iTip	Guidance from the Functional Sizing Standards Committee on topics important to you	

Shared Data
Real-time Responses

iTip # 06 (Version 1.1 *10/19/2014)*

iTips provide guidance on topics important to the FPA community. They explain the application of IFPUG FPA method in a particular situation. iTips are not rules, but interpretation of the rules, and provide guidance using a realistic example to explain the topic being covered.

This iTip is focused on describing the IFPUG FPA method as it applies to data sharing in a real-time environment from the perspective of the application providing the data. This iTip includes a series of examples but is not an exhaustive examination of the subject. For further examples, please see the current CPM and other Shared Data related iTips.

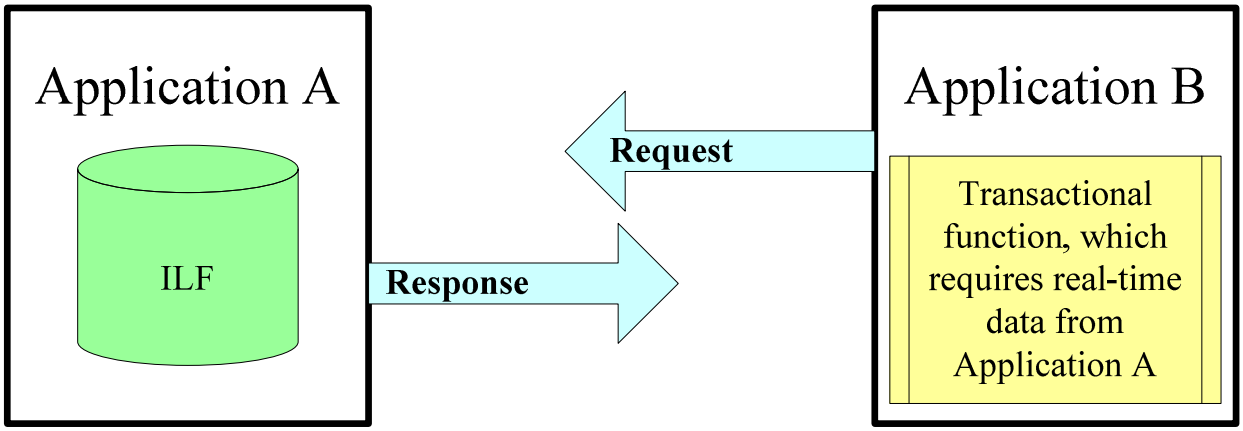
Background

Chapter 3 in Part 3 of the current CPM addresses the sharing of data between applications in a number of scenarios, but does not address the real-time environment. This iTip provides additional guidance for counting the exchange of data through implementations such as APIs, stored procedures and Web Services. The examples provide focus on situations where Application A has a functional requirement to provide data to Application B. Application A is the application being measured. For the purposes of these examples it is assumed that the responses contain no derived or calculated data and that no ILFs are maintained. In all cases counting responses should be based on functional user requirements. Please refer to the SNAP manual for further discussion as to what would be considered non-functional user requirements.

Example 1: Real-time Data Request/Response

Application B requires data from Application A to complete a real-time transactional function. Application B uses the data to complete transactional processing (e.g., to display data on a screen) in Application B.

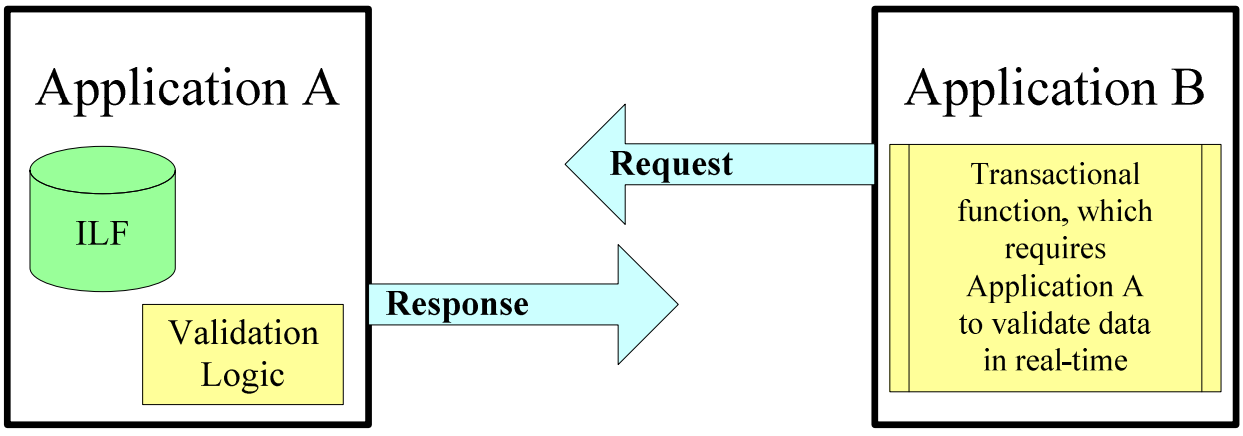
In order to obtain the required data, Application B sends a request to Application A. Application A processes the request, accesses its data and sends a response with the required data to Application B.



From Application A's perspective, there is a functional user requirement to provide data to Application B. The primary intent of this function is to present data to Application B (i.e., one of its users). Based on the primary intent, Application A counts an EQ. The complexity of Application A's EQ is determined based on the number of logical files referenced (i.e., FTRs) and the number of DETs crossing (i.e., entering or exiting) the boundary.

