BEDAG INFORMATIK

OBJECT-ORIENTED SOFTWARE COST ESTIMATION

December 1999

Dr. Simon Moser

moser@acm.org

Topics:

The Importance of Measurements & Estimates

A Measurement-Based Estimation Process

Software Models (Meta-Models)

Software Metrics

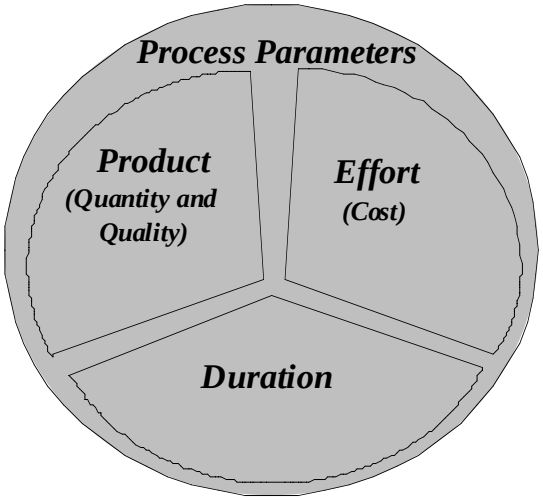
Results of a Field Study

An Example

Future Work

The Importance of Measurements & Estimates (1/2)

3 Process Parameters to Control:



Measurement = Knowing where you stand

Prerequisite for:

- Generic problem solving
- Process improvement / Quality management

The Importance of Measurements & Estimates (2/2)

Estimate = The expectations of a project

Under-estimates:
time pressure ⇨ stress ⇨ frustration ⇨ people turnover
too tight budget ⇨ save on functionality and quality ⇨ "maintenance dilemma"
no more money ⇨ late project cancelation

Over-estimates:
time for fancy stuff ⇨ the over-estimate will turn into an under-estimate

Estimation evaluation criteria:
(1) Accuracy
(2) Cost and speed of the overall estimation process

A Measurement-Based Estimation Process (1/3)

[People throwing darts to a calender, the date hit will be the estimated deadline...]

A non-measurement-based estimation process!

A Measurement-Based Estimation Process (2/3)

A non-software example: estimating the duration of a bush-walk

(**A**) Measure the walk distance on a map
(**B**) Derive a first duration according to some rule-of-thumb
(**C**) Interpret this estimate to specifics (restaurants on the way, ...)

= measurement-based estimation

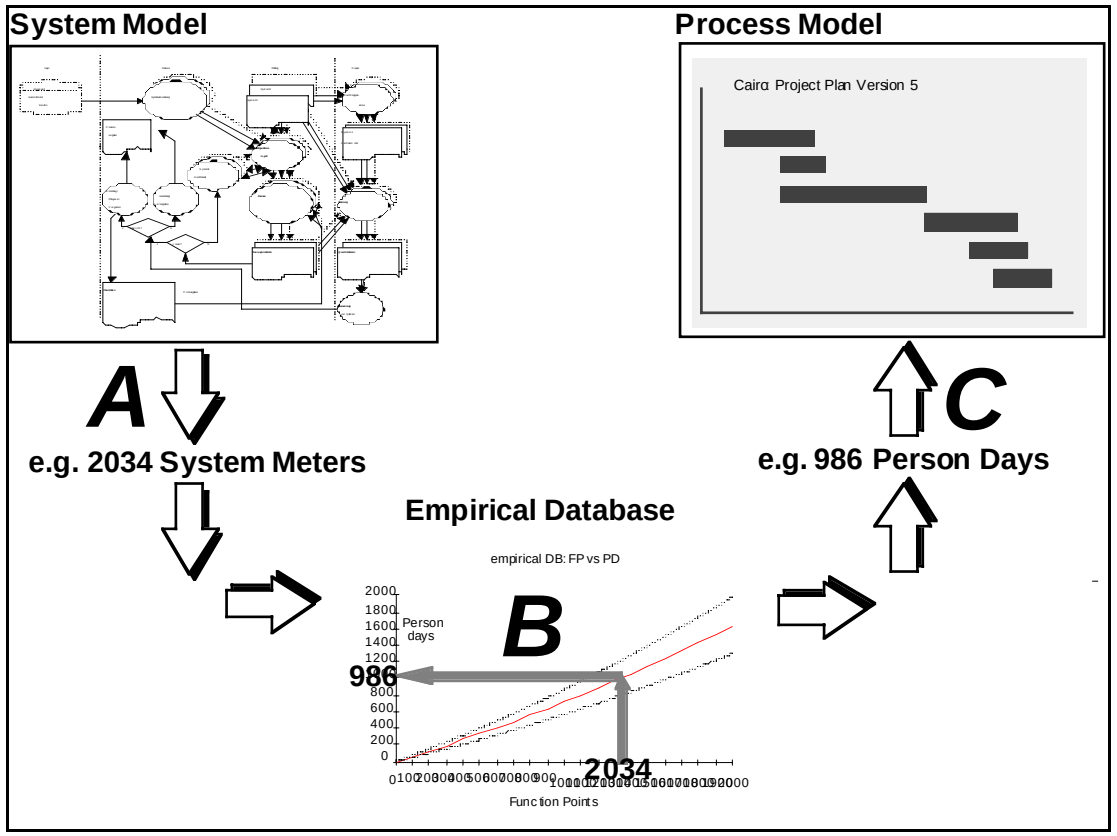
Improvements with respect to accuracy:

