**Total Quality Management for Software Process Improvement**

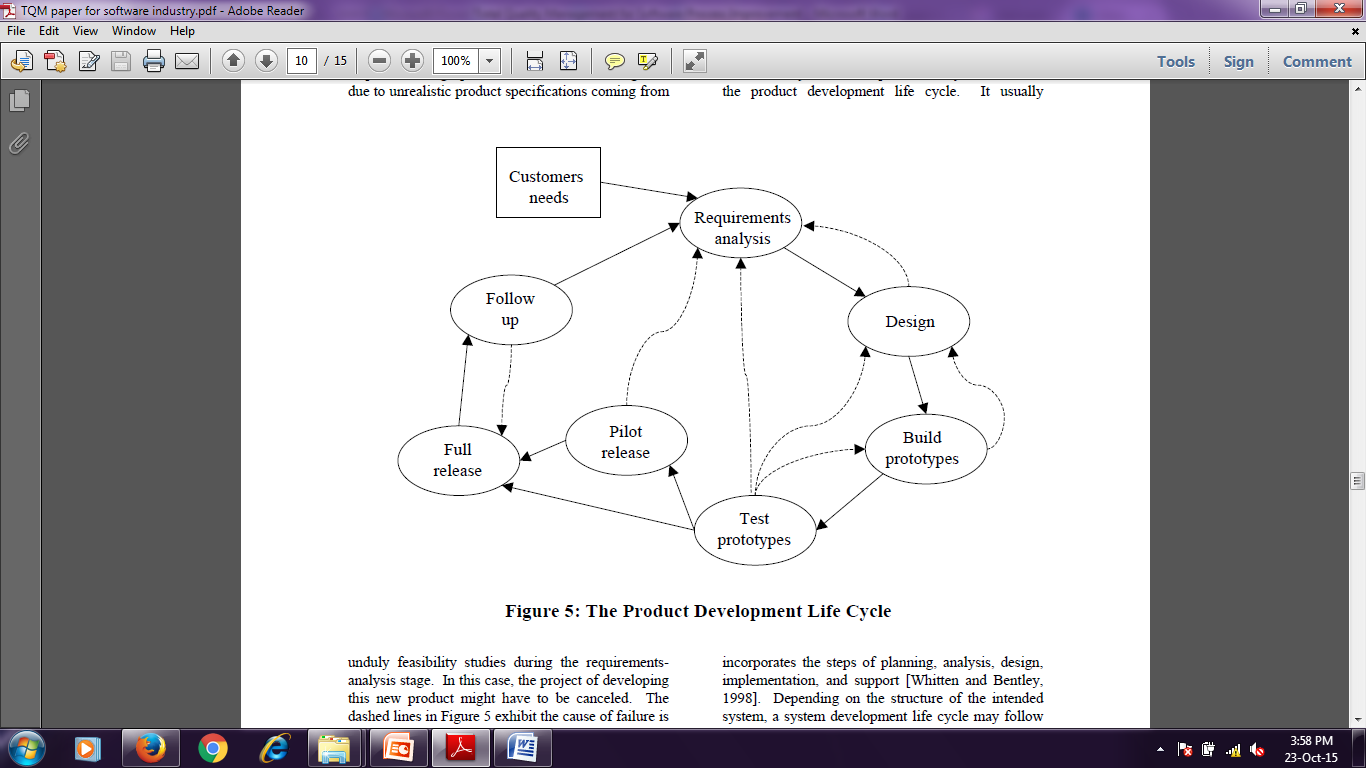
The TQM philosophy described above can be applied to any development process, be it product development or software development.

