Rapid Process Improvement

-

SPI 2000, Göteborg

Clifford Shelley & Ian Seward

Oxford Software Engineering
9 Spinners Court, 53 West End, Witney, Oxfordshire, England, OX28 1NH

www.osel.co.uk/rpi/rpi.htm
shelley@osel.netkonect.co.uk

ABSTRACT

The business environment within which software development and process improvement is undertaken has changed to become fast-moving and unpredictable. Software development practices are changing to meet changing business needs; process improvement practice must also change to meet the need of this new environment. A revised approach to software process improvement, called Rapid Process Improvement, is described and tools to support this business-oriented approach are discussed.

1. Introduction

The business environment has changed over the last ten or fifteen years to become much more dynamic and changeable. Software development activities have also changed with increased pressure to deliver rapidly and with the introduction of many new technologies and tools. Software Process Improvement activities need to reflect this new environment by reacting and delivering tangible benefits rapidly. A more appropriate approach to process improvement, Rapid Process Improvement, has been developed. It is intended to deliver: results rapidly and cheaply allowing benefits to be accrued as soon as possible; learning from mistakes without incurring major costs; and process improvements to respond to changes in the business environment. This approach is supported by a developing set of tools for process improvement. Some of these tools are well known in the software industry others, are new.

2. The Business Environment

Software development and support is conducted in enormously varied environments, from major systems development work by large defence contractors to small commercial companies developing their own products. One of the most striking common characteristics of these environments is their dynamic, or perhaps more accurately, unstable nature. The great majority of organizations will encounter change, either imposed from outside or instigated internally. Change may arise from:

- economic , currency or market fluctuations
- mergers and re-organizations
- changes in business strategy
- changes of management
- internally generated 'initiatives'.

For most organizations this tendency towards uncontrolled change is a fact of business life. A look-ahead of more than eighteen months to two years is of limited value for Software Process Improvement (SPI) because it is rarely clear what will be required then. Within this changing environment one of the key attributes of software development is now timing – not just increased development speed but sequencing and co-ordination.

The software development and support activities that take place in the business environment are now being characterized as just another set of business processes where once they were seen as specialist activities. It is also evident that the software processes are not clearly distinguished from other business processes and consequently have similar expectations placed on them. It is expected (but rarely true) that software processes are robust enough to survive: mergers and take-overs; new business; restructuring; and be available when required. However software processes and capability are rarely managed as valued assets and are often discarded or 'mislaid'.

To a large extent the failure of businesses to appreciate and manage their process assets is due to their inability to characterize them. Often what is labeled as 'software development' is either; enhancements, bug fixes, support, integration, acquisition or any variety or combination of these. In addition, the nature of the activities changes as the software matures.

The work performed to produce or support software development is usually very well understood by those actually performing the work, but these ways of working are difficult to communicate accurately or completely to others. Documentation may be correct but is often unusable especially to new recruits and is often only recorded as a user (of the development processes) view narrative. Many of the ways of working will be unique to the project or organization and can be idiosyncratic. Many of the development or support practices are also integrated (or confused with) the organization's management and business practices.

Software development (or enhancement, or bug fix…) is almost always managed as a project[1] with a project manager assigned to ensure it delivers. This approach is in many cases not appropriate. Many of the activities are repeatable and repetitive low risk activities where service or production management models would be more appropriate. Much of the work has attributes (repeatable, low risk) that real projects would aspire to. The project orientation also overloads these activities with the tools of project management that are not useful or needed. A further unhelpful side effect of this project orientation is the hindering of the understanding and development of software processes - "… our project is different …".

Within business environments with these characteristics many initiatives have striven to improve the effectiveness of software development:

- software methods and tools
- approaches to quality management
- management methods
- quality assurance
- standards
- metrics …

SPI has been one of the more successful and influential approaches. SPI is effectively the application of the SEI's Capability Maturity Model or its derivatives. The CMM takes much of the learned from earlier work and incorporates it in a simple model of software capability. This model describes what capability looks like but there is little useful information in the model on how the capabilities described are achieved.

There are a number of widely held SPI assumptions or beliefs:

- software development processes can be improved
- improvement is incremental or evolutionary
- improvement takes time
- institutionalisation makes processes robust.

These assumptions are all well founded. Ignoring them will lead to the failure of any SPI initiative – but they are not invariable verities.

But often there is little to improve – to which the typical response is "here is a process from my previous project/assignment/employer", or there is no time for conventional SPI - to which the conventional approach is "but SPI takes time".