- more detailed map or model
- better metric (e.g. taking height differences into account)
- specific empirical database instead of general rule-of-thumb

Improvements with respect to cost of estimation:

- lower-resolution map

A Measurement-Based Estimation Process(3/3)



(adapted from T. DeMarco, 1982 [1])

Software Process Models (1/2)

[The "standard" software process is like biking on gravel roads in the montains, encountering lots of detours and ... wasting lots of money (it should **not** be like this!)]

What is the standard software process?

Software Process Models (2/2)

Process Standardisation through Artefact Standards and Process Completeness Percentages ([2])
(per Artefact release state: draft% - validated% - final&maintained%)

	Artefact	***Process Completeness Percentage***
1	Requirements (=Analysis)	10% - 20% - 35%
2	Design	4% - 9% - 11%
3	Test-suites	3% - 6% - 8%
4	Code	10% - 25% - 35%
5	Documentation	1% - 3% - 5%
6	Installation/Acceptance	2% - 4% - 6%
...	[optional/repeated artefacts]	[additional %]

+ supporting artefacts:

a) Project Management results (plans, reports, ...) - 4% - 6% - 10%

b) Quality Management results (risk analysis, quality plan, ...) - 3% - 5% - 8%

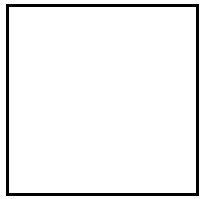
Software Models (Meta Models) (1/4)

[Nobody agrees on what systems or system models are...]

When we want to measure a system (model), the first question is:
What is a system (model)?

Software Models (Meta Models) (2/4)

Layers of Software Product and Process (Software-Life-Cycle):



Most relevant for estimation:
Analysis models = Requirements models

Software Models (Meta Models) (3/4)

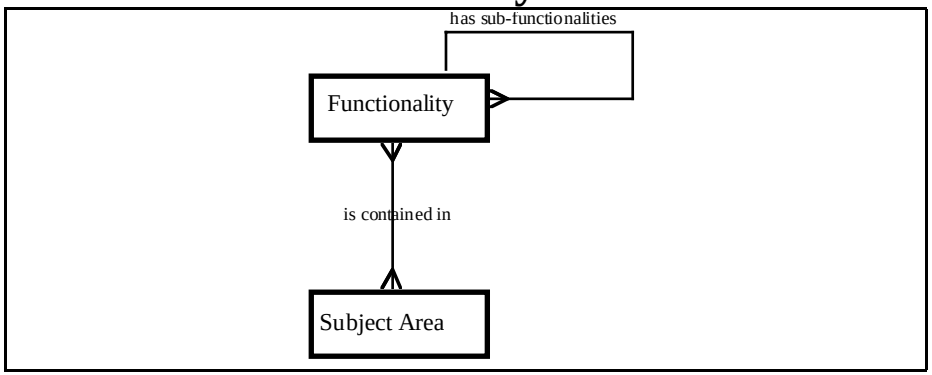
Further sub-layering of the analysis layer

