Function Points Analysis as a tool for Requirement Analysis (NGC-NGD)

Prakash SK Kumar
Satyam Computer Services Limited
Bangalore Hyderabad Road
Hebbal, Bangalore: 560024, India

www.satyam.com
Presentation Layout

- Paper Concept
- Features of Good Requirements
- Leveraging FP guidelines for Requirement Analysis
  - Extrapolating the Objectives of Function Points
  - Exploiting the User View
  - Using the Identification of Data Functions
  - Using the Identification rules of Transaction Functions
  - Interpreting the General System Characteristics
- Conclusion
FPA requires that the practitioner’s use “user requirements” for sizing.

The biggest problem today software community is facing is managing user requirements.

Then why not use Function point principles as basis for capturing functional requirements.

One line which sums up this paper is “What is Not Good for Counting (NGC) is Not Good for Developing (NGD).

The presentation is based the experience of working with more than 300 projects as estimation specialists, auditors and process consultants.
Paper Concept

- How to leverage principles of FP to better requirements.
- Using each rule of FP to bring clarity into requirements
- Capturing non functional requirements through FP
- Is the same language used in FP and Requirements Analysis?
- FP as process improvement tool for Requirement Analysis Process
Characteristics of Requirements

- Requirements are the basis for any software development or for that matter for any development activity.
- Capturing requirements is the most important and influential stage in software development is an understatement.
- The word influential is used because requirements impact all the remaining phases of software development.
- It’s like diagnosis in medical field. If the diagnosis of the doctor is not correct then the treatment would be wrong. It can also be fatal.
- A requirement is defined as a property that must be exhibited in order to solve some real-world problem.
- Defining this some real world problem in an unambiguous way is our problem on hand.
Characteristics of Requirements

- Necessary
  - The stated requirement is an essential capability, physical characteristic, or quality factor of the product or process. If it is removed or deleted, a deficiency will exist, which cannot be fulfilled by other capabilities of the product or process.

- Verifiable
  - The stated requirement is not vague or general but is quantified in a manner that can be verified by one of these 4 alternative methods: inspection, analysis, demonstration or test.

- Feasible
  - The stated requirement can be achieved by one or more developed system concepts at a definable cost. This implies that at least a high level conceptual design has been completed and cost tradeoff studies have been conducted.
Characteristics of Requirements

- Complete
  - The stated requirement is complete and does not need further amplification. The stated requirement will provide sufficient capability. The state set of requirements leave the system is a stable state once the activity or transaction is achieved.

- Consistent
  - The stated requirement does not contradict other requirements. It is not a duplicate of another requirement. The same term is used for the same item in all requirements.

- Traceable
  - Requirements should be traceable to smallest level which makes sense to the user. The requirement should be traceable from this level to a level of higher abstraction.
Characteristics of Requirements

- **Concise**
  - The requirement statement includes only one requirement stating what must be done and only what must be done, stated simply and clearly. It is easy to read and understand.

- **Implementation free**
  - The requirement states what is required, not how the requirement should be met. A requirement statement should not reflect a design or implementation nor should it describe an operation. However, the treatment of interface requirements is generally an exception.

- **Unique**
  - The requirement can be identifying uniquely within the system and can be identified by a unique number or set of associated transactions and functions associated with it.
User Requirements

- Functional Requirements
- Non-Functional Requirements
- Technical Requirements

User Requirements

Project Requirements

Implementation Requirements
Function Point Counting Procedure

Application to be Counted → Type of Count & Appln. Boundary → Transaction Functions → Data Functions

Internal Logical Files → External Interface Files

External Inputs → External Outputs → External Queries

Unadjusted Function Point Count

Final Adjusted FP Count

Value Adjustment Factor

14 GSC

© Satyam Computer Services Ltd
Extrapolating the Objectives of Function Points

- Measure functionality that the user requests and receives\(^1\)
  - The objective of measuring what the user requests and receives is so simple but very powerful.
  - This ensures that we don’t deviate from our mission of developing a system that is relevant to the users with features that they desire.
- Measure software development and maintenance independently of technology used for implementation\(^1\)
  - User requirements should be free from implementation.
  - Technology under consideration may do some aspects of functionality slower compared to the other or it may be less secure compared to other environments.
  - So these various aspects of environment may lead to capturing incomplete requirements.
  - Function point analysis is technology averse.
  - This concept of FP goes hand in hand with characteristics of Good requirements which expect that user needs be implementation free.
Identify the relevant stakeholders

The user view concept of function point can be used to filter out those posing stakeholders who don’t have an interest in the development of the system under consideration

User view emphasis on capturing core business functions. This can help us to prioritize the requirements.

User view emphasis that the requirements captured is approved by the user.

User view can vary in physical form. This ensures that we capture requirements from different sources like user manuals, help documents, hand written reports or reports in other paper formats.

This holistic approach towards capturing requirements ensures that we don’t miss any source through which we can capture user needs.
The counting scope defines the functionality which will be included in a particular function point count.

We use this to bring clarity into the end users minds as what can be the logical scope that can be developed.

This step of scoping the requirements ensures that only the necessary requirements are captured and we are on target as per the user needs.

