

# Why Good Process Is Not Good Enough

Last Updated: 16-August-2003

Joe Befumo,  
joe@befumo.com  
Shakti Development, Inc.

## Abstract

Software Development Process is gaining wider acceptance among I.T. organizations, but it should not be viewed as the next Silver Bullet. Process is just one piece in the software methodology puzzle.

## Overview

Software development processes have been around for some time now, and are beginning to gain wider acceptance. Although this increased focus on how we build (or, more often than not, fail to build) software products is a healthy trend, realistic expectations must be emphasized. In particular, it is essential to recognize that software development is a complex undertaking, whose elements are highly synergistic. A defined process is one element of the mechanism, and will not yield optimum results without a holistic understanding of the entire system. Moreover, process is neither a "purchase and deploy" commodity, nor is it a quick fix for a project on the brink of the abyss. If you're serious about bringing order to a dithyrambic I.T. organization, you must be prepared to pay for it, work at it, and fight for it, but if you stick to your guns, victory will be yours.

## Process And Methodology

Although the terms are often confused or used interchangeably, for the purpose of clarity I define the overall collection of practices, standards, tools, techniques, workflows and metrics that apply to the business of software development as **METHODOLOGY**. Within this blanket term we find processes, macro-methodologies, micro-methodologies, and a whole gamut of support components. These will be described in greater detail later (see: How the pieces fit together). For now, the salient point is to recognize that what most people think of as 'process' is one element of a larger whole. It makes as little sense to talk about process in isolation as it does to discuss a CASE tool without considering the design system it automates (1).

(1) CASE tools are a prime example of a good idea unfairly maligned by unrealistic expectations (and unscrupulous vendors).

When first introduced, many managers were led to believe that a CASE tool would instantly automate their entire development lifecycle, eliminate the need for coding, improve productivity and quality, and do all of this without training, experience, or effort. As a result, CASE tools received an evil reputation that they only recently have started to transcend.

Methodologies, processes, standards and procedures are certainly not new ideas. Progressive organizations have long realized that in order to turn transient knowledge into a corporate asset it must be captured, organized, recorded, and dispensed to future generations of practitioners. Historically, this took the form of massive paper tomes. Far too unwieldy for regular use, these were either ignored, or became the focus of corporate police actions. Typically, practitioners spent more time circumventing enforcement than deriving benefit. This suggests a critical success factor: *Methodology content must be readily available, easy to use, and so valuable that developers will revolt if you take it away.*

In order to understand the essential elements of a methodology that will be *useful, usable, and used*, we have to examine who will use it, how they will use it, and what it is. Before we do that, however, let's take a few moments to anticipate and dispel the inevitable arguments we'll encounter in our crusade to enlighten the infidels of development chaos.

### Argument #1: My Needs are Unique in all the Universe

This mindset arises from the conviction that *this* organization's needs are *so* unique that no off-the-shelf methodology could ever hope to address its inherent complexities. The source of this belief is an emotional need to justify past failures by demonstrating unprecedented adversity. Thus, you'll find this orientation most prevalent in organizations with the most impressive track records of botched projects. Since purchasing content is out of the question, and because these organizations are inevitably behind schedule already, their answer is to continue without any process at all rather than condescend to the use of anything less than perfection.

### Argument #2: The Grand Unified Methodology

At some juncture, usually after a particularly disastrous debacle, management decides to bite the bullet and put a process in place. Based on attitude #1, there's no question but that the process will have to be developed in house. Moreover, this will be the process to end all processes - the mother of all methodologies - the grand unified theory of software development which will elevate all involved to legend status and ensure their place in the annals of computer history. In other words, they approach the task the same way they develop software systems. The result is predictable.

