Response to OMG
Analysis & Design PTF
Software Process Engineering

Request for Information

By

DMR Consulting Group Inc.
1200 McGill College Avenue, Suite 2300
Montréal, Québec
Canada
H3B 4G7

Contact : **Jean-Marc Proulx, VP Corporate Knowledge, Products and Services**
Email: **Jean-Marc_Proulx@dmr.ca**

March 1st, 1999

1. Introduction

This is the DMR Consulting Group's response to the OMG Software Process Engineering Request for Information (OMG document # ad/98-10-08). Our response is presented from the point of view of a company that stands out in the marketplace as being a leading methods-based consulting firm. We are especially known for our ability to deliver business and information technology solutions.

Over 50% of our revenues (currently one billion US dollars) are from systems delivery and maintenance services.

We consolidate our "best-of-breed" processes and solutions into an integrated suite of business and technology methods, software tools and state-of-the-art-techniques called the DMR Macroscope™.

General information about the DMR Consulting Group can be fount at http://www.dmr.com and general information about the DMR Macroscope™ can be found at http://www.dmr.com/macroscope.

2. General Questions Raised by the White Paper

2.1 Is this something the OMG should be doing?

"OMG was formed to create a component-based software marketplace by hastening the introduction of standardized object software." (OMG background information on *www.omg.org*)

Our experience (see section 5. Existing Implementations) confirms the direction set forth in section 1.3 of the White Paper asserting that "it is not possible to standardize a process or methodology for all software development situations." However, it is possible to create a solid model of the software development process and its work products and then, through late binding mechanisms, adapt it to many various situations.

Consequently, the way the OMG should aim at "building consensus and convergence in the industry around system development process/methodology in a distributed environment" should be more generic than the adoption of a standard process. As presented in the White Paper, standardization at the meta-model level seems much more appropriate to the state of the practice. A definition language for system development process would be suitable provided its semantics are "domain-typed" (by domain here we mean software development). It should be rich and specialized enough to be applied in **system development/delivery and maintenance** and not only in **software development**.

Since the overarching goal of the OMG is to foster user independence from proprietary technology, it should not limit process engineering to constructs that are foreign to users as it is often the case in software development. Therefore, the foreseen meta-model should take into account not only the developer's perspective (or role) but also the user's, the owner's (owner being defined as the one responsible for the implementation of the system within the organization) and the project manager's.

This is not an obvious endeavor. Every entity of such a meta-model could be the subject of much discussion (What is a system? What is a role? What is a deliverable? What is a technique?). One could conclude that such a definition language should be typed strongly enough to promote discipline and maturity, and to support plug-and-play process engineering. Others would advocate that it should be extensible enough to allow for flexibility and late binding. Where should we draw the line between rigor and flexibility? What tradeoffs should be embodied in the meta-model?

We believe that the answer lies in a two-step approach. The adoption of an explicit **reference framework** (step 1) should precede and be in support of the adoption of the meta-model and the definition language (step 2). The **reference framework** would scope the basic dimensions of a **system development/delivery and maintenance** process: Who? What? How? and When?

Who?	The perspectives or the roles
What?	The deliverables or the work products
How?	The techniques, the guidelines, the hints, etc…This is partially provided by UML with respect to modeling.
When?	The paths, the phases, the milestones, the gates, etc…

The answers to these questions come to life in the meta-model. Process engineering is then defined as being the early selection and continuous improvement of process components against these dimensions.

Yes, the OMG should progressively extend the exchange of software components upstream towards the exchange of software process components.

2.2 Should the focus of the facility be the whole software development process or should it be limited to some part of it such as analysis and design?

In a world where concurrent engineering and change management are the rules, few benefits will result if the process engineer considers only a narrow portion of the process.

Object-orientation, like many other initiatives in software development, has largely been driven from the later portion of the life cycle where interaction with the owner of the system is traditionally at its minimum. Hence the focus of this RFI on a **software development process** instead of a **system development/delivery process**. Our experience shows that software is only one type of all the deliverables (work products, artifacts) that have to be developed in a specific development/delivery or maintenance contract. The scope of the solution delivered in a contract should match the scope of the problem to be solved and for that reason it entails much more than software.