The Function point requirement of drawing a logical boundary would enable us capture complete requirements for the desired business function.
Leveraging the Counting Scope and Application Boundary

- Defines what is external to the application
  - It helps the analyst to ensure that he captures what is required by the system.
  - It also helps him to figure out what is not part of the system.
  - Customer will cooperate in identifying the boundary because it provides him with lot of clarity regarding his requirements.
- Is the conceptual interface between the ‘internal’ application and the ‘external’ user world
  - Identify the external sources from which the requirements should be collected
  - Identify interface points within the application which interact with the external world
  - Helps determine the data formats for interaction with different external systems
  - Provide information to stakeholder of those external systems as to what they can expect from the system under consideration
Leveraging the Counting Scope and Application Boundary

- Acts as a ‘membrane’ through which data processed by transactions (EIs, EOs and EQs) pass into and out from the application
  - The presence of the application boundary helps us to identify the outbound and inbound transactions.
  - Identifying these transactions helps us to capture requirements which are complete, consistent and verifiable because these transactions are expected to leave the system in a stable state.
  - Unless they are complete or consistent or verifiable they cannot leave the system in a stable state.
  - When we ensure that each transaction is complete it leaves us with a set of requirements which are unambiguous and which are absolutely necessary for the system.
  - These transactions also ensure that they are completely traceable from the perspective of what triggered these transactions and whether these transactions are unique in nature.
Leveraging the Counting Scope and Application Boundary

- Is dependent on the user’s external business view of the application. It is independent of technical and/or implementation considerations\(^1\)
  - As we earlier discussed requirement should be implementation free. What it means is requirements should not be biased towards a technology or influenced by a technology.
  - Similarly when we draw an application boundary we are trying to draw the application boundary from logical perspective rather than the technical perspective.
  - What it means is that when we capture requirements we need to capture logically related requirements. This leaves the user needs in a stable and consistent state.
  - The requirements would be independent of time, user, location and relationships. This enables us to capture requirements which are unique in nature and can stand alone.
The diagram shows the boundary between Order details application and the external applications, Customer and Supplier.

It also shows the boundary between the user and the Billing Application.
Requirements at a Detailed Level

To Identify what the user queries from the application

Business Logic

To Identify what the user inputs into the application

Business Logic

What Data is within the application Boundary

To Identify what the user receives from the application

Business Logic

Business Logic

What Data is external to the application Boundary
Using the Identification rules of Data Functions

- Data functions represent the functionality provided to meet internal and external data requirements.
- Data functions provide us the opportunity to identify the functionality which forms the core business data of the system.
- Once we identify this core business data then it’s much easier to identify the functionality which operates this data through user requests.
- Definition of ILF and EIF
  - Internal Logical File is user identifiable group of logically related data or control information maintained within the boundary of the application. The primary intent of ILF is to hold data maintained through one or more elementary processes of the application being counted.
  - External interface file (EIF) is a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application. The primary intent of an EIF is to hold data referenced through one or more elementary processes within the boundary of the application counted.
Using the Identification rules of Data Functions

- **User Identifiable**
  - As we have stated in the very beginning we are focusing on identifying functions which the user can recognize.
  - Isolate the user functions which are absolutely necessary for the sustenance and usability of the system.
  - The set of data can be both internal to the system or can be lying outside the system.

- **Logically Related**
  - Data that the user requires as part of the system should be logically related.
  - The focus on ensures that requirements captured are complete by themselves and can stand alone.
  - Focus on logically related data enables the requirement analyst to capture requirements which are absolutely necessary for the system and at the same time the data is user identifiable.
  - This is possible since we will not use the concepts of ILF or EIF in isolation.
Elementary Process

The data that is maintained in the ILF or referenced in EIF is through the elementary process. An elementary process is the smallest unit of activity that is meaningful to the user. The elementary process must be self contained and leave the business of the application being counted in a stable state.

Here if we take the luxury of replacing the word counted with analyzed. It makes absolute sense.

The requirements should always leave the system in a stable state. When we delete the employee information all his related information should also be deleted. But at the same time we should be able to identify that set of data that is required even after the employee has left the company.

The identification of elementary process forms the core of requirements elicitation.
Using the Identification rules of Data Functions

➢ Requirement analyst should have single minded focus on ensuring that all the transactions of the system under consideration meet the definition of an elementary process.

➢ This one concept of FP is extremely practical and can be used with any set of requirements.

➢ When you are able to identify the elementary processes which affect the system practically you have identified everything that is required in the system.

➢ Identifying the elementary process is at the heart of FP and it can also form the life line for requirement analysis if used with focus and consistency.
Using the Identification rules of Data Functions

- Requirement analyst should have single minded focus on ensuring that all the transactions of the system under consideration meet the definition of an elementary process.
- This one concept of FP is extremely practical and can be used with any set of requirements.
- When you are able to identify the elementary processes which affect the system practically you have identified everything that is required in the system.
- Identifying the elementary process is at the heart of FP and it can also form the life line for requirement analysis if used with focus and consistency.
- The concept of DET can used to capture requirements at the lowest granularity.
- As per the definition each DET should be recognizable by the user and the field should be unique for the entire system.
- This ensures that requirements unambiguous and unique
We have reached a stage where we should identify that fine thread that is running through all the stages of discussion till now.

