Where do Requirements come from, Mommy?

By John Prouty, Exec. Business Analyst
MetaPower Inc.

Table of Contents

Tables and Figures

Introduction

MetaPower has been working with business processes since the late 1970's. Along the way we have learned a lot about these processes and how to effectively design, construct and deploy computer applications that automate them. This paper is about one of those lessons learned.

Joel F. Wade wrote in his Nov 2, 2011 editorial column for The Daily Bell (Wade, 2011):

> We all know where babies come from: the stork, right? Ha ha. That, of course, is a tale told to young children by uncomfortable parents. How ridiculous would it be if millions of people in America actually thought that this were so?
>
> But the reality is that with regard to other facts of life, we are not very far from this; we have become a society where too many people don't know where things come from.

I believe that Business Requirements fall into this category. As Business Analysis professionals, we have all come to understand that to be successful, we need to capture the business requirements so that the processes we develop properly support the business. But where do these requirements come from?

Before we ask where requirements come from, we should first define what they are. Wikipedia gives this definition (Wikipedia, Business Requirements, 2014):

> **Business requirements are <u>what</u> must be delivered to provide value.** Products, systems, software, and processes are the ways how to deliver, satisfy, or meet the business requirements whats. Consequently, the topic of business requirements often arises in the context of developing or procuring software or other system; but business requirements exist much more broadly. That is, 'business' can be at work or personal, for profit or non-profit.
>
> Confusion [between Business Requirements and Product Requirements] arises for three main reasons.
>
> (1) A common practice is to refer to objectives, or expected benefits, as 'business requirements.'
>
> (2) People commonly use the term 'requirements' to pertain to the features of the product, system, software expected to be created.
>
> (3) A widely held model says these two types of requirements differ only in level of detail or abstraction—wherein 'business requirements' are high-level and vague and decompose into product, system, or software requirements that are detailed.
>
> Such confusion can be avoided by recognizing that business requirements are not objectives but rather meet objectives (i.e., provide value) when satisfied. Business requirements whats do not decompose into product/system/software requirements hows. Rather, products and their requirements represent a response to business requirements—a way how presumably to satisfy the whats.

So the question remains "Where do these Business Requirements come from?" If you Google "Gathering Business Requirements", you will get over 1,960,000 results. This implies that Business Requirements already exist and that they are ready to be scooped up and used to create the business process that will "revolutionize your business". There are many applications available on the market that allow you to capture and organize business requirements. However, to date, the author is not aware of any tools (or methodologies) that exist to create (let alone design) Business Requirements.

What is wrong with Gathering Requirements?

Virtually everyone in the Business Process Management field gathers business requirements. They use interviews, workshops, focus groups, brainstorming and other techniques to gather the requirements (Liles, 2012). Everyone is doing this, so what is the problem? One can never be absolutely certain that business requirements are accurate and complete (Wikipedia, Business Requirements, 2014). In regards to completeness, it has been my experience that most business analysts stop gathering requirements when they run out of time and/or when they run out of people to talk to. They assume that they have talked to ALL the Subject Matter Experts (SME's) who collectively know ALL the business requirements. In regards to accuracy, it has also been my experience that most business analysts assume that since the SME's know EVERYTHING about the business they must know all the business requirements and they surely know when one is incorrect or is no longer needed. This is a very dangerous assumption, as all SME's are human, have limited perspective and are not omniscient.

The discipline of defining business requirements has become a hunting and gathering activity. Table 1 identifies a variety of business requirements gathering activities that IIBA suggests as part of its recommended business analysis techniques (International Institute of Business Analysts, 2009, p. 53).

Elicitation Techniques
Brainstorming
Document Analysis
Focus Groups
Interface Analysis
Interviews
Observation
Prototyping
Requirements Workshops
Survey/ Questionnaire

Table 1 - IIBA Elicitation Techniques

Although these activities will help the SME think about what requirements they might have for the process, they do not provide a systematic method that will ensure a complete set of requirements is developed and that obsolete or contrary requirements are identified. It does not produce a Requirements Design. It is still just hunting and gathering.

What is the evolution of Business Design?

There is a parallel between the evolution of society and the evolution of business design. The hunting and gathering society was replaced by the horticultural society (Grenier, Chapter 6: Horticultural Societies). Through the use of inventions and discoveries, society evolved to the agrarian society (Grenier, Chapter 7: Agrarian Societies). It was a progression that took time.

