Transactions on Information and Communications Technologies vol 8, © 1994 WIT Press, www.witpress.com, ISSN 1743-3517

Software process improvement: the route to software quality?

W.L. Smith,[a] R.I. Fletcher,[a] E.M. Gray[a] &
R.B. Hunter[b]

[a]*Department of Computer Studies, Glasgow Caledonian University, Cowcaddens Road, Glasgow, G4 0BA, UK*

[b]*Department of Computer Science, Strathclyde University, Livingstone Tower, 26 Richmond Street, Glasgow, G1 1XH, UK*

ABSTRACT

The purpose of this paper is to provide a balanced view of software process improvement and its relationship to software quality. The paper looks at the historical background to software development and explores the importance of correct management practices including sufficiency (to lead to repeatable high levels of software quality). It also discusses leading initiatives in complementary software process engineering and offers an assessment of the value and applicability of current approaches to software process improvement. In particular the discipline of software process assessment is evaluated. The import of utilising software process models and related measurement is also briefly discussed. It is shown that software process improvement approaches considered are not yet mature sciences, in particular there are significant unresolved research issues and possible inadequacies associated with software process assessment. In conclusion, although software process improvement is not yet proved to be the route to software quality, it remains a possible route.

The question of the readiness of software producers in terms of knowledge and current capability of their software production practices to make effective use of process improvement techniques is also addressed. It is shown that a very high percentage of software developers have poor knowledge of software process issues and no visible software process. This paper concludes that their best action is to seek to install appropriate elementary management practices thereby achieving a state where software process improvement may become applicable to a visible process, should provable benefits be forthcoming.

INTRODUCTION

The Import of Software Quality and Associated Problems

Production of superior **quality software** is currently a much sought after and desirable yet elusive state. Quality is becoming a demand from the user community, with its import growing steadily on a global scale. Indeed it is reckoned that by 2050AD properly certified software will be the primary global trade-flow, Slater [1]. Also as Hollocker [2] suggests, the cost of information is tightly coupled to the cost of the software that processes and directs it, so software quality affects a user's economics through its effect on income and costs. It is a competitive issue and one of strategic survival, Ashton [3], Deming [4].

Despite a seeming wealth of literature on software quality, quality software is still quite rare, Price Waterhouse [5]. We see that quality considerations for software have not kept up with progress in other fields through the years; products are often late, inefficient, not to specification, expensive or unsatisfactory to users, Jones [6]. Fenton [7] even suggests that software 'engineering' is little more than an unrealised ideal, more of a craft really. Whereas Koch [8] indicates that others have mentioned the software crisis may never end, and that a 'monster' has to be killed in software engineering. In the 1980's the solution seemed to lie with technology. However it certainly is the case that many technologies have been disappointing in the last decade or so (e.g. CASE tools, 4GLs, reusable code, graphical user interfaces, design methods, etc.). Why is it the case that these are unused (become "shelfware", Yourdon [9]) or fail to deliver properly? Curtis [10], who was the head of the SEI software process initiative when it began, attributes this failure of technology to an adequate **software process** not being in place.

The purpose of this paper in due course is to provide a balanced view of **software process improvement** and its relationship to software quality. Two main topics are explored. Firstly attention to correct management practices, and secondly software process engineering. The remainder of this paper then examines the possible requirement for and usefulness of these, as stated offering a fair-minded perspective.

THE FIRST STEP ON THE ROAD TOWARD SOFTWARE QUALITY ?

A successful **Quality Management System(QMS)** is an instrumental facilitator within an organisation that directs and balances the functions required to achieve the specified quality of the product or service as cost-effectively as possible, whilst ensuring quality information. Professional help is widely available for QMS installation.

This section looks at the installation of quality management practices for software development as a possible first step in the direction of improved quality software. Primarily a UK emphasis with details of the only certification scheme available and promoted for this purpose, called TickIT [11]. Some description of this scheme is given, together with consideration of perceived positive aspects and limitations, and also the question of sufficiency of this approach is addressed. (Sufficiency of TickIT and a QMS approach in general for leading to improved software quality).

Description of the TickIT scheme

Foundation The internationally accepted requirements for quality management systems are defined in the ISO 9000 series of standards. ISO 9001, and harmonised in the EC through an equivalent European standard, EN 29001, is interpreted for software through a quality management initiative for software quality. TickIT encapsulates the ISO 9000-3 guidelines for the application of ISO 9001 to the development, supply and maintenance of software.

Scope The intended aim of TickIT is to achieve improvements in the quality of software products and information systems throughout the whole field of IT supply including in-house development work. TickIT attempts to define best practice for software production, Dennis [12].

Coverage TickIT is largely a model of the requirements for a QMS plus guidance. The key sections place an emphasis on organisation, practices and verification, covering twenty processes. To achieve a Pass, all twenty need to be present to the satisfaction of the auditor. There is a three year re-certification period. In the UK, TickIT is recognised as the only form of accredited certification for information systems and the certificate is recognised by all Government departments and major purchasers [TickIT(a) [13]].

