Implementing Metrics with Function Points

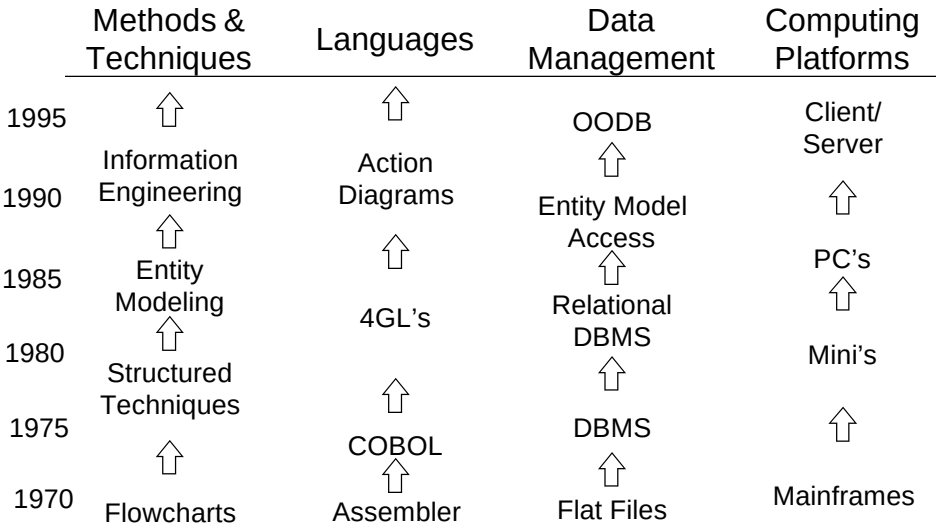
Session 240

Frank Mazzucco
Texas Instruments

Agenda

- Measurement drivers
- Function Points
- Using Composer Function Point report
- Metrics implementation techniques

Evolution of I/S Development



Quality and Productivity Trends

- Annual software productivity improvement: 4 - 7%
  - Barry Boehm
- Less than one company in five measuring software productivity and quality
  - Howard Rubin
- 76% of assessed software organizations at the initial (ad hoc / chaotic) level of software maturity
  - Software Engineering Institute, Carnegie-Mellon University)
- Many well-publicized "software disasters"

Measuring Productivity and Quality

- Historically, very little to measure
- Drivers for a new emphasis on measurement
  - Application of TQM techniques to software
    - » e.g., Malcolm Baldrige Criteria, ISO 9000-3
  - Software process improvement
    - » e.g., Software Engineering Institute (SEI)
  - Maturity and acceptance of model-based development tools
    - » e.g., Composer by IEF
  - Objective, non-technical, reproducible measures of size
    - » e.g., Function Points

Agenda

- Measurement drivers
- ➢ Function Points
- Using Composer Function Point report
- Metrics implementation techniques

Function Points

- Measure of *work-product* (or, relative size) of a software application or project
  - Based on IBM study of key project variables
  - Originally proposed by Allan Albrecht in 1979
- Synthetic metric based on user functionality
  - Measure functions requested/received
    - » Inputs, Outputs, Inquiries, Files, Interfaces
  - Components weighted by complexity
  - Total adjusted by system-level factors
    - » 14 General Systems Characteristics

