Brief Introduction to Function Modelling

Graham McLeod
inspired.org
October 2014

Definition and Purpose

It is often difficult to understand a whole organisation or the intent of a project. There may be hundreds of things that get done, processes followed, services performed and automated components to support them. One venerable technique that can be traced back to Structured Analysis in the 1970s is to use a Function Model.

This is essentially a hierarchical decomposition from the top level mission or goal, listing all the functions that must be accomplished to achieve it. The full model is a large tree. Top and middle levels typically represent logical things that must be done. The lowest level, comprising the "leaf nodes" may represent physical actions or automated actions. We normally keep the logical functions independent of *how* the activity is performed.

Function Models are very useful for a number of purposes, including:

- Understanding a business, department, project or other functional area
- Getting an appreciation of scope
- Building shared understanding in a team and between business and technical staff
- To determine which aspects should be automated
- To allocate responsibility to different teams or roles
- As a basis to use for estimating
- As a base to use for tracking progress on analysis, design, development and testing
- As a foundation for process modelling, service modelling and capability modelling
- As an input to systems design
- As a means to tie together different dimensions of the problem - e.g. Function, data, user interface, technology
- As a basis to define metrics and assign benchmarks and targets for performance

Notation

The model is best represented by a graphical tree structure comprising text in boxes and branching lines showing the structure.

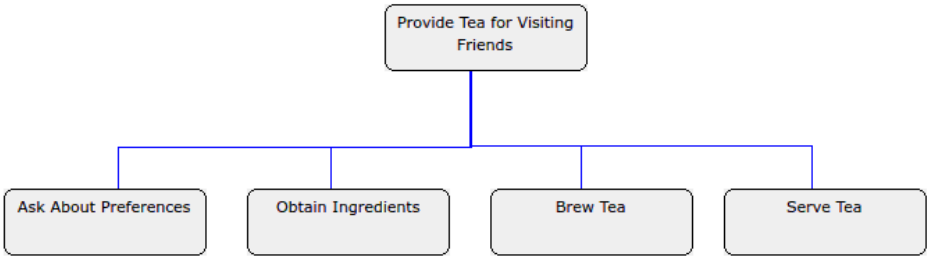

Figure 1 - Function Model Vertical Notation

An alternate which is logically equivalent is to have a tree which is oriented horizontally and expands to the right.

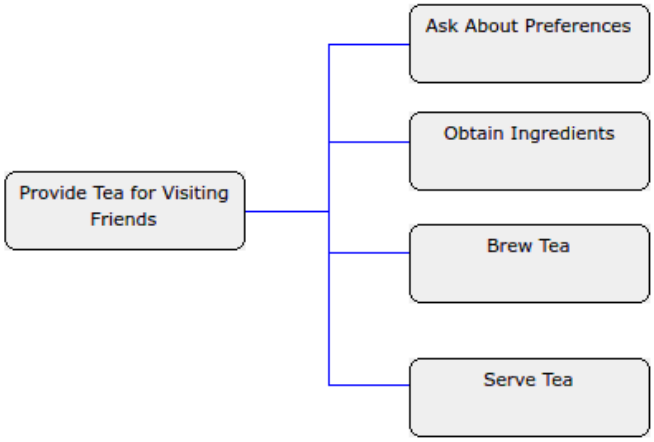

Figure 2 - Function Model Horizontal Notation

Another way is to use nested boxes, although this can become tricky if we decompose to too many levels. It is a compact and intuitive way of showing say three to four levels of decomposition.