We started with identifying functionality which is within the scope of the system. This scoping is enabled by drawing an application boundary.

The application boundary acts as a membrane through which data moves from outside the system and resides within the system and data that we need to refer to make this system complete and stable. This is the highest level of abstraction.

When we identify the data functions we are talking of next level abstraction. At this level the data is categorized along logical lines. Each logical unit is recognized both by the user and developer. These logical units are managed using elementary processes.

Elementary processes ensures that we capturing only essential data. This is the data that is recognized by the user.
An external input (EI) is an elementary process that processes data or control information that comes from outside the application boundary. The primary intent of an EI is to maintain one or more ILFs and/or to alter the behavior of the system.

Using this concept we can identify
- Transactions which bring in the data from outside the application boundary. This ensures that all the requirements that we have collected till now have proper input source. This input source is verifiable and user recognized
- The business rules of the collecting data
- All modification and deletion criteria for collected set of data
- Traceability can also be achieved since we can now know that no data function can exist without a transaction. Every data that is required by the user should be recorded through a transaction.
- Since EI’s are driven by the elementary process at the end of the transaction the system would be in a stable state.
Using the Identification rules of Transaction Functions

- Both EO and EQ present information to the user. These concepts enable us to bring clarity to requirements.
- We can know different business rules when we try to satisfy the required conditions of EO and EQ.
- Since each of them are executed by the elementary processes, once the processes are completed the system should be in a stable state.
- Through EO we can also know the derived set of requirements.
- These derived requirements are at fragmented level when we collect requirements initially but at the transaction functions level they become concrete because they form the business logic for EO.
- All the transaction functions involve an elementary process, at the end of the elementary process the system is stable.
- The stability is achieved because the requirements would be complete requiring now further amplification.
Interpreting the General System Characteristics

- Non functional requirements are applicable to the entire set of the requirements.
- They are applicable to the requirements at an aggregate level. These are requirements applicable to the product or service to be delivered.
- FP provides 14 different general system characteristics (GSC). GSC consists of functionality provided to the system.
- We can use these 14 GSC to capture requirements which normally don’t get accounted because of lack of awareness of analyst or the customer.
- FP provides us with a platform using which we can capture all the requirements under which system will have to operate.
Conclusion

- From the very beginning our focus has been to establish the fact that features of FP can be used to bring serenity and control to requirements gathering activity.
- The team can use the concepts of FP by converting into a checklist and ensuring that while they are looking for gaps in requirement the missing links can be established through this checklist.
- The logical flow of FP provides the requirement analyst with a sequence of activities which followed would ensure that the requirements he is collecting are complete, verifiable and less ambiguous.
- FP concepts can act as lighthouse for the development team. If they are not able to capture all the aspects of requirements, they at least know what is missing and how those missing requirements can influence the design and development of the application.

- We can say that” Requirements which are not good for counting is not good for developing”.

References

1. CPM 4.2.1 manual
2. Characteristics Of Good Requirement by Pradip Kar and Michelle Bailey
Function Points Analysis as a tool for Requirement Analysis (NGC-NGD)

Paper presented at second ISMA conference in Las Vegas

Prakash SK Kumar
QPRIME Team
Satyam Computer Services Limited www.satyam.com
Bangalore, India

Introduction
Function Point Analysis (FPA) requires that the practitioner’s use “user requirements” for sizing. The biggest problem today software community is facing is managing user requirements. Then why not use Function point principles as basis for capturing functional requirements. One line which sums up this paper is “What is Not Good for Counting (NGC) is Not Good for Developing (NGD)”. We here try to present based on our experience of working with more than 300 projects as estimation specialists, auditors and process consultants. The insights that we gained during these activities has made us realize that principles of Function Point Analysis would be an ideal tool to capture requirements in structured manner. This method not only reduces the time to capture requirements because you are capturing requirements in a logical format but this also reduces the time required for estimation. To sum it, there is no simple formula for writing excellent requirements. It is largely a matter of experience and learning from past requirements problems. Here we present our experienced based guiding principles which we believe would enable us to capture software requirements as we wanted for long long time.

- How we can leverage principles of FP (Function Point) to better our requirements.
- Using each rule of FP to bring clarity into requirements
- Capturing non functional requirements through FP
- Are we not talking the same language in FP and Requirements Analysis?
- FP as process improvement tool for Requirement Analysis Process

Characteristics of Requirements
Requirements are the basis for any software development or for that matter for any development activity. Unless the customers give their requirements even a jeweler
cannot properly prepare the desired ornaments which would leave the end user happy. When requirements are so important in such a superficial scenario, then to say capturing requirements is the most important and influential stage in software development is an understatement.

The word influential is used because requirements impact all the remaining phases of software development. It’s like diagnosis in medical field. If the diagnosis of the doctor is not correct then the treatment would be wrong. It can also be fatal. Similarly in software, getting the right requirements and delivering them within appropriate budget and schedule is absolutely important.

A requirement is defined as a property that must be exhibited in order to solve some real-world problem. Defining this some real world problem in an unambiguous way is our problem on hand. We propose to use Function point references which help us define the requirements in clear, unambiguous and consistent form facilitating software development.

Here in this section we will discuss the characteristics of good requirements.

