Ryan Somma

The Systems Analyst

as

Polymath

I. Introduction

The book *Systems Architecture* describes the function of the systems analyst as performing "activities of the business modeling and requirements disciplines," and goes on to expand the possible responsibilities into "business modeling, requirements, design, and management of a development project (Burd, 2006)." As this paper shall illustrate, this short description is an incredible understatement of the functions performed by the systems analyst on a project, which requires skills and methodologies that venture far outside the traditional fields of knowledge needed in business.

The systems analyst must have a fundamental understanding of computers, from storage hardware and software, to networks and protocols, to hardware and software inputs and outputs, programming languages, operating systems, and utilities, as well as communications and collaboration technologies. Additionally, they must possess knowledge and skills of the organization's business environment in which they are operating, including the organization's business functions, structure, management, and type of work. Finally, they must also have finely-developed interpersonal skills, which includes the ability to communicate with managers, users, programmers, technical specialists, customers, and vendors, translating functional and technical requirements from one group into verbiage and context that another group can understand (Satzinger, 2009).

It is impossible for a single person to possess expertise in all of these things; otherwise, there would be no need for managers, programmers, and the wide variety of users performing their specialized tasks. There would only be the systems analyst, or an army of systems analysts, running all aspects of the system. The specialization of labor,

having individuals concentrate on and become experts at a specific area of knowledge, leads to the most efficient execution of business (Smith, 1776).

The systems analyst supplements existing business roles and functions, examining the processes and exploring ways to optimize and automate them, improving quality and performance along the way. The role of the systems analyst is that of an investigator. Within the boundaries of the organization's system, they are historian, anthropologist, and cyberneticist, immersing themselves in the organization's complexities; as well as a gardener of sorts, managing the organization's existing resources to grow a solution from the infrastructure. The systems analyst's primary talent and responsibility, therefore, is *understanding*.

II. Managing Complexity

Edsgar Dijkstra, in his 1972 ACM Turing lecture, captured the incredible scope of what someone working on an information system must understand in detail:

> In computer programming our basic building block has an associated time grain of less than a microsecond, but our program may take hours of computation time. I do not know of any other technology covering a ratio of 10^{10} or more: the computer, by virtue of its fantastic speed, seems to be the first to provide us with an environment where highly hierarchical artefacts are both possible and necessary. This challenge, viz. the confrontation with the programming task, is so unique that this novel experience can teach us a lot about ourselves. It should deepen our understanding of the processes of design and creation, it should give us better control over the task of organizing our thoughts (Dijkstra, 1972).

Dijkstra speaks to computer programming specifically, but the systems analyst's understanding must exceed even these ten orders of magnitude. True, they do not need to know the technical details of how the system's clock synchronizes with the execution of software, but they do need to understand how the megahertz of the processor impacts the performance of the application. They don't need to know the specifics of how free-form entry field complicates the algorithms required to pull meaningful summarization

information out of database records for executive reports, but they do need to know how the data-entry and reporting requirements can conflict and overly-complicate the programmers' tasks.

As was mentioned in the previous section, the systems analyst cannot and should not attempt to become an expert in all aspects of the organization; however, they do need to understand enough of the system to speak intelligently to all levels of operations, from manager to user to technician. One of the most effective devices in the systems analyst's cognitive toolbox for managing complexity is *functional decomposition*, "dividing a system into components based on subsystems that are further divided into smaller subsystems (Satzinger, 2009)." Rather than become an expert in the entire organization at all levels at once, the systems analyst instead strives to become an expert at a specific subset of the system. Through this technique, a highly-complex system may be understood by reducing the scope of what the analyst needs to understand.

III. Systems Analyst as Cyberneticist

Norbert Wiener defined cybernetics as "the science of communication and control in the animal and the machine (Wiener, 1948)." Whether the type of information system is a customer relationship, supply chain, accounting and financial, human resource, manufacturing, or knowledge management system, they all involve an information system running on machines interacting with human animals. There is a reciprocal relationship between users and the system, where users input data to the system, the system outputs information in reports to users, and a feedback loop is generated.

We tend to think of cyborgs as humans augmented with artificial prostheses, and an organization's information system serves as a cognitive prosthesis, augmenting what its human users are capable of remembering with databases and file storage, and enhancing the information they may derive from the stored data with algorithms to produce reports. The systems analyst, in working to understand these interactions between users and the information system as a complete system, is working within the discipline of cybernetics.

