An evolutionary path to process definition

Dick Holland
27 August 2006
Version [0.1]

1 What makes a software product?

Here are some characteristics which could be considered to be common to software products. This list was written by the head of development of a financial services software house in 2002, so it's especially useful because it's a real-world example.

Characteristics of a software product

1. The deliverables (software, meta-data and documentation) that make up the product are released together on a predefined schedule, with a single identifying version number for each release.
2. The product must evolve to retain fitness for purpose and competitive advantage in response to changes in the business environment.
3. All the deliverables that constitute a release of the product can be placed together onto a CD or equivalent medium.
4. There is a one-to-many delivery of product to customers – all new customer installations, and all upgrades for existing customers, can be achieved purely using the content of the CD for a given release level.
5. Where the behaviour of the product must be tailored to a particular customer environment and preferences, this is achieved exclusively through changes to settings that are made using high-quality tools that prevent selection of invalid choices or combinations.
6. The installation process must be fully automated, and self-verifying.
7. The product must provide a usable business solution immediately following installation. Any critical customer-specific information that is required for the product to function must be captured during the installation process.
8. The executable components are built from a single, common code base. No customer owns or limits the distribution of any part of the software (although customers may contribute to product development costs to ensure delivery of features within their preferred timescales).
9. Upgrades to later versions can be applied in an automated fashion, without manual merging activity, and without loss or change of functionality except where functional change is inherent in the upgrade.
10. All product releases are subjected to rigorous progression and regression testing prior to delivery to customers.

These characteristics can be seen as some of the key objectives for any software product and can consequently be interpreted as ***Requirements*** for the development of the product and its supporting services. We can conclude from these requirements that we need to invest effort in:

- Standardisation of the way we produce and support the product;
- Automation where practicable in order to meet the volume and frequency goals and to reduce the impact of human error (inescapable when we use people to do things).

Both of these imply a process orientation, and therefore a need to define the processes we're going to use to meet our requirements. The SEI [CMMI00][1] defines ***Organisation process definition*** and ***Organisation process focus*** as key practices at CMMI Level 3, so it certainly looks like a Good Thing to do.

1.1 About this document

This document proposes an evolutionary path that we can follow in order to define and maintain the processes needed to enable us to meet the requirements implied by those characteristics.

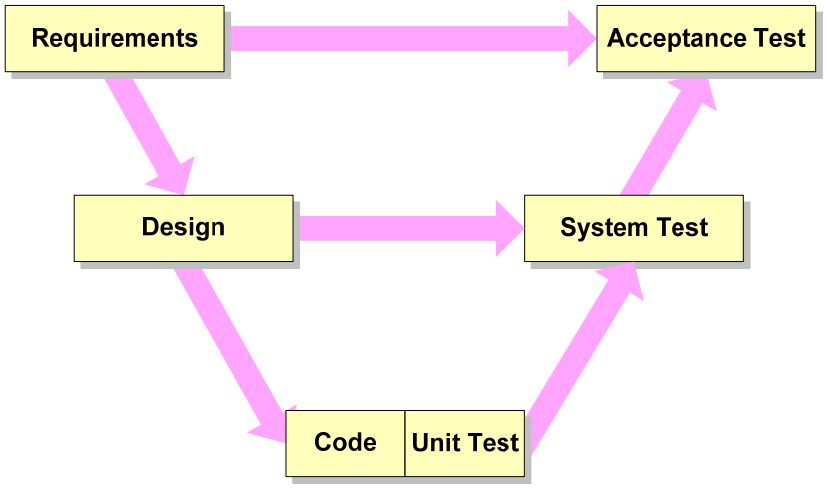
2 About processes

In seeking how to define processes, it's worth looking at how the experts do it. In ***Reengineering the Corporation*** Hammer and Champy [Hammer93] give this definition for 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"*

So, simply put, a process is any transformation we perform on inputs (sources) to create outputs (work products) which are of value to our product or service and ultimately to our customers.

In the classic "V" model of software development we can quite clearly see the transformations that take place. Here's a simplified version:



The left side of the "V" represents the specification and construction of software as successive refinements at lower levels of abstraction, in which the outputs of upstream processes are used by their neighbours downstream as sources. The right side represents successive processes of integration and quality assurance (such as testing) at higher levels of abstraction.

Notice, however, that the work product of the ***Requirements*** process in the picture is also used as a source to the ***Acceptance Test*** process, and that of the ***Design*** process is similarly used by the ***System Test*** process. This tells us that the document produced by the requirements process for example is to be consumed not only by people doing design, but also by people doing acceptance testing. It should, therefore, contain information

[1] A list of references is on page 13

relevant to both types of reader and be couched in language that they both can understand, which implies at least guidance in the form of standards or templates to help authors put the right stuff into their documents.

