

Software project estimation in the maintenance context: Applying use case points

Sergey Diev

Bank of Montreal Financial Group

sergey.diev@bmonb.com, sergey@diev.com

Our objectives for today are to consider:

- Main aspects of software estimation in a complex maintenance environment
- Size-based estimation process
- The Use Case Points approach

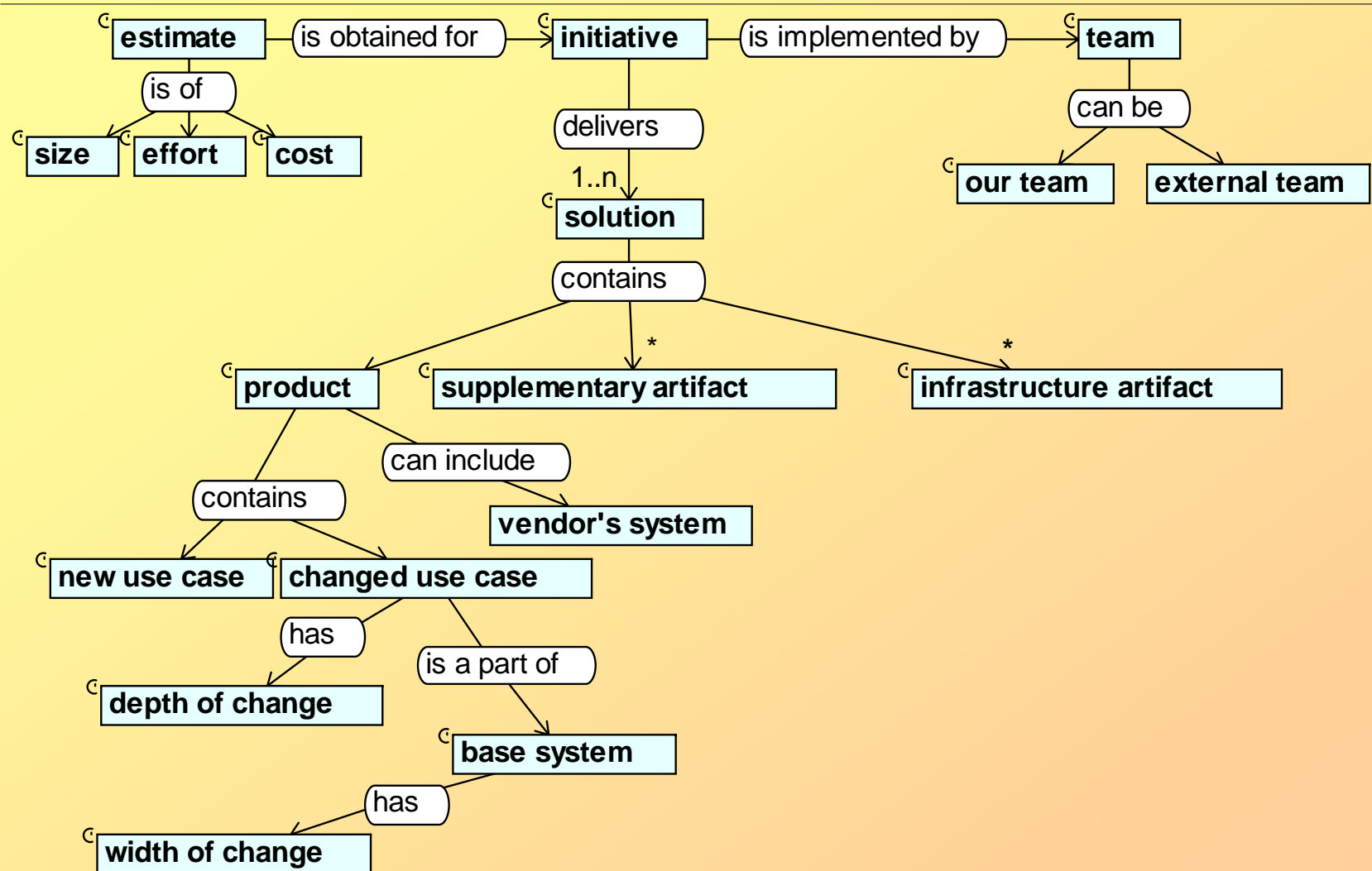
Agenda

- What is estimation
- Size and effort, or why we need size
- What is software size estimation
- UCPm: Use Case Points method modified for maintenance
- Use cases and estimation
- Lessons learned

Points that will be emphasized

- *Accurate definitions and well-defined process are essential for meaningful estimation and measurements*
- *Maintenance has its own specifics that has to be taken into account in estimation*
- *To be able to apply use cases in estimation, we need some changes in the general use case approach*

Estimation concepts



Definitions

- **Initiative**

An undertaking that results in a software product.