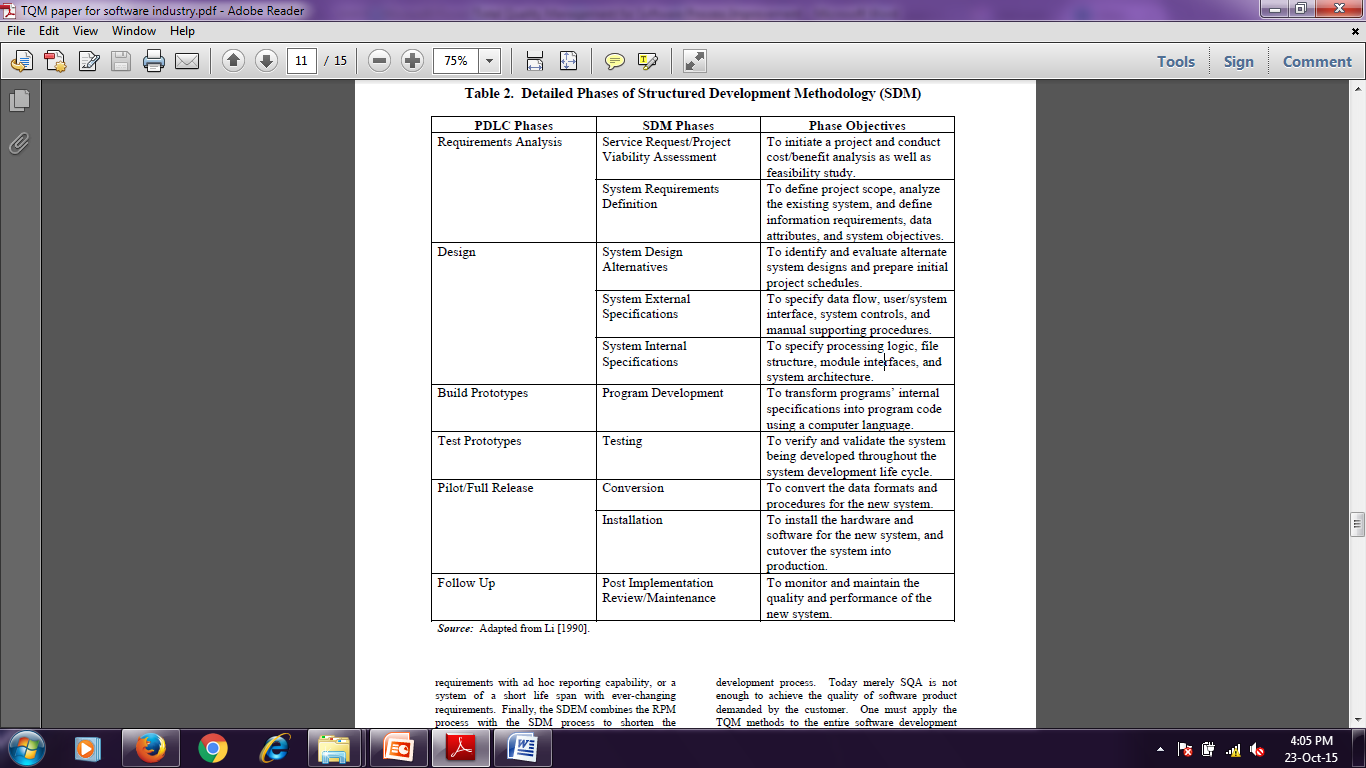
***The Product Development Life Cycle (PDLC)***

Product development life cycle is a systematic and orderly approach to managing product-development activities. It usually follows the problem-solving steps prescribed by Herbert A. Simon: intelligence, design, choice, and review. The development of a new product begins with the stage of requirements analysis. During this stage, the needs of customers are collected, analyzed, and evaluated in order to develop product specifications. Based on the customers’ needs and the product specifications, design blueprints of the product are developed during the design stage. These blueprints include manufacturing design specifications and bill of materials. According to these blueprints, prototypes of the product are built and tested to evaluate the quality of the prototypes. If a prototype fails the test, the cause of failure is analyzed and identified. It might be due to flaws in prototype building process or product design process. Worst of all, it might be due to unrealistic product specifications coming from unduly feasibility studies during the requirements analysis stage. In this case, the project of developing this new product might have to be canceled. The dashed lines in Figure 5 exhibit the cause of failure is sequentially fed back to the stage where the faulty process is. Once the prototypes passed all the tests, the best one is selected for either a pilot release (a limited-scale release to test the market) or a full release. Either release needs to develop a production line for the intended product. If the pilot release is not satisfactory, the sales information is fed back to the requirements-analysis stage and the product demand is re-evaluated. On the contrary, if the pilot release is successful, it is turned into full release and mass production of the product is performed. A stage of follow-up is preceded. If the follow-up report indicates successful sales and profit of the product, a go-ahead signal is sent back to the full release stage for continuing the mass production. If the report indicates otherwise, the requirements analysis process is triggered once again and the entire product development life cycle is repeated.