During the course of many SPI projects we have been involved in or observed we have found that:

- long term commitment[2] to develop process understanding and capability is important
- long term plans are of little use, they are made obsolete by events
- SPI often takes too long to deliver real benefits to developers or managers (and they know this)
- predicting the effects of change is difficult
- large scale SPI is very difficult to manage
- large scale SPI can be inefficient and expensive
- many of the benefits from SPI arise from small changes made quickly

---

[1] A project is a unique complex intrinsically risky piece of work.

[2] Deming's 'constancy of purpose' manifested by senior management scrutiny – not big budgets

✍ some software management techniques are fundamental

Having learned these lessons we have found it effective to:

✍ think strategically but act tactically

✍ act locally

✍ deliver many simple improvements

✍ evaluate actual results

✍ learn from cheap fast failures and build on successes

This approach is called Rapid Process Improvement.

3.   Rapid Process Improvement

Rapid process improvements have a number of characteristics that are required and consciously built in:

✍ Clearly focussed on solving real problems. These may be technical problems affecting the developers or business problems that the development processes need to address. In either case a clear problem is there to be solved.

✍ Desired results stated explicitly and preferably measurably. What will the situation be like when the problem is resolved?

✍ Speed – scale and partition tasks to deliver useful results in days or weeks, not months.

✍ Results driven – work to achieve results rapidly and then act on the results. Do not track activity; track results and evaluate them.

✍ Perhaps use a model to guide the work but do not be driven by it.

✍ Highly visible. The work being performed should be visible to those affected, not just the results.

✍ Owned locally. Those affected by the changes own and control the changes. Process improvement is rarely successful if done to people; it is done by people. Process improvement is about changing behaviour and expectations and those it affects must be involved and in control.

✍ Local Accountability: Results (good and bad) should be accountable locally rather than be remote committees or management steering groups.

✍ Opportunistic – if there is a chance to make improvements or fix a problem do it now!

These characteristics should be self evidently desirable but are often at odds with the ways of working within software development organizations. Unless they are consciously striven for the tendency is for process improvements to become a ponderous committee-driven background activity rather than foreground activity almost indistinguishable from day to day development work.

The benefits of RPI are:

✍ Speed – improvements are in place and delivering value rapidly. Lessons learned are available early. The 80/20 rule applies to SPI and with rapid improvements an organisation begins to recognise were the value is early on.

✍ Visibility – involvement by those affected by process improvements and clear indicators of progress, lessons learned and benefits lead to acceptance of change, more rapid adoption of improved processes, management endorsement and the incentive to pursue further improvements.

✍ From a business perspective rapid process improvement is low cost and low risk. It will provide tangible evidence of the return on investment early, together with shorter break-even times.

✍ There is minimal lead time – results (benefits) come in very quickly

✍ The focus on real problems eliminates the temptation for expensive and time consuming generic training of all staff.

4.   Tools and Techniques

To perform rapid process improvement activities that exhibit the characteristics described above the right tools for process improvement are required. This section describes some of these tools[3]. The tools are divided into three categories:

✍ tools to provide visibility

✍ tools that provide frameworks and structure for process improvement activities

✍ tools for analysis, construction and change

---

[3] The listing and description of tools for process improvement was prompted by a question at a national process improvement meeting asking what tools others were using to make process improvements. No one volunteered a tool they used. Surprised by this response OSEL has been identifying and classifying different SPI tools. Many of these are now well defined and described. We thought that we ought to do as we recommend others to do – 'document your procedures'.

4.1   Tools to Provide Visibility

The following are examples of tools that provide visibility into the processes used:

❧ *Focussed Quality Assurance*

Quality assurance is a valuable tool to provide an organisation's management with visibility of the development processes as well as the quality of the software produced. Where quality assurance is not in place a simplified form of QA can be rapidly introduced as a tool for process improvement and management. Focussed QA requires that the needs that software development activities are to satisfy are stated explicitly by the organisation's senior management as policies. These policy statements should be carefully structured and worded to state clearly *what* is required. These policies can then be used as the basis for a foundation level QA system that can determine simply whether the policies are being adhered to. Typical policies may state, for example that projects will have an up to date plan or that deliverables will be reviewed and signed off. These policies do not prescribe details of how – that comes later - but it is a simple matter for a competent manager to determine whether the policies are being adhered to. This important, simple tool gives management basic information that can be developed when the processes that support the policies are developed. This evolutionary approach avoids the common problem with the introduction of QA where a spurious set of checklists of alleged 'good practice' are fabricated and used to police software development work before there is a good understanding of their real value.

