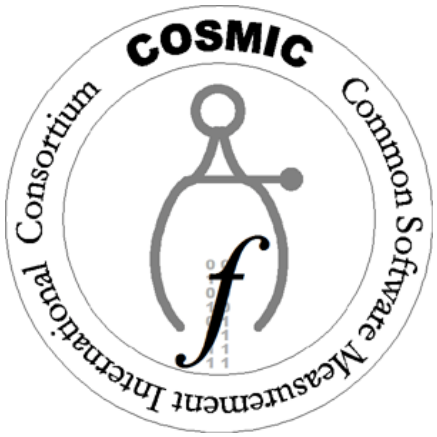
Empirische Grundlagen zur COSMIC-FFP-Anwendung für die Aufwandsschätzung

Martin Kunz, Reiner Dumke

AG Softwaretechnik, {makunz,dumke}@ivs.cs.uni-magdeburg.de, http://www.smlab.de/



Inhalt:

1. Die COSMIC Community

Das **Common Software Measurement International Consortium** (COSMIC) wurde 1998 gegründet und beschäftigt sich vor allem mit der Entwicklung von **Functional Size Measurement (FSM)** Verfahren zur funktionalen Größenbestimmung von Softwaresystemen die dann vor allem auch zur Aufwands- bzw. Kostenschätzung verwendet werden kann. Mitglieder der COSMIC Working Group sind aus Australien, England, Japan, Kanada, Finnland, Spanien, Italien, Niederlanden, Türkei, Belgien, Polen und Deutschland.

http://www.cosmicon.com/

COSMIC erhielt 2006 den **IT Professional Awards von Großbritannien** für die erfolgreiche Erarbeitung und Anwendung eines neuen Standards (ISO 19761) zur Funktionalen Größenmessung.

2. Zu den Grundintentionen der COSMIC FFP-Methode

Die COSMIC-Full Function Point (FFP) Methode hat die folgenden Grundintentionen für den Bereich der Softwareprozessunterstützung (siehe auch [Ebert 2007]):

- wurde für neuartige Software (für eingebettete bzw. **reaktive Systeme**) konzipiert,
- stellt auf einer messtheoretischen Grundlage ein **Maß** dar,
- ist vom Ansatz her ein **White-Box-Estimation.**

3. Die COSMIC-FFP-Methode für die Funktionsumfangsbestimmung

Die Entwicklung der (Full-Function-Point) FFP-Methode lässt sich auf die seit Beginn der 80-er Jahre neu entstandenen Software-Technologien und deren Auswirkungen auf das Functional Size Measurement (FSM) zurückführen [Szalowski 2005]. Die vorergehenden Function-Point-Methoden sind sehr an klassische Informationssysteme angelehnt und lassen sich schlecht für eine Aufwandsschätzung prozessintensiver Software-Systeme anwenden, die weniger auf gespeicherten Daten, denn auf in der Echtzeit-Verarbeitung gemessenen Daten arbeiten. Diese „Real-Time"-Technologien in eingebetteten Systemen, wie z.B. in der Automobiltechnik, lassen sich mit der FFP-Methode besser bzw. geeigneter messen.

Im Folgenden soll die COSMIC-FFP Methode näher erläutert werden. Die Methode, 2003 als ISO/IEC 19761 – Standard anerkannt, ist als Weiterentwicklung der Function-Point-Methode anzusehen und betrachtet die funktionalen Systemanforderungen an ein zu bewertenden Software-Produkt aus Nutzersicht. Die COSMIC-FFP Methode dient als inhaltliche Grundlage des entwickelten Tutorials und wird deshalb sehr detailliert beschrieben.

> *„Die Aufwandsschätzung mit der COCMIC-FFP Methode umfasst die Anwendung einer Menge von Modellen, Regeln und Prozeduren auf ein zu bewertendes bzw. zu messenden Software-Produkt, wie es sich aus Benutzersicht darstellt." (vgl. [COSMIC 03])*

Als Resultat dieser Anwendung von Modellen, Regeln und Prozeduren liefert die Methode einen quantitativen Wert für die Größe bzw. den Umfang des Software-Produktes aus funktionaler Sicht, allgemein als *Functional Size* bezeichnet. Dabei ist die Anwendung der COSMIC-FFP Methode unabhängig von der für die Software-Entwicklung verwendeten Implementationstechniken.

Als Grundlage für die Aufwandsschätzung mit der COSMIC-FFP Methode sind in einem ersten Schritt so genannte **Functional User Requirements (FUR)** zu bestimmen. Diese funktionalen Systemanforderungen sind als Anforderungen aus der Sicht des Anwenders bzw. Nutzers zu verstehen, die allein die Funktionalität einer Software betreffen. Anforderungen technischer Art oder qualitative Anforderungen an eine Software werden durch die „COSMIC-FFP Methode" nicht betrachtet. Aus den mittels Regeln und Prozeduren bestimmten FURs wird in einem als „Mapping Phase" bezeichneten Arbeitsschritt iterativ ein „COSMIC-FFP Generic Software Model" erzeugt, welches ein standardisiertes Modell darstellt, auf welchem die Messung bzw. Zählung der Full-Function-Points durchgeführt werden kann. Die Messung bzw. Zählung wird als „Measuring Phase" bezeichnet. Im Folgenden soll erläutert werden, wie bei der COSMIC-FFP Methode die Bestimmung der FURs erfolgt, daraus das „COSMIC-FFP Generic Software Model" erzeugt wird und die Zählung der Full-Function-Points stattfindet. Die gezählten Full-Function-Points, als Resultat der Methode, geben ein Maß für die funktionale Größe bzw. den funktionalen Umfang der zu bewertenden Software an und können somit für eine Aufwandsschätzung verwendet werden. Die folgende Abbildung zeigt den Prozess der Software-Messung mittels der COSMIC-FFP Methode:

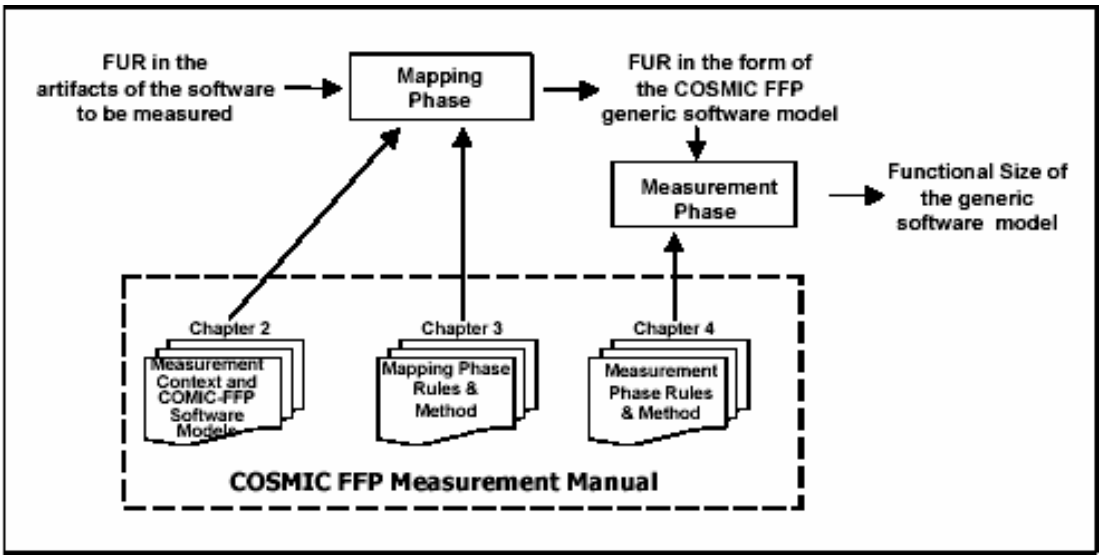


Abb. 1: Softwaremessung mit der COSMIC-FFP Methode (vgl. [COSMIC 03] S.16)

Die funktionalen Systemanforderungen sind als Anforderungen an die Software aus Nutzersicht zu betrachten. Es soll bestimmt werden, welche Funktionalität die zu messende Software dem Nutzer zur Verfügung stellt bzw. welche Funktionalität sie erbringt, um dem Nutzer ein Resultat zu liefern. Für die Bestimmung der FURs an ein Software-Produkt werden so genannte Artefakte der Software-Entwicklung herangezogen. So ergeben sich Benutzeranforderungen an eine Software vor der Entwicklung z.B. aus der formulierten Problemdefinition, wie sie am Anfang des Software-Entwicklungsprozess beschrieben ist. Weiterhin können funktionale Anforderungen aus Artefakten der Daten-Analyse oder aus Modellierungsartefakten abgeleitet werden. Die folgende Abbildung verdeutlicht das Heranziehen von Software-Artefakten zur Bestimmung der FURs:

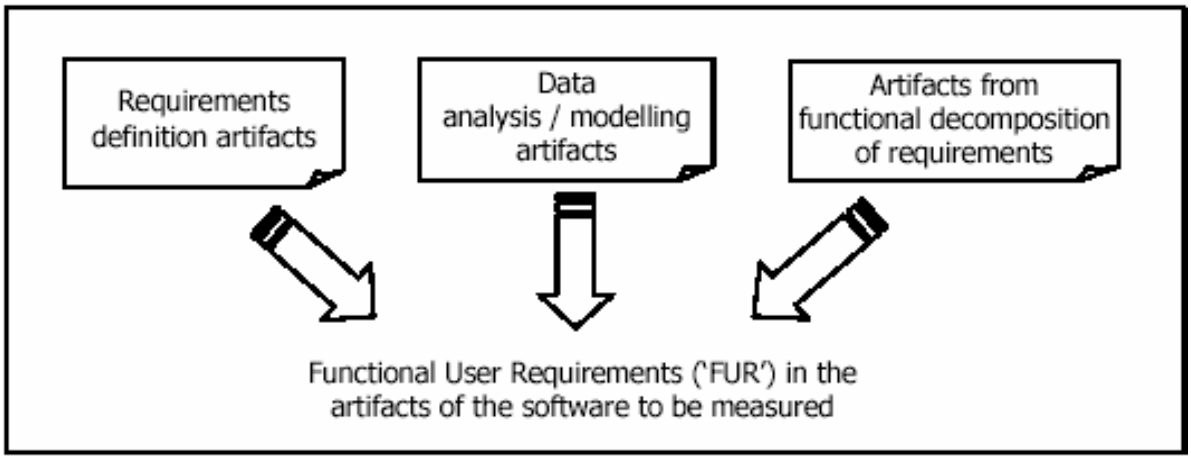


Abb. 2: Entwicklungsartefakte als FURs vor der Entwicklung (vgl. [COSMIC 03] S.17)

Des Weiteren können Benutzeranforderungen für schon existierende Software-Produkte aus anderen Software-Artefakten abgeleitet werden. So kann z.B. aus einem Benutzerhandbuch abgeleitet werden, welche Funktionalitäten eine Software erfüllt. Daneben kann aus einem bestehenden Programm, welches Funktionalitäten der zu bewertenden Software nutzt bzw. ihr zur Verfügung stellt, abgeleitet werden, welche funktionalen Anforderungen von der zu messenden Software zu erfüllen sind. Folgende Abbildung verdeutlicht diesen Prozess:

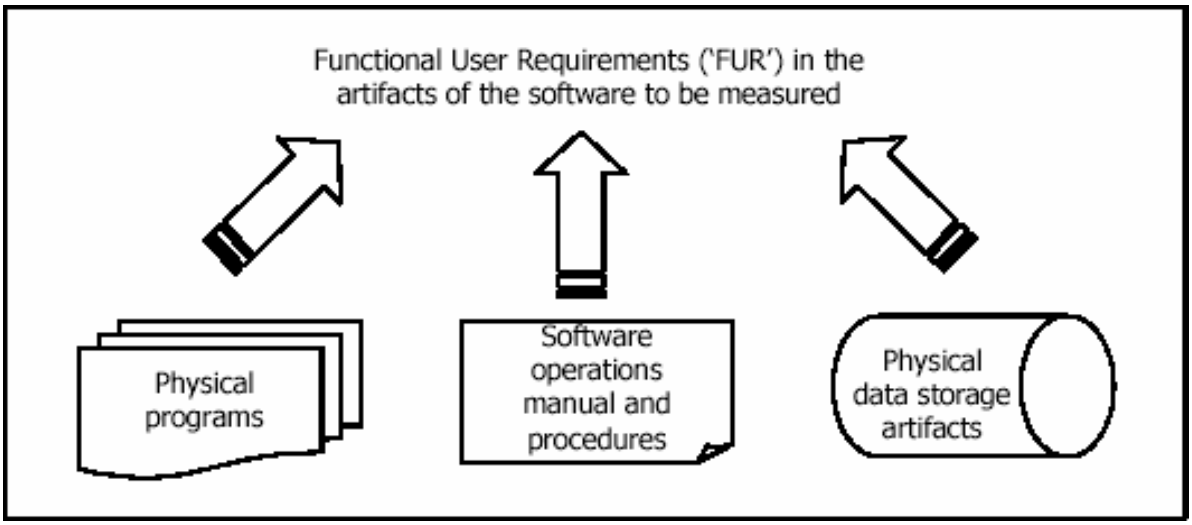


Abb. 3: FURs aus einem existierenden Produkt (vgl. [COSMIC 03] S.17)

Aus den extrahierten FURs wird bei der COSMIC-FFP Methode im Folgeschritt ein so genanntes „COSMIC-FFP Generic Software Model“ erzeugt. Dieses Modell dient als Grundlage einer standardisierten Messung der Full Function Points. Standardisiert meint in diesem Fall, dass die bestimmten FURs auf *funktionale Prozesse* abgebildet werden, die durch Datenbewegungen in die zu bewertende Software hinein oder als Datenbewegungen aus der Software hinaus charakterisiert sind. Die Phase des Modellierens und Erzeugens des „COSMIC-FFP Generic Software Model“ wird als „Mapping Phase“ bezeichnet. Abbildung 4 zeigt die Arbeitsschritte zur Erzeugung des „COSMIC Generic Software Models“:

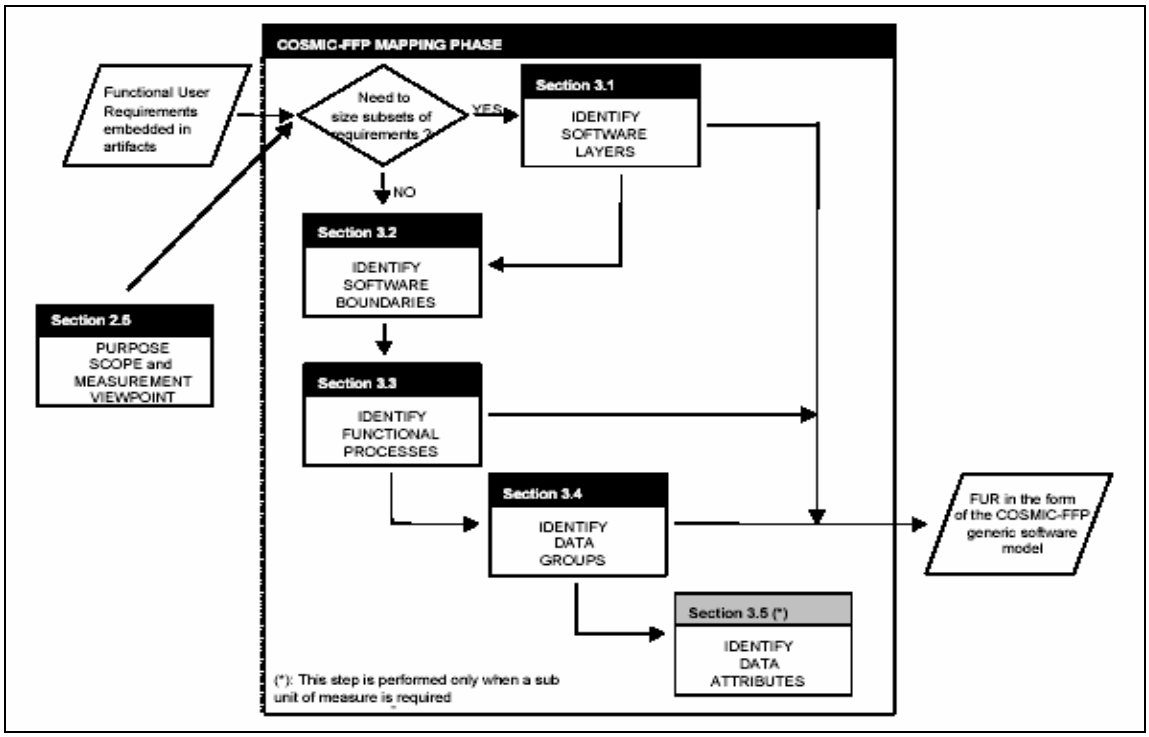


Abb. 4: „Mapping Phase“ bei der COSMIC-FFP Methode (vgl. [COSMIC 03] S.30)