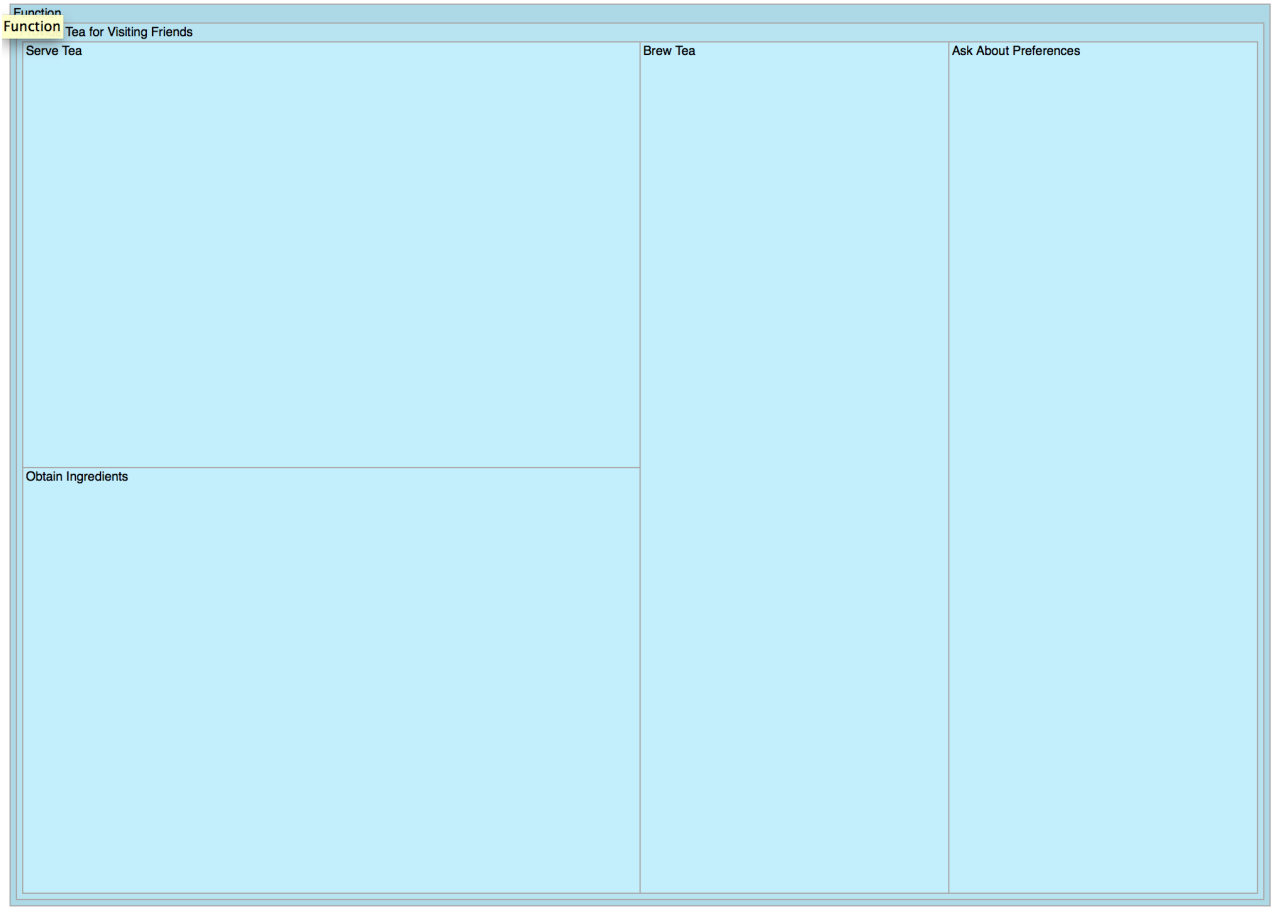

Figure 3 - Function Model, Nested Box Notation

A very simple form would be to have a list with indentation indicating the hierarchy.

Provide Tea for Visiting Friends

- Ask About Preferences
- Obtain Ingredients
- Brew Tea
- Serve Tea

The wording of each function has the following form:

[verb][object] [optional qualifying clause]…

e.g. Move Funds to Trust Account

Capture Order on Mobile Device

Hire Staff who are Qualified and Motivated

Approve Financial Results at end of Year

Using this form ensures that we are defining a *function*, i.e. something that gets done. This is why we need the verb first. Second, the object specifies *to what* the action applies. Finally, the qualifying clause can be used to bring in other considerations, such as cost, quality, timing etc.

Decomposition

At the top of the model, we are interested in the *overall intent* - i.e. what must the net end result be if all of the functions are performed successfully? You can think of this as a top level goal or a mission statement, if you like. Examples could include:

> Publish a Victorian Novel for Profit
>
> Optimise Care to Clinic Patients in Langa area within Constraints of Provided Budget
>
> Operate a Profitable and Growing Online Retailer offering Ethnic Clothing

To move to the next level, we ask the question:

> "What must I do in order to (achieve the box above..)