(1) Preliminary Analysis = coarse real-world system modelling

(2) Domain/Business Analysis = detailed real-world system modelling

(3) Application Analysis = user view of the computer system (=user manual)

Preliminary Model:



Functionality "name" [complexity number] **.**
Functionality "name" = { 'sub-functionality', } **.**
Subject Area "name" number of classes **.**
Subject Area 'name' **contains** 'functionality' **, ...**

Software Models (Meta Models) (4/4)

A Domain Model - Metamodel (compliant with most standards [4], [5]):

(1) Domain Analysis Class Model

Domain Class <ddd> { **isSubTypeOf** <base-class> } { **contains** <n> **attr** < class model object>} **.**
Domain Association <assoc-name> **one**|**many** <class1> **to one**|**many** <class2> **.**
Function Type <ddd> [**ofKind** <parameter-name> **,** ...] **.**
Consistency Rule <rrr> **=** < class model object > **,** ... **.**

(2) Use Case Model

Use Case <rrr> **isTriggeredBy** <event/time indication> [**=** <class model object > **,** ...] **.**
Signal [<sss>] **of** <use case / function type> [from|to <actor>] = <class model object > **,** ... **.**
Domain Subsystem <dss> = <use case> **,** ... **.**

(3) State Transition Model

State <st> **isSubStateOf** <state/class> [= < class model object > **,** ...] **.**
Transition <tr> **startsAt** <state1> **endsAt** <state2> [= < class model object > **,** ...] {**triggers**|**isTriggeredBy** <signal>}**.**

Software Metrics (1/2)

Function Point (Allan J. Albrecht, IFPUG [3]):

(1) Classifying the domain classes into easy-medium-complex and giving „points“
(2) Analogue procedure for rating the persistency-accesses to classes in the use cases
(3) Sum of all points = "unadjusted" Function Points
(4) Rating of 10 influence factors with percent point
(5) Adjustment (70%-130%) of the "unadjusted" Function Points = "adjusted" Function Points

Advantages:

- understandable / „intuitive“
- useful for database applications

Disadvantages:

- restricted to database applications
- requires the business model (modelling effort high)
- does not take reuse into account
- needs expert assessment (no fully automatable measurement)
- formally unsound

Software Metrics (2/2)

New approach: System Meter (Moser, 1995 cited in [6])

(1) External complexity of a single model entity:
= #new tokens in the name (+ 1, if old tokens are contained in the name)
= 1, if object is anonymous
[z.B. "theCurrentWindow" = 3; "theNewWindow" = 2; "theNewCurrentWindow" = 1]

(2) Internal complexity of a single model entity
= Sum of the external complexity of those other model entities that define the one in focus
[z.B. a class is defined through its super-classes and „members“,
a method through its parameters and implementing „messages“]