Aus den bestimmten FURs kann sich je nach Komplexität und Architektur oder gewünschter Zielsetzung für eine Messung der zu bewertenden Software eine Aufteilung des Modells in verschiedene Schichten bzw. Layer anbieten. Layer können dabei unterschiedliche Software-Schichten repräsentieren, wie z.B. „Gerätetreiber-Schicht“ oder „Betriebssystem-Schicht“. Für einen **Layer** ist folgende Definition gegeben: (vgl. [COSMIC 03] S.31)

> *„Ein Layer ist das Ergebnis der funktionalen Partitionierung der Software-Umgebung, sodass alle im Layer enthaltenen funktionalen Prozesstypen auf dem gleichen Abstraktions-Level ausgeführt werden.*
> *In einer „Multi-Layer“ Software-Umgebung, tauscht Software in einem Layer Daten mit Software in einem anderen Layer durch die miteinander in Beziehung stehenden funktionalen Prozesse aus. Diese Interaktionen sind hierarchisch in ihrer Natur; paarweise betrachtet, ist ein Layer einem anderen Layer untergeordnet. Ein untergeordneter Layer bietet Software in anderen Layern funktionale Dienste an, die er selbst bereitstellt. Die Messmethode definiert „Peer-To-Peer“-Austausch als den Austausch von Daten zwischen zwei Softwareteilen innerhalb eines Layers.”*

Das Ermitteln von Layern aus den FURs ist als iterativer Prozess zu sehen, bei dem sich die exakten Schichten mit dem Fortschritt der Mapping-Phase herausbilden. Für einen Layer gelten neben der Definition folgende Prinzipien: (vgl. [COSMIC 03] S.31)

a) Software in einem Layer liefert Funktionalität an einen Nutzer.

b) Software innerhalb eines untergeordneten Layers bietet Software in anderen Layern, die in diesem Fall Nutzer darstellen, funktionale Dienste.

c) Software eines untergeordneten Layers kann ohne Unterstützung des Nutzer-Layers arbeiten.

d) Software eines Layers kann nicht richtig arbeiten, wenn Software eines untergeordneten Layers, von dem er abhängig ist, nicht richtig arbeitet.

e) Software eines Layers muss nicht die gesamte Funktionalität nutzen, die Software eines untergeordneten Layers anbietet.

f) In einer Schichten-Hierarchie kann Software irgendeines Layers einem anderen Layer untergeordnet sein, welchem er dann Dienste anbietet.

g) Software eines Layers, sowie Software eines untergeordneten Layers können Daten physisch teilen und austauschen. Dabei interpretiert die Software eines jeden Layers die Datenattribute unterschiedlich und/oder gruppiert sie in unterschiedlichen Datengruppen.

h) Falls eine Software Daten mit einer anderen Software teilt, sollten diese nicht unterschiedlichen Layern zugeordnet sein, wenn sie die Datenattribute der geteilten Daten in gleicher Weise interpretieren.

Des weiteren sollen folgende Regeln zum identifizieren von Layern genannt werden: (vgl. [COSMIC 03] S.32)

a) Funktionale Software-Pakete, die eine Funktion erfüllen, wie Datenbank-Management-Systeme, Betriebssysteme oder Geräte-Treiber, werden generell als eindeutige Layer betrachtet.

b) Falls eine Software einer Schichten-Architektur nachempfunden ist bzw. darauf aufbaut, dann sollte diese Schichten-Architektur zur Identifikation der Layer verwendet werden.

c) Die Software-Anwendung stellt den höchsten Layer dar.

d) Im Zweifel sollten Konzepte der *Kohäsion und Kopplung* zur Abgrenzung von interagierenden Layern angewandt werden.

Neben der Bestimmung von Layern für ein „COSMIC-FFP Generic Software Model“ müssen Softwaregrenzen der zu messenden Software identifiziert werden. Die zu bestimmende Softwaregrenze (engl.: „**Boundary**“) dient der Identifizierung, was Bestandteil einer zu messenden Software ist, und was zur Umgebung einer Software gehört. Grundsätzlich ist eine Software durch Hardware begrenzt. Den Datenfluss in eine Software aus funktionaler Betrachtungsweise zeigt folgende Abbildung:

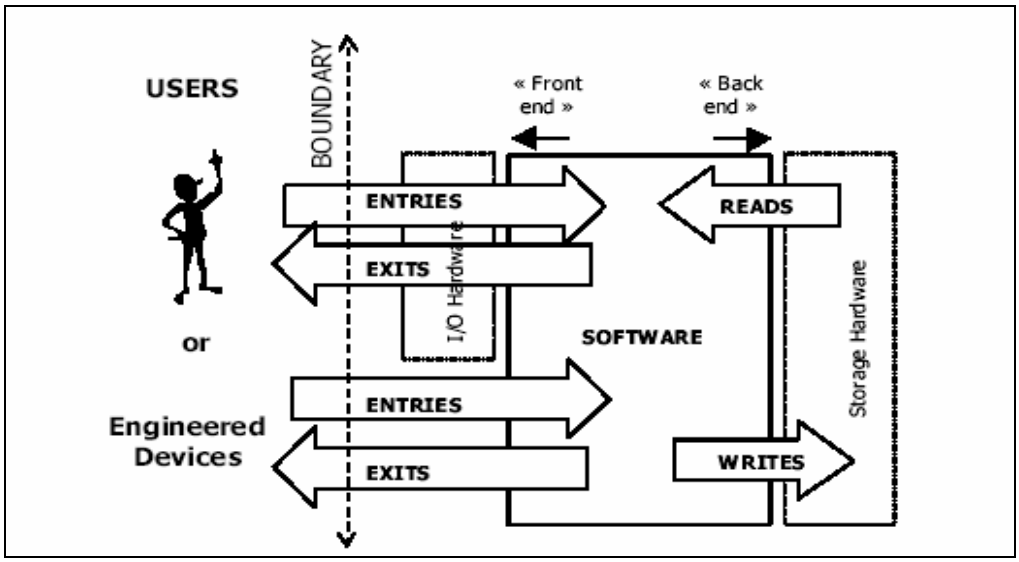


Abb. 5: Datenfluss bei der COSMIC-FFP-Methode (vgl. [COSMIC 03] S.19)

Dabei wird bei der „COSMIC-FFP-Methode“ in „front-end“ bzw. „back-end“-Hardware unterschieden. Zum „front-end“ einer Software gehören z.B. jene Bereiche, die Ein-/Ausgabe-Geräte darstellen, wie Tastatur, Mouse oder andere Eingabegeräte. Als „front-end“-Ausgabegeräte sind Drucker und Bildschirm anzusehen. Weitere Ein-/Ausgabegeräte können für eingebettete Steuerungssysteme auch Sensoren darstellen, die Messdaten liefern bzw. steuernde Funktionen haben. Als „back-end“-Hardware werden persistente Speichermedien, wie z.B. Arbeitsspeicher (RAM), ROM und Festplattenspeicher bezeichnet. Wie oben erwähnt, betrachtet die „COSMIC-FFP-Methode“ Datenbewegungen für die Messung von Full Function Points. Die in der Abbildung enthaltenen Typen von Datenbewegungen „Entry“ und „Exit“ finden in „front-end“-Richtung und die Typen „Read“ und „Write“ in „back-end“-Richtung statt. Die „COSMIC-FFP-Methode“ bietet eine genaue Definition für eine Softwaregrenze: (vgl. [COSMIC 03] S.32)

> *„Die Grenze (Boundary) ist die konzeptionelle Schnittstelle zwischen zu untersuchender Software und ihren Usern. Die Grenze einer Software ist die konzeptionelle Grenze zwischen einer Software und der Umgebung in der sie arbeitet, wie sie aus der Perspektive des Users verstanden wird. Die Grenze erlaubt es dem Software-Messenden, ohne Mehrdeutigkeit zu entscheiden, was zur zu messenden Software gehört und was zum die Software umgebenden Bereich gehört, der nicht gemessen wird.“*

Dabei wird ein User bzw. Nutzer oder Benutzer folgendermaßen definiert: (vgl. [COSMIC 03] S.32)

> *„Ein User ist jede Person oder Sache, die mit der zu messenden Software zu irgendeiner Zeit kommuniziert.*
> *Anmerkung: User können Menschen (Anwender), andere Software oder Geräte sein.“*

Auch das Bestimmen der Software-Grenzen wird als iterativer Prozess innerhalb der „Mapping Phase“ verstanden. Letztendlich muss eine Software-Grenze folgendes Prinzip erfüllen: (vgl. [COSMIC 03] S.33)

> *Es existiert eine Grenze zwischen jedem identifizierten Layer-Paar, in welchem ein Layer User eines anderen Layers ist und der letztere gemessen wird. Ebenso existiert eine Grenze zwischen zwei Software-Teilen innerhalb eines Layers, die mittels peer-to-peer-Kommunikation Daten austauschen. In diesem Fall kann jeder Software-Teil User des anderen sein.*

Für die Identifikation von Softwaregrenzen sind folgende Regeln zu beachten: (vgl. [COSMIC 03] S.33)

a) Die Softwaregrenze liegt zwischen einem Aufruf („Triggering Event“) eines funktionalen Prozesses und dem ausgelösten funktionalen Prozess.
b) Für Real-Time-Software oder spezielle Software sind die Konzepte zur Layer-Identifikation heranzuziehen.

Ein „Triggering Event“ ist dabei definiert, wie folgt: (vgl. [COSMIC 03] S.36)

> *„Ein Triggering Event erfolgt ausserhalb der Software-Grenze und initiiert ein oder mehrere funktionale Prozesse. Für eine Menge von FURs gehört jeder Triggering Event, der einen funktionalen Prozess auslöst, ausschliesslich zu diesen FURs.“*
> *Anmerkung 1: „Clock- bzw. Timer-Events“ gelten als „Triggering Events“.*
> *Anmerkung 2: Ein Event ist zustandslos, d.h. er tritt auf oder nicht.*

Da die englischen Begriffe „*Trigger*“ und „*Event*“ im technischen Bereich und in der Informatik als Begriffe fest etabliert sind, und auch in diesem, teils sehr abstrakten Zusammenhang schwer übersetzt werden können, soll der Begriff „*Triggering Event*“ weiterhin verwendet und nicht ins Deutsche übersetzt werden.

Wie oben erwähnt, ist die COSMIC-FFP Methode als ein Mittel der Software-Messung aus Nutzersicht zu verstehen. Ein User bzw. Nutzer oder Benutzer im Sinne der COSMIC-FFP Methode kann z.B. ein Anwender sein. Diese Vorstellung gilt im Besonderen für Software mit einer Benutzerschnittstelle, deren Aufgabe die Entgegennahme von Benutzereingaben ist, die Auswertung dieser Eingaben und die Lieferung eines Resultats oder Abarbeitung einer vom Benutzer angegebenen Aufgabe.

Weiterhin werden in der COSMIC-FFP Methode Softwareteile bzw. Softwarekomponenten als User betrachtet. Dieser Fall tritt ein, wenn die Softwareteile, die als Nutzer betrachtet werden, außerhalb einer Softwaregrenze zwischen zu messender Software und nutzender Software liegen. Als User können also auch Sensoren verstanden werden, die eine Software anweisen, bestimmte Aufgaben zu verrichten.
Neben der Bestimmung der Softwaregrenze durch die Eigenschaften der umgebenden Hardware, ist es notwendig zu definieren, was als Software zu verstehen ist. Für die Definition von Software im Sinne der COSMIC-FFP Methode sollen zwei generelle Prinzipien, wie sie in der COSMIC-FFP Methode zur Anwendung kommen, beschrieben werden: (vgl. [COSMIC 03] S.21)

1) Die zu modellierende und zu messende Software wird mit Eingaben versorgt und produziert für den Nutzer sinnvolle Ausgaben.
2) Die zu modellierende und zu messende Software manipuliert Informationen bzw. Informationsteile, die als Datengruppen (engl.: „Data Groups“) bezeichnet werden, welche aus Datenattributen (engl.: „Data Attributes“) bestehen.

Nachdem für jede zu messende Software bzw. Softwareteile die Grenzen bestimmt wurden, müssen die funktionalen Prozesse der zu messenden Software extrahiert werden. Diese werden aus den oben beschriebenen FUR abgeleitet. Dabei ist jeder dieser funktionalen Prozesse als eine Menge von Sub-Prozessen zu betrachten, welche wiederum durch Datenbewegungen, entweder „Entries“, „Exits“, „Reads“ oder „Writes“, gekennzeichnet sind.

Die folgende Abbildung zeigt diese Sichtweise auf das „COSMIC-FFP Generic Software Model“:

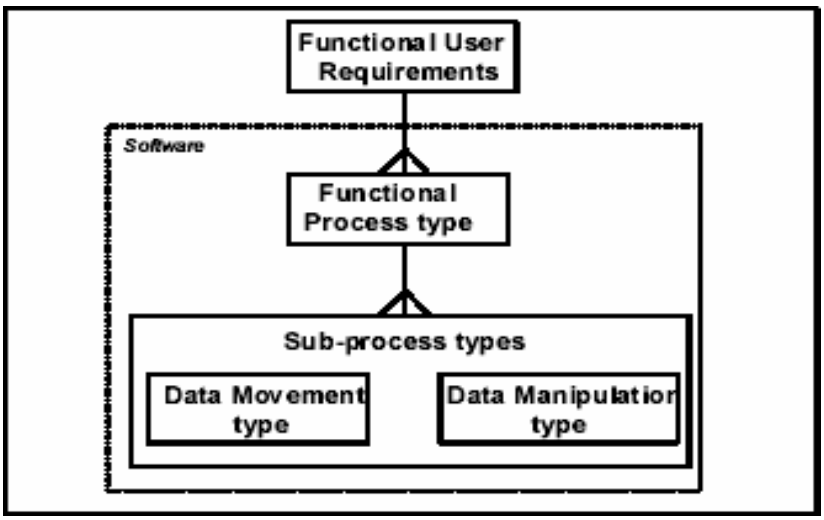


Abb.6: Das “COSMIC-FFP Generic Software Model” (vgl. [COSMIC 03] S.22)

Die Datenbewegungen werden als „Data Movements“ bezeichnet. Der in der obigen Abbildung erwähnte „Data Manipulation Type“ als Sub-Prozess-Typ wird in der Methode nicht separat betrachtet, sondern wird als Bestandteil von Datenbewegungen angesehen. Die Datenmanipulation wird in der COSMIC-FFP Methode als zu einer Datenbewegung gehörender Prozess-Typ betrachtet, da die Bewertung bzw. Messung der Datenmanipulation im COSMIC-FFP-Umfeld noch diskutiert wird und die Entwicklung der FFP-Methode darauf abzielt, eine Bewertung von Systemen zu erreichen, die sehr stark durch Datenbewegungen charakterisiert ist, wie in Echtzeit-Anwendungen oder eingebetteten Systemen, denn durch Datenmanipulationen, wie in Software-Produkten mit eher algorithmischer Charakteristik.

