The “Doppler” Phenomenon
as observed in
IT Software Development Projects

by

Alta Terblanché
13 September 1999

Synopsis

With the advent of the millennium change information technology (IT) has become a focus point with the Y2K bug or problem. This problem in the IT world has been known since the early nineteen nineties. However, very little if any attention was given to it and the possible impact it could have on the business world and our day-to-day lives. The past eighteen months have seen a flurry of activities around Y2K being initiated.

A flurry of activities around the deadline or delivery date in IT software development is observed and this is compared to the “Doppler” phenomenon in as defined sound waves. The change of the millennium with the resultant activities taking place in the past six months has highlighted this phenomenon, and this is even more noticeable in the area of IT software development and/or changes.

Typical software projects are usually late, have cost overruns and are not delivered according to expectation. This is not an exhaustive study, but merely observations in the IT software development arena.

The Critical Chain method of Goldratt in conjunction with traditional project management is seen as a means in which this “Doppler” phenomenon can be circumvented. This is however, not a “silver bullet” and should be implemented with as much care and discipline as in any traditional project.
Introduction

The world as we know it today has become increasingly more complex and difficult to manage as to what it was fifty or even twenty years ago. In the global economy we attempt to manage this complexity and as we approach the change of the millennium project management as a discipline becomes an important management tool.

With the advent of the millennium change, information technology (IT) has become a focus point with the Y2K bug or problem. This problem in the IT world has been known since the early nineteen nineties. However, very little if any attention was given to it and the possible impact it could have on the business world and our day-to-day lives. The past eighteen months have seen a flurry of activities around Y2K being initiated.

This brings me to the objective of this paper, which I call the “Doppler” phenomenon of projects. I have observed in this phenomenon in various IT software development projects. The change of the millennium with the resultant activities taking place in the past six months has also highlighted this phenomenon.

I will attempt to briefly describe what is known as the Doppler effect as observed in sound and light waves, how it is observed in IT software development projects, why it takes place and how to try and prevent a project from becoming a black hole.

The Doppler Effect

The definition of the Doppler effect in sound waves is generally accepted as the following:

Whenever relative motion exists between a source of sound and a listener, the frequency of the sound as heard by the listener is different compared to the frequency when there is no relative motion.

A graphical description is:

![Doppler Effect Diagram]
This same phenomenon is used in radar. A sound wave is reflected from a moving target and a signal of known frequency can be used to determine the velocity of the target. This is however, not a lecture in the behaviour of sound or light waves.

I will attempt to define this phenomenon as I have observed it in IT software development projects. This definition is as follows:

Whenever relative motion exists between the kick-off date of a project and the deadline, the frequency of activities as observed is different compared to when there is no motion.

If we use the previous drawing we can draw the following comparisons:

- The deadline or delivery date to the stationary object;
- The moving object being the elapsed time from the start of the project until the deadline is reached; and
- The frequency the actual delivery of the tasks as defined in the work breakdown structure.

The “Doppler” phenomenon is usually noticeable in the sense of urgency that increases as the deadline approaches. What are also noticeable are the actual sound levels that increase, and where previously people sat relatively quietly behind their PCs or workstations with intense concentration, active discussion is now taking place.

This is no in depth study but a set of observations of IT software development during my career in the IT arena.

**Results of Most Software Development Projects**

Typical software projects are usually late because of the fact that the objectives of the project is not properly defined and would then result in the scope of the project being underestimated. This normally also manifests in a cost overrun.

I am not aware of studies that have been done on software development projects, as the resultant problems caused by weak definition are not documented. This documentation does not take place due to time and money constraints. However, one of the most famous cases documented was the development of the 360/OS for IBM and was documented in the “Mythical Man-Month”. This software development project was more than a year late on delivery. The late delivery of software development projects are often experienced in the IT world. One of the outcomes of late delivery that is experienced is that the software could be out of date, and is never implemented.

The question that is also asked in “The Mythical Man-Month” is “How does a project get to be a year late?” and the answer to this is given as “one day at a time”. In “The Mythical Man-Month” it is also documented that the effort was not wholly successful, which meant that the end product was not delivered to expectations. This is also typical of software development projects.
We have also observed this recently in the delivery of the Microsoft products. One example was that Windows ’98 was eventually delivered 12 months after schedule and with known bugs. This is exactly the same characteristic as observed in most other software development projects. The only reason these examples are cited is because most people are aware of them.

Since the 1960’s until now at the change of the millennium, various methodologies and plans have been developed at an attempt in getting software projects not to display the “Doppler” phenomenon with the resultant time and cost overruns. We have heard about the waterfall approach, RAD, JAD and any other acronym that we wish to cite to attempt the on-time and on-budget delivery of software development projects. We even have Kerzner that says, “project management is much more behavioural than quantitative”, and still the “silver bullet” eludes us.

I will endeavour to postulate not some new theory but to put together some of the points that have been highlighted through the years and some of my own experience to take a stab at how to make software development projects successful.

Defining the Scope of the IT Software Development

The definition of the scope of the IT software project always seems elementary at the scope planning stage. This, however is the beginning of the end! The business objectives that define the expected outcome of the developed software are very rarely stated explicitly. It is mostly hinted at, but seldom defined in clearly defined and measurable terms. The IT software is mostly seen as the “silver bullet” that will solve all the business problems, and is rarely seen as just a component of the total solution. This also seems like a motherhood statement and probably is.

The actual business purpose or benefit of the software is not readily acknowledged or recognised by the business. There could also be multiple outcomes expected from the implementation of the software. Each and every person that is involved in the development of the software also has a different objective of what the software has to achieve. All of these factors influence the definition of the software objectives, and the actual scope is “woolly”, which suits the business and the software developers.