E.g. for the second example above, the questions and answers might be:

"What must I do in order to Optimise Care to Clinic Patients in Langa area within Constraints of Provided Budget?"

a1 Obtain Adequate Budget

a2 Determine Patient Demand Accurately

a3 Plan Services

a4 Deliver Quality Services

a5 Monitor Service Effectiveness

Lets do another level:

"What must I do in order to Deliver Quality Services"

a1 Ensure Adequacy of Physical Facilities

a2 Ensure Adequate Skilled and Motivated Staff are Present

a3 Assess Patients

a4 Treat Patients

a5 Educate Patients

a6 Discharge Patients

So, our model would then look as follows:

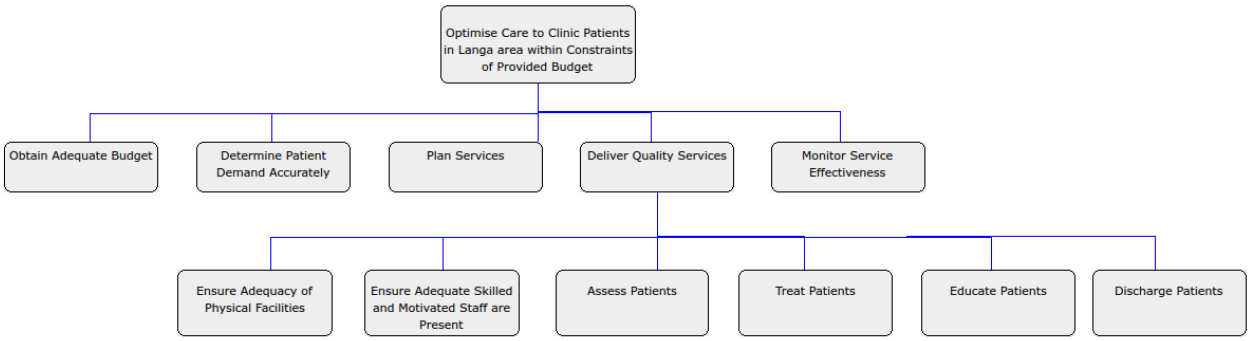

Figure 4 - Sample Function Decomposition

We can check that a function is in the right place, or discover its parent by asking the question: "Why do I do ….[this function]?" The answer should be the parent function. This can be used to work "bottom up" towards higher level functions from something that you know is done. e.g. I know we post out a statement to clients. What is its purpose? Following the logic above, we may see that it is "Advise Customers of Indebtedness". Having the logical function allows us to consider other ways of achieving the objective, e.g. "e-mail Customer Statement".

At the lowest level of the model, we will encounter physical functions that can be performed in various ways. An example of achieving a logical function in several ways is shown below:

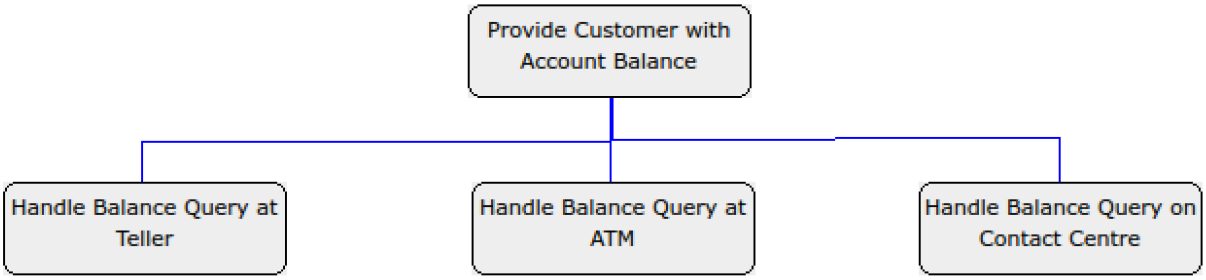

Figure 5 - Different Ways of Achieving the Same Logical Function

Note that each way is different, but they achieve the same end result. Of course, they may differ with respect to time taken, efficiency, cost etc. and so one may be preferred over the others.

