE-EMT_Back wieder in die Aktivität zurück, um die Aktivität zu beenden. Eine solche Unterbrechung kann während einer Bearbeitung mehrfach erfolgen.

4.4.5.1 Eingänge:

- MU_IN: MU Input, typischerweise das zu bearbeitende Objekt plus Ressource
- PRE-EMT_Back: MU Input, typischerweise die Ressource, welche zur Bearbeitung einer anderen Aktivität benötigt wurde, wird hier zurück geführt ,um die zuvor begonnene Aktivität zu beenden.
- PRE-EMT_Req: Input, um die Anforderung einer Ressource anzuzeigen. Bedingung: der Wert muss größer 0,5 sein.

4.4.5.2 Ausgänge:

- MU_OUT: Ausgang für MUs, typischerweise das bearbeitete Objekt plus die Ressource
- PRE-EMT_Leave: Ausgang für Ressourcen-MUs, typischerweise die Ressource, welche zur Bearbeitung einer anderen Aktivität benötigt wird
- Effort_Out: der kumulierte Aufwand.
- Defects_Out: hier wird für jede MU, die das Modul verlässt, die Anzahl der Defects ausgegeben.

4.4.6 Benötigte und produzierte Daten

In diesem Modul werden als Input-Daten die Daten, welche in den Attributen der MUs gespeichert sind, sowie die aus der Datenbank benötigten Daten zur Berechnung der Zeit für die Aktivität verwendet.
Output-Daten sind die Informationen, welche in die Datenbank geschrieben werden. Dazu zählen alle Informationen über die Zeit (Dauer, Start- und Endzeit) sowie die IDs der MUs.

MU-Attribute:

Zu bearbeitendes Objekt:

Attribut	Name im Modell	Kommentar
ID der MU	Unit_Number	Zur Identifikation der MU
Größe der MU	Size	Hier kann wahlweise die Gesamtgröße oder geänderte Größe angegeben sein.
Komplexität	Complexity	

Ressource MU:

Attribut	Name im Modell	Kommentar
ID der MU	Personnumber	Um die Ressource zuordnen zu können.
CPS	Coding Productivity Skill	
CQS	Coding Quality Skill	

Neue Attribute für Kombinierte MU:

Attribut	Name im Modell	Kommentar
Coding time	Coding_Time	Gesamtzeit für die Kodierung dieses Moduls
Remaining Coding Time	Rem_Code_Time	verbleibende, noch aufzuwendende Zeit zur Fertigstellung
Defects	Defects	Anzahl der eingefügten Fehler
Autor von Modul N	Author_Of	ID des Moduls, welches der Developer gerade bearbeitet

Input Daten aus Datenbank:

Zur Berechnung der Dauer der Aktivität wird eine Verteilung aus der Datenbank gelesen. Bei Bedarf können noch weitere Informationen in die Equation Blöcke eingegeben werden.

TABELLE: General Distributions

Diese Tabelle enthält neben anderen Verteilungen auch die Verteilung für die Berechnung der Zeit in Pre-Emptive Activity und der Anzahl der Defects in der Codeunit.

Bezeichnung	**Abk.**	**Verteilungsdaten**
Random Coding Time Factor	RCTF	Log Normal Verteilung
Random Defect Density	RDD	Log Normal Verteilung

TABELLE: General Input Data

Diese Tabelle enthält mehrere Werte die zur Kalibrierung des Modells notwendig sind.

Bezeichnung	**Abk.**	**Einheit**
Min Defect Density	MDD	Defects per LOC

Output-Daten in Datenbank

In der Datenbank wird eine Tabelle mit allen Informationen beschrieben, welche in diesem Modul anfallen.

Tabelle Coding

Unit Number	Developer	Coding Time	Start Time	End Time	Defects

Für jede MU, die durch dieses Modul geht, wird eine Zeile angelegt. Die Zeile wird über die Information in „Unit Number“ festgelegt. Dadurch ist es nicht möglich, die gleiche MU (mit der gleichen Unit Number) mehrfach durch das Modul zu schicken. Hier müsste dann eine Append Funktion genutzt werden, die das SDI Interface von Extend aber nicht bietet.

4.4.7 Formeln und Verteilungen

Es wird nur eine Verteilung zur Berechnung der Zeit in der Aktivität verwendet. Die zweite Verteilung wird in der Berechnung der Anzahl der Defects verwendet.

Vereilung:

	Typ	Param. 1	Param. 2	Param. 3
Random (Coding) Time Factor	Log Normal	Mean: 1	Std. Dev.: 0,2	Location: 1
Random Defect Density	Log Normal	Mean: 1	Std. Dev.: 0,2	Location: 1

Formel:

Coding time

$$\text{CodingTime} = \frac{\text{Random\_Coding\_Time\_Factor} \cdot \text{Size} \cdot \text{Complexity}}{\text{Max\_coding\_Productivity} \cdot \text{Coding\_Productivity\_Skill}}$$

Size und Complexity werden aus den Attributen des MUs gelesen. Der Random Coding Time Factor ist ein Wert aus der Datenbank und wird durch eine Verteilung realisiert. Das Ergebnis wird in dem Attribut Coding_Time abgelegt und in der Datenbank gespeichert.

Number of introduced Defects:

$$\text{Defects} = \text{Random\_Defect\_Density} \cdot \text{Size}_i \cdot \text{Complexity}_i \cdot \frac{\text{min\_Defect\_Density}}{\text{Coding\_Quality\_Skill}_k}$$

Das Ergebnis wird wegen der Ganzzahligkeit der Variablen gerundet und sowohl in dem Attribut Defects als auch in der Datenbank gespeichert.

4.5 Update Skills

4.5.1 Name

Update Skills
Das zugrunde liegende Template ist Set Skill.

4.5.2 Typ / Aufgabe

Typ:

Ressourcen-Management

Aufgabe:

Ein Skill der involvierten Person (Ressource) wird verändert abhängig von der Zeit, die die Person mit der Aktivität zugebracht hat, für die dieser Skill benötigt wird.

4.5.3 Domäne

Dieses Modul wurde für die Verwendung in der Software-Entwicklung entwickelt, dort allerdings nicht für eine spezielle Domäne. Ein Einsatz in verschiedenen Domänen aus dem Software-Entwicklungsbereich ist denkbar.

4.5.4 Moving Units (items)

Es ist eine MU notwendig, die Informationen zur Identifikation der Person enthalten muss sowie die Dauer der ausgeführten Tätigkeit und den aktuellen Skill-Wert. In der Vorliegenden Applikation sind die MUs des bearbeiteten Objektes sowie die Ressource (Person) in einer MU vereinigt, welche alle individuellen Informationen enthält.

4.5.5 Funktionsweise des Moduls

Die MUs werden linear durch das Modul geführt in Modellzeit t=0, d.h. es wird keine Zeit dafür benötigt, den Skill zu aktualisieren.

4.5.5.1 Eingänge:

- MU_IN: MU Input

4.5.5.2 Ausgänge:

- MU_OUT: MU output,

4.5.6 Benötigte und produzierte Daten

In diesem Modul werden als Input-Daten die Daten, welche in den Attributen der MUs gespeichert sind sowie die aus der Datenbank benötigten Daten zur

Berechnung des neuen Skills verwendet.
Output ist der aktualisierte Skill der sowohl als Attribut in der MU als auch in der Datenbank abgelegt wird.

MU-Attribute:

Zu bearbeitendes Objekt:

Attribut	Name im Modell	Kommentar
ID der bearbeiteten MU	Unit_Number	zur Identifikation der MU
Zeit der Bearbeitung	z.B. Coding Time	Falls in der MU als Attribut vorhanden, ansonsten auch aus DB

Ressource MU:

Attribut	Name im Modell	Kommentar
ID der MU	Personnumber	Um die Ressource zuordnen zu können.
Skill	CPS, CQS	Die verschiedenen Skills

Input-Daten aus Datenbank:

Zur Berechnung des neuen Skill-Wertes wird aus der Datenbank der Learning Factor für diesen Skill gelesen. Bei Bedarf können noch weitere Informationen in die Equation Blöcke eingegeben werden.

TABELLE: General Input Data

Diese Tabelle enthält mehrere Werte die zur Kalibrierung des Modells notwendig sind. Für jeden in dem Modell verwendeten Skill wird ein Lernfaktor in dieser Tabelle abgelegt.

Bezeichnung	Abk.	Einheit
Learning Factor for Skill	CPLF, CQLF	

Output-Daten in Datenbank

Neben der Aktualisierung des Attributs des zugehörigen Skills wird auch der Skill in der Datenbank aktualisiert.

TABELLE: Actual Skills

Person ID	Skill 1	Skill 2	Skill 3	Skill n

Es können beliebig viele Skills und beliebig viele Personen verwaltet werden.

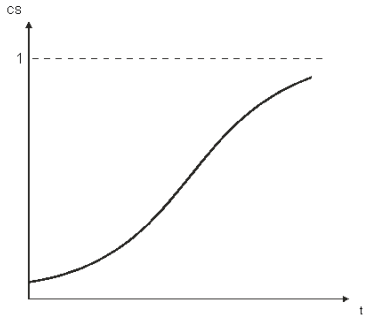
4.5.7 Formeln und Verteilungen

Es keine Verteilung verwendet.

Formel:

$$New\_Skill = \frac{1}{\left(1 + \left(\frac{1}{SKILL - 1}\right) \cdot \exp\left(-LearningFact \cdot DeltaTime\right)\right)}$$

Dabei sieht die Veränderung des Skills über die Zeit folgendermaßen aus:



4.6 Routing Decision

4.6.1 Name

Inspect?
Das zugrunde liegende Template ist Routing Decision

4.6.2 Typ / Aufgabe

Typ:

Entscheidungspunkt

Aufgabe:

Die Items werden aufgrund eines Attributes und eines Schwellenwertes aus der Datenbank durch den ersten oder den zweiten Ausgang des Routing Decision-Moduls geleitet. Welche der Attribute verwendet werden, kann durch eine Auswahl von Außen mittels einer Integer-Variable bestimmt werden. Es kann

jeweils immer nur ein Attribut Wert herangezogen werden, es ist also keine mehrstufige Entscheidung möglich mit nur einem Modul dieser Art.

4.6.3 Domäne

Dieses Modul wurde für die Verwendung in der Software-Entwicklung entwickelt, dort allerdings nicht für eine spezielle Domäne. Ein Einsatz in verschiedenen Domänen aus dem Software-Entwicklungsbereich ist denkbar.

4.6.4 Moving Units (Items)

Es ist eine MU notwendig, die die Informationen enthalten muss, aufgrund derer die Entscheidung getroffen werden kann.

4.6.5 Funktionsweise des Moduls

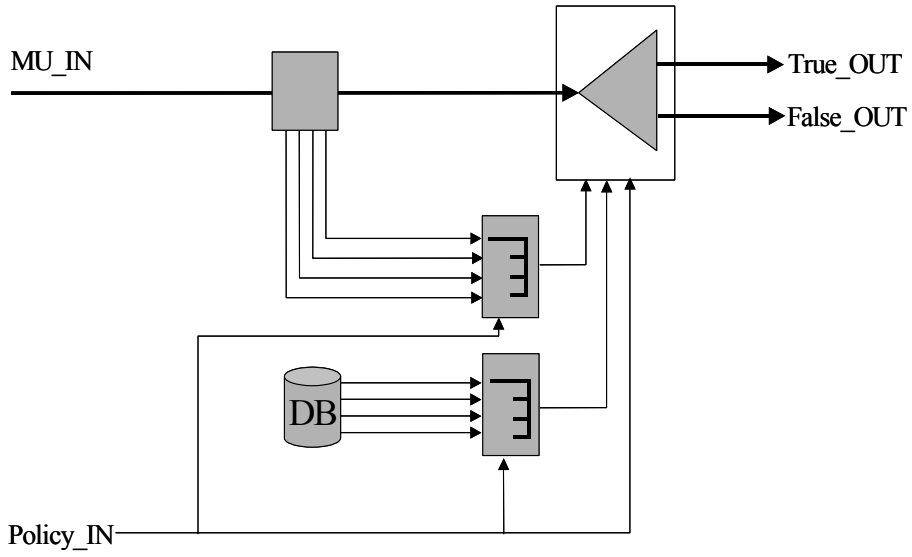


Fig 5: Routing Decission

Die MUs werden linear durch das Modul geführt in Modellzeit t=0, d.h. es wird keine Zeit dafür benötigt, die Entscheidung zu treffen.

Beim Hineingehen eines MU werden alle für die Entscheidung auswählbaren Attribute gelesen. Der Eingang Policy_IN, eine Integer-Zahl, selektiert die zu vergleichenden Werte und gleichzeitig wird der „Vergleicher" über Policy_IN so parametriert, dass der korrekte Vergleich ausgewählt wird.

Sollen Werte zur Entscheidung herangezogen werden, die nicht direkt als Attribut vorliegen, müssen die MUs vor dem Eintritt in das Modul mit diesen initialisiert (berechnen und als Attribut ablegen) werden.

4.6.5.1 Eingänge:

- MU_IN: MU input.
- Policy_IN: Selektiert die Werte, die verglichen werden.
  Es sind folgende Werte definiert:
  1.) keine Inspektion.
  2.) nach Prozent
  3.) Komplexität ;
  4.) Size ;
  5.) defect density

4.6.5.2 Ausgänge:

- True_OUT: MU Output, wenn die Entscheidung True ergibt
- False_OUT: MU Output, wenn die Entscheidung False ergibt

4.6.6 Benötigte und Produzierte Daten

MU Attribute:

Attribut	Name im Modell	Kommentar
Wert 1	Defect Density	Wird vorher ausgerechnet da kein Attribut
Wert 2	Size	
Wert 3	Complexity	
Wert 4	Unit_Number	

Datenbank Werte

Attribut	Name in DB	Kommentar
DB_Wert 1	Defect Density Threshold	Aus General Input Data
DB_Wert 2	Size Threshold	Aus General Input Data
DB_Wert 3	Complexity Threshold	Aus General Input Data
DB_Wert 4	Percent Threshold	Aus General Input Data

Input-Daten aus Datenbank:

Aus der Datenbank werden die Schwellenwerte eingelesen

TABELLE: General Input Data

Diese Tabelle enthält mehrere Werte die zur Kalibrierung des Modells notwendig sind.

Bezeichnung	**Abk.**	**Einheit**
Defect Density Threshold	DDT	
Size Threshold	ST	
Complexity Threshold	CT	
Percent Threshold	PT	

4.6.7 Formeln und Verteilungen

Es werden hier keine Verteilungen verwendet. Als Formel oder Entscheidungsvorschrift kann man die Vergleichsoperation ansehen, diese ist einem Block hinterlegt, der fünf Eingänge besitzt.

Diese sind folgendermaßen benannt

Item_Value	Wert des gewählten Attributes
Threshold	Gewählter Schwellenwert aus der DB
INSPECT	Zahlenwert des Inspektionskriteriums
Gesamt_In	Gesamtanzahl der MU bisher
Input_OUT	Gesamtanzahl der auf True getestenen MUs

Der sog. True-Ausgang ist mit dem Inspektionsblock verbunden, während der False-Ausgang diesen umgeht.

```
If (INSPECT > 1 AND INSPECT < 5){
        if (Item_value >= Threshold)
                Path = YesPath;
                    else

                Path = NoPath;}

IF(INSPECT == 1){
        If(Insp_Out/Gesamt_IN < threshold)
                Path = YesPath;
        else
                Path = NoPath;}

if(INSPECT < 1 AND INSPECT > 4)
        Path = NoPath
```

4.7 Inspections Detailed

4.7.1 Name:

Inspections Detailed

4.7.2 Typ / Aufgabe

Typ:

Komplexe Aktivität

Aufgabe:

Die Inspektion eines Items wird simuliert. Als Teilaktivitäten einer Inspektion werden die Bildung eines Teams, die Vorbereitungsphase (preparation), das anschließende Inspektionstreffen (meeting), die Berechnung des Inspektionsergebnisses, sowie das Rework der gefundenen Fehler abgebildet. Außerdem wird der kumulierte Aufwand ermittelt.

4.7.3 Domäne

Das Modul wurde für die Simulation eines Inspektionsprozesses in der Software-Entwicklung entwickelt.

4.7.4 Moving Units (Items)

Es werden drei Arten von MUs verwendet: 1. die zu inspizierenden Artefakte, die noch mit ihren Autoren gekoppelt sind (Annahme: Autoren begleiten den kompletten Inspektionsprozess); 2. Personen, die als Inspektoren tätig werden; 3. Artefakte die kopiert, mit den Inspektoren gekoppelt und später wieder getrennt und vernichtet werden.

4.7.5 Funktionsweise des Moduls

Die MUs der zu inspizierenden Artefakte kommen über Item_and_Dev_In in das Modul. Über insp_needed wird die Anzahl der angeforderten Inspektoren an ein anderes Modul (z.B. vom Typ Preemptive_Activity) gemeldet. Die MUs der Inspektoren kommen über Insp_In in das Modell.