The scope of the project depends from which point of view the software is observed. The financial person expects cost savings; the operations person expects business efficiencies, improved customer service and possibly savings in staff costs. The business implications and operational risks are not taken into account during the defining of the scope. This is seldom if ever articulated as such and no measurable are defined. The objective is for example set as developing an accounting system and not as being able to deliver the financial results at year-end in a shorter period of time.

The purpose of the project is seen as the “how” rather than “why” the software is being developed. The project is seen as an IT only responsibility and the issues of training, marketing, etc are rarely considered. Another example is the business who wishes to
launched a new product, the marketing strategy, plan, etc is complete before the IT department is informed of software changes. This brings about bad definition and bad software, which is then in continual state of change.

Let’s take the objective of the organisation of giving better customer service. To achieve this we are going to develop a customer information system. The definition of what the business sees as better customer service is not done, which ends in vague or ill-articulated software specifications. Software developers have recognised this and wait as long as possible to complete the final development, which also brings the flurry of activities at the time of delivery.

The software system is seen as the end, rather than the means to an end. This means that the focus of the software development is wrong and has poorly defined requirements. At this stage the software developers can follow whatever methodology or whichever cleverly devised plan, they will not be successful with the delivery of the software.

Business today is driven by the financial ratios and most activities have a cost focus, rather than a delivery focus. This implies that the focus in projects are on the cost factor rather, and as we know, time is money, which then focuses on time rather than business objective. This financial focus is short term and could have major cost implications further down the line. Senge in “The Fifth Discipline” says:

“we learn best from experience but we never directly experience the consequences of many of our most important decisions”.

The focus now is on the deadline, which brings about the flurry of activities and possible chaos at milestone or delivery time. This brings us back to the “Doppler” phenomenon that has very definite risks associated.

**Resultant Risks of the “Doppler” Phenomenon**

The obvious risk that stares us in the face is that of scope definition and planning. If the business objective is not understood and clearly defined, scope planning and definition with the necessary control, becomes the highest risk factor to the business and the project manager.

The IT department or the software development company becomes the scapegoat and the only responsible party to the failure of the successful delivery of the software development.

The risk tolerance of the development is now seen as only financial and not as the actual objective of possible customer loss, etc.

The fact that the objective of the project is ill defined brings about bad risk recognition and identification of risk sources. The actual risk of the project can also be ignored due to the objective not being defined.

As the deadline or implementation date comes closer the risks become more recognisable and the activities more frenzied. Once again the “Doppler” phenomenon.
Critical Chain & Theory of Constraints

If we study the work of Eli Goldratt with the theory of constraints and critical chain and the observations noted above we almost feel as if a “silver bullet” has appeared.

Goldratt in his book on Critical Chain has various mechanisms on how to avoid the “Doppler” phenomenon in projects. These mechanisms can be applied to IT software development projects. He says that certain behavioural factors influence the manner in which people act in a project:

- The first being the student syndrome, which says that there is no rush to start a task, so postpone it to the last minute.
- The second is multi-tasking, where people have to be seen to be busy all the time and to take on various tasks.
- The last is estimation of task duration to add in safety factors, which results in over estimation.

He also says that the objective of a project has to be clearly defined and understood by all people involved in the project. A measurable then has to be defined or identified to ensure that everybody understands the outcome of the project deliverable. Once a common goal or focus is established the project scope, plan and risk can be fully defined.

I agree with all that Goldratt says, but would like to add some further behavioural aspects to it:

- The business will always try to spend as little as possible on software development, as the results are not tangible.
- The business decision-makers will avoid the IT software development issues for as long as possible, because nobody likes to be associated with “failure”.
- The business decision-makers will avoid the software development project as nobody understands those “techies” and the “techies” really do not understand business, so we will humour them as long as possible.

The question I would like to venture to ask is “How do we get the decision-maker in business involved in the IT software project development?”

Conclusions

Various articles have been published in the PM Network (April & August 1999) and the Project Management Journal (June 1999) on the using of Critical Chain in the definition and management of projects. It is still in its infant shoes and could definitely be utilised in software development projects. However, it is not a “silver bullet” that will solve the problems that riddle IT software development projects with late delivery, cost overruns and under delivery on expectations.

Some IT companies in the USA have adopted the critical chain approach and feel that there is a definite benefit to the usage of the critical chain approach. We are in the process of starting to use it in a software development project and will hopefully be able to give some positive feedback at a later stage.
I do feel that the Critical Chain method of Goldratt could go a long way to help in software development projects, however the basis of the PMBOK should not be ignored but rather enhanced and supplemented with the critical chain approach.

As the millennium approaches a fresh look at software development projects is required and positive results should be published and given the necessary recognition. Until then we will continue to experience the “Doppler” phenomenon in IT software development projects.

References


I was born in Florida, Transvaal. I completed my secondary education at Vereening Hoërskool in 1968. I attended RAU from 1969 until 1972 where I was enrolled for a BSc with Maths and Math Stats as majors. I completed and MDP at Unisa and did an MComm in Information Systems at Wits in 1996 and 1997.

I am a member of the Computer Society of South Africa and the Project Management Institute. I also sat for my PMP in 1994 and achieved it.

I started my career in the IT industry as a programmer at Anglovaal in 1973, and followed the normal progression through analyst programmer, systems analyst and became the IT manager at Credit Guarantee in 1988, from there I moved to Standard Bank and is currently consulting as the executive manager for IT at the Landbank.