Similarly, most low-level (levels 1 and 2 of the SEI Software CMM) capability and maturity issues have an organizational flavor embodied in requirements management, subcontract management, quality assurance and project management. These issues go beyond software development proper and they also cover other perspectives (for roles other than the one of developer). Process engineering decisions (which are dependent on project circumstances) are more thorough if the process engineer considers the entire life cycle, including maintenance.

Although it is very clear in section 3 of the White Paper that this RFI is not about process engineering in general, it is unclear if the scope of process engineering is limited to software or not.

Process engineering (which is often described as tailoring) is not a one-off, early binding activity. It is a dynamic, continuous activity. It needs reinforcement along the entire life-cycle. It is similar to planning. An initial plan is necessary, but is far from being sufficient. An initial process is essential, but since project circumstances (project profile), and targeted system characteristics will likely change when the process is enacted, the initial process itself will always be recomposed.

As well, we should not disregard maintenance. Customers are investing considerably to integrate their legacy systems into their enterprise architecture. Costly maintenance is often the result of bad engineering decisions made earlier in the life cycle, which confirms the need to consider the whole development/delivery life cycle.

Moreover, today's customers more often than not require that any development process, although not tool free must be **explicitly tool independent or tool neutral**. Many experiences with CASE tools (connected to a repository or not) have shown poor improvement in development quality and productivity because the focus was on tools at the expense of the process. In our meta-model, a tool is a software product that automates a technique or a series of related techniques. Indeed we have embodied UML 1.1 in our delivery process as a set of modeling techniques. This allows freedom of choice among UML compliant tools, which is in line with the OMG's objective.

- **The span of the process to be engineered should cover the whole life cycle, including requirements engineering and maintenance.**
- **Interoperability requires that process independence from specific tools be made very clear.**

2.3 Should the facility specify work products, rules for producing the work products, and the relationships among the work products?

Our experience confirms that, due to the inherent invisibility and malleability of software coupled with constant change of the business environment, any development process that does not go the additional mile towards making itself more visible is doomed to low institutionalization. Activities or tasks have a propensity to be more volatile and less visible (or measurable) than the structure of the work products they should create. Consequently, being very prescriptive about the activities and not as much about the "deliverables" is of little use for project tracking and oversight. Although UML is very effective in answering the "how" question and structuring the answer to the "what" question, the developer is still puzzled when he tries to assess "How much effort to completion?" This translates into "When do I know that I am done? Is it when I have executed all the activities specified in the process or is it really when the architect or the owner has approved the work products that are the results of my activities"? Work products should be knotted explicitly with activities via process engineering and not implicitly through techniques (often called "methodologies" in the industry). Techniques describe how information elements, and therefore work products, are produced.

Our practical knowledge demonstrates that deliverables are composed from basic information elements from which progress can be tracked more easily and that there can be no process engineering without rigorously defined composite work products in conjunction with all other process elements.

The need for well-delineated work products becomes even more manifest when using concurrent engineering and iterative development. How would you ensure that a work product is completed in only one phase? How do you decide the right level of detail? And then how do you translate the proper levels of detail in the project-specific process?

We believe that this becomes possible only if the meta-model allows for a rich and rigorous structuring of the work products.

Yes, by using the SPDL, the process engineer should be able to specify a rich set of composite work products together with the required rules.

3. Proposed Clarifications to the White Paper Concepts

3.1 Is the Concept described in the White Paper correct?

" A facility (set of services) to build processes for specific projects by matching project profile "parameters" against the meta-model and selecting model components (possibly from different models.)"

In our opinion, the concept of **facility** stated in the White Paper may lead to a long-drawn-out endeavor. Past repository initiatives such as AD/Cycle and PCTE were very complex and did not succeed mainly on account of bad implementation decisions that lead to difficulties in ease-of-use and ease-of-learning. Although we believe that real-life process engineering is impossible without some sort of facility, we would like to stress that if implementations lack ease-of-use and ease-of-learning, the impact sought by the OMG will not materialize. Usability was certainly not on the agenda of the committee meetings that approved the aforementioned repository implementations.

