A Different Kind of Life Cycle: The Zachman Framework

David C. Hay
Essential Strategies, Inc.

Introduction – The System Development Life Cycle

Many methodologies are organized around the "system development life cycle," an organization of the steps required to develop systems. Oracle's is shown in Figure 1, expressed specifically as consisting of:

- ✓ ***Strategy*** – The planning of an organization's overall systems development effort. This includes determining the overall set of things of significance to a business, the application areas to be addressed, and the priorities to apply to those priorities.
- ✓ ***Analysis*** – The detailed definition of requirements for a particular area of the business. At this point, the data structures are mapped in detail, the functions of the business are described thoroughly, and the areas to be automated are defined.
- ✓ ***Design*** – The specific application of technology to the requirements defined during analysis. Here the data structures become database designs and the function definitions become program specifications. At this point attention is paid to the human interface, in the interest of defining the behavior of a prospective system.
- ✓ ***Construction*** – The actual building of the system.
- ✓ ***Documentation*** – Preparation of the user manuals, reference manuals, etc. to describe the system.
- ✓ ***Transition*** – The implementation of the system, so as to make it part of the infrastructure of the organization. This involves education, training, definition of new organizational structures and roles, and the conversion of existing data.
- ✓ ***Production*** – The ongoing monitoring of the system to ensure that it continues to meet the needs of the organization.

Notice that each of these steps addresses issues of data and function. Data and functions are typically addressed as separate topics, although ideally, they are addressed together.

Most methodologies portray the system development life cycle in terms approximating these. Some go so far as to give it the acronym "SDLC."

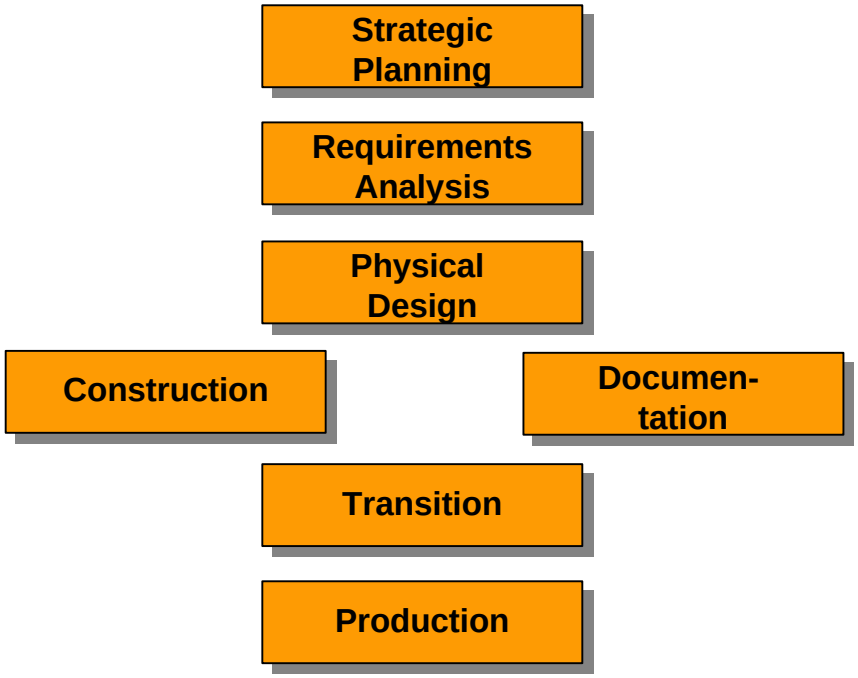

Figure 1: The System Development Life Cycle

The Zachman Framework

In 1987, John Zachman published a different approach to the elements of system development.[1] Instead of representing the process as a series of steps, he organized it around the points of view taken by the various players. These players included (1) the CEO or whoever is setting the agenda and strategy for an organization, (2) the business people who run the organization, (3) the systems analyst who wants to represent the business in a disciplined form, (4) the designer, who applies specific technologies to solve the problems of the business, and finally, (5) the system itself. Mr. Zachman represents each of these perspectives as a row in his matrix.