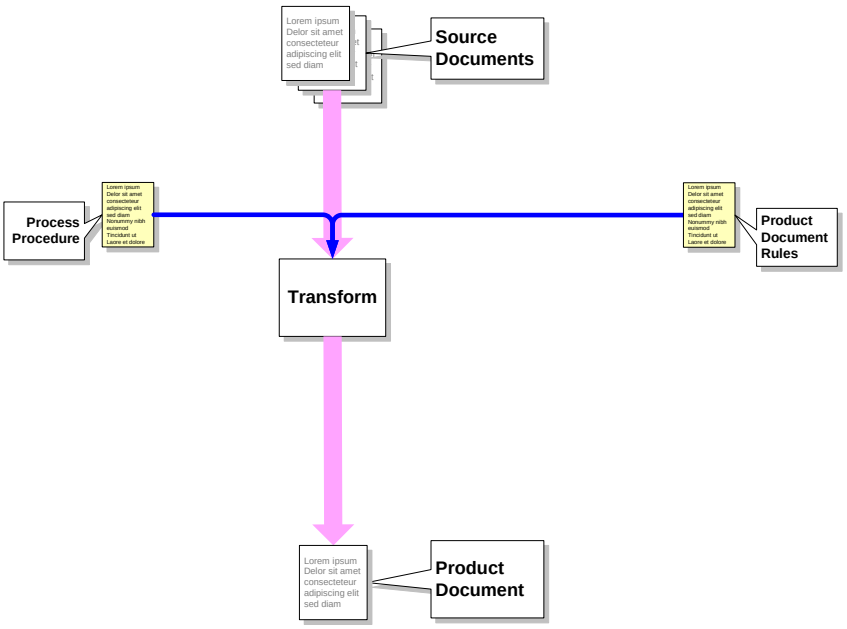
3 Stages of process evolution

Starting from an early stage of maturity in which processes may not be well-defined, we can plot an evolutionary path through increasingly comprehensive process definitions. The speed at which we move from stage to stage can be influenced by how fast we want to move and how fast we are able to move, taking account of everything else that we're attempting to do at the same time.

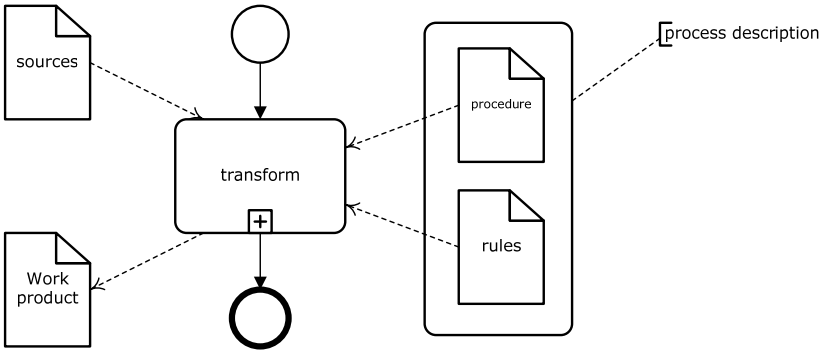
Throughout this section, we'll be using as our example a ***canonical process*** which is not any process in particular, but rather the shape of them all.

3.1 A simple process

In the pictures which follow we're going to make the not unreasonable assumption that all the processes we're concerned with here will be transforming source documents (inputs) into work product documents (outputs) – whether human- or computer-readable.



And here it is expressed in BPMN, the Business Process Modelling Notation [OMG06]:



At its simplest, a process can be defined by a procedure (how to do the transformation from sources to work product) and a description or definition of what the outcome of the process should be, defining what the work product should contain and also, possibly, how it should be formatted. These aspects of the definition are obviously vital for a process whose work product is computer-readable, but less obviously so for human-readable products. We can call this definition of the work product a set of rules or, more simply, a ***Ruleset***.

So here in our simplest case, we can define each process by two pieces of paper, which we'll call ***Process Artefacts***. It's a good idea to have a principle that no process artefact shall exceed one page of A4 in size – it prevents obesity. And small documents are far more likely to be read thoroughly than huge ones!

Below by way of illustration is an example of a process artefact. In fact, it's the ruleset for a Requirements Analysis document that we developed in 2002. Some points about its formatting are worth making:

- The paragraph following the title contains the identification, and the edit and quality status of the document (the prescriptions for this are encapsulated in the ***Generic Rules*** shown on page 5);
- Non-commentary text (stuff that's vital to the functioning of the document) is in Roman (again, guided by the ***Generic Rules***);
- Commentary text (background or supporting information) is made distinctive by italicising it (also guided by the ***Generic Rules***).

All the process artefacts shown in this paper have been formatted partly to meet the requirements of the Inspection process, which is most comprehensively described in ***Software Inspection*** [Gilb93], the definitive work on the subject written by Tom Gilb and Dorothy Graham.

The Requirements Analysis ruleset has itself been written according to sets of rules (yes – we even have rules for rules): the ***Generic Rules***; and the ***Ruleset for Rules***, both of which are shown below. In the Requirements Analysis ruleset, note that in tag RULES.RA.RA6-ATTRIB (also notice how this tag notation yields a unique reference for this single statement across all documents) reference is made to "Planguage". Planguage is a Planning Language invented by Tom Gilb [Gilb05].

RULES: Requirements Analysis

Version [0.5]; **Date** 29 November 2002; **Editor** DH; **Tag** RULES.RA; **Rules** RULES.G [0.3], RULES.R [0.3]; **Pages** 1.0; **Readers** Authors, Inspectors; **Status** Unexited > 60 majors/page;

PURPOSE	*The purpose of this document is to provide a RuleSet against which a Requirements Analysis can be written and checked.*
RA1-READERS	Readership shall include customers and customer advocates, designers, testers and technical authors.
RA2-PROB	The underlying business problem shall be stated. This is the fundamental business reason, from customers, the company or a combination of both, for making a change to the product. *This is quite different from any particular business requirement needed to address the problem. For example 'customer needing to support a new product' or 'the company wishing to move into a new market area' are business problems, while 'record details of such and such trades' is a business requirement.*
RA3-REQS	The business requirements for a change to the functionality of the product shall be stated. Business requirements comprise a breakdown of the problem statement into the requirements necessary to address the business problem. *The business requirements represent the demand side of the economic equation.*
RA4-PROD	The product requirements shall be stated. Product requirements are the requirements and constraints governing the implementation as to its practicality, cost and time. *The product requirements represent the supply side of the economic equation, e.g. base the new functionality on the X feature or meet deadline Y.*

RA5-ADDRESS	It shall be stated which of the business requirements are to be fully met and which are not.
RA6-ATTRIB	Attribute requirements shall be specified measurably and testably. Attribute requirements are business requirements or product requirements that are quantitatively stated. *Using, for example, such Planguage elements as SCALE, TEST, PAST, RECORD, MUST, PLAN.*
RA7-UNRES	There shall be no assumptions or unresolved questions or issues in the document.

