Function Point Analysis for the OO-Jacobson Method: A Mapping Approach

Thomas Fetcke, Alain Abran and Tho-Hau Nguyen

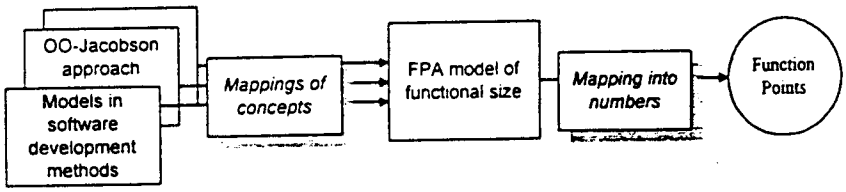
Université du Québec à Montréal
Software Engineering Management Laboratory
C. P. 8888, succ. Centre-Ville
Montréal (Québec) Canada H3C 3P8
Phone: +1 (514) 987 3000-8900
E-mail: fetcke@cs.tu-berlin.de, abran.alain@uqam.ca
http://saturne.info.uqam.ca/Labo_Recherche/lrgl.html

Introduction

- Objective: Application of Function Point Analysis (FPA) to object-oriented software development.
  - Use of the IFPUG standard "as is".
  - Focus on the OO-Jacobson *use case* approach.
  - Counting in early lifecycle phases.
- Challenges with OO methods:
  - Model concepts differ from traditional methods.
  - Different OO methods differ in the models used, particularly early in the lifecycle.

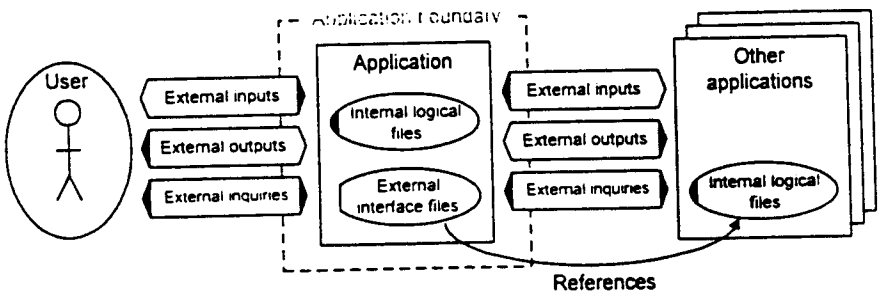
Mapping of Concepts

- FPA concepts form a model of functional size.
- IFPUG counting rules define the elements of this model and the mapping into numbers.
- We propose a mapping of the OO-Jacobson concepts into the model of functional size.
- The mapping does not change the FPA model.



Function Point Model

- The key elements in the FPA model are:
  - Transactional function types,
  - Data function types (files).



Function Point Analysis Steps

The components of the FPA model are identified in four major steps:

1. Determination of the counting boundary,
2. Identification of items within the boundary,
3. Classification of these items,
4. Assignment of weights for the items.

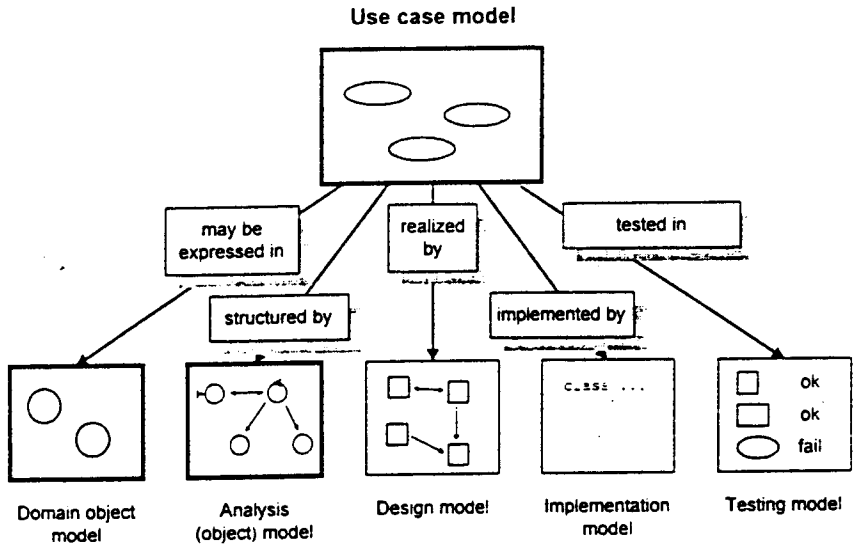
The mapping is unaffected by, and does not cover:

- The type of count (project, application),
- The general system characteristics.