He then defined columns in the matrix to represent the kinds of things people should be looking at. These include functions and data, as addressed by most methodologies. In addition, however, Mr. Zachman has set up columns to represent locations (where business is conducted), the people and organizations involved, events which cause things to happen, and the motivations and constraints which determine how the business behaves.

John Zachman's "framework for information systems architecture" is diagrammed in Figure 2. The rows represent the points of view of different players in the systems development process, while columns represent different aspects of the process. The players are:

1. Scope (Ballpark view): Definition of the enterprise's direction and business purpose. This is necessary to establish the context for any system development effort.

---

[1] J.A. Zachman, "A framework for Information Systems Architecture," *IBM Systems Journal*, Vol. 26, No. 3, 1987.

2. Model of the business (Owner's view): This defines — in business terms — the nature of the business, including its structure, functions, organization, and so forth.

3. Model of the information system (Architect's view): This defines the business described in step 2, but in more rigorous information terms. Where row two described business functions, for example, as perceived by the people performing them, row three describes them specifically as transformations of data. Where row two described all the things of interest to the enterprise, row three describes those things about which the organization wishes to collect and maintain information, and begins to describe that information.

4. Technology model (Designer's view): This describes how technology may be used to address the information processing needs identified in the previous rows. Here relational databases are chosen over network ones (or vice versa), kinds of languages are selected and program structures are defined, user interfaces are described, and so forth.

5. Detailed representations (Builder's view): Here a particular language is chosen, and the program listings, database specifications, networks, and so forth are all produced.

6. Functioning system: Finally, a system is implemented and made part of an organization.

The columns in the Zachman framework represent different areas of interest for each perspective. The columns describe the dimensions of the systems development effort. As shown in Figure 2, these are:

1. ***Data:*** Each of the rows in this column address understanding of and dealing with an enterprise's data. This begins in ***row one*** with an list of the things that concern the company and affect its direction and purpose. ***Row two***, is a contiguous model of the things seen by the participants in the business. Many-to-many and n-ary relationships may be present, reflecting the way the business views them. Also, relationships may be shown which themselves have attributes. ***Row three*** provides more of an information-based perspective, resolving many-to-many and n-ary relationships, along with relationships containing their own attributes. Indeed, attributes are more exhaustively defined, and unique identifiers are specified. Entities are generalized to more closely reflect the underlying structure of the business and its relationships. In ***row four***, entities are converted to table definitions, object classes, hierarchy segments, or whatever is appropriate for the kind of data base management system to be used. This is tantamount to creating the data definition language statements. In ***row five***, the tables are actually implemented on physical disk drives, using the underlying organization of the database management system. This is where tablespaces are defined, disk packs are allocated, and so forth. The actual database itself is created and initial data are converted and loaded for ***row six.***

2. ***Function:*** The rows in the function column describe the process of translating the mission of the enterprise into successively more detailed definitions of its operations. Where ***row one*** is a list of the kinds of activities the enterprise conducts, ***row two*** describes these activities in a contiguous model. ***Row three*** portrays them in terms of data transforming processes, described exclusively in terms of the conversion of input data into output data. The technology model in ***row four*** then converts these data conversion processes into the definition of program modules and how they interact with each other. Pseudo-code is produced here. ***Row five*** then converts these into source and object code. ***Row six*** is where the code is linked and converted to executable programs.

   Note that in the object-oriented approach, functions and data tend to be addressed together.

3. ***Network:*** This column is concerned with the geographical distribution of the enterprise's activities. At the strategic level (***row one***), this is simply a listing of the places where the enterprise does business. At ***row two***, this becomes a more detailed communications chart, describing how the various locations interact with each other. ***Row three*** produces the architecture for data distribution, itemizing what information is created where and where it is to be used. In ***row four***, this distribution is translated into the kinds of computer facilities that are required in each location, and in ***row five***, these facilities requirements are translated into specification of particular computers, protocols, communications facilities, and the like. ***Row six*** describes the implemented communications facilities.

