Document Inspection: An Agent Of Change

Dick Holland
Primark Investment Management Services Limited
27 September 1996

A Vision of the Future

This is the story of a journey; or, rather, the beginning of a journey. It's a journey that we have embarked upon whilst its destination is but a vision - a vision of the future in which our product is recognised clearly as the best of its kind on every measurable scale.

We have already caught a glimpse of this future; every month a little more of the vision becomes real to us because our journey is one of *continuous improvement*. It is also one of constant discovery, of new insights and continuous learning from our own experiences. We have started by focusing on the quality of our software but ultimately our travels will take us into every corner of our business.

Part of this paper describes the lessons we learned from Tom Gilb at the end of 1994; it also describes how we've put that learning into practice, and how we've developed and discovered new insights beyond it. The essence of the Gilb "method" is that quality measurement and improvement are embedded in the production processes (those that actually *make* your product). "Quality" thus requires no external agencies because it's *in-process*. You define the quality goals for your system, product or service by means of *Quantified Objectives*, stating their current and target values and how they are to be measured, and the method provides a number of strategies by which those goals might be reached.

The method is firmly rooted in a strong process orientation: it requires that the production processes be identified, and *process ownership* be recognised and assigned. Thereafter product and process quality, as measured by the values held by the quantified objectives, is raised continuously by means of techniques which include Defect Detection by *Document Inspection* and Defect Prevention by *Continuous Process Improvement*.

It follows, therefore, that the first steps to take must include setting the quantified objectives and identifying the processes. This may (and we found that it did) require the redesign of existing processes and the recognition of hitherto undiscovered ones. It means that the supporting information systems needed to measure, record, track and report improvements in the objectives require redesign (as we also found), and it may also require organisational change (as we are now finding).

Petrozzo and Stepper in ***Successful Reengineering*** [Petrozzo94][1] define Business Process Reengineering as "the concurrent redesign of processes, organisations, and their supporting information systems to achieve radical improvement in time, cost, quality, and customers' regard for the company's products and services".

That's a very good definition of what we're now doing; the difference in our case perhaps being that it is not a one-off exercise, for we are continuing to develop and improve the infrastructure, and the tools and techniques and, above all, to *change our culture* so that we are ready and able to redesign our processes, organisation and systems continuously.

It is in this way that we are using the powerful leverage effect of Document Inspection as an *agent of change* and we now know that we can realise our vision and reach our goal of owning a product that is recognised as the best of its kind on every measurable scale.

[1] A list of references can be found on page 35.

Measures of Software Quality

There have been, and continue to be, many attempts to define software quality. Some have concentrated on measuring the delivered software itself, while others, such as ISO9000/BS5750, have focused exclusively on the development process presumably on the premise that if the process is of good quality so will be the software it produces.

Perhaps the most interesting example of the latter is the ***Capability Maturity Model*** (CMM) developed by the Software Engineering Institute of Carnegie Mellon University in the USA [Paulk93]. This has come about from the pioneering work undertaken by, among others, Watts Humphrey while at IBM. It defines a set of criteria against which an organisation can be measured to determine the quality of its development process and the software produced thereby. A software development organisation is measured by the CMM according to the characteristics exhibited by its processes.

Figure 1 illustrates the five levels of capability defined by the CMM and the criteria that apply to each.

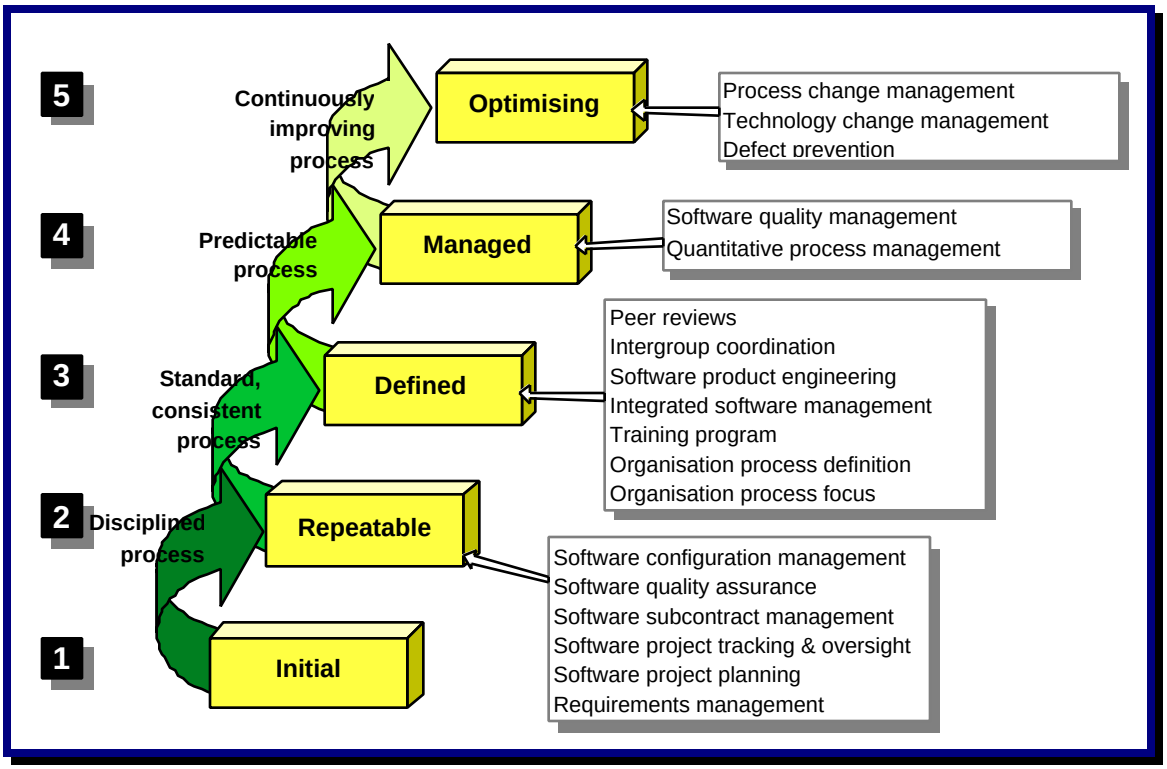

Figure 1: SEI Capability Maturity Model

Although the CMM defines "software quality management" as a characteristic of a Level 4 organisation, it does not prescribe what qualities are to be measured and managed. It does say, however, that Levels 4 and 5 characteristics are based on the concepts of statistical process control as described by J.M. Juran in ***Juran on Planning for Quality*** [Juran88].

The *Juran Trilogy Diagram*, illustrated in Figure 2, depicts quality management as three basic processes: quality planning, quality control and quality improvement.

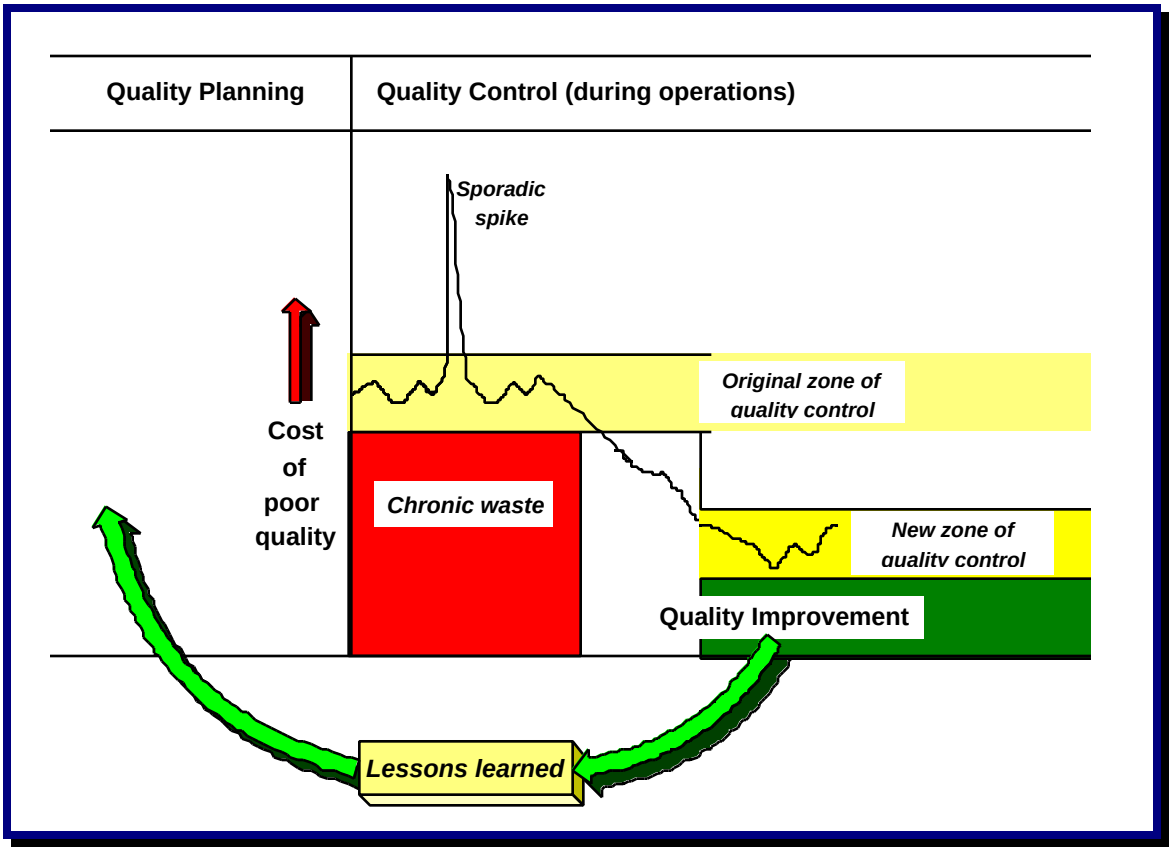

Figure 2: Juran trilogy diagram

The *sporadic spike* in Figure 2 represents what Juran calls *fire fighting activities* - a term with which all those concerned with software development are familiar. The Trilogy Diagram shows how the chronic waste caused by not managing quality can be reduced by introducing quality improvement and feeding back the lessons learned thereby into the quality planning process.

Michael Fagan, also at IBM, developed in the early 1970s a method of *formal inspection*, against documented criteria, of program code as a means of improving software quality by removal of defects before test [Fagan76]. This work was extended by Tom Gilb, as described in ***Software Inspection*** [Gilb93], into a method for formally inspecting any type of document. Gilb placed additional emphasis upon inspecting documents upstream of code in the development process on the basis that the earlier any defects are found the cheaper they are to fix.

What Gilb also did beyond Fagan was to build into his method a mechanism by which the process that created a document could also be examined to determine whether any systemic defects were being caused by the process itself. This introduced the element of *process improvement* into the method and resulted in a single technique that could be used for defect detection (a CMM Level 3 criterion) and defect prevention by process improvement (a CMM Level 5 criterion).

Gilb had also been involved for many years in the field of software metrics and it is from this work that the notion of *Quantified System Attributes* came, described in his book ***Principles of Software Engineering Management*** [Gilb88]. These provide a means of defining attributes or objectives for a system which are measurable and hence *quantifiable*.

Attributes can be chosen that refer to any aspect of a software system that will, if improved, contribute to an improvement in the overall "quality" of the software as perceived by its users. Thus they provide a very direct way of measuring improvement in software quality.

What do we want to measure?

Quality can be measured along many dimensions. Some of the more interesting questions we might want to ask about a software system include: how many defects does it have? how easy is it to use? how easy is it to learn? how easy is it to install and upgrade? how well does it perform?

These qualities or attributes are important to a greater or lesser degree to all users of the software and it is therefore worthwhile to measure them to determine which ones require improving and to what extent.

We can define as an attribute anything that can be measured numerically; ***Principles of Software Engineering Management*** [Gilb88] provides a "starter set" of such attributes.

What about productivity?

We've so far described the measures that we want to apply to the delivered software system, but what about development productivity?