### Argument #3: The Black-Belt Syndrome

Martial artists will recognize the "how long to get my black belt?" syndrome: A teenage boy walks into the dojo, ticket stub from the latest Chuck Norris movie fresh in his pocket. He just wants to know how long it will take to get his black belt. The sensei (teacher) patiently explains that training is a path, not a destination; a way of life, not a colored belt; but the would-be grand master only wants to know if he can cut the time in half by attending twice as often. The software equivalent usually involves SEI/CMM, ISO-9000, the Malcolm Baldrige or some similar badge of merit. Like our young Karate Kid, these would-be software black-belts quickly become discouraged and move on to the next whim.

### Argument #4: The Paperwork Nightmare

This attitude usually follows an unsuccessful "black belt" episode (see #3). When I suggest to management that they might consider implementing some minimal process-improvement measures, jaws clench, eyes narrow, and they hiss something to the effect of: "We tried that ISO-9000 thing, and all we got was an administrative nightmare!"

I wish I could suggest some incontrovertible chain of apodictic reasoning that would overcome these arguments, but the best I can do is the following:

- The "ultimate" methodology does not, and probably never will exist
- An effective process is the one that is useful, usable and used
- Flexibility is the key to a real-world methodology
- Good-enough now is better than perfect later - a good enough methodology is one that helps you:
  - ✓ Do what has to be done,
  - ✓ When it has to be done,
  - ✓ As well as it needs to be done,
  - ✓ And do it that way every time.

### Who Uses it, and How?

Even experienced professionals who have worked with formal processes before are often surprised at the number of elements that comprise a comprehensive process management and methodology deployment system. There are several reasons for this:

- A distressingly small percentage of the companies developing software systems follow any defined process at all.
- Of those that do, most have deployed only a few select components.
- Even among those rare organizations that have invested in the full scope of process and methodology tools and content, most are not using them consistently (2).
- Finally, the individual contributors who have been exposed to a formal process have usually experienced it from the limited perspective of their own project involvement.

Thus, in order to appreciate the full scope of this domain (3), we must first examine who participates in it and how they interact with the system. Figure 1 illustrates the main areas of activity in a methodology system.

(2) Several years ago I was being recruited by a large consulting firm, widely known for their commercially available methodologies and process management tool kit. The position they wanted me to fill amounted to "Methodology Enforcer." As they put it: "we have gigabytes of content and the best tools in the industry, but nobody here is using them on projects."

(3) I'm often questioned when I include "Software Development" in the list of subject area domains with which I am conversant. I'm really not surprised that this seems unusual, because people rarely treat software development like a business, or take the time to understand it from that perspective. The essence of methodology development is the analysis of the subject area of software development, and the definition of a system to meet its business needs.