Nachdem die zu messende Software einem Layer zugeordnet wurde und die Grenzen der Software für eine spätere Messung festgelegt wurden, werden in einem nächsten Schritt die funktionalen Prozesse einer Software bestimmt. Auch dafür gibt die COSMIC-FFP-Methode eine feste Definition an: (vgl. [COSMIC 03] S.36)

> *„Ein funktionaler Prozess ist eine elementare Komponente einer Menge von funktionalen Systemanforderungen einschließlich einer eindeutigen, geschlossenen und unabhängigen Menge von Datenbewegungen. Er wird ausgelöst durch einen oder mehrere „Triggering Events“, die entweder direkt auftreten, oder durch einen „Auslöser“ verursacht werden. Ein funktionaler Prozess gilt als abgeschlossen, wenn er alles getan hat, um das auslösende Ereignis zu beantworten.*
> *Anmerkung: Ein ’Auslöser’ ist ein User des zu messenden Systems, der Daten vom „Triggering Event“ zum funktionalen Prozess überträgt.“*

Für die Bestimmung von funktionalen Prozessen können folgende Prinzipien herangezogen werden: (vgl. [COSMIC 03] S.36)

a) Ein funktionaler Prozess ist aus wenigstens einer identifizierten funktionalen Systemanforderung abgeleitet.

b) Ein funktionaler Prozess wird ausgeführt, wenn ein „Triggering Event“ auftritt.

c) Ein funktionaler Prozess umfasst wenigsten 2 Datenbewegungen, entweder „Entry“ + „Exit“ oder „Entry“ + „Write“.

d) Ein funktionaler Prozess gehört ausschließlich zu einem Layer.

e) Ein funktionaler Prozess endet, wenn ein asynchroner Zeitpunkt eintritt. Ein asynchroner Zeitpunkt wird erreicht, wenn die letzte Datenbewegung des funktionalen Prozesses („Exit“ oder „Write“) auftritt und keine andere (gleichzeitig ausgeführte) Datenbewegung mehr existiert.

Da ein funktionaler Prozess mit Datenbewegungen verbunden ist, soll im Folgenden erläutert werden, wie Daten in der „COSMIC-FFP Methode“ verstanden werden. Dazu werden folgende Definitionen für Datengruppen und Datenattribute verwendet: (vgl. [COSMIC 03] S.37)

> *„Eine Datengruppe besteht aus einer eindeutigen, nicht leeren, unsortierten und nicht redundanten Menge von Datenattributen, in der jedes enthaltene Datenattribut einen bestimmten Teil eines bestimmten Objekts bzw. einer bestimmten Entität („object of interest“) darstellt. Eine Datengruppe ist (außerdem) durch ihre Persistenz charakterisiert.“*

Für Datengruppen sollen folgende Persistenz-Eigenschaften genannt werden:

1) ***Transiente Persistenz*:** Die Datengruppe existiert nur für die Dauer der Ausführung des sie verwendenden funktionalen Prozesses.
2) ***Kurze (Short) Persistenz*:** Die Datengruppe überdauert den sie verwendenden funktionalen Prozess. Die Datengruppe existiert jedoch nur für die Dauer der sie verwendenden Software.
3) ***Indefinite Persistenz*:** Die Datengruppe existiert auch über die Dauer der sie verwendenden Software hinaus.

Die Persistenz-Eigenschaft einer Datengruppe wird während der „Measuring Phase“ verwendet, um die vier Typen von Sub-Prozessen, „Entry“, „Exit“, „Read“, „Write“ zu identifizieren. Folgende Prinzipien können zur Bestimmung von Datengruppen herangezogen werden: (vgl. [COSMIC 03] S.37)

a) Eine Datengruppe muss auf dem, die Software ausführenden, Computer-System „materialisiert“ sein.
b) Jede identifizierte Datengruppe muss eindeutig und durch ihre Datenattribute erkennbar sein.
c) Eine Datengruppe muss in direkter Beziehung zu einem „Object Of Interest“ stehen, wie es in den funktionalen Systemanforderungen der Software beschrieben ist.

Zur „Materialisierung“ von Datengruppen sollen drei Formen genannt werden, wie sie in der Praxis auftreten: (vgl. [COSMIC 03] S.37f)

a) durch physischen Speicher „materialisiert“, z.B. als Datei, Datenbank-Tabelle oder ROM (kurze oder indefinite Persistenz)
b) durch flüchtige Speicherung z.B. im RAM oder Cache (transiente oder kurze Persistenz)
c) durch Speicherung auf I/O-Geräten (transient, kurz oder indefinit, je nach I/O-Gerät und Speicher-Medium)

Neben der Definition von Datengruppen soll hier auch die Definition von Datenattributen angegeben werden: (vgl. [COSMIC 03] S.41)

> *„Ein Datenattribut ist der kleinste Informationsteil innerhalb einer Datengruppe. Dabei repräsentiert er eine bestimmte Bedeutung aus der Perspektive der funktionalen Systemanforderungen (FUR) einer Software.“*

Für das Verständnis von Datenattributen aus der Perspektive der funktionalen Systemanforderungen werden durch die „COSMIC-FFP-Methode“ drei Typen von Datenattributen angegeben, die sich in gültige und ungültige Typen unterteilen: (vgl. [COSMIC 03] S.41)

1) ***Beschreibende oder Anwender-Daten*:** Daten, deren Bedeutung unabhängig von der Art der Verarbeitung durch die zu messende Software betrachtet werden.

2) ***Kontextbezogene Daten*:** Daten, die beschreiben, welche, wann oder wie Daten von der zu messenden Software verarbeitet werden. Die funktionale Bedeutung dieser Daten ist durch ihren Kontext festgelegt.

3) ***Daten, die Implementierungstechnik betreffend.***

Die Typen 1) und 2) sind gültige Datenattribut-Typen, der Typ 3) ist ein ungültiger Typ.

Nachdem durch die oben geschilderten Prozeduren aus den FURs ein „COSMIC-FFP Generic Software Model" erzeugt wurde, kann die Messung der Full Function Points durchgeführt werden.

Für das „Functional Size Measurement" ist die Betrachtung der Messziele („purpose of a measurement"), des Messrahmens („scope of a measurement") und des Messstandpunktes („viewpoint of a measurement") wesentlich. Vor der Anwendung der „COSMIC-FFP-Methode" zur Software-Messung sollten diese drei wesentlichen Aspekte klar herausgestellt und festgelegt werden. Für die „COSMIC-FFP Methode" ist ein Messziel definiert als: (vgl. [COSMIC 03] S.27)

> *„Eine Aussage, warum die Messung durchgeführt wird und/oder für was das Meßresultat genutzt werden soll.".*

Des weiteren wird der Messrahmen definiert als:

> *„Die Menge der funktionalen Systemanforderungen, die in eine spezifische Messung einfließen."*

Die Betrachtungsweise einer Messung ist bei der „COSMIC-FFP-Methode" gegeben durch den „End User Measurement Viewpoint" und den „Developer Measurement Viewpoint". Diese beiden sind folgendermaßen definiert: (vgl. [COSMIC 03] S.28f)

> ***End User Measurement Viewpoint:*** *„Ein Messstandpunkt, der ausschließlich etwas über die Funktionalität einer zu entwickelnden und/oder zu liefernden Anwendungssoftware aussagt, die eine bestimmte funktionale Systemanforderung zu erfüllen hat. Dieses geschieht aus der Sichtweise des Users, der entweder ein Menschen (Anwender) ist, der mit der Software interagiert oder eine andere Software, die mit der zu messenden Software Daten teilt oder austauscht, oder ein zeitlicher Mechanismus, der eine „Batch-Anwendung" steuert. Der End User Measurement Viewpoint ignoriert alle Funktionalitäten, von weiter Software, durch die der User mit der zu messenden Software interagieren kann."*
> ***Developer Measurement Viewpoint:*** *„Ein Messstandpunkt, der etwas über die gesamte Funktionalität jedes einzelnen Teils einer zu entwickelnden und/oder zu liefernden Software aussagt, die eine bestimmte funktionale Systemanforderung zu erfüllen hat."*

Nachdem wesentliche Begriffe des „Functional Size Measurement" im Zusammenhang mit der „COSMIC-FFP Methode" kurz erläutert wurden, soll im Folgenden beschrieben werden, wie aus dem „COSMIC-FFP Generic Software Model" Full-Function-Points gemessen werden. Dafür verwendet die „COSMIC-FFP Methode" Regeln und Prozeduren für die Messung und erzeugt einen Wert, der die Größe bzw. den Umfang der gemessenen Software darstellt. Diese Phase der „COSMIC-FFP Methode" wird als „Measuring Phase" bezeichnet, ist in Abbildung 7 dargestellt und wird nach folgendem Prinzip durchgeführt: (vgl. [COSMIC 03] S. 23)

> *„Die funktionale Größe bzw. der funktionale Umfang einer Software ist direkt proportional zur Anzahl der von ihr durchgeführten Datenbewegungen."*

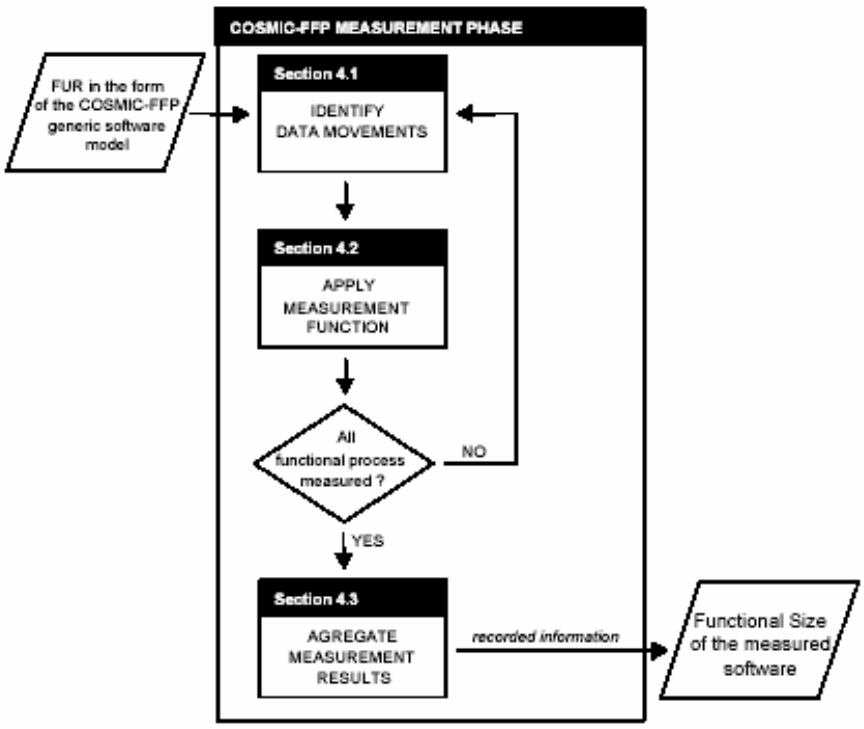


Abb. 7: Prozesse der „Measuring Phase“ (vgl. [COSMIC 03] S.43)

Die erwähnten Regeln und Prozeduren zur Bestimmung der Anzahl der Datenbewegungen sollen im Folgenden genannt und erläutert werden:

- ***Messfunktion*:** Jeder Instanz einer Datenbewegung ist durch eine Messfunktion gemäß ihres Typs ein Wert zugeteilt.
- ***Maßeinheit*:** Der Messstandard von **1 Cfsu** („COSMIC Functional Size Unit“) ist äquivalent zu einer einzelnen Datenbewegung.
- ***Additivität*:** Die funktionale Größe eines funktionalen Prozesses ergibt sich aus der Summe der Werte der Messfunktionen seiner Datenbewegungen.
- ***Zerlegung der Messung*:** Für mehr Präzision in der Messung der Datenbewegungen, kann die Maßeinheit in Teileinheiten zerlegt werden.

Aufgrund der genannten Messregeln und des generellen Prinzips Nr. 1 für die Definition einer Software (siehe oben), dass die zu bemessende Software eine Eingabe erhält und eine für den Nutzer sinnvolle Ausgabe erzeugt, ist ersichtlich, dass der kleinste zu messende Wert eines funktionalen Prozesses 2 Cfsu ist. Er kann entweder durch die Datenbewegungen „Entry“ + „Exit“ oder aber durch „Entry“ + „Write“ zustande kommen. Eine Datenbewegung ist definiert als: (vgl. [COSMIC 03] S.44)

> *„Eine COSMIC-FFP Datenbewegung ist eine Komponente eines funktionalen Prozesses, welche ein oder mehrere Datenattribute einer zugehörigen Datengruppe bewegt.*
> *Eine Datenbewegung erfolgt während der Ausführung eines funktionalen Prozesses. Es gibt vier Typen von Datenbewegungen, „Entry“, „Exit“, „Read“ und „Write“. Jeder dieser Typen enthält spezifische Datenmanipulationen. Ein COSMIC-FFP Datenbewegungstyp ist äquivalent zu einem „ISO Base Functional Component Type“ (BFC Type)“*

Dabei sind die vier Typen von Datenbewegungen folgendermaßen definiert: (vgl. [COSMIC 03] S.44)

> *„Ein **Entry** (E) bewegt eine Datengruppe von der User-Seite einer Softwaregrenze in den funktionalen Prozess. Dabei werden Datenmanipulationen, wie z.B. eine Aufbereitung der Datengruppe für die Verarbeitung durch den funktionalen Prozess nicht weiter bewertet.*

*Ein **Exit** (X) bewegt eine Datengruppe aus einem funktionalen Prozess über die Software-Gernze zum User. Datenmanipulationen, wie z.B. die Formatierung der Daten wird nicht weiter bewertet.“*

*Ein **Read** (R) bewegt eine Datengruppe aus einem persistenten Speicher, der einem funktionalen Prozess zugeordnet ist in den funktionalen Prozess. Alle Datenmanipulationen, die den Read bedingen, werden nicht mitgezählt.“*

*Ein **Write** (W) bewegt eine Datengruppe aus einem funktionalen Prozess in den dem Prozess zugehörigen persistenten Speicher. Datenmanipulationen, die das Write bedingen, werden nicht gezählt.“*

*„**De-Duplikation**: Mehrere Datenbewegungen, die innerhalb eines funktionalen Prozesses dieselbe Datengruppe mehrmals bewegen, werden nur einmal gezählt.“*

Ein Problem der Aufwandsabschätzung mit der „COSMIC-FFP Methode“ vor der eigentlichen Entwicklung einer Software besteht im frühzeitigen Messen bzw. Abschätzen von Datenbewegungen einzelner funktionaler Prozesse, da in einer frühen Entwicklungsphase noch nicht das Level der Definition von funktionalen Prozessen und Sub-Prozessen erreicht wird. Dieses Dilemma der Software-Messung anhand der Datenbewegungen kann mit der COSMIC-FFP Methode nur durch eine Schätzung der Datenbewegungen gelöst werden. Dafür sollten Skalierungsfaktoren aus schon vorhandenen Software-Produkten bestimmt werden. Unter der Annahme, dass existierende Software nutzbare Ergebnisse über die Anzahl der extrahierten funktionalen Prozesse geliefert hat, könnte eine Lösung darin bestehen, den Mittelwert der Anzahl der funktionalen Prozesse zu ermitteln. Anhand dieses Mittelwertes und einer weiteren Annahme über die in jedem funktionalen Prozess wahrscheinlich zu erwartenden Datenbewegungen, kann so zu einem frühen Zeitpunkt eine Abschätzung für die zu erwartenden Datenbewegungen erreicht werden. Die Generalisierung von Skalierungsfaktoren für die „COSMIC-FFP Methode“ befindet sich allerdings noch im Entwicklungsstadium. (vgl. [COSMIC 03] S.24) Nachdem die Bestimmung der Datenbewegungen innerhalb der funktionalen Prozesse durchgeführt wurde, muss auf diese Datenbewegungen die oben angesprochene Messfunktion angewendet werden. (vgl. [COSMIC 03] S.50f)

„Die COSMIC-FFP Messfunktion ist eine mathematische Funktion, die ihrem Argument gemäß des COSMIC-FFP Messstandards einen Wert zuordnet. Das Argument der COSMIC-FFP Messfunktion ist eine Datenbewegung.“

*„Der COSMIC-FFP Messstandard von **1 Csfu („COSMIC Functional Size Unit“) ist definiert als die Größe einer elementaren Datenbewegung**.“*

Im Fall, dass auf alle Datenbewegungen die Messfunktion angewendet wurde, kann das Messergebnis erzeugt werden, für das folgende Prinzipien zur Anwendung kommen:

a) Für jeden funktionalen Prozess ist die funktionale Größe („functional size“) die Summe der Entries, Exits, Reads und Writes:

$$Size_{Cfsu}\ (functional\ process_i) = \sum size(entries_i) + \sum size(exits_i) + \sum size(reads_i) + \sum size(writes_i)$$

b) Für jeden funktionalen Prozess ist die funktionale Größe von Änderungen in den funktionalen Systemanforderungen die Summe der dadurch geänderten Datenbewegungen gemäß der Formel:

$$Size_{Cfsu}\ (Change(functional\ process_i)) = \sum size(added\ data\ movements) + \sum size(modified\ data\ movements) + \sum size(deleted\ data\ movements)$$

c) Die Größe eines zu messenden Softwareteils innerhalb eines Layers wird durch die in a) und b) gegebenen Formeln ermittelt, wobei die funktionalen Prozesse des Softwareteils betrachtet werden.

d) Die Größe von Layern oder (mehreren) Softwareteilen innerhalb eines Layers wird durch die Summe der einzelnen Layer bzw. Softwareteile ermittelt. Das aber nur, falls alle unter dem gleichen Messstandpunkt gemessen wurden.

e) Die Größe von unterschiedlichen Layern sollte nur gemessen werden, wenn diese Bestimmung der Größe aus der Betrachtung der Messziele Sinn macht.