❧ *Assessment*

Many of the models of software development capability, for example the CMM, are supported by methods for assessing organisational capability. These assessments can provide a very detailed picture of the organisational capability. In addition the methods may be structured in such a way as to make the capability very visible to the organisation. This can be very useful as an incentive to change. Care should be taken with assessments. A high profile assessment is intended to raise expectations of change and improvement. The assessment itself provides little guidance on making change. It delivers a statement of capability with implicit assumptions about *what* should be done next (although this should be treated with care). An assessment does not provide any information on *how* to make the required changes.

❧ *Measurement*

Quantitative information can be very valuable. When methods such as Basili's 'Goal Question Metric' are used then pertinent good data, providing valuable software quality management and process improvement information, can be produced. Quantitative targets can provide a focus for performance essentials. However the manner in which data is defined, collected, validated and used should be assessed carefully. Software measurement is not easy to do well. Poor definition and incorrect analysis can lead to deeply misleading or meaningless information. The integrity of data also requires monitoring; within some organisations systematic distortion or bias is routine.

❧ *Post Implementation Reviews*

Post Implementation Reviews are well known across the industry with many developers having taken part. When they are well managed they can provide excellent first hand descriptions of the work and products. These reviews are one of the very best sources of information about the issues and concerns of software development and management and are also an excellent source of potential solutions to those issues and concerns. When PIRs are conducted over a period of time patterns and trends in the issues may begin to emerge giving a picture of the issues that should be addressed by the organisation. To ensure that PIRs do not drop out of use (as so often happens) it is essential that actions are taken based on the information gained. A phrase that comes up time and again at the end of these reviews is "That was very interesting – what are you going to do about it?". Do nothing and you will lose a valuable tool.

❧ *Records*

Rather obvious and frequently overlooked, good records of development activity can be used to throw light on their efficiency and effectiveness. Good records will typically be accessible, usable in reasonably consistent formats, and most importantly trustworthy. The caveats are the same as for measurements. While some organizations will ensure the integrity of their records other will not. You need to know whether you can trust the records before you use them.

4.2   Tools for Analysis, Construction and Change

The following are examples of tools for construction and change:

✍ *Tactical Change Method (TCM)*
The Tactical Change Method was developed following observations from a number of sources. It is a simple 5-step process that provides structure and time boxing to process improvement activities. By defining a framework for change and predefining some of the key deliverables it increases the confidence of those entrusted with making change that they can deliver a result. It also safeguards against over ambitious or ambiguous process improvement efforts. In can be considered as a lifecycle for a rapid process improvement.

✍ *Process Definition (ProcDef)*
Defining processes is perhaps one of the most widely undertaken activities of process improvement. The majority of process definition is based on either drafting and reviewing processes or procedures or, more usually, importing them from elsewhere and tailoring them to a greater or lesser degree. The processes are typically written as a user view narrative, i.e. a user manual. ProcDef takes a slightly more formal view and treats the process definition activity in a similar manner to some software developments. ProcDef contains 5 activities that can be used collectively or individually as required. These activities are:

- Modelling

- Data Gathering and Refinement

- Identification

- Production and Validation

- Publication.

As a set these activities treat the processes as elements of a system that can be formally described and organized to meet well understood needs. These processes are then drafted as user guides and published and supported in a manner that will increase the take up and acceptability of the processes. The outcome is a well defined set of process assets that clearly support ways of working.

✍ Prefabricated Process Components ('Flat Pack')
The 'Flat Pack' approach to process improvement was developed in response to a problem seen in a large number of organisations. Successful process improvement is usually undertaken by expensive or scarce individuals who do not want to operate the processes once created. The individuals that can operate the processes often do not know how to put them in place. Attempting to put process in place within an organisation can take a very long time and often fail. Process Flat Packs are prefabricated processes that are rapidly tailored

to an organisation's needs and capabilities and installed in the organisation to begin operating in a matter of days or weeks rather than months. The benefits of the process can be assessed rapidly, using real information and changes made if required. This approach has lead to the concept of 'Software Production Engineering' where a complete software capability could be installed, scaled up, duplicated or moved.