We were sufficiently concerned about what we perceived to be a productivity problem that we presented Tom Gilb in November 1994 with our main objective (which became known as our "time to market" objective):

> "To improve time to market for new Icon developments and enhancements"

His response to this was a high-level definition of the quality of the software and our success at developing, installing, maintaining and supporting it:

> "To reduce the calendar time needed to deliver what customers really want. This is defined as:
>
> Productivity
>
> Our ability to deliver customer needs in relation to cost of delivery including training, installation, help service and maintenance
>
> Reliability
>
> The degree to which the delivered functionality and quality of our total product and service meet the defined and real expectations and needs of our customers and prospects"

Clearly by this definition, quality and productivity are two halves of an indivisible whole, because the "time to market" objective includes delivering what the customer wants when he wants it. It also embraces all the necessary support activities and therefore addresses in a single definition many aspects of our business.

Four main strategies were proposed to help us meet this objective:

- Defining *Quantified System Attributes* to set the quality standards for the software
- *Document Inspection* to detect and remove defects
- *Continuous Process Improvement* to prevent defects appearing by changing the development processes
- *Evolutionary Result Delivery* to deliver incrementally the required functionality in a timely and controlled fashion

We're now going to examine each of the first three of these strategies.

Quantified System Attributes

As we have noted, Quantified System Attributes set the standards by which we wish to measure the software system, and an attribute can be anything that is directly measurable by some means. We look at attributes in two basic ways: those that are directly visible to the user of the delivered software system; and those that are not. This gives us a first-level classification of attributes into externally- and internally-focused ones, and we place each attribute that we want to define into one of these two classes.

A list of interesting externally-focused attributes, those that users of the system see, could include the following:

- How frequently does it fail to produce correct results? (which we shall call *Reliability*)
- How easy is it to use? *(Facility)*
- How easy is it to install and upgrade? *(Installability)*
- Does it do everything that I want? *(Functionality)*
- Does it do it quickly enough? *(Performance)*

We classify these as *External Attributes* of our system: those which directly affect the customers of the delivered software.

A list of internally-focused attributes, those which the developers, maintainers and supporters see, could include:

- How easy is it to fix things when they break? *(Serviceability)*
- How easy is it to test and prove that a change works? *(Testability)*
- How easy is it to diagnose errors and find their causes? *(Diagnosability)*

We classify these as *Internal Attributes* which are those that directly affect the vendors of the product and indirectly affect its customers. It is these which are also related to the productivity aspect of our "time to market" objective.

We believe that it is imperative to publish the attributes, both within and without our own organisation. Doing this has a number of aims:

- To impress customers and prospects with the numbers themselves (always assuming that they are good!) and our willingness to share them publicly
- To demonstrate to customers and prospects that we have a mission to improve continuously
- To stimulate us to reach and maintain ever higher standards (and thus to improve continuously!)
- To put pressure on our competitors

Attributes can therefore be used as very powerful marketing weapons, especially if competitors are less open about such things (as is indeed the case with our competitors). They can make a very confident and positive statement about us as a vendor, and they are also powerful *agents of change* for our own processes.

What Do We Mean By *Process*?

Of the proposed strategies two, Document Inspection and Continuous Process Improvement, are predicated on the existence of *processes*. So what are processes and how do we define them?

> ***"Process is simply a way of getting smart systematically" Tom Gilb***

In ***Reengineering the Corporation*** Hammer and Champy provide an excellent definition of a process: "a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer" [Hammer93, p.35]. So, simply put, a process is anything we do to inputs to create outputs which are of value to our product or service.

We extend the definition of customer in that we describe a process as creating an output that is also of value to another, downstream, process. This may be a process within our organisation or in a customer's organisation.

We describe all of our processes in this way. We also define a taxonomy which contains two classes of process:

- A *production process* is one whose results are used directly by the product or service we are supplying to customers
- A *meta process* is one which is concerned with the ulterior or underlying principles of the production processes

Furthermore, by our definition a meta process is also one whose results are used to monitor, measure and improve the production processes and hence one or more attributes of the product or service we are supplying. Therefore, all processes such as Defect Detection, Defect Prevention, Process Ownership and Process Change Management are to be found in the meta process class.

In order to aid standardisation of our definition, we adopt the notion of a "generic" process which doesn't describe any process in particular, but rather the shape of them all. Figure 3 illustrates this generic process structure.

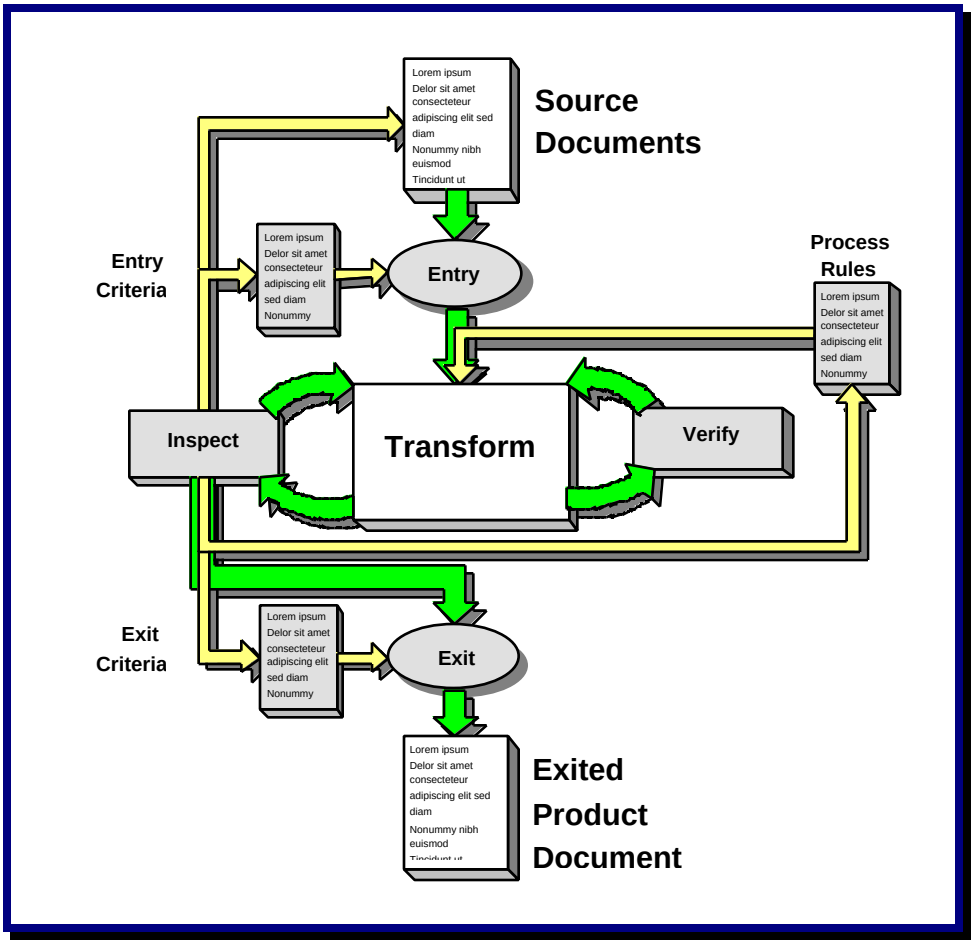

Figure 3: Generic process structure

The generic process includes the following components:

- One or more *Source Documents*, which are the inputs to the process
- A set of *Entry Criteria* against which the Source Documents are checked by an *Entry* sub-process for compliance before the process is initiated
- A set of *Process Rules*, otherwise the *Procedure*, by which the process should be operated
- A *Product Document*, which is the output from the process
- A *Transformation* (shown as *Transform*) which converts the Source Document(s) according to the process rules into the Product Document
- One or more *Correctness Verifications* (shown as *Verify*) which establish that the Product Document's technical content is correct
- One or more *Document Inspections* (shown as *Inspect*) which establish that the Product Document meets the agreed quality standards
- A set of *Exit Criteria* against which the Product Document is checked by an *Exit* sub-process before the process is terminated

The *Transform*, *Verify* and *Inspect* sub-processes are iterated until the Product Document is completed and exited from its final Inspection.

The *Verify* sub-process, analogous to what is known elsewhere as a *review* or *walkthrough*, can range from an informal exchange of views and information to a formal meeting with a variety of domain experts present. We have quite recently bounded the *Verify* sub-process by means of rules or procedures: it can either be invoked separately or by means of a formal verification role in a Document Inspection.

The *Inspect* sub-process itself is always a formal event whose rules are clearly defined; the initial reference we used being ***Software Inspection*** [Gilb93].

Each process has a written *Procedure* to be followed and a template *Form* for its Product Document. The *Exit Criteria* that the Product Document must meet before the process is complete are met by those defined for its Inspection through its *Inspection Artefacts*. Inspection itself being an instance of the meta process class is defined in precisely this way.

How We Inspect Documents

The primary objective of Document Inspection is the detection and removal of defects from documents before they are used as sources in downstream processes; for this reason it is also known as the *Defect Detection* process. For a full treatment of the Inspection process the curious are directed to ***Software Inspection*** [Gilb93].

In an Inspection, a document is checked rigorously against written rules, any breach thereof being classed as a defect. Defects fall into two categories: major defects being those likely to cause errors downstream; and minor defects being those unlikely to do so. The aim of any Inspection is to reduce the number of predicted major defects remaining in the document to an agreed, and calculable, level as defined in the *Exit Criteria* for Inspections. When a document reaches that point it is suitable to be used as a source in a downstream process and, most importantly, with a *known quality status*. Until a document has been Inspected its quality status is, of course, unknown and its defects undiscovered.

An Inspection is carried out by an *Inspection Team* created for the purpose. Within the team, there are a number of specialist roles:

Leader	the "project team leader" of the Inspection
Checker	one who checks the Product and Source Documents against the rules and checklists. Checkers may also assume more specialised roles, for example a *procedural* role or a *verification* role
Scribe	one who transcribes the issues raised during the Logging Meeting for subsequent use by the Editor while cleaning up the document
Author/Editor	the editor of the Product Document who may or may not be its original author

The Inspection process itself, illustrated in Figure 4, has a well defined procedure:

- Ensure that the *Entry Criteria* are met
- *Check* the Product Document against written *Rules*
- *Log* the issues raised by the checking
- *Edit* the Product Document where defective
- Raise *Document Change Requests* for defects in Source Documents
- *Exit* the Product Document when the *Exit Criteria* are met

In this and other such models the parenthesised codes in the boxes describing the sub-processes are the Process Identifiers; these will be seen again in the table following which enumerates the Product Documents and their intended readerships.