The ***Generic Rules*** are the rules with which every document shall comply. For each specific document type, there is a set of ***Specific Rules*** which are additive. The Requirements Analysis ruleset above is an example of specific rules.

GENERIC RULES: A Rule Set for Any Document

Version [0.3]; **Date** 29 November 2002; **Editor** DH; **Tag** RULES.G; **Rules** RULES.R [0.3]; **Pages** 1.0; **Readers** Authors, Inspectors; **Status** Unexited > 60 majors/page;

PURPOSE	*The purpose of this document is to provide a generic RuleSet against which any document can be written and checked.*
G1-PURP	The purpose of the document shall be stated within the document.
G2-NOTE	All statements which are commentary shall be clearly identifiable.
G3-EXTRA	The document shall be as brief as possible, to support its purpose.
G4-CLEAR	The document shall be crystal clear to the intended readership as to intent. The burden is on the Author/Editor, not the reader.
G5-UNIQUE	Statements shall be made only once. Thereafter, both within the document containing the statement, and in all other documents, they shall only be repeated or paraphrased as commentary and the unique Tag of the original statement given.
G6-SOURCE	The document shall declare a Source Reference for each statement, or group of statements, contained within it, whether these sources are within the document or in another document.
G7-EL	All statements shall be written to communicate one new piece of information only. This is to permit separate analysis, costing, implementation etc.
G8-TAG	All elementary statements shall have their own identity tag for direct reference from within the current document or from other parts of any larger document set.
G9-CHANGE	Editor, Edit date, Quality Status, Correctness Status and Sources of any change shall be indicated at the level of change.
G10-RISK	Any known or suspected uncertainty or risk shall be clearly indicated with a suitable explanation. *Examples: '60 -> 70 days', or 'availability 50% -> 70% of working days'.*
G11-HEAD	The document shall contain a title, Tag, Version, Author/ Editor, date last edited, Quality Status, Defect Density, Sample size if sampling used, Correctness Status, list of Rule Sets used, number of Logical Pages, intended Readership, and a list of Sources, kin and downstream documents used.
G12-HIER	Documents shall be organised into clear hierarchical structures.
G13-VERIFY	Documents shall be correct and complete for their purpose. *Specialist knowledge is required to target issues that break this rule.*

G14-READ	Sections of the document relevant to only a sub-set of the intended readership shall be marked as such with a banner stating the actual intended readership.

In addition to formal rules, we may also have ***Checklists***. Checklists are less formal, and they exist primarily to provide Inspection Checkers with additional questions and considerations when checking work products.

CHECKLIST: Generic Document Checklist

Version [0.3]; **Date** 29 November 2002; **Editor** DH; **Tag** CHECK.G; **Rules** RULES.G [0.3]; **Pages** 1.0; **Readers** Checkers; **Status** Unexited > 60 majors/page;

PURPOSE	*The purpose of this document is to provide a Checklist to aid checking of any document.*
GCK1	Has simple, clear English been used? {RULES.G.G4}
GCK2	Have abbreviations and technical terms used in the product document, been defined? {RULES.G.G4} *A specialist glossary, within the product document, or a tag reference to a main glossary could be used.*
GCK3	Have proper names such as file, field and data element names and screen prompts and messages, used in the product document, been defined? {RULES.G.G4} *A specialist glossary, within the product document, or a tag reference to a main glossary, e.g. the User Manual, could be used.*
GCK4	Have all paragraphs been tagged? {RULES.G.G8}
GCK5	Are all references to the same concept or object identified uniformly throughout the document? {RULES.G.G5}
GCK6	Do the methods used, such as textual description or diagrammatic representation, convey the document's meaning effectively? {RULES.G.G4}
GCK7	Have examples which demonstrate the meaning of statements been provided? {RULES.G.G4}
GCK8	Are definitions of glossary terms and proper names available to and understandable by all readers? {RULES.G.G6, RULES.G.G4}
GCK9	Does a statement need a source reference? {RULES.G.G6}
GCK10	Are all sources documented, even if only briefly as an appendix? (e.g. telephone conversations.) {RULES.G.G6}

And finally we mentioned above that we have rules for writing rules. This is a set of Specific Rules for Rules.

RULES: A Rule Set for Rules

Version [0.3]; **Date** 29 November 2002; **Editor** DH; **Tag** RULES.R; **Rules** RULES.R [0.3], RULES.G [0.3]; **Pages** 0.5; **Readers** Authors, Inspectors; **Status** Unexited > 60 majors/page;