We can also exploit something similar to map the leaf nodes as the current way something is done contrasted with the proposed future way of doing it, as below:

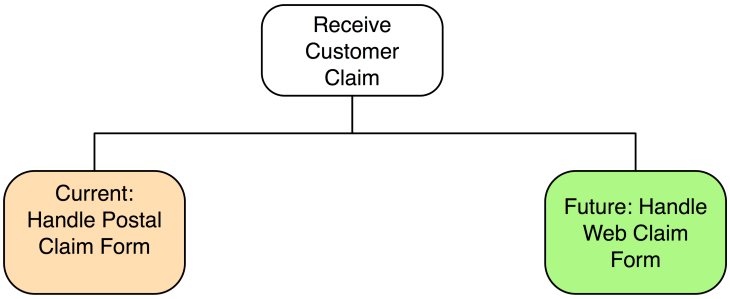

Figure 6 - Showing Current and Future Options to Achieve the Same End

Limiting Complexity

Following research into how the human brain works with short term memory, we know that most people can deal with 5-7 concepts at a time. More than that and older ones get "bumped out" of the short term storage, which is where they need to be for logical operations. Think of it like limited desk space where you can only open so many folders before something falls off the table.

To work within this limitation, we use a rule that says each parent box should be decomposed to no fewer than 2 and no more than 6 (preferably five) child boxes. There is no point decomposing to a single box, since that would, by definition, be equal to the parent box.

Validation

One important thing that function models allow us to do that is hard when doing context or process models, is to check completeness. To to this, we can think of a parent box and its children as a group, then apply some simple checks. Since the children are a decomposition of the parent, they should (together) exactly equal the parent in terms of what they achieve. If you want to think of it mathematically it is effectively Parent = Sum Of (Child1…Childn). Or graphically..

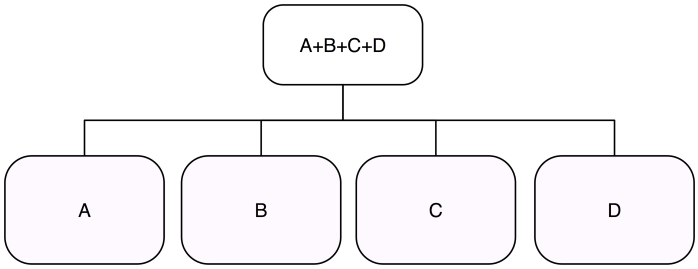

Figure 7 - Validating Completeness

If we check that the children sum to the parent, and find that there is something missing, then we should look for more child functions. If the sum of the children is greater than the parent, then either (a) there are too many children, or one or more child functions is in the wrong place or (b) the scope of the parent should be increased.

This important completeness check capability can help us to ensure that the coverage of our model is comprehensive and nothing important is excluded in our analysis or resulting system.

Patterns

Sometimes it is useful to use some “prior wisdom” in decomposing our models. For example, there is a common pattern that works well at the second layer of an enterprise model that includes the functions:

Plan

Create Infrastructure

Perform Operations

Enhance Performance

At a more detailed level of decomposition this may eliminate the infrastructure part, e.g.

Plan

Do

Check

These sorts of patterns can be adapted for most industries / enterprises and help to ensure we think about planning, creating necessary conditions for operation, normal operations, quality checking and ensuring goals are met and taking corrective or improvement actions.

Additional Information

Behind the scenes we can enhance the function model with all sorts of useful information, including (for e.g.)

- Status of the function (e.g. Is it Performed Now / Required in Future?)
- Status of our activity with respect to the function (e.g. Analysis Done, Design Done, Build Done, Testing Complete, User Accepted, Deployed into Production)
- Responsibility for the function (mapped to organisation unit or role)
- Volumes per unit time - this can be a good guide in whether it is worth automating the function or not. Complex things that occur infrequently are not good candidates for automation. Things that are simple and happen at volume and frequently are good candidates.
- Cross references to
  - where the function appears in processes
  - data required and how it is used (e.g. created, updated, deleted, archived)
  - which functions are exposed as services, who provides them, who consumes them, how they are invoked and what they return
  - where functions are required geographically
  - which functions are to be supported in which phase of a project / release of a product or system
  - who has knowledge and expertise about the function within the organisation (or externally)

By tracking status of functions and our activity with respect to the full model, we can get a much better appreciation for the progress of our project.