IV. Systems Analyst as Historian

"It takes years of experience working for a company to really understand what is going on," the fifth edition of the textbook *Systems Analysis & Design* explains, "The more an analyst knows about how an organization works, the more effective he can be (Satzinger, 2009)." Before a systems analyst may dare seek to express any conclusions about where the system should go, they must ensure that they understand where the system exists in the present, with all of its idiosyncrasies and the reasons for them:

> One of the most common errors of young, impetuous analysts is loudly to castigate the developers of the existing system, only to discover that:
>
> 1. There were, at the time, good and sufficient reasons for decisions that seem idiotic today.
> 2. The original developer is now the analyst's manager, or manager's manager. (Weinberg, 1982)

To avoid this blunder, the systems analyst must exercise the methods and skills of the historian, principally going to primary sources and documents.

Systems analysts must review existing reports, forms, and procedure descriptions. This includes forms that have been filled out, in order to ensure the data entered into the form matches the form's structure. They must interview stakeholders in order to

understand the system, as it exists today, and the history of how it came to exist the way it does (Satzinger, 2009).

V. Systems Analyst as Anthropologist

In the academia, there is often a reference to "the two cultures," the humanities and the sciences, and the very different ways these two areas of knowledge approach understanding the world. Similarly, in the business world:

> We are likewise faced with two cultures in developing a computer system. Guy Kawasaki in *The Macintosh Way* calls them "T-Shirts" and "Ties" for their sartorial preferences: T-Shirt are commonly worn by Technologists such as programmers; Ties are de rigueur in the business departments in typical companies and marketing departments in high-tech firms… As a systems analyst, a critical part of your job is to provide a bridge between these two cultures; you must be a bicultural to bring the two disciplines together (McDermott, 2002).

Even these two cultures could be further divided into sub-cultures. The technologists form tribes, with systems engineers having a culture distinct from software developers distinct from help desk workers. Although all these silos will end up working with the same information system, they have very distinct perspectives of their responsibilities to it and how they will use it, and the systems analyst must understand them.

"Anthropologists have a lot in common with systems analysis, for both of them have to enter unfamiliar worlds and extract reliable information (Weinberg, 1982)." Like Jane Goodall, who lived with chimpanzees in the wild, immersing herself in their way of life in order to understand their culture, the systems analyst must embed his or herself in the organization's business culture. The systems analyst must perform a walkthrough of where the organization's work is performed in order to gain a general understanding of the business process. They must embed themselves with the users, spending several hours

observing them at their jobs, even going so far as to become one of them by being trained as a user and performing the tasks themselves (Satzlinger, 2009).

VI. Systems Analyst as Educator

In one sense, the systems analyst must stand outside of the system that they are studying, taking the role of a dispassionate observer. At the same time, system analyst is a part of the system in which they are operating, part of the organization for which they are developing a solution. All of this research for the purpose of problem solving is worthless if the systems analyst cannot communicate it to all levels of the organization.

The metaphor is one of the most powerful tools at the systems analyst's disposal for communicating concepts to various silos within an organization. The Operating System desktop is a metaphor for the machine-level operations occurring way down inside the hardware of the physical computer system. An organization chart is a metaphor for how teams and individuals are structured to work together within an organization. For the systems analyst, DFDs, ERDs, use case diagrams, class diagrams, sequence diagrams, and others are models used to describe the abstract aspects of the system into a visual metaphor more easily understandable across cultures within the organization (Satzinger, 2009).

VII. Systems Analyst as Gardener

In the traditional waterfall model of the systems development lifecycle, the assumption was made "that the various phases of a project can be carried out and completed entirely sequentially (Satzinger, 2009)." This highly structured approach to the

SDLC requires a great deal of planning and prediction, as each phase must be completed before the next phase can begin. As a result of this strict and stringent approach to development, adaptability is lost. Each phase must get things right the first time, because the subsequent phases, depending on where they fall in the process, demand completeness in the preceding phase, be it planning, analysis, design, or implementation. Because such a strict approach to SDLC attempts to manage a very high degree of complexity, it is also subject to a much higher degree of failure.