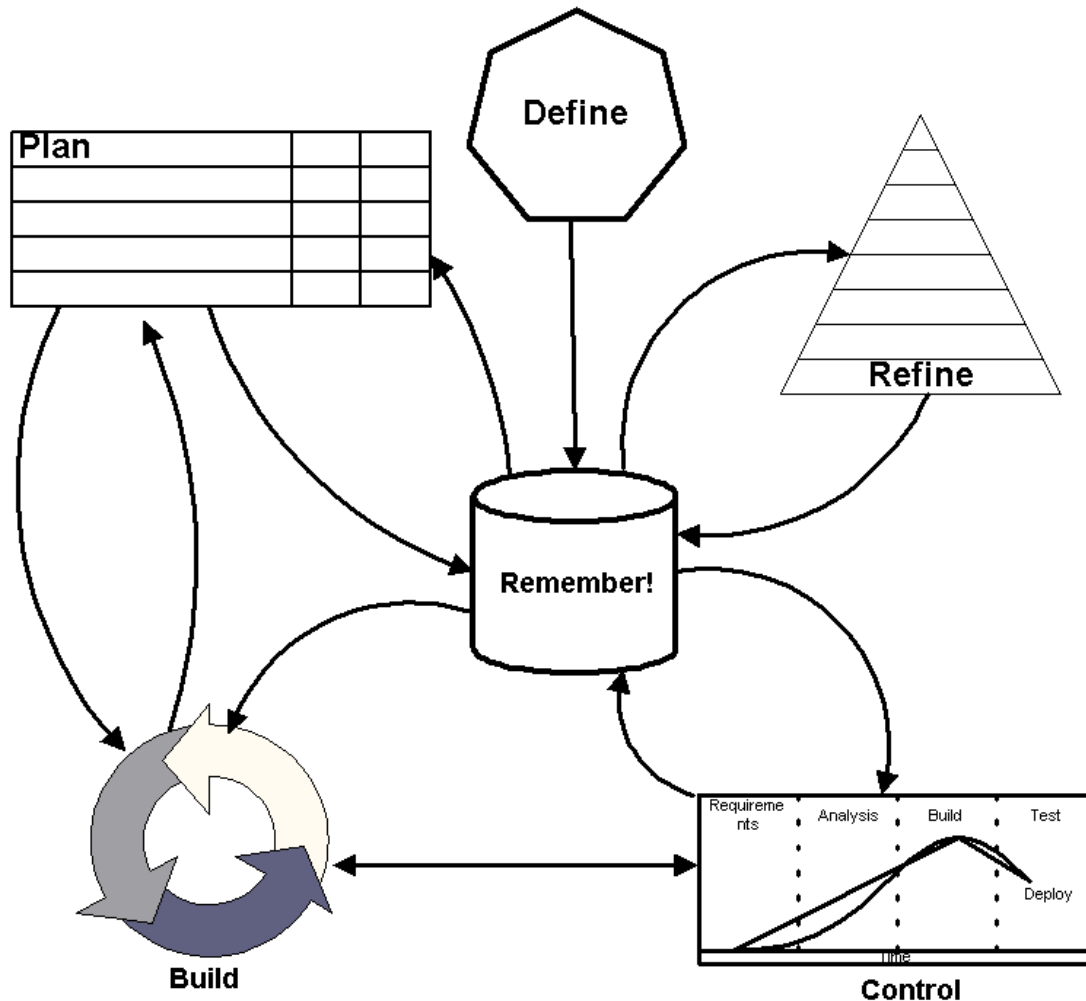


Figure 1-Main Usage Areas

## Methodologists and Process Authors

These are the people who ensure that the company's methodology remains current, reacts to the needs of those actually using it, and reflects lessons learned in the field. They interact with the system primarily in the areas marked *Define*, and *Refine* in Figure 1. Definition represents initial and subsequent external input into the system. Think of it as a best-guess first shot at how software development engagements should be handled. The most efficient way to jump-start the system is usually to buy a commercial methodology package, such as those offered by Rational, LBMS, SHL, and others. Despite their sometimes excessive cost (Rational's is the exception), few organizations have the resources to produce the volume or quality of content that these packages provide in a timely, cost-effective manner - in other words don't re-invent the wheel. Remember, this is just the starting point, and will undergo continuous modification and customization

regardless of where you start. This is a crucial point - if a methodology is static it will soon become obsolete.

## Project Managers

These are the lieutenants in the project battlefield. Whatever strategies the generals (methodologists) may have anticipated, chances are they will have to be modified in the thick of the fray. Project managers must be able to quickly identify parts of the process that are pertinent in a given situation and eliminate those that are superfluous. This process is facilitated by a streamlined tabular view of the major structures. Project managers interact with the system primarily in the areas marked *Planing*, and *Control* in Figure 1.

Effort estimates, risk assessment and tracking, progress updates, and defect control are among the areas of keen interest to the project manager. The feature most frequently requested by project managers is the ability to quickly choose a set of phases, stages, activities, tasks, etc., and automatically export it into their favorite project management tool. The better process tools will check consistency, create precedence relationships, and generate preliminary estimates using a variety of metrics.

At the beginning of a project, managers will go through many iterations until they come up with a project plan that is acceptable to all concerned. Moreover, *good* project managers constantly rethink and revise their project plans throughout the life of the project. Therefore, this capability should be considered a critical success factor.