Business design has evolved in a similar manner.

When the modern programmable computer was created in 1945 (Wikipedia, Computer, 2013), it was programmed through a binary language commonly known as Machine Language. It took very complex code to do very simple things. Arithmetic was done in binary using “Shift”, “And” and “Or” commands.

Then came programming languages: FORTRAN for the scientist; COBOL for the business community. These languages allowed the programmer to think in more natural terms, at least for doing calculations (which is what everyone thought was the role of the computer). There was a lot of trial and error in programming languages as the industry tried to figure out what was needed. Languages like ALGOL, SNOBOL, even APL (which stood for “A Programming Language”) were created and enjoyed a brief moment of acceptance as a new way of thinking about computation. But as people tried to apply computers to life situations, the complexity of the programming grew and the need for structure and design of programs became apparent. Structured languages that supported procedures and subroutines were developed or added to existing languages. FLECS (Richmond, 2001), PL/1 (Wikipedia, PL/I, 2014) and COBOL (Wikipedia, COBOL, 2014) all sought to be structured languages that people could use for using computers to meet real life needs.

IN the early 1970’s, these languages provided the power to meet the need as companies started to move beyond the scientific and business calculations to build work management applications for the work place. As the complexity rose, the need to manage the code became apparent. “Spaghetti” code (code that worked but was not organized) became the nightmare of maintenance programmers. The complexity of automation required the expansion of the computer disciplines to include Program Design as a way to manage the complexity. The flowchart (Wikipedia, Flowchart, 2014) was used to visualize the complex world of computer code.

By the late 1970’s business leaders began to be understood that computers needed to be an integral part of their business, and not just a tool (a fancy calculator). At this point computer analysis got its start. Ed Yourdon was amongst the first to propose a structured approach to describing the use of the computer as a system of automations. His text “Structured Analysis and System Specification” (DeMarco, 1978) enjoyed a large following of professionals trying to design how business would productively use the computer (which at this time was starting to provide remote access to the main-frame computer – which still resided in an air-conditioned room and was the size of an automobile).

Data flow diagrams soon became the standard for process design just as Flowcharts had become for program design. Over the years the level of detail of the data flow diagram has been debated and many alternatives have been proposed, but even today, most process design methodologies, including the current BPMN (Object Management Group, 2011) use some version of the data flow diagram.

But even these process design methodologies were missing something. They needed to know what the process was to accomplish, they needed to know what the business needed from the process. Most processes, even with the best designs, struggled to provide business with the tools needed because they didn't understand the requirements of the business (IAG Consulting, 2009, p. 3). Many approaches have been tried to make sure that the business processes understand the requirements they must meet to fulfill the larger business objectives. Some have proposed looking at the corporate goals and objectives (Wikipedia, Business Requirements, 2014). This is helpful, but they do not provide the details needed for a complete process design. So far, the most success approach has been to gathering the requirements from the parent business unit. But even that has not been very successful. It has been reported that when requirements are gather that "later, when those requirements are used to build and verify model(s), gaps in the requirements may be discovered" (International Institute of Business Analysts, 2009, p. 53). The lack of discipline methodology in gathering (or developing) these requirements appears to be the problem.

Where should Requirements come from?

The main problem with requirements gathering, besides the completeness problem described above, is that it is Process Centric. It assumes that if I optimize all my processes, that I will have the best business. As my colleague Ed Gibson has written (Gibson Jr., 2012, p. 1):

> Dr. Ackoff explains Systems Theory (Ackoff, 1970, p. 9) and how it applies to corporations as complex systems. He observes that in order to optimize a systems operation you have to de-optimize the operation of its component parts. In his paper Dr. Ackoff presents the following theorem:
>
>> If you take a system and take it apart to identify its components, and then operate those components in such a way that every component performs as well as it possibly can, then there is one thing of which you can be al- most certain – the system as a whole will not perform as well as it can.
>
> The corollary to this theorem is:
>
>> If you have a system that is performing as well as it can – then none of its parts will be (Ackoff, 1970, p. 9).
>
> At face value, these are very counterintuitive propositions. The premise behind the re-engineering concept suggests that optimizing business processes would optimize overall corporate performance. Systems theory, on the other hand, says that this is an unreasonable expectation.

