The concept of enabling workers to from a workgroup is very simple. It begins with the ability to design work groups in manner that is fit with the system. What we see here in this report is the presentation of a module that will help us in designing an in house work group, aimed primarily at encouraging of in-house development. The Systems Development Life Cycle

It is important to understand the importance of development life cycle—identifying and selecting systems development projects which are uniquely common to each and every system. The issue is to take the position that organizations can benefit from following a formal process for identifying and selecting projects. To do this and in order to achieve this objective, the report first overviews the system development life cycle (SDLC) approach to the development of Information Systems and/or software is provided. An explanation of SDLC is offered, with different models applied in implementing SDLC delineated. Advantages and disadvantages associated with each of the models will be identified.
System Development Life Cycle

According to Walsham (1993), system development life cycle (SDLC) is an approach to developing an information system or software product that is characterized by a linear sequence of steps that progress from start to finish without revisiting any previous step. The SDLC model is one of the oldest systems development models and is still probably the most commonly used (Walsham, 1993). The SDLC model is basically a project management tool that is used to plan, execute, and control systems development projects (Whitten & Bentley, 1998). System development life cycles are usually discussed in terms of the conventional development using the waterfall model or the prototyping development spiral model. It is important to understand that these are just models; they do not represent the total system (Whitten & Bentley, 1998). Models reflect the structure of the organization, its management style, the relative importance it attaches to quality, timeliness, cost and benefit, its experience and its general ability levels, and many other factors. There should not be any standard life cycle because companies are unique (Whitten & Bentley, 1998).
As suggested by Whitten and Bentley (1998), most SDLC's have five major categories. These categories include:

1. Planning
2. Analysis
3. Design
4. Implementation
5. Support

Waterfall Model

The Waterfall Model represents a traditional type of SDLC. It builds upon the basic steps associated with SDLC and uses a “top-down” development cycle in completing the system. Walsham (1993) delineated the steps in the Waterfall Model, including the following:

1. An initial evaluation of the existing system is conducted and deficiencies are then identified. This can be done by interviewing users of the system and consulting with support personnel.
2. The new system requirements are then defined. In particular, the deficiencies in the existing system must be addressed with specific proposals for improvement.
3. The proposed system is designed. Plans are developed and delineated concerning the physical construction, hardware, operating systems, programming, communications, and security issues.
4. The new system is developed and the new components and programs are obtained and installed.
5. Users of the system are then trained in its use, and all aspects of performance are tested. If necessary, adjustments must be made at this stage.

6. The system is put into use. This can be done in various ways. The new system can be phased in, according to application or location, and the old system gradually replaced. In some cases, it may be more cost-effective to shut down the old system and implement the new system all at once.

7. Once the new system is up and running for awhile, it should be exhaustively evaluated. Maintenance must be kept up rigorously at all times.

8. Users of the system should be kept up-to-date concerning the latest modifications and procedures.

On the basis of the Waterfall Model, if system developers find problems associated with a step, an effort is made to go back to the previous step or the specific step in which the problem occurred, and “fix” the problem by completing the step once more. The Waterfall Model was given its name based on the visual appearance of the schedule associated with the model. Figure 1 provides a visual depiction of the model’s development schedule.

Figure 1: Waterfall Model
According to Walsham (1993), a number of advantages have been identified in relation to the Waterfall Model. One advantage is that the models rely on the creation of a System Specification Document (SSD) that allows for the cost and schedule of the system to be known once the SSD is created. As well, the model provides extensive documentation for companies that need such documentation and it has a long history of success in the computer industry.

Disadvantages identified by Walsham (1993) in relation to the Waterfall Model include that change to contract and costs must be renegotiated if such changes are made once construction has been initiated. As well, users must wait until the end of the project, or until at least a major portion of it is complete, before observing the results. Finally, the early phases of the project often take much longer due to the time necessary to generate the detail necessary in the SSD. According to Kay (2002), another major problem associated with the Waterfall Model is that it assumes that the only role for users is in specifying requirements, and that all requirements can be specified in advance. However,
as explained by Kay, requirements emerge and change throughout the process and during later maintenance phases, leading to the need for ongoing feedback and iterative consultation. Thus many other SDLC models have been developed.

*Survivable Systems Analysis Model*

With the growing recognition that information systems (IS) play a major role in insuring the success of virtually all organizations in business, government, and defense, awareness has also increased that such success is dependent on the availability and correct functioning of large-scale networked information systems of involving extensive complexity (Whitten & Bentley, 1998). Consequently, the SDLC model has become the context for further development of IS development requirements that focus on system survivability (i.e., end products that survive) (Carnegie Mellon Software Engineering Institute, 2002).

As delineated by the Carnegie Mellon Software Engineering Institute (CMSEI) (2002), survivability is the capability of a system to fulfill its mission in a timely manner, even in the presence of attacks or failures. As further clarified by the Institute, survivability moves beyond the realm of security and fault tolerance with a focus on delivery of essential services. On the basis of a survivability perspective, the delivery of essential services remains critical, even when systems are penetrated and/or experience failures and rapid recovery of full services when conditions improve (CMSEI, 2002). According to the Institute, survivability addresses highly distributed, unbounded network environments that lack central control and unified security policies.