Nach der Bildung eines Teams (build team) werden die Teilprozesse Preparation (mit anschließendem Update der Skills) sowie das Meeting (mit anschließender Berechnung des Ergebnisses). Die Inspektoren-MUs verlassen danach das Modul über Inspectors_Out. Für die anderen MUs wird die Gesamtzahl der gefundenen Fehler ermittelt und anschließend ein Rework durchgeführt. Abschließend verlassen die Entwickler-Artefakt-MUs das Modul über Items_and_Dev_Out. Der kumulierte Gesamtaufwand bei Inspektionen wird errechnet und über Insp_Effort nach außen gegeben.

4.7.5.1 Eingänge:

- Item_and_Dev_In: Eingang für MUs bestehend aus Artefakt und Entwickler
- Insp_In: Eingang für MUs der Inspektoren

4.7.5.2 Ausgänge:

- Item_and_Dev_Out: Ausgang für MUs bestehend aus Artefakt und Entwickler
- Inspectors_Out: Ausgang für MUs der Inspektoren
- Insp_Effort_Out: der kumulierte Aufwand bei der Inspektion.
- insp_needed_out: meldet die Anforderung von Inspektoren an ein anderes Modul, welches Preemption zulässt.

4.7.6 Benötigte und produzierte Daten

Hier sind nur einige der benötigten und produzierten Daten aufgeführt. Weitere werden in den Beschreibungen der Unter-Aktivitäten genannt:

Build Team, Preparation Activity, Meeting Activity, Meeting Result, Rework Activity

In diesem Modul werden als direkte Input-Daten einige benötigt, welche in den Attributen der MUs gespeichert sind.

MU Attribute:

Attribut	Name im Modell	Kommentar
ID des Items	Unit_Number	zur Identifikation
Author_Of		Zuordnung eines Dokuments zum Entwickler

4.7.7 Formeln und Verteilungen

Außer in den Untermodulen (Build Team, Preparation Activity, Meeting Activity, Meeting Result, Rework Activity) werden keine Formeln und Verteilungen verwendet.

4.8 Build Team

4.8.1 Name:

Build Team

4.8.2 Typ / Aufgabe

Typ:

Aktivität

Aufgabe:

Für ein Artefakt-MU (welches noch mit seinem Entwickler-MU verbunden ist) werden mit Inspektoren-MUs angefordert, um ein Inspektionsteam zu bilden. Die Auswahl der Inspektoren erfolgt „zufällig" durch Preemption. Außerdem wird die Fehler-Überseh-Wahrscheinlichkeit mit 1 initialisiert.

4.8.3 Domäne

Das Modul wurde für die Simulation der Teambildung für einen Inspektionsprozess in der Software-Entwicklung entwickelt. Es kann auch grundsätzlich für die Simulation der Bildung anderer Teams angepasst werden.

4.8.4 Moving Units (Items)

Es werden zwei Arten von MUs verwendet: 1. die zu inspizierenden Artefakte, die noch mit ihren Autoren gekoppelt sind (Annahme: Autoren begleiten den kompletten Inspektionsprozess); 2. Personen, die als Inspektoren tätig werden.

4.8.5 Funktionsweise des Moduls

Die MUs der zu inspizierenden Artefakte kommen über Item_and_Dev_In in das Modul. Über insp_needed wird die Anzahl der angeforderten Inspektoren an ein anderes Modul (z.B. vom Typ Preemptive_Activity) gemeldet. Über einen Schalter im Modul wird festgelegt, ob diese Anzahl zufällig (Schalter = 0) oder über eine Tabelle (Schalter = 1) bestimmt wird. Die MUs der Inspektoren kommen über Insp_In in das Modul.

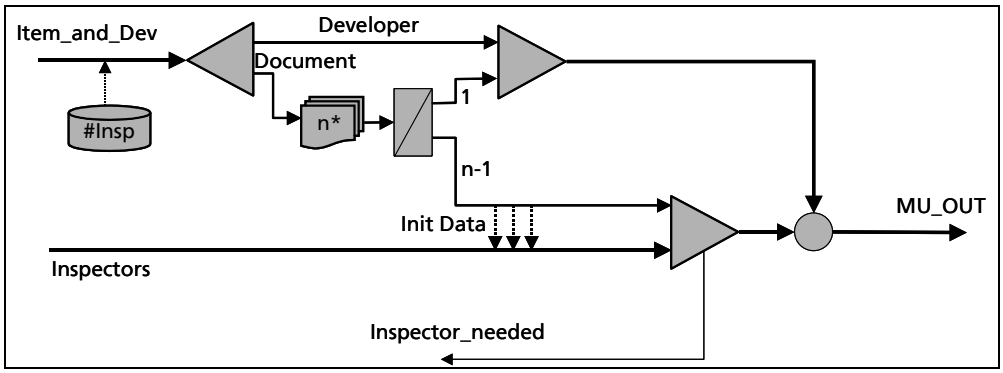


Fig 6: Build Team

Das Artefakt wird zur Kopplung mit jedem vorgesehenen Inspektor kopiert.

Aus technischen Gründen wird das Artefakt zunächst von zugehörigen Entwickler getrennt und nach dem Kopieren wieder mit dem Entwickler und mit den angeforderten Inspektoren gekoppelt. Die gekoppelten MUs, d.h. Entwickler mit Artefakt, Inspektoren mit Artefakt, verlassen das Modul über MU_OUT.

4.8.5.1 Eingänge:

- Item_and_Dev_In: Eingang für MUs bestehend aus Artefakt und Entwickler
- Insp_In: Eingang für MUs der Inspektoren

4.8.5.2 Ausgänge:

- MU_OUT: Ausgang für MUs bestehend aus Artefakt und Inspektor/Entwickler
- insp_needed_out: meldet die Anforderung von Inspektoren an ein anderes Modul, welches Preemption zulässt.

4.8.6 Benötigte und produzierte Daten

MU Attribute:

Attribut	Name im Modell	Kommentar
ID des Items	Unit_Number	zur Identifikation
	No_inspectors	Anzahl an Inspektoren

Input-Daten aus Datenbank:

Zur Berechnung der Dauer der Aktivität wird eine Verteilung aus der Datenbank gelesen. Bei Bedarf können noch weitere Informationen in den Equation Block eingegeben werden. enlich

TABELLE: Unit Input Table

Bezeichnung	Abk.	Erklärung
Number of Inspectors	no_insp	Anzahl an Inspektoren

TABELLE: General Input Data

Diese Tabelle enthält mehrere Werte die zur Kalibrierung des Modells notwendig sind.

Bezeichnung	Abk.	Erklärung
Number of Inspectors	no_insp	Anzahl an Inspektoren, Default-Wert für die Artefakte

Output-Daten in MU-Attribute:

Attribut	Name im Modell	Kommentar
	Inspector_of	Festlegung, ob Person Inspektor des Artefakts ist
	No_inspectors	Anzahl an Inspektoren
	Personnumber	wird gelöscht für kopierte Artefakte
	Author_of	wird gelöscht für kopierte Artefakte

Output-Daten in Datenbank

Tabelle Unit Input Table

Bezeichnung	Abk.	Erklärung
Number of Inspectors	no_insp	Anzahl an Inspektoren

Tabelle Inspection Meeting

Bezeichnung	Abk.	Erklärung
Overlook Probability	Overlook_p	Überseh-Wahrscheinlichkeit für einen Fehler, wird hier initialisiert mit 1

4.8.7 Formeln und Verteilungen

Keine.

4.9 Preparation Activity

4.9.1 Name:

Preparation Activity

Preparation Activity ist strukturiert wie das Template Activity, wobei hier die Variante ohne den Batch und Unbatch gewählt wurde.

4.9.2 Typ / Aufgabe

Typ:

Aktivität

Aufgabe:

Bildet den Fluss eines MUs ab, welches aus einer Person gekoppelt mit einem Artefakt besteht. Die durchgeführte Tätigkeit besteht in der Überprüfung eines Artefakts auf Fehler im Rahmen der Preparation-Phase einer Inspektion. Es werden verschiedene Daten bezüglich der Zeit (Dauer, Anfangs- und Endzeitpunkt), der individuell gefundenen Fehler, sowie der kumulierten Fehler-Überseh-Wahrscheinlichkeit gespeichert.

4.9.3 Domäne

Dieses Modul wurde für die Simulation einer Preparation-Aktivität in der Software-Entwicklung entwickelt.

4.9.4 Moving Units (Items)

Das durchgeleitete MU besteht aus einer Person gekoppelt mit einem Item.

4.9.5 Funktionsweise des Moduls

Das Modul hat einen Eingang und einen Ausgang für MUs. Im Modul wird die Aktivitätsdauer und die Anzahl der gefundenen Fehler berechnet.

4.9.5.1 Eingänge:

- MU_IN: Eingang für MUs

4.9.5.2 Ausgänge:

- MU_OUT: Ausgang für MUs
- Prep_Effort_Out: der kumulierte Aufwand.

4.9.6 Benötigte und produzierte Daten

In diesem Modul werden als Input-Daten einige Daten benötigt, welche in den Attributen der MUs gespeichert sind, sowie einige aus der Datenbank zur Berechnung der Zeit für die Aktivität.
Output-Daten sind die Informationen, welche in die Datenbank geschrieben werden (etwa Zeit (Dauer, Start- und Endzeit), die IDs der MUs, die individuell gefundenen Fehler, die kumulierten Fehler-Überseh-Wahrscheinlichkeit) sowie ein Ausgabewert des Moduls.

MU-Attribute:

Attribut	Name im Modell	Kommentar
ID des Items	Unit_Number	zur Identifikation
ID der Person	Personnumber	zur Identifikation
Defects	d	aktuelle Anzahl Fehler
Defect Detection Skill	DDS	aktueller Skill-Wert der Person in Bezug auf Fehlererkennung
Inspected_Size	ILOC	zu inspizierende Größe des Artefakts

Input-Daten aus Datenbank:

Zur Berechnung der Dauer der Aktivität wird eine Verteilung aus der Datenbank gelesen.

TABELLE: General Distributions

Diese Tabelle enthält neben anderen Verteilungen auch die Verteilung für die Berechnung der Zeit in der Preparation Activity.

Bezeichnung	Abk.	Verteilungsdaten
Random Preparation Time Factor	RPTF	Log Normal Verteilung

TABELLE: General Input Data

Diese Tabelle enthält mehrere Werte die zur Kalibrierung des Modells notwendig sind.

Bezeichnung	Abk.	Einheit
Max Preparation Productivity	MPP	Preparation-Produktivität für sehr erfahrenen Entwickler

TABELLE: Actual Skills

Diese Tabelle enthält mehrere Werte die zur Kalibrierung des Modells notwendig sind.

Bezeichnung	Abk.	Einheit
Preparation Productivity Skill	PPS	aktueller Skill-Wert der Person in Bezug auf Fehlererkennung

TABELLE: Unit Input Table

Diese Tabelle enthält verschiedene Angaben zu den Artefakten.

Bezeichnung	Abk.	Einheit
Inspection Technique	IT	zu verwendende Technik bei der Inspektion des Artefakts

TABELLE: Inspection Factors

Diese Tabelle enthält verschiedene Angaben zu der Effizienz verschiedener Inspektionstechniken.

Bezeichnung	Abk.	Einheit
Inspection (Technique) Factor	itf	Wahrscheinlichkeit für einen sehr erfahrenen Inspektor, einen Fehler mit der gegebenen Technik zu finden

TABELLE: Inspection Meeting

Diese Tabelle enthält verschiedene Angaben zum Ergebnis (Meeting) einer Inspektion.

Bezeichnung	Abk.	Einheit
Overlook Probability	Overlook_p	Wahrscheinlichkeit, einen Fehler zu übersehen

Output-Daten in MU-Attribute:

Attribut	Name im Modell	Kommentar
Preparation time	Prep_time	benötigte Zeit für die Aktivität

Output-Daten in Datenbank

In der Datenbank wird eine Tabelle mit allen Informationen beschrieben, welche in diesem Modul anfallen.

Tabelle Inspection_Preparation

Prep_ID	Unit Number	Developer	Prepara-tion_Time	Start_Time	End_Time	De-tected

Für jedes MU, das durch dieses Modul geht, wird eine Zeile angelegt. Die Zeile wird über die Information in „ID" bzw. „Prep_ID", welche über einen Count festgelegt wird.

TABELLE: Inspection Meeting

Diese Tabelle enthält verschiedene Angaben zum Ergebnis (Meeting) einer Inspektion.

Bezeichnung	**Abk.**	**Einheit**
Overlook Probability	Over-look_p	Wahrscheinlichkeit, einen Fehler zu übersehen

4.9.7 Formeln und Verteilungen

Es wird nur eine Verteilung zur Berechnung der Zeit in der Aktivität verwendet. Die Verteilungsparameter können geändert werden.

Verteilung:

	Typ	Param. 1	Param. 2	Param. 3
Random Prepara-tion Time Factor	Log Normal	Mean: 1	Std. Dev.: 0,2	

Formeln:

$$Preparation\_Time = RPTF * \frac{ILOC}{PPS * MPP}$$

ILOC wird aus den Attributen des MUs gelesen. RPTF ist ein Wert aus der Datenbank und wird durch eine Verteilung realisiert. PPS und MPP kommen aus der Datenbank. Das Ergebnis wird in der Datenbank und im Attribut Prep_Time gespeichert. Falls die Person gleich dem Autor des Artefakts ist, wird der Wert für die Preparation_Time auf 1 [h] gesetzt.

$$Det\_Defects = round(d * itf * DDS)$$

d und DDS sind MU-Attribute. Def_Det_Eff kommt aus der Datenbank. Das Ergebnis wird in der Datenbank gespeichert. Falls die Person gleich dem Autor des Artefakts ist, wird der Wert für Det_Defects auf 0 gesetzt. Die auf diese Weise ermittelten, individuell gefundenen Fehler werden NICHT verwendet, um die Gesamtzahl der gefundenen Fehler in einem Artefakt zu ermitteln.

$$Overlook\_p = Overlook\_p * (1 - itf * DDS)$$

Die Wahrscheinlichkeit, einen Fehler zu übersehen wird aktualisiert, indem der alte Wert (mit 1 initialisiert) mit der Wahrscheinlichkeit multipliziert wird, dass der aktuelle Inspektor auch diesen Fehler übersieht. Diese Wahrscheinlichkeit ist 1 – ift*DDF, die Komplementärwahrscheinlichkeit zu dem Gütefaktor der Inspektionstechnik multipliziert mit dem defect detection skill. Falls die Person gleich dem Autor des Artefakts ist, ändert sich Overlook_p nicht.

4.10 Meeting Activity

4.10.1 Name:

Meeting Activity

4.10.2 Typ / Aufgabe

Typ:

Aktivität

Aufgabe:

Bildet den Fluss einer Gruppe von MUs ab, welches aus Personen gekoppelt mit Items besteht. Die Items einer Gruppe werden zunächst gesammelt und führen dann eine gemeinsame Tätigkeit aus. Anschließend wird die Gruppe wieder getrennt. Es werden verschiedene Daten bezüglich der Zeit (Dauer, Anfangs- und Endzeitpunkt) gespeichert.

4.10.3 Domäne

Dieses Modul wurde für die Simulation eines Inspektions-Meetings in der Software-Entwicklung entwickelt, kann aber auch zur Simulation anderer gemeinsamer Aktivitäten verwendet werden.

4.10.4 Moving Units (Items)

Die durchgeleiteten MUs bestehen aus Personen gekoppelt mit Artefakten.

4.10.5 Funktionsweise des Moduls

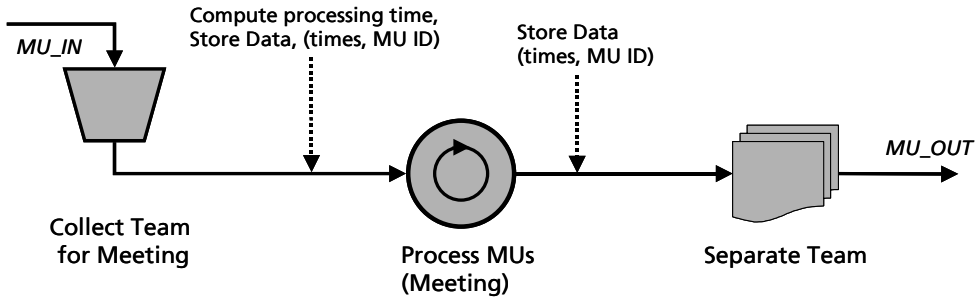


Fig 7: Meeting Activity

Das Modul hat einen Eingang und einen Ausgang für MUs. Im Modul wird die Aktivitätsdauer berechnet. Die Anzahl der gefunden und verbleibenden Fehler wird in dem Block Meeting Result berechnet.

4.10.5.1 Eingänge:

- MU_IN: Eingang für MUs

4.10.5.2 Ausgänge:

- MU_OUT: Ausgang für MUs
- Meeting_Effort_Out: der kumulierte Aufwand.

4.10.6 Benötigte und produzierte Daten

In diesem Modul werden als Input-Daten einige Daten benötigt, welche in den Attributen der MUs gespeichert sind, sowie einige aus der Datenbank zur Berechnung der Zeit für die Aktivität.

Output-Daten sind die Informationen, welche in die Datenbank geschrieben werden (etwa Zeit (Dauer, Start- und Endzeit) sowie die IDs der MUs) sowie ein Ausgabewert des Moduls.