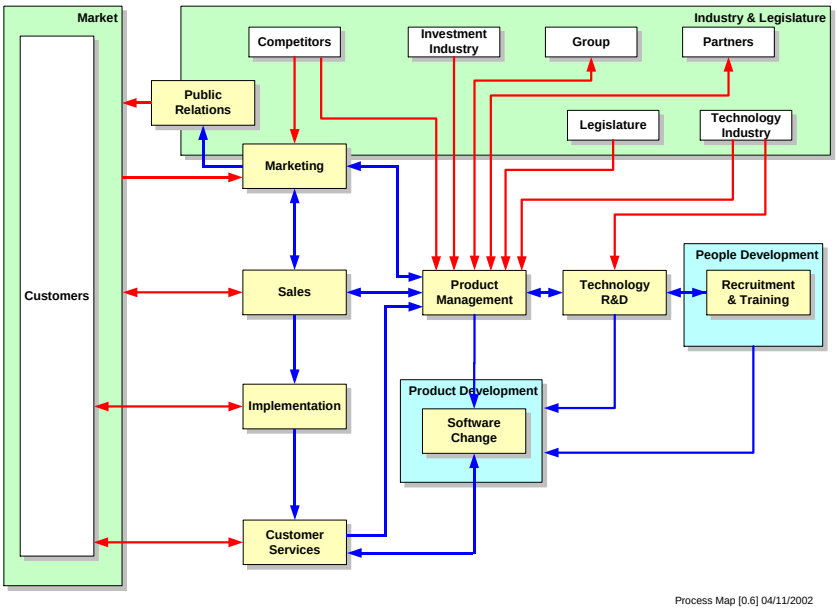
PURPOSE	*The purpose of this document is to provide a RuleSet against which a RuleSet (including Entry and Exit Criteria) can be written and checked.*

R1-PAGE	No Rule Set shall ever exceed a single A4 page or contain more than 1½ logical pages of text. *A logical page is defined as 300 non-commentary words.*
R2-EXIT	The rules within each Rule Set shall have successfully exited from an Inspection against both this Rule Set and the current revision of the generic Rule Set {RULES.G} before being taken into use.
R3-REF	Each rule's unique, full, tag shall be in the form 'brief-descriptive'. The brief tag shall be unique within the Rule Set, and shall only be used for logging. The full tag shall always be used for other purposes. The purpose of a brief tag is to aid fast logging. *This rule's brief tag is 'R3', and its full tag is 'R3-REF'.*
R9-DEL	When rules are deleted they shall be removed from the Rule Set. A list of all deleted rules' tags shall be kept.
R5-NEW	Each deleted rule's tag, both brief and full, shall not be reused.
R8-READ	Readership of each rule set shall be 'Checkers & Authors'.

Writing documents to comply with these simple rules has the immediate and enormous benefit of improving communication by eliminating misunderstanding of notations and typographical conventions. The intention of every statement in every document can be clearly understood because they all look the same and the conventions are defined.

Almost as an aside it's worth mentioning at this point that if we define all our processes even in these simplest of terms, we will have produced a process standards manual or process handbook. For some organisations that is enough and they feel no need to go further.

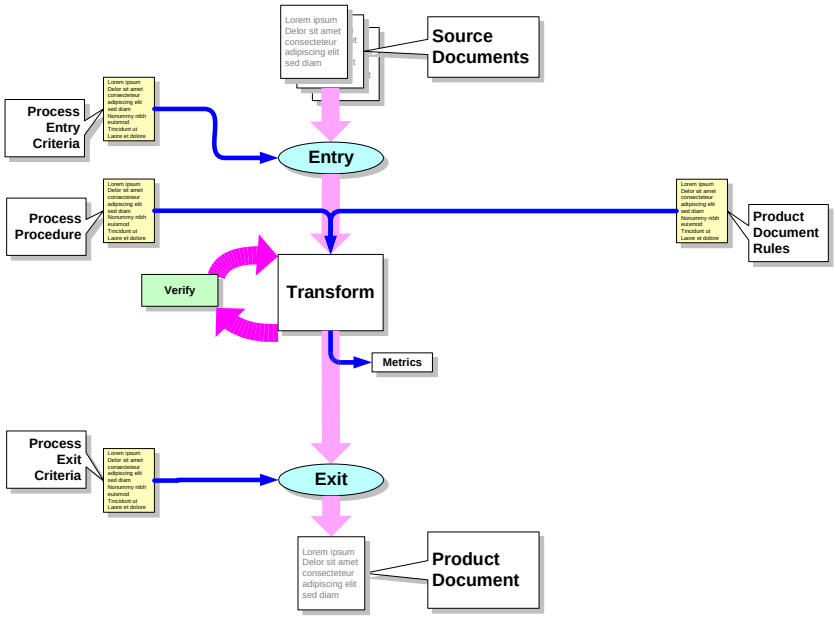
Unsurprisingly perhaps, the process of defining processes can also be documented in the same way as every other process. The ***Process Definition Process*** artefacts are exhibited in an appendix starting on page 14. You can see from those that the start point can usefully be a Process Map that shows in outline form how the processes fit together to implement different aspects of the business. Here's the outline version of one we did in a software vendor in 2002.