Reception TickIT began in 1990. Through rapid promotion, industry awareness of TickIT aims has been raised. The impact of TickIT is growing, and the uptake more than doubling each year with over two hundred organisations expected to seek TickIT certification in 1993, Fletcher(GQCN) [14]. In the final months of the recent DTi/TickIT awareness campaign, around 1400 people representing virtually every private and public sector interest participated in conferences and workshops, TickIT News [15]. Originally a UK-DTi and industry initiative, since May 1993 TickIT has been run by BSi / DISC in the UK.

Transactions on Information and Communications Technologies vol 8, © 1994 WIT Press, www.witpress.com, ISSN 1743-3517

Perceived Positive Aspects and Limitations of TickIT

Positive Aspects The authors have been involved in advising and training developers on quality issues for a number of years. Whilst hard evidence or directly measured proof of the efficacy of a QMS approach is scarce, as it does take a while to reap benefits, we are seeing a trend that TickIT certification provides a number of benefits. The scheme is worthy in a number of respects as it:

- is underpinned by existing international standards and interprets these adequately
- has a good pedigree and is well controlled/overseen
- is seen as an investment to reduce business risk in the future
- has been seen by the some customers to improve software quality in certain cases
- promotes a national focus on software quality
- (such certification) can enhance trade and confidence and is being demanded by more purchasers

However we have noted the following concerns:

- **No Guarantee** - poor quality products can still be produced despite such certification, it is thus no guarantee of resultant quality
- **Motives and Commitment** - on occasion it can be undertaken for the wrong reasons, e.g. just to pass, internal effort can be under-estimated (effort spent by company staff in preparation and assisting assessment is typically 30-50 man days), and the total cost is not cheap either, being about a half to one man year just to develop the system.
- **Audit Reliability and Correctness** - difficulty in finding an auditor qualified/experienced in a specific business area, also an effective process may be altered to meet the requirements of TickIT certification, an effective process may not be certifiable, and an immature process may obtain TickIT certification
- **Adequacy of Emphasis** - TickIT emphasises conformance, not effectiveness nor does it give adequate emphasis on technical and people factors. Documentation does not necessarily imply understanding or commitment, just because it is written down
- **Circumstantial Relevance** - there is little consideration of application/domain differences in software design and development and perhaps not enough attention to the fact that quality management systems must be built to fit individual business needs. Relating to this, a recent report on best quality management practices and performance, Best Practices Report by Ernst and Young [16] forwards some interesting ideas based on extensive practical real world investigation and analysis of data. This report indicates that performance level dictates best quality

management practices to adopt in a number of industries including computing. Perhaps organisations should adopt management practices according to focus, market and competitive position.

Despite these apparent criticisms, the authors consider TickIT to be a worthwhile contribution in the pursuit of improved quality software. Proper certification of software management capability of suppliers/developers is essential to raise the overall standard of the software industry. It seems likely that in future purchasers will increasingly demand a proper third party certified QMS in design / development of software, Slater [1]. It may become widely compulsory in contracts in due course. Overall, TickIT is an accessible vehicle for understanding and implementing ISO 9001 in software development. TickIT possesses a narrow and limited software process view/flavour and can be looked on as a useful first foot-hold, a beginning, despite its imperfections.

Necessary but Not Sufficient?

We have argued that a good QMS properly installed and maintained has the potential to enhance such things as communication, control, efficiency, utilisation of resources - but cannot always guarantee that a superior standard of software is produced. It is important to understand that there is a gap between using a QMS for software and the repeated production of quality software. QMSs are general guidelines and definitions only which help to ensure that all reasonable steps have been taken to encourage quality and trace documents. QMSs apply to the whole organisation and they cannot define the Process for a particular software development though they may constrain it, Rae et al. [17]. In fact Paulk [18] has recently shown that a certified company can still be in an ad hoc and chaotic state. This viewpoint is put forth nicely as follows - in relation to software, it is no use having a QMS if the software process is no use, as the QMS will fail due to poor commitments, similarly good engineering is at risk if poor management exists, Curtis [10]. The QMS approach can be viewed (according to this argument) as necessary but not sufficient to lead to repeatable high levels of software quality. TickIT for instance is primarily aimed at the quality management domain and not at software engineering technical practices per se. Although TickIT emphasises documentation and Quality System control in software engineering, it does not adequately address the issue of continuous improvement of the software process once the QMS is installed and operational.

THE SOFTWARE PROCESS VIEW - A COMPLEMENTARY FOCUS ON SOFTWARE PROCESS IMPROVEMENT EXAMINED

This brings us to consideration of software process engineering. At the International Standards level, with the ISO-SPICE [19] (Software Process

Improvement and Capability dEtermination) initiative, there is a focus to complement the certification to ISO 9000-3, with enhancement of the software process via its evaluation, toward improved software quality. This initiative is promoted as acting for continuous process improvement and controlled change management aligned to business needs. Certification to TickIT is thus considered within an intended overall improvement programme to include the monitoring of adherence to pre-defined Processes. A number of aspects of software process engineering will now be examined.