MU Attribute:

Attribut	Name im Modell	Kommentar
ID des Items	Unit_Number	zur Identifikation
ID der Person	Personnumber	zur Identifikation
	Inspector_Of	Zuordnung Inspektor
	No_Inspectors	Anzahl der Inspektoren für ein Artefakt

Input-Daten aus Datenbank:

Zur Berechnung der Dauer der Aktivität wird eine Verteilung aus der Datenbank gelesen.

TABELLE: General Distributions

Diese Tabelle enthält neben anderen Verteilungen auch die Verteilung für die Berechnung der Zeit in Meeting Activity.

Bezeichnung	Abk.	Verteilungsdaten
Random Meeting Time Factor	RMTF	Log Normal Verteilung

TABELLE: General Input Data

Diese Tabelle enthält mehrere Werte die zur Kalibrierung des Modells notwendig sind.

Bezeichnung	Abk.	Einheit
Average Meeting Productivity	AMP	durchschnittliche Produktivität in Inspektions-Meetings

TABELLE: Unit Input Table

Diese Tabelle enthält verschiedene Angaben zu den Artefakten.

Bezeichnung	Abk.	Einheit
Inspected Lines of Code	ILOC	zu inspizierende Größe des Artefakts

Output-Daten in MU-Attribute:

Attribut	Name im Modell	Kommentar
inspection meeting time	Insp_meeting_time	benötigte Zeit für die Aktivität

Output-Daten in Datenbank

In der Datenbank wird eine Tabelle mit allen Informationen beschrieben, welche in diesem Modul anfallen.

Tabelle Inspection_Preparation

Prep_ID	Unit Number	Developer	Preparation_Time	Start_Time	End_Time	Detected

Für jedes MU, das durch dieses Modul geht, wird eine Zeile angelegt. Die Zeile wird über die Information in „ID“ bzw. „Prep_ID“, welche über einen Count festgelegt wird.

4.10.7 Formeln und Verteilungen

Es wird nur eine Verteilung zur Berechnung der Zeit in der Aktivität verwendet. Die Verteilungsparameter können geändert werden.

Verteilung:

	Typ	Param. 1	Param. 2	Param. 3
Random Meeting Time Factor	Log Normal	Mean: 1	Std. Dev.: 0,2	

Formeln:

$$Meeting\_Time = RMTF \cdot \frac{ILOC}{AMP}$$

RMTF ist ein Wert aus der Datenbank und wird durch eine Verteilung realisiert. ILOC und AMP kommen aus der Datenbank. Das Ergebnis wird Attribut Meeting_Time gespeichert.

$$Inspection\_Effort = Meeting\_Time \cdot No\_Inspectors$$

Meeting_Time und No_Inspectors sind Attribute. Das Ergebnis wird weiterverwendet zur Berechnung des nach außen gegebenen kumulierten Aufwands.

4.11 Meeting Result

4.11.1 Name:

Meeting Result

4.11.2 Typ / Aufgabe

Typ:

Berechnung, Änderung von Daten

Aufgabe:

Nach Durchführung eines Meeting wird die Gesamt-Anzahl gefundener Fehler für ein Item berechnet.

4.11.3 Domäne

Das Modul wurde im Rahmen der Simulation eines Inspektionsprozesses in der Software-Entwicklung entwickelt.

4.11.4 Moving Units (Items)

Es werden MUs verwendet, die aus gekoppelten Artefakten und zugehörigen Entwicklern bestehen.

4.11.5 Funktionsweise des Moduls

Die MUs kommen über MU_IN in das Modell. Das Inspector_Of-Attribut wird gelöscht, die End-Zeit des Meetings wird in die Datenbank geschrieben. Die Anzahl der gefundenen und unentdeckten Fehler wird berechnet und in die Datenbank geschrieben. Die kumulierte Anzahl der gefundenen Fehler wird nach außen gegeben. Abschließend verlassen die MUs das Modul über MU_OUT.

4.11.5.1 Eingänge:

- MU_IN: Eingang für MUs aus gekoppelten Artefakten und Entwicklern

4.11.5.2 Ausgänge:

- MU_OUT: Ausgang für MUs aus gekoppelten Artefakten und Entwicklern
- found_defects_out: die kumulierte Anzahl der gefundenen Fehler

4.11.6 Benötigte und produzierte Daten

In diesem Modul werden als Input-Daten einige Daten benötigt, welche in den Attributen der MUs gespeichert sind, sowie einige aus der Datenbank zur Berechnung der Anzahl der gefundenen Fehler.

MU Attribute:

Attribut	Name im Modell	Kommentar
ID des Items	Unit_Number	zur Identifikation
defects	d	Anzahl der Fehler im Artefakt

Input-Daten aus Datenbank:

TABELLE: General Distributions

Diese Tabelle enthält neben anderen Verteilungen auch die Verteilung für die Berechnung der Anzahl der gefundenen Fehler.

Bezeichnung	Abk.	Verteilungsdaten
Random Defect Detection Factor	RDDF	Log Normal Verteilung

TABELLE: Inspection Meeting

Bezeichnung	Abk.	Erklärung
Overlook probability	over-look_p	Wahrscheinlichkeit, einen Fehler zu übersehen

Output-Daten in MU-Attribute:

Attribut	Name im Modell	Kommentar
	Det_Defects	Anzahl der insgesamt gefundenen Defects

Output-Daten in Datenbank

Tabelle Inspection Meeting

Bezeichnung	Abk.	Erklärung
found_defects	fd	Anzahl der gefundenen Fehler
undetected defects	ud	Anzahl der nichtgefundenen Fehler

4.11.7 Formeln und Verteilungen

$$fd = round(RDDF \cdot d \cdot (1 - overlook\_p))$$

RDDF ist ein Wert aus der Datenbank und wird durch eine Verteilung realisiert. d ist als Attribut gespeichert. overlook_p kommt aus der Datenbank. Das Ergebnis found_defects wird als Attribut und in der Datenbank gespeichert und als kumulierter Wert nach außen gegeben. Die Rundung erfolgt aufgrund der Ganzzahligkeit der Variablen.

$$ud = d - fd$$

d und fd sind als Attribute gespeichert. Das Ergebnis undetected defects:wird in der Datenbank gespeichert.

4.12 Rework Activity

4.12.1 Name:

Rework Activity
Eine Variante des Templates Activity ohne die Batch- und Unbatch-Blöcke.

4.12.2 Typ / Aufgabe

Typ:

Aktivität

Aufgabe:

Bildet den Fluss eines MUs ab, welches aus einer Person gekoppelt mit einem Item besteht. Die durchgeführte Tätigkeit besteht in der Beseitigung von gefundenen Fehlern in einem Artefakt. Es werden verschiedene Daten bezüglich der Zeit (Dauer, Anfangs- und Endzeitpunkt) gespeichert.

4.12.3 Domäne

Dieses Modul wurde für die Simulation einer Rework-Aktivität in der Software-Entwicklung entwickelt,.

4.12.4 Moving Units (Items)

Das durchgeleitete MU besteht aus einer Person gekoppelt mit einem Artefakt.

4.12.5 Funktionsweise des Moduls

Das Modul hat einen Eingang und einen Ausgang für MUs. Im Modul wird die Aktivitätsdauer und die Anzahl der neuen bzw. nichtbeseitigten Fehler berechnet

4.12.5.1 Eingänge:

- MU_IN: Eingang für MUs.

4.12.5.2 Ausgänge:

- MU_OUT: Ausgang für MUs.
- Rem_Def_Out: die kumulierten verbleibenden Fehler.
- Effort_Out: der kumulierte Aufwand.

4.12.6 Benötigte und produzierte Daten

In diesem Modul werden als Input-Daten einige Daten benötigt, welche in den Attributen der MUs gespeichert sind, sowie einige aus der Datenbank zur Berechnung der Zeit für die Aktivität.
Output-Daten sind die Informationen, welche in die Datenbank geschrieben werden (etwa Zeit (Dauer, Start und Endzeit) sowie die IDs der Mus) sowie zwei Ausgabewerte des Moduls.

MU Attribute:

Attribut	Name im Modell	Kommentar
ID des Items	Unit_Number	zur Identifikation
ID der Person	Personnum-ber	zur Identifikation
Defects	d	aktuelle Anzahl Fehler
Det_Defects	fd	gefundene Fehler
Coding Productivity Skill	CPS	aktueller Skill-Wert der Person in Bezug auf Coding

Input-Daten aus Datenbank:

Zur Berechnung der Dauer der Aktivität wird eine Verteilung aus der Datenbank gelesen. Bei Bedarf können noch weitere Informationen in den Equation Block eingegeben werden.

TABELLE: General Distributions

Diese Tabelle enthält neben anderen Verteilungen auch die Verteilung für die Berechnung der Zeit in der Rework Activity.

Bezeichnung	Abk.	Verteilungsdaten
Random Coding Time Factor	RCTF	Log Normal Verteilung

TABELLE: General Input Data

Diese Tabelle enthält mehrere Werte die zur Kalibrierung des Modells notwendig sind.

Bezeichnung	Abk.	Einheit
Average Defect Size	ADS	LOC per defect; resultiert aus dem Verhältnis der durchschnittlichen Überarbeitungsdauer eines Defects und der durchschnittlichen Kodierungsproduktivität
Max Coding Productivity	MCP	Kodierungsproduktivität für sehr erfahrene Entwickler
Rework Defects Factor	RDF	Anteil übersehenen/neuen Fehler beim Rework

Output-Daten in MU-Attribute:

Attribut	Name im Modell	Kommentar
Defects	d	aktuelle (verbleibende) Anzahl Fehler
new_Def	new_Def	neue/übersehen Fehler

Output-Daten in Datenbank

In der Datenbank wird eine Tabelle mit allen Informationen beschrieben, welche in diesem Modul anfallen.

Tabelle Inspection_Rework

Unit Number	Developer	Rework_ Time	Start_Time	End_Time	Defects_ Remaining	New_ Defects

Für jedes MU, das durch dieses Modul geht, wird eine Zeile angelegt. Die Zeile wird über die Information in „Unit Number" festgelegt. Dadurch ist es nicht möglich, das gleiche MU (mit der gleichen Unit Number) mehrfach durch das Modul zu schicken.

4.12.7 Formeln und Verteilungen

Es wird nur eine Verteilung zur Berechnung der Zeit in der Aktivität verwendet. Die Verteilungsparameter können geändert werden.

Verteilung:

	Typ	Param. 1	Param. 2	Param. 3
Random Coding Time Factor	Log Normal	Mean: 1	Std. Dev.: 0,2	

Formeln:

$$Rework\_Time = RCTF \cdot fd \cdot \frac{ADS}{CPS \cdot MCP}$$

fd und CPS werden aus den Attributen des MUs gelesen. RCTF ist ein Wert aus der Datenbank und wird durch eine Verteilung realisiert. ADS und MCP kommen aus der Datenbank. Das Ergebnis wird in der Datenbank gespeichert.

$$new\_def = round(fd \cdot RDF)$$

RDF kommt aus der Datenbank. Das Ergebnis wird in der Datenbank abgelegt und als Attribut gespeichert.. Die Rundung erfolgt wegen der Ganzzahligkeit der Variablen.

$$Remaining\_Defects = d - fd + new\_def$$

d, fd und new_def werden aus den Attributen des MUs gelesen. Das Ergebnis wird in der Datenbank abgelegt und als Attribut Def gespeichert.

4.13 Test

4.13.1 Name

Test
Das zugrunde liegende Template ist Activity

4.13.2 Typ / Aufgabe

Typ:

Aktivität

Aufgabe:

Verbindet ein MU mit einer zweiten MU (Ressource), z.B. einer Person, um eine Tätigkeit auszuführen. Nach der Ausführung werden beide MUs wieder getrennt. Es werden verschiedene Daten bezüglich der Zeit (Dauer, Anfangs- und Endzeitpunkt) gespeichert. Zusätzlich werden noch weitere Berechnungen ausgeführt, hier die Anzahl der gefunden und verbleibenden Fehler.

4.13.3 Domäne

Dieses Modul wurde für die Simulation einer Aktivität in der Software-Entwicklung entwickelt, dort allerdings nicht für eine spezielle Domäne. Ein Einsatz in verschiedenen Domänen aus dem Software-Entwicklungsbereich ist denkbar.

4.13.4 Moving Units (Items)

Es sind zwei Arten von MUs notwendig.

Eine MU stellt das Objekt dar, welches in der Aktivität verändert, bearbeitet oder erstellt wird. Das zweite stellt die benötigte Ressource dar, z.B. eine Person oder ein Werkzeug. Es werden beide MUs benötigt damit das Modul funktioniert

4.13.5 Funktionsweise des Moduls

Das Modul hat zwei Eingänge für MUs und ebenso zwei Ausgänge für MUs. Ein Eingang sowie Ausgang ist für die MUs bestimmt, die das Objekt, welches bearbeitet, verändert oder erstellt wird, bestimmt, z.B. Testen eines zuvor erstellten Teils eines Software-Produkts.

Der zweite Ein- sowie Ausgang ist für die Ressource bestimmt, welche benötigt wird, um die Aktivität, hier Testen, auszuführen. Da die Ressource als MU mit Attributen ausgeführt ist, bietet diese Art der Benutzung einer Ressource die Möglichkeit, Eigenschaften der Ressource zu modellieren und in die Berechnungen mit einzubeziehen.

Beide MUs werden am Anfang zu einem Paar zusammengefügt und am Ende wieder getrennt. Dadurch ist es möglich auf die Attribute beider MUs zuzugreifen.

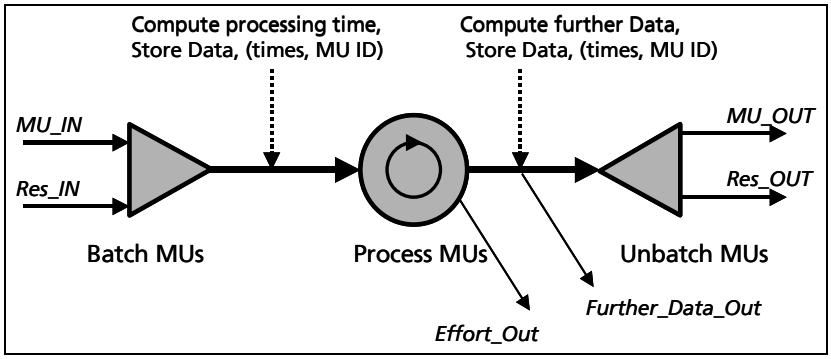


Fig 8: Schematische Darstellung des Flusses der MUs durch Simple Activity

Die MU für die Ressource wird erst in das Modul „gezogen", wenn eine MU am Eingang für das zu bearbeitende Objekt vorhanden ist. Dadurch wird die Ressource erst aus dem Ressource Pool entfernt, wenn sie auch wirklich benötigt wird und steht bis dahin für andere Aufgaben zur Verfügung.

Bei der Auftrennung der MUs am Ende des Moduls behält jedes Modul seine Attribute. Neue Attribute die während der Vereinigung hinzugefügt wurden, sind danach in keiner der MUs enthalten.

4.13.5.1 Eingänge:

- MU_IN: Eingang für MU, typischerweise das zu bearbeitende Objekt.
- Res_IN: Eingang für MU Ressource, typischerweise die Ressource, welche zur Bearbeitung benötigt wird.

4.13.5.2 Ausgänge:

- MU_OUT: Ausgang für MU, typischerweise das bearbeitete Objekt.
- Res_OUT: Ausgang für MU Ressource, typischerweise die Ressource , welche zur Bearbeitung benötigt wurde.

Effort_Out: der kumulierte Aufwand.
DefFound_OUT: kumulierte Anzahl der gefunden Fehler.
RemainingDefects_OUT: kumulierte Anzahl der verbleibenden Fehler.

4.13.6 Benötigte und produzierte Daten

In diesem Modul werden als Input-Daten die Daten, welche in den Attributen der MUs gespeichert sind, sowie die aus der Datenbank benötigten Daten zur Berechnung der Zeit für die Aktivität verwendet.
Output-Daten sind die Informationen, welche in die Datenbank geschrieben werden. Dazu zählen alle Informationen über die Zeit (Dauer, Start- und Endzeit) sowie die IDs der MUs.

MU-Attribute:

Zu bearbeitendes Objekt:

Attribut	Name im Modell	Kommentar
ID der MU	Unit_Number	zur Identifikation der MU
Größe der MU	Size	Hier kann wahlweise die Gesamtgröße oder geänderte Größe angegeben sein.
Komplexität	Complexity	
Fehler	Defects	Anzahl der unentdeckten Fehler

Resource MU:

Attribut	Name im Modell	Kommentar
ID der MU	Personnumber	um die Ressource zuordnen zu können.
Test - Skill	TQS	Qualitvitäts-Skill des Testens

Input-Daten aus Datenbank:

Zur Berechnung der Dauer der Aktivität wird der maximale Wert der Produktivität beim Testen eines Software-Artefakts aus der Datenbank gelesen. Dieser Wert, Maximale Test Produktivität, wird mit dem individuellen Skill-Wert multipliziert, um den für diesen Entwickler typischen Produktivitätswert zu ermitteln.