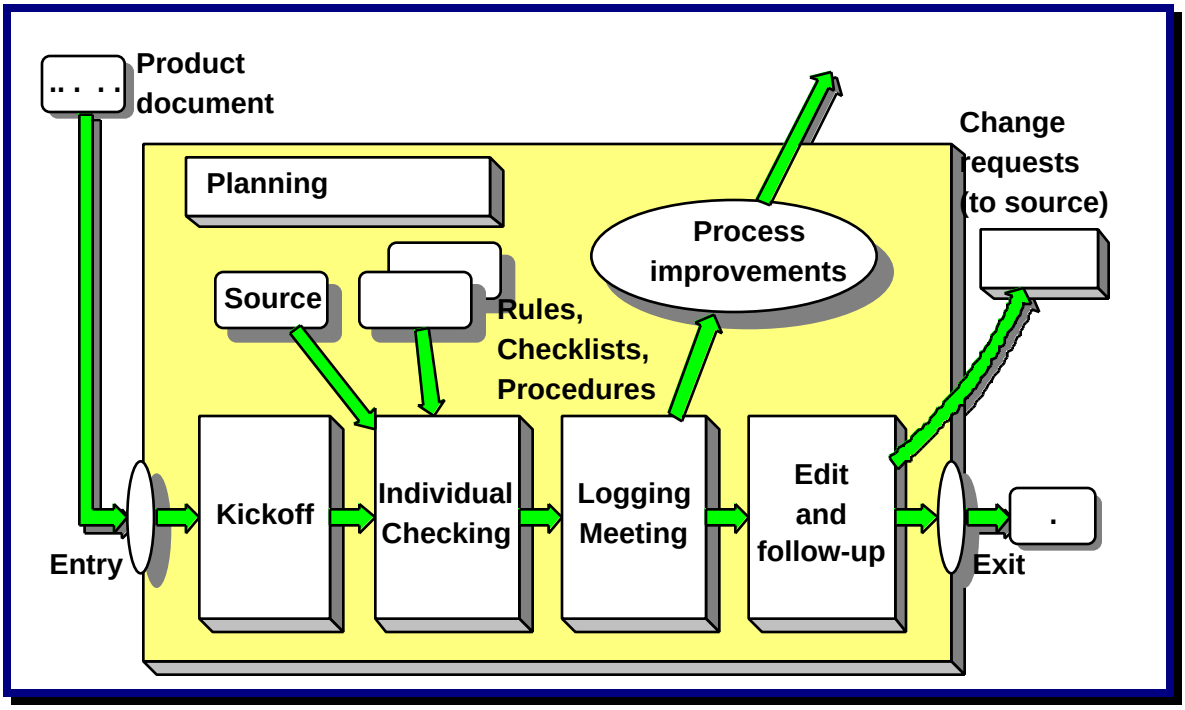

Figure 4: Defect detection (Inspection) process

A key feature of Inspections is the automatic feedback loop to process improvement. This means that every Inspection carries with it the potential to invoke a change to the process.

Process Improvement Suggestions are ideas for changes to the process that produced the document being Inspected in order to reduce systemic defect generation, and can be raised by anyone participating in an Inspection.

Process Brainstormings are held after some or all Inspections in order to generate more improvement ideas from all Inspection participants.

The results from both Process Improvement Suggestions and Process Brainstormings are fed out of Inspections to the *Process Owner*, who then decides whether to incorporate them into the process. This is depicted in Figure 3 on page 8 by the arrow from *Inspect* to *Process Rules*. The implication here, of course, is that every process which has Product Documents that are to be Inspected must have a Process Owner to implement the improvements.

The Document Inspection process, like all others, is defined by a number of artefacts:

Rules	specify the criteria against which a Product Document is to be Inspected. They include such attributes as form and content
Checklists	provide additional help and guidance to Checkers during Inspections
Procedures	define the procedures for each of the different participant roles of the Inspection Team members
Forms	contain the templates for the Inspection Artefacts, the documents used in the Inspection process
Entry Criteria	define the criteria that must be met (unless specifically waived) before a Product Document can be Inspected
Exit Criteria	specify the criteria that must be met before a Product Document can exit an Inspection

Table 1 shows the Product Documents and their intended readership for each sub-process of Inspection.

	Process	Product Documents	Intended Readership
DD1	Planning & entry check		
DD2.1	Kickoff - Leader	Master Plan	Checkers
DD2.2	Kickoff - Checker		
DD3.1	Checking - Leader		
DD3.2	Checking - Checker		
DD4.1	Logging - Leader	Data Summary	Inspection Leader
DD4.2	Logging - Checker		
DD4.3	Logging - Scribe	Author Advice Log	Editor
DD4.4	Logging - Author		
DD5.1	Process Brainstorming - Leader	Process Brainstorming Log	Process Owner
DD5.2	Process Brainstorming - Checker		
DD6.1	Editing - Leader		
DD6.2	Editing - Editor	Document Change Request	Source Document Editor
DD7	Follow-up		
DD8	Exit Check		
DD9	Product Release		

Table 1: Defect detection processes & product documents

Table 1 also illustrates how we document every process we have defined: by recording its unique process identifier; its name or description; its Product Document(s); and their intended readership.

The sub-processes of an Inspection as illustrated in Figure 4 are:

Planning & Entry in which the Inspection Leader establishes that the Product Document meets the Entry Criteria and then forms the Inspection Team

Kickoff in which the Inspection Team meet, the Leader allocates any specialist roles and completes the *Inspection Master Plan*

Checking in which the Inspection Team check the Product Document against the Source Documents, *Rules* and *Checklists*

Logging in which the Team log the issues raised by the Checking on the *Author Advice Log* and the Leader completes the *Inspection Data Summary*

Edit & Follow-up in which the Editor, who may or may not be the document's original author but is its current owner, cleans up the Product Document and raises any necessary *Document Change Requests* for Source Documents and *Process Improvement Suggestions*

Exit in which the Leader checks that all the logged issues have been addressed by the Editor, calculates the predicted remaining defects in the Product Document and exits it if it now meets the *Exit Criteria*

Process Brainstorming in which the Inspection Team brainstorm to discover if any systemic defects are being caused by process deficiencies

How To Prevent Defects

As we have seen, Gilb incorporates process improvement into his method by means of a feedback loop from Inspections to process change, implemented partly by *Process Improvement Suggestions* (illustrated in Figure 4 on page 11) emanating directly from Inspections and partly by *Process Brainstormings* (illustrated in Figure 6 on page 14).

Thus is the fundamental principle established of enabling the customers, practitioners and owners of processes to change them as they determine from their real-world experience of process usage. It is continuous, occurring as part of every Inspection that takes place, and requires no periodic review process because it's built in to the way everything is done as a matter of course.

This feedback is based on the principle of the *control cycle*, long used in manufacturing industries for quality control and improvement. Figure 5 illustrates the Shewhart/Deming Cycle, which defines a simple framework for continuous process improvement. This was first described as a control cycle by Dr. Walter Shewhart and developed by W. Edwards Deming in his seminal work on quality, ***Out of the Crisis*** [Deming86].

The cycle defines four stages:

- *Plan* to do something;
- *Do* it;
- *Study* the results;
- *Act* to improve the process for the next time.

This cycle can also be clearly seen in the Juran Trilogy Diagram in Figure 2 on page 4 as the *Lessons Learned* loop from Quality Improvement to Quality Planning.

In our context, the *Plan* and *Do* phases are those of the production process itself whilst the *Study* phase is represented by Process Improvement Suggestions from Inspections and Process Brainstormings, and the *Act* phase by any resultant process change effected by the Process Owners.

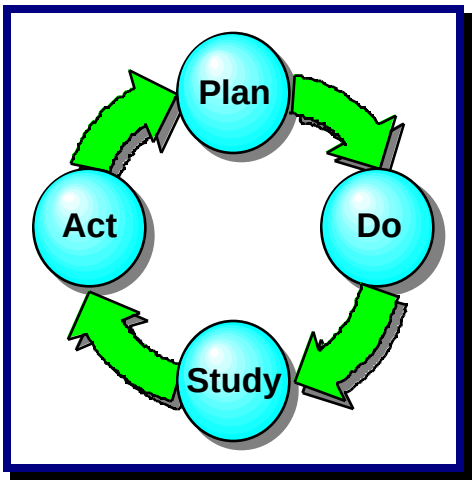

Figure 5: Shewhart/Deming cycle

Thus it can be seen that process ownership is a key element: for without process owners, processes cannot be improved.

Continuous Process Improvement, as we shall see shortly, is at the very heart of the method and is more than any other element what gives it such leverage. Because Document Inspection and Process Improvement feed each other and are built right into the production processes themselves, we are automatically in a virtuous cycle of change and improvement fuelling further change and improvement.

Synthesis

It is from all of these strands that we have woven our strategy for software quality improvement.

Bringing together Process Ownership, Document Inspection and the overarching Continuous Process Improvement yields the "big picture" shown in Figure 6.

We have seen how defining the production processes and their Product Documents allows us to embed Document Inspections into those processes, and how the feedback mechanisms inherent in Inspection together with Process Ownership provide the defect prevention framework of Continuous Process Improvement.

Defining Quantified System Attributes for our software and processes is also a vital part of this picture, and in some senses is the well-spring from which everything else flows. Remember that the attributes describe the targets or goals that we wish to aspire to; everything else in Figure 6 can be viewed as a set of strategies to enable us to attain those goals.

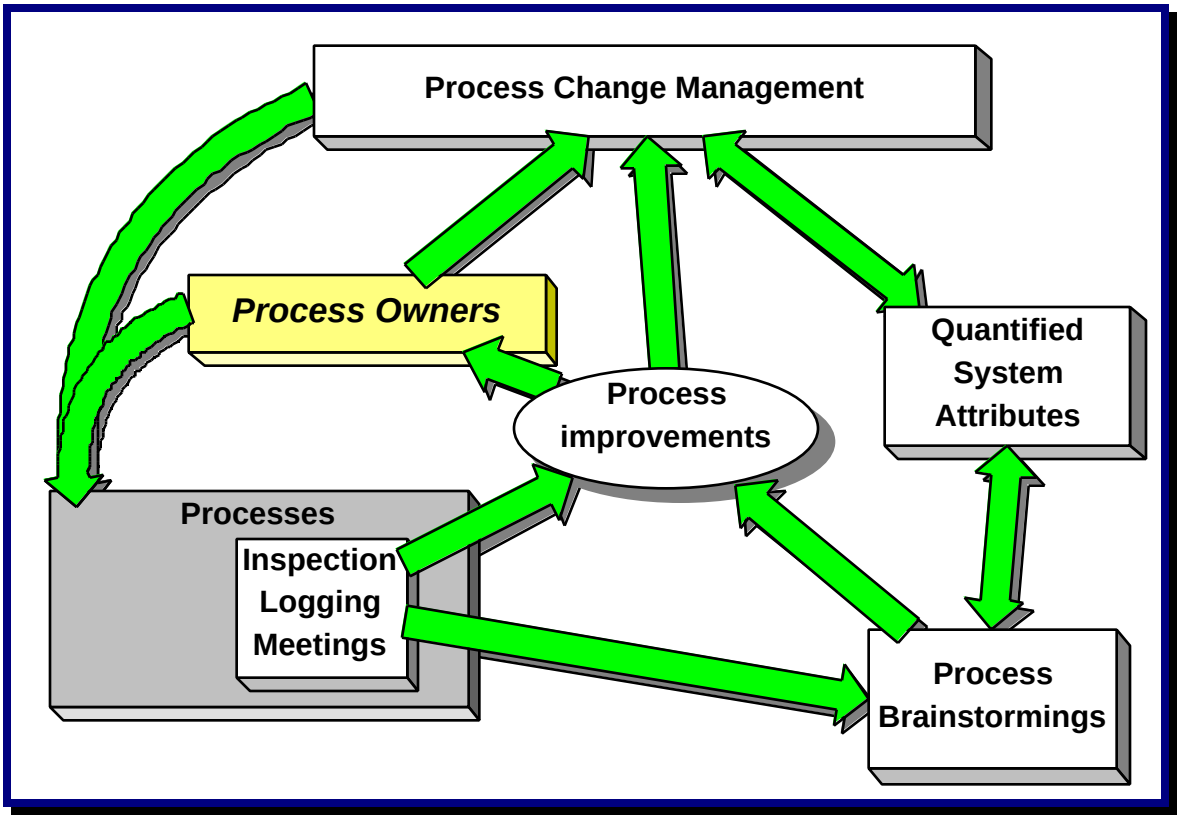

Figure 6: Continuous process improvement framework

In Figure 1 on page 3 we depicted the *SEI Capability Maturity Model* as an interesting, and industry-recognised, model of software development capability. Everything that we have described since then is intended to take us towards CMM Level 5, and thus become an organisation which possesses a *"continuously improving software development process"*.

We're now going to look at what we have done since December 1994 as we set out on the long journey to CMM Level 5.

How Did We Get Started?