Customers are asking for task support services that are articulated through a web-enabled EPSS (Electronic Performance Support System). Skill shortage in today's marketplace puts an additional value on decreasing time to proficiency for practitioners. We must rely more and more on just in time and just enough process fragments to execute. Is the overwhelming requirement of usability met at the expense of engineering rigor? From our experience, adoption (institutionalization) is more dependent on usability including learnability than rigor. Rigor is taken for granted while usability adds much perceived value. How should we weigh this requirement against the perceived complexity of the MOF?

The design of the facility must cover ease-of-use and ease-of-learning as early as possible seeing that both will make the short list of selection criteria when customers select implementations.

3.2 Is a meta-model sufficient to support "plug and play" software development process definition from components?

It is necessary but not sufficient. It is a necessary first step along the long road of process engineering technology adoption.

Late binding of software development process components requires that the process definition language address not only coherence and robustness issues but also learning issues. Development process composition is a management decision with technical

implications. Assembled process components must be coherent, easy to learn and conducive to the production of sustainable software.

In addition to the clear-cut definition of software engineering concepts, the SPDL must allow for efficient communication of roles and responsibilities, work product definitions (deliverable structures, states and evolution) and task sequences.

The usability of the resulting process should benefit from the effort put on assessing and eventually improving the meta-model. The ultimate goal should be ease of use and ease of learning of the defined process.

Plug and play of process components does not depend only on a well-defined meta-model. It also depends largely on implementation decisions. Since compliance of specific processes at the meta-model level does not ensure interoperability per se, the endeavor should also encompass non-technical issues that have an important bearing on the adoption process, especially ones related to work organization and performance support.

A well-defined meta-model is only a necessary structural condition for plug and play.

The model should support an **additive approach** to software engineering by starting with a baseline that applies to many project circumstances and system characteristics. The model should also propose criteria and guidelines to address greater complexity and scale while preserving the integrity of the baseline. The process engineers would then have the opportunity to select components for more complex projects.

Plug and play of process components requires a well-defined meta-model that supports an additive approach to customization.

3.3 Are the UML and MOF sufficient to define the Software Process engineering Meta-Model? Are extensions required?

The DMR Consulting Group has extensive experience in defining system development/delivery processes. The processes that we use to conduct our business and that we license to our customers are defined in a knowledge repository that we call The DMR Macroscope™. They are defined according to our own meta-model (see section 5).

Over the past 10 years, through our R&D program, we conducted many experiments and pilot projects based on abstract constructs like Conceptual Graphs (Sowa) or technology-influenced meta-modeling languages such as PCTE. We can conclude that the “ideal” basic information needed to define a SPDL should allow a practical, understandable, tool-neutral meta-language based on work organization constructs.

From our point of view, the MOF specifications (ad/97-10-02 and ad/97-10-03) are oriented towards only a technical perspective. We recommend that the complex vocabulary of MOF be raised to a level that allows simple and efficient definitions for roles, activities, modeling techniques, etc.

Moreover, we believe that although UML is sufficient to define the structural dimension of a system, its expressive power applies better to the system's structure than the system's dynamics. Although we recommend and use UML to model information systems, we believe that business processes (and system development/delivery and maintenance are of this type) are better represented at a user level by a technique that we have developed and used extensively. This easy-to-learn technique is analogous to UML activity diagrams with object flow relationships (see the OPAL technique in section 5). It represents not only the structural dimension of a process but also its dynamic dimension. Our experience shows that this contributed significantly to the usability and learnability of the resulting processes.

We recommend that the DPM technique of OPAL be used to extend UML in order to better represent the dynamic dimension.

- **The OMG should raise the MOF representation to an abstraction level understandable by the users.**
- **UML should be extended to better represent processes.**

4. Future Needs

The OMG should go one step further and consider the impact of communication and distribution technologies. New authoring and publishing environments will have a major impact on the way processes are enacted. There is a need to consider how XML can be used to foster the adoption of sound process engineering.

Knowledge management is also a burgeoning field that can impact process engineering particularly in the capture, management and dissemination of best practices and related process components.