## Sales and Marketing

Early marketing calls often set expectations that will affect (and often plague) the entire course of the project. ("You promised them what?") It is therefore essential for sales and marketing representatives to tap into the same guidelines that will be used to run the project. Specifically, high-level metrics must support preliminary sizing estimates, generic deliverables must be identified, change control procedures and sign-off standards explained, and so forth. Moreover, this information must be available in form that is comprehensible and unambiguous to a non-technical audience. This usually draws from elements of the *Plan* and *Build* areas in Figure 1.

## Technical Contributors

Once the project is underway, developers must refer to the process frequently in order to check on their deliverables, confirm standards, access templates, and understand where they are in the overall scheme of things. This area of interaction is shown as the *Build* section in Figure 1. Team members receive active guidance to ensure that tasks are completed in a manner consistent with established standards. In addition, the system may also capture status information and communicate this back to the project manager. Critical communications are also recorded and tracked without incurring additional burden on the team members. Developers should also be able to drill down to whatever

depth they require in order to get technical guidance when faced with unfamiliar tasks - that is, the methodology infrastructure must support Just-In-Time Training.

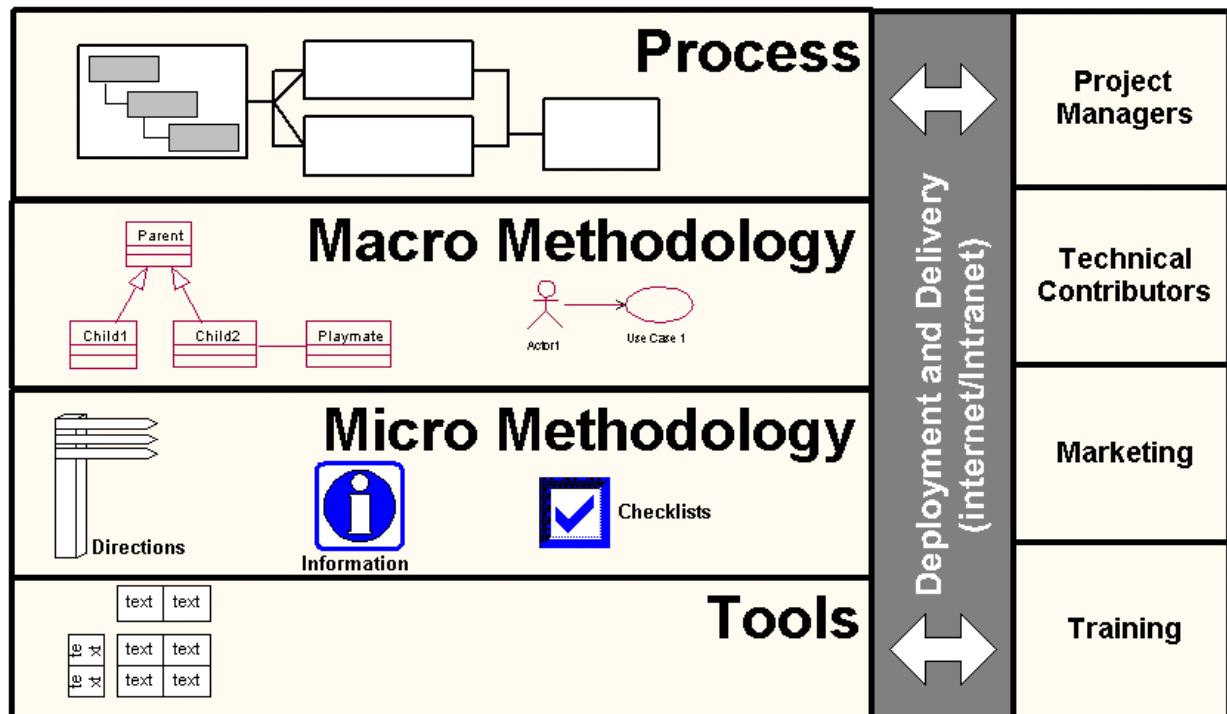
In order to ensure a consistent approach across the enterprise, the deployment mechanism must address the needs of each of these groups. In the next section we'll take a look at the parts of a complete methodology and see how they address these needs. Finally, we'll examine the deployment backbone itself, in order to better understand what services are needed to pull all the pieces together.