Die in der „Measuring Phase“ erzeugten Full Function Points geben ein Maß für den funktionalen Umfang einer gemessenen Software an. Für die Abschätzung des Aufwandes für die Software-Entwicklung kann dieses Maß verwendet werden. Abschließend zeigt die folgende Abbildung 8 eine graphische Gegenüberstellung von IFPUG FP’s und COSMIC Cfsu’s, die durch eine Messung von 14 Sub-Systemen aus 3 Systemen zustande gekommen ist.

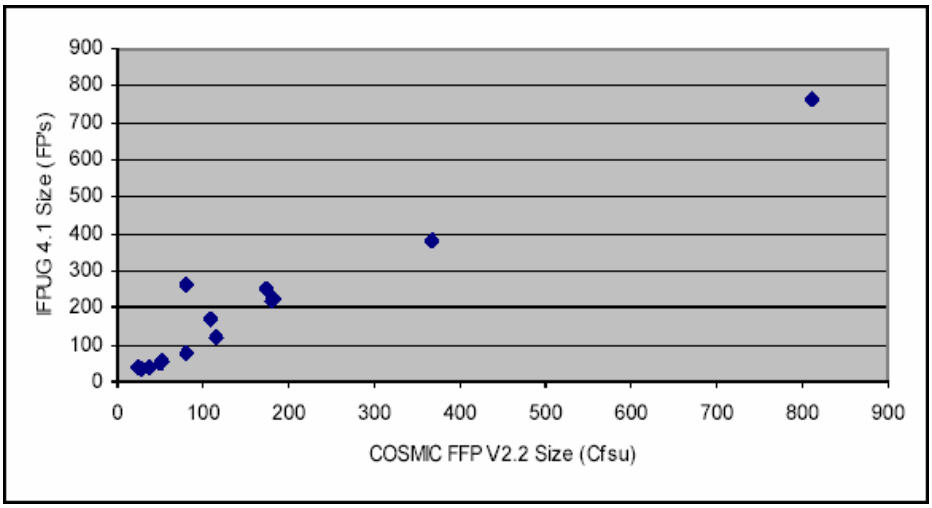


Abb.8: Gegenüberstellung von IFPUG FP’s und COSMIC Cfsu’s (vgl. [COSMIC 03] S.55)

Das folgende, an der Uni Magdeburg entwickelte Tool, ermöglicht die Anwendung der COSMIC FFP im Web (http://www.smlab.de).

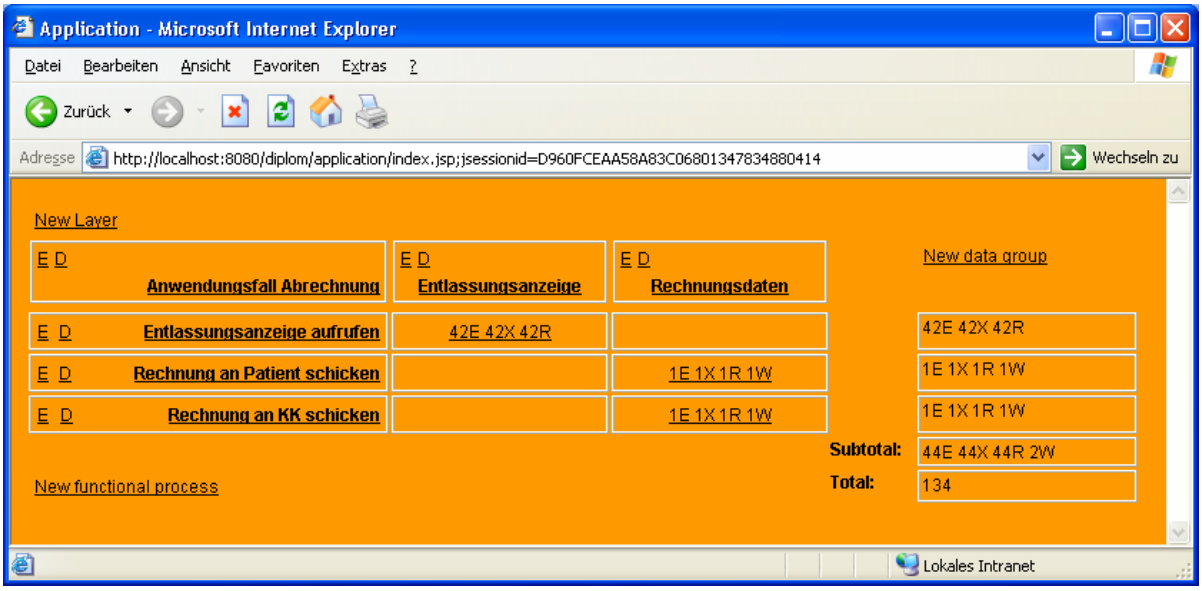


Abb. 9: Anwendungsbeispiel zum COSMIC-FFP-Tool

Zu den möglichen Tools für eine semiautomatische COSMIC-FFPAnwendung siehe vor allem [Stambollian 2006]. Hier werden beispielsweise u. a. die FFP-Tools *COSMICXpert, ISBSG Release 9, MeterIT Tool Suite, ExperiencePro 3.1, KnowledgePlan 4.2, SIESTA Version 1.2.3* und *XForns-Format* näher erläutert.

4. Anwendungserfahrungen zur COSMIC-FFP

4.1 COSMIC-FFP-Messungen nach Efe et al.

Im Folgenden zitieren wir die wesentlichsten Ergebnisse der Anwendung unterschiedlicher FSM-Methoden, darunter auch der COSMIC-FFP, nach Efe et al. [Efe2006].

"The case project is a web based, **military inventory management project integrated with a document management system.** It is a data-strong system, which also involves a number of algorithmic operations. The general characteristics are summarized in the following:

- The development life cycle was Waterfall.
- The project staff consisted of 6 people, 1 project manager, 2 senior software engineer, 2 part time software engineers and 1 part time software engineer.
- The types of software products and programming language(s) used for the project are; Internal Development Framework and Java as programming languages, IBM WebSphere Application Developer as development environment, Rational Rose as Analysis and Design tool, Oracle 9i as Database Management System, Tomcat as Application Server.
- The project documents were prepared in compliance with the organizational document standards. The company uses an SRS standard developed by the company itself.
- The project was started in October 2004 and completed in December 2005.
- The total effort spent for this project **is approximately 7500 man-hours**.
- Both size estimations presented here are conducted using Software Requirements Specification (SRS) document of the project, which involves 123 Use Cases. "

"For each type of count, we first identified the Transactions based on the SRS document of the software product. This document involves the detailed information the three FSM methods require. There were 123 Use Cases in the SRS document which correspond to 123 Transactions. The functional size measurement of the software product was done by IFPUG FPA, Mark II FPA and COSMIC FFP, consequently. The functional size measurement of the case project using the unification model was implemented by the same person, who also made the conventional measurement, in order to avoid the subjectivity of the FSM methods which can affect the measurement results. As the Transactions, Data groups and DETs have already been identified during the conventional measurement, we used these figures during the implementation of the unification model. The results of conventional measurement and the measurement by the unification model for IFPUG FPA, Mark II FPA and COSMIC FFP measurements are given in Table 1, Table 2 and Table 3 respectively.

	Number of Elementary Processes	Functional Complexities for Function Types					Total Functional Complexity
		ILFs	EIFs	EIs	EOs	EQs	
Traditional Measurement	123	294	0	262	343	26	925
Unification Model Measurement	123	294	0	262	343	26	925

Tab. 1: Case Project - IFPUG FPA Size Measurement Details

	Number of Logical Transactions	Number of Input DETs	Number of Output DETs	Number of Data Entity Types Referenced	Functional Size (Mark II FP)
Traditional Measurement	123	559	1,679	343	1,330.14
Unification Model Meas.	123	559	1,679	343	1,330.14

Tab. 2: Case Project - Mark II FPA Size Measurement Details

	Number of Functional Processes	Number of Entries	Number of Exits	Number of Reads	Number of Writes	Functional Size (Cfsu)
Traditional Measurement	123	206	364	334	156	1,060.0
Unification Model Meas.	**123**	**206**	**364**	**334**	**156**	**1,060.0**

Tab. 3: Case Project COSMIC FFP Size Measurement Details"

Ausgehend von diesen Ergebnissen erhält man folgende ***Aufwandsschätzungen für Neuentwicklung***:

- geschätzte Dauer = 7500 Mh = 7500/(21*8) MM ≈ 44,64 MM für 1060 COSMIC-FFP:

$$1\ Cfsu_{Neuentwicklung} \approx 0{,}04\ MM = 0{,}04\ PM$$

- oder: Konvertierung von 925 UFP nach 1060 FFP

nach Bundschuh: $PM = 0{,}015216\ FP^{1{,}29}$ [Bundschuh2000]

unter der Annahme, dass die Kostentreiben die UFP senken (um den Faktor 0,65) oder erhöhen können (um den Faktor 1,35 (alle Kostentreiber Wichtung 5) kann man die Bundschuh-Formel als ***durchschnittlichen Aufwand*** verwenden, was ergibt:

$925^{1{,}29}0{,}015216 \approx 102$ PM auch für 1060 COSMIC FFP, also

$$1\ Cfsu \approx 0{,}1\ PM$$

- oder: Konvertierung nach Jones (in Sneed2005]) 22 FP/PM und 1 FP ≈ 1,13 Cfsu (nach Abran s.u.) ergibt

$$1\ Cfsu \approx 0{,}04\ PM$$

(MM – Mannmonate, PM – Personenmonate, FP – IFPUG Function Points)

4.2 Die COSMIC-FFP-Anwendung für die Softwarewartung bei Sogeti

Die Niederländische Firme Sogeti hat die COSMIC-FFP vor allem für die Wartung von Softwaresysteme angewandt [Koppenberg2004].

"Function Point Analysis will likely be replaced by next generation functional size measurement methods like COSMIC-FFP in the next decade [4]. The main reason for this is that Function Point Analysis does not always works well with a growing number of contemporary software developing environments. COSMIC-FFP can be seen as the most important candidate to replace Function Point Analysis. From a functional point of view, planned enhancement projects can be split into: adding new functionality, changing existing functionality, deleting existing obsolete functionality, replacing existing functionality (= delete + new)"
"The measurement of new functionality is identical with the regular measurement of functional size. The others are more difficult and more controversial because there are different ways of dealing with those enhancements within Function Point Analysis.
In COSMIC-FFP for any functional process, the functional size of changes to the Functional User Requirements is aggregated from the sizes of the corresponding modified data movements according to the following formula:

$\sum$ *size(added data movements)* +
$\sum$ *size(modified data movements)* +
$\sum$ *size(deleted data movements)"*

"Retesting is not mentioned in the Measurement Manual. The number of functions that had to be retest can be much larger than the number of enhanced functions. Retesting can be estimated by multiplying the unchanged functions that have to be retest by a productivity rate. Analogously to

the removal of functions Sogeti has determined this productivity rate as the productivity rate for new development multiplied by an impact factor 0.10. Based on own experiences, this factor can be calibrated. NESMA doesn't explicitly estimate retesting of unchanged functions. The effects of applying the approach with the default values is shown in table 4:

Funct. process	Before	New	Change	Remove	Test	Total	After
FP-1	5						5
FP-2	10				10		10
FP-3	5		2				5
FP-4	8		4				8
FP-5	6		4				7
FP-6	5			5			0
FP-7	0	9					9
Cfsu	**39**	**9**	**10**	**5**	**10**		**44**
Prod. Rate (h/Cfsu)		8	8	8	8		
Impact		1.00	1.00	0.10	0.10		
Hours		**72**	**80**	**4**	**8**	**164**	

Tab. 4: Example estimating enhancement project with COSMIC-FFP

The advance of using Cosmic for measuring enhancements is that the productivity rate for new development can be reused and so it's possible to compare the productivity rate of new development and enhancements. A disadvantage is that the right impact factor for deleting functions and retest has to be found. The advantages of having a calibrated impact factor are obvious: the estimates will be more accurate and the productivity is in balance with the effort of removal and testing. There are also disadvantages in regard to calibrated impact factors: calibration takes time for research and analysis and a above that creates different "sizes" for each organization. The last complicates benchmarking. The disadvantages make that managers takes the usually minor effects of a "methodical" fixed impact factor for granted and no calibration is done."

Hierbei ist zunächst zu verzeichnen, dass eine spezielle ***Systemwartung*** ergab (konkretes Projekt)

39 Cfsu → 44 Cfsu (Δ 5 Cfsu)

Darüber hinaus ergaben sich folgende relative ***Aufwandsmerkmale für die Wartung***:

- **Hinzufügen neuer Funktionalität: 9 Cfsu → 72 PH ≈ 0,43 PM → 0,048 PM/Cfsu**
- **Änderung von Funktionalität: 10 Csfu → 80 PH ≈ 0,48 PM → 0,048 PM/Cfsu**
- **Ersetzen von Funktionalität: 5 Cfsu → 4 PH ≈ 0,024 PM → 0,005 PM/Cfsu**
- **Testen von Funktionalität: 10 Cfsu → 8 PH ≈ 0,048 PM → 0,005 PM/Cfsu**

Damit ergibt sich eine durchschnittliche Wartungsbewertung für den Aufwand von

$$1\ Cfsu_{Wartung} \approx 0{,}027\ PM$$

4.3 Die Konvertierungsuntersuchungen von Abran et al.

Im Beitrag von Abran et al. werden allgemeine Konvertierungsansätze zwischen den verschiedenen FP-Methoden an Beispielen diskutiert [Abran 2005]. Wir wählen wiederum den Zusammenhang zwischen FPA und FFP.

> ***"Context:*** The duplicate measurement results reported next were collected in 2005 by one of the authors (Desharnais) using FPA 4.1 and COSMIC-FP 2.2. This data set comes from one governmental organization and was measured using the documentation of completed projects.
> ***Measurement results:*** The measurement results of the duplicate measurement of the four applications are reported in Table 5. These data points are also presented graphically in Figure 4a, with the FPA data on the x-axis and the COSMIC data on the y-axis.
> ***Analysis and interpretation:*** The linear regression model of the data in Figure 4 provides the following convertibility formula, with a coefficient of determination (R^2) of 0.91:
>
> **Y(Cfsu) = 0.84 * (UFP) + 18**
>
> Again, there is a large difference in convertibility results for project number 2 at 362 FPA points, both in absolute and relative terms. This means again that there must be some peculiarities in the way that functionality is measured that leads to non straightforward convertibility. In the FPA measurement method, the data is taken into account from multiple perspectives, once as logical data files (ILF – Internal logical file and EIF – External interface file) and once again whenever that are references in FPA transactions (Input, Output, Enquiries transaction types).This has already been noted in [Abran 2005] where it is reported that in FPA-like methods 30 to 40% of functional size comes from the data files. By taking into account only the FPA data files points from the FPA transaction types points, it is investigated next whether a better convertibility ratio could be derived by excluding the FPA data files, that is by taking only the NESMA points coming from the transactions (TX) only"

Software	**FPA (1)**	**COSMIC 2.2 (2)**	**With convertibility formula (3)**	**Convertibility Delta (4) = (3) – (2)**	**% Delta (5)=(4)/(2)**
1	103	75	105	30	39%
2	362	209	322	113	54%
3	124	170	122	-48	-28%
4	263	203	239	36	18%
5	1146	934	981	47	5%
6	570	675	497	-178	-26%

Tab. 5: Desharnais 2005 Data set

Bildet man die Durchschnittswerte, so erhält man hierbei für die ***Neuentwicklung***:

jeweils 428 FPA ergeben ca. 378 Cfsu

Verwendet man wiederum die Umrechnung von Bundschuh zur Aufwandsbestimmung so erhält man:

$428^{1,29}0{,}015216 \approx 37{,}75$ PM auch für 378 COSMIC FFP, also

1 Cfsu ≈ 0,1 PM

4.4 Die Messwertegegenüberstellung von Levesque

Levesque und Bevo diskutieren vor allem die unterschiedlichen funktionalen Größen unter Verwendung der allgemeinen Softwareprozessmerkmale [Levesque2001].