5. Existing Implementations

5.1 Processes that DMR develops, uses and licenses

We have made significant investments to ensure that we leverage the intellectual capital accumulated in over 25 years of consulting assignments, as well as in a multi-year, multi-client applied R&D program.

We have consolidated our "best-of-breed" processes and solutions into an integrated suite of business and technology methods, software tools and state-of-the-art techniques known as the DMR Macroscope™. These are continuously improved based on experience with clients, and supported by a complete set of learning programs.

DMR Macroscope establishes a comprehensive work environment aimed at supporting key functions such as strategic IT planning, business transformation, IT architecture, systems development/delivery, deployment and maintenance, knowledge transfer, benefits realization and change management.
DMR Macroscope methods provide a structured approach to change that, when implemented, renders a tangible set of services to the organization (see the table below). Methods identify the changes required, the manner in which the changes can be made, and the monitoring of the results. The methods make decisions visible and explicit by associating them with concrete deliverables and thus facilitate the accomplishment of the required change.

Over the past 15 years, we have released continuously improved versions of our methods. In version 3.1 (the current commercial version) DMR Macroscope has six methods.

Method	**Services Provided to the Organization**
Benefits Realization	• Enhances the realization of benefits associated with business and IT investments. • Helps organizations evaluate and manage their investments consistently.
Strategy	• Helps organizations resolve business and IT strategic issues. • Defines business and IT-enabling visions, strategic targets, and courses of action. • Steers integrated business and IT strategies.
Enterprise Architecture	• Designs or redesigns business processes, management controls, organizational structures, physical resources, products and services, relationships with external agents, and the human factor to achieve performance improvements. • Provides the design and plans to structure the delivery of IT components.
Technology Infrastructure Architecture	• Analyzes, designs, and implements the optimum technological infrastructure for system delivery projects. • Monitors the evolution of the technological infrastructure in order to maximize benefits by ensuring elements such as flexibility towards future business changes, new technology trends, and efficiency.
Delivery of Information Systems	• Designs and implements efficient information systems on time and within budget, in close cooperation with system users, while ensuring a systematic knowledge transfer. • Helps organizations select the right IT solutions, whether custom-built or assembled from commercial, off-the-shelf components.
Maintenance of Information Systems	• Supports, maintains and enhances information systems to preserve the quality, integrity and availability of all software assets.

The last two methods on this table jointly bear the commercial name of DMR ProductivityCentre™ and form the basis of our software development capability.

According to industry analysts, DMR Macroscope is the most extensive set of integrated methods, techniques and tools in the marketplace today.

DMR Macroscope—through its EPSS tools and techniques—provides professionals with the information, advice, and learning experiences to be more productive more rapidly, with minimum support from other people. It puts in place an electronic infrastructure to capture, store, and distribute knowledge about processes throughout the organization, enabling its customers to learn and apply knowledge faster than their competitors.

The path we followed in developing the DMR Macroscope was one of learning. The lessons learned required us to create models, concepts, approaches, techniques, frameworks, principles and processes which are central to any Systems Engineering Process and to Process Engineering. We have incorporated these lessons into today's DMR Macroscope. Some of them are highlighted below.

5.2 What DMR has developed in the area of process engineering

In order to manage the continuous improvement of the DMR Macroscope, DMR has created a series of process engineering components. It includes:

The OPAL (Objects and Processes Aligned) modeling technique.

- The purpose of the OPAL modeling technique is to produce a model that graphically represents processes and the relationships required between the architecture components to support their execution.
- The OPAL modeling technique is used to create models representing processes of organizations. The models produced with this technique describe processes and objects which participate in the processes. The model is presented as two specialized dimensions which show the process components and the different types of relationships. These two dimensions are known as the Dynamic Process Model (DPM) and the Object Relationship Model (ORM).
- The OPAL simulation technique allows the OPAL model dynamics to be defined and "executed". The results of this execution of the model can then be analyzed to yield insights as to how the process can be changed to improve performance.
- The OPAL modeling technique is supported by a tool integrated into the DMR Macroscope Workplace™ called DMR ArchitectureLab modeler. The DMR ArchitectureLab simulator, also integrated into DMR Macroscope Workplace, supports the simulation or OPAL models. These are PC-based (and compatible) automated tools that facilitate development and on-going management of the model. They facilitate the analysis of the possible impact of changes to systems through simulation of the architecture model.