OO-Jacobson Approach

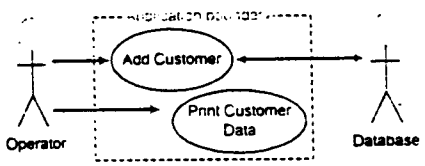
- The OO-Jacobson approach develops a sequence of models from requirements to implementation.
- The use case model is the central starting point.
- The requirements analysis produces:
  - the use case model, and
  - the domain object model.
- The robustness analysis structures the use cases in the analysis model.

OO-Jacobson Approach Models



Step 1: Boundary Concepts

- The application boundary determines the object of the measurement.
- The use case model includes this boundary concept.
- Actors represent users and other applications.
- Use cases represent the functionality.
- Actors representing underlying systems or hardware are rejected as users.



Step 2a: Transactional function types

- The corresponding concept is the use case concept.
- However, there is no one-to-one mapping:
  - Different flows of interaction in one use case each represent candidate transactions.
  - Use case *extensions* are candidate transactions.
  - *Abstract* use cases are rejected as candidates.
  - Use cases related to actors rejected in step 1 are not counted as transactions.

Step 2b: Data function types

- The corresponding concept is the object concept.
- Our proposal considers two object models:
  - The domain object model.
  - The analysis (object) model.
- Domain objects represent all (data) concepts which are relevant for the application:
  - Data entities which correspond to files.
  - Application environment, users or other systems that do not represent files.

Step 2b: Analysis Objects

- The analysis model structures the use cases with typed objects:
  - entity objects,
  - interface objects,
  - control objects.
- Entity objects model information existing in the system for a longer time.
- Entity objects are candidates for data function types.

Step 3: Classes of Items

- The standard Function Point counting rules are applied to the candidates from step 2.
- Candidates from step 2 may be rejected in step 3.
- Step 3a: The candidate use cases are classified into external inputs, external outputs and external inquiries.
- Step 3b: The candidate objects are classified as internal logical files or external interface files.
- The evaluation of the rules is based on information from the requirements and analysis models.

Step 4: Weights of Items

- Weights are based on the structure of items.
- For data function types:
  - Data elements correspond to object attributes.
  - Records are determined by the user view.
- For transactional function types:
  - Data elements correspond to object attributes.
  - File references are determined by references to objects counted as data function types.

Counting Experiments

- Three ongoing industry software projects have been counted with the proposed mapping rules.
- Calculated sizes in Unadjusted Function Points:

Project 1	265
Project 2	181
Project 3	215

- Documentation of Project 1 provided detailed models, from which weights were calculated.
- Projects 2 and 3 did lack some detail in models, some weights have therefore been estimated.

Conclusions

- FPA was applied to OO software development.
- The IFPUG standard was used.
- The FPA model was not found invalid for OO software development.
- The count is based on requirements model documents from the early lifecycle phases.
- The approach is formulated in a set of mapping rules that support the counting process.
- The mapping rules were applied to three industrial software projects.

References

- Australian Software Metrics Association: *Sizing in object-oriented environments*, 1994.
- Fetcke, Abran and Nguyen: *Mapping the OO-Jacobson approach into Function Point Analysis*, Proc. TOOLS USA 1997.
- Gupta and Gupta: *Object Point Analysis*, IFPUG Fall Conf. 1996.
- International Function Point Users Group: *Function Point Counting Practices Manual*, Release 4.0, 1994.
- Jacobson, Christerson et al.: *Object-oriented software engineering* Addison-Wesley, 1992.
- Whitmire: *Applying Function Points to object-oriented software models* in Keyes: Software engineering productivity handbook, McGraw-Hill, 1992.