> "The results presented in this article were obtained by applying the **COSMIC FFP (1999) method, version 2.0**, and the process described in the Measurement Manual, version 1.0, prepared for the company. In Table 6, all the measured projects are listed. Three of the projects measured are not included in this sample for various reasons related to cost and timing. Only the names of the projects have been changed to preserve confidentiality. The relation between the cost of each project and its corresponding functional the correlation coefficient is 0.62. The average COST/COSMIC FFP ratio is $6.95 K per FFP and the average COSMIC FFP /man-month is 4.09. The regression line for the cost model indicates that the fixed cost to start a project is $36,351 and that the variable cost is $3,400 per function point. All the projects used to calculate these values have a cost which is higher than this fixed cost. In terms of man-months, the fixed cost is 4.34 man-months per project and the variable cost is 0.26 man-months per function point. This fixed cost is higher than the amount observed for three of the projects (fewer than 4 man-months). These figures are valid for projects in the 0 to 350 COSMIC functional size units range. The quality of these data is quite good, as can be seen from Figures 3 and 4. For the mean cost per function point, only one point was out of the 1-sigma range, and the deviation is more than 3-sigma for this point. This example is taken from the "I" (for infrastructure) project. In the other case, for the number of function points per man-month, four points are out of the 1-sigma range, one of them being out of the 2-sigma range, which confirms the earlier observation. When divided into subgroups according to project type, two subgroups have shown promising results; however, the size of the sample is too small to provide more information on these."

Projects	**FFP**	**Costs $K**	**man-months**	**Cost/FFP**	**FFP/MM**
A	30	49.0	3.8	1.63	7.89
B-1	7	51.0	5.0	7.29	1.41
C	170	254.0	21.7	1.49	7.83
B-2	17	41.0	3.4	2.41	5.00
B-3	61	59.0	7.8	0.97	7.82
D	167	137.0	12.6	0.82	13.25
E	8	40.0	3.2	5.00	2.50
F	13	81.0	6.5	6.23	2.02
G	158	574.0	53.1	3.63	2.98
H	349	1617.0	121.2	4.62	2.88
I	18	713.0	55.7	39.61	0.32
J	60	259.0	19.4	4.32	3.09
K	15	91.0	7.4	6.07	2.03
L	8	97.5	9.3	12.19	0.86
M	35	278.0	25.1	7.94	1.39
Mean	74.40	289.43	23.67	6.95	4.09
Average deviation	72.85	271.43	21.39	5.23	2.85
Standard deviation	96.79	419.16	31.79	9.55	3.60

Tab. 6: FFP/MM relations of different software projects

Dabei ergab sich ein relativ hoher durchschnittlicher Wert für die ***Neuentwicklung*** von:

1 Cfsu ≈ 0,32 PM

Mit einem Maximum von 3,1 und einem ***Minimum von 0,08***. Allerdings wurde hierbei eine ältere, ***nicht genau dem ISO-Standard*** entsprechende FFP-Version angewandt.

4.5 COSMIC-FFP Anwendung bei Siemens und Bosch

Die folgenden Anwendungen verdeutlichen nur ein Größenordnung für ausgewählte Automotiv-Software aus so genannten Machbarkeitstudien.

SIEMENS: [Scheikl2000]

"Counting example
Process: **Brake Servo Unit function** (File 1F402601.00A, Simtec 81)

Sub-processes Type	DET's	F F P
ICR	2 (V_PBSU_BAS, IP_PBSU_V_PBSU)	1
ICR	2 (AMP_AD, C_AMP_BRAKE_TOL)	1
ICR	3 (TQI_REQ_SLOW, N_32, IP_MAP_HOME_ESTIM__TQI_N_32)	1
ICR	1 (LV_BLS)	1
ICR	1 (LV_ERR_PBSU)	1
ICR	2 (C_PBSU_DIF_HYS,C_PBSU_DIF_BOL)	1
ICW	1 (PBSU)	1
ICW	1 (PBSU_DIF)	1
ICW	1 (LV_PBSU_AVL)	1
ICW	1 (LV_PBSU_EXC)	1
ICW	1 (LV_BRAKE_REQ_PRS	1
ICW	1 (LV_BRAKE_REQ)	1
	Unadjusted FFP for all sub-processes	**12**

Tab. 7: FFP of the brake servo unit software

DET contribution to UCG: 7 (*LV_BRAKE_REQ, PBSU, PBSU_DIF, LV_PBSU_EXC, LV_PBSU_AVL, LV_BRAKE_REQ_PRS, MAP_HOM_ESTIM*)
DET contribution to RCG: 9 (*C_AMP_BRAKE_TOL, C_PBSU_DIF_BOL, C_PBSU_DIF_HYS, C_PBSU_MAP_HYS, IP_PBSU__V_PBSU_BAS, LDP_V_PBSU_BAS, IP_MAP_HOM_ESTIM__TQI_REQ__N_32, LDP_TQI_REQ_SLOW__MAP_HOM_ESTIM, LDP_N_32__MAP_HOM_ESTIM*)"

BOSCH: [Lother2003]

"Because of the representation style of the information in the beginning it is difficult to extract the needed information for the sub process identification. With a bit of experience the information can be transferred to the following schematic diagram.

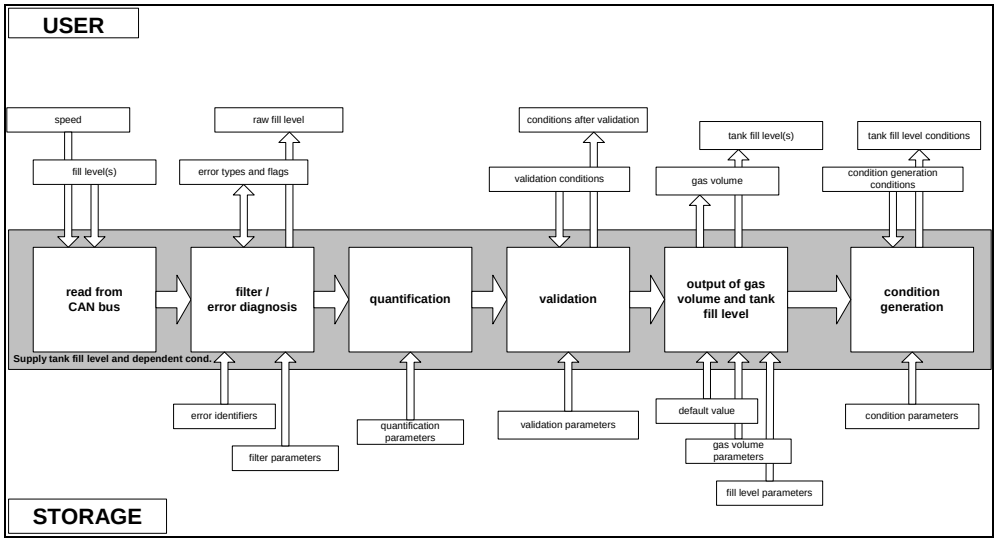


Abb. 10: Typical Schematic Diagram

With help of this figure the functional size measurement can be realized as follows.

Proc.	ID	Process Description	Trigger			
1	**1.1**	**Supply tank fill level and dependent conditions**	**CAN bus event**			
No.	Sub Process Description		Data Group	SP-Type	FFP	Σ
1.1.1	Read fill level(s) from CAN-Bus		DAttr1 DAttr2	E	1	
1.1.2	Read speed from CAN		DAttr3	E	1	
1.1.3	Read filter parameters		DAttr4 DAttr5 DAttr6	R	1	
	...					
1.1.7	Output raw fill level(s)		DAttr7 DAttr8	X	1	
	...					
						19

Tab. 8: FFP measurement of tank fill software “

Diese beiden Bewertungen zeigen zumindest die Größenordnung einzelner (abgeschlossener) Komponenten im Automotiv-Softwaresystembereich, wobei nach obigen empirischen Relationen die *Brake Servo Unit ca. 0,5 PM* während die *Tank Fill Supply etwa 0,8 PM* Entwicklungsaufwand bedeuten würde.

4.6 Die Anwendung der ISBSG für die Aufwandsschätzung nach Schoedon

Spezifikation eines ***Krankenhausmanagementsystems*** mit der folgenden ausgewählten Layer-Bewertung [Schoedon2005]:

	Diagnosen	Prozeduren	ICD-10-Code	ICPM-Code	DRG-Daten	ENTRY (E)	EXIT (X)	READ (R)	WRITE (W)
Diagnosen eingeben	E W					1			1
ICD-10 Code erzeugen			E X R W			1	1	1	1
Prozeduren eingeben		E W				1			1
ICPM-Code erzeugen				E X R W		1	1	1	1
Hauptdiagnose festlegen					E W	1			1
Eingabe der Falldaten					9E 9X 9R	9	9	9	
Eingabe von ICPM-Code (OP)					E X R	1	1	1	
DRG-Daten generieren					28E 28X 28W	28	28		28
Kosten generieren					E X 28R	1	1	28	
Status festlegen					E X R W	1	1	1	1
DRG-Daten ändern					E X R W	1	1	1	1
Kodierplausibilität prüfen					28X 28R		28	28	
DRG-Daten prüfen					29X 29R		29	29	
				Zwischensumme:		46	100	99	35
				Summe:					280

Tab. 9: FFP einer Komponente in einem Krankenhausmanagementsystem

und der Gesamtbewertung über alle Systemfunktionen:

	ENTRY (E)	EXIT (X)	READ (R)	WRITE (W)	Summe
Tabelle 4.1	72	7	13	60	152
Tabelle 4.2	24	35	35	24	118
Tabelle 4.3	71	7	13	59	150
Tabelle 4.4	7	3	3	7	20
Tabelle 4.5	45	41	41	45	172
Tabelle 4.6	57	112	111	56	336
Tabelle 4.7	29	34	33	29	125
Tabelle 4.8	46	100	99	35	280
Tabelle 4.9	1	0	0	1	2
Tabelle 4.10	51	47	47	56	201
Tabelle 4.11	44	44	44	2	134
Tabelle 4.12	25	32	32	29	118
Zwischensummen der Datenbewegungen in Cfsu:					
	472	462	471	403	
Total in Cfsu:					1808

Tab. 10: FFP eines Krankenhausmanagementsystem insgesamt

Eine Auswertung in der ISBSG 2004 gab die folgende Aufwandsschätzung, also

0,002 PM ≤ 1 Cfsu ≤ 0,006 PM bzw.

646 Cfsu ≥ 1 PM ≥ 157 Cfsu

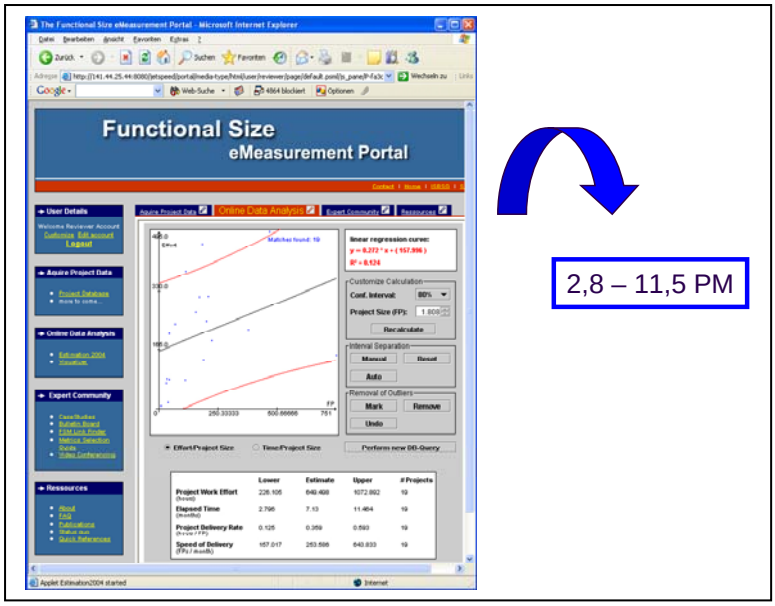


Abb. 11: Aufwandsbestimmung mit der ISBSG 2004

Die ***ISBSG (International Software Benchmarking Standards Group):*** (siehe [Ebert2007] und [Hill 2003])

> "The ISBSG is an International 'not-for-profit' organisation with 13 members. Members are software measurement associations like IFPUG (USA, Brazil etc.), ASMA (Australia), GIFPU (Italy), NESMA (Netherlands), NASSCOM (India), CSPIU (China) and JFPUG (Japan) [ISBSG 2003]. Based on a questionnaire, data is collected from all over the world to fill the benchmarking repositories. Data available is almost completely related to tailor-made software. The "New &

Enhancement Projects" contains data of over 3,000 projects. The repository "Maintenance & Support" comprises 115 applications or programs. The data can be acquired directly by ISBSG or by the members associations. Under construction are repositories for "business systems software package acquisition and implementation". With this benchmark data it is possible to validate performance, estimates and proposals. … The validation with the use of ISBSG benchmarking data is shown in this real-life case. A request for proposal is sent out to the various suppliers. In the table the main project characteristics. The base for the size calculation is provided as part of the information package.

Project size	540 function points
Domain	Business application
Language	Cobol
Platform	Mainframe
Constraints	Duration: 10 months , Cost: 1,000,000 Euro
An average hourly rate of 100 Euro is used.	

Tab. 11: Case characteristics

For a quick assessment of the characteristics, the reality checker is used (included in the ISBSG repository package). The screenshot shows the results based of the matching repository data."

Eine Beispielanwendung zeigt die folgende Abbildung aus [Ebert2007]:

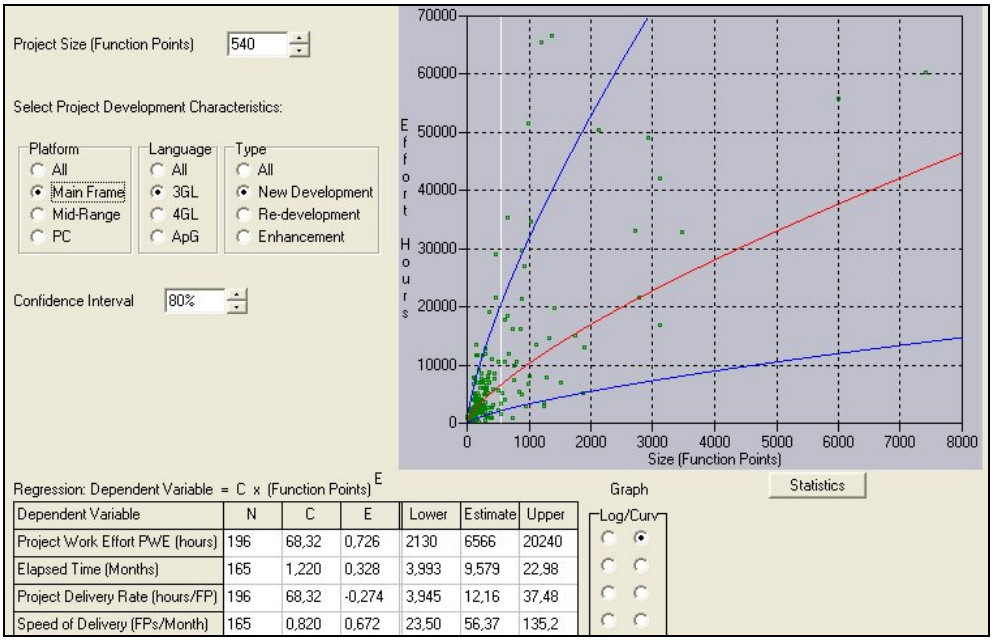


Abb. 12: Aufwandsbestimmung mit der ISBSG 2006

Das Projektrepository der ISBSG kategorisiert die jeweils gespeicherten Projektinformationen auf der Grundlage der

- *Implementationsplattform* (Main Frame, Mi Range und PC),
- *Implementationssprache* (in der allgemeinen Charakterisierung als 3GL, 4GL u.a.),
- *Projekttyps* (bezüglich einer Neuentwicklung, einer Umstellung oder Wartung).