Good requirements should have the following characteristics

1. **Necessary**
   The stated requirement is an essential capability, physical characteristic, or quality factor of the product or process. If it is removed or deleted, a deficiency will exist, which cannot be fulfilled by other capabilities of the product or process.

2. **Verifiable**
   The stated requirement is not vague or general but is quantified in a manner that can be verified by one of these 4 alternative methods: inspection, analysis, demonstration or test.

3. **Feasible**
   The stated requirement can be achieved by one or more developed system concepts at a definable cost. This implies that at least a high level conceptual design has been completed and cost tradeoff studies have been conducted. The system is broken down into smaller set of logical requirements and is analyzed to ensure they meet the higher level complex requirements.

4. **Unambiguous**
   Each requirement must have one and only one interpretation. Language used in the statement must not leave a doubt in the reader's mind as to the intended descriptive or numeric value.

5. **Complete**
   The stated requirement is complete and does not need further amplification. The stated requirement will provide sufficient capability. The state set of requirements leave the system is a stable state once the activity or transaction is achieved.
6. Consistent
The stated requirement does not contradict other requirements. It is not a duplicate of another requirement. The same term is used for the same item in all requirements.

7. Traceable
Requirements should be traceable to smallest level which makes sense to the user. The requirement should be traceable from this level to a level of higher abstraction.

8. Concise
The requirement statement includes only one requirement stating what must be done and only what must be done, stated simply and clearly. It is easy to read and understand.

9. Implementation free
The requirement states what is required, not how the requirement should be met. A requirement statement should not reflect a design or implementation nor should it describe an operation. However, the treatment of interface requirements is generally an exception.

10. Unique
The requirement can be identifying uniquely within the system and can be identified by a unique number or set of associated transactions and functions associated with it.

The requirements can be functional requirements or non functional requirements.

Analysis of Function Point Technique to capture good Requirements
We now look at different aspects of Function Point analysis which we can use to refine requirements to meet the above stated characteristics for good requirements. We will highlight characteristics of good requirements along the discussion by underling and italicizing them.

Extrapolating the Objectives of Function Points
We will start with considering the objectives of function point. The objectives of function point is to

- Measure functionality that the user requests and receives
- Measure software development and maintenance independently of technology used for implementation
The objective of measuring what the user requests and receives is so simple but very powerful. This ensures that we don’t deviate from our mission of developing a system that is relevant to the users with features that they desire. That is stated in short the developers are **concise and precise** for the job on hand.

In the characteristics of requirements we mentioned that the requirements should be free from implementation. In our experience we have seen that the developers sometimes reject the user requests stating the technology under consideration cannot support a particular functionality because it may require multiple platforms. Technology under consideration may do some aspects of functionality slower compared to the other or it may be less secure compared to other environments. So these various aspects of environment may influence capturing requirements. But according to function point analysis the functionality should be independent of technology considered. Function point analysis is technology averse. This concept of FP goes hand in hand with characteristics of Good requirements which expect that user needs be **implementation free**.

**Exploiting the User View**

The next concept that we are considering is of identifying the User view. A user view represents a formal description of the user’s business needs in the user’s language. Developers translate the user information into information technology language in order to provide a solution. A function point count is accomplished using the information in a language that is common to both user(s) and developers.¹

We understood the user view from FP perspective. The concept of user view ensures that the requirements are provided only by relevant stakeholders. This ensures that only the **necessary** requirements are captured. When we want to capture the requirements we want to make sure that we talk to people who can provide use with relevant information that would help us to develop the desired system. The user view concept of function point can be used to filter out those posing stakeholders who don’t have an interest in the development of the system under consideration. User view emphasis on capturing core business functions. The focus on business functions ensures that the core requirements of the system are captured upfront and can help us to prioritize the requirements. The user view emphasis that the requirements captured is approved by the user. This enables us to get the approval from the user before we start with next phases of development. The other aspect of user that user view can vary in physical form make sure that we capture requirements from different sources like user manuals, help documents, hand written reports or reports in other paper formats. This holistic approach towards capturing requirements ensures that we don’t miss any source through which we can capture user needs.

The concept that the language used by the user and developers should be
common can enhance the quality of the user needs captured. We will understand this with an example. The user of a banking application may want the system to search on last name or first name. But the developers may comeback saying this can lead to a search which gives infinite number of results which can be of no use. So the developers can further probe as why the user wants this search. Upon probing they get to know that users want to know the history of the customer when they approach them physically or over the phone or through any other media. Here the developer may suggest the users that they can ask the customer more questions like the type of account and branch to narrow the search. Such a search criterion enables the user to retrieve the data faster and to be more accurately. This aspect of user view ensures that the developers and end users talk to each other to capture a feasible and verifiable set of requirements. Since both the users and developers agree on captured set of requirements there is less chance of the requirements being vague or ambiguous.

Leveraging the Type of Count
The next concept of FP is about identifying the type of count. FP has three different types of count namely Development, Enhancement and Application Function Point Counts. This concept may not be of much relevance here since under any of these counts we need to capture requirements which have the characteristics stated in the earlier section.