***The System Development Life Cycle (SDLC)***

A system development life cycle resembles the product development life cycle. It usually incorporates the steps of planning, analysis, design, implementation, and support. Depending on the structure of the intended system, a system development life cycle may follow a structured development methodology (SDM), a rapid prototyping methodology or a spiral development. The SDM typically is applied to a system with clear requirements definitions, well structured processing and reporting, and a long and stable life expectancy. Table shows a detailed breakdown of the SDM process. Under this methodology, iterations between phases in the process are strongly discouraged. It is therefore call a "water-fall" process. On the contrary, the Rapid prototyping method process allows and encourages such iterations. It uses very high-level development tools to quickly produce an operational prototype for users to gain hands-on experience. The prototype is then improved based on users' feedback. This process is suitable for a system of ambiguous or incomplete requirements with ad hoc reporting capability, or a system of a short life span with ever-changing requirements.





***The Deming Management Method***

*The Deming Philosophy:*

*Definition of quality, “A product or a service possesses quality if it helps somebody and enjoys a good and sustainable market.”*

*1) Create constancy of purpose for improvement of product and service.*

*2) Adopt the new philosophy of total quality.*

*3) Cease dependence on mass inspection to achieve quality.*

*4) End the practice of awarding business based on price tag alone.*

*5) Improve constantly and forever the system of production and service.*

*6) Institute training on the job.*

*7) Institute leadership.*

*8) Drive out fear of job insecurity.*

*9) Break down barriers between departments or staff areas.*

*10) Eliminate slogans, exhortations, and targets for the workforce.*

*11) Eliminate numerical quotas, goals, and work standards.*

*12) Remove barriers to pride of workmanship.*

*13) Institute a vigorous program of education and retraining for everyone.*

*14) Put everyone to work to accomplish the transformation.*

***Applying Deming's Fourteen Points to Software Development***

Deming’s fourteen points of management approach provide guidelines for implementing the TQM concept. These fourteen points can be applied to managing software development processes. The following discussion is based on the framework of the system development life cycle presented in the last section.

1) **Create constancy of purpose for improvement of product and service**.

Software development process traditionally ends when the completed system is handed over to the support group and put into production mode. Under the TQM culture, there is no finish line for the development team. Maybe there is a shift of focus from one project to another. The development team should be responsible for what they delivered, not the support group. Any quality problem occurs during the production should be addressed to the development team. Management must:

• Establish operational definitions for each step in the software development process.

• Define what is meant by “service to the customer.”

• Define standards of development, maintenance, and service for the next year and five years ahead.

• Define the internal and external customer.

• Develop ways to provide better systems and services in less time, using fewer resources.

• Invest in tools and techniques for better software development.

2) **Adopt the new philosophy of total quality**.

Quality is everyone’s business. Not just the worker, management is part of the quality team.

Under the TQM culture, quality comes first and everyone must join in. Corporate management, from top to bottom, must embrace the TQM concept and clearly communicate their support of this concept to all members in the software development team.

3) **Cease dependence on mass inspection to achieve quality**. Quality is built in, not added on. It is better to prevent errors in code, rather than reworking the code to remove the errors. Inspection or testing cannot prevent errors from happening, only experience and knowledge can. Management must install programs to continually improve software development processes. Examples of such programs are job training and job incentive programs.

4) **End the practice of awarding business based on price tag alone**. Many software organizations today are outsourcing their projects to subcontractors. It is important not to award a software contract based on price tag alone. Quality is more important than the difference in costs but for quantitative approach, cost should be considered. Low quality in the short run will result in high total cost. It is better to create long-term relationship with a few loyal and trustworthy suppliers who can produce quality code for your system.