Within the last seven years there has been a marked increase in attention to the efficacy of the software process and its improvement with the possible relation to resultant software quality, [e.g. Humphrey [20]; SEI-CMM [21,22]; Curtis et al. [23]; IEEE [24]; IS&T [25]]. The quality of the software construction process and the development of proper techniques to improve such quality through analysis, control and improvement are now clearly seen by many as vital considerations in the pursuit of superior software quality. Perhaps the answer to poor software quality lies in what was suggested a number of years ago by Miller [26], "the ability of the software industry to deliver high quality products and systems is dependent on the quality of the processes used to support their development" and Lysy [27], "the achievement of software product quality is based on the quality of the engineering process used to develop the product, and quality can be better ensured by building the quality requirements into the engineering process", through to: "the quality of a software system is governed by the quality of the process used to develop it", Humphrey [20]. Curtis [10] may also be right in stating that "in attempting to improve software, the software process is the right focus".

There have been a number of papers produced, conferences organised, techniques developed and things said, as above, in favour of this focus on the software process. But a number of questions remain unanswered. How much of this is rhetoric and promotion and how much is factual and of concrete proved benefit? What evidence is there that it makes a difference? When does change equal improvement? How mature are the approaches to software process improvement? What standardisation exists? What unresolved research issues are there? What can people reasonably claim? With what certainty can it be stated that software process improvement is the bow for the development arrow to hit software engineering targets?

This section looks at software process improvement as a possible complementary step in the direction of improved quality software. Aspects of main approaches to software process improvement are discussed from a maturity viewpoint (their value and applicability, currently and in the future). In particular the relevance of **software process assessment** is explored and evaluated. The import of utilising software process models and related

measurement is briefly discussed. The question of readiness of software producers in terms of knowledge and current capability of their software production practices, to make effective use of process improvement techniques is also addressed.

Outline of Main Approaches to Software Process Improvement plus Definitions

There have been a number of worthwhile approaches to software process improvement suggested, however most attention has been aimed at software process assessment and software process modelling approaches. Within these areas there are a number of schemes and techniques, some valuable theory, research work and some findings have emerged, but firstly some definitions are required,

- Software Process is defined Curtis [10], as the set of activities, methods, practices and transformations that integrate managers and software engineers in using technology to develop and maintain software. The software process can be defined differently at different levels, e.g. the software process in industry, in an organisation, in a project, or at sub-process level.

- Process Improvement is defined [ISO/IEC [28]], as the operation of putting in place measures to strengthen processes which have been identified as sources of defects or risks to quality, cost or schedule performance. Process improvement is based on the premise that product quality is highly dependent upon the processes used in its creation.

- Process Assessment is defined [ISO/IEC [28]], as the disciplined examination of the processes used by an organisation against a set of criteria to determine the ability of those processes to perform within quality, cost and schedule goals.

- Process Capability is emerging as a utilisation of the results from assessment for external consumption.

- Process Modelling is a term used several times in this paper, is not a new activity. Business process modelling for instance has been around for a number of years, refer to Snowdon [29] for an analysis. With the current tidal wave of interest toward quality in software, software process modelling can potentially aid the software developer by allowing an operational definition and documentation of project purpose and action for process re-use, communication, project management, and process improvement. Whilst not a complete definition, where the term is mentioned in this paper this is the intended meaning.

Description of Software Process Assessment techniques

Introduction to Software Process Assessment - foundation, scope, coverage and reception The software process assessment approach to software process improvement has captured most attention recently. In particular the valuable pioneering work at SEI in North America (e.g. SEI-CMM [22]) on assessment techniques for capability. There are several variations of this progressive framework; and also the on-going state-of-the-art ISO standardisation effort [ISO-SPICE [19]], mentioned already and which will be described and discussed in due course.

It is becoming recognised that software process assessment techniques are an important consideration at least, some think [e.g. ISO/IEC [28]; SEI-CMM [21,22]; ISO-SPICE [19]] instrumental in achieving software process improvement and capability determination. Such evaluation of the software process is a possible approach to help developers produce better software and also to help determine which developers are the most capable for a particular purpose, latterly with the intention of reducing risk with an emphasis on process fitness. But how does this work?

Software process assessment concerns the systematic examination of how a company develops software and provides services. The approach looks at how a developer operates (its processes) and not primarily its end-products. The idea is that providing software companies / developers with a proper way to rate their process of developing software enables them to gain a credible idea of their state of practice and thus indulge in continual process improvement for better quality end-products and services; and in addition process evaluation for capability purposes to provide developers a means of proving their fitness for a particular named development purpose. Not all software process assessment schemes have the dual purpose, their emphasis varies, as will be described in due course. As an approach to software process improvement, assessment apparently leads to the identification of key areas for process improvement and provides the input into process improvement action plans. Ideally it is envisaged by proponents that if processes are defined, understood, and controlled then the end-product is more likely to meet its quality targets. The focus is on prevention not detection, the intention of getting the processes correct to hopefully bring about correct product, correct service. Taking this notion to its ultimate conclusion, if correct the need for product inspection / metrics may be minimised in due course. That is the ultimate potential benefit to be derived from software process assessment, built-in assurance of end-product quality target actualisation, higher quality products at a reduced cost (in the interest of all) rather than current low quality products at a high cost.