Example

Note that a full function model for a whole organisation can be large (typically 250-300 boxes). We are showing a fragment of a model below for space reasons, but it should give you a good idea.

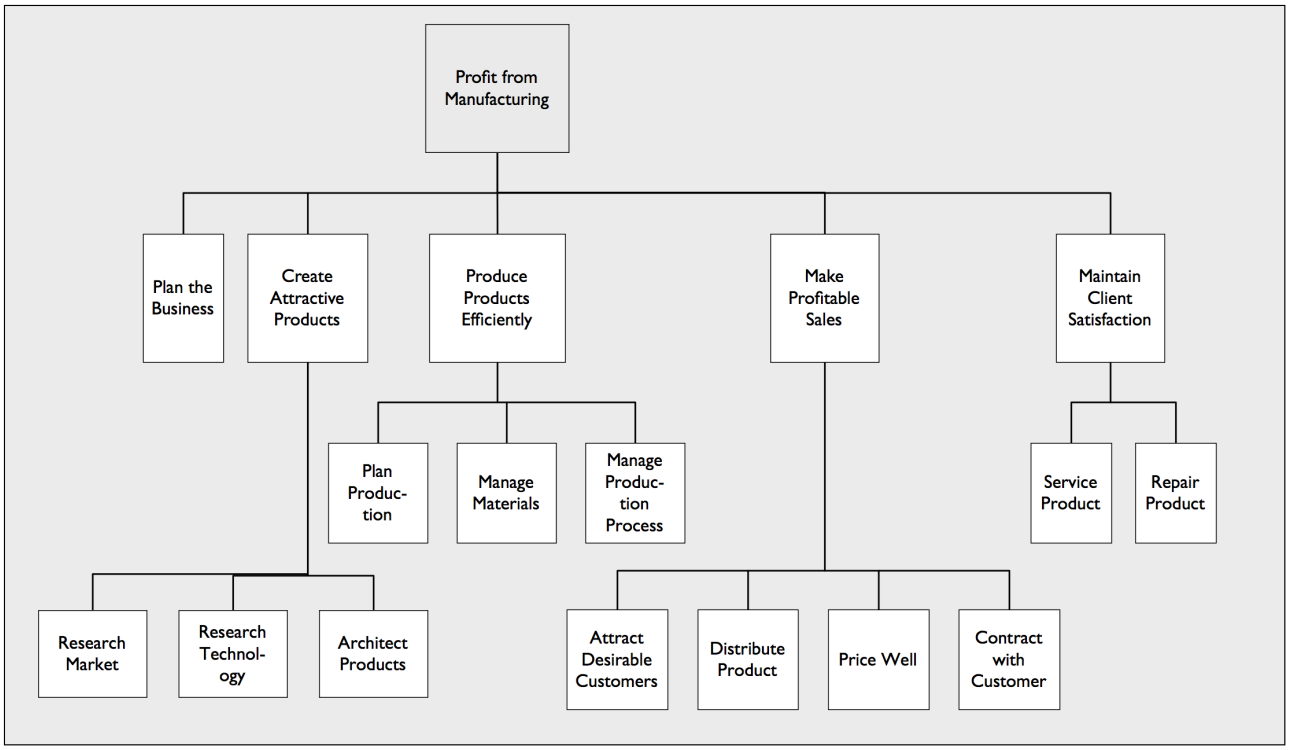

Figure 8 - Representative Function Model Sample

We could carry a lot more information on the model by, for example, colour coding the boxes to indicate status..

Reconciling Function Modelling with Other Perspectives

There are a variety of modelling paradigms that are often seen as competing. In this section we will show how they are related, can be reconciled, and even leveraged to good effect.

Motivation Dimension

This addresses the strategic intent of the organisation - i.e. “What do we want to achieve”. It can be expressed as a decomposition of the Mission to Goals and subsequently Objectives. The Mission should be expressed in a succinct and unambiguous way and encapsulate the overall goal of the enterprise.