A meta-model to represent a method/process

- The model contains all concepts related to the definition of a development/delivery process
- These concepts can be specialized to a specific situation
- Relationships between concepts can also be specialized to a specific situation

A repository to store knowledge objects associated to a method

- The meta-model has been implemented in a SQL database
- Knowledge objects are instances of the concepts (i.e. entities) of the meta-model
- Knowledge objects can be text or graphical elements
- The design allows for multiple natural language definitions of the knowledge objects

A toolkit to capture knowledge objects

- Functions have been devised for capturing knowledge objects using widespread tools such as MS-Word or MS-Access
- Functions have been developed to ensure referential integrity
- Internal hyperlinks for specific concepts can automatically be generated (e.g. to build a glossary)

Application modules to package process description and derivatives in multimedia format

- Functions developed to automate the production of documentation from the repository
- Documentation can be produced as documents formatted for print, or an integrated set of HTML pages suited for an Internet browser, or templates for the production of deliverables

A task support environment

- A browsing environment (DMR Macroscope Workplace) which allows to navigate in the software process documentation and to add personal annotations to the method components.
- A tool that allows creating the initial documentation framework of a project (Deliverable Assistant) using reference information derived from the repository.
- A tool that allows to create the project deliverable templates (Deliverable Wizard). The deliverable templates created contain suggested contents and have a contextual link to the online methods available in the browsing environment.

5.3 About our Meta–model

5.3.1 The meta-model is key in ensuring consistency and is the architecture's foundation.

The DMR Macroscope methods are stored in a repository after having been validated against the meta-model. This repository supports the need for continuous improvement of our methods.

The meta-model helps to enforce the many aspects of the architecture of a method: methodological, performance support, learning and transition.

DMR Macroscope methods define tasks to be performed and aim at providing full performance support, tools and learning events. Performance support is sometimes described as requiring four types of material: reference material (definitions), advice (nuances, hints, warnings, etc.), assistance (active help and coaching), and reuse (existing material to be used as a basis).

A method is a cohesive set of principles, concepts, activities and techniques employed to provide deliverables/products and services. All these categories can include prescriptive information (the method proper), guidelines and other types of performance support, as well as other forms of learning material.

Describing a method involves:

- Identifying the required service and setting the expected service level;
- Determining which deliverables or sections of deliverables (which we refer to as information elements) will be produced;
- Choosing a process that will produce deliverables according to principles for the service level chosen (right order, right level of detail, etc.);
- Determining which techniques will be used to produce the information elements in each deliverable;
- Providing sufficient advice to allow for the proper execution of the suggested activities;
- Providing learning support for the user to become competent in describing a method.

The meta-model and the architecture help decide what tradeoffs should be made between structure and dynamics.

DMR Macroscope methods follow their own precepts: their description clearly separates dynamics (paths and techniques to be followed, roles and responsibilities) from the required resources (deliverables and information elements, human actors, etc.)

The following considerations need to be balanced when designing methods, especially with an eye on their future evolution:

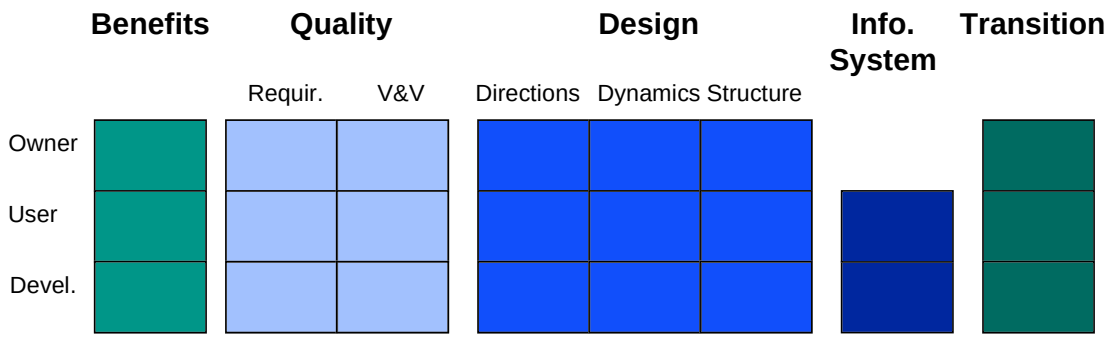
- Concepts represent the stable core of methods. Their stability ensures that different paths can be written to use the same information elements and the same techniques.
- Techniques and information elements cannot be designed separately. It is often the case that in order to reach a conclusion about a given key concept (say cost/benefits), different techniques require different information that corresponds to technique-specific concepts. It is extremely useful to consider several alternate techniques when designing information elements, so that appropriate generalizations are made. This tends to improve the robustness of the resulting set of information elements.

- Paths and deliverables cannot be designed separately. It is desirable to have similar deliverables in related paths. Such similarity cannot be achieved without looking at several paths at once. In fact, our experience so far has been that a single reference structure can often be built if a broad enough sample of paths is used.

5.4 Lessons learned

5.4.1 Ensure a Total Business Solution

DMR Macroscope methods deal with change in organizations, at various levels and from various points of view. The following framework summarizes key conceptual distinctions common to all DMR Macroscope methods. For example, all methods view the organization as a system, distinguish dynamics from the structural dimension, distinguish "as is" and "to be" states, have measurable criteria, distinguish business aspects from technical ones, etc. Methods can and do use concepts from all areas, as required by the services they provide. In addition, methods have their own specific frameworks (the technology infrastructure module of DMR ArchitectureLab suggests a very detailed model for developer-level design and requirements, for example).



A useful partition of concepts

The three center blocks (Quality, Design and Information System *aspects*) contain concepts that help describe an organization at a point in time (different methods do so for different purposes - strategic guidance, overall architecture or actual delivery). The two outer blocks (Benefits and Transition *aspects*) contain concepts that help plan and manage changes in the organization (i.e., the three center blocks) over time (benefits and organizational change management concepts, for instance).

We consider each of these aspects from the *viewpoint* of the owner, user and developer.

5.4.2 Balance Roles and Responsibilities

In DMR ProductivityCentre, the business need, the intrinsic quality of the solution, and the use of resources are three carefully balanced perspectives. For example a system that is correctly built and on budget, but does not meet the need will likely be shelved. Ignoring any one perspective causes similar unwanted effects.
In order to achieve the necessary balance, a strong decision-making and management role is given to the business system manager, which acts as the business representative on the project. The architect is given an essential voice in the planning and organization of the project. The project manager complements this team with strong leadership and organizational skills. This team approach to management is a critical success factor.

5.4.3 Risk management

- **Manage by Deliverables**

In DMR ProductivityCentre, the focus is on tangible results. Activities are organized around the production of visible results in the form of deliverables. Deliverables break down the work to be accomplished in small, manageable pieces. Project management focuses on the availability of those visible results to measure progress objectively. Deliverables provide a concrete way to manage efforts by tailoring content to the needs of the project, and managing deliverable scope and level of detail.

- **Enable Concurrent Engineering of All *Aspects***

DMR ProductivityCentre covers all aspect of information system delivery and maintenance, and goes much beyond strict software and database design to cover detailed work organization, business benefits, quality criteria, and transition issues. Furthermore, these issues must be considered simultaneously, as they often influence one another. This is in contrast with traditional approaches where a strict sequence is used, starting with benefits and ending with transition. DMR ProductivityCentre suggests instead that proper risk management is a matter of shifting emphasis: all issues are considered but with a weight appropriate for the stage of the project.

- **Enable Concurrent Engineering of All *Viewpoints***

Business, use, and technical issues need to be made explicit and the proper tradeoffs made. This requires that they be considered simultaneously, but with an emphasis appropriate to the stage of the project.
Again, this is in contrast with traditional approaches where the viewpoints are considered in sequence. By explicitly asking that viewpoints be considered simultaneously, DMR ProductivityCentre mitigates the risk that large sums of money be spent on a solution that makes business sense but causes major problems for people or is technologically unsound (or vice-versa).