On reception, both interest in and activity surrounding software process assessment has grown well since 1987. A number of schemes have emerged

and it has been taken up by a number of large organisations. International standardisation is currently being developed and is not yet available.

A Brief Overview of Major Existing Software Process Assessment Schemes (maturity focus) There are a number of existing software process assessment schemes in use around the world. How have different schemes arisen? Which are considered to be the main ones? Are any adequately mature and suitable to be considered as a standard?

It is the case that existing software process assessment schemes have emerged from the specific needs of organisations or market sectors. Specific needs have shaped their creation and subsequent evolution, availability and use, according to ImproveIT [30] [to date the most comprehensive survey of such schemes looking at more than twenty according to a number of relevant criteria].

The follow up report, ISO/IEC [28], which refines and articulates the need and requirements for an ultimate software process assessment method and associated standard, considers six existing schemes to be deserving of special attention. These are: Software Technology Diagnostic(STD) [31] by Scottish Enterprise/Compita Ltd.(UK); Software Quality and Productivity Analysis(SQPA) [32] by Hewlett Packard/Capers Jones(USA); Bootstrap [33], originally an ESPRITII project (Europe); SEI-Capability Maturity Model(SEI-CMM) [21] by SEI(USA), TRILLIUM [34] by Bell Canada, SAM [35] by B.T. plc (UK). So far the SEI-CMM is recognised as the prime effort, with most of the others being derived in some way from it.

There is an important question of the overall maturity of these individual schemes and the ability of any to satisfy requirements to merit installation as a single world standard. There are a number of requirements and criteria which candidate schemes would have to fulfil, for instance ability to provide capability and improvement data; flexibility (there are a number of dimensions to this criterion e.g. flexibility on application types, sectors, project size, organisation size, business needs; cultural independence; adequate availability. These are only a few of many. According to ISO/IEC [28] none of these (the six named above) methods has all of the required features and characteristics to enable it to be adopted as the new standard, but each has features that could be included. (This does not mean to imply that each does not perform adequately in the environment it was developed for and evolved in). Dorling [36] states that the standards effort will build on the best features of the major existing software assessment methods, SEI-CMM [21,22], TRILLIUM [34], Bootstrap [33], STD [31] and bring about a common standard to provide comparable results.

It is intended not to comprehensively compare schemes nor exhaustively describe their individual coverage and inadequacies. The summary fact is that there exists a wide diversity of evolution and range of foci in existing assessment schemes and varying degrees of immaturity are displayed in relation to adequacy as a standard.

Standardisation Effort International Standardisation in the area of software process assessment is an on-going, incomplete but promising undertaking. The leading edge is identified as a special project initiative already briefly mentioned and called SPICE under the auspices of a relevant ISO Working Group, ISO/IEC JTC1/SC7/WG10 (divided into three world areas via a management board), ISO-SPICE [19]. The initiative is based on previous study articles, [ImproveIT [30]; ISO/IEC [28]; which identified the need and requirements for a common acceptable process assessment method and associated standard]. The international ISO/IEC [28] study put forward a recommendation that a universal technological standard for software process assessment should be developed and be available by the mid-nineties. There is an unusual fast-track completion requirement of two to three years instead of the usual ten. This rapid development indicating according to Peltu [37] from Dorling of Brameur Ltd. that there is an increasing priority being given to software quality issues throughout the world and there is a growing realisation that persistent software development problems can be eradicated only by systematically assessing, measuring, and improving underlying processes.

To its credit SPICE is attempting to draw knowledge, information and expertise from representative supplier and purchaser organisations, as well as academia, trade associations, national standards bodies, government agencies, and developers of assessment schemes (who were identified in the ISO/IEC [28] study, as main contenders for consideration). It is hoped that it will be possible to bring about a shared, public-domain, state of the art, wholly and continually relevant assessment standard, fulfilling all requirements while supporting existing standards, and showing cultural independence.

There are four main areas to be looked at by the initiative and these are -

- development of a universal set of steps for process assessment;
- ascertaining and defining best practices against which each process is to be assessed;
- an accepted software process model;
- the specification of a consistent method to rate processes.

The process definitions / best practice definitions will cover technology, people, and processes in software. Standardisation of measurement for conformance and effectiveness is very much on the agenda.

The SPICE key-word for assessment focus is 'Organisational Unit'. This describes the department, project(s), section, division etc being assessed. The SPICE work will be founded on a Common Process Model and this model will support the use of an assessment sensor to gather the data as well as underpinning evaluation and profiling. The standard will consider two modes od application post-assessment - that of process improvement and capability determination, with the aim that the assessment is carried out the same way for each mode. SPICE will rate and assess individual processes and produce profiles accordingly. Process categories to be covered are those for procurement, development, delivery, operation, support, planning, management, control, and improvement. Tools will be provided to profile processes and facilitate display. Full documentation will also be forthcoming.