Dazu ist jeweils ein Wert eines Größenmaßes für das entwickelte Softwaresystem angegeben, welches dann den entsprechenden Projektaufwand bestimmen lässt.

5 Methoden zur Aufwandsschätzung auf der Basis von COSMIC-FFP

5.1 Kostentreiber

Für die endgültige Bestimmung des Aufwandes (in Personenmonaten, Projektzeit oder –kosten) werden so genannte Kostentreiber herangezogen. Beim ***COCOMO II*** sind dies beispielsweise [Boehm 2000] die ***Produktattribute***: CPLX: Komplexität des Produktes, DATA: Größe der Datenbasis, DOCU: entwicklungsbegleitende Dokumentation, RCPX: Zuverlässigkeitsanforderungen, RELY: geforderte Zuverlässigkeit, RUSE: Entwicklung für die Wiederverwendung; die ***Hardware-Eigenschaften*:** VOL: Plattformdynamik, STOR: benötigter Speicherplatz, TIME: benötigte Rechenzeit, TURN: Zugangsformen; die ***Personalattribute****:* ACAP: Fähigkeit zur Analyse, APEX: Anwendungserfahrungen, LTEX: Erfahrungen mit der Programmiersprache, PCAP: Programmierkenntnisse, PCON: Personalstabilität, PDIF: Plattformschwierigkeiten, PERS: Personalfähigkeiten, PLEX: Plattformerfahrungen, PREX: Personalerfahrungen, TEAM: Teamstabilität; die ***Projektattribute***: FCIL: Einsatzmöglichkeiten, FLEX: Entwicklungsdynamik, MODP: moderne Programmierpraktiken, PREC: Erstentwicklungsform, RESL: Risikomanagement, PMAT: Prozessgüte nach dem CMM, SITE: Entwicklungsteamverteilung, SCED: erforderliche Entwicklungszeit, TOOL: Anwendung von CASE-Tools. Die Bewertungsvorgabe nach Boehm gibt für diese Kostentreiber Wertintervalle vor, bei die 1,0 als Faktor keinen Einfluss bedeutet und die Intervalle von ***maximal 7,8*** bis ***minimal 0,5*** reichen. Wichtig ist dabei, dass COCOMO II beispielsweise folgende Projektausrichtungen unterscheidet:

- **COCOMO**: für die Aufwandsschätzung „allgemeiner“ Projekt zur Neuentwicklung und zur Wartung.
- **COPSEMO:** für Projekte, die OO-Systeme entwickeln und dabei den RUP-Prozess zugrunde legen,
- **CORADMO:** für die Aufwandsschätzung von Entwicklungsprojekten nach dem Rapid Application Development (RAP) Verfahren,
- **COCOTS:** für die Aufwandsschätzung für Projekte nach der komponentenbasierten Software-Entwicklung,
- **COPROMO:** für Projekte, die vor allem phasenweise Produktivitätsangaben berücksichtigen (können),

Bei der Umfangsschätzung nach dem ***IFPUG Function Pionts*** Verfahren wird bereits bei der „Auszählung“ der Funktionalität eine Kalibrierung vorgenommen [Dreger 1989]. Die allgemeine Berechnungsformel lautet

$$UFP = a \times inputs + b \times outputs + c \times requires + d \times internal\ data + e \times external\ data$$

mit der ***Kalibrierung***:

Merkmal	Wichtungsfaktor		
	einfach	mittel	komplex
Eingaben	3	4	6
Ausgaben	4	5	7
Abfragen	3	4	6
interne Datenbestände	7	10	15
externe Datenbestände	5	7	10

Tab. 12: IFPUG FP Wichtungsfaktoren

und der ***Justierung*** nach

TCF = (Summe der Einflussgrößenbewertung) * 0,01 + 0,65 .

mit den konkreten Einflussgrößen (jeweils ordinal mit „0“ bis „5“ bewertbar)

1. ***Datenkommunikation*** *als Einflussgröße für eine vernetzte Anwendung.*
2. ***Verteilte Anwendung*** *für eine Anwendung, die mit verteilten Daten bzw. mit verteilten Verarbeitungsfunktionen arbeitet.*

3. ***Leistungsanforderungen*** *für eine Anwendung, die besondere Ansprüche im Hinblick auf Zeitverhalten und Datendurchsatz stellt.*
4. ***Konfigurationshandhabbarkeit*** *für eine Anwendung, die für eine besondere Hardware und Systemumgebung realisiert wird.*
5. ***Transaktionsrate*** *für eine Anwendung die aufwendige Geschäftsprozesse gewährleistet.*
6. ***Online-Dateneingabeform*** *für Anwendungen, die Online Dateneingaben zur Verfügung stellen müssen.*
7. ***Benutzeroberflächenart*** *für Anwendungen, die eine Benutzeroberfläche als User Interface enthalten.*
8. ***Online-Änderungsform*** *für eine Anwendung, die eine Pflege des Datenbestandes auch online ermöglicht.*
9. ***Komplexität der Verarbeitung*** *für Anwendungen, deren Abläufe sehr komplexe Arbeitsschritte bzw. Funktionen erfordern.*
10. ***Wiederverwendbarkeit*** *für eine Anwendung, deren Entwicklung auf eine Wiederverwendung des Codes abzielt.*
11. ***Installationsanforderungen*** *für Anwendungen, die eine Installationsbegleitung erfordern bzw. bereitstellen.*
12. ***Operationale Einfachheit*** *für Anwendungen, die benutzerfreundlich in der Bedienung und Anwendung sind.*
13. ***Mehrfachinstallationen*** *für Anwendungen, die in mehreren Einsatzgebieten oder Unternehmen zur Anwendung kommen.*
14. ***Änderungsfreundlichkeit*** *für Anwendungen, die gezielt auf eine leichte Änderung der Anwendung entwickelt werden.*

Die daraus erhaltenen FP können dann beispielsweise nach Bundschuh in folgender Weise in Personenmonate (PM) umgerechnet werden [Bundschuh2000]

$$PM = 0{,}015216\ FP^{1{,}29}$$

Die so genannten Kostentreiber bilden die empirische Grundlage für die Aufwandsschätzung auf der Basis einer Umfangsbewertung. Dabei zeigt sich,

- dass beim COCOMO für verschiedene Prozessformen auch verschiedene Schätzansätze existieren (speziell für die komponentenbasierte Entwicklung (CBSE) das COCOTS usw.), die eine „Aufwandslast“ in den integrativen bzw. systemarchitekturellen Bereich verlagern bei einem deutlich geringeren Aufwand für die Neuentwicklung)

- dass beim IFPUG FP bereits bei der Umfangsbestimmung des zu entwickelnden Systems eine aufwandsbezogene Kalibrierung vorgenommen wird, die sich dann durch weitere Wichtungen auf der Grundlage eher „klassischer“ Softwaresysteme fortsetzt.

Andererseits handelt es sich bei beiden Verfahren um so genanntes ***Black-Box-Estimation.*** Das bedeutet, dass die empirische Grundlage der Berechnungsausprägung

- auf einer unbekannten Anzahl von Projekten beruht,
- der genaue Charakter der zugrunde gelegten Projekte ebenso wenig bekannt bzw. nachvollziehbar ist.

5.2 Prozessmerkmalskausalitäten

Ein weiterer allgemeiner Hintergrund für die Aufwandsschätzung sind die ***Beziehungen zwischen den Kostentreibern*** bzw. Aufwand bestimmenden Prozessmerkmalen. Ausgangspunkt für die Darstellung derartiger Zusammenhänge sind beispielsweise ***semantische Netze***, wie zum Beispiel nach [Laird 2006].

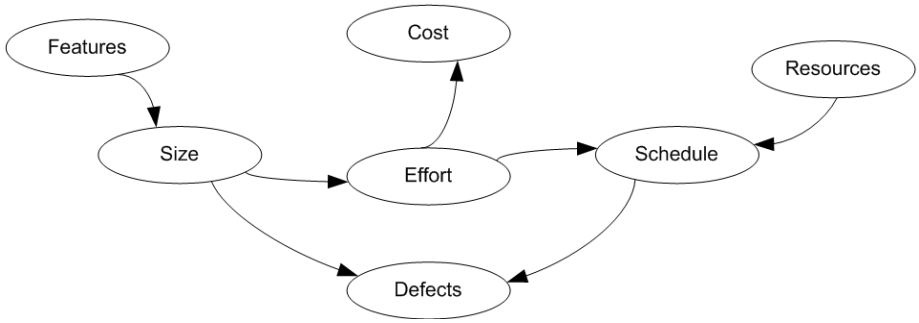


Abb. 13: Semantisches Netz von Prozessmerkmalen

Den Zusammenhang er so genannten *Five Core Metrics* nach [Putnam 2003] zeigt uns die folgende Spezialform eines semantischen Netzes als ***kausales Netz.***

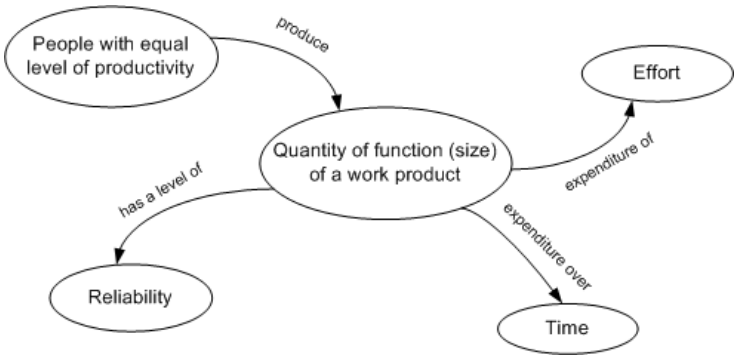


Abb. 14: Kausales Netz zu den Five Core Metrics von Putnam und Myers

Ein spezielles semantisches Netz bei der Anwendung eines konkreten Prozessverbesserungsverfahrens (dem *Personal Software Process* (PSP)) nach [Humphrey 2000] lautet beispielsweise

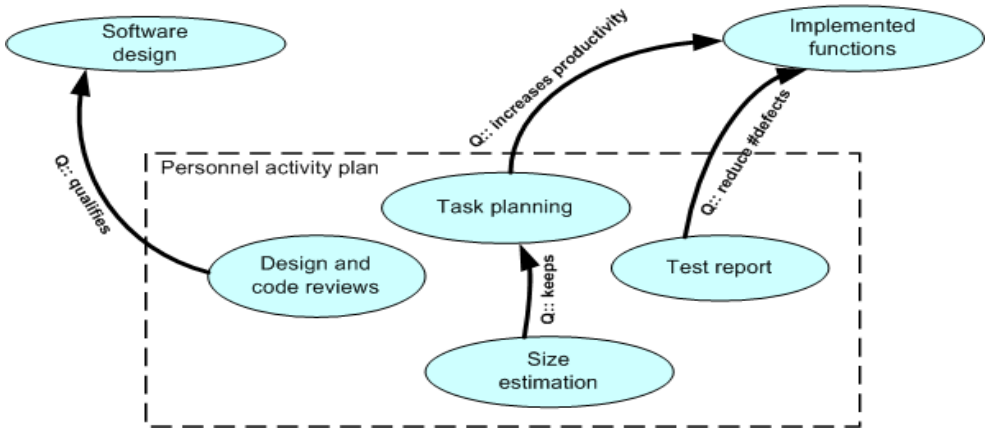


Abb. 15: Kausales Netz für den PSP-Prozess

Eine ganzheitliche Sicht auf den Software-Entwicklungsprozess mit den möglichen Bereichen von Prozessmerkmalsabhängigkeiten verdeutlicht die folgende Abbildung nach [Dumke 2006].

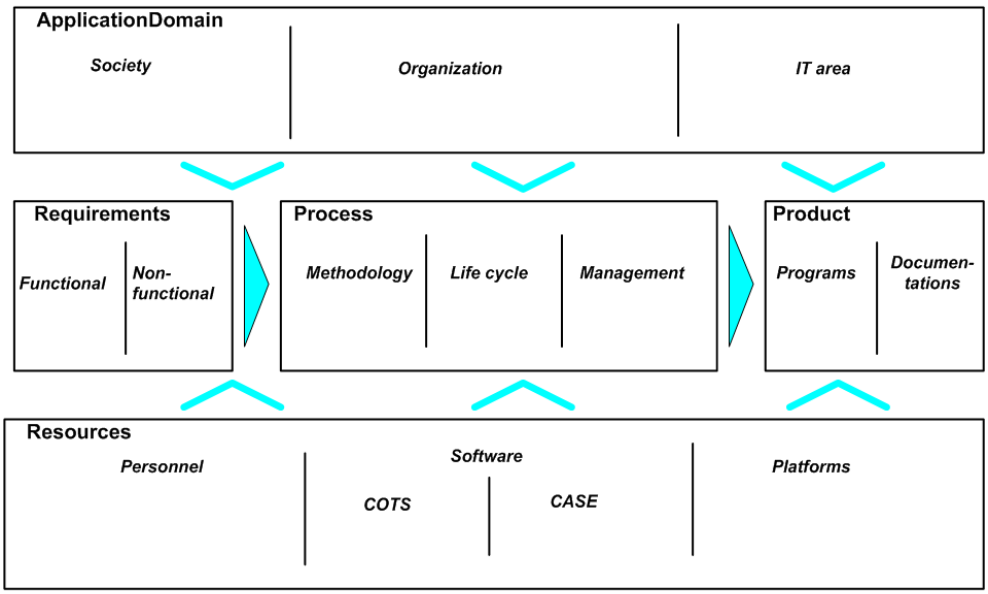


Abb. 16: Ganzheitliche Darstellung der Software-Prozesskomponenten

Beispielsweise hat dabei der Software-Entwicklungsprozess (das mittlere Kästchen in der obigen Abbildung) folgende mögliche Ausprägungen.

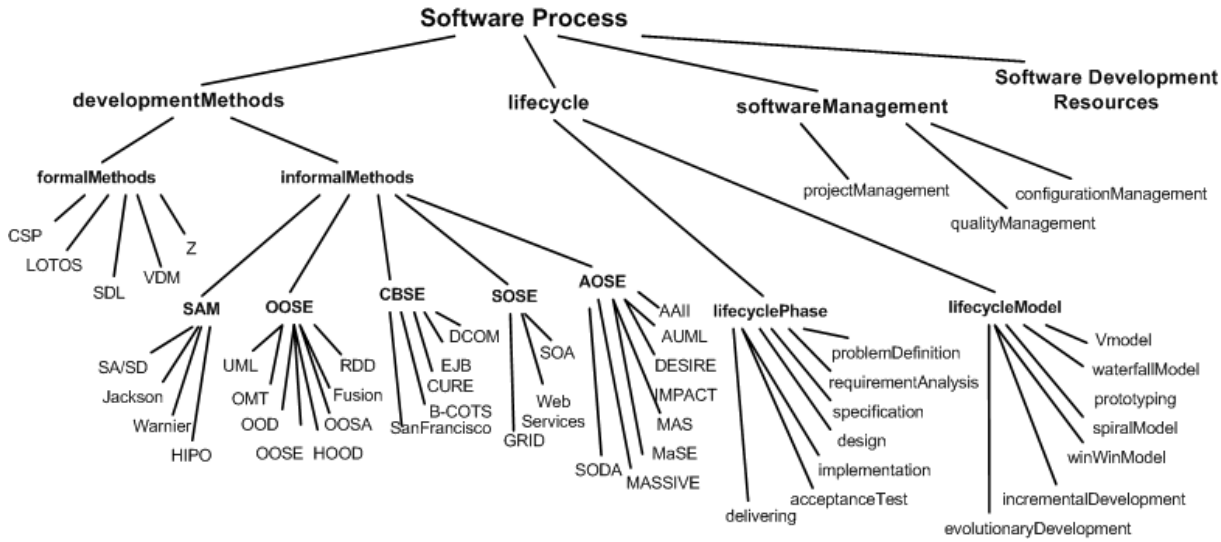


Abb. 17: Mögliche Software-Prozessausprägungen

Analoges gilt für die anderen Bereiche der Produktanforderungen, des Produktes/Systems selbst, der Ressourcen (als Personal, der Software (COTS oder CASE) oder der Hardware) sowie dem Anwendungsbereich.