Leveraging the Counting Scope and Application Boundary
By very title of this concept it should be clear that how helpful this can be when we want to capture user needs. FP requires that the counting scope be identified before we start counting. It expects the counter to draw a boundary around the application being counted. Some may feel that I am trying to misinterpret the concept of counting scope and application boundary. But I would suggest that these concepts can be used in a more proactive manner while we capture requirements.

The counting scope defines the functionality which will be included in a particular function point count. We use this to bring clarity into the end users minds as what can be the logical scope that can be developed. This step of scoping the requirements ensures that only the necessary requirements are captured and we are on target as per the user needs. This type of scoping also ensures that we are capturing requirements are complete by themselves and incomplete or unclear requirements are left for development at a later stage.

The concept of identifying the application is like icing on the cake for the requirement analyst.
The Function point requirement of drawing a logical boundary would enable us capture complete requirements for the desired business function. What we need is that requirements should be complete in itself and we should be able to tell what it lacks to make it complete. Function point’s principle of drawing an application boundary would precisely helps us to do that. Let us look at the definition of application boundary and how they can influence requirements analysis.

- Defines what is external to the application
  The concept of drawing an application boundary is so apt during capturing requirements. It helps the analyst to ensure that he captures what is required by the system. It also helps him to figure out what is not part of the system. This concept of drawing an application boundary can also be easily understood by the customer and he will cooperate in identifying the boundary because it provides him with lot of clarity regarding his requirements. Of course drawing the correct application boundary is very important because it would influence the requirements considered for development in turn for counting. So as a requirement analyst one should make sure correct application boundaries are drawn which may be quite unclear in the beginning but as the activity progresses the boundaries can be identified more accurately providing opportunity to capture requirements which are complete, concise and unambiguous.

- Is the conceptual interface between the ‘internal’ application and the ‘external’ user world
  The opportunity to identify what is internal to system and what is external to the system provides the requirement analyst with following guidelines to capture user needs
  - Identify the external applications which influence the system under consideration
  - Identify the external sources from which the requirements should be collected
  - Identify interface points within the application which interact with the external world
  - Helps determine the data formats for interaction with different external systems
  - Provide information to stakeholder of those external systems as to what they can expect from the system under consideration

The above mentioned guidelines are not complete set since the very discussion is out of consideration under the current article. It requires a completely separate paper where we can discuss this concept of FP and its influence on capturing requirements.
Coming to point as to how we can relate this to characteristics of Good requirements, it helps us to identify those necessary external systems that we should consider during development has having influence on the system under development.

- Acts as a ‘membrane’ through which data processed by transactions (EIs, EO$s, and EQ$s) pass into and out from the application  
  After identifying the application boundary the logical next step is to identify those transactions which cross the application boundary. Earlier we gave definition for FP which was “FP counts what user requests and receives”. This precisely helps to identify the transactions which would influence the system in the form of updating the system or altering the behavior of the system. Identifying these transactions helps us to capture requirements which are complete, consistent and verifiable because these transactions are expected to leave the system in stable state. Unless they are complete or consistent or verifiable they cannot leave the system in a stable state. When we ensure that each transaction is complete it leaves us with a set of requirements which are unambiguous and which are absolutely necessary for the system. These transactions also ensure that they are completely traceable from the perspective of what triggered these transactions and whether these transactions are unique in nature. The presence of the application boundary helps us to identify these outbound and inbound transactions.

- Encloses the logical data maintained by the application (ILFs)  
  With the help of the application boundary till now we identified what is external to the system, what are the transaction which influence the system stability and now the next logical step is to clearly mark the data we have to capture to satisfy user needs. At this stage of requirements we might not be complete clear as to what is expected by the user for satisfying the business need. But we should be definitely being able to pin point to those data that would influence the business and that data with no business significance.

- Assists in identifying the logical data referenced by but not maintained within this application (ELFs)  
  Requirement analyst can use this concept to identify the external source systems from which he needs to collect the requirements. When we talk of capturing requirements of those external interface requirements we might go astray by the mere complexity that we might encounter. But since we are using the concepts of FP, this becomes simple since we have identified those transactions and data which are required by the system. At this stage we just look for that data which influences these transactions or the data. This reduces the time required for capturing requirements and also helps requirement analyst to keep the user needs concise.
• Is dependent on the user’s external business view of the application. It is independent of technical and/or implementation considerations. Business users would not like to listen that they will not get a feature because it is not supported by the technology used for development. As we earlier discussed requirement should be implementation free. What it means is requirements should not be biased towards a technology or influenced by a technology. Similarly when we draw an application boundary we are trying to draw the application boundary from logical perspective rather than the technical perspective. What it means is that when we capture requirements we need to capture logically related requirements. This leaves the user needs in a stable and consistent state. The requirements would be independent of time, user, location and relationships. This enables us to capture requirements which are unique in nature and can stand alone.

Using the Identification rules of Data Functions

Data functions represent the functionality provided to meet internal and external data requirements. Transactional functions represent the functionality provided to the user for the processing of data by an application. Based on the discussion in the earlier paragraphs we can conclude that the requirement analyst by this time would have identified the user functionality which forms the core of the target system. Now comes the part of identifying the details required for the core system. This is where we can use the concepts of data functions and transaction functions.

