FiSMA 1.1 Functional Size Measurement Method Guideline for Multi-layer Architecture

1. General principles

Software's functional size measurement is usually measured by one application at the time. Therefore we apply an *application decomposition model*, which divides the functional services of the software into applications from functional size measurement point of view. According to this application principle, each application is measured separately taking into account the *messaging between the layers*.

There are usually several issues to consider in deciding the application decomposition. The purpose of the measurement may affect the application borders. In some cases project's situational factors, like project's operational environment or development project organization, or technology applied may be the ground for dividing the software into applications in the measurement. System architecture, deployment on different, technical platforms and development tools may require different kind of measurement approach e.g. because measurement results are to be used in the context of productivity assessment.

The single application in the decomposition has to integrate to some of the other applications via a *referencing mechanism*. On the other hand, the software's decomposition model cannot have an application with no other identified FiSMA 1.1 base functional component (BFC) in addition to software's internal messaging. So inside the software measurement border, the decomposition may not have an application with functionality consisting only of messages from and/or to other applications.

Application measurement pattern is a predefined generalized decomposition model dividing the functional services into applications from functional size measurement point of view. This kind of pattern is a default decomposition of functionality, usually made on logical and high level separation of concerns regarding software's functional services.

2. Multi-layer architecture measurement pattern

A general model to measure software of multi-layer architecture system is a *3-layer measurement pattern*. The system consists of three partitions (applications), which are (figure 1)

- User interface
- Business logic
- Data access.

User Interface layer is also called presentation layer. The end-user interface functionality consists of ten base functional component (BFC) types, seven of them for navigation and query, and three for input services. Three BFC types (menu, selection list, icon) of the navigation and query services are not considered to need any interaction with the business logic layer, but the seven other BFC types typically do (login functions, data inquiries, generation screens, browsing screens, and 1-, 2- and 3-functional input screens).

Business Logic Layer is sometimes also called application layer. The functions of the business logic layer usually consist of algorithmic services, interfaces and output services.

Data Access Layer holds and manages the data entities. Physical implementation of this layer is often a database service.

All the end-user functions performed on the user interface layer requiring interaction with the business logic layer, and all the functions on the business logic layer requiring interaction with the data access layer are considered to use additional functions for this kind of communication. The base functional components of FiSMA 1.1 method applied to internal (i.e. inter-layer within the system to be measured) communication are

- Messages from others applications
- Messages to other applications.

The functionality of a message to other layer is naturally assigned to the layer responsible for sending it. In the same way functionality of a message from other layer is assigned to the layer responsible for receiving it.