Die jeweiligen Prozessmerkmale (speziell als Kostentreiber für die Aufwandsschätzung) haben unterschiedliche Indikationen für den Aufwand eines zu entwickelnden Software-Systems. Wendet man aus der Literatur bekannte Kausalitätserfahrungen bezüglich eines semantischen bzw. kausalen Zusammenhangs auf die oben bereits angegebene Gesamtdarstellung an, erhält man beispielsweise hinsichtlich der Implikationen einer Verbesserung (als *Process Improvement Model (PIM)*) [Dumke 2006b].

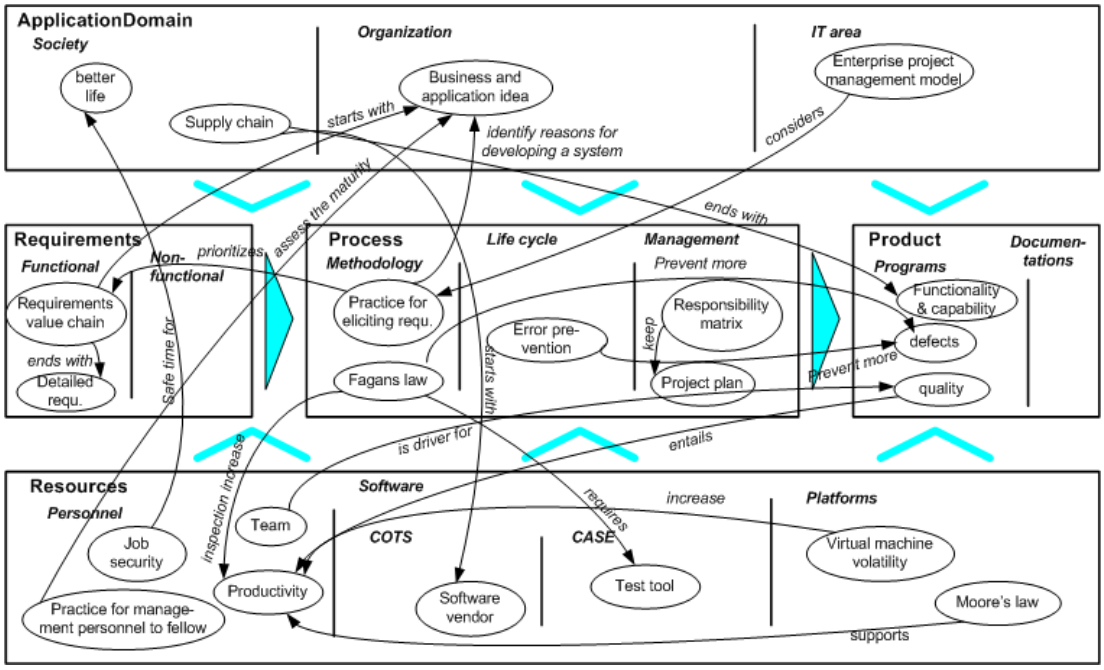


Abb. 18: Ein PIM-basiertes semantisches Netz

Andererseits sind bezüglich der Kostentreiberbeziehungen vor allem Erfahrungen, wie so genannte ***Gesetze, Faustregeln*** oder ***Fallstudien*** bei einer Aufwandsschätzung von Bedeutung [Endres 2003]. Kandt fast eine unterschiedliche *Rangfolge der Kostenteiber* hinsichtlich ihres Einflusses auf die Aufwandsschätzung in folgender Tabelle zusammen [Kandt 2006].

	Boehm's Ranking	**Clark's Ranking**	**Neufelder's Ranking**
1	Personnel/Team	Product complexity	Domain knowledge
2	Product complexity	Analyst capability	Non-programming managers
3	Required reliability	Programmer capability	Use of unit testing tools
4	Timing constraint	Constraint on execution time	Use of supported operating systems
5	Application experience	Personnel continuity	Testing of user documentation
6	Storage constraint	Required reliability	Use of automated tools
7	Modern programming practice	Documentation	Testing during each phase
8	Software tools	Multi-site development	Reviewed requirements
9	Virtual machine volatility	Application experience	Use of automated fracas
10	Virtual machine experience	Platform volatility	Use of simulation

Tab. 13: Rangfolge zwischen den Software-Kostentreibern

[Verzuh 2005] betrachtet die Kosten für das Auftreten von Fehlern im Projektverlauf in der Art:

> *"If a defect caused by incorrect requirements is fixed in the construction of maintenance phase, it can cost 50 to 200 times as much to fix as it would have in the requirements phase."*

> *"Each hour spent on quality assurance activities such as design reviews saves 3 to 10 hours on downstream costs."*

Einen allgemeinen Aufwandsverlauf zeigt die hinreichend bekannte Tabelle von Royce für so genannte konventionelle Projekte in der folgenden Form [Royce 1998].

ACTIVITY	COST
Management	5%
Requirements	5%
Design	10%
Code and unit testing	30%
Integration and test	40%
Deployment	5%
Environment	5%
Total	100%

Tab. 14: Expenditures by activity for a conventional software project

Detaillierter bestimmt Emam diese Aufteilung bezogen auf unterschiedliche Projektarten (siehe folgende Tabelle) [Emam 2005].

Process activity	System project (%)	Commercial project (%)	Military project (%)
Design	21	16	19
Requirements Definition	11	6	13
Project Management	17	16	17
Documentation	10	16	13
Change Management	14	8	15
Coding	27	39	23

Tab. 15: Prozentuale Aufteilung von Prozessaktivitäten in unterschiedlichen Projektarten

Hinsichtlich der Aufwandsschätzerfahrungen, basierend auf der IFPUG FP-Methode, kam Sneed zu der folgenden Verteilung [Sneed 2005].

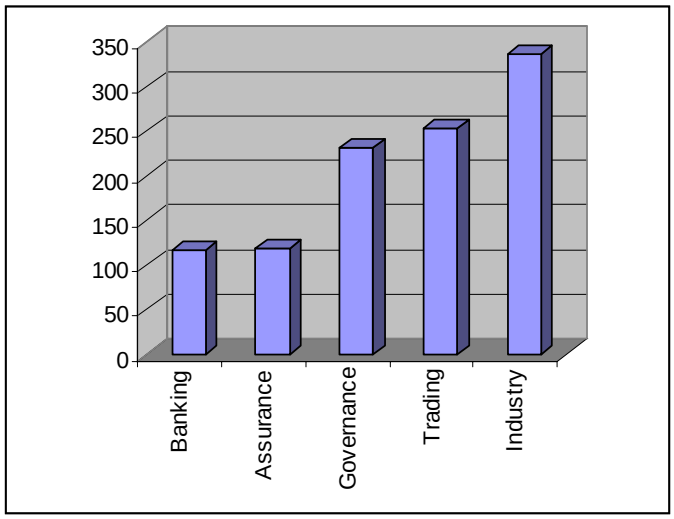


Abb. 19: Function Points per PH in unterschiedlichen IT-Bereichen

Damit ergibt sich für die ***Produktivität der Software-Systementwicklung*** in den unterschiedlichen Bereichen:

1 [Industrie] ≈ 0,75 [Handel] ≈ 0,69 [Öffentlicher Dienst] ≈ 0,35 [Versicherungen] ≈ 0,34 [Banken]

Außerdem unterscheidet Sneed die Aufwandsschätzung für die unterschiedlichen Projektarten hinsichtlich [Sneed 2005]:

Projektart	***Empfohlene Schärtzmehtode***
Protoypprojekte (Beispielentwicklung)	Analogien
Entwicklungsprojekte (Neuentwicklung)	COCOMO I, IFPUG FP, Data Point
Evolutionsprojekte (Weiterentwicklung)	COCOMO II, Data Point, Object Point
Wartungsprojekte (Erneuerung, Verbesserung)	COCOMO I, NASA, Arthur, proprietär
Sanierungsprojekte (Erschließung, Reengineering)	COCOMO II, Soft-Calc
Migrationsprojekte (Umstellung)	COCOMO I, COCOMO II
Integrationsprojekte (Einbettung)	Object Point, COCOMO II
Installationsprojekte (Softwaresystem-Einführung)	Analogien

Tab. 16: Schätzmethoden vs. Projektart nach Sneed

[Emam 2005] analysierte das Vorhandensein von Qualitäts- bzw. Metriken-Initiativen in unterschiedlichen Firmen weltweit und kam zu dem folgenden Ergebnis.

Business Domain	**Existence of a QA Function (%)**	**Existence of a Metrics Program (%)**
Aerospace	52	39
Computer manufacturing	70	42
Distribution	16	-
Finance	57	25
Government	30	5
Health	51	8
Manufacturing	52	25
Oil and gas	40	20
Software	60	36
Telecommunication	54	44
Transportation	33	33
Utility	40	10

Tab. 16: Anteil des jeweiligen Verfahrens in Firmenbereichen weltweit

Eine Anwendung ausgewählter Faustregeln (als *Established Measurement Process (EMP)*) in einem semantischen Netz zeigt die folgende Abbildung [Dumke 2006].

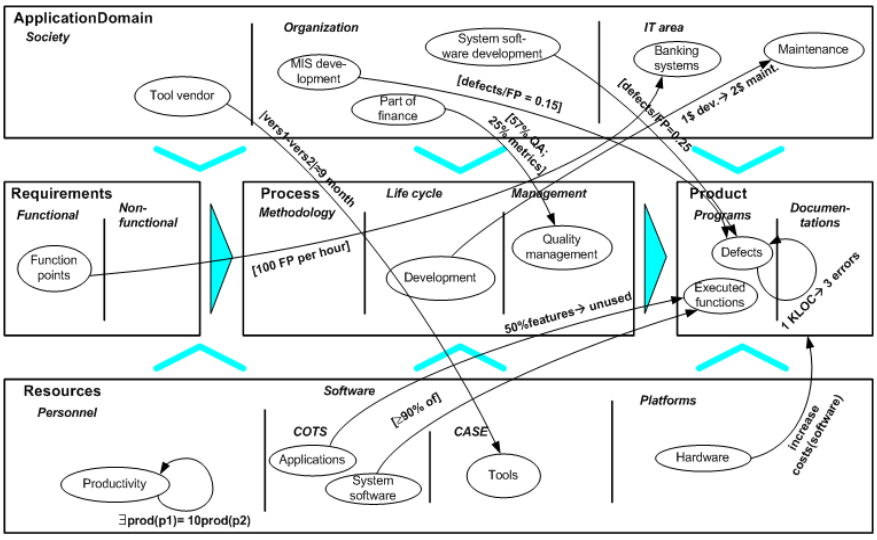


Abb. 20: Ein EMP-basiertes semantisches Netz

Schließlich soll noch auf die Möglichkeit der Anwendung (soweit vorhanden) von ***formalen Zusammenhänge*** (z. B. Metriken) hingewiesen werden. Im folgenden geben wir einfache Beispiele aus der Literatur, zusammengefasst in [Dumke 2006a] (siehe auch [Florac 1999]), an:

"The priorities of each attribute $a_{i,j}$ were executed as an approximation by computing p_i. Another formula by Kandt helps to evaluate the SQA situation as

Requirements coverage = (*Number of Requirements traced to functional test cases*)/
(Number of requirements)

System architecture statement coverage = (*Executed SLOC of system architecture*)/
(Total SLOC of system architecture)

System architecture edge coverage = (Executed decision outcomes of system architecture)/
(Total decision outcomes of system architecture)

System Statement coverage = (Executed SLOC of system)/(Total SLOC of system)

System edge coverage = (Executed decision outcomes of system)/
(Total decision outcomes of system)

Otherwise, the defect estimation techniques are summarized by Kandt in the following manner [Kandt 2006]

$$D_1 = (l \times d) - D_d$$

where D_1 stands for the number of remaining defects in a module, l is the number code lines, d is the typical number of defects per source line of code, and D_d is the number of detected defects.

$$D_2 = ((N_1 + N_2)log(n_1 + n_2))/3000 - D_d$$

as an estimation based on the Halstead's software science. Finally as a capture-recapture technique for defect estimation the formula

$$D_3 = (m_1 \times m_2)/(m_{12} - (m_1 + m_2 - m_{12})$$

where m_1 and m_2 are the number of defects found in these research groups and m_{12} denotes the common defects found in both groups."

"The ***customer cost*** of a software product was executed by Emam [Emam 2005] in the following manner.

$$Customer\ Cost = Defect\_density \times KLOC \times Cost\_per\_defect \times Defects\_find\_by\_customer$$

"The general relationship between different indicators of quality, quantity, effort and productivity are defined by Sneed in the following manner [Sneed 2005]:

1. $quantity = (productivity \times effort) / quality$
2. $quality = (productivity \times effort) / quantity$
3. $productivity = (quantity \times quality) / effort$
4. $effort = (quantity \times quality) / productivity$ "

"SLIM: Considering the *Software Life Cycle Management* (SLIM) Putnam adapted the Rayleigh curve for the software development area in the following manner [Putnam 1992]

$$Current\_effort = (Total\_effort/duration) \times t \times e^{(-t \times t/2 \times duration)}$$

where *duration* stands for the square of total duration of the development and *t* means the time point of evaluation. The current effort was measured in *personal years.* Another kind of estimation based on the Rayleigh formula is known as ***software equation*** (see also [Keyes 2003]) as

$$System\_size = technology\_constant \times Total\_effort^{1/3} \times duration^{4/3}$$

where the *technology_constant* depends on the development methodology."

Ordnet man diese und andere Metriken den jeweiligen Prozessbereichen zu, so erhält man schließlich eine Darstellung ausgewählter Metriken, Maße oder funktionaler Zusammenhänge (als *Software Measurement Process (SMP)*) in der folgenden Abbildung [Dumke 2006].

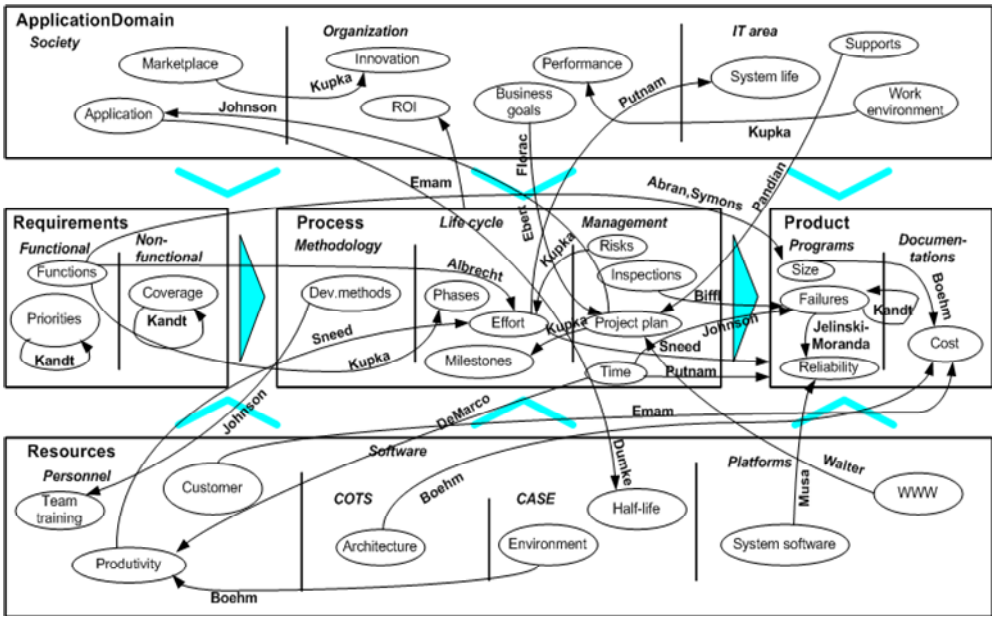


Abb. 21: Ein SMP-basiertes semantisches Netz

Berücksichtigt man die *Beziehungen zwischen den Kostentreibern* für die Aufwandsschätzung, so zeigt sich dabei, dass

- die unterschiedlichen Prozessmerkmale sich mehr oder weniger beeinflussen bzw. aufgrund von Faustregeln, Fallstudien oder auch Metriken unmittelbar zusammenhängen und somit den Kosten tragenden Anteil direkt oder indirekt erhöhen bzw. ***senken*** können,
- die Aufwandsanteile aber auch Produktivitäten für unterschiedliche Anwendungsbereiche sehr stark differieren können; bei Softwaresystemen für den Öffentlichen Dienst ist diese ca. ***doppelt so hoch*** wie im Bankwesen,
- der Anteil an qualitäts- bzw. metrikenbasiertem Projektmanagement international durchschnittlich kaum 50% beträgt, derartige Maßnahmen jedoch den Aufwand insgesamt ***drastisch senken*** können.