An initiative may be a formal project or enhancement (in other contexts, an initiative may also be a bug fix)

- **Product**

A set of deliverables that will be working or used in production

- **Base system**

If the purpose of an initiative is to change an existing system, the latter is called *base system*

Base systems can vary in complexity: they may be simple, medium or complex

Definitions, cont'd

- **Supplementary artifact**

A software engineering artifact necessary to deliver the product, such as script, code written to migrate data, etc.

- **Solution**

Product plus supplementary artifacts plus infrastructure artifacts – that is, all artifacts that have to be delivered within the initiative in order to provide the business value (excluding hardware)

Product

High-level requirements

Detailed requirements

Architecture

Design

Code

Data structures

Test artifacts

Documentation

French translations

Supplementary artifacts

Throw-away prototypes

Transition code

Environment setup

Configuration mgt artifacts

Release management artifacts

Regression test artifacts

Estimate

- **Estimate** is a statement of the size, or effort, or cost of a product, solution, or initiative or some part of the product, solution, or initiative

Properties of estimates

- Estimates are characterized by the accuracy with which they are produced
- Estimates may differ in how they relate to the life cycle
- Estimates may also differ in delivery scope: which part of the product, solution or initiative they consider

Why do we need size?

- Effort depends on qualification, tools, etc.
- Size does not depend on that, on work distribution, workers' participation, etc.
- Effort is not a deliverable, it is what is invested to produce deliverables
- Estimated effort may be consumed too soon if it is not related to an independent variable (i.e. size)
- Size is the only way to measure objectively the amount of work to do

How do we use size?

- To derive effort from size
- To predict characteristics of projects
 - E.g. expected number of defects in requirements
- To normalize measurement data
 - “Project A had 20 change requests, while project B had 30”
 - “Project A had 0.01 change request per size unit, while project B had .5 change requests per size unit”

UCP: Use Case Points method

- UCP was created
in 1993 by Gustav Karner of Objectory AG (later
acquired by Rational),
following the Function Points method,
as a tool for estimating person/hours for projects
based on use cases,
for new system development
it does not consider the specifics of the maintenance
context

UCPm: Use Case Points method extended to maintenance

The specifics of the software development context that we cannot ignore:

- Many initiatives are enhancements or technology upgrades => need to take into account base systems
- Products usually contain new functionality and changes at the same time
- Regression testing is necessary, though it may be not exhaustive each time
- Etc.