TABELLE: General Input Data

Diese Tabelle enthält unter anderem maximale Produktivitätswerten auch den für den Test notwendigen Wert.

Bezeichnung	Abk.	Kommentar
Max Test Productivity	MTP	die maximal pro Strunde testbare Anzahl der Lines of Code

Output-Daten in Datenbank

In der Datenbank wird eine Tabelle mit allen Informationen beschrieben, welche in diesem Modul anfallen.

Tabelle Test

Unit Number	Developer	Test Time	Start Time	End Time	Detected Defects	Defects

Für jedes MU, das durch dieses Modul geht, wird eine Zeile angelegt. Die Zeile wird über die Information in „Unit Number“ festgelegt. Dadurch ist es nicht möglich, das gleiche MU (mit der gleichen Unit Number) mehrfach durch das Modul zu schicken. Hier müsste dann eine Append Funktion genutzt werden, die das SDI Interface von Extend aber nicht bietet. Workaround wäre eine laufende Nummer zu vergeben.

4.13.7 Formeln und Verteilungen

Es werden keine Verteilungen zur Berechnung der Testzeiten herangezogen

Formel:

Annahme für die Berechnung der Zeit, welche für das Testen aufgewendet wird, ist das am Anfang viele Fehler gefunden werden und weitere Fehler immer mehr Aufwand erfordern, um sie zu identifizieren [=67 - Weinberg 1991 Quality Software Man...=]. Ein solcher Zusammenhang kann durch eine Exponentialfunktion beschrieben werden. Bei der Umstellung der Exponentialfunktion nach der Zeit (TestTime) ergibt sich folgende Formel

$$TestTime = -\ln(1 - TestThoroughness) \cdot \frac{1}{DFR \cdot TQS} \cdot Defects$$

Das Attribut *Defects* wir aus dem MU gelesen, welches das zu testende Artefakt ist. *TQS (Test Quality Skill)* ist ein Attribut der Ressource und *DFR (Defect Find Rate)* ist der maximale Wert für die Testleistung aus der Datenbank Dabei ist TestThoroughness ein Wert zwischen 0 und 1.

Neben der Zeit für die Testaktivität wird noch die Anzahl der Fehler berechnet , welche in der errechneten Zeit gefunden werden. Der Wert *Found Defects* wird aus der Gesamtanzahl von *Defects* und einer Exponentialfunktion berechnet. Daraus lassen sich dann durch Subtraktion die verbleibenden Fehler berechen.

$$FoundDefects = Defects\left(1 - e^{-\frac{TestTime \cdot DFR \cdot TQS}{Defects}}\right)$$

Die hier angegebene Exponentialfunktion trägt der Annahme Rechung, dass nach einer anfänglich hohen Anzahl von gefunden Fehlern pro Zeiteinheit diese zurück geht und weitere Fehler nur mit größerem Zeitaufwand gefunden werden können.

4.14 Rework

4.14.1 Name:

Rework
Das zugrunde Liegende Template ist Simple Activity

4.14.2 Typ / Aufgabe

Typ:

Aktivität

Aufgabe:

Verbindet ein MU mit einer zweiten MU (Ressource) z.B. einer Person, um eine Tätigkeit auszuführen. Nach der Ausführung werden beide MUs wieder getrennt. Es werden verschiedene Daten bezüglich der Zeit (Dauer, Anfangs- und Endzeitpunkt) gespeichert.

4.14.3 Domäne

Dieses Modul wurde für die Simulation einer Aktivität in der Software-Entwicklung entwickelt, dort allerdings nicht für eine spezielle Domäne. Ein Einsatz in verschiedenen Domänen aus dem Software-Entwicklungsbereich ist denkbar.

4.14.4 Moving Units (Items)

Es sind zwei Arten von MUs notwendig.

Eine MU stellt das Objekt dar, welches in der Aktivität verändert, bearbeitet oder erstellt wird. Das zweite stellt die benötigte Ressource dar, z.B. eine Person oder ein Werkzeug. Es werden beide MUs benötigt damit das Modul funktioniert

4.14.5 Funktionsweise des Moduls

Das Modul hat zwei Eingänge für MUs und ebenso zwei Ausgänge für MUs. Ein Eingang sowie Ausgang ist für die MUs bestimmt, die das Objekt, welches bearbeitet, verändert oder erstellt wird, z.B. Design erstellen aus Anforderungen.

Der zweite Ein- sowie Ausgang ist für die Ressource bestimmt, welche benötigt wird, um die Aktivität auszuführen. Da die Ressource als MU mit Attributen ausgeführt ist, bietet diese Art der Benutzung einer Ressource die Möglichkeit, Eigenschaften der Ressource zu modellieren und in die Berechnungen mit einzubeziehen.

Beide MUs werden am Anfang zu einem Paar zusammengefügt und am Ende wieder getrennt. Dadurch ist es möglich, auf die Attribute beider MUs zuzugreifen.

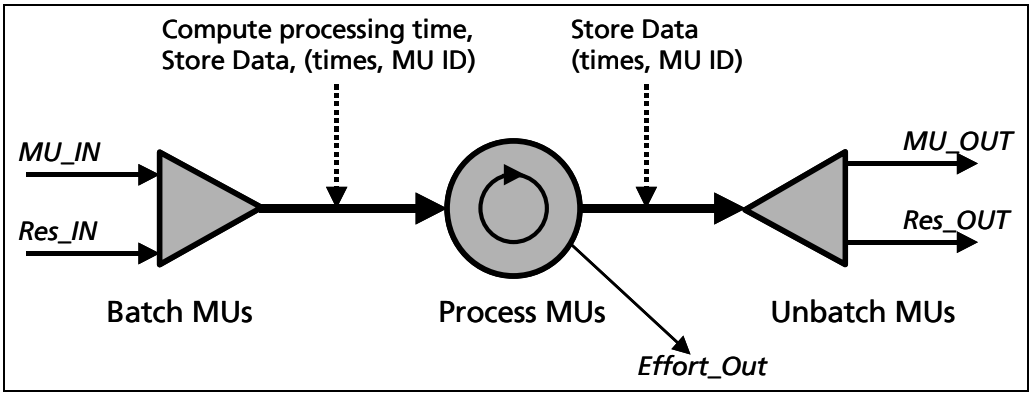


Fig 9: Schematische Darstellung des flusses der MUs durch Simple Activity

Die MU für die Ressource wird erst in das Modul „gezogen", wenn eine MU am Eingang für das zu bearbeitende Objekt vorhanden ist. Dadurch wird die Ressource erst aus dem Resource Pool entfernt, wenn sie auch wirklich benötigt wird und steht bis dahin für andere Aufgaben zur Verfügung.

Bei der Auftrennung der MUs am Ende des Moduls behält jedes Modul seine Attribute. Neue Attribute, die während der Vereinigung hinzugefügt wurden, sind danach in keiner der MUs enthalten.

4.14.5.1 Eingänge:

- MU_IN: Eingang für MU, typischerweise das zu bearbeitende Objekt
- Res_IN: Eingang für MU Ressource, typischerweise die Ressource , welche zur Bearbeitung benötigt wird.

4.14.5.2 Ausgänge:

- MU_OUT: Ausgang für MU, typischerweise das bearbeitete Objekt
- Res_OUT: Ausgang für MU Ressource, typischerweise die Ressource, welche zur Bearbeitung benötigt wurde

Effort_Out: der kumulierte Aufwand.

4.14.6 Benötigte und produzierte Daten

In diesem Modul werden als Input-Daten die Daten, welche in den Attributen der MUs gespeichert sind, sowie die aus der Datenbank benötigten Daten zur Berechnung der Zeit für die Aktivität verwendet.
Output-Daten sind die Informationen, welche in die Datenbank geschrieben werden. Dazu zählen alle Informationen über die Zeit (Dauer, Start- und Endzeit) sowie die IDs der MUs.

MU-Attribute:

Zu bearbeitendes Objekt:

Attribut	Name im Modell	Kommentar
ID der MU	Unit_Number	zur Identifikation der MU
Anzahl der Fehler	Size	

Ressource MU:

Attribut	Name im Modell	Kommentar
ID der MU	Personnumber	um die Ressource zuordnen zu können.
Coding Skill	CPS	

Input Daten aus Datenbank:

Zur Berechnung der Dauer der Aktivität wird eine Verteilung aus der Datenbank gelesen. Bei Bedarf können noch weitere Informationen in den Equation Block eingegeben werden.

In einer Tabelle ist die Verteilung für die Berechnung der Zeit in Simple Activity abgelegt. Da Simple Activity hier für die Aktivität Rework Code eingesetzt wird, ist die Bezeichnung Random Rework Time Factor. Für andere Aktivitäten muss dies geändert werden. Weitere Daten aus der Datenbank sind die Durchschnittliche Größe eines Fehlers, Average Defect Size, und die maximale Kodierungs-Produktivität der Entwickler.

TABELLE: General Distributions

Bezeichnung	Abk.	Verteilungsdaten
Random Rework Time Factor	RDTF	Log Normal Verteilung

TABELLE: General Input Data

Bezeichnung	Abk.	Verteilungsdaten
Average Defect Size	ADS	
Max Coding Productivity	MCP	

Output-Daten in Datenbank

In der Datenbank wird eine Tabelle mit allen Informationen beschrieben, welche in diesem Modul anfallen.

Tabelle Test_Rework

Unit Number	Developer	Rework Time	Start Time	End Time	Defects

Für jedes MU, das durch dieses Modul geht, wird eine Zeile angelegt. Die Zeile wird über die Information in „Unit Number“ festgelegt. Dadurch ist es nicht möglich, das gleiche MU (mit der gleichen Unit Number) mehrfach durch das Modul zu schicken. Hier müsste dann eine Append Funktion genutzt werden, die das SDI Interface von Extend aber nicht bietet.

4.14.7 Formeln und Verteilungen

Es wird nur eine Verteilung zur Berechnung der Zeit in der Aktivität. Dies ist auch die einzige Formel / Berechnung in diesem Modul.

Vereilung:

	Typ	Param. 1	Param. 2	Param. 3
Random Rework Time Factor	Log Normal	Mean: 1	Std. Dev.: 0,3	Location : 1

Formel:

$$Rework\_Time = RandRewTimeFact \cdot DetDef \cdot AvDefSize \cdot \frac{2}{MaxCodeProd \cdot CPS}$$

Detected Defects (DetDef) und *Code Productivity Skill (CPS)* werden aus den Attributen der MUs gelesen. Der *Random Rework Time Factor (RandRewTime-Fact)*, *Average Defect Size (AvDefSize)* und die *Max Coding Productivity (Max-CodeProd)* sind Werte aus der Datenbank und werden durch Verteilungen realisiert. Das Ergebnis wird in dem Attribut *Rework_Time* abgelegt und in der Datenbank gespeichert. Das Attribut *Rework_Time* wird bei dem Auftrennen der MUs gelöscht aufgrund der Einstellung Preserve Uniqueness in den Batch-Blöcken.

5 Generic Templates

Zuerst werden mehrere sog. Generic Templates (GT) für bestimmte Aufgaben beschrieben. Die Beschreibung orientiert sich an dem in Kapitel 3.3 beschriebenen Template. Generell gilt für die generischen Templates, dass sie weitgehend domänenunabhängig zur Prozess-Modelierung verwendet werden können.

Es werden folgende Templates beschrieben:

GT Simple Activity,
GT Activity,
GT Generate Code Units,
GT Pre-emptive Activity,
GT Set Skill,
GT Routing Decision.

Abhängig von der Häufigkeit sowie der Unterschiede in den Daten, Verteilungen und Formeln der Module werden sie in Kapitel 5.3 als Beispiele aufgeführt.

5.1 Generic Template Simple Activity

5.1.1 Name:

GT Simple Activity

5.1.2 Typ / Aufgabe

Typ:

Aktivität

Aufgabe:

Verbindet ein MU mit einer zweiten MU (Ressource), z.B. einer Person, um eine Tätigkeit auszuführen. Nach der Ausführung werden beide MUs wieder getrennt. Es werden verschiedene Daten bezüglich der Zeit (Dauer, Anfangs- und Endzeitpunkt) gespeichert.

5.1.3 Domäne

Dieses Modul wurde für die Simulation einer Aktivität in der Software-Entwicklung entwickelt, dort allerdings nicht für eine spezielle Domäne. Ein Einsatz in verschiedenen Domänen aus dem Software-Entwicklungsbereich, aber auch in anderen Modellierungsfeldern ist denkbar.

5.1.4 Moving Units (Items)

Es sind zwei Arten von MUs notwendig.

Eine MU stellt das Objekt dar, welches in der Aktivität verändert, bearbeitet oder erstellt wird. Das zweite stellt die benötigte Ressource dar, z.B. eine Person oder ein Werkzeug. Es werden beide MUs benötigt, damit das Modul funktioniert.

5.1.5 Funktionsweise des Moduls

Das Modul hat zwei Eingänge für MUs und ebenso zwei Ausgänge für MUs. Ein Eingang sowie Ausgang ist für die MUs bestimmt, die das Objekt, welches bearbeitet, verändert oder erstellt wird, festlegt, z.B. Design erstellen aus Anforderungen.

Der zweite Ein- sowie Ausgang ist für die Ressource bestimmt, welche benötigt wird, um die Aktivität auszuführen. Da die Ressource als MU mit Attributen ausgeführt ist, bietet sich die Möglichkeit, Eigenschaften der Ressource zu modellieren und in die Berechnungen mit einzubeziehen.

Beide MUs werden am Anfang zu einem Paar zusammengefügt (Batch) und am Ende wieder getrennt (Unbatch). Dadurch ist es möglich, auf die Attribute beider MUs zuzugreifen.

Bei der Auftrennung der MUs am Ende des Moduls behält jedes Modul seine Attribute. Neue Attribute, die während der Vereinigung hinzugefügt wurden, sind danach in keiner der MUs enthalten. Dies wird durch die Selektion von „Preserve Uniqueness“ sowohl im Batch- als auch im Unbatch-Block erreicht.

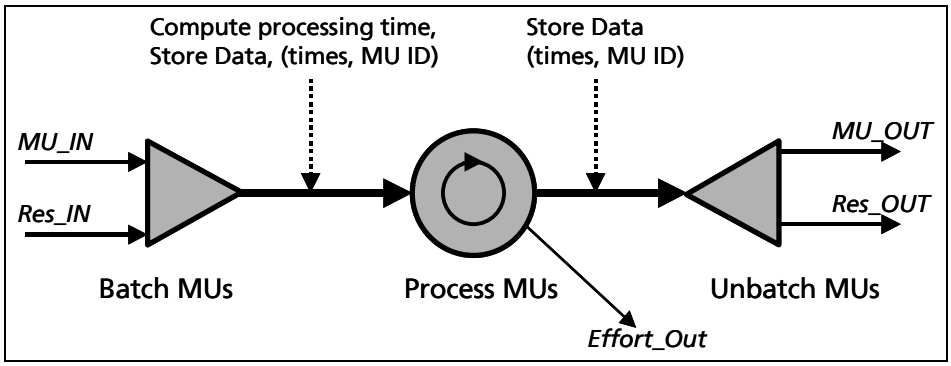


Fig 10: Schematische Darstellung des Flusses der MUs durch ein Simple Activity Modul

Die MU für die Ressource wird erst in das Modul „gezogen", wenn eine MU am Eingang für das zu bearbeitende Objekt vorhanden ist. Dadurch wird die Ressource erst aus dem Resource Pool entfernt, wenn sie auch wirklich benötigt wird und steht bis dahin für andere Aufgaben zur Verfügung. Dies wird durch Selektion der Checkbox „DelayKit" im Batch-Block erreicht.

5.1.5.1 Eingänge:

MU_IN: Eingang für MU, typischerweise das zu bearbeitende Objekt.
Res_IN: Eingang für MU Ressource, typischerweise die Ressource, welche zur Bearbeitung benötigt wird.

5.1.5.2 Ausgänge:

MU_OUT: Ausgang für MU, typischerweise das bearbeitete Objekt.
Res_OUT: Ausgang für MU Ressource, typischerweise die Ressource, welche zur Bearbeitung benötigt wurde.
Effort_Out: der kumulierte Aufwand.

5.1.6 Benötigte und produzierte Daten

In diesem Modul können als Input-Daten die Daten, welche in den Attributen der MUs gespeichert sind, sowie aus der Datenbank benötigten Daten zur Berechnung der Zeit der Aktivität verwendet werden.
Output-Daten sind die Informationen, welche in die Datenbank geschrieben werden. Dazu zählen zum Beispiel alle Informationen über die Zeit (Dauer, Start- und Endzeit) sowie die IDs der MUs.

MU-Attribute:

Zu bearbeitendes Objekt:

Attribut	Beispiel	Kommentar
ID der MU	Unit_ID	zur Identifikation der MU
Größe der MU	Size	Hier kann wahlweise die Gesamtgröße oder geänderte Größe angegeben sein.
Komplexität	Complexity	Ein weiteres Maß, welches die Dauer der Aktivität beeinflusst

Ressource MU:

Attribut	Beispiel	Kommentar
ID der MU	Personen_ID	Um die Ressource zuordnen zu können.
Skill	Coding_Skill	Fähigkeit im Ausüben der Tätigkeit, welche die Aktivität modelliert.

