The Cataract Methodology for Systems and Software Acquisition[1]

Joseph Kasser DSc CEng CM MIEE
Systems Engineering and Evaluation Centre
University of South Australia
School of Electrical and Information Engineering F2-37
Mawson Lakes Campus
South Australia 5095
Telephone: +61 (08) 830 23941, Fax: +61 (08) 830 24723
Email: joseph.kasser@unisa.edu.au

Abstract. This paper documents the Cataract methodology, which is based on the recognition that the "Build" approach used in the operations and maintenance phase of the system and software development for software maintenance is also applicable to the initial development phase. The Cataract Methodology has been constructed out of components in existing methodologies, each of which have been shown to be effective. The Cataract methodology extends the spiral approach by emphasizing the criticality of configuration management and the type of information needed to control system and software development in an integrated engineering and management environment. The Cataract methodology with its focus on configuration and knowledge management can produce systems that converge with the needs of the customer with fewer cost and schedule escalations and project failures provided appropriate knowledge management and configuration tools are used.

INTRODUCTION

The current systems and software acquisition paradigm is characterized by project failures and cost and schedule overruns. Data from the USA (Chaos 1995) and UK (OASIG 1996) show that the problem is an international one. Conventional wisdom states that the Waterfall approach does not cope well with changing requirements. Thus efforts to overcome the problem have reacted to the effects of poorly articulated and changing user requirements during the development process and have focussed on changing the production process from the waterfall approach to some type of rapid, spiral, or other methodology, but without much of an improvement. Now, from an information systems and Knowledge Management perspective these acquisition programs do not fail because the requirements change, they fail because of poor requirements management, namely the failure to manage the changing requirements. This paper analyses the system and software methodology, provides some insight into the nature of the process generally thought of as represented by Figure 1. The customer has a need that is documented in a statement of work and a contract is awarded for development of a product or system that meets the need. The contractor then develops the product over some period of time. There are a number of milestone reviews along the production process to attempt to verify that the development contractor is producing the correct system.

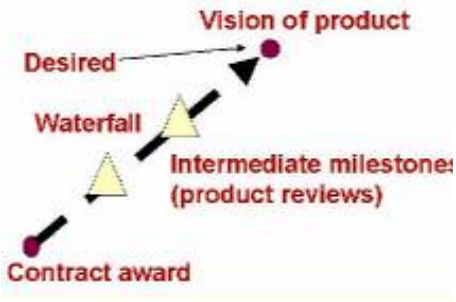

Figure 1 The ideal process

The Waterfall methodology shown in Figure 2 was the first attempt to document the production process. It showed the process as a serial sequence of events.

The requirements analysis phase is the phase in which the user needs and constraints are examined. This is then followed by the production of the initial set of user requirements or needs. The user's

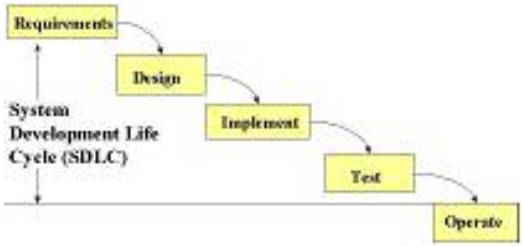

Figure 2 The Waterfall Methodology

[1] This work was partially funded by the DSTO Centre of Expertise Contract

Figure 3 Process for accepting requirements

needs are then translated into system requirements, which these days are stored in the database of a Requirements Management tool. The process of accepting the initial set of requirements may be represented as shown in Figure 3. Each requirement must be considered as a request until accepted and is allocated an identification number. The requirement must be assessed for priority, and cost and schedule impact, as well as for risks. However, during the pre System Requirements Review (SRR) period, some of these assessments are currently generally not performed. The requirements must be considered as not being firm until all the initial system requirements have been documented. During this process the customer must resolve conflicts in the requirements. At that time, the process of gathering the initial set of requirements generally terminates with the SRR in which both customer and development contractor accept the requirements and the requirements are frozen (no further changes allowed). The customer agrees that the requirements represent the needs, and the development contractor agrees to produce a system that meets the requirements. The next phase in the waterfall methodology is the design phase, which follows once the requirements have been accepted. It is the phase in which modules of the system are designed to meet the requirements. The implementation phase in which the system is constructed then follows. Once the system is constructed it is formally tested and finally delivered to the customer for use.