4. ***People:*** The fourth column describes who is involved in the business and in the introduction of new technology. The ***row one*** model of people is a simple list of the organizational units and each unit's mission. In ***row two***, this list is fleshed out into a full organization chart, linked to the function column. Here also, requirements for security are described in general terms. In ***row three***, the potential interaction between people and technology begins to be specified, specifically in terms of who needs what information to do his job. What roles do each play and what data are necessary for those roles? Along with this are specific definitions of security requirements, in terms of who (which role) is *permitted* access to what. In ***row four***, the actual interface between each person and the technology is designed. In this row, issues of interface graphics, navigation paths, security rules and presentation style are addressed. In ***row five,*** this design is converted into the outward appearance of each program, as well as the definitions of access permissions in terms of specific tables and/or columns each user can have access to. In ***row six,*** you have trained people, using the new system.

5. ***Time:*** The fifth column describes the effects of time on the enterprise. It is difficult to describe or address this column in isolation from the others, especially column two. At the strategic (***row one***) level, this is a description of the business cycle and overall business events. In the detailed model of the business (***row two***), the time column defines when functions are to happen and under what circumstances. ***Row three*** defines the business events which cause specific data transformations and entity state changes to take place. In the technology model (***row four***), the events become program triggers and messages, and the information processing responses are designed in detail. In ***row five***, these designs become specific programs. In ***row six*** business events are correctly responded to by the system.

6. ***Motivation:*** As Mr. Zachman describes it, this is concerned with the translation of business goals and strategies into specific ends and means. This can be expanded to include the entire set of constraints that apply to an enterprise's efforts. In ***row one***, the enterprise identifies its goals and strategies in general, common language terms. In ***row two,*** these are translated into the specific rules and constraints that apply to an enterprise's operation.

   In row three, business rules may be expressed in terms of information that is and is not permitted to exist. This includes constraints on the creation of rows in a database as well as on the updating of specific values.

   In row four, these business rules will be converted to program design elements, and in row five they will become specific programs. In row six, business rules are enforced.

	Data (What)	Function (How)	Network (Where)	People (Who)	Time (When)	Motiva-tion (Why)
Objectives / Scope	List of things important to the enterprise	List of processes the enterprise performs	List of locations where the enterprise operates	List of organizational units	List of business events / cycles	List of business goals / strategies
Business Owner's View	Entity relationship diagram (including m:m, n-ary, attributed relationships)	Business process model (physical data flow diagram)	Logistics network (nodes and links)	Organization chart, with roles; skill sets; security issues.	Business master schedule	Business rules
Architect's View	Data model (converged entities, fully normalized)	Essential Data flow diagram; application architecture	Distributed system architecture	Human interaction architecture (roles, data, access); Security requirements	Dependency diagram, entity life history (process structure)	Business rule model
Technology Designer's View	Data architecture (tables and columns); map to legacy data	System design: structure chart, pseudo-code	System architecture (hardware, software types)	User interface (how the system will behave); security design	"Control flow" diagram (control structure)	Business rule design
Builder's View	Data design (denormal-ized), physical storage design	Detailed Program Design	Network architecture	Screens, security architecture (who can see what?)	Timing definitions	Rule specification in program logic
Function-	(Working systems)					
ing system	Converted data	Executable programs	Communica-tions facilities	Trained people, using the system	Business events	Enforced rules

Figure 2: The Zachman Framework

Implications

This approach has several immediate effects on our understanding of the SDLC (see Figure 3):

- ✓ First, Mr. Zachman doesn't call them "phases" or "steps." Each row in his matrix represents the perspective of one of the sets of players in the development process. It is more important, he asserts, to recognize that systems are developed by distinct groups with different points of view, than that it is to see the movement of systems from one step to another.
- ✓ Second, the scope perspective is the one taken when doing strategic planning, just as the technical designer is responsible for the design step. Analysis, however, typically takes on two different

perspectives: one is to describe the situation in purely business terms, while the second, without yet addressing technology, describes the situation in information architectural terms.

- ✓ Third, he addresses more than data and functions. He establishes a matrix that encompasses, for each phase, data, function, location, people, time, and motivation.
- ✓ Finally, he does not address either documentation or transition explicitly. The matrix itself provides an organization for system documentation. And transition is the process of moving from the "as is" matrix to the "to be" matrix.