Example 2: Real-time Data Validation Request/Response

Application B processes a transaction that requires Application B to validate employment information. Since Application A owns and maintains Employee Data, this is accomplished by Application B sending a request to Application A to verify that an individual is a current full time employee. Application A accesses its Employee Database and sends a response with the results of the validation to Application B. The code for the validation resides in and is maintained by Application A. Application B uses the response to complete its processing.

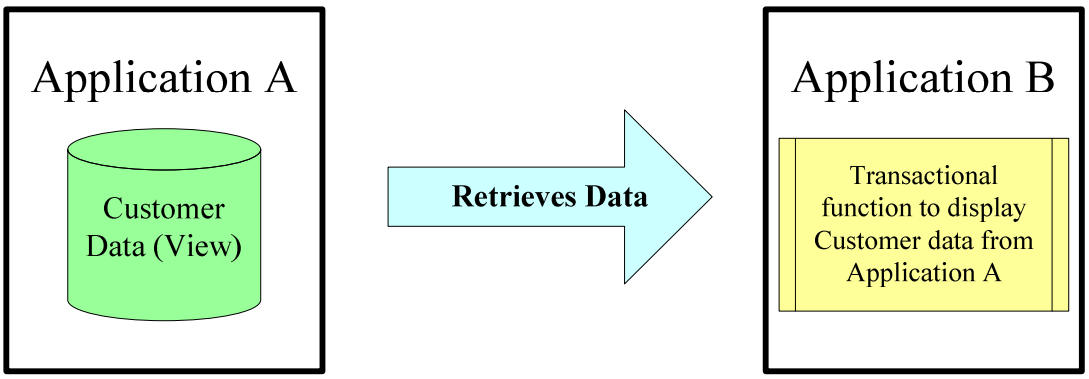


From Application A's perspective, there is a functional user requirement to search its Employee Database and return information to Application B (i.e., one of its

users) based on the request. The primary intent of this function is to present information to Application B. The response to the validation request provided is either the success or failure of retrieving the name and identification provided by Application B. Based on the primary intent, Application A counts an EQ. The complexity of Application A's EQ is determined based on the number of logical files referenced (i.e., FTRs) and the number of DETs crossing (i.e., entering or exiting) the boundary.

Example 3: Database View Created for Reference

Application B presents data that is owned and maintained by Application A to the user in an on-line query. Application A provides a database view that Application B uses to reference the data. In the implementation of this requirement, Application A creates a database view of its Customer data, filtering the data so that Application B can reference a specific subset. This view of Application A's data is created and maintained specifically for Application B; this view is not utilized in any of Application A's other transactional functions.



From Application A's perspective, there is a functional user requirement to provide specific data to Application B. The Customer database view is created solely to fulfill this requirement (i.e., the view is not used by Application A for any other purpose). The primary intent of this function is to present information to Application B based on provided requirements (i.e., filtering). The data attributes (i.e., DETs) in the database view logically cross (i.e., exit) the Application A boundary. Application A counts the database view as an EQ . The complexity of Application A's EQ is determined based on the number of logical files referenced (i.e., FTRs) and the number of data attributes (i.e., DETs) crossing Application A's boundary.

If the view is provided solely for performance reasons, it would be considered to be a non-functional implementation. For additional perspectives on how to measure non-functional requirements, the reader is referred to the SNAP (Software Non-Functional Assessment Process) framework at www.ifpug.org.

Summary

While this iTip illustrates data sharing scenarios specific to a real-time environment, the approach is intended to be technology-independent and can be applied to many technologies and platforms. These examples translate a number of scenarios of "how data is provided" back to the focus of "what function is provided" per the Functional User Requirement. In all cases, the counting interpretation is based on Application A's user view and functional requirements. Analyzing the primary intent is key to that determination. In all of these examples, the primary intent for Application A is to provide data to Application B (i.e., one of Application A's users) in response to a real-time request. As a result, in each example an EQ is counted within the Application A boundary, regardless of "how" the data is physically provided.

Frequently Asked Questions (FAQ)

1. What happens when other applications in addition to Application B use the same interface to request or validate data?

   In this variation, Application A counts only one EQ regardless of the number of applications that invoke the interface.

2. What happens when there are multiple APIs, Web Services or Stored Procedures within Application A to provide or validate data?

   Each access point receiving a request from another application is candidate for a separate elementary process within Application A that must be evaluated for uniqueness. If the same data is provided via different physical implementations (e.g., an API and a Web Service require the same set of DETs and the same set of FTRs and the same set of processing logic to satisfy the request), they should not be counted separately.

   The additional means of request may also be considered to be a non-functional implementation. For additional perspectives on how to measure non-functional requirements, the reader is referred to the SNAP (Software Non-Functional Assessment Process) framework at www.ifpug.org.

3. What if the response includes derived data?

   Application A would count the response as an EO.

4. What if Application B directly accesses Application A's ILF in real-time?

   Application A is passive in this sharing and there is nothing to count.

5. How does Application B count its request to Application A for data or Validation?

   Please refer to iTip #5 Shared Data – Real-time Requests.

Further Reading

IFPUG Counting Practices Manual, Part 1, Section 5.5 – Measure Transactional Functions

IFPUG Counting Practices Manual, Part 2, Chapter 7 – Measure Transactional Functions

IFPUG Counting Practices Manual, Part 3, Chapter 3 – Shared Data

IFPUG iTip #5 Shared Data – Real-time Requests

IFPUG APM 2.1 Software Non-functional Assessment Process Manual (SNAP)

IFPUG offers iTips at no charge to the international function point community to stimulate the further promulgation and consistent application of the IFPUG FPA Method. IFPUG would appreciate if you or your organization would support IFPUG in its mission by becoming a member. For further information please visit www.ifpug.org or send an email to ifpug@ifpug.org. IFPUG thanks you for your support.