## The Elements and Their Relationships

RUP, Rational's Unified Process (discussed in grater detail below) describes process as follows:

*"A process is a set of partially ordered steps intended to reach a goal; in software engineering the goal is to build a software product, or to enhance an existing one; in process engineering, the goal is to develop or enhance a process."*

This is an excellent analysis because it recognizes the sometimes slippery notion that we are engaged in two distinct, but related, activities. Unfortunately, RUP neglects the crucial point that, in both cases, process is only one of several interrelated elements. These are shown in *Figure 2*.



*Figure 2*

As *Figure 2* illustrates, each component builds on activities that occur below it in the model.

## Process

We must continually re-emphasized that there is no single "master process" that addresses the needs of every project and every engagement. Rather, a real-world process should suggest a number of "typical" frameworks (4), allowing project leaders to mix and match components, as necessary, to come up with a route map that reflects the nuances of the individual project.

(4) Some representative frameworks might address the typical needs of Package Evaluation, Internal Utility Development, Client/Server System Development, Business Process Reengineering, ActiveX Control Development, Web Applications, Bid Preparation, and so forth.

This suggests that we should not focus on producing a single process to which every practitioner is compelled to conform. Rather, we seek to ensure consistency in approach and outcome through a combination of:

- Standardized process and macro/micro-methodology components.
- Well-defined guidelines that ensure that a minimal selection of elements are combined in legitimate sequences.
- User aids such as 'Wizards' to assist the user in selecting from multiple possible approaches to (for example) Concept Definition, Requirements Gathering, or Analysis

## Macro Methodology

The Macro Methodology components detail the procedures of each major activity outlined in the Process. For instance, Macro Methodology would define the steps a practitioner must go through when creating a logical database design, a physical database design, or an object model. Thus, UML (Unified Modeling Language) represents one possible Macro Methodology corresponding to several analysis and design activities. If your customer demands that OMT or Booch be used, it should be easy to retain the same process and replace appropriate segments of the Macro Methodology. Similarly, if a particular engagement requires Yourdon/DeMarco, or Gane & Sarson, the corresponding process blocks for analysis and design would be swapped out and replaced with procedural components (5).

(5) Yes, I know this is sacrilegious, but some people *did* manage to build successful software before OO. And, while

we're on this subject, I would also argue that groups who applied the procedural disciplines thoughtfully and consistently inevitably achieved better results than those who apply OO in a sloppy or haphazard manner.

## Micro Methodology

Micro Methodology fills in the remaining details left out of the Macro Methodology. Examples include the proper use of the OOA/OOD and database design notations, application of a testing methodology, project management, and steps to follow in network design. The Micro Methodology supports the Macro Methodology by providing detailed instructions, templates, examples, and so forth. Also included in this partition are policies and procedures addressing things like naming conventions, configuration control, and testing standards.

## Tools

All tools supporting the Micro Methodology component (e.g., CASE, project management, estimation, etc.) should be outlined, and their use fully documented. In addition, a full-lifecycle project backbone will provide a means for integrating the tools and their output into the methodology. For example, when the project manager assigns an object-modeling task to a team member, the methodology will typically stipulate the tool to be used for this purpose, the naming convention for the workproducts, and a directory structure in the configuration management tool where they are to be kept. In some of the more ambitious deployment tools (e.g. LBMS Process Engineer(tm)) this assignment shows up in the team member's "to-do" list, pre-associated with the appropriate tool. The user has only to click on the work item to launch the tool in the defined context. While some may dismiss this level of automation as "fluff," there are good reasons for favoring such control. I've worked in environments where one group unilaterally decided to use FrameMaker(tm) for their documentation, but nobody else in the organization had access to the tools necessary to read the resulting documents. Similarly, if the company has invested in a full-featured CASE tool such as Rational Rose(tm), they probably don't want team members using their favorite flow-charter just because it's what they already know.