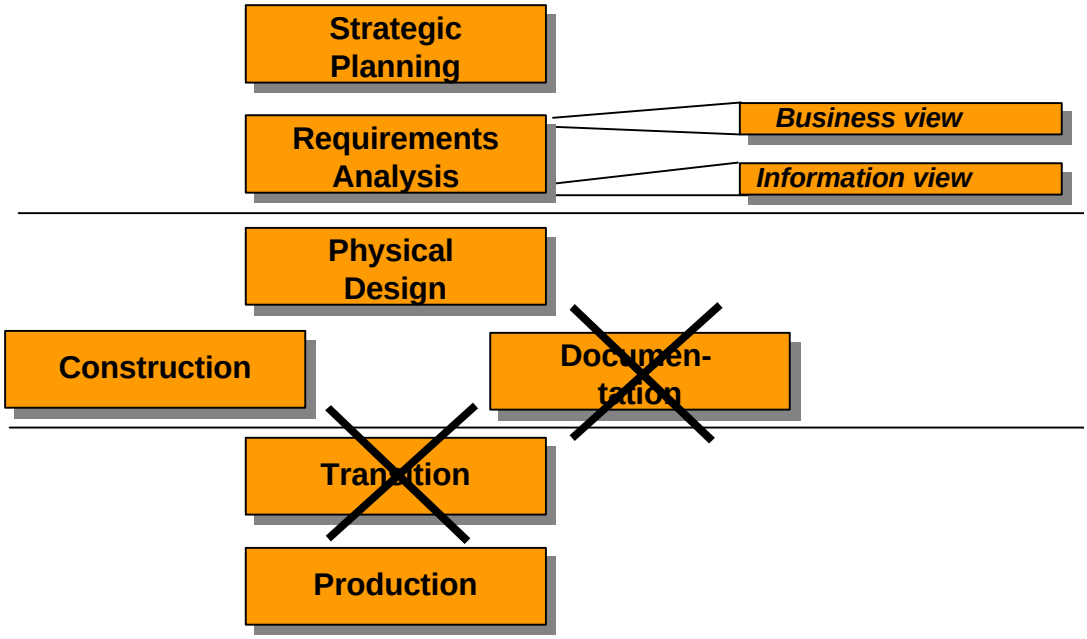

Figure 3: Life Cycle Steps and Perspectives

Let's look at those points in more detail:

Phases vs. Perspectives

Where the system development life cycle is the "process model" of the system development process, the Zachman Framework is analogous to the "data model". The framework describes the things of interest that the various players are looking at. The row 3 data model or the row 5 network architecture are models of the things that are of interest to the player at that level.

The Analysis Process

Where other methods look at analysis as a single process, the Zachman Framework makes an important distinction. As analysts who view the world in terms of information, it is hard sometimes to realize that not everyone sees things that way. It is illuminating to be forced to recognize that the terms of reference for the user community are not the same as ours.

For functions, this distinction is similar to Gane's and Sarson's distinction between current physical and current (or future) logical.[2] The sentiment is the same: Start with the user's view of the world and then go from there. On the data side, it is also similar to the data modeling approach of starting with "divergent models," in which entities represent current physical things seen by the users, and "convergent models," in which entities are more generalized abstractions, encompassing many physical object classes.

On the data side, there is something more specific at work here, however. Experienced data modelers have a tendency to make intuitive transformations from what they are told – thereby translating what is really row two information into row three models. Specifically, four elements are most noteworthy:

- ✓ The world at large has lots of many-to-many relationships. If you ask someone about customers and products, 'e will tell you that each product is sold to lots of customers, and (ideally) each customer buys lots of products. This is the description of the world as 'e sees it. It is probably worth while for us to recognize this and use it as a starting point, instead of immediately translating the situation into intersect entities.
- ✓ The world at large has "n-ary" relationships. When a student registers for a course from a teacher, that is a three-way relationship between student, course and teacher. Yes, we can resolve that into binary relationships, and probably do so intuitively immediately, but what the user is *telling us* is that it is a three-way relationship.
- ✓ The world at large has relationships with attributes. From your client's point of view, it is the relationship itself that has the attributes, not an objectification of that relationship. Again, we can objectify it as naturally as we breathe, but that is not exactly what we are being told.
- ✓ The world is not in third normal form, nor even in first normal form. A user will tell you about a purchase order with multiple line items, or a class with multiple students. As perceived by the business, these are multi-valued attributes.