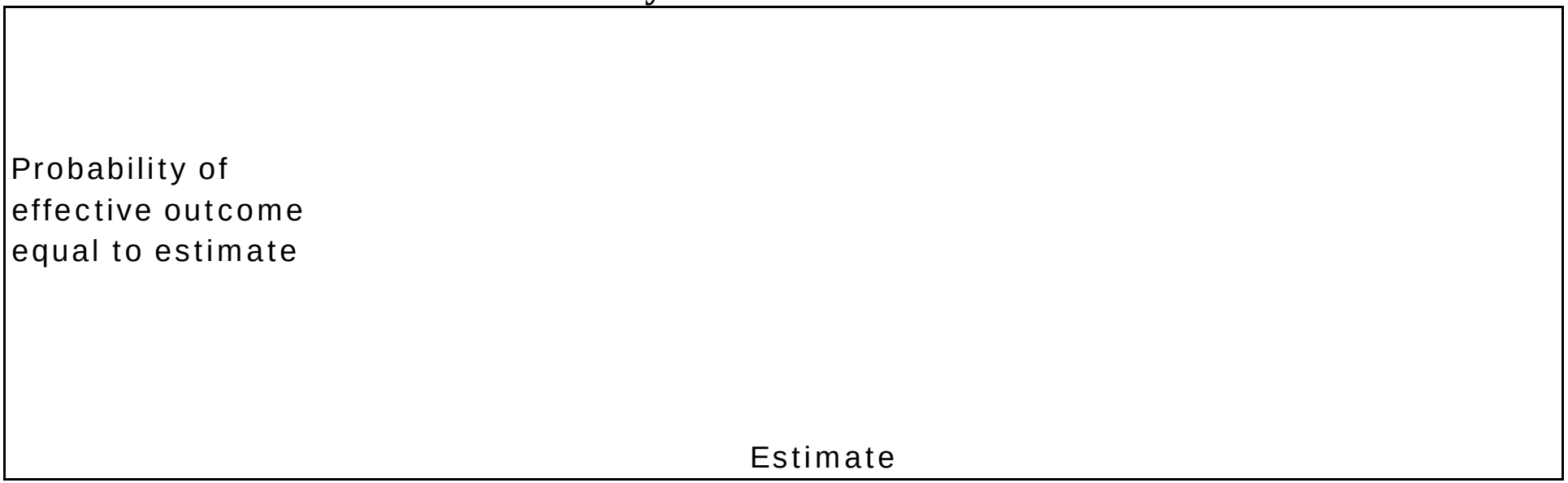
(3) Sum up all complexities (just the external complexity for reusable objects)

Advantages:

- generic (also non-persistency features are counted)
- takes reuse into account
- can be applied on preliminary models, business models as well as code
- measurement is fully automated and objective

Results of a Field Study (1/4)

Main analysis: Effort estimation bias

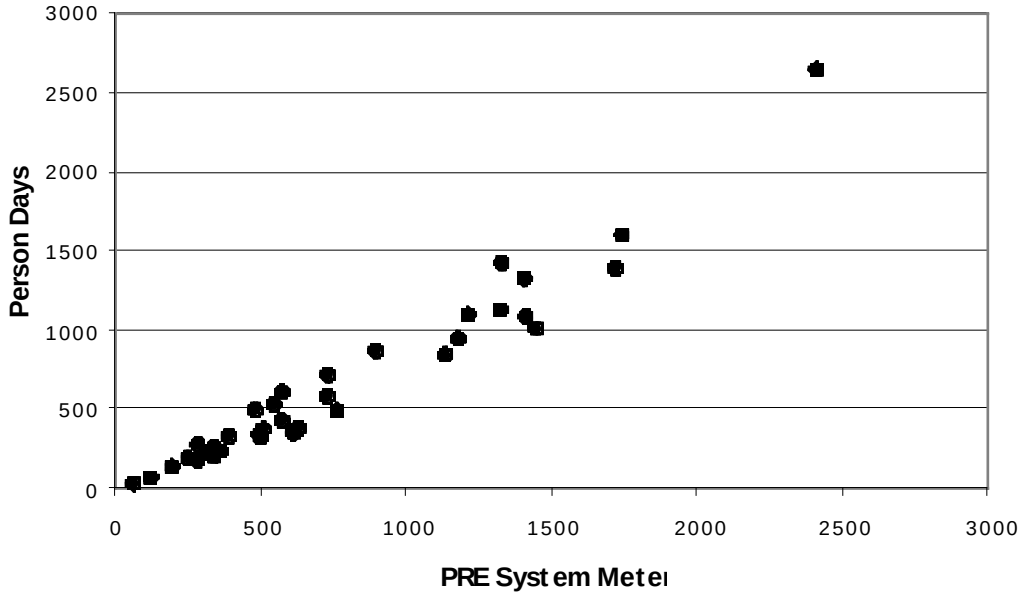


36 Projects:

- 33 industry projects (6 companies); 3 universitary projects
- time span: completion date mainly in 1994/95)
- C++: 4 4GL: 6 Smalltalk: 26
- client/server database-applications: 29