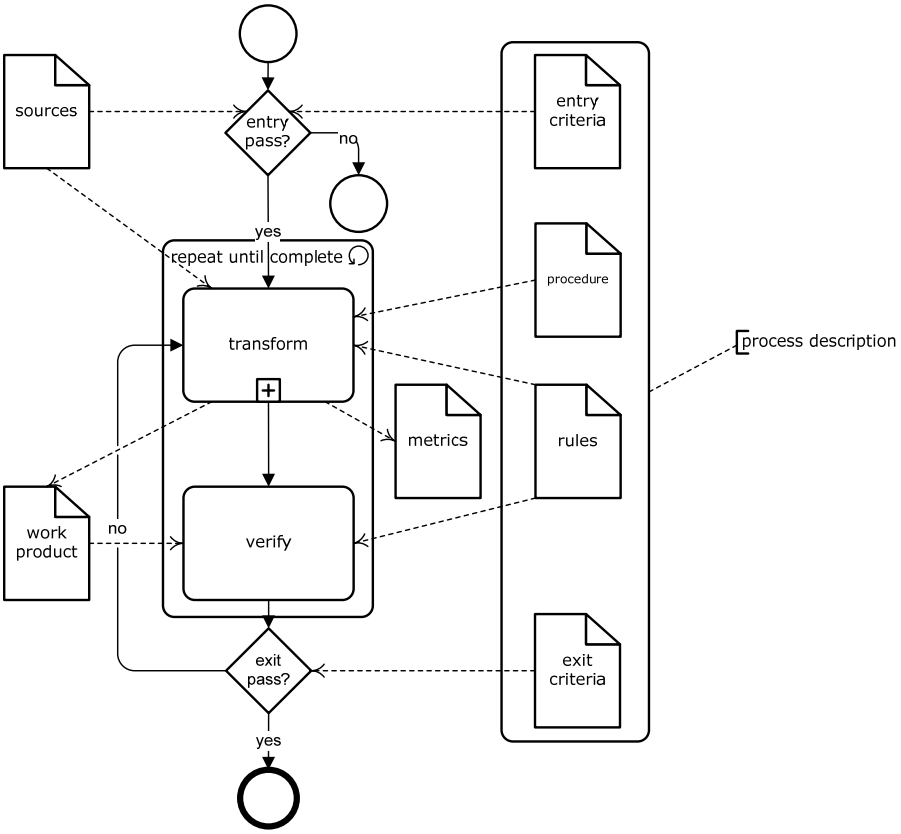
In this picture, the software development processes are exploded out of the ***Software Change*** Process Group box. At this level of abstraction, the process map for any software company may be very little different from this.

3.2 A controlled process

Having described our processes very simply, we may now wish to introduce some control into their execution. So far we've shown how they fit together and what they do, but we've not been explicit about under what conditions they start and stop. That's been left to the judgement of the practitioners.



And here it is expressed in BPMN:



To the procedure and the work product rules we can add entry and exit criteria. Entry criteria state what must exist before we can start. Exit criteria state what must exist before we can claim that we have finished. We often dispense with the latter if the downstream process(es) are ours, because some or all of their entry criteria will by definition be the exit criteria of the subject process. But if the processes downstream are, for example, in a customer then it's a good idea to retain exit criteria because there's no guarantee that customers will have entry criteria checks.

We can also add two other things: process metrics so we can measure how we're doing; and a verification sub-process to verify the correctness of what we're doing. These can be added later in our evolution, and are there to help us control the execution of the process and the quality of its work products. The verification sub-process is shown in the picture as an iterative cycle with the transformation, illustrating that we expect to do them more than once in any process execution cycle. This is an example of the opposite of the much-vaunted and flaunted "do it right first time" edict, and it's in recognition that in the real world that is pretty unlikely ever to happen in an essentially intellectually creative endeavour such as software development. At the process's external interface, however, we strive indeed to "do it right first time", and our process metrics can be constructed to help us measure the degree to which we are being successful in doing that.

The verify sub-process very often takes the form of a review, either by peers or by experts or a mixture of both, or some kind of walkthrough, structured or otherwise. Whichever it is, it is often in the early stages a fairly informal process. The metrics can help us identify some systemic defects on our process, for example if we're doing too many reworks of the work product after whatever reviews we're conducting.

To illustrate this, here is a process artefact: in fact the procedure for a Correctness Verification sub-process we developed in 2002. And it's a good example of the simplicity of typical procedure documents.

PROCEDURE: Correctness Verification Process

Version [0.3]; **Date** 29 November 2002; **Editor** DH; **Tag** PROC.CV; **Rules** RULES.G [0.3]; **Pages** 0.5; **Readers** Process Operators; **Status** Unexited > 60 majors/page;

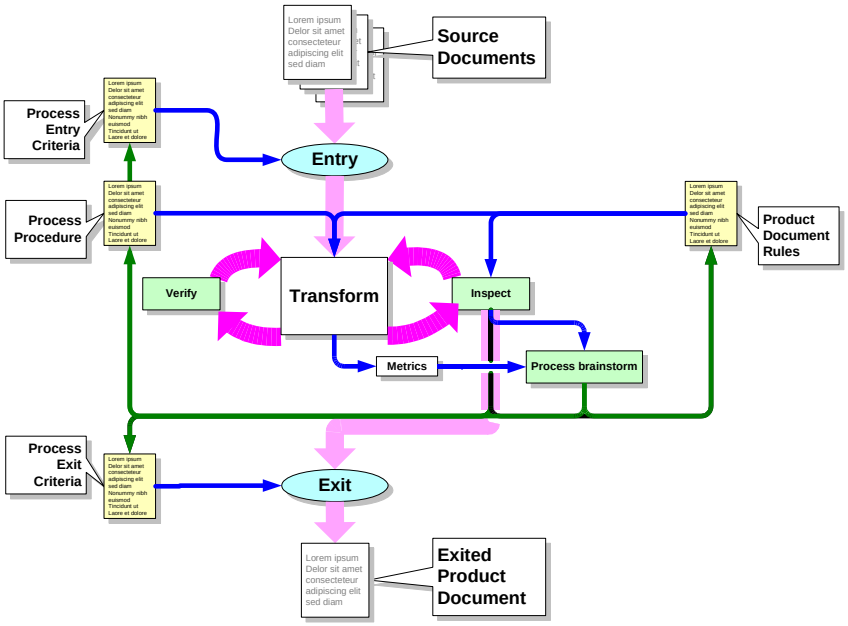
PURPOSE		*The purpose of this document is to provide a Procedure for Correctness Verifying any document.*
1.		The team required to operate the procedure comprises at least one author/editor of the product document and at least one of the following, known as the "verifier":
	i)	a user or customer of the process about which the product document is written;
	ii)	a consultant or other third-party acting on the customer's behalf;
	iii)	an industry expert with requisite domain knowledge.
2.		The procedure is iterative, with as many cycles as are required to satisfy the verifier that the document is correct.
3.		Step through the product document one statement or paragraph at a time, whichever is most appropriate for the composition of the document. *Whichever granularity is used, the logical unit of work will be referred to as a "statement" hereinafter.*
4.		For each statement in the document:
	i)	explain it;
	ii)	ensure the verifier fully understands the statement;
	iii)	encourage the verifier to raise issues and ask questions;
	iv)	record the issues and questions raised.
5.		Ensure that all participants are looking for:
	i)	omissions;
	ii)	incorrect statements;
	iii)	ambiguous statements;
	iv)	unnecessary or superfluous statements.
6.		Resolve the issues and answer the questions raised.
7.		Edit the document to remove the issues and questions.
8.		Add the Correctness Status, which comprises the date and the names of the team members, to the document's header.