## Content Delivery and Deployment

So far, we have discussed *what* has to happen in a full-lifecycle process management system, and *why* these activities are important. We can now complete the picture by taking a look at *how* these goals can be accomplished.

## The Role of the Internet

Early attempts to create a full-lifecycle process delivery platform were hampered by technological, and, oddly enough, process limitations. Though well-intentioned, these

were all essentially LAN-based tools, adequate for small workgroups, but not capable of addressing the needs of a large distributed development team (6). The internet provides an obvious solution, but care must be taken if it is to be used in the most efficacious manner. In particular, simply publishing your favorite methodology in HTML format may look pretty, but fails to add much value beyond navigational ease when compared with paper publications.

(6) Even if you're not currently involved in large, geographically distributed projects, the same infrastructure will allow you attract developers by supporting telecommuting, tap into talent that may not be readily available in your area, and maintain control of your projects from wherever you happen to be. These are all features that will become increasingly important as the technical talent shortage becomes more acute.

In order to deliver significant benefit, methodology and process information must be modeled in a more condensed and abstract manner, and then extracted, via the internet, to the various formats that serve the needs of the identified user classes. Moreover, the system must be designed to facilitate a two-way dialog in order to foster process feedback and continuous improvement.

The ideal integrated, full lifecycle, round-trip, process management backbone will provide the following minimal services:

- *An open repository for maintaining methodology meta-data* - It is important that this information exist at an abstract level (e.g. in a relational or object database), in order to facilitate flexible access in a variety of formats.
- *A suite of tools for authoring and maintaining methodology content* - Within certain limits, both the knowledge schema and the content detail must be definable by the methodologist. What this means is that each organization should be able to describe the basic objects that best model their process (phases, activities, deliverables, precedence, patterns, and so forth), as well as the specific instantiations of each (Analysis Phase, Large Project Startup Pattern, Peer Review, activity, Status Report Deliverable, etc.)
- *Intrinsic internet capability* - Remote access to the methodology content should be built into the system, and not require publication of a static 'snapshot' of the process at a particular point in time.
- *Active feedback capabilities* - The system should be capable of gathering data about the way it's actually being used in order to model the way people really work.

- *Integrated estimating facilities* - multiple approaches should be provided. Examples include object/artifact-based, top-down apportionment, rule-based, function points, etc.,
- *Round trip integration with project management/scheduling tools*
- *Flexible customization facilities* - the system must recognize that the front-line project manager has final responsibility for the execution of the project, and must therefore have the ability to alter the "roadmap" to match the needs of the engagement. Consistency checks should advise, but not enforce. (Typical consistency checks would ensure that every work product (output) is used as to input to another process, that every input is produced before it is used, and that precedence relationships are respected.)
- *Tight integration with configuration/source management tools*
- *A Central communication facility* - Critical communications, issue tracking, agreements, and so forth should be maintained within the system, and must be easily traceable.
- *Online reference/JIT Training* - Considerable evidence exists to suggest that training is most effective when delivered soon before it is to be applied. By combining online training with process guidance, practitioners can receive training/knowledge-refreshment on demand, precisely when it is required.
- *'Live' access to corporate experts (possibly at remote sites) via integrated IRC (internet relay chat) facilities.*

Just a few years ago, this list would have been a distant dream. Today, every component is readily available. Although nobody has yet combined these functions into a single package, it's only a matter of time. As it is, several offerings come tantalizingly close. These are discussed in the next section.

## What's Available Today?

Until recently, few people even knew about process management or methodology tools. Thus, there was little incentive for widespread development. This situation is certain to change. Our dismal failure to deliver reliable software with anything approaching regularity is a national disgrace. It is no longer possible to ignore or deny the fact that those organizations that *do* succeed invariably employ a robust methodology and a thoughtful approach to process management. Several forward-thinking companies have been providing process management tools for several years now, and we can expect the number and quality of offerings to increase to meet demand. Currently available products include:

## LBMS Process Engineer(r)

LBMS (acquired some years ago by Platinum Technology, which was subsequently absorbed by Computer Associates) was one of the earliest pioneers of Process Management technology, with their LBMS Process Engineer suite. Process Engineer consisted of:

- \* PE/Process Library(tm) - A repository of flexible development processes, each built of reusable process components called 'kernels'.