Results of a Field Study (2/4)

PRE System Meter



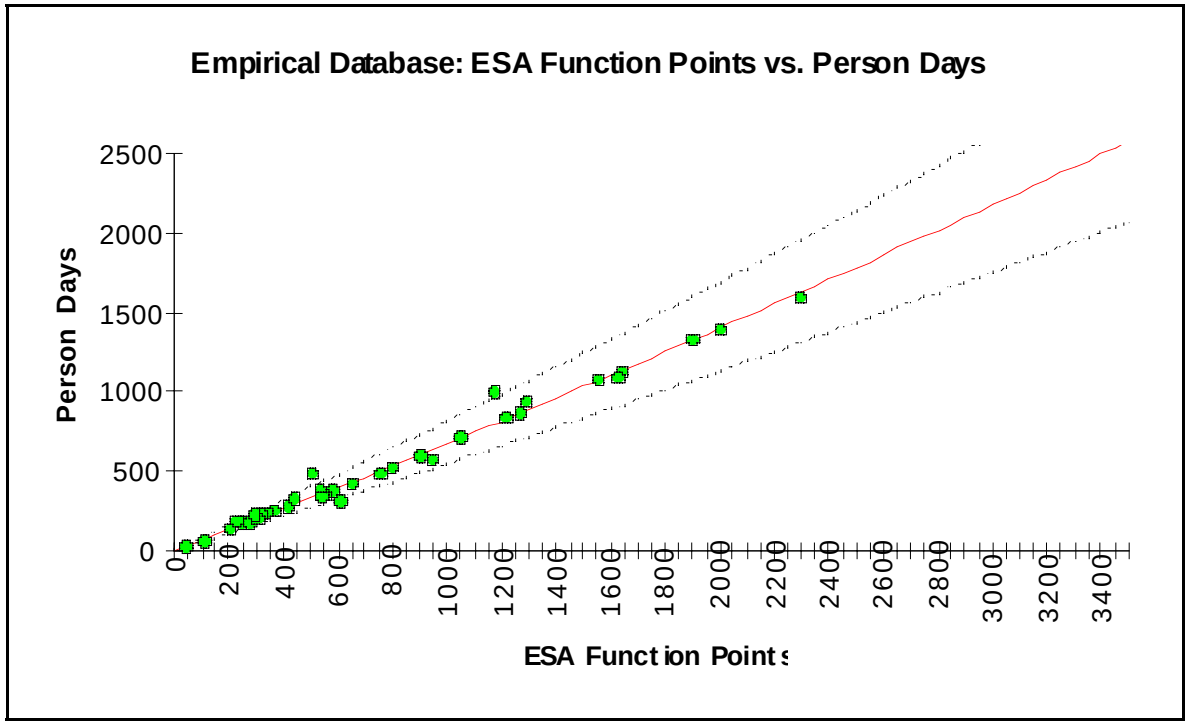
PRE-SM Survey Results: $A = 0.605 \cdot s + 0.0001779 \cdot s^2$, $dA = \pm 33\%$

Additional analysis (compared to Function Points):

- Better adjustment for reuse
- Better correlation in the 7 non-IS projects

Results of a Field Study (3/4)