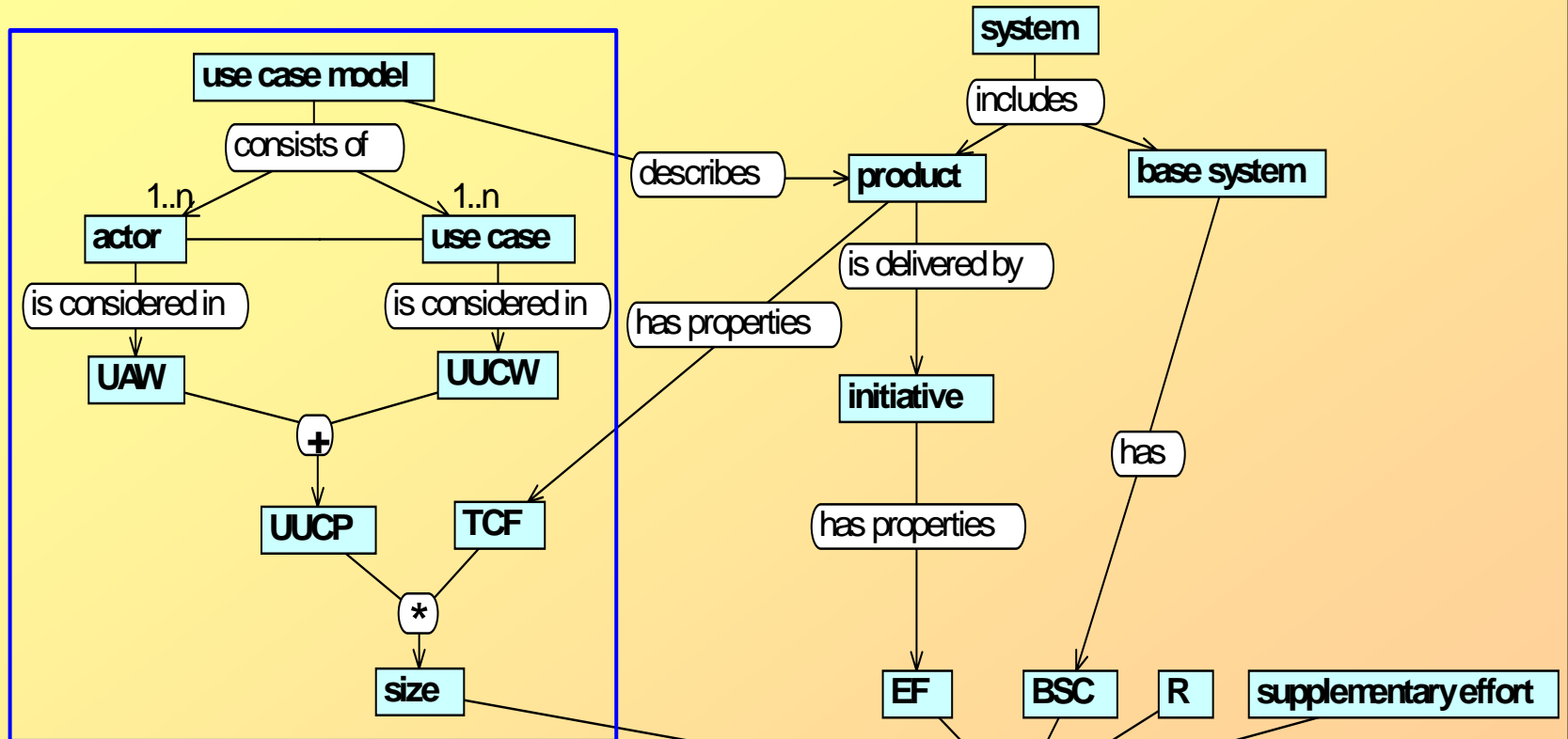
Use case model

- Use case
 - Is a set of sequences of actions a system performs that yields an observable result of value to a particular actor
 - Has many realizations
 - Not a function
 - Examples: *buy equity; get client activity report*
- Use case model contains actors and use cases
 - If it does not exist, the use case model can be built on the base of high-level requirements
- The size of a use case model is calculated in use case points

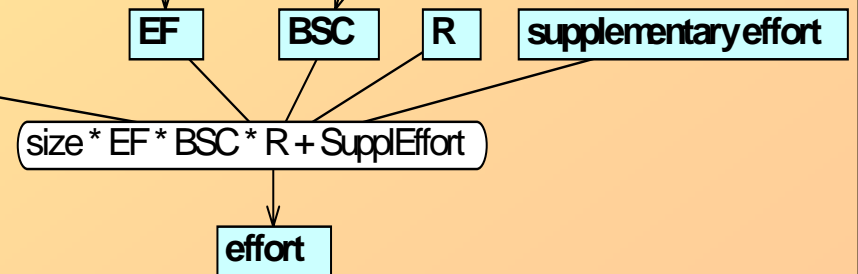
Use case transaction

- An atomic set of activities between actor and system, which is either performed entirely or not at all
- The smallest unit of activity that is meaningful from the actor's point of view
- Self-contained and leaves the business of the application being sized in a consistent state
- Example: update address; define beneficiary