At this stage we have the process defined and its execution controlled by the use of entry and exit criteria. So we have documented how to execute the process, what needs to exist before it can start and what must exist when it's finished. We also have some ability to determine what improvements can or should be made from the metrics we're gathering. But this is yet ad hoc – we still lack a systematic, rigorous method of measuring the quality of the work products and the processes that create them.

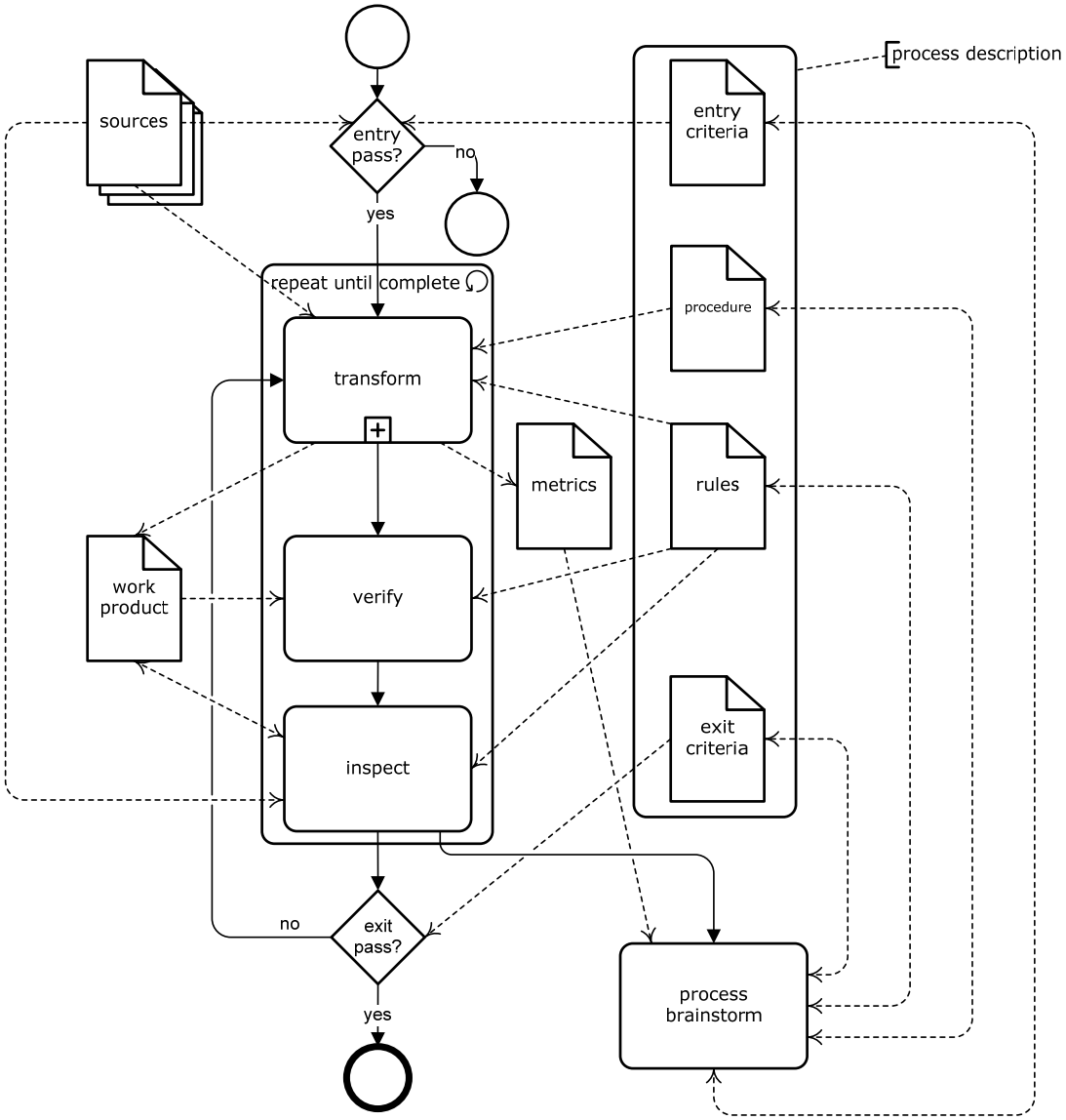
3.3 An improvable process

To solve these problems, we can introduce one additional process into the mix: ***Inspection***.

Inspection is an example of a statistical quality control process which originated in manufacturing industries. The Inspection process we're describing here is an adaptation designed specifically for written documents of any kind [Gilb93]. There are myriad examples of it being used not just for software engineering documents but also for manuals, contracts, agreements and even letters to customers – if it can be written down, it can be Inspected to measure and improve its quality.