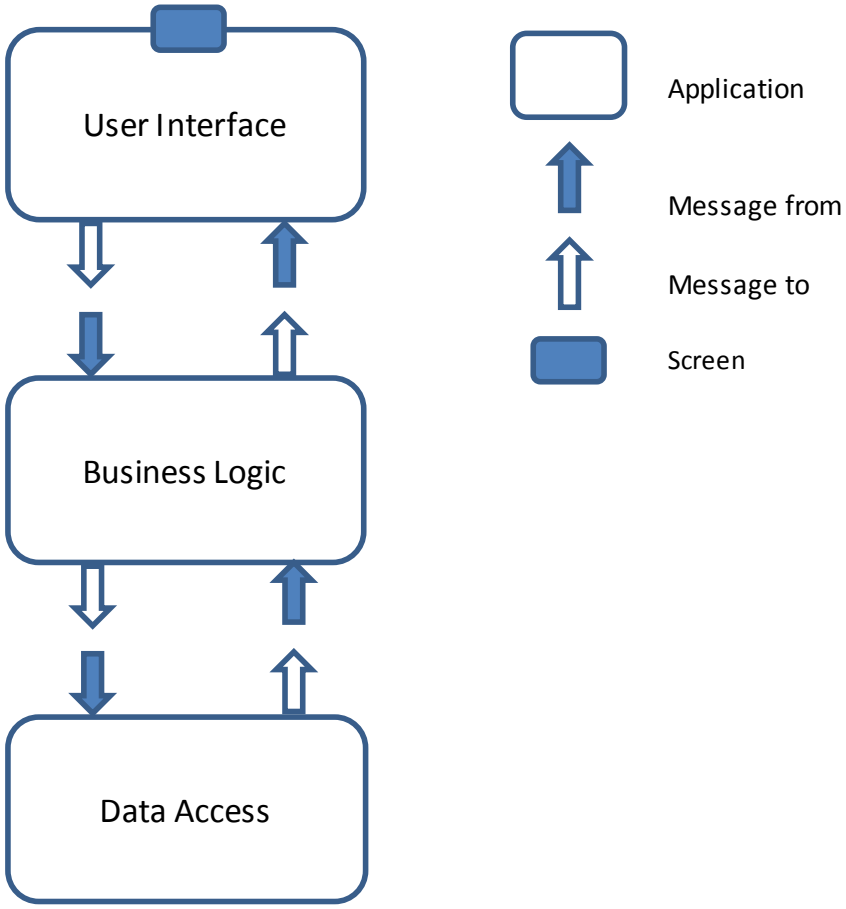

Figure 1. Application measurement pattern for multi-layer architecture.

3. Measurement of User Interface Layer

If a detailed specification is available, you can measure the user interface functions normally, according to the method standard and FiSMA 1.1 Functional Size Measurement Method Guideline for Graphical User Interface (Guideline_for_GUI_FiSMA11.pdf).

If there is no detailed specification available, recommended default values for UI functions' numbers of read and write references are:

- Selection list, icon or menu: 1 reading reference
- Query, browsing or generation screen: 2 reading references
- Input screen: 1 reading and 1 writing reference.

Each independent browsing, query or update function (dialog or screen) is considered to send a unique message to and from the business logic. The number of data elements in the message is the number of data elements in the user interface function.

The calculation parameters for the user interface *layer* ***internal messages*** are determined according tables 1 and 2.

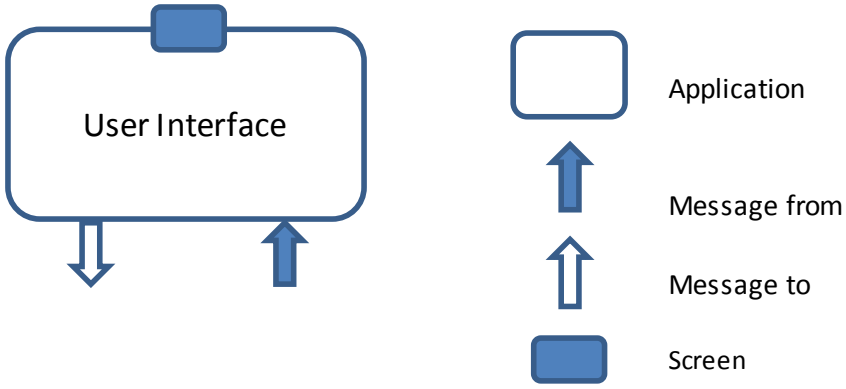

Figure 2. User interface layer messages.

Table 1. Message to the business logic layer.

Parameter	**Value** [1]
Number of data elements	Number of data elements of user interface function
Number of reading references	1

Table2. Message from the business logic layer.

Parameter	**Value**[1]
Number of data elements	Number of data elements of user interface function
Number of reading references	1
Number of writing references	1

[1] Value in this and all other tables within this chapter represent the most common practice based on logical user needs. However, exceptions may occur both in the number of data elements and the numbers of references.

4. Measurement of Business Logic Layer

Algorithms and reports are measured normally.

The possible ***external*** *integration* ***messages***, joining the system with other systems, are normally measured with the same BFC types as the internal messages. However, these external messages are dealt case by case, and there is no recommendation for their default values here. For business logic layer messaging on the contrary, the UI function and entity dependent and default values of calculation parameter are proposed in the tables below (tables 3 – 6).