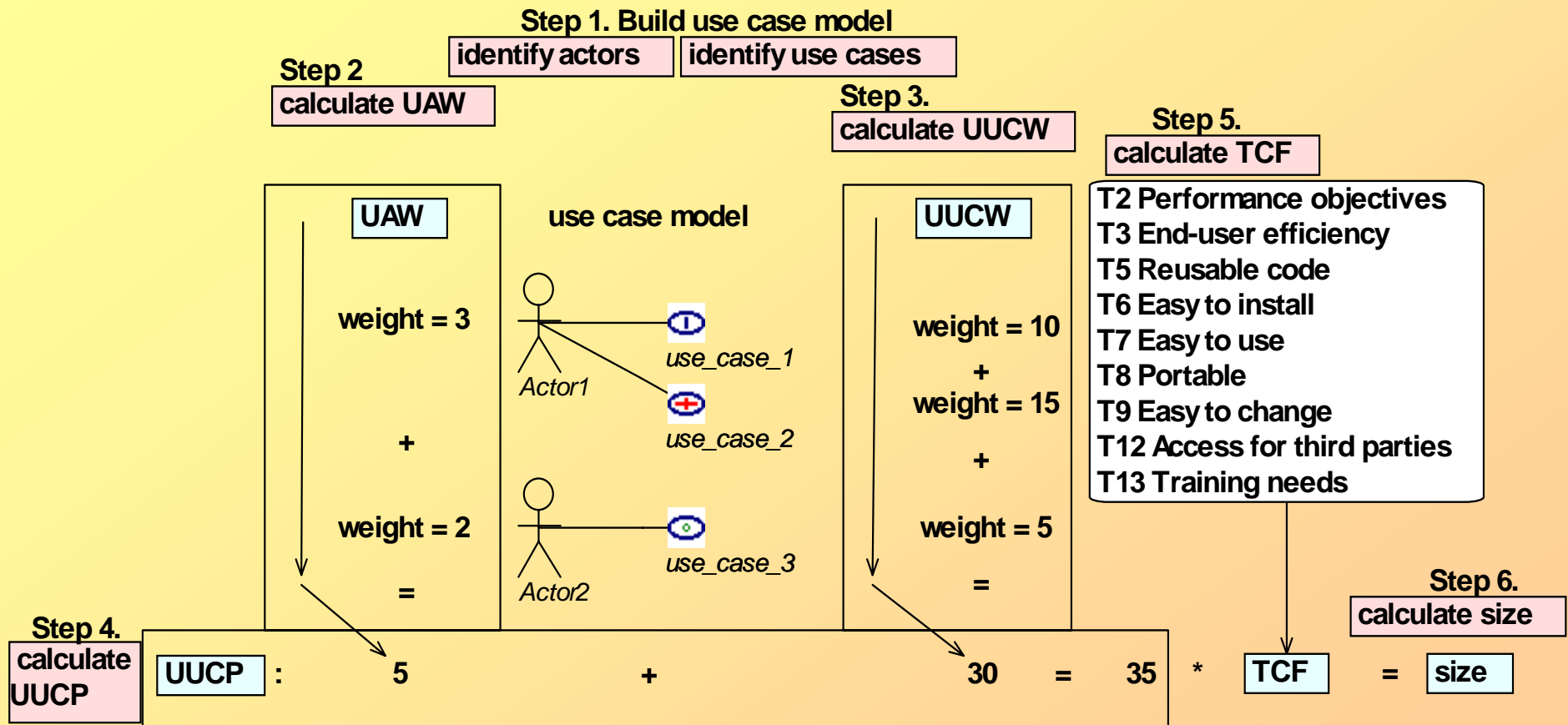
UCPm: Concepts



UAW unadjusted actors weight
 UUCW unadjusted use case weight
 UUCP unadjusted use case points
 TCF technical complexity factor
 EF environment factor
 BSC base system complexity



Sizing procedure



Unadjusted Actors Weight (UAW)

a. Qualify actors:

<i>Type of actor</i>	<i>weight</i>
• Simple (Program interface)	1
• Average (Interactive, or protocol-driven interface)	2
• Complex (Graphical interface)	3

b. Add weights of all actors to obtain **UAW**

Unadjusted Use Case Weight (UUCW)

a. Qualify each use case:

	<i>weight</i>
• Simple (1-3 use case transactions)	5
• Average (4-7 use case transactions)	10
• Complex (>7 use case transactions)	15

Apply four technical sub-factors

- distributed system, complex processing, concurrency, security

b. Add weights of all use cases to get UUCW

Unadjusted size of new functionality

Size(NF)

$\text{Size(NF)} = \text{UAW} + \text{UUCW}$
(Unadjusted Actors Weight
plus
Unadjusted Use Case Weight)

Depth of change

- The extent to which a use case is affected by change: N_2 / N_1

where

- N_1 is the number of use case transactions in a use case before the change
- N_2 is the number of added, updated or deleted use case transactions

Unadjusted change size $\text{Size}(\text{Ch})$

$$\text{Size}(\text{Ch}) = \sum_{j=1 \dots \text{Width}(\text{BS})} (\text{Weight}(\text{BUC}_j) * \text{Depth}(\text{Ch}_j)),$$

where

- BUC_j is the j -th use case of base system
- $\text{Depth}(\text{Ch}_j)$ is the depth of change to the j -th use case of the base system
- $\text{Width} * \text{BS}$) is the number of affected use cases from the base system

Unadjusted Use Case Points

$$\text{UUCP} = \text{Size}(\text{NF}) + \text{Size}(\text{Ch})$$

(size of new functionality

plus

size of changes)

Technical Complexity Factor (TCF)

Describes the technical complexity of the product

a. Rate 9 technical factors on the scale from 0 to 5:

[T01: Distributed system]

T02: Response or throughput performance objectives

T03: End-user efficiency (online)

[T04: Complex internal processing]

T05: Code must be reusable

T06: Easy to install

T07: Easy to use

T08: Portable

T09: Easy to change

[T10: Concurrency]

[T11: Includes special security features]

T12: Provides direct access for third parties

T13: Special user training facilities are required

T02: Performance objectives factor

- This factor describes the degree to which response time and throughput performance considerations influence the product development.
- Rate Influence
 - 0 No special performance requirements were stated by the user.
 - 1 Performance and design requirements were stated and reviewed but no special actions were required.
 - 2 Response time or throughput is critical during peak hours. No special design for CPU utilization was required. Processing deadline is for the next business cycle.
 - 3 Response time or throughput is critical during all business hours. No special design for CPU utilization was required. Processing deadline requirements with interfacing systems are constraining.
 - 4 In addition, stated user performance requirements are stringent enough to require performance analysis tasks in the design phase.
 - 5 In addition, performance analysis tools were used in the design, development, and/or implementation phases to meet the stated user performance requirements.