5.3 Zweckmäßige empirische Adaption der Cfsu zur Aufwandsbestimmung

Im Folgenden soll nun eine sinnvolle Auswahl einer Aufwandsschätzung nach der COSMIC-FFP-Methode diskutiert werden.

5.3.1 FFP-basierte Aufwandsschätzung mittels COCOMO II

Hierbei wird der Umfang, der im COCOMO II die Schätzgrundlage bildet, mit der COSMIC FFP bestimmt und dann die COCOMO II insgesamt angewendet. Die Grundformeln lauten [Boehm 2000] für den ***Personalaufwand PM*** für die Produktentwicklung:

$$PM = A \times Size^{E} \prod_{i=1}^{n} EM_i + PM_{Auto}$$

Während A einen jeweils speziell „eingestellten“ Wert als *kalibrierbaren Aufwandskoeffizienten* darstellt, ergibt sich E aus der folgenden Berechnung:

$$E = B + 0{,}01 \times \sum_{j=1}^{5} SF_j$$

B ist dabei wiederum ein *kalibrierbarer Wert*, den wir weiter unten noch konkret angeben werden, und die SF_j sind die *Skalierungsfaktoren* PREC, FLEX, RESL, TEAM und PMAT (siehe 5.1).

Die EM_i in der obigen Formel sind die *Aufwandsmultiplikatoren* und zwar RCPX, RUSE, PDIV, PERS, PREX, FCIL und SCED für die Produktentwicklung (s. o.) und die restlichen oben aufgeführten für die Produktwartung. Schließlich beinhaltet die obige Formel noch ein PM_{Auto}, welches den Aufwand in Personenmonaten für die automatische Codeerschließung und wie folgt berechnet wird:

$$PM_{Auto} = (adaptedSLOC \times (AT/100))/ATPROD\ ,$$

wobei *SLOC* für die Quellcodezeilenanzahl, *adaptedSLOC* für Codezeilen, die zu übernehmen bzw. nur anzupassen sind, *AT* für den Prozentsatz der adaptedSLOC, die durch ein Reengineering automatisch „erschlossen“ worden sind, *ATPROD* die Produktivität der automatischen Erschließung bzw. Translation von Quellcodezeilen.

Das COCOMO II.2000 ermöglicht nach dieser Aufwandsschätzung auch die ***Abschätzung der Entwicklungszeit TDEV***. Die Berechnungsformel dafür lautet:

$$TDEV = (C \times (PM_{NS})^{D + 0.2 \times (E - B)}) \times (SCED\% / 100)\ .$$

Dabei stellt B den so genannten Basisexponenten für die Skalierung und C, D und E die kalibrierbaren Entwicklungszeitexponenten dar. Beim *SCED%* ist der erforderliche Prozentsatz an Entwicklungszeitverkürzung im Verhältnis zur normalen bzw. üblichen Entwicklungszeit anzugeben. Als PM_{NS} sind die oben berechneten bzw. geschätzten Personenmonate ohne den SCED-Kostentreiber und ohne PM_{Auto} einzusetzen. Das *TDEV* berechnet hierbei die Entwicklungszeit in Kalendermonaten. Für eine „übliche“ Entwicklungszeitbestimmung werden die Werte C = 3,67 und D = 0,28 vorgeschlagen.

Die COCOMO II Methode motiviert also bei der Verwendung des Produktumfangs auch bzw. vor allem auf die Function Point Methode, d.h. die FP anstelle einer LOC-Version zu verwenden. Alle anderen Relationen und Justierungen zur Aufwandsschätzung basieren dann auf formelle bzw. Kostentreiber-Eigenschaften untersuchter Projekte, über die keine weiteren detaillierten Informationen zur Verfügung stehen. In diesem Sinne qualifiziert die FP-Verwendung zwar das COCOMO-Verfahren, hebt allerdings die grundlegende Eigenschaft des COCOMO II als ***Black Box Estimation*** nicht auf [Abran 2007].

5.3.2 FFP-Benchmarking auf der Grundlage der ISBSG

Die ISBSG *(International Software Benchmarking Standards Group)* enthält nach [Ebert 2007] ca. 3000 Projekte und darunter einige hundert Projekte, die bereits mit COSMIC FFP bewertet wurden. Hauptsächlicher Schätzumfang liegt jedoch in der IFPUG FP vor, darüber hinaus sind auch Projektaufwände in *Mark II* und *NESMA FP* angegeben [Hill 2003]. Gegenwärtig gehören zu den ISBSG-Mitgliedern die ASMA (Australien), die DASMA (Deutschland), die FISMA (Finnland), die NASSCOM (Indien), die GUFPI (Italien), die JFPUG (Japan), die NESMA (Niederlanden), die AEMES (Spanien), die SWISMA (Schweiz), die UKSMA (England) sowie die IFPUG (USA).

Das ISBSG-Repository erfasst pro Projekt folgenden Hauptmerkmale:

- Projektart (Neu, Weiterentwicklung usw.),
- Entwicklungsplattform (CASE, Technologie),
- Projektumfang (i. a. in Function Points verschiedener Berechnungsmethoden),
- Primäre Programmiersprache,
- Business-Bereich bzw. Kategorie.

Eine Auswertung unterschiedlicher Versionen bzw. Projektbereiche ergab die folgenden Schätzintervalle bzw. -wertebereiche:

ISBSG 2004 gab die folgende Aufwandsschätzung (enthält ca. 20 FFP Schätzungen) [Schoedon 2005]:

$$0{,}002\ PM \leq 1\ Cfsu \leq 0{,}006\ PM \quad \text{bzw.}$$

$$646\ Cfsu \geq 1\ PM \geq 157\ Cfsu$$

Bei der **ISBSG 2006** lauten diese Aufwandsschätzverhältnisse (auf der Grundlage der durchschnittlichen IFPUG FP und der entsprechenden Umrechnung nach Abran) speziell für Projekte für den ***Öffentlichen Dienst*** ([Report 2006], S. 11):

„Government projects have a median speed of delivery rate of 38.2 function points delivered per month."

also $1\ Cfsu_{\ddot{O}ffentlicherDienst} \approx 0{,}023\ PM$

Speziell bei den Schätzungen für Erweiterungen bzw. für die Wartung (korrektive, preventive, adaptive und perfektive zusammengefasst) ergab sich folgende Aufwandszahl für eine Teamgröße von *12 Personen* und einer Projektgröße von 3410 FP [Report 2005]:

$$1\ FP_{Wartung} \approx 0{,}015\ PM$$

also nach Abran $1\ Cfsu_{Wartung} \approx 0{,}013\ PM$

Für die Projektdauer kann auf der Basis der ISBSG die folgende Umrechnung nach Bourque verwendet werden [Bourque 2007]

- für Mainframe Plattformen: $D = 0{,}458 * Aufwand^{0{,}366}$
- für Mid-Range Computer: $D = 0{,}548 * Aufwand^{0{,}360}$
- für PCs: $D = 1{,}936 * Aufwand^{0{,}201}$

Damit kann auf der Basis dieses internationalen Projekt-Repositories für die verschiedensten Projektarten und Projektgrößen eine Aufwandsschätzung auf der Basis ermittelter Cfsu vorgenommen werden.

5.3.3 FFP-Benchmarking auf der Grundlage von Referenzprojekten

Das Benchmarking mit speziellen ***Referenzprojekten*** bzw. den damit zusammenhängenden Transformationsregeln führt zu folgender Vorgehensweise:

- Anwendung der ***Umrechnungsformel*** je nach Projektart, also

$$0{,}04\ PM \leq 1\ Cfsu_{Neuentwicklung} \leq 0{,}1\ PM$$

durchschnittlich

$$1\ Cfsu_{Neuentwicklung} \approx 0{,}07\ PM$$

und

$$0{,}005\ PM \leq 1\ Cfsu_{Wartung} \leq 0{,}048\ PM$$

oder nach ISBSG [Report 2005]:

$$1\ Cfsu_{Wartung} \approx 0{,}013\ PM$$

also durchschnittlich

$$1\ Cfsu_{Wartung} \approx 0{,}02\ PM$$

- Berücksichtigung ***möglicher Umrechnungen*** aufgrund anderer Realisierungs- bzw. Anwendungsbereiche, wie zum Beispiel

1 [Industrie] ≈ 0,75 [Handel] ≈ 0,69 [Öffentlicher Dienst] ≈ 0,35 [Versicherungen] ≈ 0,34 [Banken]

- Gewährleistung ***möglicher Aufwandssenkungen*** aufgrund besonderer Erfahrungen des Teams mit dem Produkt, dem Prozess und den Ressourcen sowie durch die Anwendung grundlegender, qualitätssichernder Entwicklungsmaßnahmen.

- Berücksichtigung einer ***Erprobungszeit*** bzw. -aufwandes, der sich nach [Sneed 2005], S. 93, wie folgt abschätzen lässt:

$$1\ Cfsu_{Erprobung} \approx 1/5\ Cfsu_{Neuentwicklung} \approx 0{,}014\ PM$$

- Berücksichtigung einer ***Regressionstestzeit*** auf der Basis der Erfahrungen in der Praxis nach [Koppenberg 2004], die sich wie folgt abschätzen lässt:

$$1\ Cfsu_{Regressionstest} \approx 0{,}005\ PM$$

- Abschätzung der ***Projektdauer*** auf der Basis der ISBSG nach Bourque [Bourque 2007], der zwischen einer Mid-Range Architektur und einer PC-Plattform liegt, in der Form:

$$0{,}548 * Aufwand^{0{,}360} \leq D \leq 1{,}936 * Aufwand^{0{,}201}$$

Damit ist eine Mischform aus explizitem Referenzprojektvergleich und einer Analogie auf der Basis durchschnittlicher Projekterfahrungen gegeben, die im betrachteten Kontext eine qualifizierte Schätzung möglicher *(machbarer*) Projektentwicklungszeiten ermöglicht.

6 Literatur

[Abran 2005] Alain Abran, Jean-Marc Desharnais, Fatima Aziz: *Measurement Convertibility - From Function Points to COSMIC-FFP.*Poc. of the IWSM, Montreal, Sept, 2005, pp. 227-240

[Abran 2007] Abran, A.; Ndiaye, I, Bourque, P.: *Evaluation of a Black-box Estimation Tool: A Case Study.* Software Process Improvement and Practice, 12(2007), pp. 199-218

[Boehm 2000] Boehm, B. W.: *Software Cost Estimation with COCOMO II.* Prentice Hall, 2000

[Bourque 2007] Bourque, P.; Oligny, S.; Abran, A.; Fournier, B.: *Developing Project Duration Models in Software Engineering.* Journal of Computer Science and Technology, 22(3), pp. 348-357

[Bundschuh 2000] Bundschuh M.: *Aufwandschätzung von IT-Projekten,* MITP Publ., Bonn, 2000

[COSMIC 03] COSMIC-FFP: *Measurement Manual – The COSMIC Implementation Guide For ISO/IEC 19761:2003*. http://www.cosmicon.com, 2003

[Dreger 1989] Dreger, J. B.: *Function Point Analysis.* Prentice Hall, 1989

[Dumke 2006b] Dumke, R.; R.; Blazey, M.; Hegewald, H.; Reitz, D.; Richter, K.: *Causalities in Software Process Measurement and Improvement.* Proc. of the MENSURA 2006, Nov. 6-8, 2006, Cardiz, Spain, pp.42-52

[Dumke 2006a] Dumke, R.; Braungarten, R.; Blazey, M.; Hegewald, H.; Reitz, D.; Richter, K.: *Software Process Measurement and Control – A Measurement-Based Point of View of Software Processes*. Preprint No 11, University of Magdeburg, 2006 (http://ivs.cs.uni-magdeburg.de/sw-eng/agruppe/froschung/Preprints.shtml)

[Dumke 2006] Dumke, R.; Braungarten, R.; Blazey, M.; Hegewald, H.; Reitz, D.; Richter, K.: *Structuring Software Process Metrics - A holistic semantic network based overview.* Proc. of the IWSM 2006, Potsdam, Nov. 2006, pp. 483-498

[Ebert 2007] Ebert/Dumke: *Software Measurement – Establish, Extract, Evaluate, Execute*. Springer-Verlag, 2007

[Efe 2006] Pinar Efe, Onur Demirors, Cigdem Gencel: *Mapping Concepts of Functional Size Measurement Methods. Proc. of the IWSM/Metrikon 2006, Potsdam, Nov. 2006, pp. 343-358*

[Emam 2005] Eman, K. E.: *The ROI from Software Quality.* Auerbach Publ., 2005

[Endres 2003] Endres, A.; Rombach, D.: *A Handbook of Software and System Engineering*. Pearson Education Limited, 2003

[Florac 1999] Florac, W. A.; Carleton, A. D.: *Measuring the Software Process – Statistical Process Control for Software Process Improvement.* Pearson Education, 1999

[Hill 2003] Hill, P. R.: *Software Project Estimation – A Workbook for Macro-Estimaton of Software Development Effort and Duration.* Kwik Kopy Printing, Australien, 2003

[Humphrey 2000] Humphrey, W. S.: *The Personal Software Process: Status and Trends*. IEEE Software, Nov/Dec. 2000, pp. 71-75

[ISBSG 2003] Software Project Estimation – *A Workbook for Macro-Estimation of Software Developmet Effort and Duration.* Melbourne, 2003

[Kandt 2006] Kandt, R. K.: *Software Engineering Quality Practices.* Auerbach Publications, Boca Raton New York, 2006

[Koppenberg 2004] Tom Koppenberg, Ton Dekkers*: Estimating maintenance projects using COSMIC-FFP.* Proc. of the IWSM/Metrikon 2004, Berlin, Nov. 204, pp. 165-174

[Laird 2006] Laird, L. M; Brennan, M. C.: *Software Measurement and Estimation: A Practical Approach.* John Wiley & Sons Publ., 2006

[Levesque 2001] Levesque/Bevo: Measuring Size for the Development of a Cost Model: A Comparison of Results Based on COSMIC FFP and SLIM Back-Firing Function Points. Proc of the IWSM 2001, Montreal, August 2001, pp. 284-299

[Lother 2003] Mathias Lother, Reiner R. Dumke, Thomas Böhm, Horst Herweg, Willy Reiss: *Applicability of COSMIC Full Function Points for BOSCH specifications.* Prof. of the IWSM 2003, Montreal, Sept. 2003, pp. 204-217

[Putnam 2003] Putnam, L. H.; Myers, W.: *Five Core Metrics – The Intelligence Behind Successful Software Management.* Dorset House Publishing, New York, 2003

[Putnam 1992] Putnam, L. H.; Myers, W.: *Measures for Excellence – Reliable Software in Time, within Budgets.* Yourdon Press Publ., 1992

[Report 2005] *Application Software Maintenance and Support – An initial analysis of new data.* ISGSB Report, 2005

[Report 2006] *Software Development Projects in Government – performance practice and predictions.* ISGSB Report 2006

[Royce 1998] Royce, W.: *Software Project Management.* Addison-Wesley, 1998

[Schoedon 2005] Schoedon, M.: *Krankenhausinformationsysteme – Überblick und UML 2.0-basierte Modellierung.* Diplomarbeit, Universität Magdeburg, Fakultät für Informatik, 2005

[Schweikl 2000] Ulrich Schweikl, Stefan Weber, Erik Foltin, Reiner Dumke: *Applicability of Full Function Points at Siemens AT.* Proc. of the IWSM 1999, Regensburg, Oktober 1999, Gabler-Verlag, 2000, S. 171-182

[Sneed 2005] Sneed, H.: *Software-Projektkalkulation.* Hanser Publ., 2005

[Stambollian 2006] Stambollian, A.; Abran, A.: *Survey of utomation Tools Supporting COSMIC FFP – ISO 19761.* Porc. Of the IWSM/Metrkon 2006, Potsdam, November 2006, pp. 435-454

[Szalowski 2005] Szalowski, S.: *Konzeption und Implementation eines Tutorials für die COSMIC-FFP Methode.* Diplomarbeit, Universität Magdeburg, Fakultät für Informatik, 2005

[Verzuh 2005] Verzuh, E.: *The Fast Forward MBA in Project Management.* John Wiley & Sons, 2005