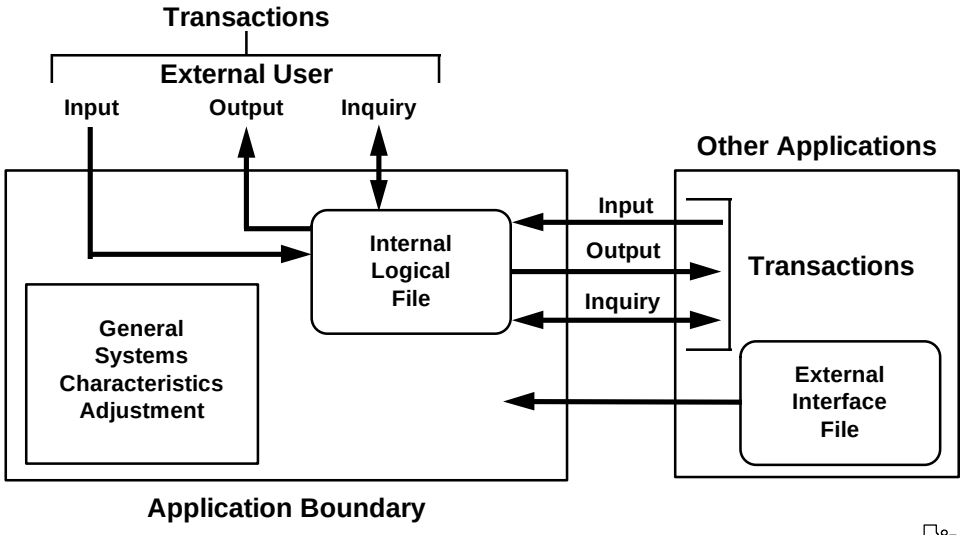
Applications of Function Points

- As a technology-independent component of software metrics
  - Development productivity (function points per person-month)
  - Maintenance productivity (function points supported per full-time maintenance staff)
  - Defect density (defects per function point), etc.

Applications of Function Points (cont.)

- To validate project estimates
  - Can be measured early in the project lifecycle (early measures are estimates which must be updated at project completion)
  - Early use allows validation based on project history (or, with caution, industry averages) on projects with *similar attributes* of the estimate developed from the *detailed project plan*
- As a standard means of communication among project managers, software users, and management