Milestone reviews take place between the phases to confirm that the work allocated to a specific phase is complete and the process is ready to advance to the next phase. The name of the methodology was adopted because the pictorial representation shows each phase seeming to flow naturally into the next phase like water flowing over a series of falls.

The Waterfall process is ideal when the vision of the product exists (all requirements are known) at the time that the contract is awarded, and the contractor just builds it. However, in the real world the situation is different as shown in Figure 4. During the time that the contractor advances towards the vision of the product that existed at the time the contract was awarded, the vision itself changes. In

Figure 4 The process in the real world of changing requirements

other words, the target is moving. Thus, while the delivered system may meet its original requirements, the system will not meet the "new" requirements in effect at the time of delivery. The target moves for various reasons including

- The customer requirements change over time for different reasons.
- The customer has non-articulated requirements at the start of the process and manages to articulate them as time passes.
- Externally driven changes such as changes in government regulations, changes in the marketplace, and changes in other systems that interface with the system at any level within the meta-system, system, and subsystem hierarchy.

This situation leads to poorly controlled construction as represented by the chaotic waterfall model shown in Figure 5. If the Spiral model (Boehm 1988) is opened up, it can be seen to be a series of waterfalls. While the spiral approach emphasizes risk management, and facilitates the articulation of requirements, it does not emphasize configuration control. Thus while the Spiral model provides some improvement, the lack of configuration control tends to result in moving baselines and confusion, which leads to cost escalation and schedule delays.

The real world of continuously changing requirements is recognized by Kasser (2001) who writes, the goal of system engineering is to provide a system that

- Meets the customer's requirements as stated when the project starts.
- Meets the customer's requirements, as they

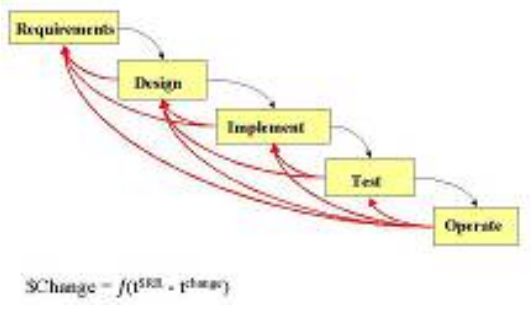

Figure 5 The Chaotic view of the waterfall

exist when the project is delivered.
- Is flexible enough to allow cost effective modifications to be implemented as the customer's requirements continue to evolve during the operations and maintenance phase of the system life cycle.

Thus the goal of the SDLC must then be to manage change in a manner that achieves convergence between the needs of the user and the capability of the as-built system in a cost-effective manner as shown in Figure 6.

Figure 6 The road to convergence

The way to achieve this goal seems to be not to attempt identify all the requirements at the start of the project, but to only identify the highest priority requirements and the real requirements. Then to achieve convergence by fleshing out the requirements in a controlled manner and delaying design decisions using a just-in-time approach (Kasser 2000a) in the cataract implementation of the budget tolerant methodology (Denzler and Kasser 1995). The Cataract methodology relies on two factors
- The waterfall methodology works very well over a short period of time as shown by the Spiral model.
- Implementation and delivery of systems and software are often performed in partial deliveries, commonly called "Builds" in which each successive Build provides additional capabilities.

Build planning is not a new concept. It has been used in software maintenance for many years. The insight presented herein is that as soon as the first Build in the development process has begun, the development process and the maintenance process are identical. In the software world, a Build means a defined software component. Successive Builds enhance the capability of the software. In the hardware world, Builds can comprise subsystems, or the integration of two or more subsystems.

The cataract approach to Build Planning may be likened to a Rapid Prototyping scenario within the spiral in which the requirements for each Build are frozen at the start of the Build. Once the initial set of requirements has been signed off, the system architecture designed, and the implementation allocated into a series of Builds, the implementation phase embodying the cataracts begins and the activities in the development organization can be shown in the traditional GANTT chart format depicted in Figure 7.