- **Develop Levels of Detail Iteratively**

It is typically impossible to consider all elements of a problem in detail. It is therefore desirable to identify key cases that are examined in detail while the rest of the cases are examined in a more summary fashion. This allows for good risk management by focusing resources on high payoff elements.

- **Manage Business Risk Through Releases**

New information systems or changes to information systems are put into production through a disciplined process. A release is a business notion, and its content and timing are determined to maximize business benefits. A large change will typically be effected in several releases, to better match the organization's rate of evolution and to better utilize development and maintenance resources.

- **Manage Development Risk Through Phases**

Within any project, it is necessary to have clear decision points where the evolution of the project can be reassessed from a business standpoint. This also enables a "do less, but sooner" approach where initial outlines of the business solution are approved before going on to more detailed solution. In delivery projects, phases provide key milestones where business need, feasibility, overall design and individual release content can be assessed.

- **Use a Controlled, Iterative and Incremental Approach**

DMR ProductivityCentre advocates a progressive approach, which enables more continuous discussion of the needs and ensures that the project can react to changes in its environment. In delivery projects, this is made concrete by a disciplined approach to prototyping, with clearly measurable criteria.
Prototyping has two main benefits: it makes the results of decisions visible, thereby reducing misunderstandings and rework, and it provides team members with smaller, concrete steps to aim at. When prototyping cannot be used, a similar iterative approach (analyze, design, validate and evaluate) can be used with deliverables. The key idea is to validate often, and always with measurable objectives and criteria in mind.

5.4.4 Tailor the Approach to Project Circumstances and System Characteristics

No two projects take place under the same circumstances, and no two information systems have the same characteristics. While DMR ProductivityCentre provides a first degree of adaptation by providing several paths, the very essence of the method is to allow customization via deliverables.
Depending on the project and the solution being proposed, it may be possible to:

- add or remove content from a deliverable (additional topics may need to be covered, and some may be non applicable);

- increase or decrease the amount of detail required, compared to the guidelines provided in the paths;
- determine whether more or fewer cases need to be covered, compared to the suggested coverage;
- determine whether a number of deliverables can be summarized into one summary deliverable;
- determine whether a deliverable can be omitted outright, or postponed to a later phase.

All such decisions should be taken after due consideration and justified in the documentation plan. Excessive removal of information increases regarding the solution and excessive documentation requirements increase risks on the schedule or budget.

- **Assure the Quality of the Process**

Business system management addresses the requirement for constantly increasing the quality of the solutions being developed. Quality can be defined as the set of properties and characteristics of an information system that enable it to meet a client's stated or implied needs.
This quality can be assured by properly balancing the concerns of the various project participants. This balancing revolves around three project management perspectives:

Needs: defining needs, ensuring they are met, making the transition to the solution, realizing benefits, and controlling system operation and evolution.
Resources: planning, organizing, directing, and controlling the human and financial resources assigned to the project, and supervising schedules.
Solution: designing, constructing, and maintaining the solution.

- **Techniques as Reusable, Skill-Oriented Processes**

The framework above provides a useful criterion for defining techniques: any process that operates within a category is probably operating on a clear set of concepts with clearly defined skills. Such a process is a good candidate for definition as a technique, since this packages it in the appropriate way for use in several paths.
Some techniques, however, are intrinsically about making tradeoffs, establishing strategy or making a synthesis. House of quality, distribution, transition strategy are all examples. Such techniques often require special skills and pose challenges for learning and exposition because of their wide prerequisites. It is especially important to identify them.

- **Paths Coordinate Techniques**

Paths can be thus seen as filling the various categories with information, over time. The production of a deliverable results from the coordinated application of techniques to produce the requisite information. The sequence of deliverables therefore synchronizes the multiple concerns and different abstraction levels.

6. Standards

Since we recommend consideration of the entire life-cycle and of many different perspectives, process engineering standardization should be done in conjunction with other standards institutes such as the SEI , the PMI and the Workflow Management Coalition.