Goals are broad statements of direction. E.g. “Reduce Costs”, “Increase Market Penetration in Asia” etc.

Objectives are goals which are made Specific, Measurable, Achievable, Realistic and Time Bounded (SMART). E.g. “Reduce Costs by 5% in real terms over the next 24 months”, “Increase Market Penetration in Asia to 5% of the Market by 2017Q4”.

When work to achieve the mission has been defined (see next sections on Function and Services), then Objectives can be usefully employed to drive the definition of KPI’s for organisation units and roles.

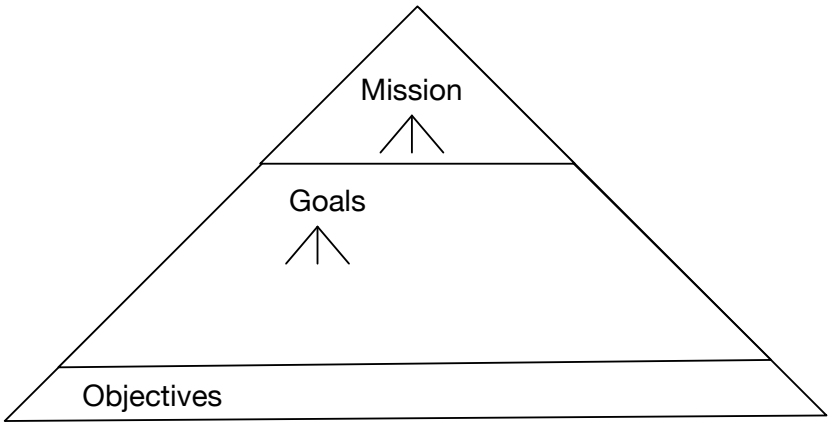

Figure 9 - Motivation / Goal Model

Functional Paradigm

One traditional approach derived from structured design is functional modelling. It has the advantage that a single model can be built for a whole enterprise and that, properly done, it is also relatively easy to ensure a fair degree of rigour and completeness. The basic idea is to begin with a Mission Statement that encapsulates the overall goal of the enterprise in an unambiguous and succinct way. This is then decomposed into things that have to be done (functions) at successive levels of detail, until we reach a fairly detailed level of activities. It can be illustrated as follows:

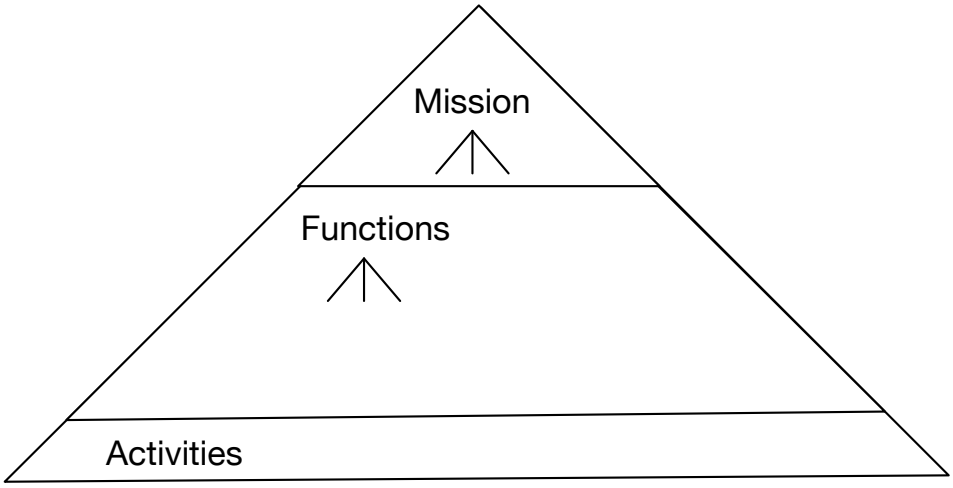

Figure 10 - Functional Decomposition

A mission statement might read as follows:

> “Become a top three telecommunications service provider within Africa by offering excellent value and experience to customers and other stakeholders”

This could be decomposed at the next level as follows:

Plan Enterprise Development

Run Effective Change Projects

- Develop Profitable Products and Services that Delight Customers
- Create Industry Leading Infrastructure to Deliver Cost Effective Service
- Recruit High Quality Personnel Appropriate to our Culture and Industry

Operate the Business Profitably whilst Delighting Customers and other Stakeholders

- Increase Sales by Maintaining Excellent Customer Relations and Experience
- Manage Cash Flow to Ensure Effective Investment and Returns
- Maintain Network to Offer High Quality Service
- Ensure Staff are Skilled and Motivated

Monitor Performance of the Business

At the lowest level, functions will be decomposed into activities - detailed actions performed by a person or automated device or software, e.g. “Print Project Progress Report” OR “Complete Appointment Form for New Hire”.

Note that functions are typically expressed at a logical level, whereas activities are at a physical level. The same function could thus be achieved in several ways. E.g. The function “Transmit Order to Warehouse” could decompose to “Fax Order Form to Warehouse” or “Send EDI Order Message to Warehouse”. This allows for alternates in how we implement lower level functions. It can also reflect a progression in methods over time. e.g. the Fax option could be the current way, while the EDI option could be the future plan.

We can use a variety of patterns to make the function definition more rigorous and easier to perform as well as to structure the overall hierarchy. In particular, the wording of a Function should include the following pattern: [Verb] [Object] [Qualifying Clause]… which translates to “do something” to “something” “in a particular way or achieving a particular goal”. For example: “Hire Staff who are Knowledgeable in our Industry and Highly Motivated”.

Process Paradigm

The process paradigm seeks to ensure that all activities necessary to produce a required end result are performed successfully in the necessary sequence (respecting dependencies). In addition, it underpins the automation of processes to achieve high levels of quality, consistency and efficiency in operation.

The highest level of process thinking is typically the Value Chain. This can be organised around product, customer or other concepts. It relates the highest level activities that are performed to achieve the strategic mission. Typically the value chain will have less than ten elements. For example, the classic Porter value chain has just five core value chain steps, supported by a number of support activities which span the core activities. Other variants (e.g. Kotler, or the Customer Value Chain from Business Engineering) may have more steps.

Below the value chain, core processes should provide a value to an end stakeholder. The latter can include customers, channel partners, suppliers, regulators, employees and others. Process analysis starts at a high level of abstraction and proceeds through several layers until it is at a similar level of detail to the Activities described in the Function Paradigm in the previous section.

When a process achieves this level of detail, we can describe it as a Procedure, since it will be describing in detail what is (or what should be) done by the participants in a process to do it in practice.

If we have previously done functional modelling, then we can see processes as containers which hold related functions (with dependencies) at a logical process level, or activities at a procedure level. This is illustrated below.

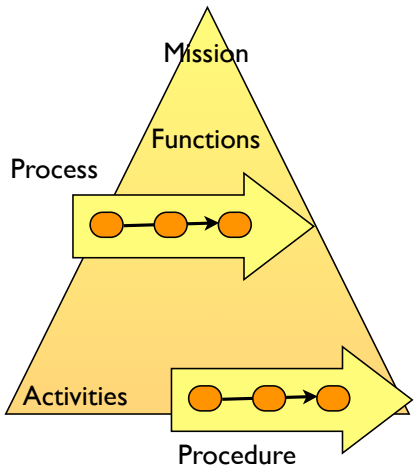

Figure 11 - How Process groups Function

Service Paradigm

Services have more recently come to the fore as a powerful concept for achieving modularity, allocating responsibility and supporting agility in organisations and the application and technology support for effective and efficient operations.

A simple definition of a Service is “A function that we perform for another party”. This implies that the service has the following characteristics:

- It has a unique identity
- It has a defined way to invoke it (These two points require that the service is *published*)
- We know what information (or other inputs) must be provided for the service to function correctly
- We know what information (or other outputs) the service will produce as a result
- We know who will provide the service
- We know who the target consumers of the service are
- We should have specified service levels that the consumers can expect
- It is kept stable by the provider, unless a new agreement is reached with the consumers

Thought of in this way, we can derive services from functions and apportion responsibility to units within the organisation.