With a proposal date of 2/93, following on are periods of development, trial and awareness, with issue planned for 6/95, and a full standard in place by 1996 approximately, Dorling [38]. Such a standard, it is envisaged, on being successful will lead to increased global trade-flow. The success of SPICE will of course depend on expertise and funding, awareness, government approval, barrier removal and speed [Dorling [38]], as well as co-ordination and communication once available to institutionalise it in the marketplace. The SPICE standard will move away from process certification, and more toward process improvement in line with business needs. SPICE is certainly one of the few co-ordinated initiatives in the software process for quality area. It will be very interesting to see what emerges.

Note there is a newly formed European Software Institute in Bilbao which intends to rapidly improve levels of software engineering and afford a competitive advantage for members whilst delivering:

- a way for companies to make individual and consistent comparative evaluations of their software engineering process
- independent and active support for company programmes to improve capability
- available collective experiences of top practitioners

Making good use of software process assessment techniques and promoting continuous improvement are very much on the agenda.

Perceived Positive Aspects and Limitations of Software Process Assessment

Positive Aspects Several benefits to developers and purchasers are promoted by proponents of the assessment approach -

- built-in assurance of meeting the end-product quality targets
- higher quality products at a reduced cost
- identification and bringing into line of high risk processes, encouragement toward the engineering of processes to meet business requirements
- reductions of development times and cost leading to increased profits and return on investment

- reduction in insurance premiums and maintenance labour on fixed price contracts
- developers who can prove the efficacy of their software development process are afforded a competitive advantage
- it enables the purchaser to become more discerning and demanding
- optimisation of resources
- an environment of constant improvement

Concerns We have noted the following concerns -

- uncertainty on standardisation leading to optimal and lasting software process improvement
- a number of unresolved research issues
- a number of possible inadequacies
- much is promotion and conjecture still, much remains unproved.
- generality v. specificity
- complexity handling, accuracy, and reliability
- general immaturity as a technological discipline and possible premature standardisation
- unanswered questions on related technological disciplines, modelling and measurement
- the ability of standardisation to deal fully with effectiveness issues
- the meeting of requirements and precursors to success by standardisation
- the task of convincing developers in particular of its worth, as they pay for it

The following section will provide further explanation of some of these concerns.

Evaluation of Software Process Assessment and also briefly Related Techniques, for Software Process Improvement The effect of software process assessment should be very visible over the coming years. Prominently ISO 9000-3 (TickIT in UK) will be complemented by external software process assessment standardisation described and concerned with evaluation and ultimately continual improvement of the process. Usefully this can be thought on as a circle within a circle, ISO 9000-3/TickIT as the inner circle and software process assessment standardisation and associated process improvement techniques as the surrounding outer circle. Although initiated after ISO 9000-3, the complementary software process assessment standardisation and associated techniques, if successful, seem likely to grow to be far more important and influential than TickIT or other initiatives. Whether it will actually lead to optimal and lasting software process improvement and in turn optimal software quality as promoted is currently not certain. There are a number of unresolved research issues and considerations including possible inadequacies to be discussed.

Much is promotion and conjecture still, much remains unproved. It is very difficult to prove that software process improvement has occurred at all by any approach, proof by measurement is very difficult, Hersh [40], and how could it be proved that it directly leads to quality improvements? When is process change equal to process improvement? Measuring the value of process improvement is difficult in terms of resultant quality and productivity, customer satisfaction etc. and defining Return On Investment in relation to software engineering and technology is not agreed, to prove improvement never mind justify it. To prove the worth of software process improvement and indeed approaches to it, proof by measurement must be forthcoming and currently it is not, this must be worked on. As Curtis [10] states, process improvement makes a difference but no proper validation studies to prove it - providing evidence that it makes a difference then is problematic.

Some would accuse questionnaire-based software process assessment of being too general and not specific enough to bring about optimal and lasting software process improvement. The most advanced scheme currently, SEI-CMM [22], concerns What not How and employs a process maturity model which is general only. In-depth questions of measurement and software process modelling for process improvement need to be addressed at upper levels - not so at present. For instance exactly which specific advanced measures to use where and when?, which software process modelling techniques are best under which situations to bring about optimal and lasting software process improvement? Such questions are probably unanswerable just now by anyone, standardisation in these matters does not exist. Software measurement is a minefield (Fenton [7]) and software process modelling is young and the research range is still being drawn up, Curtis et al. [23]. Integration of modelling and measurement may be the way forward in process improvement, Rombach [41], however it is more of an idea than a reality at present. Several questions on relevant measurement and modelling employment remain unanswered. More attention to these approaches will surely bear fruit. The SEI-CMM [22] authors are aware of the possible worth of software process modelling and measurement for software process improvement, Curtis et al. [23] stating it is essential at upper levels in assessment. By saying more on How at upper levels not just What activities should be assessed for without comment on best methods, would be more convincing that optimal and lasting software process improvement is a possibility directly by using such schemes. As stated however the new ISO-SPICE [19] initiative is usefully attempting to standardise measurement categories.