Technical Complexity Factor (TCF)

b.
$$\text{TFactor} = \sum_{i=2,3,5..9,12,13} \text{TRate}_i * \text{Weight}_i$$

(each factor has its weight)

c.
$$\text{TCF} = 0.6 + 0.01 * \text{TFactor}$$

Product size

$$\text{Size(Pr)} = \text{UUCP} * \text{TCF}$$

where

- UUCP – number of unadjusted use case points
- TCF – technical complexity factor

Environmental factor (EF)

Describes the project

a. Rate each of eight factors on the scale from 0 to 5

E01: Familiarity with the standard process

E02: Application experience

E03: Methodology experience (e.g. with OO)

E04: Lead analyst capacity

E05: Motivation

E06: (In)stability of requirements

E07: Part-time workers

E08: Difficult programming language

Environmental factor (EF), cont'd

b. $E_{\text{factor}} = \sum_{i=1..8} E_{\text{Rate}_j} * \text{Weight}_j$
(each factor has its weight)

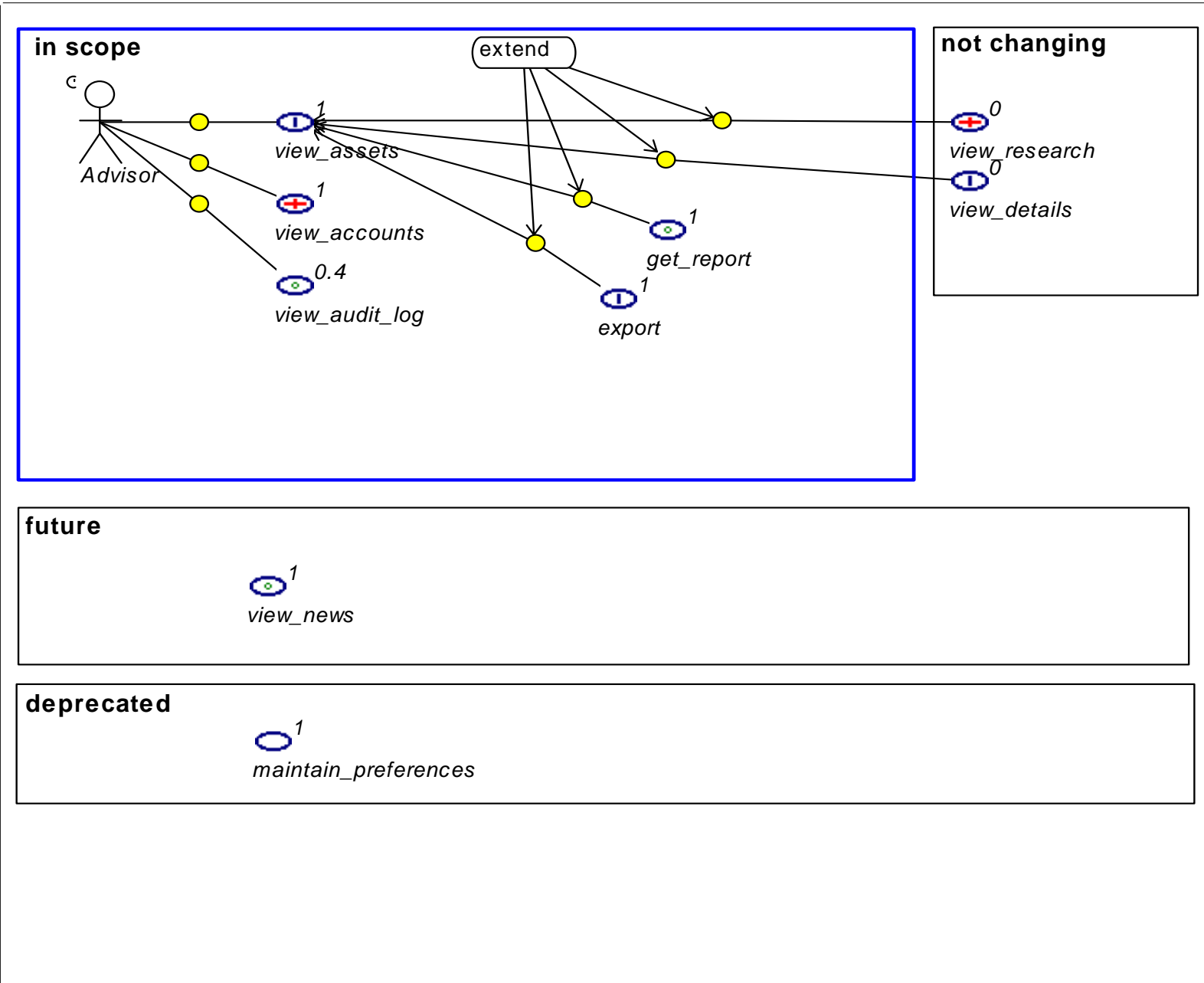
c. $EF = 1.4 + (-0.03 * E_{\text{factor}})$