Data functions provide us the opportunity to indentify the functionality which forms the core business data of the system. Once we indentify this core business data then it’s much easier to identify the functionality which operates this data through user requests. Now we will see how we can leverage the rules of data functions to bring clarity to our requirements analysis activity.

As we know data functions are of two types. They are Internal Logical File (ILF) and External Interface File (EIF). We will explore the core definitions of these two data functions.

Internal Logical File and External Interface File

Internal Logical File is user identifiable group of logically related data or control information maintained within the boundary of the application. The primary intent of ILF is to hold data maintained through one or more elementary processes of the application being counted. ¹

External interface file (EIF) is a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application. The primary intent of an EIF is to hold data referenced through one or more elementary processes within the boundary of the application counted. ¹
In both the above definitions read the application being counted as the application for which we are analyzing the user requirements?

As we have stated in the very beginning we are focusing on identifying functions which the user can recognize. Isolate the user functions which are absolutely necessary for the sustenance and usability of the system. As per FP user identifiable refers to defined requirements for processes and groups of data that are agreed upon, and understood by, both the users and developers. This set of data can be both internal to the system or can be lying outside the system. Let consider an example. The developers may see the need for unique id to easily identify a set of employee information. But the user cannot identify such data. The user can only identify with first, last name and date of joining of the employees. So developer can convince the user saying that he is using the id for easy of development and the user will not have to remember or deal with the id when he interacts with the system. This sort of trade off ensures that requirement is unambiguous.

Data that the user requires as part of the system should be logically related. The requirement analyst should remember that he captures all related information with a set of user needs. As part of banking system the user needs functionality of opening a bank account. When opening a bank account the requirement analyst should ensure that he not only captures the primary user information but also the nominee information. It is a set of logically related data which the user may not understand. But since the requirement analyst is focused on capturing logically related data. This would make the requirements complete by themselves and which can stand alone. Focus on logically related data enables the requirement analyst to capture requirements which are absolutely necessary for the system and at the same time the data is user identifiable.

As part of the definitions we have used a term called elementary process. The data that is maintained in the ILF or referenced in EIF is through the elementary process. An elementary process is the smallest unit of activity that is meaningful to the user. The elementary process must be self contained and leave the business of the application being counted in a stable state. Here if we take the luxury of replacing the word counted with analyzed. It makes absolute sense. The requirements that we capture should be at a level which is traceable to smallest granularity possible. Even at that level the requirements should be meaningful to the user. Further the requirements should always leave the system in a stable state. When we delete the employee information all his related information should also be deleted. But at the same time we should be able to identify that set of data that is required even after the employee has left the company. This is where we can use the concept of elementary process. The concept of elementary process directly tackles the characteristics of requirements like completeness, stability, verifiability and uniqueness. The identification of elementary
process forms the core of requirements elicitation. Requirement analyst should have single minded focus on ensuring that all the transactions of the system under consideration meet the definition of an elementary process. This one concept of FP is extremely practical and can be used with any set of requirements. When you are able to identify the elementary processes which affect the system practically you have identified everything that is required in the system. **Elementary processes forms the core of any system under the sun. Identifying the elementary process is at the heart of FP and it can also form the life line for requirement analysis if used with focus and consistency.**

Data functions have two categories. ILF helps to identify the functionality which forms the system whereas EIF enables the analyst to identify the functionality that would be external to the system. Using EIF we can recognize the data that we need to refer to make the system complete. **Recognizing the functionality that application needs to refer can always be tricky since the possibilities are unlimited but indentifying external data in collaboration with elementary processes and application boundary can reduce the risk of going astray.**

The complexity of Data functions is determined by the number of Data element types (DET) and Record element types. A DET is a unique user recognizable, non-repeated field. The concept of DET can used to capture requirements at the lowest granularity. As per the definition each DET should be recognizable by the user and the field should be unique for the entire system. This ensures that there are no duplicate requirements that we are collecting. Further this reduces the ambiguity in the requirements collected since even at the lowest level we seek that the user should be able to identify him with the collected requirements. Once we start collecting requirements at this level we are sure that we are collecting requirements which are **unique.** By the discussion till now we can understand that we are drilling down from the highest of abstraction to lowest level of details. This helps us to keep the requirements **concise and traceable.**

Once we start collecting requirements under these concepts we can ensure that we are in that “zone” where very little can go wrong.

**Discussion**

We have reached a stage where we should identify that fine thread that is running through all the stages of discussion till now. We started with identifying functionality which is within the scope of the system. This scoping is enabled by drawing an application boundary. The application boundary acts as a membrane through which data moves from outside the system and resides within the system and data that we need to refer to make this system complete and stable. This is the highest level of abstraction. When we identify the data functions we are talking of next level abstraction. At this level the data is categorized along logical lines. Each logical unit is recognized both by the user and developer. These logical units are managed using
elementary processes. Elementary processes ensures that we capturing only essential data. This data is recognized by the user.

In the following paragraphs we will leverage the transaction function and general system characteristics of Function point analysis.

**Using the Identification rules of Transaction Functions**

Transactional functions represent the functionality provided to the user for the processing of data by an application. Transactional functions are defined as external inputs (EIs), external outputs (EOs), and external inquiries (EQs).

Till now we discussed business data which constitutes the system. Now we will look at how we can identify the business rules which influences that the business data. The transaction functions act upon the data that has been collected based on data functions concepts.