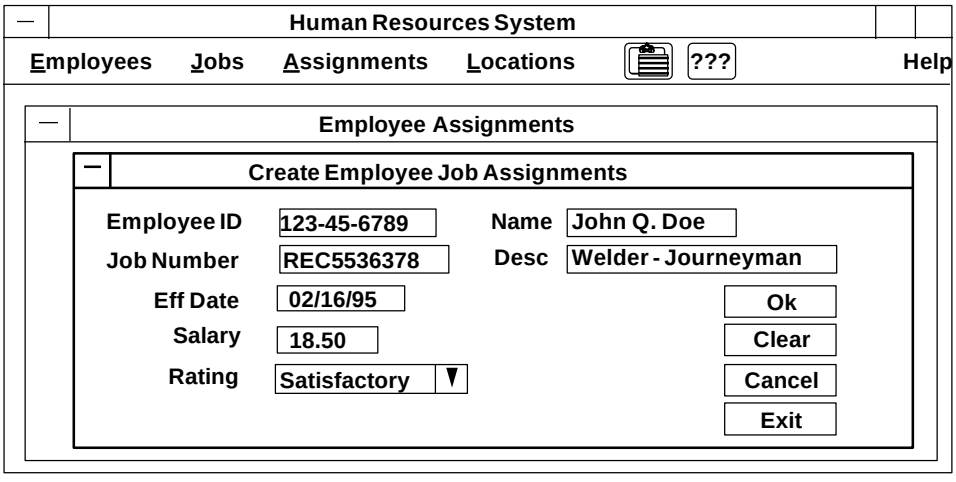
Components of Function Point Analysis



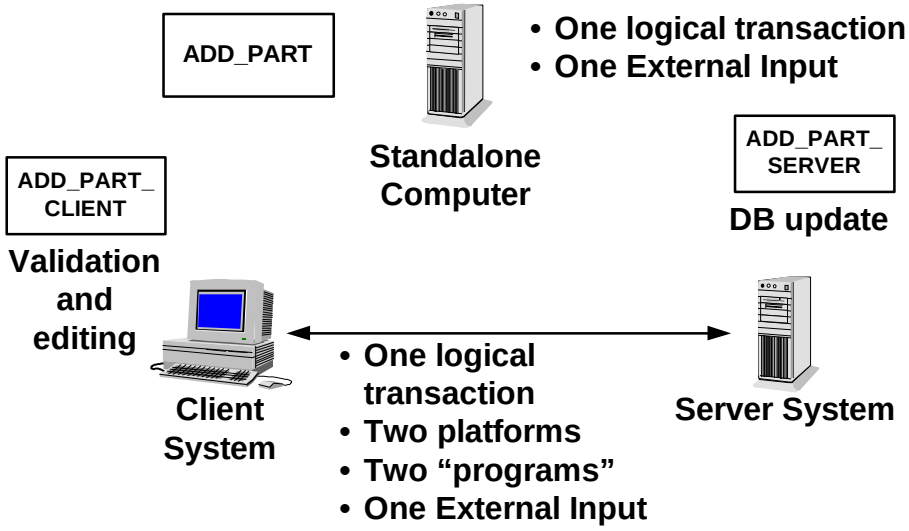
Applicability of Function Points

- FPs count “logically, from the user’s point of view”
- Logical user data and transactions–not dependent on implementation technique
- Can be easily applied to new development technologies
  - – GUI’s
  - – Client/Server
  - – Object-Oriented
- Logical user data ≈ Entity type (Files, Interfaces)
- Logical user transaction ≈ Elementary process (Inputs, Outputs, Inquiries)

FP Example–GUI



FP Example–Client/Server



Why Function Points?

- Independent of the technology used
  - Well-suited to measuring impact of new technology
- Can be used early in project lifecycle
- Can be used to validate project estimates
- Reproducible
  - ±10% accuracy verified by MIT research
  - Accuracy can be much higher in controlled environments
- Supported by active, worldwide user group (IFPUG)

International Function Point Users Group

- Non-profit organization - promotes and supports Function Points and related metrics
  - 600+ member companies worldwide
  - 11 international affiliate organizations
- Membership services
  - Counting practices and case studies
  - Certification
  - Management reporting
  - Measurement start-up
  - Conferences and workshops
  - Hotline support
  - WWW home page (http://www.ifpug.org/ifpug)

Contact IFPUG, (614) 895-7130, for additional information

Agenda

- Measurement drivers
- Function Points
- Using Composer Function Point report
- Metrics implementation techniques

Using the Composer Function Point (FP) Report

- Composer counting rules (simplified)
  - Elementary processes (or action blocks) counted as Inputs, Outputs, or Inquiries
  - Entity types counted as Files or Interfaces based on usage
  - Classification and complexity based on actual usage in action diagrams
- Matches the spirit of IFPUG 4.0 rules quite well
- Must be adjusted to account for:
  - Differences in development methods
  - Objects unknown to Composer
  - General systems characteristics

Using the Composer FP Report (cont.)

- Adjusting for development method differences
  - Composer assumes:
    - » Action block (w/entity actions) = Business function (EI, EO, or EQ)
  - "Elementary process level or equivalent" – for both Analysis and Design report options (Note: Design is used most often)

Using the Composer FP Report (cont.)

- Adjusting for development method differences (cont.)
  - Consistent with Composer Method, but implementation variants can cause discrepancies
  - Examine action block hierarchy
  - Remove (manually) non-business function AB's
    - »e.g., paging commands
  - Aggregate (manually) partial business function AB's into single inputs, outputs, or inquiries
    - »e.g., "modular I/O," validation routines

FP Report–Activities

```
MODEL NAME:            CORPORATE_ORDER_PROCESSING
BUSINESS SYSTEM:  ORDER_MAINTENANCE

ACTION BLOCKS:
XXXXXXXXXXXX
 XXXXXXXXXXXXXX
 XXXXXXXXXXXXXX
ORDER_LINE_MAINTENANCE
 CREATE_ORDER_LINE
 DELETE_ORDER_LINE
 PAGE_BKWD_ORDER_LINE
 PAGE_FWD_ORDER_LINE
 READ_ORDER_LINE
 UPDATE_ORDER_LINE
 VALIDATE_PRODUCT_CODES
XXXXXXXXXXXX
 XXXXXXXXXXXXXX
```

Paging commands–should be deleted

Validation routine - delete or combine with add/update functions

FP Report–Activities (cont.)

```
MODEL NAME:            CORPORATE_ORDER_PROCESSING
BUSINESS SYSTEM:  ORDER_MAINTENANCE

ACTION BLOCKS:
XXXXXXXXXXX
 XXXXXXXXXXXXX
ORDER_LINE_MAINTENANCE
 ASSOC_ORD_LINE_WITH_ORDER
 ASSOC_ORD_LINE_WITH_PRODUCT
 CREATE_ORDER_LINE
 DELETE_ORDER_LINE
 READ_EACH_ORDER_LINE
 READ_ORDER
 READ_ORDER_LINE
 READ_ORDER_LINE_FOR_UPDATE
 READ_PRODUCT
 UPDATE_ORDER_LINE
 UPDATE_PRODUCT
```

Typical “modular I/O” action blocks–should be combined with other AB’s or deleted

(*Note*: If the entire model has been built this way, it may be preferable to do a manual function point count.)