- \* PE/Process Manager(tm) - A comprehensive methodology authoring and maintenance environment.

- \* PE/Project Manager(tm) - The project management component, through which the user could mix and match components from the various process templates, check for consistency, perform estimates, export to and import from Microsoft Project(tm), assign work, and communicate with team members. This tool also allows the modified template to be reintroduced into the Process Library, thereby capturing actual project information. In addition, it could publish detailed project roadmaps in both paper-based and hypergraphic form. The latest incarnation can create dynamic web sites directly from the methodology database.

- \* PE/Activity Manager(tm) - This is the team member's workbench. Using the Activity Manager component, developers receive work assignments, confirm their understanding, correct estimates, launch the assigned tools, and capture their work metrics. This, too, is web-based in the newest versions.

- \* Processware Program(tm) - In addition to the methodology content contained in the PE/Process Library, LBMS offers a variety of methodologies from outside vendors. Several well known processes have been migrated to the LBMS platform.

Process Engineer's main strength is the sheer scope and capability of its toolkit, covering most of the key areas discussed earlier. For its day, it was an incredibly ambitious and powerful tool, and it remains such today, having been resurrected into the web world by Computer Associates..

The Process Engineer's content has also kept pace with modern technology, offering various project templates for a variety of web technologies..

By far, Process Engineer remains the best of breed for process maintenance tools.

## SHL Transform(tm)

(As of August,1998, Viasoft has Acquired Exclusive Rights to the SHL TRANSFORM Product Line). SHL Systemhouse was the other pioneer at the process tool frontier. SHL

Systemhouse was a professional services consultancy, and Transform originated as their internal tool for managing their engagement methodologies. TRANSFORM's strength is its sheer volume of high-quality content (around 2.5 gigabyte at last count).

TRANSFORM includes multimedia training courses, a complete ISO-9000 compliant procedure manual, the full SEI/CMM documentation set, checklists, templates, and so forth.

On the downside, their toolkit started life as an internal utility, and still looks it. What you get is a rather disjointed hodgepodge of 16-bit Visual Basic tools that more or less allow you to write, circulate, and access methodology content. Installation is a nightmare of directory creation, command-line copy operations, editing .INI files, running database scripts . . . you get the idea. Like LBMS, users can extend and enhance the content, but the authoring process is far less intuitive, and requires a significant learning curve. Here again, internet support is an afterthought, and takes the form of a published snapshot. None of the tools actually work through the web interface, which can be used for browsing content only.

#### Client/Server Connection CS/10000(tm)

CS/10000 from Client Server Connection, LTD. is an interesting little tool, aimed primarily at organizations who have never built a client/server system before. CS/12000 utilizes a 'Wizard' approach to walk the user through the process of characterizing the system they plan to build. It then generates a process to be followed in developing that system. Clearly, this is a far less ambitious product than the previous two, and to be fair, what it tries to do it actually does quite well. Nonetheless, its narrow scope and shallow depth take it out of the category of enterprise-level process tools.

#### Rational Unified Process (RUP)(tm)

Rational Software - purveyor of the popular Rational Rose CASE tool and home of the Three Amigos (Booch, Rumbaugh, and Jacobson) - is a relative newcomer to the process/methodology arena. Based solely on their leadership position in the Object-Oriented Tool domain and considerable influence on UML (Universal Modeling Language), RUP will certainly be a major player.

RUP has a lot going for it. First and foremost, it's a good methodology. Unlike some of the others discussed here, RUP does not bring with it a lot of ancient baggage. RUP is new process, based on the latest Object-Oriented thinking from some of the best minds in the business. Five minutes with this product is enough to tell you that it was written by people who have worked on real-world projects - big ones! RUP also goes deep, with plenty of samples, templates, and so forth. Moreover, RUP costs about 1/100 the price of SHL's or LBMS's content!