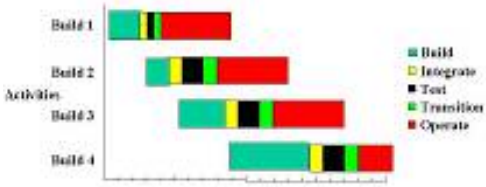

Figure 7 The traditional GANTT Chart format

The work associated with each Build takes place in the three parallel streams of activities (management, development and test/Quality), which include
- **Systems engineering** performs requirements and interface engineering, change assessment, risk management allocates the system level requirements between the hardware and software components, coordinates technical performance analysis and measurement, and produces the process-products (documentation).
- **Software engineering** turns the software requirements into code, and evaluates and perhaps incorporates commercial-off-the-shelf (COTS) software.
- **Hardware engineering** may be working with the computers, workstations, disk drives or other storage elements, networks, and custom hardware elements.
- **Test and evaluation** develops test plans and procedures, then performs the tests and reports on the results.
- **System Integration** integrates the hardware and software units and verifies their working together.
- **Final testing** in which the integrated Build is tested prior to acceptance by the customer.
- **Transition** is the time in which the Build is turned over to the customer or user.
- **Operations and maintenance** is the time span when the Build is operated by the customer, or by the maintenance contractor.
- **Management** is the planning, organizing, directing, and controlling the technical and administrative work. This includes making sure that the needed resources are available at the appropriate time.

BUILD ZERO AND SUBSEQUENT BUILDS

The Cataract methodology incorporates an initial Build, Build Zero, which contains the same initial two phases, requirements and design, of the Water-

fall methodology with the exception that there is recognition that

- All the requirements are not finalized at SRR.
- Additional requirements will become known as the project progresses.
- Design and implementation decisions will be deferred and made in a just in time manner (Davies 1998; Kasser 2000a). These decisions must also be made so as to maximize the "don't care" situations (Kasser 2001).

The work in Build Zero is to

- Identify the highest priority requirements.
- Baseline an initial set of user needs and corresponding system requirements.
- Develop the Framework for Requirements Engineering in a Digital Integrated Environment (FREDIE) incorporating the Quality System Elements (QSE) for each of the baselined requirements (Kasser 2000). The FREDIE provides the data necessary for making informed decisions about accepting the initial set of requirements and subsequent changes.
- Complete the first draft of the Systems Engineering Management Plan (SEMP) and Operations Concept Documents (OCD) (Kasser and Schermerhorn 1994).
- Design the Architecture Framework for the system in accordance with the Defence Evaluation and Research Agency (DERA) Reference Model (DERA 1997).
- Perform risk assessment to determine the proposed Architecture Framework can meet all of the highest priority requirements.
- Document the assumptions driving the Architecture Framework and a representation of operational scenarios (Use Cases) that the Architecture Framework prohibits. This activity also helps identify missing and non-articulated requirements early in the SDLC. The design of the Architecture Framework for the entire system in Build Zero introduces a risk that it may not be suitable for changes years later in its operations and maintenance phase (or even earlier). This is why part of the Build Zero effort is to determine scenarios for which the system is not suitable. The customer is then aware of the situation. The goal of the Cataract methodology is to achieve convergence between the customer's needs and the operational system. In the course of time, one can expect that the need will change to something for which the system cannot provide capability. At that time, a revolutionary Build will be needed to replace the system. However, it will be done with full knowledge in a planned manner, rather than the ad-hoc manner of today's environment.
- Develop the work breakdown structure (WBS) to level the workload across the future Builds and implement the highest priority requirements in the earlier Builds (Denzler and Kasser 1995).

From Build One inclusive, each subsequent Build is a waterfall in itself. The requirements for the Build are first frozen at the Build SRR. Then the design effort begins. Once the design is over, the Build is implemented and the system turned over for integration. While the design team does assist with the integration, their main effort is to start to work on the design of the next Build. Once the first Build has been built and is working, the requirements freeze, design - integrate - test - transition and operate stages of the system life cycle (SLC) commences for the second Build. This cycle will continue through subsequent Builds until the system is decommissioned although the contract may change from the development organization to the maintenance organization. Each Build is an identical process but time delayed with respect to the previous one. Each successive Build provides additional capabilities. When the Builds are placed