And here it is expressed in BPMN:



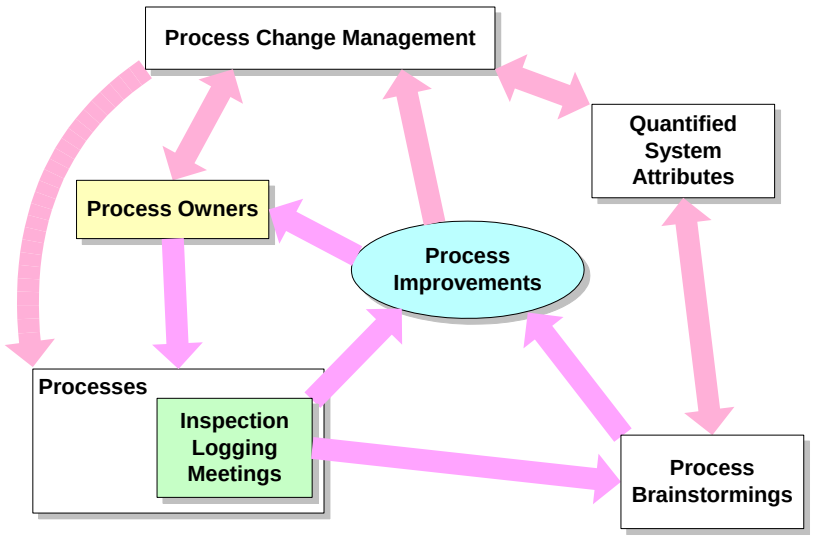
Notice in the picture that the product document now goes to the Exit Criteria Check from the Inspection process; this is to illustrate our desire to have Inspections embedded in every production process in order to control the quality of the work products. Notice also that we now have two iterative loops around our transformation: one that includes our correctness verification and one that includes Inspection. Many organisations at this stage of their evolution dispense with the separate verification process, choosing instead to assign a ***Correctness Verification Role*** to someone in an Inspection team. Inspection, however, is quite a technical process and may be somewhat daunting for participants such as customers and external industry experts; so keeping the Verification process is a good idea for documents which such people are helping you to write. And as it's just another process, Inspection is itself defined by a set of process artefacts.

At the end of some or all Inspections we can also conduct Process Brainstormings at which we're looking in much greater depth for systemic defects in our process. The Inspection and Process Brainstorming processes between them provide feedback loops to modify not only the source and work product documents but also all the process definition artefacts; the latter are shown by the dark arrows in the picture.

It's this tight integration between the production processes, the Inspection process and Process Brainstormings which make our processes improvable. Every time we run an Inspection we're creating the opportunity for people to look at the defects in our documents and our processes and suggest improvements, and with our processes instrumented to provide metrics we can very quickly measure the effects of any changes that we've made. This is ***Continuous Process Improvement***.

3.4 Continuous process improvement

The SEI [CMMI00] defines ***Quantitative process management*** as a key practice at CMMI Level 4 and ***Process change management*** at Level 5. These are both part of Continuous Process Improvement; the whole picture looks like this:



In this picture we can see the Process Improvements feedback loops going from Inspections and Process Brainstormings to Process Owners. Process Ownership is a key to an organisation's ability to improve its processes: if no-one owns a process, who ensures its improvement? In some enterprises, all process ownership resides with the management of the relevant part of the organisation. This is, however, inefficient as it necessarily reduces the rate at which change can happen. It's far better to devolve ownership and accountability for process performance to nominated Process Owners and maintain an oversight through a Process Change Management Team as the picture shows.

This picture also introduces one other concept: ***Quantified System Attributes***. These are simply numerically quantified objectives for whatever the products and processes happen to be: in our case it's a software product and its development and support processes. So we could define quantified targets for such attributes as software reliability, responsiveness to customer calls for help and so on. In the presence of contracts,

licence agreements or policy documents which commit service levels to customers such attributes may already be defined and their numerical targets set.

Quantified System Attributes, then, define the targets for qualities of various kinds; possessing instrumented processes which provide us with metrics is the most powerful way of ensuring we know whether we're meeting those targets. And having a Continuous Process Improvement framework is the most powerful way of ensuring that we can get ever closer to them – especially if the targets are continually getting tougher! Quantified System Attributes are therefore in some senses the well-spring from which everything else flows. The attributes describe the goals that we must aspire to; everything else in the picture can be viewed as a set of strategies to enable us to attain those goals.

3.5 Journey's end?

So we've seen that we can define each process very simply using only two artefacts: the process procedure and the work product rules. We can make that process controllable at the cost of adding at most three more: entry criteria, exit criteria (which are optional) and a verification procedure. And we can make that process improvable by adding the Inspection Process with its own entry and exit criteria, rules and procedures. And the Process Brainstorming process which gives us that vital feedback loop to enable ***Continuous Process Improvement***. And having got there, we have at least one practice CMMI Level 5, and others at CMMI Levels 3 and 4.

And we can introduce those increasing levels of control at whatever pace we want and tackle our processes in whatever order we choose. So we have control over the cost and time we are willing to put into this evolution – the important thing for us to remember at each step along the way is to tackle the highest priority issue we have. By doing that we can adjust the pace and timing of the evolution, but we will always know that our metrics will be telling us always whether what we just did works or not. That makes it very easy to change, or even undo, anything that does not take in the direction of our goals

So is this the end of the evolutionary journey? Not a bit of it – it's just a waypoint that we've reached so far. The ultimate industry-recognised goal for a software organisation is currently certification at CMMI Level 5, and what we've described here gets us a significant part, but not all, of the way there.