RUP's major weakness, compared to LBMS Process Engineer and SHL TRANSFORM is that it is one single approach to project development, and is not part of an integrated tool suite. Although I have no doubt that an experienced organization that follows RUP's

process will develop software in a reliable and predictable manner, the truth of the matter is that such an organization would probably do a good job anyway. Initially, while RUP acknowledged the necessity of modifying, customizing, and fine-tuning the process, it did not include the means of doing so. In its most recent release (as of 7/2004), it does offer some customization capabilities through a component called "RUP Builder." This, unfortunately, is an unwieldy kludge that was clearly an afterthought. For example, here are the steps I currently go through to model our methodology in RUP:

With the trend toward major CASE tool vendors moving into the full-lifecycle process management arena (See LBMS and Platinum, above), it will be interesting to see if Rational follows suite. With their already extensive offering of robust enterprise-class tools, their deep commitment to UML and overall understanding of the engineering processes, a Rational process platform could quickly become a major player in this growing market.

As we discussed earlier, there can never be one master process that is all things to all projects. A process that contains information on every possible subject is simply too unwieldy to be used, for example, to plan a simple package evaluation project. Similarly, without the ability to easily integrate special tasks, and do so in a seamless manner, the process can not grow to meet unanticipated situations. Most methodologists acknowledge that any "store-bought" process is, at best, a starting point, and to become truly useful, it must grow to reflect the organization's unique combination of talent, approach, domain, and so forth. For this to happen, the methodology must include the tools and infrastructure support; process alone just isn't enough, which brings us full circle back to the premise of this paper.

## Conclusions

Our lamentable inability to build reliable software that meets our end-user's expectations, in a predictable, cost-effective manner can only be improved through application of a consistent, dependable set of processes, tools, standards and procedures. The inherent complexity of our current undertakings demands the support of industrial strength tools. Unfortunately, the process tools now available are immature and do not provide the functionality required for today's complex software projects.

Nevertheless, the processes, tools, and techniques that are available represent a significant improvement over the way most organizations currently approach software development projects. Thus, organizations must accept the reality that a good-enough process today is better than the "perfect" process sometime in the future. In reality, the ideal process is the one that best fits into the organization's culture, and therefore, the one that will be used.

The key to successfully introducing and maintaining a development process is the flexibility of both the methodology and the supporting infrastructure. Practitioners must be able to mould the process to their needs, and, the process must be able to grow with the organization. The most critical factor for supporting these goals is the

delivery/deployment infrastructure. Although no current tools offer all of the critical services necessary to support and maintain an enterprise-scale methodology environment, increasing interest is certain to drive rapid improvements. In the meantime, there's no excuse for not using what's already there.

## Appendix: Process and Methodology Resources

SHL TRANSFORM: <http://www.viasoft.com/>

Client Server Connection, LTD: <http://www.cscl.com/>

Platinum Technology/LBMS: <http://www.platinum.com/>

Rational Unified Process: <http://www.rational.com/products/rup/>

## Bibliography

Ambler, Scott W. 1998, Process Patterns, Building Large-Scale Systems Using Object Technology, Cambridge, UK, Cambridge University Press.

Booch, G. 1996, Object Solutions, Managing the Object-Oriented Project, Addison Wesley Publishing Company, Inc.

Stone, John A. 1996, Developing Software Applications in a Changing IT Environment, NY, McGraw-Hill

## Biographical Sketch

Joe Befumo is a Senior Managing Partner and Chief Technology Officer at Clarity Development, LLC. After working in software development since 1974, he became interested in process improvement issues in 1989, while at Digital Equipment Corporation's Advanced Semiconductor Development Group. As a methodologist and head of AT&T Solutions' Center for Estimating and Metrics he contributed to the development of software processes geared toward the large-scale distributed development projects. He has been Chief Technology Officer of Clarity Development since 1997.