We have seen our industry go from Structured Languages, to Structured Applications to Structured Processes. We at MetaPower believe that our industry needs to have a structured methodology for developing business requirements. This methodology must address not only the completeness problem, but also the sub-optimization problem. Our colleague, Ken Allen, developed such a methodology in early 1990's. He called this methodology "Program Design" and was an integral part of a structure, disciplined approach to designing and deploying comprehensive business process that are optimized to achieve specific business objectives.

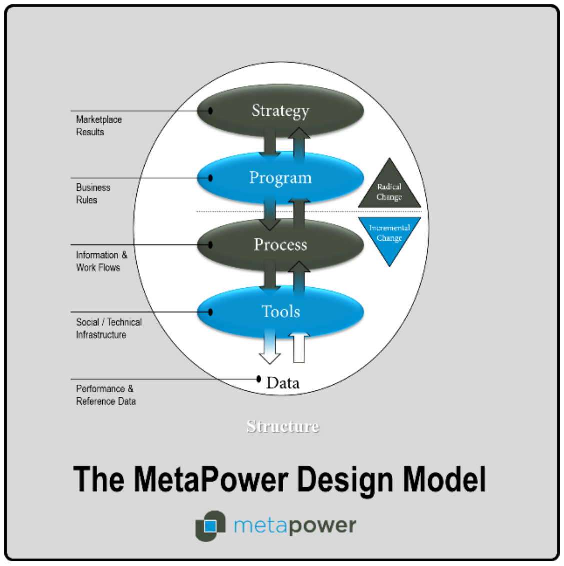
What is a Program? First of all we are not talking about computer programs (applications). The discipline of Program Management is "the process of managing several related projects, often with the intention of improving an organization's performance." (Wikipedia, Program Management, 2014) We at MetaPower see the performance goals of the program being defined by Strategic design. There are many business design methodologies today that include performance goals and objectives in their suite of disciplines. These goals and objectives define the success of the business. We have found that they are the basis of Program Design. A Program Design is a "formulated" specification for what is required of the company's processes in order to achieve specific goals or objectives.

In some respects Program Design is not a new concept. It actually is inherent in any modern corporation. When a business is initially established, the founders must engage in some "programmatic thinking" in order to determine what they need to do. They need to determine what needs to be done in order to be successful (as they define success). They may not realize that they are de-optimizing processes; they are only concerned with what needs to happen to be successful (e.g. deliver the product to the customer in less than 48 hours). It is out of this programmatic thinking that the requirements of the business are created. Only when this thinking is done in a systematic and logical fashion will we get systematic and logical requirements.

The Design Model

Ken Allen, Business Analyst and founder of MetaPower, in working out the relationship between corporate strategy and process implementation, developed the Design Model (Figure 1). It presents the design framework inherent in any enterprise.

The model shows the relationship between the five major elements of the enterprise design. Strategy is at the top of the model and basically specifies the results the corporation needs to have in order to be successful. This information is passed to the next level, the Program Level, as goals or objectives that must be achieved in order to get the strategic results.



For each goal or objective, the Program level designs what must be done in order to accomplish the goal or objective. These "what's" (business requirements) are passed to the Process level that are to be implemented by the various processes in the corporation.

The Process level develops Business Process Designs that determine how to do what is required by the programs. While it is beyond the scope of this paper to discuss this level in detail, it is worth noting that Process Designs specify the functional requirements (information systems, procedures, and such) for the tools necessary to implement the processes.

The Tools level is where the tools of the work are designed and implemented. We tend to think of this as only software design and construction, but from an enterprise perspective, it includes all aspects of tooling (for example: software, equipment, forms, procedures, tools, personnel and organization positions).

The Data level is a part of every level and a level of its own; it provides a vocabulary and definition of the information of the enterprise. Each level has its own data modeling techniques that are used to capture the data requirements of that level. Many methodologies consider the definition of data to be the domain of application design and not Process (or even Program) design. MetaPower concurs with the Business Rules Solution admonition: “A shared, well-structured business vocabulary is fundamental for requirements” (Ross, 2013, p. 20).

The key to the Design Model is that it avoids the Law of the Instrument (Wikipedia, Law of the Instrument, 2014):

> "I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail."

Each level uses its own design methodology. Most professionals are familiar with the standard design methodologies for data, applications and process (as described earlier in this paper). What MetaPower is proposing is that Programs be designed.