4 References

[CMMI00] CMMI Product Development Team. *CMMI^{SM} for Systems Engineering/Software Engineering, Version 1.02 (CMMI-SE/SW, V1.02) Staged Representation CMU/SEI-2000-TR-028*. Pittsburgh, PA; Carnegie Mellon Software Engineering Institute, 2000.

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

[Gilb05] Tom Gilb. *Competitive Engineering*. Oxford, England: Elsevier Butterworth-Heinemann, 2005.

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

[OMG06] *BPMN Specification.* http://www.bpmn.org/Documents/OMG_Final_Adopted_BPMN_1-0_Spec_06-02-01.pdf. Object Management Group, 2006.

5 Appendix: The Process Definition Process

There follow the process artefacts for the ***Process Definition Process*** we defined in 2003:

- Its Entry and Exit Criteria;
- The Rules for the key work product, the Process Procedure for the process being defined;
- The Process Procedure for the Process-definition Process itself.

5.1 Entry and Exit Criteria

The Entry Criteria state what must exist before we can start defining the target process.

ENTRY CRITERIA: Process Definition Process

Version [0.2]; **Date** 6 November 2002; **Editor** DH; **Tag** ENTRY.PDP; **Rules** RULES.G [0.1]; **Pages** 1.0; **Readers** Process Owners, Operators and Work Product Consumers; **Status** Unexited > 60 majors/page;

PDPE1	The process definition team shall be formed of at least one process operator and at least one work product consumer.
PDPE2	The owner of the process being defined shall be represented on the process definition team.
PDPE3	All documents describing the current process shall be available.

The Exit Criteria state what must exist before we can claim we have finished defining our target process.

EXIT CRITERIA: Process Definition Process

Version [0.2]; **Date** 6 November 2002; **Editor** DH; **Tag** EXIT.PDP; **Rules** RULES.G[0.1]; **Pages** 1.0; **Readers** Unrestricted; **Status** Unexited > 60 majors/page;

PDPX1	All artefacts for the process being defined shall have exited at least Correctness Verifications or, ideally, Inspections.
PDPX2	All artefacts for the process being defined shall be recorded in the process database.

5.2 Work Product rules

The work products of the Process Definition Process are:

- A ruleset for each work product that the subject process is to produce;
- A procedure document for the transformation to produce each of them;
- A set of Entry Criteria;

- Optionally a set of Exit Criteria.

There follows the ruleset for the procedure document.

RULES: Process Procedure Document

Version [0.4]; **Date** 6 November 2002; **Editor** DH; **Tag** RULES.PROC; **Rules** RULES.R [0.2]; **Pages** 1.0; **Readers** Authors, Inspectors; **Status** Unexited > 60 majors/page;

PPR1-PURP	The purpose of the process shall be stated such that the transformation of its sources into its work product(s) shall be clear. [0.2]
PPR1-STEPS	All steps of the process shall be stated in the order in which they are to be carried out.
PPR2-ORDER	Where two or more steps can be carried out simultaneously, this shall be clearly stated in each affected step.
PPR3-CLEAR	The operation of the process shall be made clear to a reader unfamiliar with it.
PPR3-READ	The readership shall be Process Owners, Process Operators and Work Product Consumers. [0.2]
PPR4-METRICS	The metrics to be used to measure the procedure's operation shall be enumerated together with the means of gathering their values. *For non-production processes this rule may be waived at the discretion of the process owner if it's certain the performance of the process will not require measuring.* [0.3]

5.3 Procedure

This is the procedure which is used to define a process.

PROCEDURE: Process Definition Process

Version [0.4]; **Date** 6 November 2002; **Editor** DH; **Tag** PROC.PDP; **Rules** RULES.G[0.2], PROCEDURE.RULES [0.4]; **Pages** 1.0; **Readers** Process Owners, Operators and Work Product Consumers; **Status** Unexited > 60 majors/page;

PURPOSE	The Process Definition Process transforms sources of all kinds, written and unwritten, into a set of artefacts for the Process being defined: i) the minimum set of artefacts for the Process is: Process Entry Criteria; Process Procedure; Process Exit Criteria ii) the minimum set of artefacts for each Work Product of the Process is: Work Product Specific Rules iii) optional artefacts for each Work Product of the Process are: Work Product Checklist, Work Product Entry Criteria and Work Product Exit Criteria. *Examples of methods to achieve this include brainstorming, on-the-job knowledge elicitation, studying extant documents etc.* [0.2]
STEP0	Determine what the Purpose of the Process is. The Purpose is written into the Process Procedure. *Answer the question: "What is the Process there to do?" in the form "to transform X into Y". X then defines the Sources and Y the Work Product.* [0.4]
STEP1	Determine which Source Documents must exist before the Process can be started. These are written as the Process Entry Criteria. *The Sources have been identified by STEP0. This step can be carried out simultaneously with STEP2.*

STEP2	Determine which Work Product document the Process is to create, and any other conditions which must exist before the Process can be considered to be complete. These are written as the Process Exit Criteria. *The Work Product has been identified in STEP0. This step can be carried out simultaneously with STEP1.*
STEP3	Establish the steps, and the order in which they must be carried out, needed to transform the Source Documents {←STEP1} into the Work Product document of the Process {←STEP2}. These are written into the Process Procedure. [0.2]
STEP4	Write the Specific Document Rules for the Work Product of the Process. *Rules shall comply with the Ruleset for Rules (RULES.R) and contain product-specific extensions to the Generic Rules (RULES.G).* [0.4]
STEP5	Decide what metrics will be used to measure process performance. Metrics must be practical and cost-effective to collect and must measure the effects of any process changes. Each metric and the way it will be actually measured are written into the Process Procedure. [0.3]