Input-Daten aus Datenbank:

Zur Berechnung der Dauer der Aktivität können bis zu fünf Werte in dem Equation Block von Extend verwendet werden. Diese können zum Teil aus der Datenbank gelesen werden und sind üblicherweise Verteilungen oder Parameter, welche zur Berechnung der Bearbeitungszeit benötigt werden. Als weitere Parameter können die Attribute der MUs verwendet werden.

Output-Daten in Datenbank

In der Datenbank wird eine Tabelle mit allen Informationen beschrieben, welche in diesem Modul anfallen. Üblicherweise hat die Tabelle den Namen der Aktivität, die durch sie modelliert wird. Die Daten, welche in dieser Tabelle abgelegt werden sind üblicherweise die IDs der MUs sowie die Aktivitätsdauern, d.h. die Bearbeitungszeiten des Objekts.

Für jedes MU, das durch dieses Modul geht, wird eine Zeile angelegt. Die Zeile wird über die Information einer ID festgelegt. Dadurch ist es nicht möglich, das gleiche MU (mit der gleichen ID) mehrfach durch das Modul zu schicken.

Hier müsste dann eine Append-Funktion genutzt werden, die das SDI-Interface von Extend aber nicht bietet. Als Workaround kann aber ein Counter diese Funktion übernehmen und eine temporäre ID generieren.

5.1.7 Formeln und Verteilungen

Zur Berechnung der Zeit wird üblicherweise ein Größenmaß des MU, welches bearbeitet wird, herangezogen, sowie eventuell eine Verteilung und ein Skill-Wert der Ressource, sofern vorhanden.

$$\text{Time} = \text{Size} \cdot \text{Random\_Time\_Factor} \cdot \text{Skill}$$

Size und Skill sind Attribute der MUs. Der Random Time Factor ist ein Wert aus der Datenbank und wird durch eine Verteilung realisiert. Das Ergebnis wird in einem Attribut abgelegt und ebenfalls in der Datenbank gespeichert. Die in dem Attribut abgelegte Zeit wird dem Activity Block als Input-Wert am Eingang für die Delay-Zeit zur Verfügung gestellt.

5.1.8 Varianten

Als Variante ist ein Modul denkbar, welches keinen Batch- und Unbatch-Block am Anfang bzw. am Ende besitzt. Vorteil ist, dass man auf die Kombination von zwei Items verzichten kann oder sie individuell gestaltet, d.h. zum Beispiel verschiedene Algorithmen zur Kombination von Modulen verwendet.

5.2 Generic Template Activiy

5.2.1 Name

GT Activity

5.2.2 Typ / Aufgabe

Typ:

Aktivität

Aufgabe:

Verbindet eine MU mit einer zweiten MU (Ressource) z.B. einer Person, um eine Tätigkeit auszuführen. Nach der Ausführung werden beide MUs wieder getrennt. Es werden verschiedene Daten bezüglich der Zeit (Dauer, Anfangs- und Endzeitpunkt) gespeichert. Zusätzlich werden noch weitere Berechnungen ausgeführt.

5.2.3 Domäne

Dieses Modul wurde für die Simulation einer Aktivität in der Software-Entwicklung entwickelt, dort allerdings nicht für eine spezielle Domäne. Ein Einsatz in verschiedenen Domänen aus dem Software-Entwicklungsbereich ist denkbar.

5.2.4 Moving Units (Items)

Es sind zwei Arten von MUs notwendig.

Eine MU stellt das Objekt dar, welches in der Aktivität verändert, bearbeitet oder erstellt wird, das zweite stellt die benötigte Ressource dar, z.B. eine Person oder ein Werkzeug. Es werden beide MUs benötigt, damit das Modul funktioniert.

5.2.5 Funktionsweise des Moduls

Das Modul hat zwei Eingänge für MUs und ebenso zwei Ausgänge für MUs. Ein Eingang sowie Ausgang ist für die MUs bestimmt, die das Objekt, welches bearbeitet, verändert oder erstellt wird, bestimmt, z.B. Testen eines zuvor erstellten Teils eines Software-Produktes.

Der zweite Ein- sowie Ausgang ist für die Ressource bestimmt, welche benötigt wird, um die Aktivität, hier Testen, auszuführen. Da die Ressource als MU mit Attributen ausgeführt ist, bietet diese Art der Benutzung einer Ressource die Möglichkeit, Eigenschaften der Ressource zu modellieren und in die Berechnungen mit einzubeziehen.

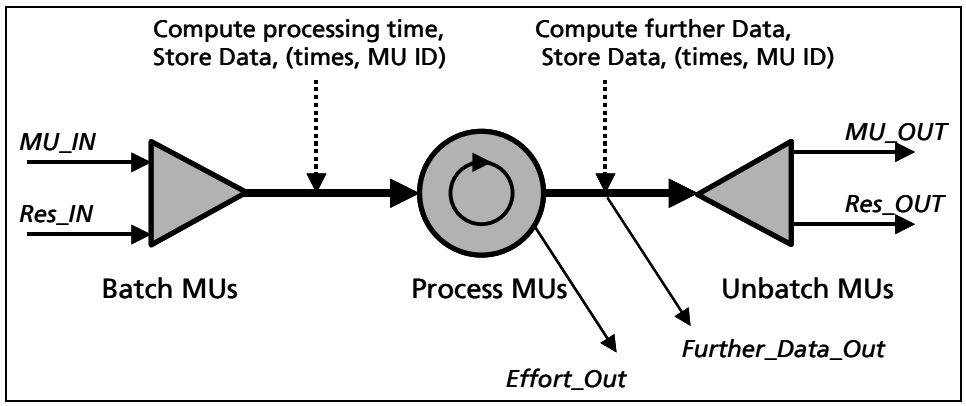


Fig 11: Schematische Darstellung des Flusses der MUs durch Simple Activity

Beide MUs werden am Anfang zu einem Paar zusammengefügt (Batch) und am Ende wieder getrennt (Unbatch). Dadurch ist es möglich, auf die Attribute beider MUs zuzugreifen.

Bei der Auftrennung der MUs am Ende des Moduls behält jedes Modul seine Attribute. Neue Attribute, die während der Vereinigung hinzugefügt wurden, sind danach in keiner der MUs enthalten. Dies wird durch die Selektion von „Preserve Uniqueness" sowohl im Batch- als auch im Unbatch-Block erreicht.

Die MU für die Ressource wird erst in das Modul „gezogen", wenn eine MU am Eingang für das zu bearbeitende Objekt vorhanden ist. Dadurch wird die Ressource erst aus dem Resource Pool entfernt, wenn sie auch wirklich benötigt wird und steht bis dahin für andere Aufgaben zur Verfügung. Dies wird durch Selektion der Checkbox „DelayKit" im Batch-Block erreicht.

5.2.5.1 Eingänge:

MU_IN: Eingang für MU, typischerweise das zu bearbeitende Objekt.
Res_IN: Eingang für MU Ressource, typischerweise die Ressource, welche zur Bearbeitung benötigt wird.

5.2.5.2 Ausgänge:

MU_OUT: Ausgang für MU, typischerweise das bearbeitete Objekt.
Res_OUT: Ausgang für MU Ressource, typischerweise die Ressource, welche zur Bearbeitung benötigt wurde.
Effort_Out: der kumulierte Aufwand.
AkkumulatedValue1_OUT: kumulierter Wert von der ersten berechneten Größe.
AkkumulatedValue2_OUT: kumulierter Wert von der zweiten berechneten Größe.

5.2.6 Benötigte und produzierte Daten

Analog zu den benötigten und produzierten Daten in GT Simple Activity. Zusätzlich kommen hier noch die Daten hinzu, welche für die zusätzlichen Berechnungen aus der Datenbank gelesen werden oder gespeichert werden müssen.

Für genauere Informationen siehe „Benötigte und produzierte Daten" in GT Simple Activity und die Beschreibungen der Beispiele.

5.2.7 Formeln und Verteilungen

Analog zu der Formel in GT Simple Activity zur Berechnung der Dauer der Aktivität kann auch in GT Activity die Dauer der Aktivität berechnet werden. Zusätzlich dazu sind noch 2 weitere Equation Blöcke nach dem Aktivity Block vorhanden, um weitere Berechnungen auszuführen z.B. Anzahl der erzeugten Fehler oder gefundene Fehler.

5.2.8 Varianten

Als Variante ist ein Modul denkbar, welches keinen Batch- und Unbatch-Block am Anfang bzw. am Ende besitzt. Vorteil ist, dass man auf die Kombination von zwei Items verzichten kann oder sie individuell gestaltet, d.h. zum Beispiel verschiedene Algorithmen zur Kombination von Modulen verwenden.

5.3 Generic Template Pre-emptive Activity

5.3.1 Name:

GT Pre-emptive Activity

5.3.2 Typ / Aufgabe

Typ:

Aktivität

Aufgabe:

Zwei bereits verbundene MUs (zu bearbeitendes Objekt und Ressource) verbleiben in einer Aktivität für die Dauer der Bearbeitungszeit. Diese kann auf Anforderung unterbrochen werden, um die Ressource zwischenzeitlich einer anderen Aktivität mit höherer Priorität zuzuführen. Nach der Ausführung bleiben beide MUs zusammen. Gründe sind die Möglichkeit, einzelne Ressourcen während der Aktivität zu unterbrechen und dort fortfahren zu lassen, sowie die Möglichkeit, dass eine qualitätssichernde Aktivität (z.B. Inspektion) im Anschluss erfolgt. Dies kann erfordern, dass beide MUs weiterhin zusammen bleiben. Aus diesem Grund muss das „Batch“ und „Unbatch“ außerhalb des Blocks erfolgen.

Es werden verschiedene Daten bezüglich der Zeit (Dauer, Anfangs- und Endzeitpunkt) gespeichert.

5.3.3 Domäne

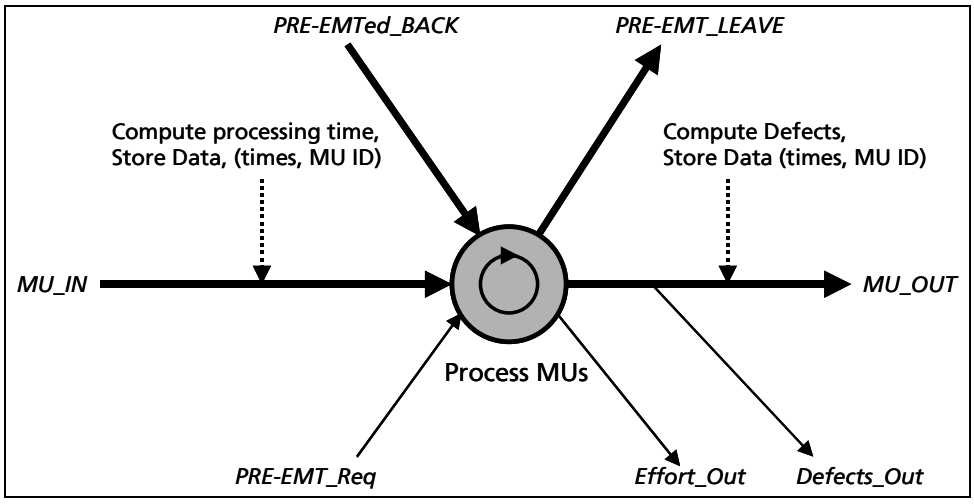
Dieses Modul wurde für die Simulation einer Aktivität in der Software-Entwicklung entwickelt, dort allerdings nicht für eine spezielle Domäne. Ein Einsatz in verschiedenen Domänen aus dem Software-Entwicklungsbereich ist denkbar.

5.3.4 Moving Units (Items)

Es sind zwei Arten von MUs notwendig, die bereits vorher durch einen sog. Batch-Block zu einer MU zusammengefasst wurden.

Eines dieser MUs stellt das Objekt dar, welches in der Aktivität verändert, bearbeitet oder erstellt wird, das zweite stellt die benötigte Ressource dar, z.B. eine Person oder ein Werkzeug. Prinzipiell kann auch ein MU verwendet werden, dass alle Informationen besitzt, oder wenn Informationen über die Ressource nicht notwendig sind.

5.3.5 Funktionsweise des Moduls



Die MUs kommen über MU_IN in das Modul und die Bearbeitungszeit, z.B. die Dauer des Kodierens, wird berechnet. Nach Ablauf der Zeit verlassen die MUs das Modul wieder durch MU_OUT. Dabei wird der Gesamtaufwand errechnet und über Effort_Out nach Außen gegeben. Eine weitere Berechnung, z.B. die Anzahl der Fehler im Code, wird vor dem Verlassen des Moduls durchgeführt und sowohl als Attribut in dem MU gespeichert als auch über Value_Out nach Außen geliefert. Über den Eingang PRE-EMT_Req kommen die Anforderungen nach einer Ressource für eine Tätigkeit mit höherer Priorität z.B. für eine Inspek-

tion, dann verlässt ein MU die Aktivität und speichert intern die noch aufzuwendende Zeit (Attribut z.B. Remaing_Time). Nach Beendigung der Unterbrechung kommt die MU über den Eingang PRE-EMT_Back wieder in die Aktivität zurück, um die Aktivität zu beenden. Eine solche Unterbrechung kann während einer Bearbeitung mehrfach erfolgen.

5.3.5.1 Eingänge:

MU_IN: Eingang für MU, typischerweise das zu bearbeitende Objekt plus Ressource
PRE-EMT_Back: Eingang MU, typischerweise die Ressource, welche zur Bearbeitung einer anderen Aktivität benötigt wurde, wird hier zurück geführt , um die zuvor begonnene Aktivität zu beenden.
PRE-EMT_Req: Eingang, um eine Anforderung nach einer Ressource anzuzeigen. Bedingung: der Wert muss größer 0,5 sein.

5.3.5.2 Ausgänge:

MU_OUT: Ausgang für MU, typischerweise das bearbeitete Objekt plus die Ressource
PRE-EMT_Leave: Ausgang für MU Ressource, typischerweise die Ressource. welche zur Bearbeitung einer anderen Aktivität benötigt wird
Effort_Out: der kumulierte Aufwand.
Value_Out: hier wird für jede MU, die das Modul verlässt die Anzahl der Defects ausgegeben.

5.3.6 Benötigte und produzierte Daten

In diesem Modul werden als Input-Daten die Daten, welche in den Attributen der MUs gespeichert sind, sowie die aus der Datenbank benötigten Daten zur Berechnung der Zeit für die Aktivität verwendet.
Output-Daten sind die Informationen, welche in die Datenbank geschrieben werden. Dazu zählen alle Informationen über die Zeit (Dauer, Start- und Endzeit) sowie die IDs der MUs.

MU-Attribute:

Zu bearbeitendes Objekt:

Attribut	Beispiel	Kommentar
ID der MU	Unit_ID	zur Identifikation der MU
Größe der MU	Size	Hier kann wahlweise die Gesamtgröße oder geänderte Größe angegeben sein.
Komplexität	Complexity	

Ressource MU:

Attribut	Beispiel	Kommentar
ID der MU	Person_ID	um die Ressource zuordnen zu können.
Skill	Coding Productivity Skill	

Neue Attribute für kombinierte MU:
Durch Berechnungen werden neue Attribute hinzugefügt, welche später wieder benötigt werden. Es ist einfacher, sie als Attribute zu speichern als in der Datenbank.

Attribut	Beispiel	Kommentar
Activity time	Coding_Time	Gesamtzeit für die Bearbeitung dieses Objekts, wird auch in der DB gespeichert
Remaining Time	Rem_Time	Verbleibende, noch aufzuwendende Zeit zur Fertigstellung
Defects	Defects	Anzahl der eingefügten Fehler
Autor von Modul N	Author_Of	ID des Moduls, welches der Developer gerade bearbeitet

Input-Daten aus Datenbank:

Zur Berechnung der Dauer der Aktivität können Daten aus der Datenbank gelesen werden. Bei Bedarf können noch weitere Informationen in die Equation Blöcke eingegeben werden.

Die Datenbank enthält Verteilungen und Kalibrierungswerte, welche zur Berechnung verwendet werden.

Beispiele:

Name	Abk.	Typ
Random Coding Time Factor	RCTF	Verteilung
Random Defect Density	RDD	Verteilung
Min Defect Density	MDD	Defects per LOC

Output-Daten in Datenbank

In der Datenbank wird eine Tabelle mit allen Informationen beschrieben, welche in diesem Modul anfallen.

Beispiel:

Unit Number	Developer	Coding Time	Start Time	End Time	Defects

Für jede MU, die durch dieses Modul läuft, wird eine Zeile angelegt. Die Zeile wird über die Information in „Unit Number“ festgelegt. Dadurch ist es nicht möglich, die gleiche MU (mit der gleichen Unit Number) mehrfach durch das Modul zu schicken. Hier müsste dann eine Append-Funktion genutzt werden, die das SDI-Interface von Extend aber nicht bietet.

5.3.7 Formeln und Verteilungen

In jedem der beiden Equation Blöcke können Verteilungen und Parameter so wie Attributwerte zur Berechung verwendet werden. Der Equation Block vor dem Activity Block muss für die Berechung der Bearbeitungszeit verwendet werden.

