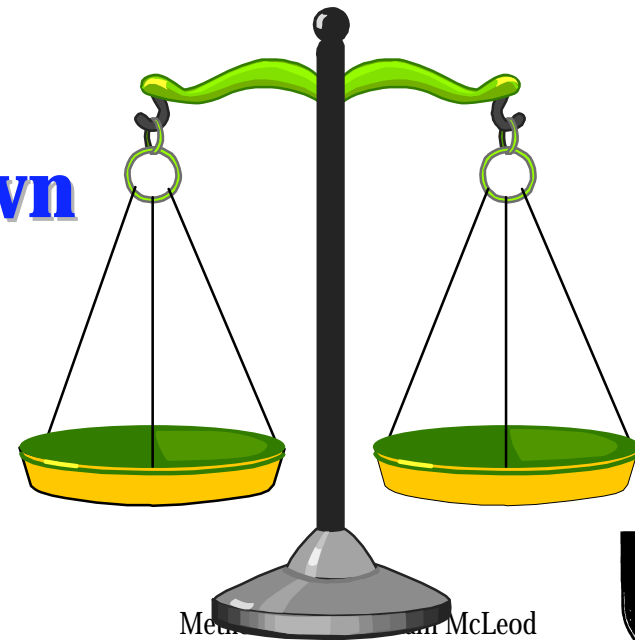


# Method Points

## Towards a Metric for Method Complexity

---

**Graham McLeod**  
**University of Cape Town**



Method Points  
Graham McLeod  
University of Cape Town June '97



# Outline

---

## ▲ Methods Engineering

- Definition
- Objectives
- Need for Complexity Metric

## ▲ Method Modeling

## ▲ Function Points

## ▲ Method Points

- Graphic, Tabular and Textual Deliverables
- Task Complexity
- An example(I.E. vs UML)

## ▲ Suggestions for Future Work



# Methods Engineering

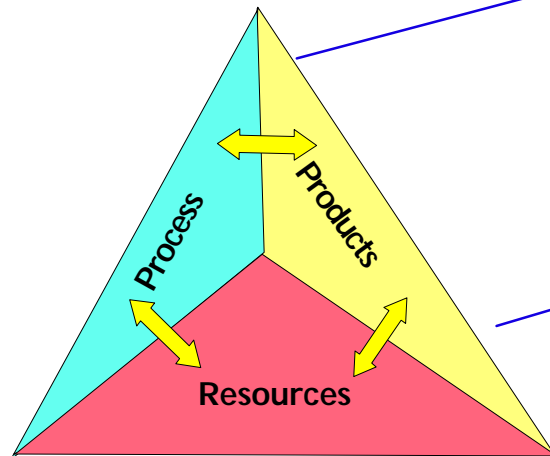
---

- ▲ The process whereby methodologists develop and enhance methods related to information systems in a disciplined way
- ▲ "Method engineering is the coordinated and systematic approach to establishing work methods" - James Odell
- ▲ Objectives
  - Structured, repeatable process for practitioners
  - Integration of techniques -> comprehensive approach -> broader problem
  - Incorporation of more powerful/sophisticated techniques
  - Specification of capable model notation and representation
- ▲ Difficulties
  - Method success dependent upon fit to problem and
  - difficulty of becoming proficient in the use of method
  - Complexity affects ease of use and quality of application
  - BUT no published techniques to measure complexity of methods to make quantitative comparisons
- ▲ Proposal
  - Develop a metric along similar lines to function points for information systems
  - Measure complexity via meta-data not size via deliverables