These four elements are very specific examples of the kinds of intuitive leaps we analysts often take, without even being aware that we are doing so. What this means is that when we make these intuitive leaps, we are changing the perspective. We are dropping immediately into row three, without giving proper respect to row two. Yes, we can often get away with it if we are particularly good at explaining what we are doing, but often we are insensitive of the gap in perceptions that we have created.

The New Columns

---

[2] C. Gane, and T. Sarson, *Structured Systems Analysis: Tools and Techniques* (Englewood Cliffs, NJ:Prentice Hall, Inc., 1979.

The major contribution of the Framework is its explicit recognition that there is more at work here than functions and data. From the beginning, we should be recognizing the organizational issues; from the beginning, we should be dealing with multiple locations; from the beginning we should be explicitly concerned with timing – events, schedules, and so forth.

An interesting side effect of this is that now that object-oriented aficionados want to bring functions and data together, the matrix also shows us how to do that – simply merge columns one and two.

Another interesting aspect of all this is that we do not have models, or even well developed methods for dealing with many of the cells. John Zachman does not advocate the use of any particular modeling style for those cells where multiple techniques are available, and he is the first to recognize that in some cells no good techniques exist. It is difficult, for example, to model the logic (row three) of a distributed information network – at least in a way that links to our models for functions and data.

This represents an assignment for us all. He has pointed out things that we should be capturing and accounting for in our systems. It is for us to figure out how to do so.

Documentation

It can be argued that documentation is not really a discrete phase in the life cycle. Its presence as a phase in Oracle's approach is as much to make the task visible as it is truly a separate step on the scale of analysis and design. In fact, the output of *each* phase (especially the models) themselves constitute important documentation. An argument can be made, for example, for writing the design document in the style of a user guide: It is, after all, a description of what we expect the system to do in response to each thing a user might enter. To make this a separate effort, rather than assuring the community that it will happen may have the opposite effect, to the extent that it allows us to separate the effort from analysis, design, and construction.

For the Zachman Framework to not make it explicit is not a serious problem.

Transition

This is a more serious problem. In any situation where a new system represents a significant change to the infrastructure of an organization, transition can easily become the most expensive and complex phase of all. It has to be planned for, and time and money must be spent to make sure that the project is successful. Because it is clearly a step – an activity – instead of a "perspective," the Zachman Framework doesn't explicitly address it.

The Zachman Framework does, however, address many of the issues that come into play in transition. In row two, column four, it deals with the company's organization. Transition may involve changing that. Transition may also require education to change people's skill sets. The people to do this are the ones who have the row two perspective.

In row six, the "people" column, there is a view of the functioning system and its effect on people. The objective of a systems development effort is to ensure that that cell has people and systems that are compatible.

The conversion of data from legacy systems would be done in column one, row six, although the mapping of one to the other would be a function of row four.

Biography

A veteran of the Information Industry since the days of puched cards, paper tape, and teletype machines, Dave Hay has been producing data models to support strategic information planning and requirements planning for over thirteen years. He has worked in a variety of industries, including, among others, power generation, clinical pharmaceutical research, oil refining, forestry, and broadcast. He is President of Essential Strategies, Inc., a consulting firm dedicated to helping clients define corporate information architecture, identify requirements, and plan strategies for the implementation of new systems. He is the author of the book, *Data Model Patterns: Conventions of Thought*, published by Dorset House, and, more recently, producer of ***Data Model Patterns: Data Architecture in a Box***™, an Oracle Designer repository containing his model templates.

He may be reached at davehay@essentialstrategies.com, (713) 464-8316, or http://www.essentialstrategies.com.