5.3.8 Varianten

5.4 Generic Template Set Skill

5.4.1 Name

GT Set Skill

5.4.2 Typ / Aufgabe

Typ:

Ressourcen-Management

Aufgabe:

Ein Skill-Wert der involvierten Person (Ressource) wird verändert abhängig von der Zeit, die die Person mit der Aktivität zugebracht hat, für die dieser Skill benötigt wird.

5.4.3 Domäne

Dieses Modul wurde für die Verwendung in der Software-Entwicklung entwickelt, dort allerdings nicht für eine spezielle Domäne. Ein Einsatz in verschiedenen Domänen aus dem Software-Entwicklungsbereich ist denkbar.

5.4.4 Moving Units (Items)

Es ist ein MU notwendig, das Informationen zur Identifikation der Peron enthalten muss sowie die Dauer der ausgeführten Tätigkeit sowie den aktuellen Skill-Wert.

5.4.5 Funktionsweise des Moduls

Die MUs werden linear durch das Modul geführt in Modellzeit t=0, d.h. es wird keine Zeit dafür benötigt, den Skill zu aktualisieren.

5.4.5.1 Eingänge:

MU_IN: Eingang für MU.

5.4.5.2 Ausgänge:

MU_OUT: Ausgang für MU.

5.4.6 Benötigte und produzierte Daten

In diesem Modul werden als Input-Daten die Daten, welche in den Attributen der MUs gespeichert sind, sowie die aus der Datenbank benötigten Daten zur Berechnung des neuen Skill-Werte verwendet.
Output ist der aktualisierte Skill-Wert, der sowohl als Attribut in der MU als auch in der Datenbank abgelegt wird.

MU-Attribute:

Bearbeitetes Objekt:

Attribut	Name im Modell	Kommentar
ID der bearbeiteten MU	Unit_ID	zur Identifikation der MU
Zeit der Bearbeitung	z.B. Coding Time, Design Time,	Falls in der MU als Attribut vorhanden, ansonsten auch aus DB

Ressource MU:

Attribut	Name im Modell	Kommentar
ID der MU	Person_ID	um die Ressource zuordnen zu können.
Skill	***AB***S ***A:*** Coding, Design, Test, Preparation, ***B:*** Quality, Productivity	die verschiedenen Skills

Input-Daten aus Datenbank:

Zur Berechnung des neuen Skill-Werts wird aus der Datenbank der Learning Factor für diesen Skill gelesen. Bei Bedarf können noch weitere Informationen in die Equation Blöcke eingegeben werden.

Bezeichnung	Abk.	Einheit
Learning Factor for Skill n	***AB***LF ***A:*** Coding, Design, Test, Preparation, ***B:*** Quality, Productivity	--

Output-Daten in Datenbank

Neben der Aktualisierung des Attributs des zugehörigen Skills wird auch der Skill in der Datenbank aktualisiert.

Beispiel einer Skill-Tabelle:

Person ID	Skill 1	Skill 2	Skill 3	Skill n

Es können beliebig viele Skills und beliebig viele Personen verwaltet werden.

5.4.7 Formeln und Verteilungen

Es wird keine Verteilung verwendet.

Formel:

$$New\_Skill = \frac{1}{\left(1 + \left(\frac{1}{SKILL - 1}\right) \cdot exp\left(-LearningFact \cdot DeltaTime\right)\right)}$$

Daraus ergibt sich folgender Verlauf des Skill über die Zeit.

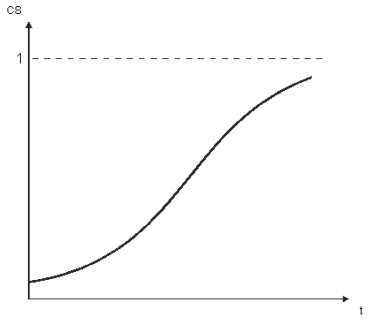


Fig 12: Verlauf der Skill-Kurve

6 Simulationsexperimente mit dem Model

Im folgenden sollen drei unterschiedliche experimentelle Untersuchungen mit dem Model beschrieben werden.

- Variation der Anzahl der Inspektoren
- Selektion der Module zur Inspektion bei einer festen Anzahl von Inspektoren.
- Einfluss der Veränderung von Skills während des Verlaufs der Simulation.

Zur Dokumentation der Experimente werden zuerst Aufbau des Versuchs und die Einstellungen im Model beschrieben. Besonderheiten bei der Durchführung der Simulation sollen dann im folgenden erläutert werden. Anschließend folgt die Darstellung der Ergebnisse und eine Interpretation mit einer Betrachtung der Gültigkeit der Ergebnisse für die Praxis.

6.1 Variation der Anzahl der Inspektoren

6.1.1 Aufbau des Versuchs

Mit dem oben beschriebenen Modell wurden bei konstanten Eingangswerten für die Module (Größe, Komplexität) sowie der Verteilungen (in Tabelle 3: General Distributions) und anderen Parametern die Anzahl der Inspektoren variiert. Auch die Veränderung der Skills wurde deaktiviert.

Um einen Vergleich mit einem Prozess ohne Inspektion zu bekommen wurde auch ein Simulationslauf ohne Inspektion ausgeführt. Die Ergebnisse der Simulationsläufe wurden in ein Excel Sheet eingefügt und dort graphisch dargestellt.

6.1.2 Einstellungen des Modells

Für die Einstellung des Modells sind im wesentlichen drei Tabellen zuständig. Die Daten in den Tabellen Tabelle 2: Unit Input Table und Tabelle 3: General Distributions müssen über eine Versuchsreihe konstant gehalten werden sind aber für das Verhalten des Modells im Versuch von geringerer Bedeutung.

In der Tabelle 4: Allgemeine Input-Daten (General Input Data) sind alle für den Versuch relevanten Daten abgelegt. Folgende Einstellungen sind für den Versuch vorgenommen worden.

Name	Abk.	Wert	Einheit	Kommentar
Max Design Productivity	MDP	0		
Max Coding Productivity	MCP	12	LOC	
Max Preparation Productivity	MPP	200	LOC	
Average Meeting Productivity	AMP	200	LOC	
Defect Find Rate	DFR	0.75	def/h	Defect find rate
Design Quality	DQ	0		
Min Defect Density	MDD	0.01	defects/LOC	
Average Defects Size	ADS	20	LOC	
Max Test Defect Density	MTDD	1	def/KLOC	Defects per 100 Loc
Coding Quality Learning Factor	CQLF	0		
Coding Productivity Learning Factor	CPLF	0		
Preparation Productivity Learning Factor	PPLF	0		
Defects Detection Learning Factor	DDLF	0		
Rework Defects Factor	RDF	0.1		
domain skill weight	DSW	0		
Person Cost	PC	0		
Test_Thoroughness	TT	0.75		defects found factor
Defect_density_Threshold	DDTsh	1	Defects/LOC	for deciding if the modul will be inspected
Size_threshold	STsh	200	LOC	for deciding if the modul will be inspected
Complexity_Threshold	CmplxTsh	0.75		for deciding if the modul will be inspected
Percent_Treshold	Perc_Tsh	1	%	for deciding if the modul will be inspected
Number of Inspectors	No_Insp	10		

Selection policy	Insp_Pol	1		0:keine insp. 1: Percent ; 2: Compl ; 3: Size ; 4 defect density
Number_of_units	#Units	100		Wird vom Model mit dem im Model eingestellten Wert beschrieben
Number_of_Developers	#Devlo-pers	20		Wird vom Model mit dem im Model eingestellten Wert beschrieben

Tabelle 15: Einstellwerte für den Versuch

Für den Versuch wurde der Wert *Number of Inspectors* von 1 bis 10 variiert. Um den Fall 0 Inspektoren, also keine Inspektion, zu realisieren wurde der Wert *Selection Policy* auf 0 gesetzt. Damit alle Codemodule inspiziert werden wir die *Selection Policy* auf 1 (Percent) gesetzt und der *Percent_Threshold* auf 1 gesetzt.

6.1.3 Durchführung der Simulation

Die Simulation wurde jeweils 10 mal für jede Einstellung durchgeführt und in ein Excel Sheet kopiert. Aus den 10 Werten wird dann der Mittelwert gebildet um den Einfluss der stochastischen Effekte zu kompensieren.

6.1.4 Ergebnisse

Verlauf der Defects bei Erhöhung der Anzahl der Inspektoren

#Inspectors	Initial Def	Found Def insp	Def after Insp	Def Found Test	Final Def
0	1437	0	1437	1041	396
1	1432	595	892	629	263
2	1436	931	602	413	189
3	1437	1106	446	296	150
4	1437.8	1204.8	358.1	226.4	131.7
5	1430.2	1251.7	308.6	195.2	113.4
6	1434	1281	287	172	115
7	1435.8	1302.8	274.3	162	112.3
8	1432	1313	256	148	108
9	1433.2	1320.6	254.2	146	108.2
10	1436.7	1316.1	259.3	153	106.3

Tabelle 16 : Ergebnis von je zehn Simulationsläufen mit einer unterschiedlichen Anzahl von Inspektoren.

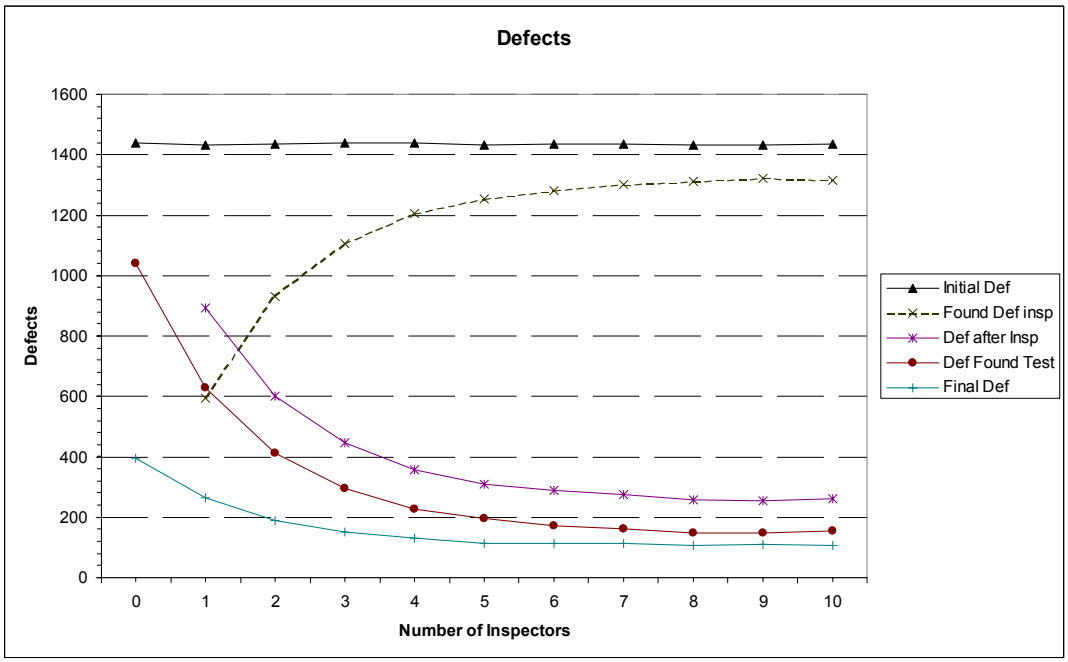


Fig 13 Verlauf der gefundenen und verbleibenden Fehler abhängig von der Anzahl der Inspektoren.

Verlauf der Aufwände gesamt und für die einzelnen Aktivitäten

#Inspectors	overall Eff	Duration	design	coding	inspection	test	rework
0	33429.122	2356.176	1816.057	12509.39		5312.3	13791.39
1	31370.30	2229.29	1816.06	12509.39	5441.65	3297.53	8305.66
2	30445.86	2309.17	1816.06	12509.39	8283.75	2225.46	5611.20
3	30176.29	2339.47	1816.06	12509.39	10206.89	1649.70	3994.26
4	30420.38	2388.76	1816.06	12509.39	11693.73	1326.85	3074.35
5	31031.17	2414.68	1816.06	12509.39	12975.26	1148.12	2582.34
6	31533.26	2340.10	1816.06	12509.39	13859.83	1067.75	2280.24
7	32254.79	2384.48	1816.06	12509.39	14746.28	1021.31	2161.75
8	32934.92	2348.27	1816.06	12509.39	15689.19	954.04	1966.24
9	33294.34	2758.46	1816.06	12509.39	16115.46	946.50	1906.93
10	34375.36	2804.82	1816.06	12509.39	17008.48	967.51	2073.92

Tabelle 17: Ergebnis von je zehn Simulationsläufen mit einer unterschiedlichen Anzahl von Inspektoren.

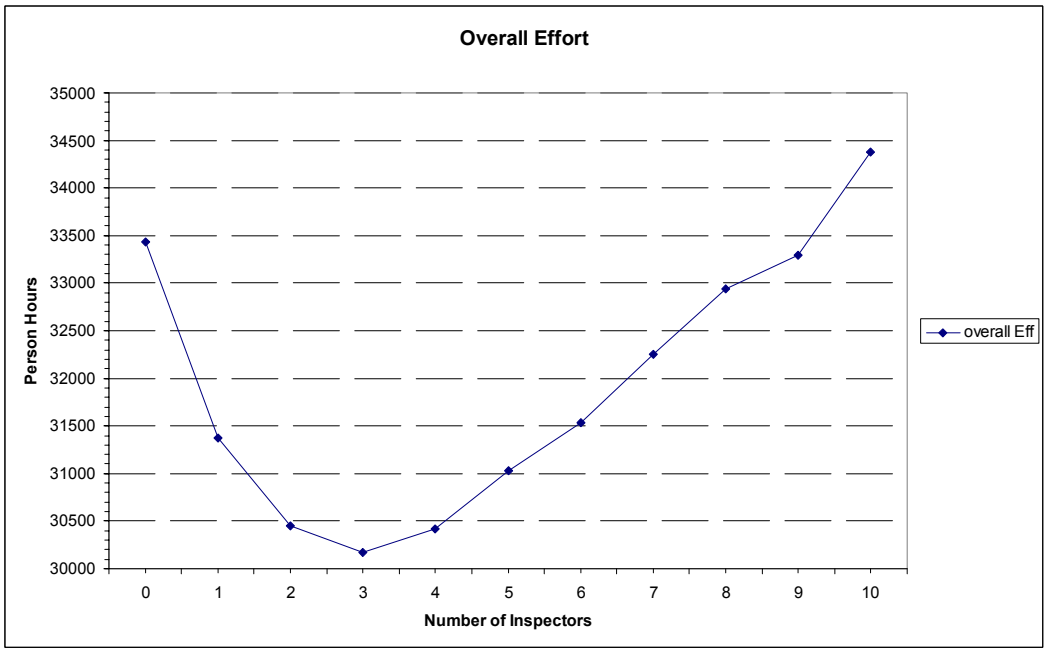


Fig 14: Verlauf des Gesamtaufwandes abhängig von der Zahl der Inspektoren

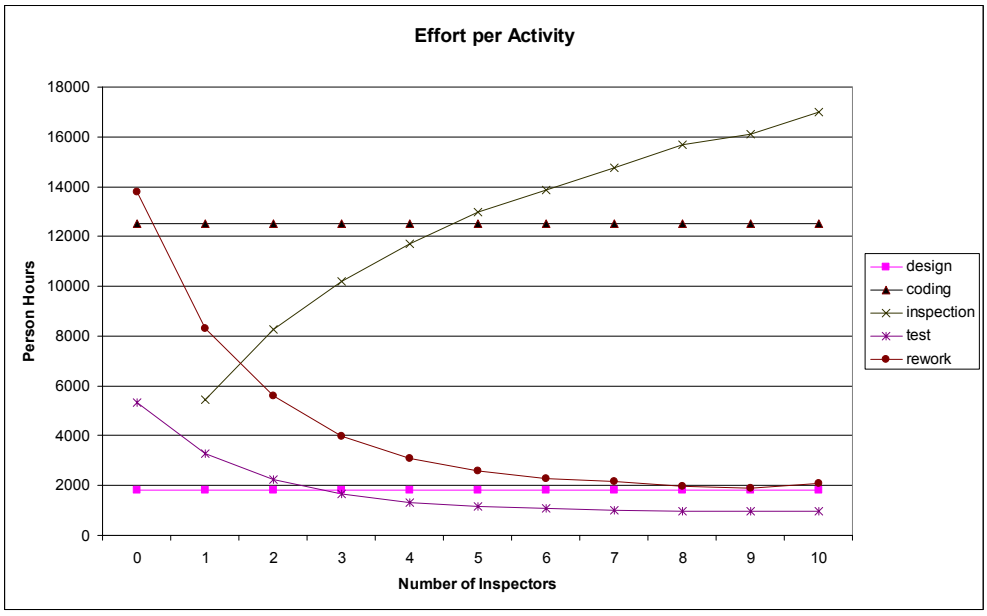


Fig 15: Verlauf der Aufwände für die einzelnen Aktivitäten abhängig von der Zahl der Inspektoren

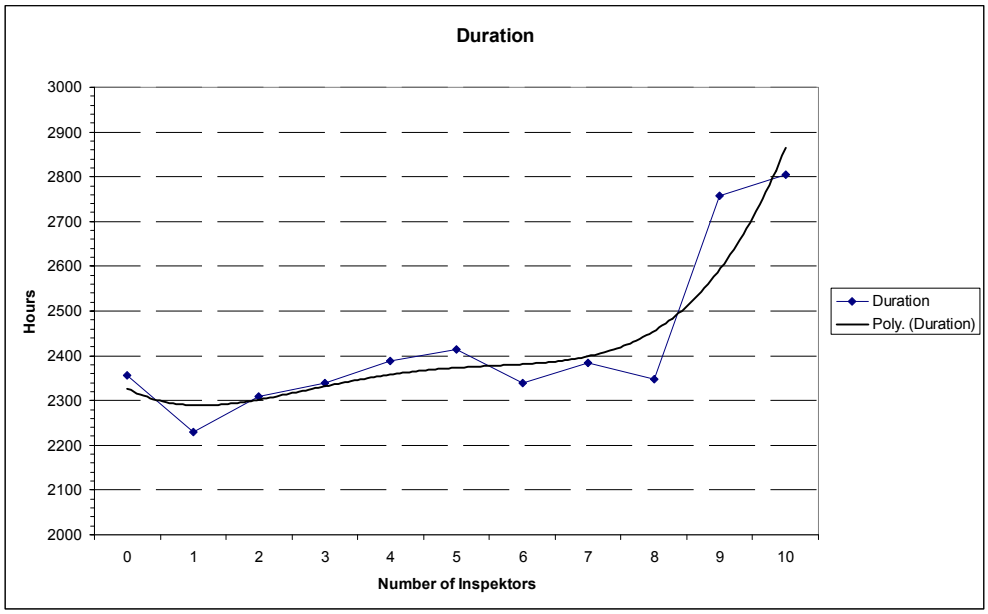


Fig 16: Verlauf der Gesamtdauer des Projektes

6.1.5 Interpretation

Die Anzahl der Fehler im Endprodukt sinkt mit steigender Anzahl von Inspektoren bei gleicher Anzahl von initialen Fehlern nach der Kodierungs-Aktivität. Betrachtet man die Anzahl der gefundenen Fehlern in der Inspektion, so steigt diese bei mehr als 7 Inspektoren nicht mehr signifikant an. Dies deckt sich mit den Erfahrungen in [=68 - Laitenberger 1999 Quantitative Modelin...=].