Using the Composer FP Report (cont.)

- Adjusting for development method differences (cont.)
  - Examine entity types for possible aggregation
    - »Files (ILFs) and interfaces (EIFs) must be maintained independently and *may* consist of multiple entity types
    - »Examine attributive and associative entity types and combine (manually) where necessary

FP Report–Data

```
MODEL NAME:            CORPORATE_ORDER_PROCESSING
BUSINESS SYSTEM:  ORDER_MAINTENANCE

FILES/ENTITY TYPES:        FILES    INTERFACES
                         S  A  C    S  A  C

CUSTOMER                   1
ORDER                      1
ORDER_LINE                 1
ORDER_LINE_DESCRIPTION     1
PRODUCT
XXXXXXXXXXXX
XXXXXXXXXXXX
XXXXXXXXXXXX
```

This is likely an attributive entity type of ORDER_LINE, and not maintained independently–should be combined with ORDER_LINE as one internal logical file

Using the Composer FP Report (cont.)

- Adjusting for unknown objects
  - External databases
    - » e.g., DL/1 database referenced by external action block
  - Activities implemented outside Composer
    - » e.g., reports produced with a 3GL or 4GL
- Including 14 general systems characteristics
  - Calculate adjusted function point count

Spreadsheet option can be helpful - but be aware of missing procedure step indent!

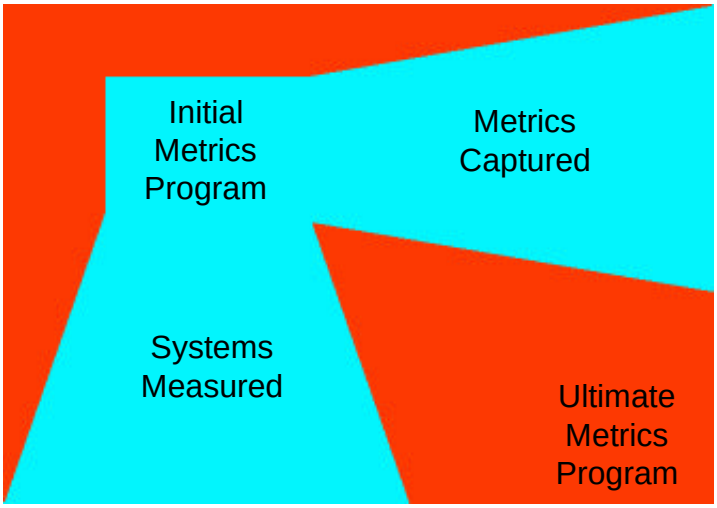
Using the Composer FP Report (cont.)

Model Characteristic	User function = elementary process	Some user functions as EPs, others split into CABs		Structured programming AB 'modules'
FP Report Discrepancy	**10-25%**	**30-70%**		**70-200%**
FP Report Useful?	**Yes**	**Probably**	**Maybe**	**No**

Agenda

- Measurement drivers
- Function Points
- Using Composer Function Point report
- Metrics implementation techniques

Metrics Implementation



Metrics Implementation Plan

Near-term (≈ 6 months)

- Obtain management commitment
- Join IFPUG and/or local user group
- Train 1-2 people in function point analysis
- Count 1-3 new applications as metrics pilots
- Measure:
  - Development and enhancement productivity (FP/person-month)
  - Maintenance productivity (FP/full-time equiv. maintenance staff)
  - Defect density (defects/FP @ release + 6 months)
- ⇨ Demonstrate results to management

Metrics Implementation Plan (cont.)