Effort

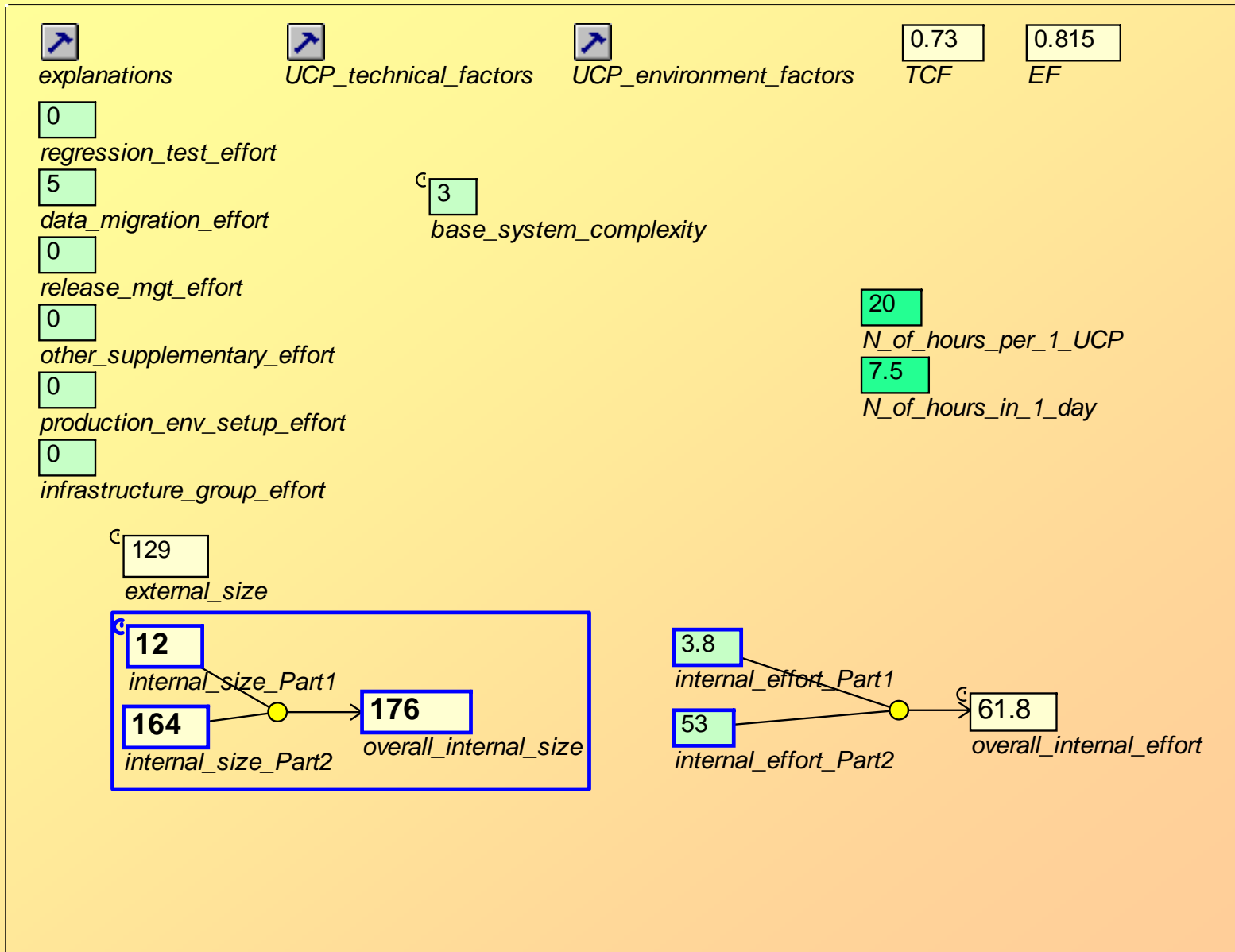
$$\text{Effort(In)} = \text{Size(Pr)} * \text{EF} * \text{BSC} * \text{R} + \text{Effort(Suppl)}$$

where

- In – initiative
- Pr – product
- EF – environmental factor
- BSC – complexity of the base system
- R is the effort in person/hours that corresponds to one unadjusted use case point (R=20)
- Effort(Suppl) is the effort to deliver supplementary artifacts



Size/effort results screen



B	C	D	E	F	G	H	I	J	K	L	
INITIATIVE			date			ESTIMATION					
Initiative name:	DO NOT EDIT THIS TEMPLATE! Make a copy, place it as #3 from the						product size:	0.0	UCP		
Project Manager / Lead:							effort:	0.0	pm	1UCP	
Business Analyst(s)							cost:	0.0	dollars	\$	-
documents:							effort by decomposition (p/m):	0.0	difference:		
base system(s):										%	
base system complexity:	1.0						effort actual (p/m):	0.0	difference:		
apl. artifacts effort (p/m):	0.0									%	