Our start-up comprised a number of parallel activities:

- Identifying and defining the Quantified System Attributes that we wish to have for the system that we deliver to customers
- Identifying and defining our business processes and their inputs and outputs and relationships with each other
- Learning about and practising Document Inspection
- Setting up a support infrastructure for recording and logging all defect control activities - Defect Detection and Defect Prevention. This is our *QA Database*
- Initiating the necessary change in mindset and attitudes towards work and fellow workers. This is often referred to elsewhere as *culture change*

Defining Quantified System Attributes

Principles of Software Engineering Management [Gilb88] defines attributes as being of the software system shipped to its users. We extend that to include other measures that we believe are important not just to the delivered software but also to the processes by which we develop and support it. Examples of these are such attributes as Enhancement Completeness and Correctness which measure how well we've understood a user's requirements and translated those into system functionality (Completeness) and how right our solution is (Correctness).

Each Attribute is chosen to measure a single aspect of the system or a process and classified into either externally-focused (External Attribute) or internally-focused (Internal Attribute).

Figure 7 shows the External Attributes we have defined. In this figure, as in the Internal Attributes on page 17, the darker boxes are "molecular" attributes which are broken down into the "atomic" ones shown in the lighter boxes. It is the "atomic" attributes for which definitions are made.

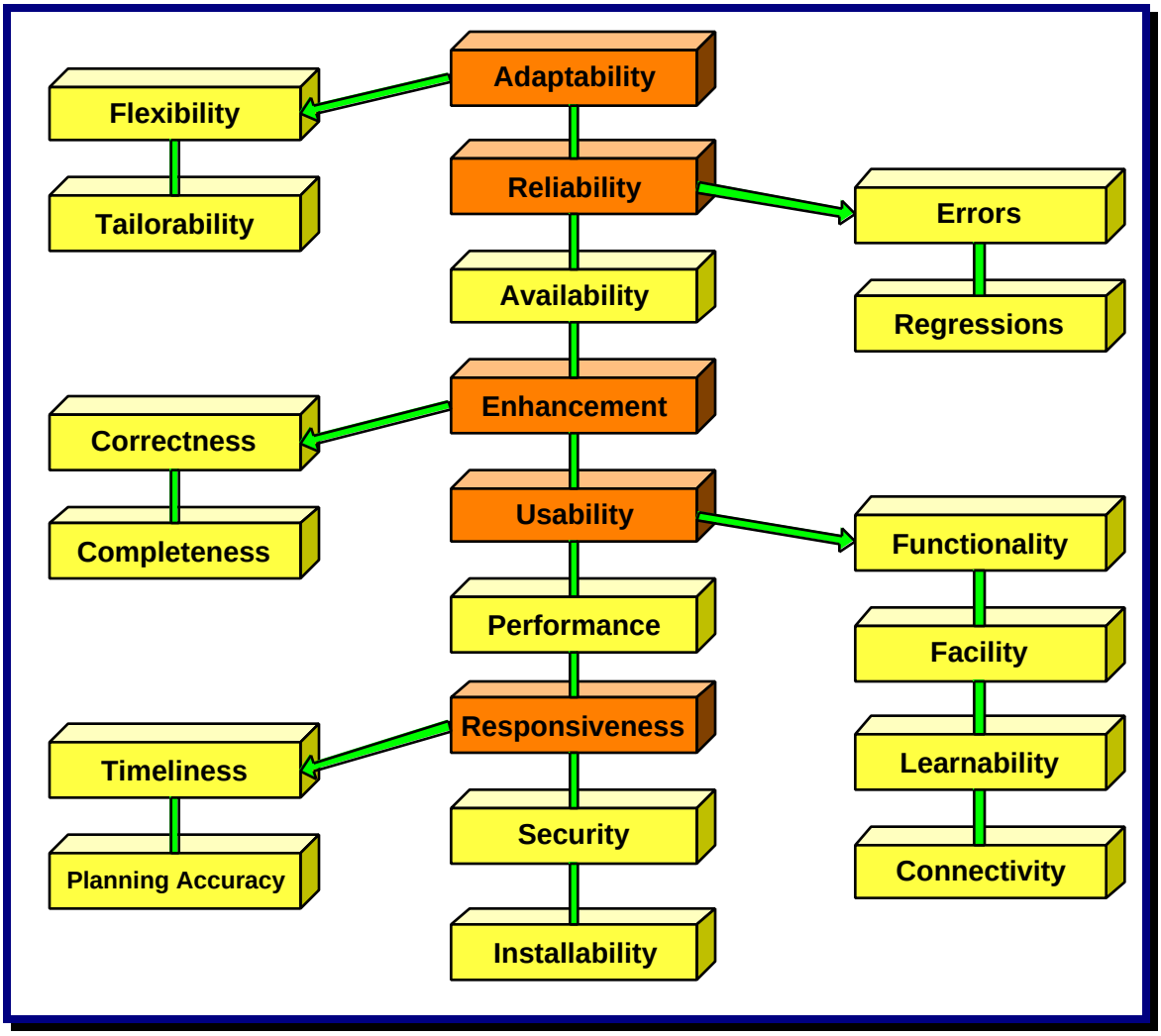

Figure 7: External system attributes

This is the meaning of each of them:

Availability	the percentage of scheduled time the software is available for customers to use
Completeness	the degree to which the development process delivers complete new functionality to customers
Connectivity	the degree to which the software can be connected to and integrated with other systems, services and technologies that customers require for efficient running of their businesses.
Correctness	the degree to which the development process delivers correct new functionality to customers
Facility	the ease with which a trained user can operate the system
Flexibility	the degree to which existing functionality can be adapted to customers' needs by on-site configuration without requiring software changes
Functionality	the degree to which the software provides the functionality requested by customers and agreed by us
Installability	the degree of disruption required to users' normal business for the installation or version upgrade of the software
Learnability	the speed at which a feature of the system new to the user can be learnt to be operated with proficiency
Performance	the degree to which the software facilities which are part of customers' critical business processes enable those processes to meet their business deadlines successfully

Planning Accuracy	the degree to which the planning process accurately plans and forecasts changes to the system
Reliability	the degree to which the software delivers error-free running for customers
Security	the degree to which the software detects and rejects intruders
Tailorability	the degree to which house style requirements for such as report templates can be met by on-site configuration without requiring software changes
Timeliness	the degree to which the problem management process meets its required fix times

Figure 8 shows the Internal Attributes that we have defined.

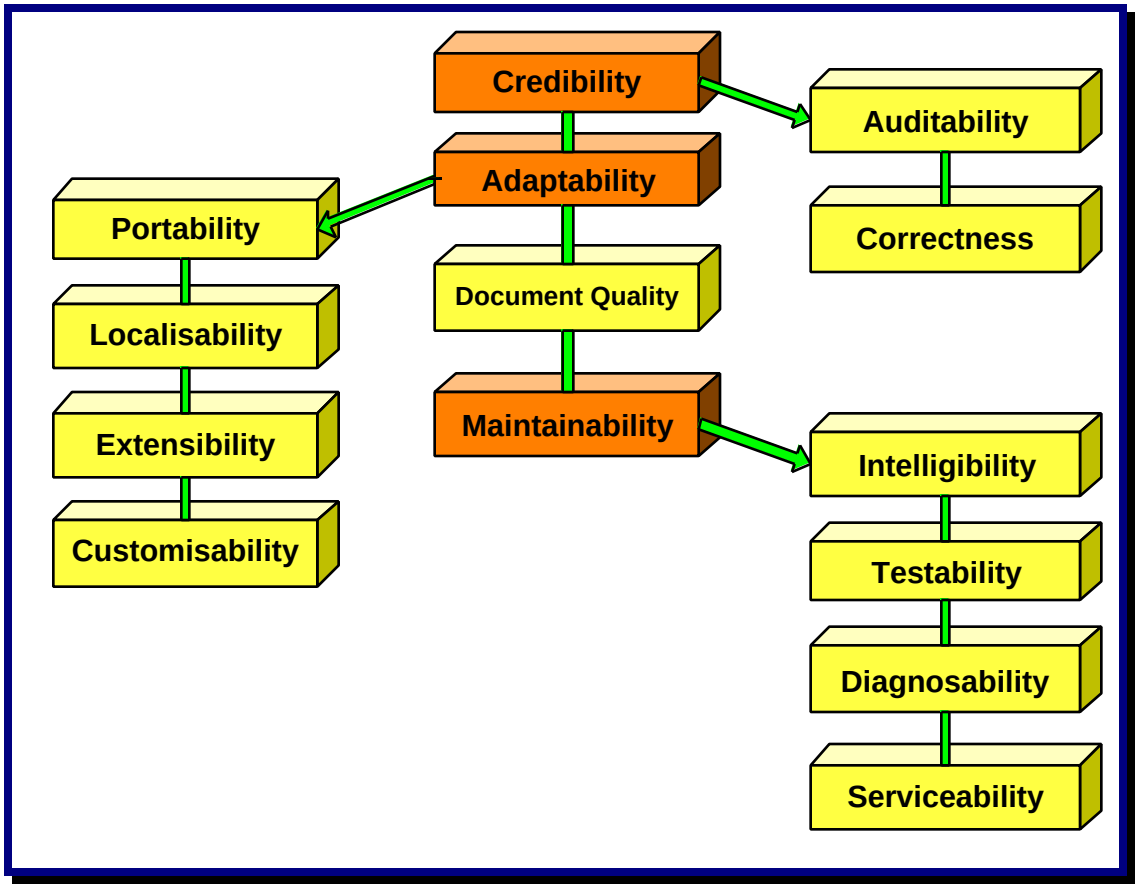

Figure 8: Internal system attributes

This is the meaning of each of them:

Auditability	the degree to which the system maintains an audit trail of its activities
Correctness	the degree to which the system behaves correctly in response to user actions
Customisability	the ease with which customer-specific functionality can be added to the system
Diagnosability	the ease with which a problem can be identified and its cause isolated
Document Quality	the permitted defect rate of all software engineering documents
Extensibility	the ease with which new functionality can be added to the system
Intelligibility	the ease with which the system, or parts thereof, can be understood to a sufficient degree to apply changes to it
Localisability	the ease with which the system can be configured for each target locale
Portability	the degree to which the system can be proven to operate with identical results in all of the target hardware and software environments
Serviceability	the ease with which changes can be applied to the software
Testability	the degree to which the system, or parts thereof, can be rigorously tested

We have defined each attribute in the following terms:

Gist	describes informally the essence of the attribute (in the Attribute lists above, it is the Gist of each that is shown)
Scale	specifies the scale upon which it is measured
Meter	describes how it is to be measured
Past	states the value that the attribute held in the recent past
Record	states any known industry or world best for this attribute (the ceiling to which we can aspire)
Must	states the value to which we believe we must take the attribute in the short term in order to achieve an acceptable level
Plan	describes a timed plan for longer term improvement of the attribute

Of the numeric values shown for an Attribute (*Past*, *Record*, *Must* and *Plan*), *Record* perhaps requires some further explanation. The purpose of showing the "world record" for an Attribute is to act as a reality check on our own aspirations to improve it - such records are typically set by organisations with considerable resources and are usually very costly to achieve. So the *Record* is there to stop us "chasing rainbows" when we know there's no point in trying to beat, for example, an IBM or an AT&T.

To illustrate an Attribute, this is how we have defined Reliability.

GIST	The degree to which the software delivers error-free running for customers
SCALE	Two numbers: reported errors per customer per year which require and those which do not require Local Fix Releases
METER	Sum error Software Change Requests (SCRs) raised in previous year by customers, grouped by Problem Severity Level(PSL): PSL 1-3 and PSL 4-6 Critical Reliability = $\frac{\sum(\text{error SCRs of PSL (1-3) in year})}{\sum(\text{customers})}$ Non-critical Reliability = $\frac{\sum(\text{error SCRs of PSL (4-6) in year})}{\sum(\text{customers})}$
PAST	56 : 58 [May94 - Apr95] fl Analysis of historical CR figures.
RECORD	0. Space shuttle. fl IBM Systems Journal Sep94.
MUST	24 : 60 [End 1995]
PLAN	18 : 45 [End 1996] 12 : 30 [End 1997] 6 : 15 [End 1998]

The Problem Severity Levels in this example are those defined in the Problem Resolution Service section of our standard Support Services Description, wherein Severity Levels 1 through 3 refer to problems experienced with processes that are critical to the customers' business and Levels 4 through 6 refer to those problems with processes that aren't. So the measures will be meaningful to customers in terms of the reliability of the system facilities that support their critical business processes. This is a good example of *customer orientation* of an attribute, and how we have approached the definition of all of them.