The complexity of the software process at a stage where software process improvement is appropriate, having a defined and fully visible process onwards, is great, with multifarious feedback loops etc.. Some would

question that using rigid questionnaires and interviews and short study of documents often by novices trained up in a day, is a valid method to bring about optimal and lasting software process improvement, in terms of the said complexity handling. The long-term use of questionnaire-based assessment schemes is dubious, direct measurement and forms of automation may be better.

There are other approaches to software process improvement which may handle complexity of the process better, e.g. direct comparison of ideal and current software process models to recognise and bring about improvement, Hinley and Bennet [42]. Software process improvement being actually tackled at the modelling and measurement level, so not only the traditional questionnaire-based software process assessment approaches to software process improvement exist.

No software process assessment model is anywhere near perfect currently, none is good enough, and very little is proved in relation to software process improvement. Such ideas need to be kept in mind on hearing promotions of software process assessment as the approach to software process improvement and the key to increased software quality. No-one currently has the proof that their approach leads to maximal software process improvement on a continuing basis. No-one can reasonably claim so.

Gray and Hunter [43] indicate that research issues must be understood before a standard comes into existence i.e. standard follows technology. Developing standards prematurely is a concern, if they require constant changing they may not be used or respected. There are a number of unresolved research issues pertaining to software process assessment. They outline a number of these. Some which could be added to theirs are:

- how repeatable/reproducible are process assessments, inter- and intra-auditor reliability?
- how can maximal and lasting process improvement via assessment be proved by measurement?
- software products and processes are special but in what way and to what extent?
- goal capability of a project or organisation - how does one know that an organisation is capable of achieving its improvement goals recommended?, the link between assessment and goal achieving ability needs to be formalised.
- is software process assessment leading straight to process improvement from sensors and interviews etc. reliable for example. in handling complexity?

- to what extent does software process assessment constrain the process without properly defining it for a particular development? Are they too general / not specific enough?
- to what extent are software developers at large knowledgeable and ready in terms of visibility/health of their process to receive improvement approaches and be amenable to software process improvement at all?

The software process assessment approach in general as is, leading to process improvement may not be wholly satisfactory, in particular in terms of accuracy and complexity handling, also reliability factors.

Software process standards constrain the process which is used to develop and maintain software by defining characteristics/properties which a conforming process must possess. Ideally the end result is quality. Mandatory compliance to standards within contracts is probably just around the corner in software, unresolved research issues and unanswered questions indicative of immaturity make this a concern in general.

The Question of Readiness to Make Effective Use of Process Improvement Techniques Addressed

There is an important question mentioned above of the readiness of software producers in terms of knowledge and current capability of their software production practices, to make effective use of process improvement techniques in any case. Process improvement by definition would require a completely visible, wholly defined software process, where measures are interpretable (free from noise interference at project level) with the possible employment of process modelling techniques for process support and improvement. This is currently the case only in a small percentage of developers.

A recent figure on the maturity(richness) of companies software processes, Yourdon [44], states that around 75% of software projects are at level 1, (SEI process maturity scale). This figure rises to 81% for sites and 88% for projects if one believes a later estimate, Yourdon [9], for USA. At the ISE Belfast, Peltu [37], reports more than 85% of their assessed UK organisations were at Level 1(chaos). Thompson [45] [who worked at ISE, Belfast] in fact states the figure is 95% for Information Systems developers versus 85% for technical developers at ad-hoc Level 1. Dorling [38] states the latest figures from the SEI in the USA on maturity are 85% of organisations high risk at Level 1(initial), 14% at Level 2(repeatable), and 1% at Level 3 (defined process). Whatever the exact figures, it is clear that currently the readiness for software process improvement efforts is present only in a handful of companies, because their software process health is so bad (that is if they have any development process at all). In order to be ready

for such improvement efforts, one would require to have a visible and defined software construction process in place as stated.

Many developers are apparently still unaware that there is such a thing as a software process. Rubin [46] in a survey of conference delegates in USA found only two percent were actively doing something about the SEI process maturity model, with some viewing it as an inhibitor of progress or not applicable to them. Rubin [47,46] has a number of interesting ideas on readiness and poor technology transfer. Readiness for software process improvement is very poor in the vast majority of organisations, there is in general a lack of process knowledge, skills and commitment.

Hopefully this state of affairs will change in coming years. It will need to change if they are to take advantage of potential future advancements in this area. It is equally clear that currently the priority for the vast majority of software developers is attention to the correct elementary management practices, having these installed and coming to the position where software process improvement perhaps starts to be a possibility for them. Where a developer has a poor process, it is no use investing in advanced measurement, money is better spent on installing basic management practices, Best Practices Report [16]. Hopefully in parallel advances in maturity of approaches to software process improvement which themselves are currently evolving: assessment, modelling and measurement techniques, will take place and will can offer provable benefits. The knowledge of process issues within academia and specialist research centres and agencies is good but not complete. Process improvement approaches are not yet mature sciences, and cannot currently offer complete solutions nor proof. Practitioners themselves can feasibly contribute to knowledge of process improvement. There is rightfully expectation in this area.