TECHNICAL FACTORS - assign rate 0--5 TCF: 0.60

name	weight	rate	value
distributed system	2		0
response time or performance objectives	1		0
end user efficiency	1		0
complex internal processing	1		0
reusable code	1		0
easy to install	0.5		0
easy to use	0.5		0
portable	2		0
easy to change	1		0
concurrent use	1		0
security objectives	1		0
access to third parties	1		0
training needs	1		0

ENVIRONMENTAL FACTORS - assign rate 0--5 EF: 1.40

ID	name	weight	rate	value
E1	familiar with development process	1.5		0
E2	application experience	0.5		0
E3	object-oriented experience	1		0
E4	lead analyst capability	0.5		0
E5	motivation	1		0
E6	stable requirements	2		0
E7	part time staff	-1		0
E8	difficult programming language	-1		0

ACTORS number of actors: 0 UAW: 0

actor	include	factor	rank	weight	value	comments
	no	1	simple	1		
	no	1	average	2		
	no	1	complex	3		

CASES number of use cases: 0 UUCV: 0

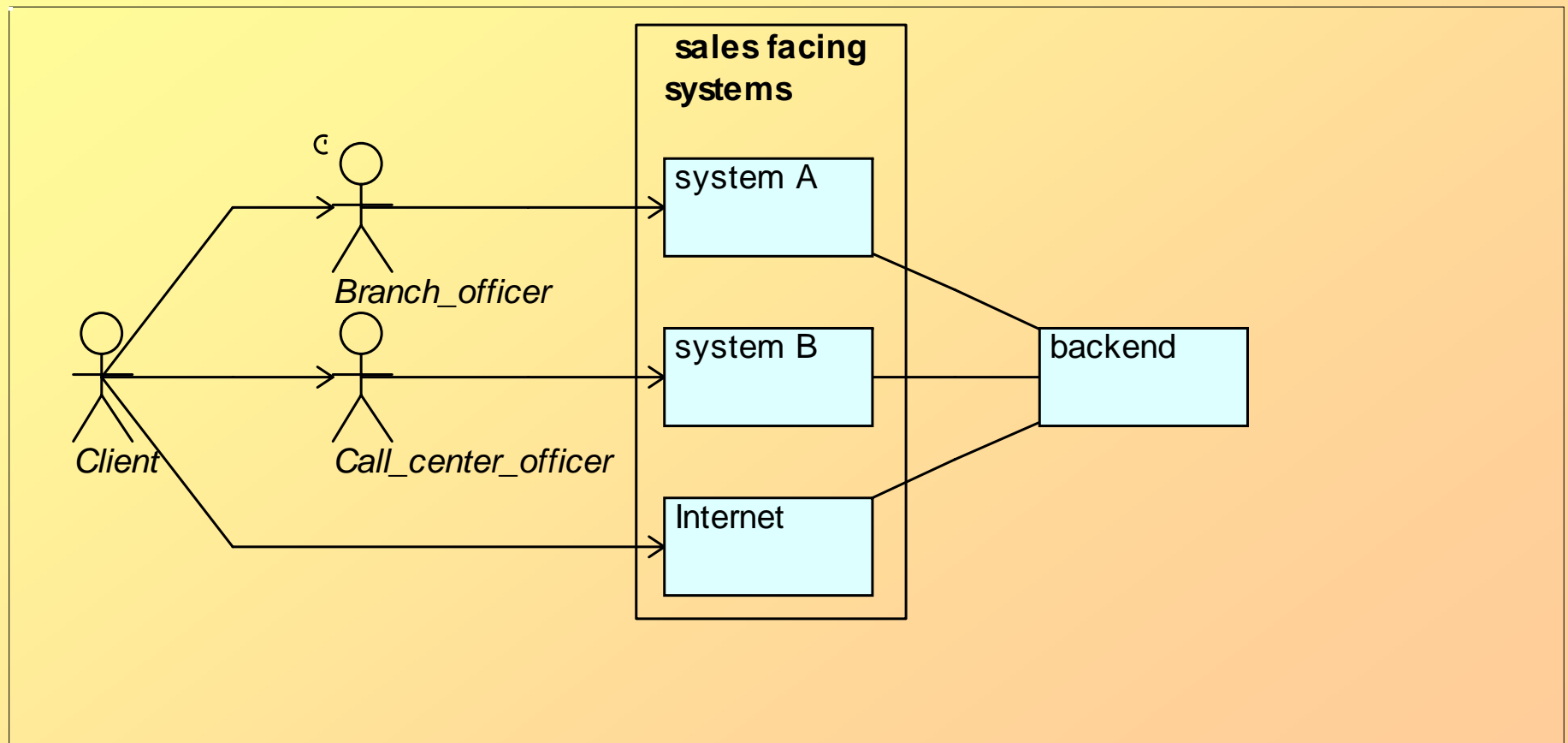
actor	use case	include	factor	rank	weight	value	comments
		no	1	simple	5		
		no	1	average	10		
		no	1	complex	15		
			1				
			1				

Use cases in estimation

- It is necessary to take into account details of the context, for example:
 - Use case duplication
 - “Vertical” distribution of work
 - Programmatically implemented use case
 - Etc.