Pursuant to our goal of publishing openly all the key performance measures and indicators for the system, at the time of writing the first version of the External Attributes, containing the *Gist*, *Scale* and *Meter* for each, is with the representatives of the our product' User Group to share with them what we intend to measure and how. We are also interested in their feedback - they may have some ideas that had not occurred to us!

Defining Processes

We started by identifying the processes that are used to develop the software. This was a relatively easy start point since the processes are, or should be, well understood. For each process thus identified, we enumerated all of its sources of input and its outputs or Product Documents, and to each Product Document was ascribed a readership. The inputs helped us define the procedure to be used to transform the sources into the Product Documents. The outputs helped us define the rules for the Product Document, particularly the specific rules of content.

As part of this exercise we also defined the nomenclature to be used for document tags and the library (directory) structure into which all such documents shall be placed for subsequent access. This information is published in a navigational compendium, which is itself, of course, available publicly. The whole collection represents our *Document Repository*.

We saw earlier that each process must have an owner or owning group whose responsibilities include maintaining the process definition artefacts for subsequent use by process operators. This is illustrated in Figure 6 on page 14. We have found a good choice to be one person who performs the process and one who is a customer of the process - a recipient of the Product Document. Our Process Owners are also published in our compendium.

We developed the first cut of the Inspection Artefacts using ***Software Inspection*** [Gilb93] as a source; these are now owned by the process owners who amend them and publish new versions in the Document Repository as part of the Continuous Process Improvement program.

The Meta Processes

We identified earlier the meta processes as those outside the production processes - primarily the Defect Detection and Defect Prevention Processes. These are well defined by Gilb in ***Software Inspection*** [Gilb93] and we have used, without embellishment, the definitions to be found there.

There are other identified meta processes which we are still defining: for example Process Ownership Processes and Process Definition Processes. Beyond those, there are certainly yet more processes, unidentified and even undreamt of.

The Process Owners of the meta processes are, by default the *Process Change Management Team* (PCMT). The PCMT also has a broader responsibility to oversee all process change as illustrated in Figure 6 on page 14.

Training

Training for Document Inspection fell into two categories: that required for Inspection Leaders; and that for other participants on Inspections. The more important of the two is the former - well-trained Leaders can perform very satisfactory On-The-Job training for the other participants. It is also vital that every Inspection is led by an accredited Leader if time is not to be wasted and the real benefits are to be seen.

We were very fortunate in that we received Tom Gilb's own Inspection Leader course; he packed a five-day course into three days by extending the hours and at the end we had eight accredited Inspection Leaders. We also had an Inspection Overview Briefing for most of the other people in development which gave them an insight into what the process is all about and, most importantly, explained the terminology that would be used. This clearly has helped everyone to feel positive about the process from the very beginning.

Training Inspection Checkers, Scribes and Editors has continued ever since; every Inspection is a learning experience for everyone concerned, and we have found that new starters have taken readily to the process even as Authors once their initial fears have been overcome. In this respect, the positive and supportive attitude of their fellow inspectors has proved to be of immense benefit.

Training in Inspection is never finished: we are continuously learning more about how to run better Inspections, how to do smarter checking and how to improve the efficiency and effectiveness of defect discovery and removal.

Setting up the Support Systems

We realised that it was very important to begin recording the results of Inspections right from the start, so we set up our *QA Database* at the outset - it is very simple but very effective for our current needs. As our process matures, we shall enhance the database to cater for changes in our requirements; as we shall see later it is already providing us with invaluable information about our progress.

To provide for the introduction of Document Inspection into our development process, we have enhanced our Change Management System so that we can record the Quality Status for software changes whose code is to be Inspected.

We have set up a Document Repository on our file server for all documents whether part of the production processes or the meta processes and we are working on some navigation aids to help people find what they want. (Our first foray into the production of our navigational compendium has resulted in a document that is so complex as to require its own navigational aid!)

To enable unambiguous identification of statements in documents, especially important in Inspections and in citing them as sources in downstream documents, Gilb recommends that atomic statements be *tagged*. To assist with the tagging of documents and importantly to help authors meet the Generic Rules, we have developed a Word For Windows document template which provides a number of facilities. The use of this template has the additional benefit of giving authors a framework within which to write their documents and ensures that all documents so produced have a standardised look and feel.

What Have We Achieved (And Learned) So Far?

The main achievements we set out to attain have been:

- Improvements in software quality as a result of detecting and removing defects
- Improvements in document quality as a result of having better Product Document definitions and templates
- Improvements in the production processes themselves

We have made measurable gains in all of these areas which we will now examine; there have, in addition, been many other smaller wins too, some of which we will describe on the way.

Improvements in software quality

We now have no doubt that software quality is improving, and improving at a *spectacular* rate. History will be the judge as the results filter into installed products and get used by customers. The Scientific Method would have us conduct a control experiment: we are, however, unable to do this and so the savings in downstream costs are simply predicted. As we shall see later we are also tracking the defects removed as a result of Inspections against the error rate reported in usage of the system.

> ***"There is no doubt that the amount of code re-writes have decreased since last November. This is because most of the work is done on paper and coding becomes a clean implementation aspect" Hemant Mistry - Developer***

We have been collecting the statistics from all Document Inspections since the very beginning last December. It is worth noting that all the time spent by all participants in every sub-process of Inspections is logged and accumulated; thus we track very accurately the true cost of the process.

From the number of major defects found in an Inspection, we predict the hours saved downstream by their not being translated into errors discovered in test and field usage. This prediction is based on a conservative industry norm recommended by Tom Gilb of eight hours saved downstream for every major defect found in Inspection; although Trevor Reeve of Thorn-EMI has actually measured nine hours saved [Gilb93, p.20], and IBM have reported over a long period a ratio of one hour spent in Inspection having saved 20 hours test and 82 hours should a defect get into field use.

> ***"The experience gained and the standards I have had to apply as a checker, has made me more aware of quality in my own work. A major benefit of this is that I spend less time and effort wastefully revisiting past work" Iozeph Okosieme - Developer***

The Inspection metrics are, like everything else, made very public to help us feel good about our achievements and to spur us on to even greater things in the future!

Figure 9 illustrates some of the results achieved from all the Inspections we have conducted to date. We show the number of major and minor defects together with the process improvement suggestions and change requests for upstream documents ("Document CRs") that are logged in Inspections.

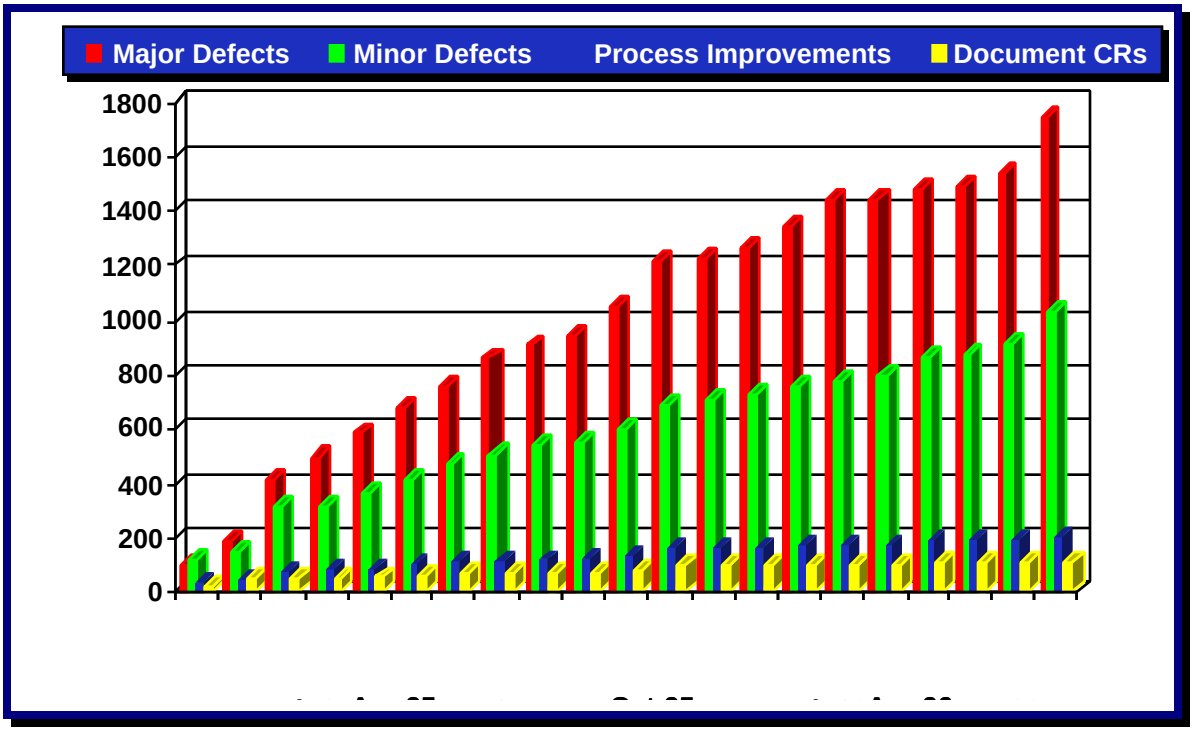

Figure 9: Cumulative inspection results

Figure 10 shows the cumulative amount of time spent in Inspections since we started in December 1994, together with the predicted time saved.

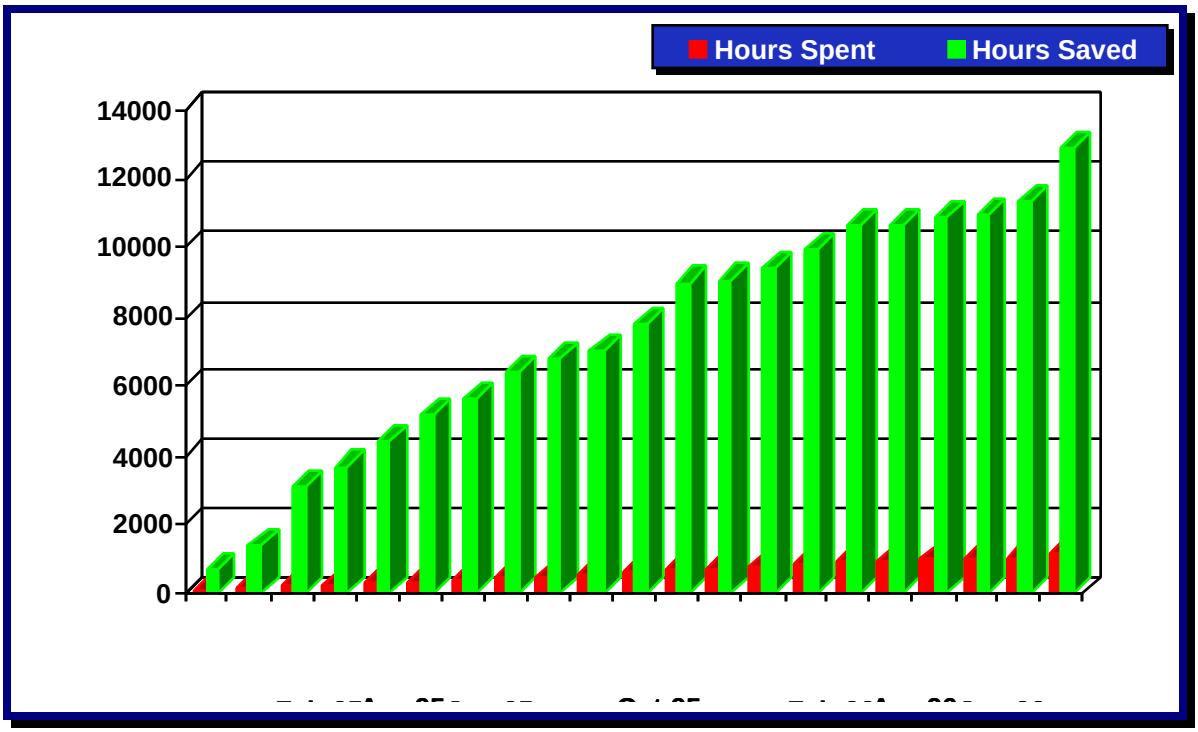

Figure 10: Cumulative time spent in and & saved by inspections

We have long recorded metrics on the changes requested to the software through our Change Management System - one of the things we measure is the arrival rate of error reports from all quarters (customers as well as our own in-house testing and usage of the system).