In contrast, an iterative approach to SDLC, such as the spiral model, evolves a system incrementally, growing closer and closer to what the customer needs by degrees. Development methodologies such as Extreme Programming (XP) and Scrum are lightweight, adaptive, and self-organizing (Satzinger, 2009). The systems analyst must understand and take advantage of the self-organizing character of these approaches. The analyst is not the systems engineer; he or she is not the one who will be writing the code or setting up the hardware.

The systems analyst must complement the development process by providing the necessary requirements to the engineers, but that is all they can do. They must allow the engineers, who are the experts in how the logical models will be physically executed, to execute the systems construction. The systems analyst must understand the boundaries of their role so that they may stand back and allow the emergent phenomenon to develop on its own, like a gardener watching their flowers grow, only interceding to nurture the process with requirements and the seeds of information.

VIII. Conclusion

As Frederick Brooks noted in his historic essay, *No Silver Bullet*, there is no escaping the complexity of software development because:

> The complexity of software is an essential property, not an accidental one. Hence, descriptions of a software entity that abstract away its complexity often abstract away its essence. For three centuries, mathematics and the physical sciences made great strides by constructing simplified models of complex phenomena, deriving properties from the models, and verifying those properties by experiment. This paradigm worked because the complexities ignored in the models were not the essential properties of the phenomena. It does not work when the complexities are the essence (Brooks, 1987).

The best the systems analyst can do then is to reduce the scope of what they are attempting to understand to a subset of the whole system. While this technique reduces the scope of the complexity, they must still seek to understand the system the way a Cyberneticist understands it, as a feedback loop between artificial construct and human users. Two crucial techniques to understanding the system cybernetically is to research its history, how it came to exist in its present form, and to study it as an anthropologist, embedding oneself in the organizations culture to understand its business practices.

Once the system analyst has and understanding of the system, they must be able to educate others on it. They must sell solutions to management, communicate technical requirements to engineers, and instruct users on how to best integrate the system into their daily activities. They must communicate different aspects of the system to different cultures within the organization without over-simplifying or avoiding the complexity:

> The designer is human, and the human mind is limited in its ability to predict the behavior of an unbuilt system. To overcome this limitation, we often aid the mind with analytical tools, such as diagrams and equations. But more often we use the opposite approach, often unconsciously. Instead of increasing the power of our mind, we decrease the complexity of the problem. Instead of considering all potential variations of a design, we omit certain cases because we cannot analyze them with our limited minds and tools (Weinberg, 1982).

An important aspect of not excluding design variations is to allow the experts to act on their own knowledge. The system analyst must understand their personal limits and the boundaries of their expertise, allowing software and systems engineers to do their jobs and meet the defined requirements with the technical solution they believe is best. The systems analyst, in this regard, is like a gardener, guiding and monitoring an emergent phenomenon, understanding what they can do to foster it, and where they must leave the process to another kind of expert.

The systems analyst, therefore, is not a "jack of all trades, master of none," but rather a polymath. They must be well educated and excel in a variety of fields. The systems analyst is an expert at many different, highly intellectual disciplines. They are academics on a variety of levels, teachers to a variety of aptitudes, and a very unique type of expert in their own right.

IX. References

Brooks, Frederick P., Jr. (April 1987). *No Silver Bullet: Essence and Accidents of Software Engineering*, Computer Magazine.

Burd , Stephen D. (2006). *Systems Architecture Fifth Edition,* Thomson Course Technology.

Dijkstra, Edsger W. (1972). *The Humble Programmer*, ACM Turing Lecture.

McDermott, Patrick (2002). *Zen and the Art of Systems Analysis*, Writers Club Press, Lincoln, NE.

Satzinger, John W., Jackson, Robert B., Burd, Stephen D. (2009). *Systems Analysis and Design in a Changing World, Fifth Edition*, Course Technology, Boston, MA.

Smith, Adam (1776). *An Inquiry into the Nature and Causes of the Wealth of Nations*, W. Strahan and T. Cadell, London.

Wiener, Norbert (1948). *Cybernetics or Control and Communication in the Animal and the Machine*, Massachusetts Institute of Technology,.

Weinberg, Gerald M. (1982). *Rethinking Systems Analysis & Design*, Dorset House Publishing Co. Inc., New York.