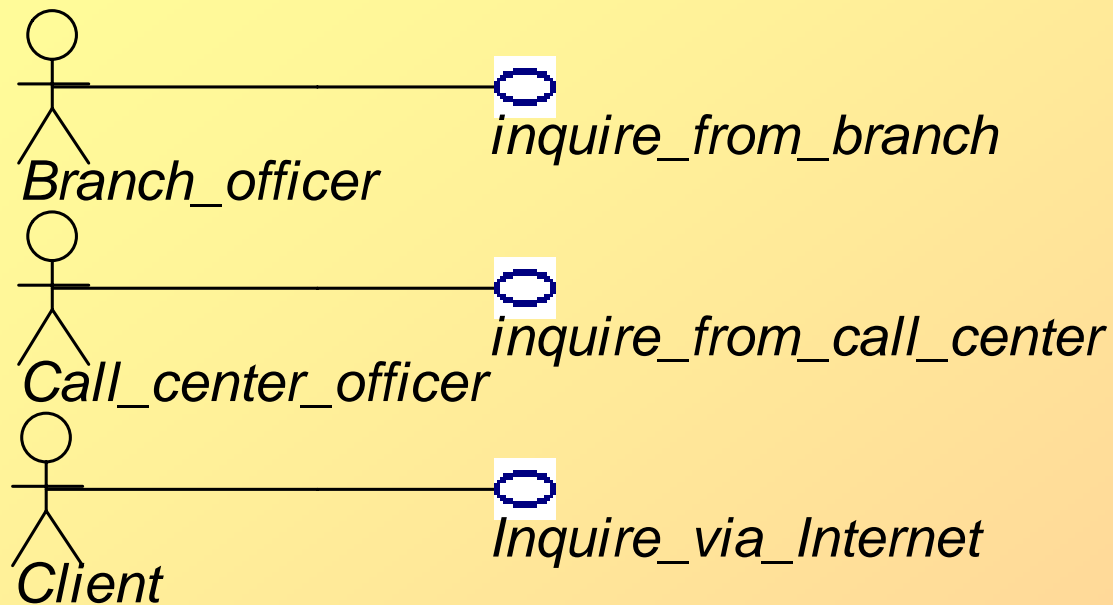
Use cases in estimation: Example

Suppose we are enhancing a system with the following architecture:



Example, cont'd

Use case Inquire:



Concluding remarks

- The UCPm has been applied to 80+ projects
- The measurement programme is heavily based on size
- The sizing approach and its implementation got very high marks from CMM appraisers
- It was found that it is rather easy to build a use case model even when requirements are not produced in use case style

Concluding remarks, cont'd

- On average, estimation takes appr. 1% of a project's budget (Capers Jones). The effort to get a UCP estimate is significantly smaller
 - And the major part of it is the effort to build a use case model(!)
- A relatively high level of UCP/UCPm reduces the amount of work on estimation
 - In our estimates, 1 UCP maps to approximately 3.1 function points. That seems to correlate with the amount of work corresponding to each method

Concluding remarks, cont'd

- What is required from the estimator:
 - A good command of use case modeling
 - Knowledge of UCP/UCPm
 - A good understanding of the business domain
 - A good understanding of existing system(s)

Concluding remarks, cont'd

- A repository of use case models helps to maintain consistency between estimates and also provides opportunities for model reuse
- The repository includes
 - UCPm templates
 - A catalogue of modeling constructs
 - UCPm guidelines and reference materials
 - Use case models of estimated projects

Bibliography

- Karner, Gustav. Metrics for Objectory.
Diploma thesis, University of Linköping, Sweden. No. LiTHIDA-Ex-9344:21. December 1993.
- Schneider, Geri, Winters, Jason P. Applying Use Cases: A Practical Guide.
Addison-Wesley, 1998.
- Diev, Sergey. Software estimation in the maintenance context.
ACM SIGSOFT Software Engineering Notes, 2006, Volume 31, Issue 2 (March), 8 pages.
- Diev, Sergey. Use cases modeling and software estimation: Applying Use Case Points.
ACM SIGSOFT Software Engineering Notes, 2006, Vol. 31, Issue 6 (November), 4 pages.

Questions?