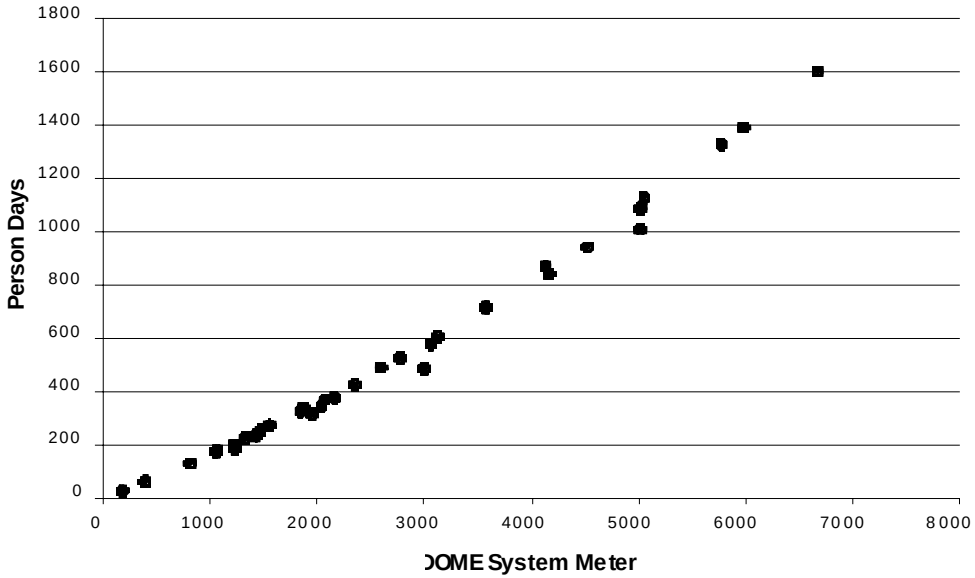
Function Points:



FPM survey results: $A = 0.656 \cdot s + 0.000235 \cdot s^2$, $dA = \pm 20\%$

Results of a Field Study (4/4)

DOME System Meter:



DOME-SM Survey Results: $A = 0.151 \cdot s + 0.0000126 \cdot s^2$, $dA = \pm 9\%$

Additional analysis (Wilcoxon-signed-ranksum-test):

- The correlation improvement over FP is significant

An Example (1/5)

Prerequisite for Step A: a System Model ...

```
; PRE description of TOIS, the tiny order information system
Subject Area “Customer Information“ 5 .
Subject Area “Order Information“ 3 .
Subject Area “Stock Information“ 5 .
Functionality “Manage Objects“ 4 .
Functionality “Do Statistics“ 2 .
Functionality “Do Forecasts“ 2 .
Subject Area ’Customer Information’ contains ’Manage Objects’ .
Subject Area ’Order Information’ contains ’Manage Objects’ .
Subject Area ’Stock Information’ contains ’Manage Objects’,
        ’Do Statistics’, ’Do Forecasts’ .
```

... eventually refined with information about reuse:

```
...
;ma-entry: category library
Functionality “Manage Objects“ 4 .
;ma-entry: category project
...
```

An Example (2/5)

Step A: measuring the System Model

Measurement should be algorithmic ⇒ use an automated tool (download, e.g., `SEBT`, the software estimation basic toolkit from ftp://ftp.csse.swin.edu.au/outgoing/simonm/sebt.zip, use the unzipper ftp://ftp.csse.swin.edu.au/outgoing/simonm/pkunzip.exe)

Measurement is then as simple as typing some (DOS) command ...

```
ma -v -f tois.sdf
```

... and watch the result to plop out:

```
System Meters = 563
```

November 1999: New tool with GUI: http://www.softengprod.com

An Example (3/5)

Prerequisite for Step B: setting-up or obtaining an Empirical Database
= measuring predictor and result values of completed projects

Use the databases (edb_pre.xls, edb_dome.xls) contained in SEBT (37 projects)

Step B: using the Empirical Database

PRE-SM Survey Results: $A = s \cdot 0.605 + s^2 \cdot 0.0001779$, $dA = \pm 33\%$

$$563 \times 0.605 + 563\_ \times 0.0001779 = 340.6 + 56.4 = 397 \text{ PD}$$

An Example (4/5)

Prerequisite for Step C: a Software Process Model (**XX%** = standard, * = repeatable)