✍ PIR Led Process Improvement (PirL)
PirL is a logical extension of Post Implementation Reviews. The PIRs deliver good information about the performance of software development and support. PirL provides the mechanisms to ensure that this information is acted on either in real-time, to enable the work to be performed more effectively immediately, or in the longer term, by recognising patterns and trends in the information and ensuring that the organisation acts on this information (where individuals or projects cannot).

✍ DevPIP
Assessments (described above) provide a good description of an organisational capability, in effect a snapshot of capability. Assessments will often produce a number of recommendations on what should be done to improve capability. However the recommendation are rarely well considered. The assessment process focuses on understanding capability not on producing plans for improvement. Developing a Process Improvement Plan (DevPIP) has been designed to take the output of an assessment, including recommendations and use this information as part of the input to a process improvement planning exercise. DevPIP develops a plan base on capability, and business needs and priorities. This structured process evolves a prioritised plan that is relevant and owned by the organisation. An important part of the DevPIP process is the transformation of understanding of capability by the assessment team into a plan for improvement driven by the organisation's senior management and endorsed by those it will affect.

4.3 Examples of Tools that Provide Frameworks and Structure

The following are examples of tools that provide the frameworks and structures for process improvement:

✍ Process Improvement Templates
Not all improvements are the same. The templates identify the different types of improvements so that the correct approaches

can be deployed. In essence there are four types of improvement:

- introduce something new

- change something already in use

- promulgate an existing but localised best practice

- resurrect an required but perhaps unpopular or discredited practice

The tools to ensure that the changes are made will be different in each case. The PI templates identify appropriate tools and tactics.

✍ Process Infrastructure
A process infrastructure is a description or definition of the components of the organisation that support software development activities with processes, procedures, methods and tools. The process infrastructure provides a shared mental model of the functions supporting process within the organisation. It also provides stability for processes and functions by providing a context.

A generalized example of a process infrastructure is shown in Fig 1.

specialised part of the process infrastructure. It defines the infrastructure for process improvement activities and explicitly describes how senior management, managers and developers organise and manage the limited resources required to make improvement happen. The two most important parts of the process improvement infrastructure are:

- the senior management oversight and feedback mechanism

- the resourcing model describing how tasks get assigned and completed

✍ Process Architecture
The process architecture is an aid for those developing processes or procedures because it provides context. It shows how the various elements fit together. The example diagrams shown below is a real example developed for a software development organisation. It was not intended to be part of the user documentation set of procedures and guidelines but rather, like software design documentation, intended to aid those managing the production of the documentation. A more common approach used to provide context for process and procedures is a software development lifecycle model. This is also very useful for managers and developers in providing structure for plans but tends to have
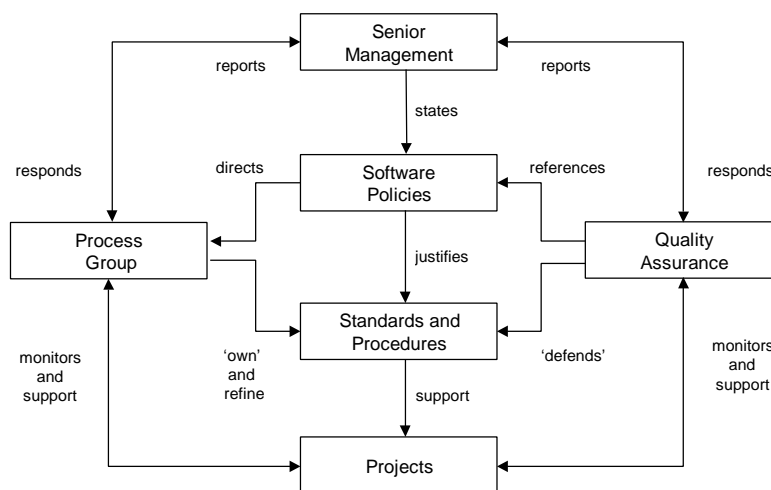

Fig. 1. An example of a process infrastructure

Process Improvement Infrastructure
The process improvement infrastructure is a

problems placing ongoing processes such as change control or formal reviews in context.