According to CMSEI (2002), the focus of IS development when survivability is a critical component is on insuring three key capabilities of IS: resistance, recognition, and
recovery. Therefore, as outlined by CMSEI, IS a requirement for development and implementation are those which help to insure that the final product has the following capabilities in order to be successful:

1. Resistance: the capability of a system to repel attacks

2. Recognition: the capability of a system to detect attacks as they occur and to evaluate the extent of damage and compromise.

3. Recovery: the capability of the system to maintain essential services and assets during attack, limits the extent of damage, and restores full services following attack.

CMSEI (2002) has developed an ISD approach, the Survivable Systems Analysis (SSA) method (formerly the Survivable Network Analysis method) that focuses on the application of requirements in development and implementation to insure an end product capable of survivability. According to the Institute, SSA is a practical engineering process that permits systematic assessment of the survivability properties of proposed systems, existing systems, and modifications to existing systems. As delineated by the Institute, the SSA process is composed of four steps, as follows:

1. Step One: System Definition: developing an understanding of mission objectives, requirements for the current or new system, structure and properties of the system architecture, and risks in the operational environment.

2. Step Two: Essential Capability Definition: identification of (as based on mission objectives and failure consequences) essential services (i.e., services that must be maintained during attack) and essential assets (i.e., assets whose integrity, confidentiality, availability, and other properties must be maintained during attack) as
characterized by usage scenarios, which are traced through the architecture to identify essential components whose survivability must be ensured.

3. Step Three: Compromisable Capability Definition: selection of intrusion scenarios based on assessment of environmental risks and intruder capabilities and identification of corresponding compromisable components (components that could be penetrated and damaged by intrusion).

4. Step Four: Survivability Analysis: analysis of IS components and the supporting architecture for the key survivability properties of resistance, recognition, and recovery the production of a survivability map that enumerates, for every intrusion scenario and corresponding compromised component effects, the current and recommended IS architecture strategies for resistance, recognition, and recovery.

As suggested by Center for Technology in Government (CTG) (1998), while the SDLC has been used extensively, it has a number of associated problems. According to CTG, SLDC has been criticized due to rigid design and inflexible procedure. Consequently, as addressed by CTG, SDLC fails to account for the fact that real projects rarely follow the sequential flow that the model proposes. Because the SDLC model is a sequential process, any variations in its process create problems for the developer. As well, as identified by CTG, most ISD projects experience a great deal of uncertainty about requirements and goals in the beginning phase of the project, and it is therefore difficult for customers to identify these criteria on a detailed level. The SDLC model does not accommodate this natural uncertainty very well. The end result is that
implementation of the SDLC model can be a long, painstaking process that fails to provide a working version of the system until late in the process. Thus, such criticisms led to alternative SDLC processes that offered faster results, require less up-front information, and offer greater flexibility.

**Prototyping Model**

One such model is that which is known as the Prototyping model. According to the CTG (1998), the Prototyping model was developed as a means to compensate for some of the problems identified as associated with other SDLC model. The Prototyping model is based on the assumption that it is often difficult to know all of IS requirements during the beginning phases of a project. Through the application of the Prototyping model in ISD, the developer builds a framework of the proposed system and presents it to the customer for consideration as part of the development process. The customer in turn provides feedback to the developer, who goes back to refine the prototype to incorporate the additional information. This collaborative process continues until the prototype is developed into a system that can be implemented. As reported by the CTG, the Prototyping model is probably the most imitated ISD. Variations of the model include: Rapid Application Development (RAD) and Joint Application Development (JAD).

Figure 2 provides a visual depiction of the prototyping model.

**Figure 2: Prototyping Model**
According to the CTG (1998), overall criticisms of the Prototyping model have generally fallen into the following categories:

1. **False expectations**: Prototyping often creates a situation where the customer mistakenly believes that the system is “finished” when in fact it is not.

2. **Poorly designed systems**: Because the primary goal of Prototyping is rapid development, the design of the system can sometimes suffer because the system is built in a series of “layers” without a global consideration of the integration of all other components. Attempting to retroactively produce a solid system design can sometimes be problematic.

*The Exploratory Model*
The Exploratory Model represents an effort to move further away from an IS development framework in which requirements are not necessary in development and implementation activities. According to the CTG (1998), the exploratory model is most often used in fields such as astronomy or artificial intelligence because much of the research in these areas is based on guesswork, estimation, and hypothesis. The steps in the Exploratory Model involve initial specification development, system construction modification, and system test and system implementation when modifications and testing are indicative of a finished product. As explained by the CTG, criticisms of the Exploratory model include that it is very much limited to use very high-level languages, such as LISP; cost effectiveness is difficult to measure or predict; and the final product is often inefficient.

