



G A L O R A T H

Normalized Use Cases – A Sizing Metric

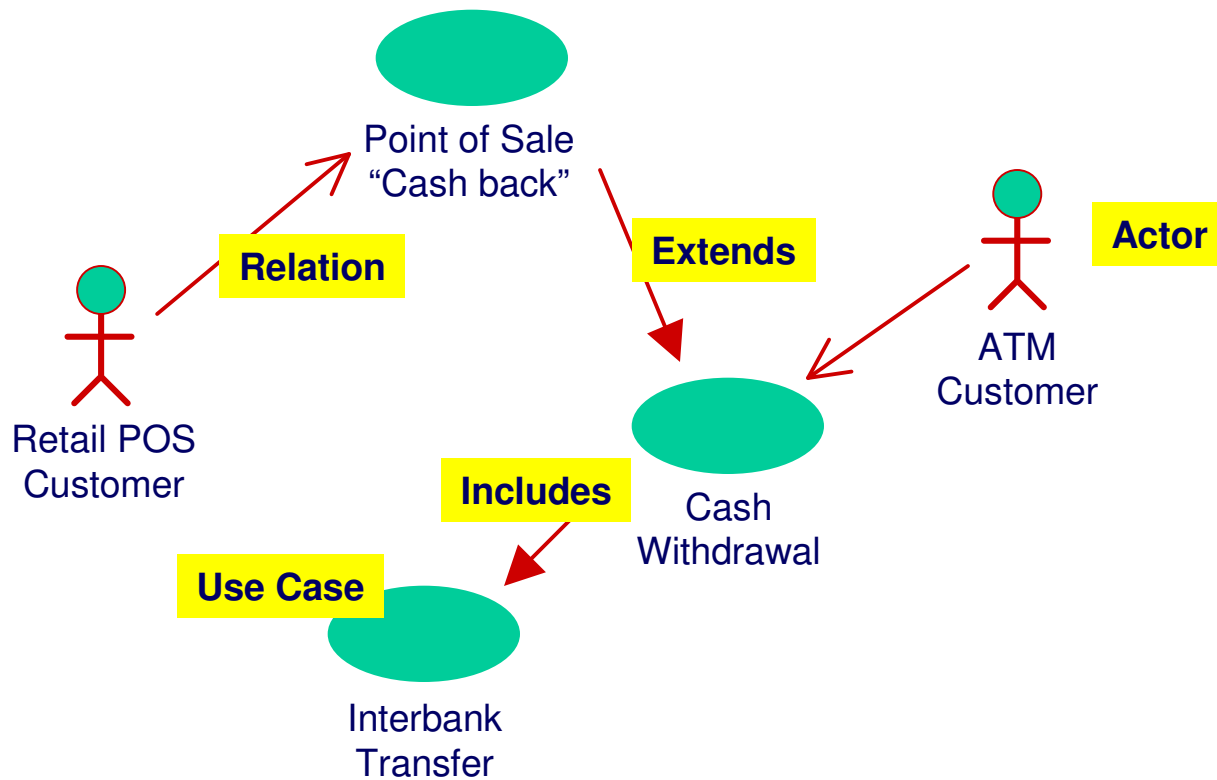
**Lee Fischman and Mike Ross
Galorath, Inc.**



SEER



The Use Case



- "X uses Y" indicates that the task "X" **has a** subtask "Y"; that is, in the process of completing task "X", task "Y" will be completed at least once.

- "X extends Y" indicates that "X" **is a** task for the same type as "Y", but "X" is a special, more specific case of doing "Y". That is, doing X is a lot like doing Y, but X has a few extra processes to it that go above and beyond the things that must be done in order to complete Y.

The “Ubiquitous” Use Case Point

**# Use Case Points (UCP) = Unadjusted Actor Weights (UAW)
+ Unadjusted Use Case Weights (UUCW)**

UAW = sum (Actors)

An actor is weighted according to three complexity levels:

Simple (plain API call): 1

Average (interaction through basic protocol): 2

Complex (complex actor): 3

UUCW = sum (Use Cases)

An actor is weighted according to three complexity levels:

Simple (3 or less steps): 5

Average (4 to 7 steps): 10

Complex (7 or more steps): 15

Use Case Points: Far From Perfect

On one hand, UCPs are:

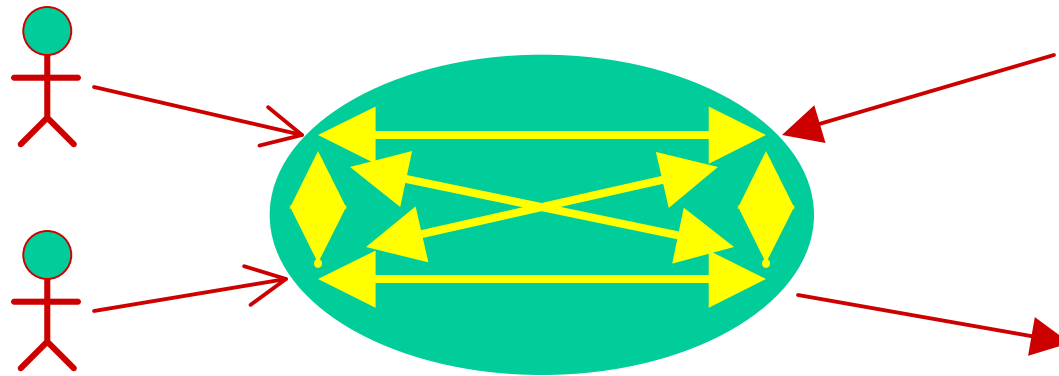
- Intuitive
- Easy to obtain
- Not terribly *inaccurate*

On the other, UCPs are not a real metric – too much varies:

- **Level of decomposition**: UCs specified at a high level can miss much of the richness within a system.
- **Development phase** at which estimate is conducted: later phases are likely to be better specified.
- **Specification standards**: specifying styles that vary from “standard” introduce inconsistencies when translated to UCPs.
- **Complexity characteristics**: existing rating scales and simplistic math may be inadequate for detailed estimates.

Re-examining the Use Case Is There A Better Way?

Characteristics Having An Interrelated Effect On Size



Number of actors, includes and extends are likely to have both linear and multiplicative effects on use case size and complexity.

Use Case Points are not capturing this information.

Normalized Use Case – A Recipe

1. Count all available artifacts:

- Associated Relations, i.e. relations with actors
- Use Case (1 by def'n)
- Unassociated Relations
- Includes
- Extends

Actors are not explicitly counted

2. Rate average functionality (complexity) in each type of artifact using:

- For use cases - # of steps (or an estimate thereof)
- For relations (associated and unassociated) - use case point ratings

3. Cross-multiply all items (easiest to do):

- Includes * extends
- Associated relations * Unassociated relations
- Etc.

4. Weight each term by a coefficient, and you have a NUC:

$$\text{NUC} = a * \text{assoc_relations}() + u * \text{unassoc_relations}() + \\ [au * \text{assoc_relations}() * \text{unassoc_relations}()] + \\ i * \text{includes}() + e * \text{extends}() + ie * [(\text{includes}() * \text{extends}())] + \dots$$

Normalized Use Case Considerations So Far

- Relative functionality (complexity) still matters, thus ratings are accepted for actor and use case complexity
- Associated relations are used as proxies for actors, mitigating variation in number of actors specified
- Interactions between artifacts (captured by “cross terms”) may be significant indicators of scope
- Not all terms may end up being useful
- NUCs are calculated for individual use cases and then summed, while UCPs are calculated in aggregate:

For Use Case Points, calculations are made on a combined basis

$$\text{UCP} = \text{sum}(\text{Actors}) + \text{sum}(\text{Use Cases})$$

For Normalized Use Cases, calculations are made individually

$$\text{sum NUC} = \text{sum} [\text{NUCs}]$$

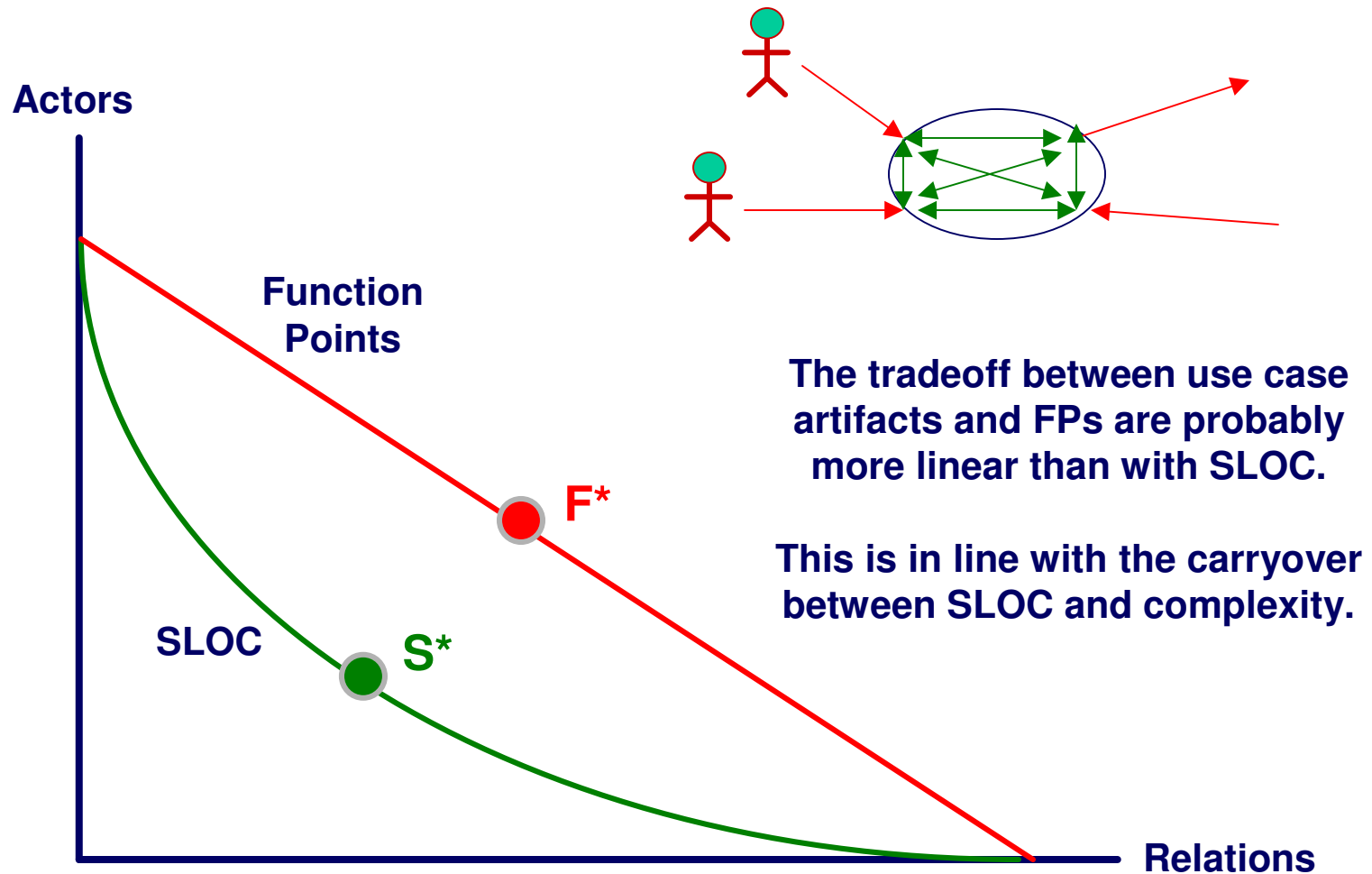
NUC Recipe – Mapping To End Values

Use NUC To Obtain Final SLOC, or other values

5. Calibrate NUCs to a SLOC (S^*) or other value, which must be specific to the use case specification level. “NUC” thus are a class which vary depending on the end mapping chosen: SLOC, FP, effort, etc.
6. For alternate end mappings, such as function points or effort, there must be:
 - A corresponding set of baseline values
 - A new set of coefficients for the NUC
7. NUC coefficient calibrations would be dynamic – at least until more data is available, there is no standard set of coefficients.



Mapping NUCs to SLOC & FPs Via F^* and S^*



Level-Adjusted Starting Values For S*

Choose values for NUCs that also are compatible with UCPs;
1 nominal NUC = 1 nominal UCP

SLOC Ranges by Level

Unit	25	78	242
Component	228	693	1452
Program	1970	5812	9688
Rollup	16441	47395	67424
Rollup of Rollups	134188	381013	476328

Source: Smith

Least

Likely

Most

Direct to effort: 150-350 hours per use case (Probasco).

Conclusion

NUCs vs. UCPs

- Better sensitivity to varying characteristics of use cases
- Useful for single use cases, rather than requiring aggregation
- May reduce over-dependence on complexity ratings
- More information required and more difficult to calculate: intended for automatic counting (as in the CriticalMass project).
- Differentiates between different metrics: separate tables and NUC calculations are needed.
- Can be made comparable to UCPs, so 1 NUC = 1 UCP

References

Henderson-Sellers, Brian, Object-Oriented Metrics, Prentice Hall, 1996

Karner, G, 1993, "Metrics for Objectory". Diploma thesis, University of Linköping, Sweden. No. LiTHIDA-Ex-9344:21. December 1993.

Lorentz, Mark, Object-Oriented Software Metrics, Jeff Kidd, Prentice Hall, 1994

Probasco, Leslee, What About Function Points and Use Cases?, Rational Edge, Rational Software, August 2002

Ribu, Kirsten, Estimating Object-Oriented Software Projects with Use Cases, Master of Science Thesis, Department of Informatics, University of Oslo, November 2001

Smith, John, The Estimation of Effort Based on Use Cases, Rational Software white paper TP-171, October 1999