At present time there is no way of building software with a specified degree of quality nor of satisfactorily measuring the quality of software once developed, Hunter and Lloyd [48]. Improving process quality and associated factors such as change management, skills and technology use may in due course increase the quality of software produced; risk reduction, design methods and domain knowledge may be important as well. The road to superior software quality is long. Proof is essential.

Controlled Implementation

On undertaking improvement, one must consider the pace, degree and breadth of change, and also the things that get rewarded at the right levels get done, Best Practices Report [16]. Executing change correctly is vital.

Cultural change is vital, the "Kaizen" approach to company management and change (Huda and Preston [49]) may be useful. Kaizen is a philosophy of

controlled change in small incremental improvements using consensus decision making. Changes are small enough not to disrupt the environment but on accumulation lead to evolutionary change, fully considering human factors. Business culture change is difficult in the West, some features of Kaizen may be applicable to facilitate change/improvement. This view of a little but often (not too much change at once) is also supported by Pressman [50]. It seems highly feasible that when business culture change/widespread continuous and controlled process improvement and a good QM system are present, then TQM ideals will evolve for business-wide improvement, such overall improvement will be a possibility.

CONCLUSIONS

The two main conclusions are as follows. Firstly, software process improvement is not yet proved to be the route to superior software quality, it remains a possible route though (software process improvement approaches considered though evolving are not yet mature sciences, and cannot currently offer complete solutions nor proof, in particular there are significant unresolved research issues and possible inadequacies associated with software process assessment). Secondly as a very high percentage of software developers have poor knowledge of process issues and no visible software process anyway, it is concluded that their best action is to seek to install appropriate elementary management practices in order to achieve a state where software process improvement may become applicable to a visible process, should provable benefits be forthcoming. There is no panacea waiting just around the corner.

ACKNOWLEDGEMENT

We would like to acknowledge the support of D. Murray, Glasgow Caledonian University Computer Studies Department Head, for kindly supplying sponsorship of this paper from Departmental funds.

REFERENCES

1. Slater, J., 'The TickIT scheme', paper at conference, *TickIT- making a better job of software,* Glasgow, UK, Dec. 1992 (organised by I.T. World, sponsored by DTi, supported by SSF).
2. Hollocker, C.P., 'Finding the Cost of Software Quality', *IEEE Trans. Eng. Man.* (USA), Vol. EM-33, No. 4, pp. 223-228,Nov. 1986.
3. Ashton, G., 'The Role of Certification Bodies', paper at conference, *TickIT- making a better job of software,* Glasgow, UK, Dec. 1992 (organised by I.T. World, sponsored by DTi, supported by SSF).
4. Deming, W.E., *Out of the Crisis*, Cambridge University Press, 1982.
5. Price Waterhouse, *Software Quality Standards, the costs and benefits report,* Report to UK/DTi, April 1988.
6. Jones G.W., *Software Engineering*, J. Wiley and sons, 1990.
7. Fenton N.E., *Software Metrics - A Rigorous Approach.*, Chapman and Hall, 1991.

8. Koch , G., 'Process assessment : the Bootstrap approach' paper at conference *Software Process Modelling in Practice,*Kensington, London, UK, 22-23 April, 1993 (Butterworth-Heinemann conference).
9. Yourdon, E., *Decline and Fall of the American Programmer,* Yourdon Press, 1992.
10. Curtis, Prof. W., 'Software Process Improvement' and 'The Superior Software Organisation' papers at conference *Software Process Modelling in Practice,*Kensington, London, UK, 22-23 April, 1993 (Butterworth-Heinemann conference).
11. TickIT, *The TickIT Guide to Software Quality Management System Construction and Certification using EN 29001,* (The TickIT Guide) versions 1990, 1992, (available from DISC TickIT Office, London, UK).
12. Dennis, M., 'Background and Aims of TickIT', paper at conference, *TickIT- making a better job of software,* Glasgow, UK, Dec. 1992 (organised by I.T. World, sponsored by DTi, supported by SSF).
13. TickIT(a), 'TickIT: making a better job of software', part of conference literature, *TickIT- making a better job of software,*Glasgow, UK, Dec. 1992 (organised by I.T. World, sponsored by DTi, supported by SSF).
14. Fletcher, R.I., (GQCN)*Grampian Quality Club Notes,* unpublished, Glasgow Caledonian University, 1993.
15. TickIT News, 'TickIT's future with DISC', *TickIT News,* Issue 4, Nov.1993.
16. Best Practices Report, based on the International Quality Study, Ernst and Young / American Quality Foundation, so far unpublished as of July 1993.
17. Rae, A.K., Hunter, R.B., Kirkwood, K.B., *Beyond Quality Management Systems.* Strathclyde University Computer Science Research Report SQ-1-90, Glasgow, UK, June 1990. Also presented at the Workshop on Software Quality Assurance, Dundee, 1990.
18. Paulk, M.C., *Mapping fron ISO 9001 to the CMM,* SEI Memorandum June 1993.
19. ISO-SPICE (Software Process Improvement and Capability dEtermination), Special initiative for Software Process Assessment Standardisation, ISO/IEC JTC1/SC7/WG10, 1993-96.
20. Humphrey, W.S., *Managing the Software Process,* Addison Wesley, 1989. (SEI series on Software Engineering).
21. SEI-CMM(1991), Software Engineering Institute Capability Maturity Model. *Capability Maturity Model for Software,* M.C. Paulk, B. Curtis, M.B. Chrissis. CMU/SEI-91-TR-24, August 1991.
22. SEI-CMM(1993), Software Engineering Institute Capability Maturity Model. *Capability Maturity Model for Software, version 1.1,* Paulk, M.C., Curtis W., Chrissis, M.B., Weber C.V., CMU/SEI-93-TR-24, February 1993.
23. Curtis, W., Kellner, M.I., Over, J., 'Process Modelling', *Communications of the ACM,* Vol.35, No.9, pp75-90, September 1992.
24. IEEE, IEEE Software Journal special edition on the Process Maturity Movement, July 1993.
25. IS&T, Information and Software Technology Journal special edition on Software Process Modelling in Practice, Vol. 35, No. 6/7, June/July 1993.
26. Miller, C., 'BS 5750: standard for software quality assurance', paper at conference, *Software Quality Assurance Reliability and Testing,,* London, UK, 9-10 Dec. 1986., pp.17-23. (Uxbridge,UK : Unicom Seminars 1986).
27. Lysy, K.A., 'Software Quality Engineering and Structured Methods', *Proc. COMPSAC'87 - The 11th Annual Int. Comp Software and Applications conference,* Tokyo 7-9 Oct. 1987, pp. 103-9. (Washington D.C.: IEEE Comput. Soc. Press. 1987).
28. ISO/IEC, ISO/IEC JTC1/SC7 N944R, The Need and Requirements for a Software Process Assessment Standard, Study Report, Issue 2.0 (June 1992).
29. Snowdon, R.A., Software Process Modelling - technical aspects, paper at conference *Software Process Modelling in Practice,*Kensington, London, UK, 22-23 April, 1993 (Butterworth-Heinemann conference).
30. ImprovelT, ISO/IEC JTC1/SC7 N865, Issue 1.0a, June 1991.