Transaction functions are of three types.

**External Inputs (EI):**

An external input (EI) is an elementary process that processes data or control information that comes from outside the application boundary. The primary intent of an EI is to maintain one or more ILFs and/or to alter the behavior of the system. Using this concept we can identify

- Transactions which bring in the data from outside the application boundary. This ensures that all the requirements that we have collected till now have proper input source. This input source is verifiable and user recognized.
- The business rules of the collecting data.
- All modification and deletion criteria for collected set of data.
- Since complexity of EI’s is indentified by DET’s and FTR, we can know which data is being updated through which business process under which data functions. This ensures **verifiability** of each business transaction.
- **Traceability** can also be achieved since we can now know that no data function can exist without a transaction. Every data that is required by the user should be recorded through a transaction.
- Since EI’s are driven by the elementary process at the end of the transaction the system would be in a stable state. This means that the transaction or the requirement is **complete** by itself.

**External Outputs (EO) and External Inquiries (EQ):**

An external output (EO) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external output is to present information to a user through processing logic other than, or in addition to, the retrieval
of data or control information. The processing logic must contain at least one mathematical formula or calculation, create derived data, maintain one or more ILFs or alter the behavior of the system.\textsuperscript{1}

An external inquiry (EQ) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external inquiry is to present information to a user through the retrieval of data or control information from an ILF of EIF. The processing logic contains no mathematical formulas or calculations, and creates no derived data. No ILF is maintained during the processing, nor is the behavior of the system altered.\textsuperscript{1}

Both EO and EQ present information to the user. We can know different business rules when we try to satisfy the required conditions of EO and EQ. Since each of them are executed by the elementary processes, once the processes are completed the system should be in a stable state. All the reports required by the user can be known. Through EO we can also know the derived set of requirements. These derived requirements are at fragmented level when we collect requirements initially but at the transaction functions level they become concrete because they form the business logic for EO.

Each of the above transaction functions improve the \textit{verifiability} of the requirements collected because we try to make sure all the requirements are associated with at least one of three transactions. Since all the transaction functions involve an elementary process, at the end of the elementary process the system is stable. The stability is achieved because the requirements would be \textit{complete} requiring no further amplification. These transactions functions enable the requirement analyst to capture requirements as the functional requirements. Functional requirements are qualitative requirements describing what the system needs to do without describing them in quantitative terms. These requirements are usually descriptive and are verified by the summation of the associated requirements.

\textbf{Interpreting the General System Characteristics}

Non functional requirements are applicable to the entire set of the requirements. They are applicable to the requirements at an aggregate level. These are requirements applicable to the product or service to be delivered. FP provides 14 different general system characteristics (GSC). GSC consists of functionality provided to the system. We now discuss how we can use these 14 GSC to capture requirements which normally don’t get accounted because of lack of awareness of analyst or the customer. Now FP provides us with a platform using which we can capture all the requirements under which system will have to operate. This would make the requirements \textit{complete}.

\textbf{Data Communication}

Data Communication describes the degree to which the application communicates directly with the processor\textsuperscript{1}. Using this we can gather requirements regarding
- Whether the system is stand-alone or does batch processing
- What is the distance the data will have to travel?
- What are physical communication channels available for data transmission?
- Is the system involved in remote computing?
- Whether the system requires different protocols to operate
- Whether the data enters the system through another file or system or it involves user entry

**Distributed Data Processing**
Distributed Data Processing describes the degree to which the application transfers data among physical components of the application. Probing should provide answers for the following questions

- Does the system need Centralized computers, processing, data, control, support
- How about if Computers are dispersed throughout organization
- If desktop is provided with more processing power
- Whether Data is sent between client and server in one direction only
- Does the application is client server or web enabled architecture
- Does the application the ability to share data across multiple servers

Answers for these questions would provide the developers with lot more information to design the application.

**Performance**
Performance describes the degree to which response time and throughput performance considerations influenced the application development. Application performance objectives, stated or approved (or implied) by the user, in either response or throughput, influence (or will influence) the design, development, installation, and support of the application. Performance of the application should be collected for different scenario of execution.

- Performance during peak operating time
- Performance required for critical modules of the application
- Capture quantitative details for desired performance parameters. These factors should be explored for feasibility
- Requirement analyst should be able to determine what is with the control of the application and what is beyond the application scope. Because sometimes network performance might be poor and any amount of application tuning would not improve the performance
- Requirement analyst should focus on capturing as-is performance of the application or deployment environment so that he can compare it to-be one once it is completed
• Response time and throughput required for various transactions should be documented

**Heavily Used Configuration**
Heavily Used Configuration describes the degree to which computer resource restrictions influenced the development of the application.¹

A heavily used operational configuration may require special considerations when designing the application. For example, the user wants to run the application on existing or committed equipment that will be heavily used. The requirement analyst should capture details of the configuration required to set up the application. Whether the application is deployed on servers where there are multiple other applications or is being deployed on a standalone basis has an impact on the design and development of the application. Further, what is data growth in the deployed environment can also be captured.