Capabilities Paradigm

Capabilities are much discussed in modelling and planning today, but are seldom well defined. We like to take quite a hard line on this and consider a capability as “The ability to do something for someone at a specific place at a certain service level including the aspects of function, information, resource and infrastructure necessary.”

So, it is effectively a service plus:

- Location
- Volume / Capacity
- Resources (including people, skills, supplies, equipment..)
- Information
- Technology / Infrastructure
- Language
- etc.

Putting a summary of the foregoing discussion together in one picture, we can see the relationships between the paradigms as follows:

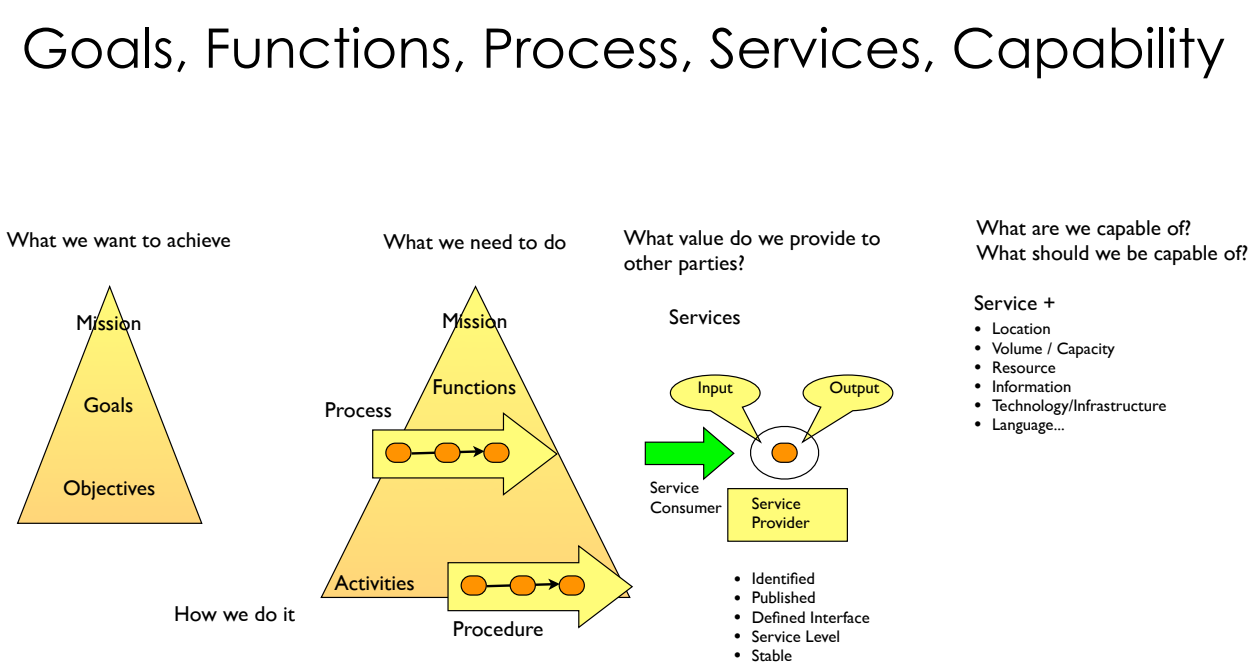

Figure 12 - Summary of How Paradigms Relate

Thus we can usefully decompose the motivation aspect as a hierarchical model and associate metrics with the objectives.

We decompose the mission to functions and activities. Functions can be used to compose processes. Activities to compose procedures.

By adding to functions the aspects of provider, consumer, interface and results, as well as the agreed service level and stability, we define services.

Finally, services can be enhanced to capabilities by associating the relevant locations, resources, information, technology and language of delivery.

Conclusion

We hope that this paper proves useful to you in your thinking around function and other modelling. Please do not hesitate to contact the writer at: graham@inspired.org

Note too that all models mentioned can be managed in our flexible SaaS enterprise modelling and knowledge management toolset and repository, EVA Netmodeler. The tool also allows cross

referencing all aspects and producing them as matrices or reports. An additional “Hierarchy Plus” report style allows showing the hierarchy with an associated type of choice (graphically) or with multiple related types (report). So, for example, we can easily see the function hierarchy and associated automation components, or responsibility, or status etc.