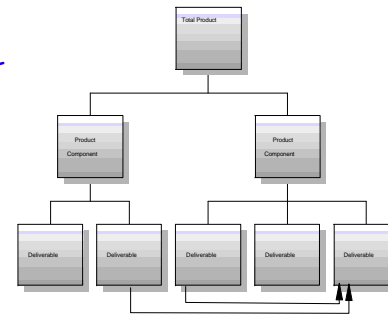


# Modeling Methods

McLeod Method Model



Product Facet Node Structure

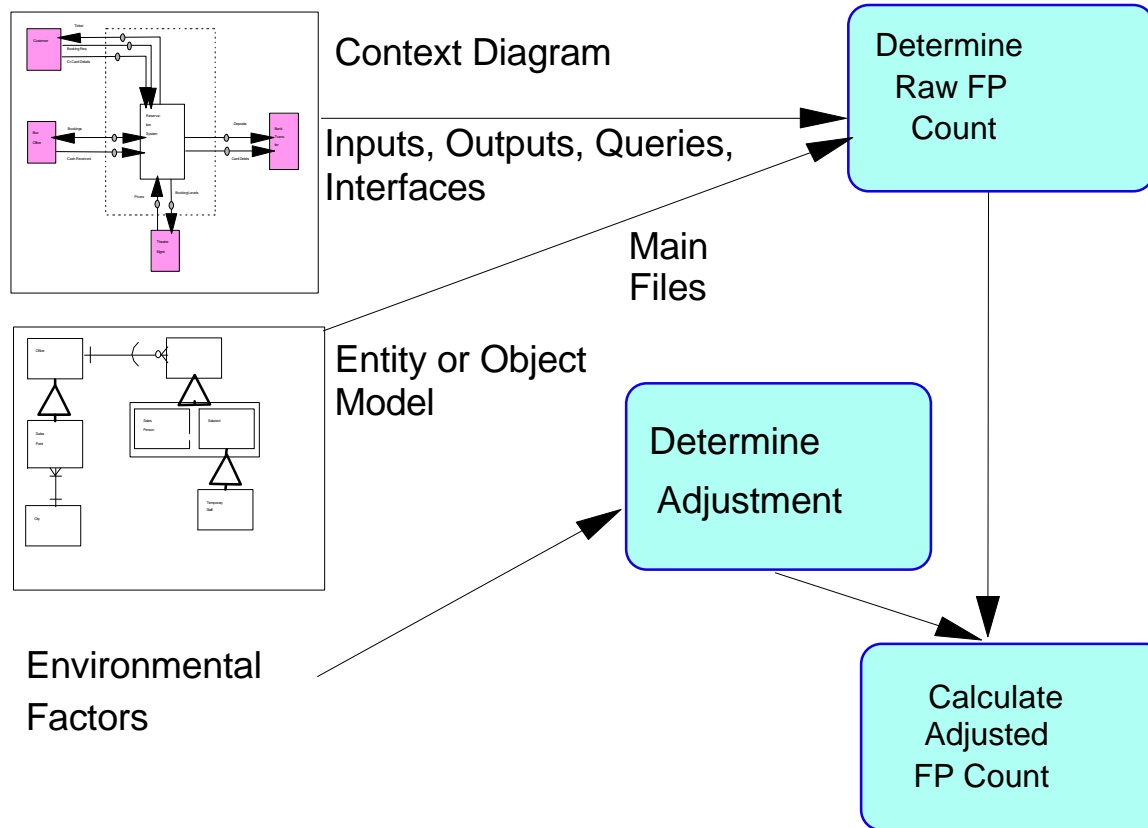


- Product Name
- Parent
- Children
- Prerequisites
- Subsequents
- Representation
- Data Content
- Purpose
- Validation Rules
- Example
- Quality Standards
- Associated Tasks
- Associated Resources
- Permitted States
- Estimating Method
- Tool Support



# Function Points

## Function Point Calculation



# Method Points Process

---

- ▲ Express the method i.t.o. tasks, resources and deliverables
- ▲ Determine the counts for each type of deliverable
  - Graphic, Tabular, Textual
- ▲ Determine and add the count for task complexity
  
- ▲ Graphic Deliverables(weight)
  - Symbol types(1), Link types(.5), Embelishments(.5), Decomposition(.5,.5)
- ▲ Tabular Deliverables (Col .25,Embel .25, Decomp .5)
- ▲ Textual Deliverables (Sect .25,Pg .25, Hyperlink .5, Hierarchy .5)
- ▲ Compensate for Task Complexity



# An Example: UML vs I.E.

---

- ▲ Static Structure Diagram vs Entity Relationship Diagram
- ▲ Use Case Diagram vs Context Diagram
- ▲ Static Structure Diagram (Score: 16.5)
  - Symbols: Class, Type, Template, Ternary Association (4)
  - Link types: Interface, Imports, Binary Association, Generalisation, Dependency, Refinement (3)
  - Embellishments: Bound Element, Utility Modifier, MetaClass Modifier, Pathname Modifier, Various on Associations, Details, Constraints, Derived Element, Navigation Expression (7)
  - Decomposition (2.5)
- ▲ ERM (Score: 4.5)



# Example Continued

---

## ▲ Use Case Diagram (Score: 5.5)

- Symbols: Actors, Use Case (2)
- Link types: Communication, Extension, Uses, Refinement (2)
- Embellishments: Boundary (.5)
- Decomposition (1)