31. STD, Software Technology Diagnostic, by Scottish Enterprise/Compita Ltd., v3, 1993.
32. SQPA, Software Quality and Productivity Analysis by Hewlett Packard/Capers Jones (USA).
33. Bootstrap, Commission of the European Communities, ESPRIT project 5441 BOOTSTRAP.
34. TRILLIUM, Telecom Software Product Development Capability Assessment Methodology, by Bell Canada, Draft 2.2.1, Nov. 1992.
35. SAM, SAM assessment methodology by B.T. plc (UK).
36. Dorling, A., (b), 'SPICE: Software Process Improvement and Capability dEtermination', *Information and Software Technology,* Volume 35, Number 6/7, June/July 1993.
37. Peltu, M., 'Project v Process Management', (perspective/software development), *Integration,* Sept.1992.
38. Dorling, A., 'Process Assessment Standards', paper at conference *Software Process Modelling in Practice,* Kensington, London, UK, 22-23 April, 1993 (Butterworth-Heinemann conference).
39. Fletcher, R.I., *Quality - The SPICE of Life - a report on the ISO SPICE project on software assessment,* Glasgow Caledonian University, UK, Unpublished, Feb. 1993.
40. Hersh, A., 'Where's the Return on Process Improvement', *IEEE Software*, p12, July 1993.
41. Rombach, H.D., 'Process Modelling and Metrics', paper at conference *Software Process Modelling in Practice,*Kensington, London, UK, 22-23 April, 1993 (Butterworth-Heinemann conference).
42. Hinley, D.S., Bennet, K.H., 'A process modelling approach to managing software process improvement.', paper at conference, *SQM'93,* Southampton, UK, 1993. In Proceedings [A BCS/Wessex Institute of Technology conference].
43. Gray, E.M., Hunter, R.B., 'Process Assessment and Process Improvement - the need to standardise?', paper at conference, *SQM'93,* Southampton, UK, 1993. In Proceedings [A BCS/Wessex Institute of Technology conference].
44. Yourdon, E., 'An Interview with Watts Humphrey.' *American Programmer,* Sept. 1990.
45. Thompson, K., 'Software Process Maturity and the Information Systems Developer', paper at conference *Software Process Modelling in Practice,* Kensington, London, UK, 22-23 April, 1993 (Butterworth-Heinemann conference).
46. Rubin, H., 'Software Process Maturity', *American Programmer*, January 1991.
47. Rubin, H., 'How to become a software engineering 'bigfoot'', *American Programmer*, January 1990.
48. Hunter, R., LLoyd, I., *Legal Liability and the State of the Art in Software Engineering.* Strathclyde University Computer Science Research Report SQ-4-93, Glasgow, UK, February 1993. Previously presented at the European Conference on Software Quality, Oslo, 1990.
49. Huda, F., Preston, D., 'Kaizen: the applicability of Japanese techniques to IT', *Software Quality Journal,* Vol. 1, No. 1, pp9-26, Mar. 1992.
50. Pressman, R.S., *Making Software Engineering Happen.* Prentice-Hall, 1988.