Now that we are also measuring defects found by Inspections, we can compare the two sets of figures as in the graph in Figure 11 which shows, on a monthly basis, the arrival rate of Error Change Requests (CRs) into our Change Management System (shown as "Total Count" since it includes those found both externally by customers and internally by ourselves) and the number of Major Defects found in Inspections since they started.

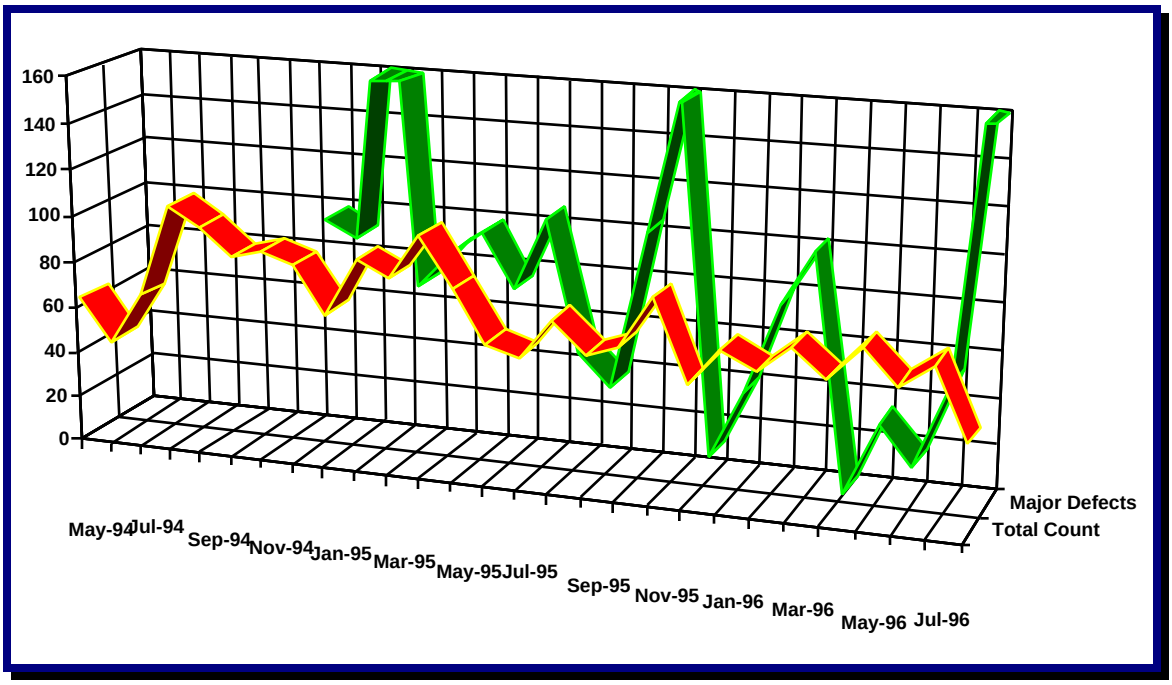

Figure 11: Inspection defects *vs* downstream errors

There will, of course, be a time lag between the major defect removal rate and the reduction in error arrival rate from the field since it takes time for the improved software to get into field usage, but we are going to track these two key indicators from now on and expect to see reliability improve (the software error totals to fall) as the number of major defects removed continues to rise.

As we saw earlier, we have an External Quantified System Attribute called ***Reliability***. This measures the number of high- and low-severity errors per customer per year and is therefore a good indicator to a customer (or prospective customer) of what he might expect from our software. Shown below is the graph of that Attribute which we publish monthly to ourselves and our customers.

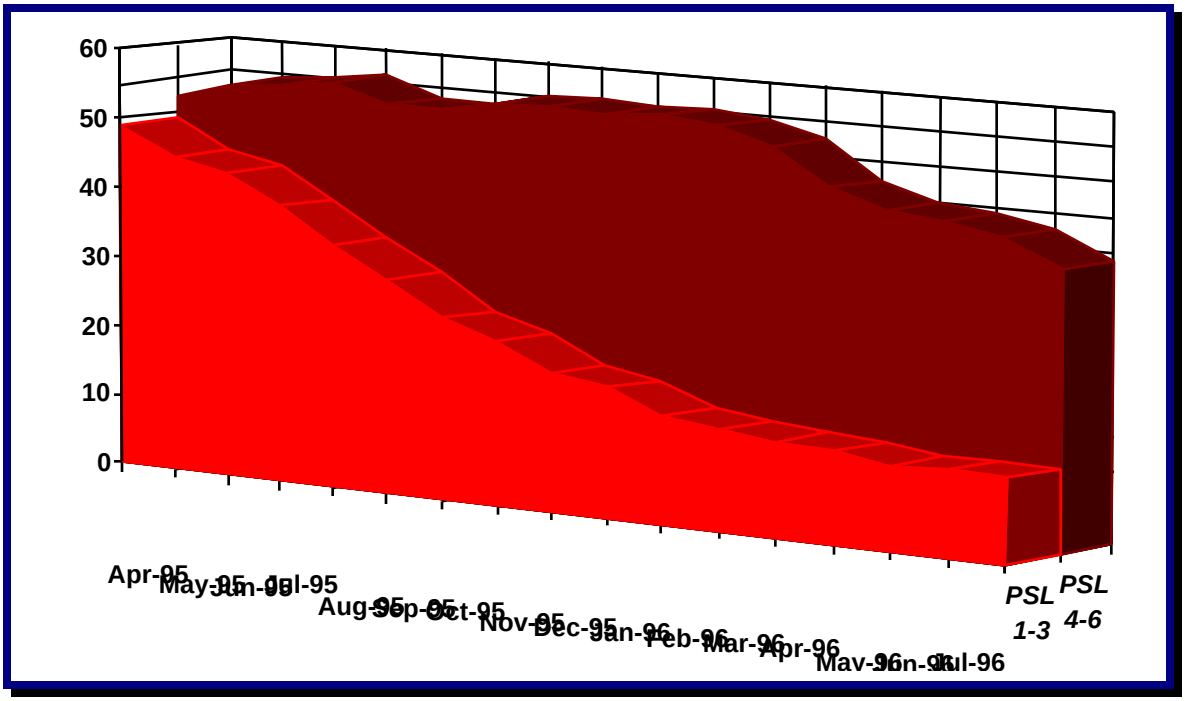

Figure 12: *Reliability* Attribute

Recall that we started doing Inspections in December 1994. The results of those Inspections would have started to reach the field in May and June 1995. You might imagine the impact this graph has on us, on customers and on prospects!

Improvements in document quality

We are experiencing an *amazing* improvement in the quantity and quality of documents produced - people are now striving to produce better documentation *because they want to*.

The inculcation of everyone with improved software engineering practice has led to a much clearer understanding and acceptance by most of the importance of sound documentation in support of the development and maintenance of the system.

We are, of course, motivated very strongly by our having taken the radical step of defining our Quantified System Attributes, and making them available publicly to our customers!

"It has raised awareness of how important good documentation is, and where errors come from. It has made me more wary of dodgy documentation, and given me a way of dealing with it - the author can no longer claim that a bad document is good if I can demonstrate that it isn't!" John Connor - Developer

We are always trying to find better and smarter ways of documenting the system and our processes, with most of the improvement suggestions originating, as expected, from Inspections.

Improvements in processes

Defining our processes has helped us understand what they and the Product Documents actually are, and as a result there is no longer any confusion about it. Because process definition and ownership is now a responsibility visibly shared by everyone, most people feel that generally we have the "right" processes and, more importantly, that together we have the ability to change them whenever we see fit.

> ***"Inspection has improved not only the quality of documents and code and forced standardisation but also improved communication and cohesion between departments" Jason Da Cunha - Developer***

Process definition has obviously helped us focus on the processes we *want* to have and the types of documents we *want* to generate in the development and support of our product. As always, the trick is in striking the right balance between speed and safety: just like driving a car, the faster you go and the more corners you cut, the more risk you run but the quicker you'll arrive (assuming you survive!). But the Process Ownership/Continuous Process Improvement framework gives us immense power to change things very quickly if we wish.

There is nothing prescriptive or proscriptive and nothing mandatory at all about "the way we now do things around here". We have guidelines published as rules, procedures, checklists and entry and exit criteria and we publish *Best Practice* documents for people to aspire to, and everyone has the power to initiate any kind of change. We produce no documents simply to obtain a signature on a piece of paper, because we don't have any sign-offs.

Everyone now understands the penalty of not doing it right: we deliver software of unknown quality. So we are all strongly motivated to get the defect rate down, and we now have the tools to do that: producing documents (and code) to better those already in the best practice catalogue, and Inspecting them!

As we shall see later, we have only just begun to understand the implications of having all this power at our disposal. We have not yet looked beyond the development processes to the wider issues of the Icon business as a whole. We are certain that when we do, we shall discover that further process improvements can be wrought to the lasting benefit of the product.

What difficulties have we experienced?

One of the major difficulties we have encountered is finding a way of giving people enough time to conduct Inspections.

> ***"What improvements have I seen from process ownership and continuous process improvement? I'm afraid I haven't seen any. I have had NO feedback from document CRs raised through inspections. Process ownership is too obscure, and (here's a quote from someone else) 'There's just too many document types'" Dominic Thomas - Developer***

What we have done is to concentrate on selecting critical and high risk developments and to Inspect some or all of the Product Documents of those. We've recently improved that strategy by selecting a major new project and planning to Inspect every Product Document we generate. We are also going to revise our development capacity model to account better for the time consumption of Inspections and we're going to incorporate Inspections in everyone's objectives for 1996.

Allocating time to the other activities such as process ownership activities, for example artefact definition (rules, procedures etc.) has also proved to be difficult.

We believe now that our failure to address process ownership adequately is due to its not being part of anyone's objectives, so people have found it easy to ignore.

> ***"This is one area that I feel we haven't taken enough action on. We have plenty of process change requests/ideas but we have not really acted on most of them" Niru Reid - Project Manager***

The same goes for process change management, where our Process Change Management Team, comprising your author and two others, has done precisely nothing so far, although to do something would elevate us, at least on that score, to CMM Level 5 instantly!

Changes in the way we think

> ***"The mentality of most, if not all developers pre-Inspection was 'this is my document and my code, as I understand it that is good enough'. Now the 'for my eyes only' approach is out the window and I believe everyone thinks about the possible readership of what they are writing" Shaun Hooper - Developer***

One of the benefits of the new mindset is that most people are now very keen on sharing their work with others. We encourage people to submit documents, either their own or those of others, to be included in our catalogue of best practice documents which is available for all to see. The purpose of this is to give good guidance to authors, new and old alike, without requiring cumbersome standards manuals to be followed. It also enables us to change the content and layout of documents, and even invent entirely new document types very rapidly by simply publishing a new best practice example and exhorting people to emulate it.

Table 2 shows an example of the best practice catalogue.