**Transaction Rate**
Transaction Rate describes the degree to which the rate of business transactions influenced the development of the application.¹

The transaction rate is high, and it influences the design, development, installation, and support of the application. Users may require what they regard as normal response time even during times of peak volume. Transaction rate can vary based on the time interval the application is being used. The information regarding when the users want maximum transaction should be captured and agreed upon. Knowing the transaction rate also helps to decide what type of tools should be used for the development of the application.

**Online Data Entry**
Online Data Entry describes the degree to which data is entered or retrieved through interactive transactions.¹ On-line User Interface for data entry, control functions, reports, and queries are provided in the application. Application involving more online data entry should have minimum number clicks required to complete an operation. Development team should work towards reducing the amount data entry required to complete a meaningful transaction. FP makes sure that requirement analyst is thinking on these lines.

**End-User Efficiency**
End-User Efficiency describes the degree of consideration for human factors and ease of use for the user of the application measured.¹ The on-line functions provided emphasize a design for user efficiency (human factor/user friendliness). The requirements can be captured under some of these factors which influence end user efficiency. Such a list provides both the customer and Requirement analyst with an
opportunity to create transparency upfront. Based on these factors the development
team can create prototypes to reduce ambiguity in the requirements. We have here
provided a few points to be considered during design application for end user
efficiency. For a more detailed study we would ask the reader to refer the CPM manual.

- Navigational aids (e.g., function keys, jumps, dynamically generated menus, hyper-links)
- Menus
- Scrolling
- Pre-assigned function keys (e.g., clear screen, request help, clone screen)
- Batch jobs submitted from on-line transactions
- Heavy use of reverse video, highlighting, colors, underlining, and other indicators
- Mouse interface
- Pop-up windows
- Bilingual support or Multi-lingual support

Online Update
On-line Update describes the degree to which internal logical files are updated on-
line. If there is lot of online update the requirement analyst should look at breaking
down the functionality in smaller meaningful transactions to ensure that the transaction
rate, performance and online data entry are minimal for faster and accurate online
update. if the processing is too complex then online updated could error prone and
also difficult for development. Breaking the complex transactions in more manageable
chunks can be achieved through intensive discussion the customer.

Complex processing
Complex processing describes the degree to which processing logic influenced the
development of the application. Identifying these critical complex processing enables
the development team to allocate suitable resources for development of such
features. Further the requirement analyst can devote more time for such complex
processes.

Reusability
Reusability describes the degree to which the application and the code in the
application have been specifically designed, developed, and supported to be usable
in other applications. This provides the development team an opportunity to identify
reusable components in the development scenario. They can also look for reusable
components in the customer environment.

Installation Ease
Installation Ease describes the degree to which conversion from previous environments
influenced the development of the application. Conversion and installation ease are
characteristics of the application. Focusing on such requirements is quite uncommon
during most of the development activities. But this can to great extent influence the dosing of the application. If the users are quite educated the installation may require complete automation whereas educated users may not need too much automation. Also the options provided during the installation can also clarified during requirements capturing activity. Further during installation there might one time activity like data conversion which if know upfront would reduce ambiguity in development activities.

**Operational Ease**
Operational Ease describes the degree to which the application attends to operational aspects, such as start-up, back-up, and recovery processes. Operational ease is a characteristic of the application. The application minimizes the need for manual activities, such as tape mounts, paper handling, and direct on-location manual intervention. Capturing requirements on these lines is tantamount for the success of the system under consideration. These have direct impact of the cost of the hardware the organization should maintain. Further the ease with which these can be done ensures that these critical activities are regularly done. If an operation fails what exactly should be done should be captured right during requirements gathering.

**Multiple Sites**
Multiple Sites describes the degree to which the application has been developed for different hardware and software environments. We have come across instances where applications are not compatible with different versions of certain software forget different platforms. Availability of such information right at beginning of project provides the development team with an opportunity to avoid those features of the operating system or supporting software. This can reduce drastically the cost of the overall maintenance of the application during its existence.

**Facilitate Change**
Facilitate Change describes the degree to which the application has been developed for easy modification of processing logic or data structure. This factor influences the design and development effort to a very great extent. Applications which need more flexibility are more complex to develop. Further capturing requirements for such an application is tough. The requirement analyst should be capable of thinking through various permutation and combination that would be possible under different scenarios. But with FP he is at least aware that he needs to capture such user needs.

**Conclusion**
From the very beginning our focus has been to establish the fact that features of FP can be used to bring serenity and control to requirements gathering activity. It keeps the focus of the requirement gathering team because they are aware as what they need to collect during this of software development. The team can use the concepts of FP by
converting into a checklist and ensuring that while they are looking for gaps in requirement the missing links can be established through this checklist.

FP can also act as set of best practices during the requirement analysis activity. The logical flow of FP provides the requirement analyst with a sequence of activities which followed would ensure that the requirements he is collecting are complete, verifiable and less ambiguous.

There are various other factors which influence requirement gathering activity like available time, available resource or stake holder involvement. But under these circumstances FP concepts can act as lighthouse for the development team. If they are not able to capture all the aspects of requirements, they at least know what is missing and how those missing requirements can influence the design and development of the application.

At last I think we can say that “Requirements which are not good for counting is not good for developing”.

References

1. CPM 4.2.1 manual
2. Characteristics Of Good Requirement by Pradip Kar and Michelle Bailey