Program Design

Program design begins with identifying a sustainable, on-going performance. These come from the Strategic Design as goals or objectives. The key is to define the goal of the program, not what you want to change but what you want as a sustainable, on-going performance. Some examples of program goals are:

- Zero Accidents / Incidents (Safety)
- Zero Defects (QA)
- Living within our financial means (not spending more than we make or finance) (Finance)
- 100% Operational Availability (Maintenance)

Using the strategic goal, a Program design is developed that determines the Essential Business Logic needed to be accomplished by the organization’s suite of business processes. Most BPM practitioners are familiar with Process Design. Program Design utilizes the same principles as Process design with the following exceptions:

- The focus of Program Design is “What” needs to happen in order to achieve the sustainable, on-going performance. It does not focus on the “How” or “Who”.
- There are no mistakes, so there is no need for approvals or other checks and balances.
- There are no physical forms or technology; Program Design defines only essential data.

Some practitioners of Process Design may take short cuts and not fully define the process, but in doing Program Design, those shortcuts cannot be taken. Program design requires all three aspects of business design:

- Data Flow Diagrams (DFD) are graphical representations, similar to flow charts. The DFD illustrates how individual program activities are linked by information (or data) to make up the overall program. It is the central design document and is further defined through the Data Dictionary and Logic Specifications documents.

- The Data Dictionary defines the meaning of all the data that is presented on the Data Flow Diagram. The document includes not only the English definitions, but it also describes the detailed sub-data elements that are contained in the data flows illustrated on the DFD. The data of a Program Design is the "essential" data – only the data needed to perform the activity. It is not concerned with the physical representation of the data (e.g.: forms, screens, records, etc.) as that is the jurisdiction of the Process Design.

- Logic Specifications are simple, structured English statements of the logic that describe the activities depicted in the Data Flow Diagram. The Logic Specification must describe how the data that leaves each process is created from the data that enters the process. The objective of the Logic Specification is to prove that the original process problem has been solved and all the data entering and leaving the process is consistent and coherent. The Logic Specification also identifies the Role that will be responsible for the program activity.

These rules and definitions are the basis by which processes can be expected to work together to achieve the program goals. Through the development of the Program Design, new fundamental assumptions (axioms) of rules are also identified. These assumptions need to be documented so that they can be accommodated in the Process Designs that will follow.

Program to Process Transition

The Program Design is not an end in itself. It is the bridge between Strategy and Process. It determines what the organization's business processes need to do in order to sustain the on-going performance needed to achieve the business strategy. It is used to develop direction (requirements) for the business processes. The direction is two-fold:

- Business Requirements

- Process Mapping

While defining the logic and data needed to sustain on-going performance, some of the data and logic is essential to the strategic goal and some is not – it is needed in order to provide the context for the essential logic and data. The first step in the transition from Program to Process is to identify the logic and data that is essential to the Program. This essential logic and data, without which the Program would not achieve the sustained on-going performance, is what is called Business Requirements.

A Business Requirement is something that is either done or known. This is similar to the behavioral rules that govern business conduct and the definitional rules that govern business knowledge described by Business Rules Solutions (Ross, 2013, p. 91).

The second step in the transition from Program to Process is to identify for each Business Requirement the Process that needs to implement the Business Requirement. During the Strategic design, the Process Architecture is defined. This gives us the inventory of business processes to which we can map the requirements.

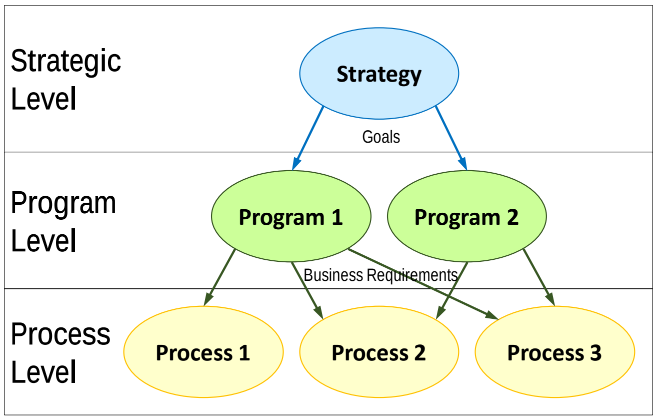


Figure 2 - Process Design Model Transitions

This mapping of Business Requirements to Processes is a many-to-many relationship. A Program Design will create Business Requirements that will be implemented by many Processes. And at the same time, a Process can be required to implement Business Requirements from many Programs. It is up to each process to determine HOW to meet all the Business Requirements required of it. Figure 2 illustrates these relationships. In this graphic, strategy has two goals that each pass to a program. Each Program then develops the Business Requirements that processes must use in executing the business of the organization. The Programs develop the process rules without consideration of the other Programs and their goals. If there are conflicts in these requirements, they must be identified at the process level and resolved to the mutual satisfaction of each Program. As illustrated in the figure, all Processes may not be affected by all Programs. But each Program will affect at least one Process.