## ▲ Context Diagram (4)

## ▲ Findings and field validation

- UML SSD considerable more complex than IE ERD
- Use Case only somewhat more complex than Context
- Practitioners very this



# Further Work

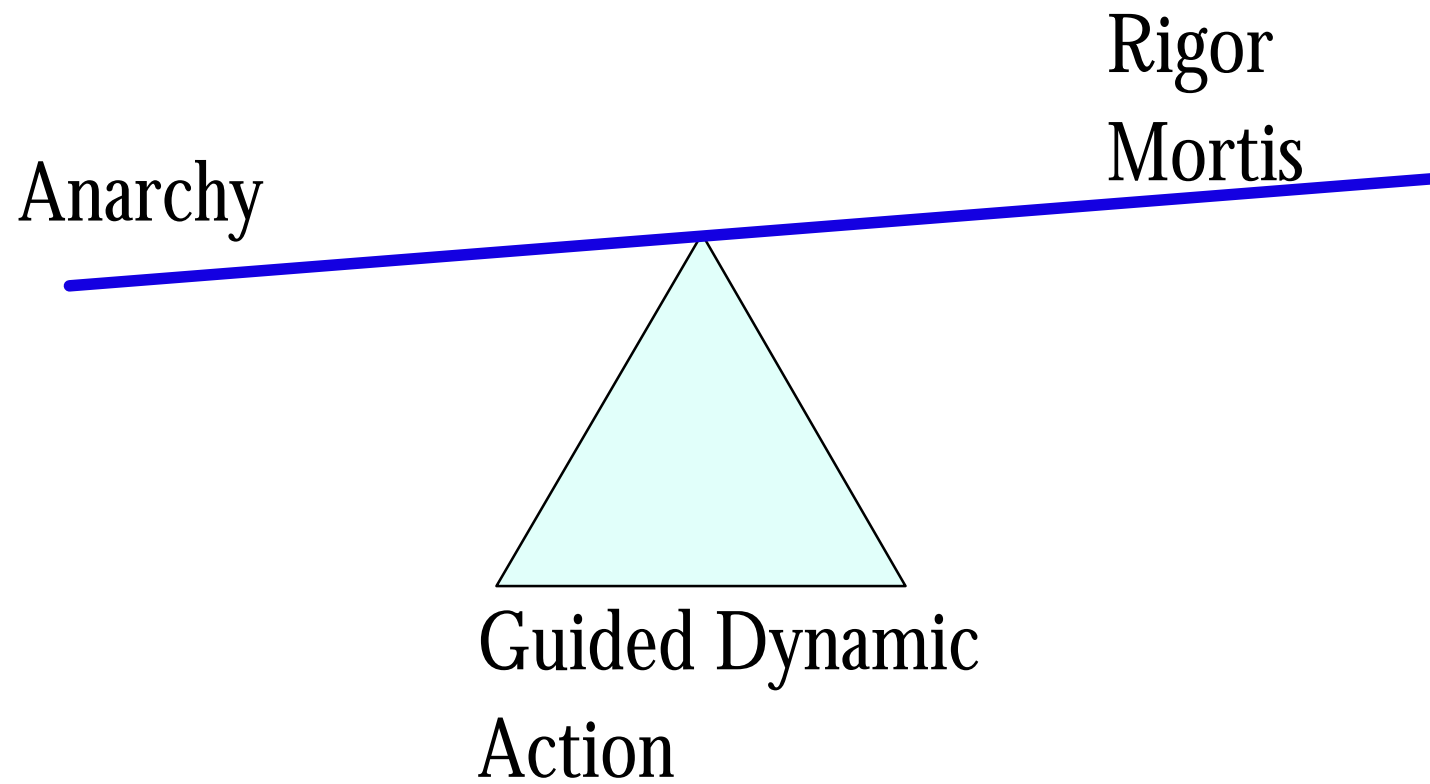
---

- ▲ Complexity is one half of the story
- ▲ also need *comprehensiveness*
  - Lifecycle Coverage
  - Dimensions
  - Integration
- ▲ Adjustment for resource demands
- ▲ Uses
- ▲ Refinement of weights in the model
- ▲ Benchmark = 1 for I.E. improving with time
- ▲ Help me!



# LiveMethod

---



Method Points Graham McLeod  
University of Cape Town June '97