Type	Description	Document	Author/Editor	Nominated by	Date
FRS	Functional Requirements	lm0028.doc	John Connor	Shaun Hooper	11/04/95
		lb0023.doc	Paul Westgate	Paul Westgate	11/04/95
		bg0646_5.doc	Shaun Hooper	Shaun Hooper	11/04/95
SS	Solution Spec.	lb0058.doc	Jason DaCunha	Jason DaCunha	21/04/95
PDS	Program Design	pv7p2.doc	John Connor	John Connor	11/04/95
SC	Program Source Code	coat.c	Hemant Mistry	John Connor	11/04/95
		coin.c	Philip White	Niru Reid	10/10/95
CAF	Concepts & Facilities	vlve.doc	Niru Reid	Niru Reid	11/04/95
	Test Spec. & Plan	testdc37.doc	Andrew Smith	Andrew Smith	11/04/95
		bg0844.doc	Shaun Hooper	Shaun Hooper	11/04/95
	Test Data Spec. & Plan	testdc38.doc	Andrew Smith	Andrew Smith	11/04/95
TG	Technical Guide	trialbal.doc	Rob Dixon	Manoj Gupta	11/04/95
		bg0842_3.doc	Shaun Hooper	Shaun Hooper	11/04/95
	Memo	bg6461.doc	Shaun Hooper	Shaun Hooper	11/04/95
	Minutes	bg646m2.doc	Shaun Hooper	Shaun Hooper	11/04/95

Table 2: Best practice documents

People now mostly understand that control rests with them. The majority have responded well to the challenge of being told "it's now up to you - there are no safe hiding places anymore - you get out what you put in and no-one else is going to do it for you".

Of course, with this empowerment goes real responsibility, and people are now beginning to rise to it, exploring ways of doing things better, trying new techniques in documents, looking for ways of improving checking on Inspections, inventing experimental rules to check by, and a host of other incremental improvements.

Not everyone has grasped the nettle to the same extent, however.

Because this is such an immense change in the way that work is carried out, it has proved to be a shock for many, and some are still having difficulty in coming to terms with the ramifications. One of the things we are still learning is how to convey the message of the method in such a way that we get people to "buy in". Peter Senge, MIT Professor of Systems Thinking and Organisational Learning, writes about sharing a vision in his book ***The Fifth Discipline*** [Senge90] and offers the following spectrum of responses.

"X to Y 'Y, can you attend an inspection kickoff on Thursday please?' The reply, 'sorry too busy, too many important things to do, maybe next time'" X - (anonymous but committed)

Possible Attitudes Toward A Vision.

Commitment - Wants it. Will make it happen. Creates whatever "laws" (structures) are needed.

Enrolment - Wants it. Will do whatever can be done within the "spirit of the law".

Genuine Compliance - Sees the benefit of the vision. Does everything expected and more. Follows the "letter of the law". "Good soldiers".

Formal Compliance - On the whole, sees the benefits of the vision. Does what's expected and no more. "Pretty good soldier".

Grudging Compliance - Does not see the benefits of the vision. But, also, does not want to lose job. Does enough of what's expected because he has to, but also lets it be known that he is not really on board.

Non-compliance - Does not see the benefits of vision and will not do what's expected. "I won't do it; you can't make me".

Apathy - Neither for nor against vision. No interest. No energy. "Is it five o'clock yet?".

Amongst the development team, we have people with attitudes drawn from right across this spectrum; our ultimate goal is to share the vision in such a way that the majority of the attitudes move to those of commitment and enrolment. But at the same time we recognise that for some the change required is too great and the environment too uncomfortable for them - we have indeed already experienced a limited amount of "fall out" as a result of the need for such radical change.

"I agree in principle with the idea of inspections" Z, when declining the opportunity of checking on an Inspection - (anonymous and possibly apathetic)

Where To Next?

One of the most tangible effects of our new-found culture is that decisions are no longer seen as the exclusive preserve of "management", *everybody* is now actively involved in change. We continuously generate new ideas about how to build on the start we have made and maintain the momentum - these we shall describe next.

Immediate steps

We are concerned that Document Inspection is still too much of a "big deal" for us and is not yet sufficiently embedded into our way of life.

> ***"Inspection needs to be more 'In Your Face'. The current motivation method is to let the combination of Tom's talk on checking, and the proof, the statistics, speak for themselves. This is okay, but we need to shout out the results of Inspection. (I did a spot poll and it is amazing how many people don't know how successful Inspection is.)" Shaun Hooper - Developer***

We are exploring a number of strategies to overcome this:

- Automate the capture of all the data from Inspections
- Put the whole Inspection process on-line electronically to eliminate the paper entirely
- Making Inspections part of everyone's core objectives
- Setting Inspection objectives for every project that we run
- Inventing better ways of sampling big documents so that we can do more, smaller Inspections
- Encouraging people to Inspect a document even while it's being developed (at the time of writing this your author is conducting an experiment with a document of his authorship)

We've also very recently revitalised the Process Change Management Team and made it more fluid and dynamic and less management-oriented in an effort to concentrate on many of the Process Ownership issues that have lain fallow.

Spreading the word

> ***"It's too early to evangelise - you should wait two or three years" Tom Gilb***

We are going to tackle next our Customer Services processes and visit again our Problem Management process. The latter is particularly interesting in that the process must operate on a very short timescale (4 hours for an Operational Fix for Problem Severity Level 1 is called for by our standard Problem Resolution Service) and we want and need to Inspect and exit documents in that time!

So far we have addressed the product development and shipment processes in isolation. They exist, however, in the context of the Icon business as a whole. We are now therefore going to develop a holistic process map for the Icon business.

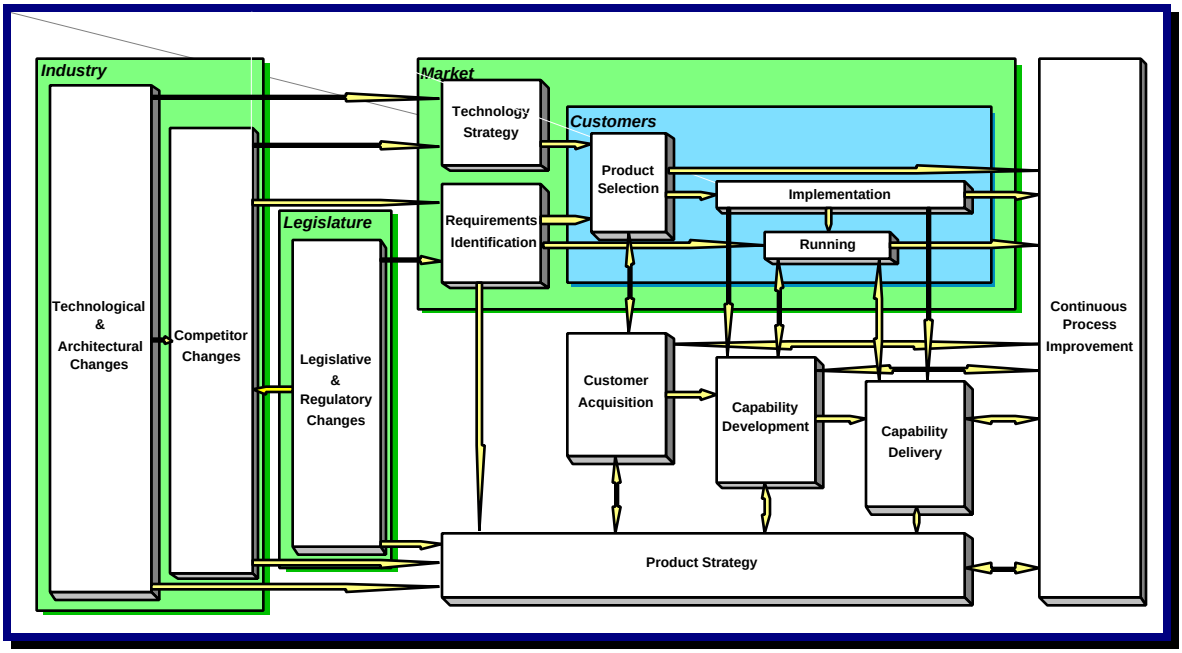

Figure 13: High level process map

Within each of the process groups in the map are the atomic processes that we use to run our business - including software development and shipment, and the Customer Services activities that we are going to address next. This map helps us to look at the whole business from a process viewpoint and thereby ignore functional boundaries when designing new and better processes

Looking at the fundamentals

What are we observing today?

The resource consumption of testing and problem diagnosis and resolution continues to rise: indeed, this has been a quite deliberate act. However, in terms of our original “time to market” objective this is effort which could be otherwise employed in developing more product functionality.

This does, and in the short term will continue to, bring significant benefit to the product: we are removing more and more defects not just from newly developed or enhanced facilities, but also from much of the “old” code right across the system. As a result of this focus, the defect rate from field usage is expected to fall in the short term, but as more functionality is incorporated into the product, and more customers use it in diverse ways, the rate will rise again.

So we have a defect rate which rises and falls, as more or less effort is put into defect correction and testing, but the underlying trend is always upwards. Ultimately, the resources required to maintain the defect rate down to an acceptable level may absorb all the available staff for all of their time: this scenario is illustrated in Figure 14 below.

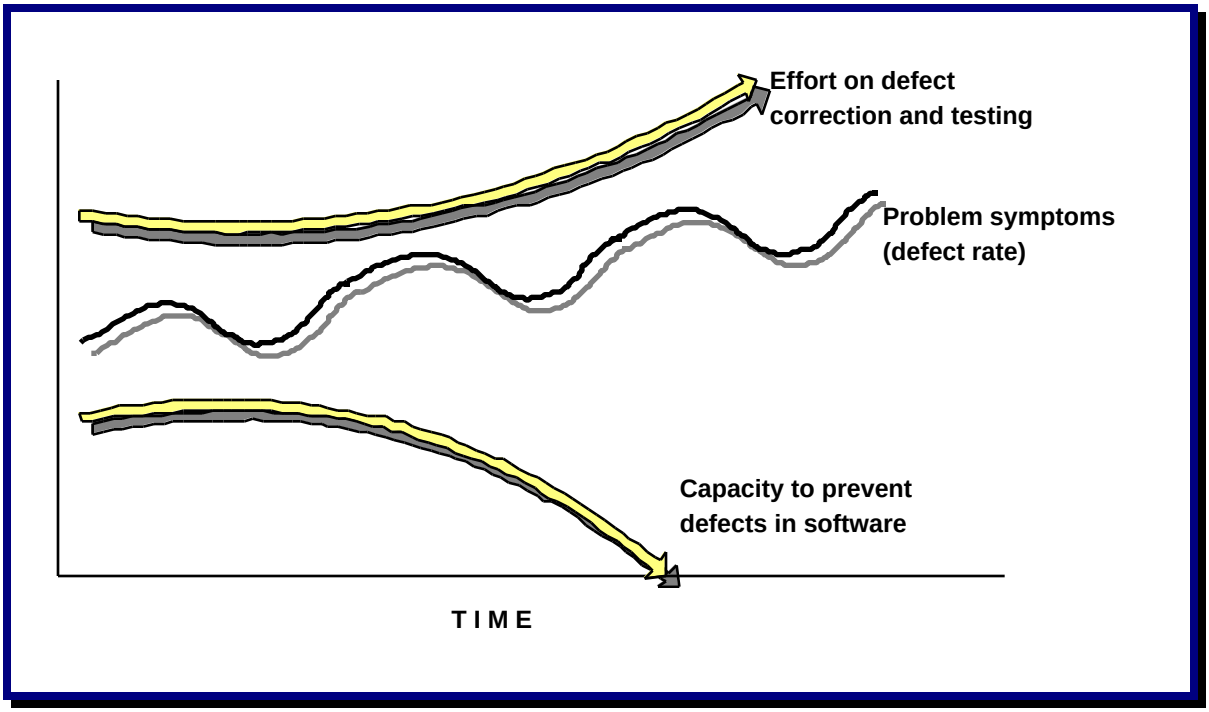

Figure 14: Effect of defect correction and testing

What could be the causes?

This situation can be expressed succinctly in a *Causal Loop Diagram* or system model, as described by Peter Senge in ***The Fifth Discipline*** [Senge90]. In the book Senge defines a number of standard template models or *system archetypes*; he calls this one "Shifting the Burden".

Figure 15 depicts the "Shifting the Burden" archetype, applied to our defect correction and testing issue.