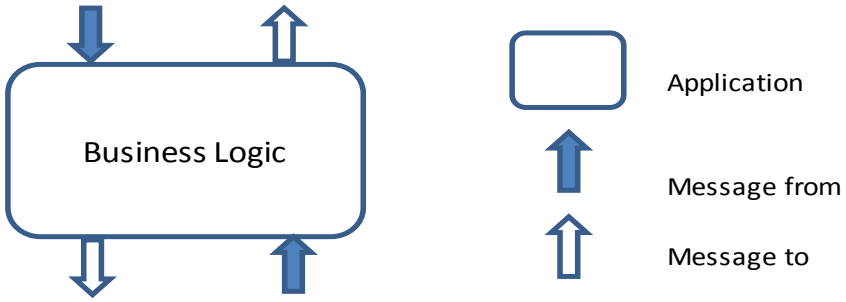

Figure 3. Business logic layer messages.

Table 3. Message from the user interface.

Parameter	**Value**[1]
Number of data elements	Number of data elements of user interface function
Number of reading references	1
Number of writing references	Number of messages to data access layer required to satisfy needs of the message from user interface layer

Table 4. Message to the user interface.

Parameter	**Value**[1]
Number of data elements	Number of data elements of user interface function
Number of reading references	Number of messages from data

	access layer required to satisfy needs of the message to user interface layer

Table 5. Message to the data layer.

Parameter	**Value**[1]
Number of data elements	Number of data elements of entity
Number of reading references	2 (message from user interface layer + system entity)

Table 6. Message from the data layer.

Parameter	**Value**[1]
Number of data elements	Number of data elements of entity
Number of reading references	1 (system entity)
Number of writing references	1 (message to user interface layer)

5. Data Access Layer Messaging

The entities are measured normally according to the logical data model.

Between the business logic and data access layer, each entity in the logical data model is considered to have a unique message to and from the data access layer. The number of data elements in the message is the number of attributes in the entity.

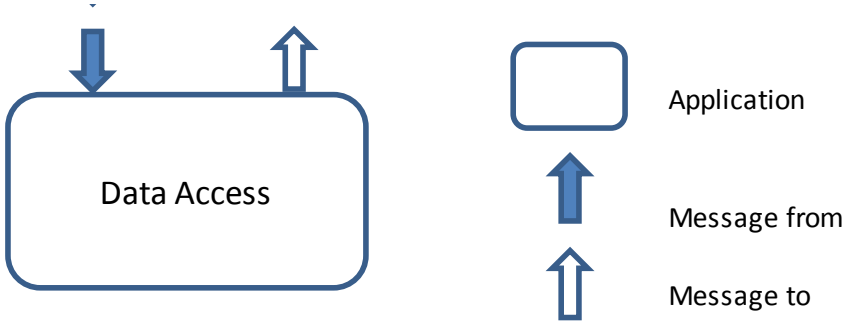

Figure 4. Data access layer messages.

Table 7. Message from the business logic.

Parameter	**Value**[1]
Number of data elements	Number of data elements of entity
Number of reading references	0
Number of writing references	1

Table 8. Message to the business logic.

Parameter	**Value**[1]
Number of data elements	Number of data elements of entity
Number of reading references	1

6. Terminology

Generally, a tier is a row or layer in a series of similarly arranged objects. In software architecture, the parts of a program can be distributed among several tiers, each located in a different computer in a network. Such a program is said to be multi-tiered. Terms 'tier' and 'layer' are often used interchangeably. Specialists usually *refer by tier to physical layout* or deployment of software components. *Layer is logical separation* of software's functional services.

From FSM point of view in measurement of service-oriented type of software, we are more interested in the functionality and its orientation. Therefore we prefer in this document term 'layer' instead of term 'tier'. We define layer here as a partition of functional software components.

Multi-layer systems consist of two or more layers (i.e. separate applications), which usually interact with each other through internal messages. All internal messages are considered to be within the system border. When the system communicates with other systems, the messages are called external messages. NOTE! In certain cases the interaction between the layers of a system or interaction between two systems may be made through signals or batch records instead of on-line messages.