Conclusion

Program Designs provide a logical approach to defining Business Requirements. Business Requirements can be deleted if the Program designers agree that the Business Requirements are not required by the Program's essential logic. If we have designed all the Programs needed to implement the corporate Strategy, then we know that we have ALL the requirements for the Business Process. If a Business Process is unable to find a solution that meets ALL its assigned Business Requirements, it must negotiate with the Program owners to resolve the conflicting Business Requirements.

Using Program Designs we no longer collect Business Requirements, but design them based on the corporate strategy within the context of the Process Architecture.

Bibliography

Ackoff, R. L. (1970). *The Second Industrial Revolution.* Retrieved Dec 26, 2014, from In Thinking Network - Ongoing Discussions: in2in.org/od/thought/Ackoff-SecondIndustrialRevolution.pdf

DeMarco, T. (1978). *Structured Analysis and System Specification.* New York: Yourdon Press.

Gibson Jr., E. G. (2012). *Performance Programs.* Vancouver, WA: MetaPower, Inc.

Grenier, D. G. (n.d.). *Chapter 6: Horticultural Societies.* (Florida International University) Retrieved Nov 5, 2014, from Florida International University: Miami's Public Research University: www2.fiu.edu/~grenierg/chapter6.htm

Grenier, D. G. (n.d.). *Chapter 7: Agrarian Societies.* Retrieved Nov 5, 2014, from Florida International University: Miami's Public Research University: www2.fiu.edu/~grenierg/chapter7.htm

IAG Consulting. (2009). *Business Analysis Benchmark 2009 - the Path to Sucess.* New Castle, DE: IAG Consulting.

International Institute of Business Analysts. (2009). *A Guide to the Business Analysis Body of Knowledge* (2.0 ed.). Toronto, Ontario, Canada: www.iiba.org.

Liles, J. (2012, May 14). *Methods for Eliciting - not gathering - Requirements.* Retrieved from Business Analyst Times: www.batimes.com/articles/methods-for-eliciting-not-gathering-requirements.html

Object Management Group. (2011). *Business Process Model and Notation - Version 2.0.* Retrieved from OMG: www.bpmn.org

Richmond, C. (2001, Aug 16). *FLECS.* Retrieved from alt.folklore.computers: http://www.inwap.com/pdp10/usenet/flecs

Ross, R. G. (2013). *Business Rule Concepts - Getting to the Point of Knowledge* (4th ed.). Business Rules Solutions, LLC.

Wade, J. F. (2011, Nov 2). *Where do things come from?* Retrieved from The Daily Bell: www.thedailybell.com/editorials/3176/Joel-F-Wade-Where-Do-Things-Come-Frome d

Wikipedia. (2013, Dec 31). *Computer.* Retrieved Jan 02, 2014, from Wikipedia: en.wikipedia.org/wiki/Computer

Wikipedia. (2014, Nov 20). *Business Requirements.* Retrieved Nov 26, 2014, from en.wikipedia.org/wiki/Business_requirements

Wikipedia. (2014, Nov 19). *COBOL.* Retrieved Nov 20, 2014, from en.wikipedia.org/wiki/COBOL

Wikipedia. (2014, Nov 18). *Flowchart.* Retrieved Nov 20, 2014, from en.wikipedia.org/wiki/Flowchart

Wikipedia. (2014, Nov 2). *Law of the Instrument*. Retrieved Nov 28, Nov, from https://en.wikipedia.org/wiki/Law_of_the_instrument

Wikipedia. (2014, Oct 5). *PL/I*. Retrieved Nov 20, 2014, from en.wikipedia.org/wiki/PL/I

Wikipedia. (2014, Oct 29). *Program Management*. Retrieved Nov 28, 2014, from https://en.wikipedia.org/wiki/Program_management