Intermediate Term (≈ 1-2 years)

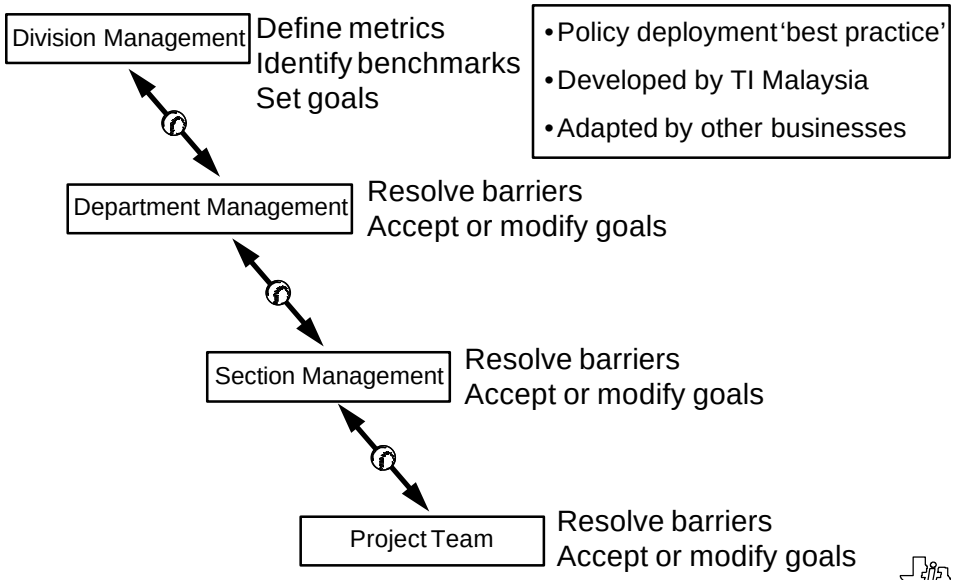
- Train additional staff in function points
- Deploy metrics program to all new development
- Begin counting existing systems for a baseline
- Consider a software process assessment
- Add one or more metrics
  - Cycle time (elapsed time/FP)
  - On-time project completion (actual vs. plan)
  - Customer satisfaction (user survey)
- ⇨ Demonstrate results to management
  - Use best practices to align metrics with business goals and objectives, such as:
    - » Goal-Question-Metric paradigm, Catchball process

Goal/Question/Metric Paradigm

- Overall goals of the measurement effort:
  - object, environment, purpose, perspective
    - »Assess the reliability of a software product as perceived by the customer using it.
- Quantifiable questions that support each goal, such as: Does the customer associate reliability with the product failures?
- Quantifiable data to answer the questions
  - easy to collect, calculate, and understand
    - »Failure density
    - »Mean time between failures

Reference: Basili, V.R., and D.M. Weiss, "A Methodology for Collecting Valid Software Engineering Data," *IEEE Transactions on Software Engineering* , November 1984

Catchball Process



Metrics Implementation Plan (cont.)

<u>Long-term</u> (≈ 3-5 years)

- Establish a metrics repository
- Initiate metrics benchmarking with other companies
- Deploy all metrics to all projects, all systems
  - Milestone completion (% on-time)
  - Return on investment
  - Risk analysis
- ⇨ Demonstrate results to management

Implementation FAQs

- Centralized or distributed metrics function?
  - How are other, similar functions in your company organized?
- How many people should be trained?
  - Distributed function – 1-2 per project team
  - Centralized function – ≈ 2 - 4% “overhead”
  - Note: phase implementation allows phased training
- How many hours in a person-month?
  - Varies by company based on HR policies, overtime, etc.
  - Typically 130 to 140 hours

Implementation FAQs (cont.)

- No formal time-reporting system?
  - Use equivalent person-months
- Lacking support, resources, infrastructure for a baseline study?
- Compare new projects against industry averages (with some caution)
  - e.g., ≈ 8 FT/PM for new MIS development
- Build baseline gradually as projects complete

Summary

- Model-based development can be successful if you:
  - Manage the implementation
  - Measure the results (implementing gradually, if necessary)
- Function points are a useful and industry-accepted technique for software size measurement
- Composer function point report may be useful; but, if not, counting manually is a valid alternative
- *You may already be* a success story!
  - So, why not find out?

Implementing Metrics with Function Points

Session 240

Frank Mazzucco
Texas Instruments