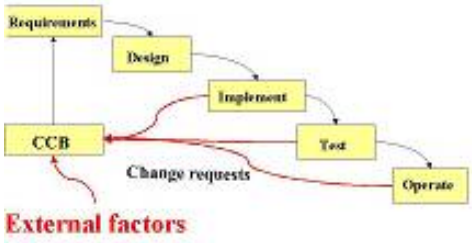

Figure 8 Configuration Control View of Waterfall

under configuration control, the Waterfall may initially be drawn as shown in Figure 8 however this figure is misleading. Externally driven changes are requested and problems tend to show up during the integration and test phases. When a problem is noticed, a discrepancy report (DR) is issued against the symptom. This DR is analysed and the cause identified. A change request is then issued by the Configuration Control Board (CCB) to resolve the defect either in the current Build before delivery, or by assigning it to be fixed in a subsequent Build. Thus Figure 8 should be replaced with Figure 9 showing that the Cataract methodology explicitly adds the management of changing requirements to the DERA Evolutionary Lifecycle approach (DERA 1997 Figure 24). The feedback and externally driven change request, if accepted, may be implemented in the appropriate future Build. Think of each Build as being completed a little behind the arrowhead of the advancing requirements. From this perspective, the gap between the user's need

and the completed section of the system converges over time.

Project personnel move from one Build to the

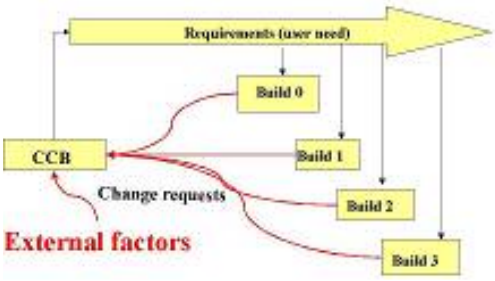

Figure 9 The Cataract Methodology

next; the development team moves from one Build to the next, as does the testing team. Ideally the Builds are sequential with no wasted time between them. The customers tend to get increasingly involved with the system during later Builds by virtue of being able to use early Builds.

Each Build is placed under configuration control and may be delivered to the customer. Accepted change requests modify the requirements for future Builds, with the sole exception of "stop work" orders for Builds-in-progress if the change is to remove major (expensive to implement) requirements being implemented in a Build-in-progress. The milestone reviews within a Build are identical to those in the Waterfall methodology, since the Build is implemented within the Waterfall. All change requests received during any Build are processed and if accepted are allocated to subsequent Builds. Freezing of the requirements for each Build at the Build SRR means that when the Build is delivered it is a representation of the customer's needs at the time of the Build SRR. It may not meet the needs of the customer at the time of delivery, but the gap should be small depending on the time taken to implement the Build. Thus achieving convergence between the needs of the customer and the capability of the as-delivered system.

CHANGES

Donaldson and Siegel (1997) state that there are two types of changes during the SDLC namely planned and unplanned.

Change requests. The process for dealing with both types of change is the same. The change requests are processed via the configuration control board (CCB) in the same way that they process DRs. Requests for planned changes tend to be processed well before the change is to be implemented. Requests for unplanned changes however, need to be categorized by priority. Typical categories may be "routine", "urgent", or "do by yesterday" or their equivalents. A typical "do by yesterday" change request is the result of an analysis of a DR reporting that the system crashes. The process for handling a change request is shown in Figure 10. Some internal or external source generates a change request, which is logged and assigned an identification number. The impact of the requested change on the product and process (Builds) is then assessed and a decision made as to whether to accept or reject the request. The source is then notified of the decision, if the change request is accepted, then

Figure 10 The Change request process

- From the product perspective, the requirements are changed to reflect the new situation. This is done by adding, deleting or modifying (a combination of adding and deleting) requirements.
- From the process perspective, the Build Plan is changed to show when and where the change will be implemented.
- The SEMP and OCD are modified as appropriate.

The change request process shown in Figure 10 is the same as the process for accepting the initial set of requirements at the start of the SLC shown in Figure 3. Thus the only difference between a requirement at the start-up phase and a change sometime later in the entire SLC is that a start-up is a transition from no system to some system, while a change is a transition from some system configuration to a different system configuration. This perspective complies with Hitchins' (1998 p72) requirement that a system-to-be-created must be viewed during conception and design as though it already existed and was operating. By viewing the SLC from an information system perspective, the initial SDLC can be seen to repeat repetitively during the development contractor's Builds as well as during the operations and maintenance phase of the SLC.

The key to effective control of the process is effective configuration control and informed decisions about the impact of any change request on the product (capability) and process (cost and schedule) which requires knowledge management. The poor management of the multi-phased, time-ordered, parallel activities, and the lack of informa-

tion precluding informed decisions about the impact of the decisions are major contributors towards the current cost and schedule escalations and project failures.