Vergleicht man den Verlauf des Gesamtaufwands, so erkennt man, dass sich dieser mit Zunahme der Inspektoren sinkt und bei 3 Inspektoren ein Minimum erreicht. Bei sechs Inspektoren wird der Gesamtaufwand etwas größer als bei einem Inspektor und erst bei zehn Inspektoren wird der Gesamtaufwand überschritten, welcher ohne Inspektion benötigt wird.

Wird die Kombination von gefundenen und verbleibenden Fehlern und des Gesamtaufwandes betrachtet empfiehlt sich eine Anzahl von maximal sieben Inspektoren aus Qualitätssicht und zwei bis fünf aus Sicht des Gesamtaufwandes.

Der Verlauf der Dauer ist durch den impliziten Scheduling-Algorithmus verfälscht. Der First Come First Serve-Algorithmus, nach dem die Resourcen den Codemodulen zugeordnet werden, ermöglicht keine optimale Ausnutzung der Resourcen. Wird eine Trendlinie durch den Chart der Prozess-Dauer gelegt, wird deutlich das die Gesamtdauer erst ab ca. sieben bis acht Inspektoren deutlich ansteigt..

6.2 Selektion der Module zur Inspektion

6.2.1 Aufbau des Versuchs

Mit dem oben beschriebenen Model wurde das Verhalten des Modells untersucht, wenn nur ein Teil der Codemodule inspiziert wird. Als Kriterium wurde die Anzahl der Lines of Code (LOC) herangezogen. Das heißt, Codemodule mit mehr LOC als ein definierter Schwellenwert werden zur Inspektion herangezogen, bei weniger LOC wird das Codemodul ohne Inspektion direkt in den Test weitergeleitet.

6.2.2 Einstellungen des Modells

Für die Einstellung des Modells sind im wesentlichen drei Tabellen zuständig. Die Daten in den Tabellen Tabelle 2: Unit Input Table und Tabelle 3: General Distributions müssen über eine Versuchsreihe konstant gehalten werden sind aber für das Verhalten des Modells im Versuch von geringerer Bedeutung.

In der Tabelle 4: Allgemeine Input-Daten (General Input Data) sind alle für den Versuch relevanten Daten abgelegt. Folgende Einstellungen sind für den Versuch vorgenommen worden.

Max Design Productivity	MDP	0		
Max Coding Productivity	MCP	12	LOC	
Max Preparation Productivity	MPP	200	LOC	
Average Meeting Productivity	AMP	200	LOC	
Defect Find Rate	DFR	0.75	def/h	Defect find rate
Design Quality	DQ	0		
Min Defect Density	MDD	0.01	defects/LOC	
Average Defects Size	ADS	20	LOC	
Max Test Defect Density	MTDD	1	def/KLOC	Defects per 100 Loc
Coding Quality Learning Factor	CQLF	0		
Coding Productivity Learning Factor	CPLF	0		
Preparation Productivity Learning Factor	PPLF	0		
Defects Detection Learning Factor	DDLF	0		
Rework Defects Factor	RDF	0.1		
domain skill weight	DSW	0		
Person Cost	PC	0		

Test_Thoroughness	TT	0.75		% of defects found
Defect_density_Threshold	DDTsh	1	Defects/LOC	for deciding if the modul will be inspected
Size_threshold	STsh	600	LOC	for deciding if the modul will be inspected
Complexity_Threshold	CmplxTsh	0.75		for deciding if the modul will be inspected
Percent_Treshold	Perc_Tsh	1	%	for deciding if the modul will be inspected
Number of Inspectors	No_Insp	4		
Selection policy	Insp_Pol	1		0:keine insp. 1: Percent ; 2: Compl ; 3: Size ; 4 defect density
Number_of_units	#Units	100		Wird vom Model mit dem im Model eingestellten Wert beschrieben
Number_of_Developers	#Devlopers	20		Wird vom Model mit dem im Model eingestellten Wert beschrieben

Tabelle 18: Einstellwerte für den Versuch

6.2.3 Durchführung des Versuchs

Die Simulation wurde jeweils 10 mal für jede Einstellung durchgeführt und in ein Excel Sheet kopiert. Aus den 10 Werten wird dann der Mittelwert gebildet, um den Einfluss der stochastischen Effekte zu kompensieren.

6.2.4 Ergebnisse

Verlauf der Defects bei verschiedenen Schwellenwerten für das Selektionskriterium.

LOC	Initial Def	Found Def insp	Def after Insp	Def Found Test	Final Def
alle	1437.8	1204.8	358.1	226.4	131.7
>100	1435.2	1180.7	358.5	240	137.5
>200	1436.6	892.2	275.6	444.2	194.9
>300	1438	503	155	704	282
>400	1437	245	85	874	343
>500	1437	139	46	946	366
>600	1438	85	28	985	376
keine	1437			1041	396

Tabelle 19: Ergebnis von je zehn Simulationen mit unterschiedlichem Selektions-Kriterium

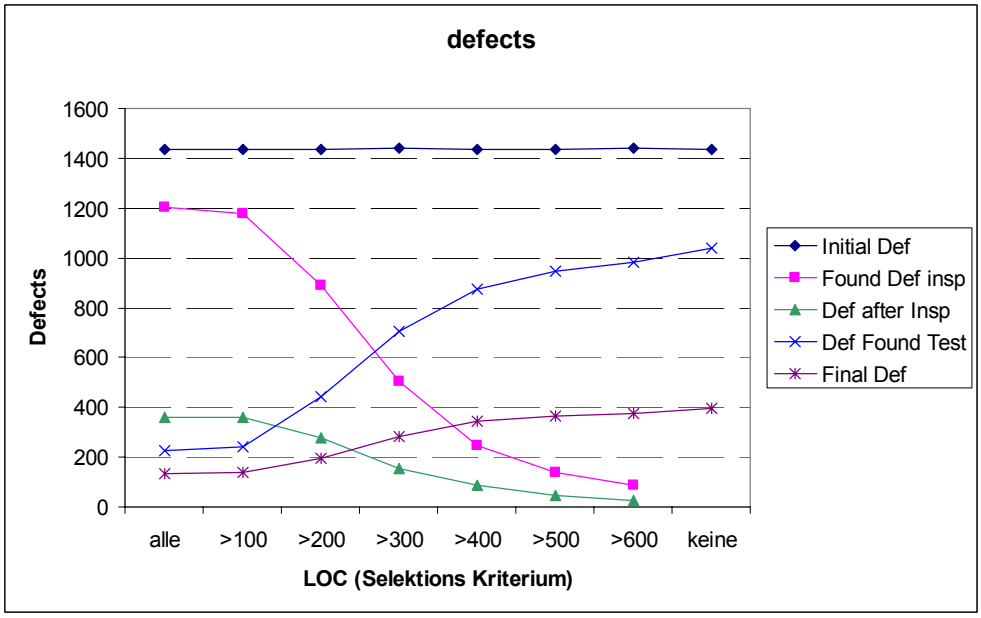


Fig 17: Verlauf der Defects bei der Auswahl der zu inspizierenden Module abhängig von deren Größe.

Verlauf der Aufwände und Dauer abhängig von der Auswahl der inspizierten Module.

LOC	overall Eff	Duration	design	coding	inspection	test	rework
alle	30420	2389	1816	12509	11694	1327	3074
>100	30403	2220	1816	12509	11457	1398	3223
>200	31038	2116	1816	12509	8616	2363	5734
>300	32761	2966	1816	12509	4953	3645	9838
>400	32748	2359	1816	12509	2135	4499	11789
>500	32965	2280	1816	12509	1415	4850	12374
>600	33127	2328	1816	12509	881	5031	12889
keine	33429	2356	1816	12509		5312	13791

Tabelle 20: Verlauf der Aufwände bei der Auswahl der zu inspizierenden Module abhängig von deren Größe

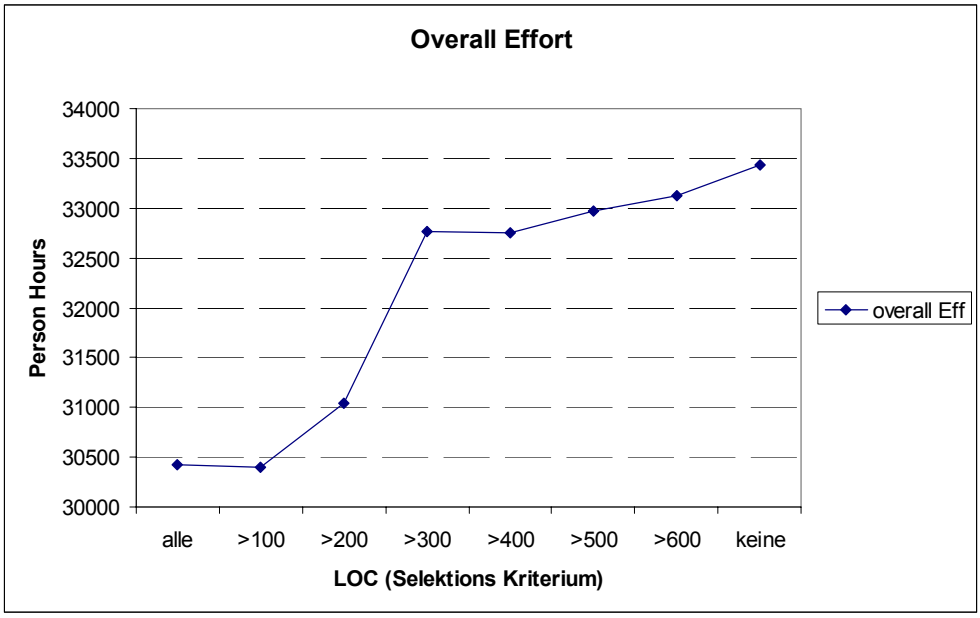


Fig 18: Overall Effort bei der Auswahl der zu inspizierenden Module

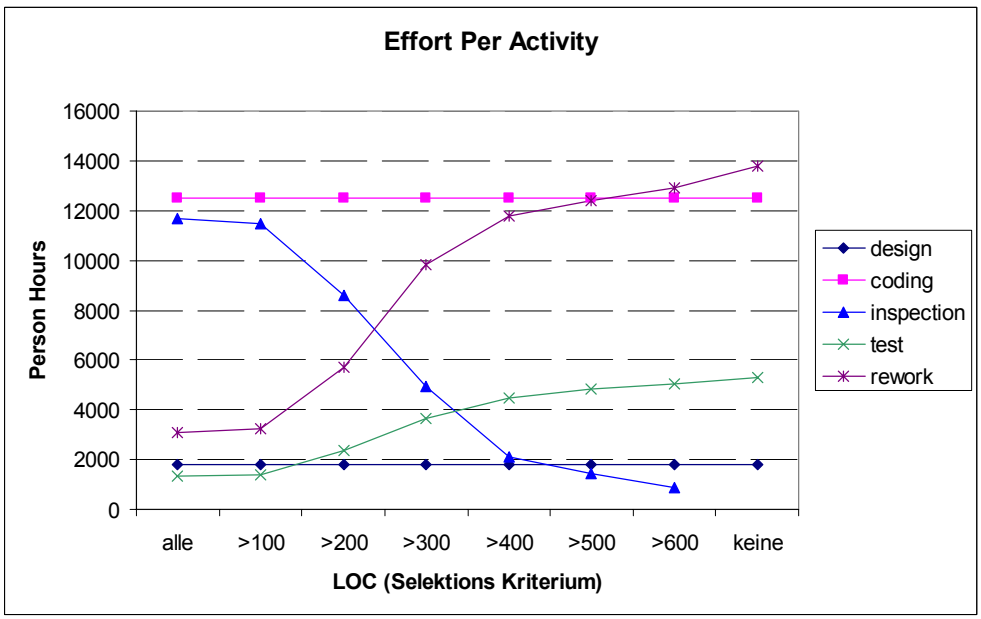


Fig 19: Aufwand pro Aktivität bei der Auswahl der zu inspizierenden Module

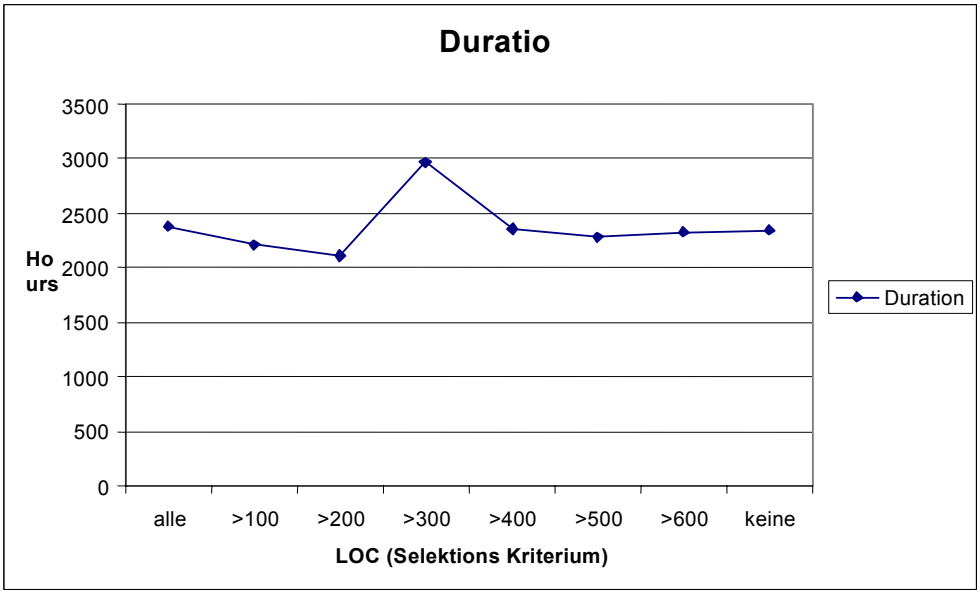


Fig 20: Gesamtdauer in Stunden

6.2.5 Interpretation

Die Anzahl der Fehler im Endprodukt ist bei niedrigen Schwellenwerten (viele Module inspiziert) niedriger als wenn nur die großen Module Inspiziert werden. Die Fehler, die während der Inspektion gefunden werden reduzieren die Anzahl der Fehler vor dem Test und dadurch müssen im Test weniger Fehler identifiziert werden.

Setzt man diesen Verlauf zu dem Aufwand ins Verhältnis ist zu erkennen das die verstärkte Testtätigkeit einen negativen Einfluss auf den Gesamtaufwand hat. Setzt man den Gesamtaufwand bei 100 LOC und 600 LOC ins Verhältnis erhöht sich der Gesamtaufwand um ca. neun Prozent.

Bei der Gesamtdauer (Duration) für einen Simulationslauf zeigt sich ein uneinheitliches Verhalten, welches durch den Scheduling- Algorithmus bestimmt wird. Bei niedrigen Schwellenwerten werden viele Module nspiziert, welches zusätzlichen Aufwand verursacht, der beim Testen wieder eingespart wird. Der hohe Wert für die Duration bei >300 LOC ist ein Ausreißer welcher durch die Stochastik verursacht wurde.