5) **Improve constantly and forever the systems of production and service**. System development processes must be constantly improved by introducing new and working methodology, paradigm, standards, practices, techniques, tools, policies, and procedures. All these require the organization to constantly keep tracking the best practice in the field of management information system (MIS)—the so-called learning organization. Each individual staff member is required to improve oneself by updating or even expanding one’s skill set.

6) **Institute training on the job**. To build quality into the software, the development team must have appropriate experience and knowledge. On-the-job training program is an effective means of obtaining such experience and knowledge. In the broadest sense, all MIS staff members must know what their jobs entail and how to do their work. Management must assess the skill level of an employee before he or she is assigned to a software project. Different skill levels can play different roles and assume different responsibilities in a project.

7) **Institute leadership.** Management must lead, not punish. It is manager's job to help MIS staff do a better job and create a better system. Project managers must be trained in basic interpersonal and analytical skills. They must have a solid understanding of statistical process control. They should know that in any software development team whose performance is in statistical control, half of them would always be below average. They should focus on those members whose performance is out of statistical control.

8) **Drive out fear of job insecurity.** Employees must feel secure before they are willing to ask questions, make suggestions, or even expose their weaknesses by asking for help. The policy of long-term employment could easily drive out the fear of job insecurity. Moreover, any MIS staff whose performance is out of statistical control should be offered help in retraining or reassignment. However, if one consistently rejects helps from one's co-workers or supervisors, a layoff may be the last resort.

9) **Break down barriers between departments or staff areas.** Software development requires collaborative effort between users and IS staff. For as long as we can remember, communication gap has been the major factor to many MIS implementation failures. Furthermore, today's business system projects would most likely involve different functional

areas and require expertise in database processing, client-server computing, and network installation, etc. Therefore, open communication among functional areas and general knowledge across disciplines are necessary for a successful system implementation. This requires appropriate education and training for team members to change their behavior and improve their knowledge.

10) **Eliminate slogans, exhortations, and targets for the workforce.** Slogans do not build quality systems. MIS management should not ask for impossible target or schedule, or unrealistic level of productivity. Instead, they should post their progress in responding to suggestions and in helping the staff improve quality. Let the employees put up their own signs and slogans.

11) **Eliminate numerical quotas and work standards.** Quotas (such as metrics), goals (such as schedules), and work standards (such as unit times) address numbers, not quality. A software development project that causes haste and non-conformities accomplishes nothing and services no one. Let the project members put up their own goals. Managers should concentrate on helping people do a better job by reducing rework, errors, and waste. Everyone must work toward constant improvement, not the achievement of some arbitrary, short-term goals.

12) **Remove barriers to pride of workmanship.** All people are motivated. They would like to make quality products. However, a good workmanship relies on good materials, good tools, good methods, and good timing. Poor materials, broken tools, ineffective methods, or belated schedule are all barriers to pride of workmanship and should be eliminated. Let the software development team put its group identity or team members' names on the software product to take the credit (or the responsibility) of their work.

13) **Institute a vigorous program of education and retraining for everyone**. On-the-job training is effective, but slow, for an employee to acquire skill set for a particular type of job. In today's MIS arena, technology is changing so fast that new skill set is needed for the same type of job in a short period of time. Management must set aside enough budgets to

execute a generous education and retraining program for everyone to improve oneself. Under the TQM culture, all employees must know enough statistical method to understand the nature of variation, to manage the special causes of variation. Support for training employees to acquire necessary statistical method should be institutionalized.

14) **Put everyone to work to accomplish the transformation**. The TQM transformation is everyone's job. Everyone has a customer. Ask yourself who is the person receiving your work? All of us must identify our customers in order to determine precisely what our jobs are. Everyone belongs to a team, to work in the Plan-Do-Check-Act cycle, to address one or more specific issues, to find special causes detected by statistical signals. Moreover, we

must put management to work. Only management can change the culture and environment that dominate any individual's performance. Management must agree on their meaning and on the direction to take. They must acknowledge their mistakes, if any, and have the courage to change. They must explain to a critical mass of people in the organization

why change is necessary and that the change will involve everybody. Obviously, people must understand the Fourteen Points to know what to do and how to do it [Walton, 1986].