THE CATARACT PERSPECTIVE

From the cataract perspective

- Successive Builds do not have to be incremental or evolutionary, they can also be revolutionary, i.e. an entire replacement system can be factored into the schedule. Thus legacy systems can be upgraded and replaced with minimal waste of resources using the Cataract methodology. By knowing when parts of the system will be replaced (in which Builds), informed decisions can be made as to which defects to fix, and which modifications to make, to the current system. As well as which to defer to the replacement system.
- The Year 2000 issue was just a DR and changes made as a result of the analysis of the problem.
- Effective configuration control and information about the state of the project is vital.
- The Cataract methodology depends on a new generation of tools and information displays such as the QSE, FREDIE, and Categorized Requirements in Process (CRIP) charts (Kasser 1997).
- The Cataract methodology is an integrated product-process (engineering and management) methodology that can be used to control costs and schedules and minimize project failures.

SUMMARY

By viewing the SDLC from the perspective of Builds it can be seen that

- The SDLC is a time-ordered task. In addition, since the development contractor may be working on more than one Build at a time, each Build being in a different part of its SDLC, the total SDLC is also a parallel process with phase-delayed elements.
- Except for Build Zero, the work performed in the SDLC, namely up to the time the development contractor turns the system over to the customer (and the maintenance contractor) is identical to the work performed during the operations and maintenance phases of the SLC.

CONCLUSION

Both the SDLC and the SLC are multi-phased, time-ordered, parallel-processing tasks. The Cataract methodology with its focus on configuration and knowledge management can produce systems that converge with the needs of the customer with fewer cost and schedule escalations and project failures provided appropriate knowledge management and configuration tools are used.

REFERENCES

Boehm B., "A Spiral Model of Software Development and Enhancement," IEEE Computer, May 1988.

CHAOS, The Standish Group, 1995, http://www.pm2go.com/sample_research/chaos_1994_4.asp last accessed January 16, 2002.

Davies J., "Making Choices at the Right Time: Producing Better Systems", *Fourth Annual Symposium of the INCOSE-UK*, 1998.

Denzler D., Kasser J.E., "Designing Budget Tolerant Systems", *5th Annual International Symposium of The National Council of Systems Engineering (NCOSE),* 1995.

DERA Systems Engineering Practices Reference Model, Issue 1.0 May 1997, available via http://www.incose.org/stc/SEGD12_2.htm, last accessed January 18, 2002.

Donaldson S.E., Siegel S.G. *Cultivating Successful Software Development*, Prentice Hall PRT, 1997.

Hitchins D.K., "Systems Engineering…In Search of the Elusive Optimum", *Fourth Annual Symposium of the INCOSE-UK*, 1998.

Kasser J.E., "Enhancing Conferences and Symposia using Web Based Asynchronous Techniques", *11th International Symposium of the INCOSE*, Melbourne, Australia, 2001.

Kasser J.E., "Writing Requirements for Flexible Systems", *INCOSE-UK 5th Annual Symposium*, 2001

Kasser J.E. "A Framework for Requirements Engineering in a Digital Integrated Environment", *the Systems Engineering Test and Evaluation Conference (SETE), 2000.*

Kasser, J.E., "The WebConference: A Case Study", The INCOSE - Mid-Atlantic Regional Conference, Reston, VA, 2000a.

Kasser J.E., "What Do You Mean, You Can't Tell Me How Much of My Project Has Been Completed?", *INCOSE 7th International Symposium*, Los Angeles, CA, 1997.

OASIG, The performance of information technology and the role of human and organizational factors. Report to the Economic and Social Research Council, UK, 1996, available at http://www.shef.ac.uk/~iwp/publications/reports/itperf.html, last accessed January 16, 2002.

AUTHOR

Joseph Kasser has been a practising systems engineer for over 30 years. As well as being a Certified Manager, he has a doctorate in Engineering Management (Systems Engineering). He is the author of "*Applying Total Quality Management to Systems Engineering*", Artech House, 1995, and many conference papers. Dr.

Kasser performs research into improving the acquisition process. He is a recipient of NASA's Manned Space Flight Awareness Award for quality and technical excellence (Silver Snoopy), for performing and directing systems engineering and has many other awards (certificates and plaques) as well as letters of commendation from previous employers and satisfied customers. Dr. Kasser also teaches systems and software engineering subjects both in the classroom and Internationally via distance education.