**Spiral Model**

As indicated by the CTG (1998), the Spiral model represents an incorporation of the best features of the SDLC and Prototyping models, while introducing a new component risk-assessment. The term “spiral” is used to describe the process that is followed as the development of the system takes place. Similar to the Prototyping model, an initial version of the system is developed, and then repetitively modified based on input received from customer evaluations. However, within the Spiral model, the development of each version of the system is carefully designed using the steps involved in the SDLC model. Each version of the development system is evaluated to assess associated risk and to determine if the ISD process should continue. The steps implemented within the Spiral model include planning; risk assessment; engineering; and, customer evaluation. Figure 3 provides a visual depiction of the spiral model.
According to the CTG (1998), few criticisms have as of yet been directed at the Spiral model due to its relative newness. There has been an indication that the risk assessment component of the Spiral model provides both developers and customers with a measuring tool that earlier model do not have.

**FAST Methodology**

FAST is a modern system development life cycle methodology. In the application of FAST methodology, activities are assigned to different personnel. While a single person may assume several roles, a single role may require several people to accomplish it (Whitten & Bentley, 1998).

As explained by Whitten and Bentley (1998), the FAST methodology consists of eight phases:
1. The Survey Phase establishes the project context, scope, budget, staffing, and schedule.

2. The Study Phase identifies and analyzes both the business and technical problem domains for specific problems, causes, and effects.

3. The Definition Phase identifies and analyzes business requirements that should apply to any possible technical solution to the problems.

4. The Targeting Phase identifies and analyzes candidate technical solutions that might solve the problem and fulfill the business requirements. The result is a feasible, target solution.

5. The Purchasing Phase identifies and analyzes hardware and software products that will be purchased as part of the target solution.

6. The Design Phase specifies the technical requirements for the target solution.

7. The Construction Phase builds and tests the actual solution.

8. The Delivery Phase puts the solution into daily production.

There are two ways that FAST projects are most frequently initiated. They are either planned or unplanned. The driving force for most projects is some combination of problems, opportunities, or directives from upper management (Whitten & Bentley, 1998).

**PIECES Framework**

According to Whitten and Bentley (1998), there are many potential problems or opportunities that could arise during the feasibility stage of a FAST project. As identified by Whitten and Bentley, PIECES is a useful framework for classifying problems,
opportunities, and directives. PIECES are also used to identify the urgency of the problem. It is identified as PIECES because each of the letters represents one of six categories:

P - the need to improve performance
I - the need to improve information (and data)
E - the need to improve economics, control costs, or increase profits
C - the need to improve control or security
E - the need to improve efficiency of people and processes
S - the need to improve service to customers, suppliers, partners, employees, etc.

(Whitten & Bentley, 1998)

Life Cycle Procedures

A typical approach to SDLC is to have many phases. These phases are established to provide an effective tool for controlling projects under development. The SDLC provides a straightforward review mechanism that enables the user to monitor and assess progress, performance, and budget status of the project. At times, it is necessary to reevaluate, reschedule, or terminate the development project. The SDLC approach also includes frequent reporting to top management. Problems that are found are discussed and resolved by the project team (Whitten, Bentley & Barlow, 1994).

As addressed by Whitten, Bentley and Barlow (1994), good documentation when using the SDLC approach should accomplish the following:
1. Furnish management with an understanding of system objectives, concepts, and outputs of the project.

2. Ensure correct and efficient processing within schema development.

3. Provide a convenient reference for systems analysts and programmers.

4. Provide material for training.

5. Serve as an aid for review of standards established in relation to the project.

Without a formal SDLC, as explained by Ahituv (1982), IS systems may be developed without well-defined requirements. This could also lead to documents being developed in-house when off-the-shelf solutions may be available. Also, when using a formal SDLC, the system is tested before production use whereas a failure to do so could result in inadequate project planning and tracking, inadequate systems documentation, no empowered group to represent users, and inadequate training plans (Ahituv, 1982).

*Optimum development method for the wireless industry*

Ideally, as developers, we would prefer to spend most of our time on new and exciting tasks. In this paper, The Information Technology Infrastructure Library (ITIL) is a framework that describes best practices for the wireless industry. These best practices do not prescribe the development process one must use, or the tools to use - but instead describe the tasks, procedures, ownership, and handshakes between process step owners. The application management process of ITIL defines the application management lifecycle as depicted in figure 4 below.

Figure 4: The Application Life Cycle
As developers, the process steps in the Application Development Phase triangle should be familiar. Depending on the development lifecycle process you have adopted, there are likely iterations within this phase.

The requirements process step is where the development team works closely with the business stakeholders to identify what the requirements and their relative priorities are.

The design process step is where the specifications are written to provide direction on how to implement the requirements.

The build process step includes the coding and various levels of testing the code to prepare the application for production.

The IT operations and support department typically handle the service management phase.

The deployment process step includes all the planning required to roll the application out to production.

The operate process step includes all the tasks to ensure that the application remains available for the end users.
The optimize process step is used to evaluate how the application is meeting the needs of the end users, and will identify new requirements that should be considered by development.

References


Gauteng Department of Education (GDE) and Gauteng Institute for Curriculum development (GICD) 1999: Technology Draft Progress Map. GICD, Norwood.


Shield, G. 1996. Learning technology through a process approach: The implementation of curriculum innovation through the pragmatic interventions of the teacher. International journal of Technology and Design Education.


Sage publications.