In this archetype, a symptom-correcting process is adopted in an attempt to rectify a problem. This process is shown in the upper balancing loop (indicated by a balance on a fulcrum in its centre). The problem-correcting process, the one that would cure the disease instead of merely treating the symptoms, remains, however, unexplored. This process is shown in the lower balancing loop. The long-term consequence of pursuing the wrong strategy is shown in the large reinforcing loop (indicated by a snowball effect in its centre): a reduction in the capacity of this system to fix itself over time because the *addiction* to the symptom-correcting process diverts increased resources away from the ability to correct the real problem.

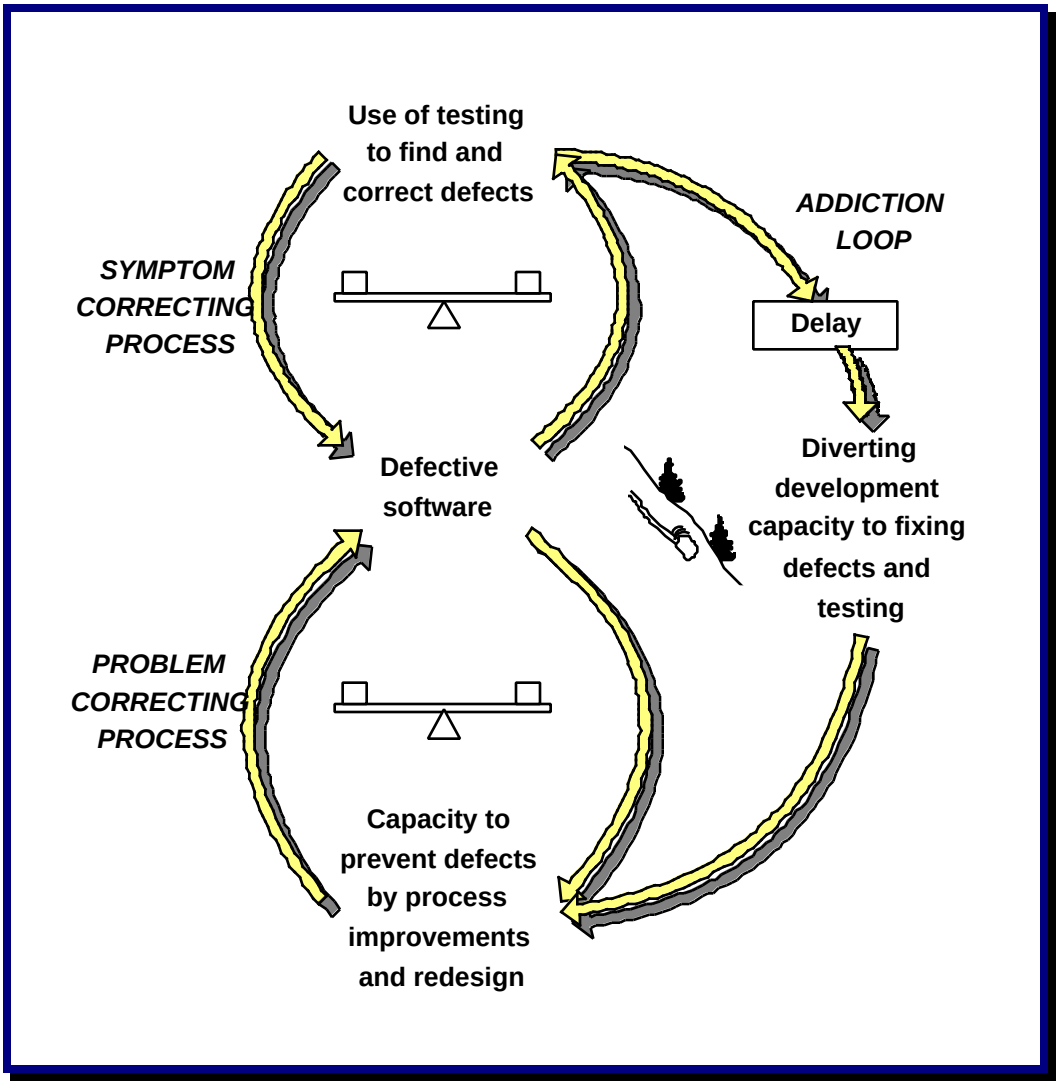

Figure 15: "Shifting the Burden " archetype

In our case, the symptom-correcting process being followed is defect correction and testing of defective software which brings short-term benefits, but the capacity to invoke the fundamental problem-correcting process, preventing defects by process improvements and system redesign, is slowly eroded by the addiction loop of diverting more development capacity to removing defects and testing the results.

Fixing the problem by improving the processes

This is, of course, to the detriment of reaching our “time to market” goal: defect correction and testing between them take far too much time in practice. Recall IBM’s measured times of 1-20-82, and the conservative 1-8 we’re using to predict time saved by defect detection in Inspections instead of test and field use. Document Inspection is just such an improvement which prevents defects getting to test and thence the field and so is part of the problem-correcting process. Other parts are defect prevention by Continuous Process Improvement and, equally importantly, fundamental software redesign.

Our experience in this regard is, however, not atypical of most software development organisations: removing the defects found by testing the software after it’s been written instead of not inserting them in the first place. It is still only the advanced organisations using such techniques as Document Inspection and Continuous Process Improvement [Gilb93] who are seriously attacking that fundamental problem. And, of course, we intend to become one of those advanced organisations!

These arguments make it compelling to study what other systems may be underlying our processes; the "Shifting the Burden" archetype illustrated is only one of many such systems at work.

Projectising Our Work

The methods and techniques that we have adopted and developed encourage a very open, egoless culture. We have recently formalised this in a set of policies and rules for processes and documents, with the emphasis on *projectising* our work. Projectising, a term coined by Tom Peters in ***Liberation Management*** [Peters92], means turning all of your activity into projects with defined targets, goals or objectives, and running each with a defined project team.

Here is our current policy for projects and project teams:

A Policy for Development Project Teams

Version [0.3]; **Date** 22 March 1996; **Editor** DH; **Tag** POLICY.DT; **Rules** RULES.G;
Pages 0.5; **Readers** Unrestricted; **Status** Unexited > 60 defects/page; Unverified

DT1	PROJECTS	All work shall be carried out within projects, either for customers as defined in {CUSTOMERS} or for internal goals as defined in {GOALS}.
DT2	TEAM	A project team shall be created for each project and disbanded when the project objectives have been met.
DT3	LEADER	A team shall have a leader, who shall be selected by a method chosen by the team members.
DT4	OBJECTIVES	A project shall have written objectives, as defined in **"SPECIFIC RULES: Project Objectives"** {RULES.PO}, agreed between its customers and the project team.
DT5	START	The first task of the project team shall be to author the Project Objectives Document as defined in {OBJECTIVES}.
DT6	CUSTOMERS	The two classes of customer herein defined are external (commercial product Customers) and internal (other parts of the vendor organisation) who, as downstream processes, are ***de facto*** customers.
DT7	GOALS	Internal goals for all teams include, but are not limited to, education, customer business exposure, project team leadership; external goals are defined in **"SPECIFIC RULES: Project Objectives"** {RULES.PO}.
DT8	AIM	The ultimate aim, when this policy is pervaded across all technical and commercial processes is an organisation of Self-Managing Case Teams.

The final tag, *DT8 AIM*, in this policy refers to *Case Teams*. Case Teams are self-managed work teams which are characterised by being small, fast, empowered and highly focused on their project, what Tom Peters calls "Customerised" [Peters92]. We are already making significant strides in that direction with an increasing number of customer-focused development projects.

The policy above also refers to a Project Objectives Document, and this is the current rule set for that document.

SPECIFIC RULES: Project Objectives

Version [0.6]; **Date** 12 July 1996; **Editor** NR; **Tag** RULES.PO; **Rules** RULES.G;
Pages 0.5; **Readers** Unrestricted; **Status** Unexited > 60 defects/page; Unverified

PO1	READER	Readership is unrestricted.
PO2	PRODUCT	Project deliverable products shall be defined.
PO3	TARGET	Tracking methods, progress reporting including estimate revision, communication with customers, and milestones shall be defined.
PO4	STRATEGY	Variations from product strategy both business and technical shall be documented with an impact and risk analysis.
PO5	ATTRIBUTES	Relevant system attributes {flATTRIBUTES} shall be listed and an estimates of improvements on them shall be enumerated.
PO6	PEOPLE	Targets for developing the people on the project team by: adopting new team roles, learning and applying new skills and expertise, shall be set.
PO7	EXPERIMENT	Conducting at least one organisational, project management or technical experiment in pursuit of the continuous improvement goal

As you can see, there is considerable emphasis on developing more than simply application functionality: meeting our targets for Quantified System Attributes, and developing the people on the project team attract as much attention. The last rule, tagged *PO7 EXPERIMENT*, enjoins project teams to try at least one new thing on every project and to share the results with others; in this way we believe we can enjoy a constant influx of new insights and discoveries through the activities of the project teams.

The Continuing Challenge

It is a constant battle against the forces of evil - deadlines, schedules and promises made to customers!

We continue to seek ways of doing *everything* better. Document Inspection is clearly not a complete answer, there are other mechanisms that we must look for to bring domain knowledge, skills and expertise, energy and commitment to bear on every aspect of our business.

We are now continually asking ourselves questions:

- *Do we have too many document types? Should we dispense with some of them?*
- *Do we have too few document types? Does this make documents too big?*
- *Are we doing Inspections optimally?*
- *Are there smarter ways of doing checking? Can we automate any of it?*
- *Are there smarter ways of choosing Inspection teams to make it more efficient?*

Look at these quotes from practitioners

"Do we inspect too early? Are inspections sometimes used as extended requirements gathering or review? Should we encourage peer review and walkthroughs as a way of getting to an inspectable document?" Paul Westgate - Business Analyst

"I have become a great believer in choosing checkers for an inspection based upon their knowledge of the subject matter being inspected. The idea that ANYONE can inspect a document is fundamentally flawed. A document can be entirely self-consistent, beautifully tagged, and clearly reference all sources whilst still containing serious fundamental errors" Dominic Thomas - Developer

"It should be noted that Document Inspection is not the ultimate solution. It will not necessarily improve the technical accuracy of the original source document. This fundamental concept must be understood to avoid disappointment" Shaun Hooper - Developer

In other words we may yet be making the BS5750 mistake - having an elegant process that produces beautifully documented garbage! This is one reason why we are now concentrating on defining the Verify sub-process in our generic process structure (see Figure 3 on page 8).

What these and many of the other quotes throughout this paper do illustrate, however, is that more and more people are now starting to question everything we do - and that's the beginning of the real awakening.

References

[Deming86] W. Edwards Deming. *Out of the Crisis*. Cambridge, England: Cambridge University Press, 1986.

[Fagan76] Michael E. Fagan. *Design and Code Inspections to Reduce Errors in Program Development.* Armonk, NY: IBM Systems Journal Vol. 15 No. 3, 1976.

[Gilb88] Tom Gilb. *Principles of Software Engineering Management.* Wokingham, England: Addison-Wesley, 1988.

[Gilb93] Tom Gilb and Dorothy Graham. *Software Inspection.* Wokingham, England: Addison-Wesley, 1993.

[Hammer93] Michael Hammer and James Champy. *Reengineering the Corporation*. London, England: Nicholas Brealey Publishing, 1993.

[Juran88] J.M. Juran. *Juran on Planning for Quality*. New York, NY: Macmillan, 1988.

[Paulk93] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis and Charles V. Weber. *Capability Maturity Model for Software, Version 1.1*. CMU/SEI-93-TR-24. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, February 1993.

[Peters92] Tom Peters. *Liberation Management.* London, England: Macmillan, 1992.

[Petrozzo94] Daniel P. Petrozzo and John C. Stepper. *Successful Rengineering.* New York, NY: Van Nostrand Reinhold, 1994.

[Senge90] Peter M. Senge. *The Fifth Discipline*. London, England: Century Business, 1990.