6.3 Variation der Fähigkeiten und der Lernfaktoren

6.3.1 Aufbau des Versuches

Mit dem oben beschriebenen Modell soll der Einfluss auf das Qualitätsmaß Anzahl von Fehlern und den Aufwand für die verschiedenen Aktivitäten in Abhängigkeit von den Skill-Werten und den Lernfaktoren der Entwickler untersucht werden. Dazu werden einerseits die Skills andererseits die Lernfaktoren variiert.

Die Skills nehmen folgende Werte an: 0.3; 0.5 und 0.7. Hierbei werden fünf verschiedene Profile der Skills definiert

- Alle Entwickler besitzen Skill-Werte mit 0.3.
- Alle Entwickler besitzen Skill-Werte mit 0.5.
- Alle Entwickler besitzen Skill-Werte mit 0.7.
- 50% besitzen den Skill-Wert 0.3 und 50% den Skill Wert 0.7.
- 25% besitzen den Skill-Wert 0.3; 50% besitzen den Skill-Wert 0.5 und 25% den Skill Wert 0.7.

Für vier Lernfaktoren werden jeweils auch 3 verschiedene Werte angenommen.

	Szenario 1	Szenario 2	Szenario 3
CPS (code productivity skill)	0	0,0001	0.001
CQS (coding quality skill)	0	0,0001	0.001
DDS (defect detection skill)	0	0,001	0.01
PPS (preparation productivity skill)	0	0,001	0,01

6.3.2 Einstellungen des Modells

Long Name	Abbreviation	Value	Einheit
Max Design Productivity	MDP	0	
Max Coding Productivity	MCP	12	LOC
Max Preparation Productivity	MPP	200	LOC
Average Meeting Productivity	AMP	200	LOC
Defect Find Rate	DFR	0.75	def/h
Design Quality	DQ	0	
Min Defect Density	MDD	0.01	defects/LOC
Average Defects Size	ADS	20	LOC
Max Test Defect Density	MTDD	1	def/KLOC
Coding Quality Learning Factor	CQLF	0.001	
Coding Productivity Learning Factor	CPLF	0.001	
Preparation Productivity Learning Factor	PPLF	0.01	
Defects Detection Learning Factor	DDLF	0.01	

Rework Defects Factor	RDF	0.1	
domain skill weight	DSW	0	
Person Cost	PC	0	
Test_Thoroughness	TT	0.75	
Defect_density_Threshold	DDTsh	1	Defects/LOC
Size_threshold	STsh	200	LOC
Complexity_Threshold	CmplxTsh	0.75	
Percent_Treshold	Perc_Tsh	1	%
Number of Inspectors	No_Insp	4	
Selection policy	Insp_Pol	1	
Number_of_units	#Units	100	
Number_of_Developers	#Devlopers	20	

Tabelle 21 Einstellwerte für die Experimente

6.3.3 Durchführung der Experimente

Es wird für jede Kombination von Skill-Werten und Lernfaktoren je 10 Simulationsläufe ausgeführt. Daraus wird dann jeweils das arithmetische Mittel gebildet und mit den anderen Ergebnissen in diesem Versuch gegenüber gestellt.

6.3.4 Ergebnisse

Fehler bei gleichen Skill-Werten und verschiedenen Lernfaktoren

StartSkill	Lernfaktor Kodierungs und Quality Skill	Lernfaktor Preparation und Defect Detection Skill	Initial Defects	Found Defects in Inspection	Defects after Insp and Rework	Def found in Test	Final Defects
0.3	0	0	2422.9	1601.1	988.7	702.2	286.5
0.3	0.0001	0.001	2292.9	1565	893.6	630	263.6
0.3	0.001	0.01	1656.3	1297.5	492.2	327.7	164.5
0.5	0	0	1434.1	1205.3	353.1	224.8	128.3
0.5	0.0001	0.001	1400.7	1182.5	341.9	218.6	123.3
0.5	0.001	0.01	1189	1042.5	255	151.2	103.8
0.7	0	0	1011.4	918.5	189.7	99.4	90.3
0.7	0.0001	0.001	999.3	909.5	185.5	97.5	88
0.7	0.001	0.01	926.6	850.9	165	83.4	81.6

Tabelle 22 Fehler bei gleichen Skill-Werten und verschiedenen Lernfaktoren

Aufwände bei gleichen Skill-Werten und verschiedenen Lernfaktoren

StartSkill	Lernfaktor Kodierungs und Quality Skill	Lernfaktor Preparation und Defect Detection Skill	Overall Effort	Process Duration	Design Effort	Coding effort	Inspection Effort	Test Effort	Rework Effort
0.3	0	0	68338	4656	1816	20849	23873	6092	15709
0.3	0.0001	0.001	61104	4267	1816	19701	21464	5506	12616
0.3	0.001	0.01	36278	2224	1816	14104	12869	3036	4453
0.5	0	0	30332	2383	1816	12509	11665	1311	3032
0.5	0.0001	0.001	29401	2209	1816	12204	11336	1270	2776
0.5	0.001	0.01	23398	1626	1816	10334	8761	956	1531
0.7	0	0	19589	1454	1816	14104	12869	3036	4453
0.7	0.0001	0.001	19230	1449	1816	8840	7137	511	926
0.7	0.001	0.01	17530	1254	1816	8184	6382	459	689

Tabelle 23 Aufwände bei gleichen Skill-Werten und verschiedenen Lernfaktoren

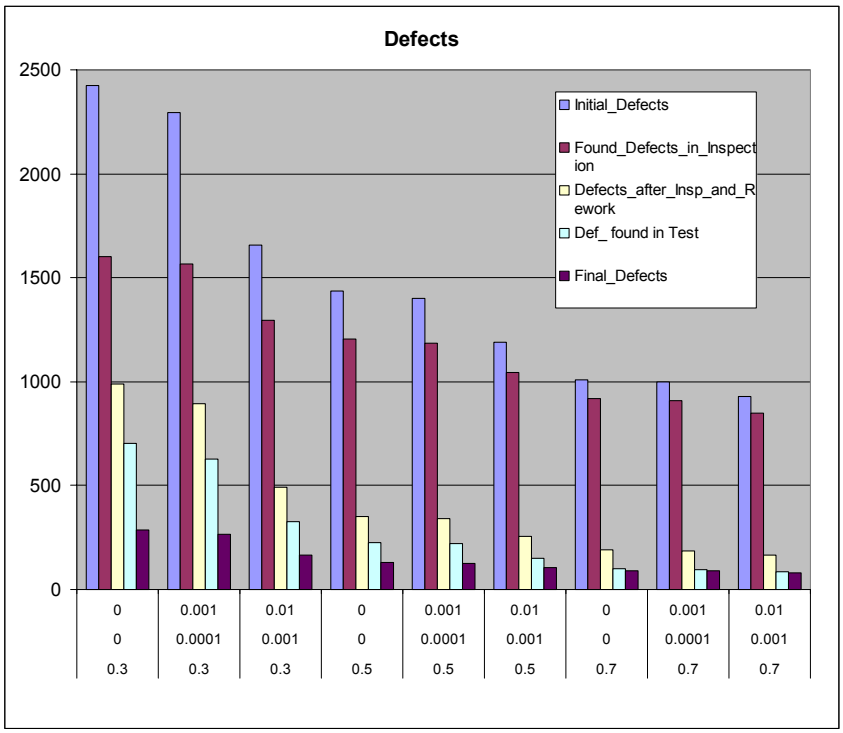


Fig 21 Fehler bei verschiedenen vorgegebenen Skill-Werten und Lernfaktoren

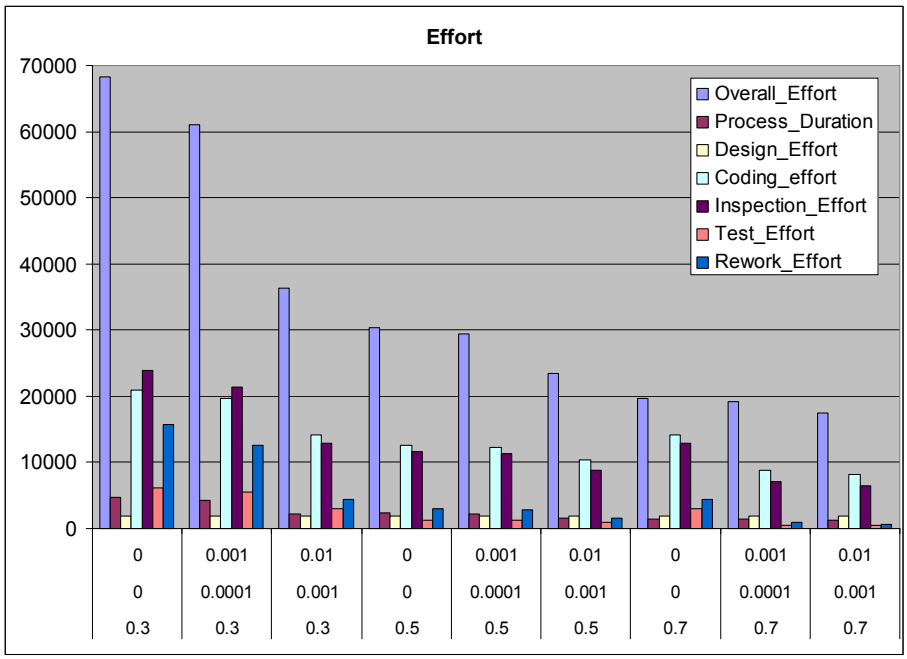


Fig 22 Effort bei verschiedenen vorgegebenen Skill-Werten und Lernfaktoren

Fehler bei unterschiedlichen Werten der Skills am Beginn der Simulation.
50/50 bedeutet 50% der Personen sind mit Skill- Werten von 0,3 und die anderen 50% mit 0,7 initialisiert.
25/50/25 bedeutet 25% 0,3 ; 50% 0,5 und 25% 0,7.

Skill-verteilung	Lernfaktor Kodierungs und Quality Skill	Lernfaktor Preparation und Defect Detection Skill	Initial Defects	Found Defects in Inspection	Defects after Insp and Rework	Def found in Test	Final Defects
50/50	0	0	1453.1	1129.9	436.6	285.5	151.1
50/50	0.0001	0.001	1451.9	1136.2	428.6	280	148.6
50/50	0.001	0.01	1258.1	1039.4	322.5	202.2	120.3
25/50/25	0	0	1420	1149.9	389.8	251	138.8
25/50/25	0.0001	0.001	1445.5	1164.3	404.1	262.7	141.4
25/50/25	0.001	0.01	1214.4	1032.2	290.7	177.8	112.9

Tabelle 24 Fehler bei unterschiedlichen Skill-Werten am Beginn der Simulation

Aufwand bei unterschiedlichen Werten der Skills am Beginn der Simulation.

Skill-verteilung	Lernfaktor Kodierungs und Quality Skill	Lernfaktor Preparation und Defect Detection Skill	Overall_ Effort	Process_ Duration	Design_ Effort	Coding_ effort	Inspection_ Effort	Test_ Effort	Rework_ Effort
50/50	0	0	34801	3152	1816	12630	12548	2236	5571
50/50	0.0001	0.001	34198	3331	1816	12552	12463	2222	5145
50/50	0.001	0.01	26070	2096	1816	10941	9186	1659	2467
25/50/25	0	0	31684	2514	1816	12413	11650	1706	4099
25/50/25	0.0001	0.001	32402	3063	1816	12601	11856	1853	4276
25/50/25	0.001	0.01	24546	1857	1816	10576	8808	1286	2061

Tabelle 25 Aufwand bei unterschiedlichen Skill-Werten am Beginn der Simulation

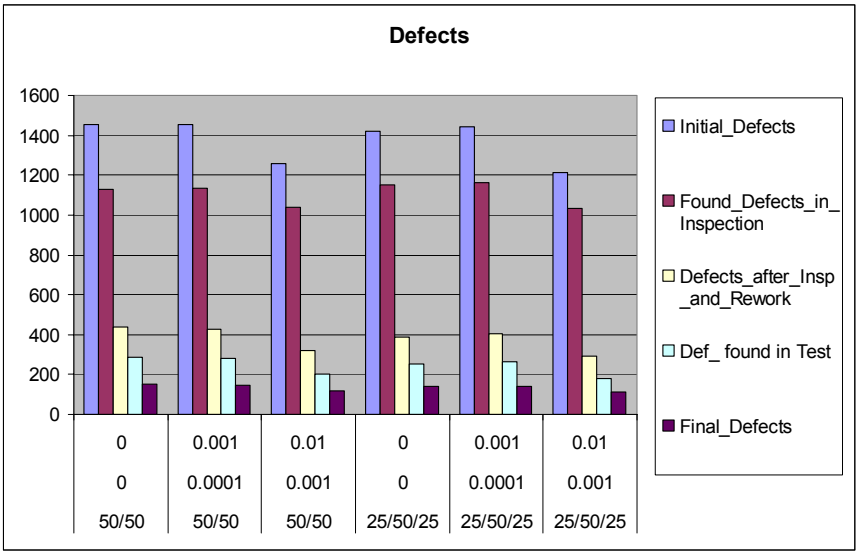


Fig 23 Fehler bei unterschiedlicher Verteilung der Startskills und unterschiedlichen Lernfaktoren

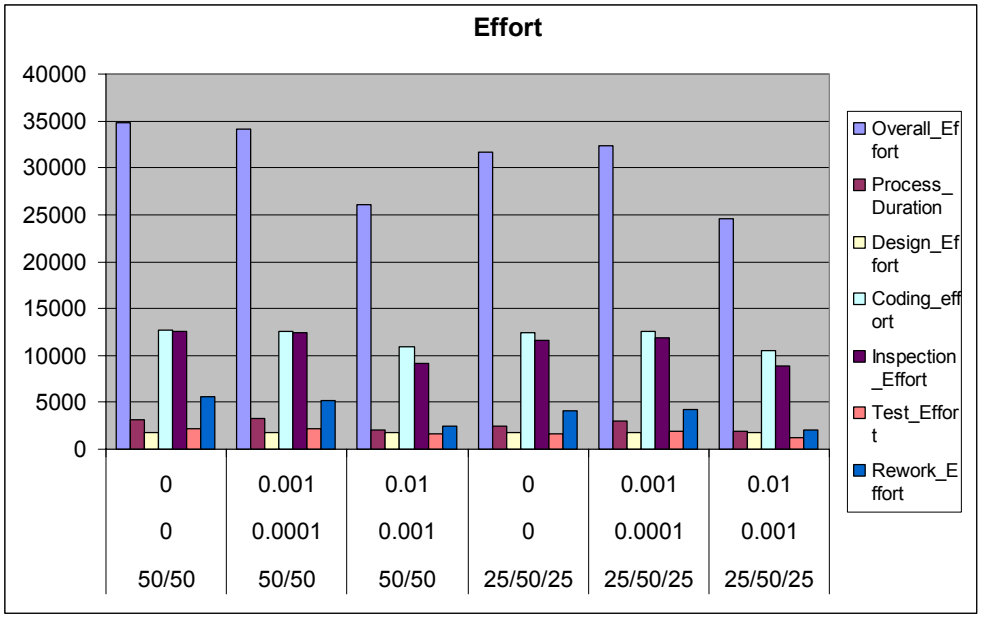


Fig 24 Effort bei unterschiedlicher Verteilung der Startskills und unterschiedlichen Lernfaktoren

6.3.5 Interpretation

In den Grafen Fig 21 bis Fig 24 ist der Einfluss der Skills auf die Anzahl der Fehler und den Aufwand der verschiedenen Aktivitäten abgebildet. Deutlich zu erkennen ist das eine Erhöhung der Skill-Werte am Beginn der Simulation positive Auswirkungen auf die Anzahl der Fehler als auch auf den Aufwand haben. Eine Erhöhung der Lernfaktoren hat einen ähnlichen Effekt auf die Fehler und den Aufwand.

Vergleicht man die Ergebnisse der Simulationen, in denen die Skill-Werte zu Beginn der Simulation unterschiedlich waren, ist die homogenere 25/50/25-Verteilung geringfügig besser als die 50/50-Verteilung. Grund dafür kann die etwas größere Anzahl mittelguter (0,5) und guter (0,7) Entwickler sein, obwohl im Mittel die Skills gleich sind. Hier kann ein optimiertes Scheduling weitere Einsichten liefern als das hier implizit verwendete First Come Ffirst Serve-Verfahren.

Literatur

[AHM91] T. Abdel-Hamid, S. E. Madnick: Software Project Dynamics. Prentice Hall, Englewood Cliffs, NJ, 1991.

[Ste00] Sterman J.D.: Business Dynamics, Systems Thinking and Modelling for a Complex World.. McGraw-Hill, 2000.

[KMR99] M. I. Kellner, R. J. Madachy, D. M. Raffo: Software process simulation modeling: Why? What? How?, In: Journal of Systems and Software, Issues 2-3, S.113-122 (1999).

Dokumenten Information

Titel:	Ein modularisiertes Simulationsmodell zur Entscheidungsunterstützung in Software-Entwicklungsprozessen
Datum:	Dezember 2003
Report:	IESE-098.03/D
Status:	Final
Klassifikation:	Öffentlich