Two components of an example process architecture model are shown in Fig 2.

and situations. However to make real progress certain fundamental practices should be in place to provide a foundation for processes and process improvements. If these fundamentals are not in place then they should be put in place first. They stabilize
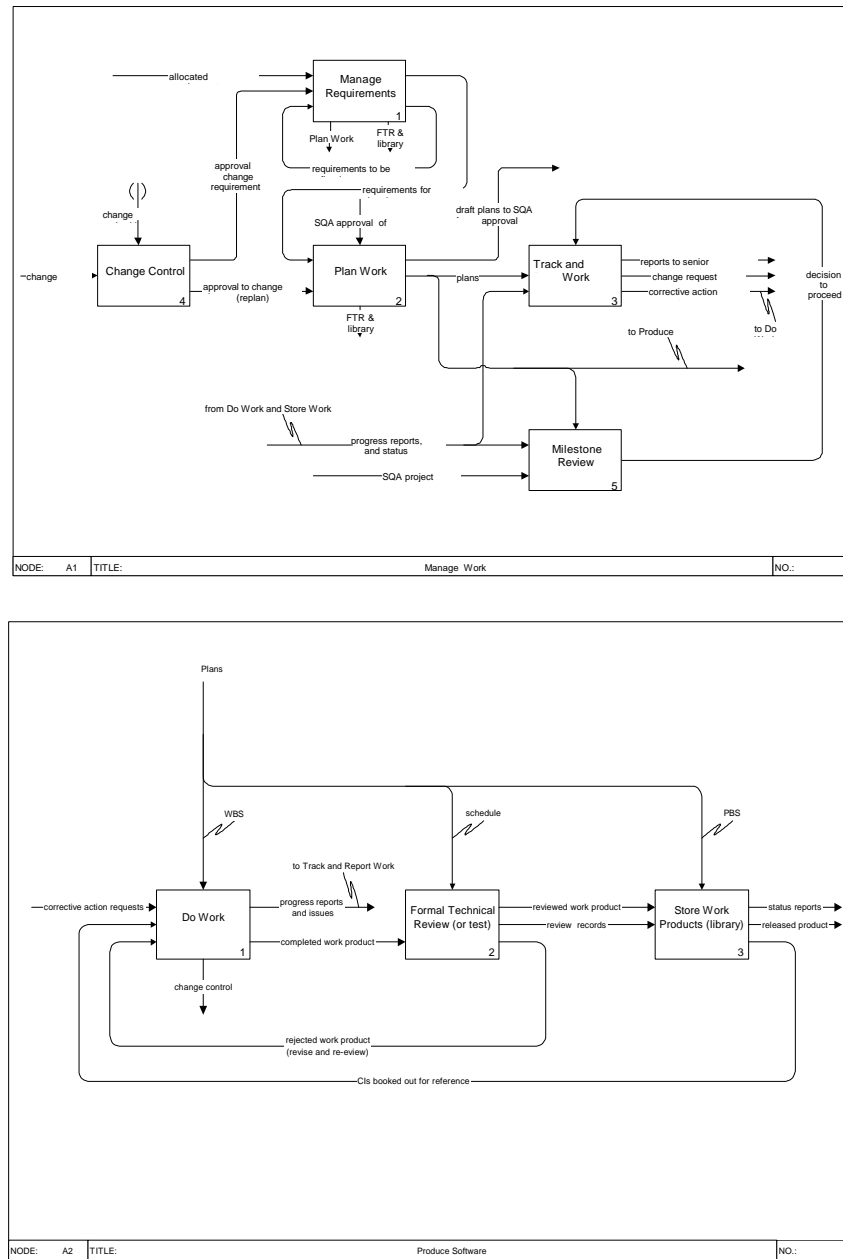

Fig. 2. Two components of a process architecture model

5. The Fundamentals

The tools and techniques briefly described above have been used in a wide variety of organisations

processes and quality and can act as a catalyst for further improvement. The fundamentals are:

- ✍ Quality Control
  formal review of documents and tests for software

- ✍ Change Control
  to ensure assets are not lost and effort is focussed where required

✍ Management Review
to assess progress (as distinct from activity) on an ongoing basis.

Those involved in process improvement should assess these fundamentals and ensure they are functioning effectively. They are not only essential for software development and management, they are also tools for process improvement.

6. Closing Remarks

Rapid Process Improvement is a conscious effort by OSEL to adapt Software Process Improvement activities to the new rapidly changing small project environment. Many of the tools for understanding and improving software development have a role and can be developed further. However it has been necessary to identify and develop new ones. This process of equipping process improvement initiatives has only just begun and we can expect continued development of new tools for Rapid Process Improvement.

Further inform,ation about rpi and the rpi tool set can be found at http://www.osel.co.uk/rpi/rpi.htm