BIO Layer	Result	Exp. %	Evol. %	Full %	BIO Layer	Result	Exp. %	Evol. %	Full %
Preliminary Analysis 5%	Subject Areas	1/2 %	1 %	**2 %**		*[replication cont.]*			
	Goals	1/2 %	1 %	**3 %**		User Manual / Online Help *	1 %	3 %	4 %
Domain Analysis 14%	Use Case Model	1 %	3 %	**5 %**		Forms (for manual processes)	1/2 %	1/2 %	1 %
	Domain Class Model	1 %	3 %	**5 %**	**Delivery 6%**	Acceptance	1/2 %	2 %	**3 %**
	State-Transition Models	1 %	2 %	4 %		Installations *	1/4 %	1 %	**1 %**
	Non-essential Requirements	1 %	2 %	**4 %**		User Instruction *	1%	1 1/2%	**2 %**
Application Analysis 18%	Specification Types *	1 %	3 %	**5 %**		Organisational Changes	1/2 %	1 1/2 %	2 %
	Models	2 %	3 %	**5 %**		Data Migration	2 %	3 %	4 %
	System States	2 %	3 %	4 %	**Project Management 10%**	Plans	1 %	1 1/2 %	**2 %**
	Application Class Model	2 %	4 %	**6 %**		Estimates	1/2 %	1 %	**1 %**
	Non-functional Requirements	1/2 %	1 %	**2 %**		Configuration Mgmt	2 %	2 %	**3 %**
Construc-tion 19%	Implementation Patterns *	2 %	4 %	**5 %**		Problem and Change Mgmt	1 %	2 %	**3 %**
	Relational Model	1 %	3 %	**4 %**		Controlling and Reporting	1 %	1 %	**1 %**
	Technical Class Model	1 %	2 %	**2 %**		Evaluations *	1/2 %	2 %	3 %
	Test Data	1 %	3 %	**4 %**		Prep. Organisational Changes	1/2 %	2 %	3 %
	Test Cases	2 %	3 %	**4 %**		Prep. User Instructions	1 %	2 %	3 %
Replica-tion 38%	Tuned Items *	2 %	4 %	5 %		Prep. Data Migration	2 %	2 %	3 %
	Code	10 %	25 %	**30 %**	**Quality Management 8%**	Risk Analysis / Quality Plans	1 %	1 1/2 %	**2 %**
	Admin. & Installation Code	1 %	4 %	**5 %**		Measurements	1 %	2 %	**3 %**
	Platform Port *	2 %	8 %	10%		Defining Standards	1/2 %	1 1/2 %	**3 %**
	Layout (GUI) Translation *	4 %	5 %	5 %		Developer Instruction	1/4 %	1/2 %	1/2 %
	System Admin. Manual *	1 %	2 %	**3 %**		Project Reviews	1/4 %	1/2 %	1/2 %

An Example (5/5)

Step C: adapting the original estimate to the tailored process model

Example: we conduct a full application analysis and a prototypical construction focussing on 3 implementation patterns without formal test preparation:
= 18% + 3x2% + 1% + 1% = 26%

The resulting effort estimate therefore is:

$$397PD \times 26\% = 103PD$$

Additional adaptations:

- Optimum team sizes, maximising speed of development
- Reducing budget overrun risks by adding "buffer effort"

Future Work

- The System Meter is used in industry

- Due to its formal properties the System Meter may be used in derived measures

References:

1. DeMarco T, Controlling SW Projects, Prentice-Hall, Englewood Cliffs, N.J., 1982
2. Moser S, Cherix R, Flueckiger J, HERMES/Bedag Informatik Vorgehensmodell, Bedag Informatik, Berne, Switzerland, 1993-1999
3. IFPUG, Counting Practices Manual V4.0, Westerville, Ohio, USA, 1996
4. Rumbaugh J, Jacobson I, Booch G, The Unified Modeling Language (UML) Ref. Manual, Addison-Wesley, Reading MA, 1999
5. Firesmith D, Henderson-Sellers B, Graham I, OPEN Modeling Language (OML) Ref. Manual, SIGS Books, NY, 1997
6. Moser S, Measurement and Estimation of Software and Software Processes, Ph.D. thesis, University of Berne, Berne, Switzerland, 1996
7. Henderson-Sellers B, Graham IM, Younessi H, The OPEN Process Specification, Addison-Wesley, UK, 1998